João Leite Paolo Torroni
Thomas Ågotnes Guido Boella
Leon van der Torre (Eds.)

LNAI 6814

Computational Logic in Multi-Agent Systems

12th International Workshop, CLIMA XII
Barcelona, Spain, July 2011
Proceedings

Lecture Notes in Artificial Intelligence 6814

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

Subseries of Lecture Notes in Computer Science

João Leite Paolo Torroni
Thomas Ågotnes Guido Boella
Leon van der Torre (Eds.)

Computational Logic in Multi-Agent Systems

12th International Workshop, CLIMA XII
Barcelona, Spain, July 17-18, 2011
Proceedings

Springer

Volume Editors

João Leite
Universidade Nova de Lisboa, Quinta da Torre, 2829-516 Caparica, Portugal
E-mail: jleite@di.fct.unl.pt

Paolo Torroni
Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
E-mail: paolo.torroni@unibo.it

Thomas Ågotnes
University of Bergen, P.O. Box 7802, 5020 Bergen, Norway
E-mail: thomas.agotnes@infomedia.uib.no

Guido Boella
Università degli Studi di Torino, Corso Svizzera 185, 10149 Torino, Italy
E-mail: guido@di.unito.it

Leon van der Torre
Université du Luxembourg
6 Rue Richard Coudenhove-Kalergi, 1359 Luxembourg, Luxembourg
E-mail: leon.vandertorre@uni.lu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-22358-7 e-ISBN 978-3-642-22359-4
DOI 10.1007/978-3-642-22359-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: Applied for

CR Subject Classification (1998): I.2, F.3, D.2, C.2.4, F.4.1, D.3

LNCS Sublibrary: SL 7 – Artificial Intelligence

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

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

These are the proceedings of the 12th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA XII), held during July 17–18, 2011 in Barcelona, and co-located with the 22nd International Joint Conference on Artificial Intelligence.

Multi-agent systems are systems of interacting autonomous agents or components that can perceive and act upon their environment to achieve their individual goals as well as joint goals. Research on such systems integrates many technologies and concepts in artificial intelligence and other areas of computing as well as other disciplines. Over recent years, the agent paradigm gained popularity, due to its applicability to a full spectrum of domains, from search engines to educational aids to electronic commerce and trade, e-procurement, recommendation systems, simulation and routing, to mention only some.

Computational logic provides a well-defined, general, and rigorous framework for studying syntax, semantics and procedures for various tasks by individual agents, as well as interaction amongst agents in multi-agent systems, for implementations, environments, tools, and standards, and for linking together specification and verification of properties of individual agents and multi-agent systems.

The purpose of the CLIMA workshops is to provide a forum for discussing techniques, based on computational logic, and for representing, programming and reasoning about agents and multi-agent systems in a formal way.

Former CLIMA editions have mostly been conducted in conjunction with major computational logic and artificial intelligence events such as CL in 2000, ICLP in 2001 and 2007, FLoC in 2002, LPNMR and AI-Math in 2004, JELIA in 2004 and 2008, AAMAS in 2006, MATES in 2009, and ECAI in 2010. In 2005, CLIMA VI was organized as a stand-alone event.

CLIMA XII closely followed the format established by its predecessor, with regular proceedings and two special sessions: "Logics for Games and Social Choice", organized by Thomas Ågotnes, and "Norms and Normative Multi-Agent Systems", organised by Guido Boella and Leon van der Torre.

Norms are pervasive in everyday life and influence the conduct of the entities subject to them. One of the main functions of norms is to regulate the behavior and relationships of agents. Accordingly, any agent or multi-agent system, if intended to operate in or model a realistic environment has to take norm regulating into account. Norms have been proposed in multi-agent systems and computer science to deal with coordination and security issues, and to model multi-agent organizations and legal aspects of electronic institutions and electronic commerce.

Logic and game theory form two theoretical underpinnings of multi-agent systems. On one hand, formal logic is a foundation for knowledge representa-

tion and reasoning, and opens the door to techniques for formal specification and automated verification. On the other hand, the interaction between rational decision makers has been studied in game theory for a long time. However, traditional game theory is not concerned with formal languages or reasoning systems, nor with computational issues, and until relatively recently formal logic was not concerned with expressing properties of game-like situations. For reasoning about interesting properties of many, if not most, multi-agent systems, we need game theoretic concepts such as strategies, preferences, etc. In particular, many multi-agent systems can be seen as implementing social choice mechanisms.

This 12th CLIMA edition received an exceptionally high number of submissions. Many of those involved in the revision and selection process acknowledged the high quality of the program. In many instances the authors expressed their satisfaction with very informative and constructive reviews, for which CLIMA is renown.

This book features regular papers as well as abstracts of invited talks. These were delivered by Simon Parsons (Brooklyn College, USA), Ulle Endriss (University of Amsterdam, The Netherlands), and Jan Broersen (Utrecht University, The Netherlands). The chapters in this book are organized according to the workshop schedule, in topical sessions. The main track and each of the special sessions started with an invited talk.

CLIMA opened with a session on "Secrets and Trust". In his invited talk, Simon Parsons discussed the use of argumentation for reasoning about which individuals to trust, and for relating sources of information to conclusions drawn from information provided by those sources. Robert Demolombe then looked at information and trust propagation, exploring formal notions of sincerity, competence, vigilance, cooperation, validity and completeness. Afterwards, Sara Miner More and Pavel Naumov presented a follow-up of their CLIMA XI work, with a theoretical study of a logic of dependence between secrets, addressing the question: which secrets functionally determine which others?

In the second half of the morning, we had three presentations on "Knowledge and Beliefs", covering topics such as knowledge-based protocols, security in information exchange, but also modalities for modeling beliefs and information sources, belief merging, information aggregation, and the concept of definability. Hans Van Ditmarsch and Fernando Soler-Toscano opened the session by presenting a three-step protocol which allows two players to (publicly) inform each other about their playing cards without making it known to a third player, the eavesdropper. Emiliano Lorini, Laurent Perrussel, and Jean-Marc Thévenin then illustrated a framework for processing signed information before incorporating it in the agent's beliefs. The last presentation of the morning, based on work by Hans Van Ditmarsch, David Fernández and Wiebe van der Hoek, dealt with epistemic logic: under what conditions can individual Kripke models be uniquely characterized (up to bi-simulation) by a single epistemic formula?

The afternoon, entirely devoted to the special session on "Logics for Games and Social Choice", was opened by Ulle Endriss's invited talk. While illustrating recent work conducted by members of his group at the University of Amster-

dam, Ulle showed many different ways in which modern logic can contribute to the study of social choice theory. Following the invited talk, five regular papers covered theoretical aspects and applications. Jan Van Eijck presented a new (geometric) proof of the Gibbard – Satterthwaite theorem, based on the Saari triangle for three alternatives, and elaborated on the notion of non-manipulability and the need for a finer granularity. Tiago De Lima's talk on alternating-time announcement logic presented a very general logic of action and change. Jan Calta, Dmitry Shkatov and Holger Schlingloff presented results on synthesizing strategies for multi-agent systems. Thomas Ågotnes and Natasha Alechina then discussed how coalition logic reasoning can be done in standard PDL-like logics. A final presentation by Daniele Porello and Ulle Endriss illustrated a proposal to regard ontology merging as a problem of social choice.

The second day was opened by a session featuring four talks on "Cooperation", covering aspects related to interaction protocols, teams, commitments, query-answering, monitoring, verification and diagnosis. Özgür Kafalı and Paolo Torroni started with a systematic analysis of types of delegation, similarity delegation and improper delegation. Taolue Chen, Marta Kwiatkowska, David Parker and Aistis Simaitis proposed using probabilistic model checking as an analysis tool for organizational construction. Samy Sá and João Alcântara presented some strategies to generate cooperative answers in query-answering systems: when there is no correct answer to a given query, it is more helpful to return some answers related to the query using query relaxation. In the last presentation, Özgür Kafalı, Francesca Toni and Paolo Torroni discussed an assumption-based argumentation approach to diagnosis of commitment exceptions.

Three presentations on "Logic and Languages" for agent programming concluded the morning. Domenico Corapi, Daniel Sykes, Katsumi Inoue and Alessandra Russo discussed a proposal for rule learning, aimed to enrich abductive reasoning with a probabilistic component as well as to model inductive logic programming tasks as a special form of abductive reasoning. Richard Stocker, Maarten Sierhuis, Louise Dennis, Clare Dixon and Michael Fisher then presented a formal semantics for the Brahms modeling and simulation framework for human – agent teamwork. Finally, Alfredo Gabaldon showed how norm-enforcing mechanisms can be accommodated in the Golog situation calculus-based programming language.

The final part of the CLIMA program featured the special session on "Norms and Normative Multi-Agent Systems", with one invited talk and five regular paper presentations. In his invited talk, Jan Broersen discussed modeling of obligations to attempt an action in a probabilistic *stit* framework extended with deontic modalities, and the effects of reasoning with probabilities on the semantics of deontic modalities like obligation and prohibition. Jan's talk was followed by two presentations on actions and norms that used modal style logics. Andreas Herzig, Emiliano Lorini and Nicolas Troquard defined a logic of action which enables reasoning about the distinction between physical actions bringing about brute facts and institutional actions bringing about institutional facts. Mathieu Beirlaen and Christian Straßer, instead, proposed a paraconsistent ap-

proach for dealing with normative conflicts in multi-agent systems. Afterwards, a last group of three presentations discussed research in norms and normative multi-agent systems using rules style logics. Marco Alberti, Ana Sofia Gomes, Ricardo Gonçalves, João Leite and Martin Slota showed an application of a logic combining DL ontologies with rule-based knowledge bases for representing and reasoning about norms in multi-agent systems. Nir Oren, Wamberto Vasconcelos, Felipe Meneguzzi and Michael Luck then presented an approach in which norms are constraints and are used to generate plans that satisfy those norms that yield the highest utility. In the concluding talk, Guido Governatori and Antonino Rotolo extended a logic of violation with time, thus enabling the representation of compliance with respect to different types of legal obligation and different temporal constraints over them, as well as the representation of reparative obligations.

The 22 papers presented at CLIMA XII were selected from 43 submissions. These were on average of very high quality. In line with the high standards of previous editions, the final acceptance rate, after two rounds of reviewing and selection, was circa 50%. The Program Committee consisted of 48 top-level researchers from 35 institutions located in 5 continents and 15 countries. Nine additional reviewers helped in the process. The papers in this book have been authored by 58 researchers worldwide.

Further information about CLIMA XII is available from the website `http://centria.di.fct.unl.pt/events/climaXII/`. General information about the workshop series, with links to past and future events, can be found on the CLIMA workshop series home page, `http://centria.di.fct.unl.pt/~clima/`.

We thank all the authors of papers submitted to CLIMA XII, the invited speakers, the members of the Program Committee, and the additional reviewers, for ensuring that CLIMA keeps up to its high standards. A special thank you goes to Adele Howe, the IJCAI 2011 Workshop Chair, and to the local organizers in Barcelona for their help and support.

July 2011

João Leite
Paolo Torroni
Thomas Ågotnes
Guido Boella
Leon van der Torre

Organization

Workshop Chairs

João Leite	Universidade Nova de Lisboa, Portugal
Paolo Torroni	University of Bologna, Italy

Special Session Chairs

Thomas Ågotnes	University of Bergen, Norway
Guido Boella	University of Turin, Italy
Leon van der Torre	ILIAS, University of Luxembourg

Program Committee

Natasha Alechina	University of Nottingham, UK
Jose Julio Alferes	Universidade Nova de Lisboa, Portugal
Alexander Artikis	NCSR *Demokritos*, Athens, Greece
Rafael H. Bordini	Federal University of Rio Grande do Sul, Brazil
Gerhard Brewka	Leipzig University, Germany
Jan Broersen	Utrecht University, The Netherlands
Nils Bulling	Clausthal University of Technology, Germany
Stefania Costantini	Università di L'Aquila, Italy
Célia Da Costa Pereira	University of Nice Sophia Antipolis, France
Mehdi Dastani	Utrecht University, The Netherlands
Marina De Vos	University of Bath, UK
Louise Dennis	University of Liverpool, UK
Juergen Dix	Clausthal University of Technology, Germany
Michael Fisher	University of Liverpool, UK
Nicoletta Fornara	Università della Svizzera Italiana, Switzerland
Dov Gabbay	King's College, London, UK
Chiara Ghidini	FBK-irst, Trento, Italy
Guido Governatori	NICTA, Brisbane, Australia
Davide Grossi	University of Amsterdam, The Netherlands
Paul Harrenstein	Technische Universität München, Germany
Hisashi Hayashi	Toshiba Corporation, Japan
Koen Hindriks	Delft University of Technology, The Netherlands
Katsumi Inoue	NII, Tokyo, Japan
Wojtek Jamroga	University of Luxembourg
Jérôme Lang	Université Paris Dauphine, France
Alessio Lomuscio	Imperial College London, UK
Emiliano Lorini	IRIT, Toulouse, France
Viviana Mascardi	University of Genova, Italy

John-Jules Meyer	Utrecht University, The Netherlands
Jan Odelstad	University of Gävle, Sweden
Mehmet Orgun	Macquarie University, Sydney, Australia
Eric Pacuit	Tilburg University, The Netherlands
Maurice Pagnucco	The University of New South Wales, Australia
Gabriella Pigozzi	Université Paris Dauphine, France
Jeremy Pitt	Imperial College London, UK
Enrico Pontelli	New Mexico State University, USA
R. Ramanujam	Chennai Mathematical Institute, India
Antonino Rotolo	University of Bologna, Italy
Fariba Sadri	Imperial College London, UK
Chiaki Sakama	Wakayama University, Japan
Ken Satoh	NII and Sokendai, Tokyo, Japan
Tran Cao Son	New Mexico State University, USA
Michael Thielscher	The University of New South Wales, Australia
Nicolas Troquard	University of Essex, UK
Wiebe Van Der Hoek	University of Liverpool, UK
M. Birna Van Riemsdijk	Delft University of Technology, The Netherlands
Wamberto Vasconcelos	University of Aberdeen, UK
Cees Witteveen	Delft University of Technology, The Netherlands

Additional Reviewers

Dongmo Zhang	The University of New South Wales, Australia
Frederic Moisan	IRIT, Toulouse, France
Ji Ruan	The University of New South Wales, Australia
Johannes Oetsch	Vienna University of Technology, Austria
Massimo Benerecetti	Università di Napoli *Federico II*, Italy
Nir Piterman	University of Leicester, UK
S P Suresh	Chennai Mathematical Institute, India
Sunil Simon	CWI, The Netherlands
Tristan Behrens	Clausthal University of Technology, Germany

CLIMA Steering Committee

Jürgen Dix	Clausthal University of Technology, Germany
Michael Fisher	University of Liverpool, UK
João Leite	Universidade Nova de Lisboa, Portugal
Fariba Sadri	Imperial College London, UK
Paolo Torroni	University of Bologna, Italy

CLIMA Publications

Special Issues

- *Journal of Logic and Computation*, Special Issue on Computational Logic and Multi-Agent Systems. Expected, 2012.
- *Annals of Mathematics and Artificial Intelligence*. Special Issue on Computational Logic and Multi-Agent Systems, guest-edited by Jürgen Dix and João Leite. Expected, 2011.
- *Journal of Autonomous Agents and Multi-Agent Systems*, 16(3), 2008. Special Issue on Computational Logic-Based Agents, guest-edited by Francesca Toni and Jamal Bentahar.
- *Annals of Mathematics and Artificial Intelligence*, 42(1-3), 2004. Special Issues on Computational Logic and Multi-Agent Systems, guest-edited by Jürgen Dix Dix, João Leite, and Ken Satoh.
- *Annals of Mathematics and Artificial Intelligence*, 37(1-2), 2003. Special Issue on Computational Logic and Multi-Agent Systems, guest-edited by Jürgen Dix, Fariba Sadri and Ken Satoh.
- *Electronic Notes in Theoretical Computer Science*, 70(5), 2002. Special Issue on Computational Logic and Multi-Agency, guest-edited by Jürgen Dix Dix, João Leite, and Ken Satoh.

Proceedings

- *Computational Logic in Multi-Agent Systems XII, Proceedings*. Vol. 6814 of *Lecture Notes in Artificial Intelligence*, edited by João Leite, Paolo Torroni, Thomas Ågotnes, Guido Boella, and Leon van der Torre. Springer-Verlag Berlin Heidelberg 2011.
- *Computational Logic in Multi-Agent Systems XI, Proceedings*. Vol. 6245 of *Lecture Notes in Artificial Intelligence*, edited by Jürgen Dix, João Leite, Guido Governatori, and Wojtek Jamroga. Springer-Verlag Berlin Heidelberg 2010.
- *Computational Logic in Multi-Agent Systems X, Revised Selected and Invited Papers*. Vol. 6214 of *Lecture Notes in Artificial Intelligence*, edited by Jürgen Dix, Michael Fisher, and Peter Novák. Springer-Verlag Berlin Heidelberg 2010.
- *Computational Logic in Multi-Agent Systems IX, Revised Selected and Invited Papers*. Vol. 5405 of *Lecture Notes in Artificial Intelligence*, edited by Michael Fisher, Fariba Sadri, and Michael Thielscher. Springer-Verlag Berlin Heidelberg 2009.
- *Computational Logic in Multi-Agent Systems VIII, Revised Selected and Invited Papers*. Vol. 5056 of *Lecture Notes in Artificial Intelligence*, edited by Fariba Sadri and Ken Satoh. Springer-Verlag Berlin Heidelberg 2008.

- *Computational Logic in Multi-Agent Systems VII, Revised Selected and Invited Papers*. Vol. 4371 of *Lecture Notes in Artificial Intelligence*, edited by Katsumi Inoue, Ken Satoh, and Francesca Toni. Springer-Verlag Berlin Heidelberg, Germany, 2007.
- *Computational Logic in Multi-Agent Systems VI, Revised Selected and Invited Papers*. Vol. 3900 of *Lecture Notes in Artificial Intelligence* (State-of-the-Art Survey), edited by Francesca Toni and Paolo Torroni, Springer-Verlag Berlin Heidelberg, Germany, 2006.
- *Computational Logic in Multi-Agent Systems V, Revised Selected and Invited Papers*. Vol. 3487 of *Lecture Notes in Artificial Intelligence*, edited by João Leite Leite and Paolo Torroni. Springer-Verlag Berlin Heidelberg, Germany, 2005.
- *Computational Logic in Multi-Agent Systems IV, Revised Selected and Invited Papers*. Vol. 3259 of *Lecture Notes in Artificial Intelligence*, edited by Jürgen Dix and João Leite. Springer-Verlag Berlin Heidelberg, Germany, 2004.

Early Editions

- Proceedings of the 5th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA V), edited by João Leite Leite and Paolo Torroni. Lisbon, Portugal, ISBN: 972-9119-37-6, September 2004.
  `http://centria.di.fct.unl.pt/~jleite/climaV/climaV-preprocs.pdf`
- Proceedings of the 4th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA IV), edited by Jürgen Dix and João Leite. ITZ Bericht 1(5). Papierflieger Verlag, Clausthal-Zellerfeld, Germany, ISBN 3-89720-688-9, 2003.
  `http://centria.di.fct.unl.pt/~jleite/climaIV/climaIV-TR.pdf`
- Proceedings of the Third International Workshop on Computational Logic in Multi-Agent Systems (CLIMA III), edited by Jürgen Dix, João Leite, and Ken Satoh. Datalogiske Skrifter 93, Roskilde University, Denmark, ISSN 0109-9779, 2002.
  `http://centria.di.fct.unl.pt/~jleite/papers/clima02_procs.pdf`
- ICLP 2001 Workshop on Computational Logic in Multi-Agent Systems (CLIMA II), held in association with ICLP 2001, Paphos, Cyprus, December 1, 2001. Organized by Ken Satoh and Jürgen Dix.
  `http://research.nii.ac.jp/~ksatoh/clima01.html`
- CL-2000 Workshop on Computational Logic in Multi-Agent Systems (CLIMA I), held in association with the International Conference on Computational Logic, Imperial College London, UK, July 24-25, 2000. Organized by Ken Satoh and Fariba Sadri.
  `http://research.nii.ac.jp/~ksatoh/clima00.html`
- Workshop on Multi-Agent Systems in Logic Programming (MAS-LP), held in conjunction with the 16th International Conference on Logic Programming, Las Cruces, NM, November 30, 1999. Organized by Stephen Rochefort, Fariba Sadri, and Francesca Toni.
  `http://www.cs.sfu.ca/news/conferences/MAS99/`

Table of Contents

Secrets and Trust

Knowledge and Beliefs

Logics for Games and Social Choice

Cooperation

Logic and Languages

Norms and Normative Multi-agent Systems

Some Thoughts on Using Argumentation to Handle Trust

(Invited Talk)

Simon Parsons[1,2], Yuqing Tang[2], Kai Cai[2], Elizabeth Sklar[1,2], and Peter McBurney[3]

[1] Department of Computer & Information Science, Brooklyn College,
City University of New York, 2900 Bedford Avenue, Brooklyn, NY 11210 USA
{sklar,parsons}@sci.brooklyn.cuny.edu
[2] Department of Computer Science, Graduate Center
City University of New York, 365 Fifth Avenue, New York, NY 10016, USA
{ytang,kcai}@gc.cuny.edu
[3] Department of Informatics, Kings College London,
Strand Building,
peter.mcburney@kcl.ac.uk

Abstract. This paper describes some of our recent work on using argumentation to handle information about trust. We first discuss the importance of trust in computer science in general and in multi-agent systems in particular. We then describe the setting of our work, situating it within the broad area of work on trust. Next we provide an overview of two lines of work we are currently pursuing — using argumentation to reason about which individuals to trust, and using argumentation to relate sources of information to conclusions drawn from information provided by those sources. Finally, we outline our current initiatives and briefly highlight other work that is closely related to ours.

1 Why Trust Is Important

Trust is a mechanism for managing the uncertainty about autonomous entities and the information they deal with. As a result, trust can play an important role in any decentralized system. As computer systems have become increasingly distributed, and control in those systems has become more decentralized, trust has become an increasingly more important concept in computer science [3,12].

Much of the work on trust in computer science has concentrated on dealing with specific scenarios in which trust has to be established or handled in some fashion. Thus, we see work on trust in peer-to-peer networks, including the EigenTrust algorithm [19] — a variant of PageRank [27] where downloads from a source play the same role as outgoing hyperlinks and which is effective in excluding peers who want to disrupt the network. [1], also in the area of peer-to-peer networks, develops a mechanism that prevents peers manipulating their trust values to get preferential downloads. [44] is concerned with slightly different issues in mobile ad-hoc networks, looking to prevent nodes from getting others to transmit their messages while refusing to transmit the messages of others.

The internet, as the largest distributed system of all, is naturally a target of much of the research on trust. There have, for example, been studies on the development of

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 1–12, 2011.

trust in ecommerce through the use of reputation systems [34] and studies on how such systems perform [33,39] and how such systems can be manipulated [21]. Another area of concern has to do with the reliability of sources of information on the web. [43], for example, investigates mechanisms to determine which sources to trust when faced with multiple conflicting sources, while [4] looks at the related question of how to resolve conflicting information, and [2] extends this idea to rate the individuals who provide information by looking at the history of the information they have provided. Issues related to trust in the social web have also attracted much attention [11,22,39,42].

Trust is an especially important issue from the perspective of autonomous agents and multiagent systems. The premise behind the multiagent systems field is that of developing software agents that will work in the interests of their owners, carrying out their owners' wishes while interacting with other entities. In such interactions, agents will have to reason about the amount that they should trust those other entities, whether they are trusting those entities to carry out some task, or whether they are trusting those entities to not misuse crucial information. As a result we find much work on trust in agent-based systems [35], including the influential model proposed by [6].

We are studying the use of formal systems of argumentation for handling trust. In this paper we present a brief, and largely informal, overview of our work to date, with pointers to the detailed treatment of all the topics we introduce.

2 The Setting for Our Work

Our work considers an agent Ag which is part of a society of agents Ags. Each agent $Ag_i \in Ags$ has a knowledge base Σ_i that contains the information that the agent has about the world. The agents communicate, and so each Ag_i can make use of not only the information that is in its Σ_i, but also the information that comes from other agents. We model this situation by considering that each Ag_i has a *commitment store* CS_i which contains information that that agent has communicated. Since at the moment we are only concerned with the perspective of the one agent Ag, we only consider the commitment stores of agents other than Ag to hold information that the respective agents have communicated to Ag. That is we ignore issues around how the information was communicated, who else knows the same information, and so on. We just consider all information that some agent Ag_i has communicated to Ag to be contained in CS_i.

We assume that all agents that have communicated information to Ag are members of Ag's social network, and so it is possible to construct a graph which relates Ag to all these agents. We further assume that it is possible to attach a numerical measure to each link in this social network to quantify the extent to which an agent trusts those to which it is linked in the social network. In common with the literature on trust in social networks, we call such a structure a *trust network*. Figure 1 is a fragment of the trust network for an agent $john$ identifying $john$'s relationships with the agents $dave$, $mary$, $alice$ and $jane$. (This social network is taken from an example in [20], which itself is drawn from the FilmTrust network [7,11], with the trust values normalized between 0 and 1.)

Our work has concentrated on establishing mechanisms by which our featured agent Ag, $john$ in the case of Figure 1, can use the information it obtains from its acquaintances in a way that reflects its trust in them. Thus, there is an assumption that if some proposition $p \in CS_j$, then Ag will accord that proposition the belief $bel(p)$ where:

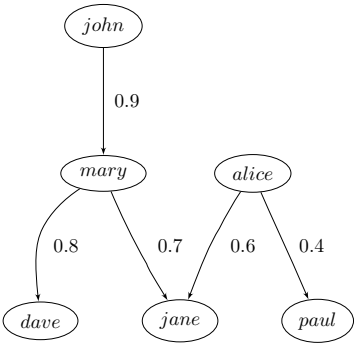


Fig. 1. A simple trust network

$$bel(p) = ttb(tr(Ag, Ag_j)) \tag{1}$$

That is, $bel(p)$ is some function of the trust that Ag has in Ag_j regarding its belief in p. Thus, in the example of Figure 1, we consider that if $john$ is told p by $jane$, we assume that $john$'s belief in this is a function of his trust in $jane$.

Thinking a little about $john$'s relationship to $jane$ raises some issues, one directly, and others that follow from that first issue. The first issue is that according to Figure 1, $john$ doesn't know $jane$. $john$ only knows $mary$, and so only directly states his trust in $mary$. However, it is common in the literature of trust in social networks to assume that trust propagates (or, to describe it in another way, is transitive) so that since $john$ trusts $mary$ and $mary$ trusts $jane$, then $john$ can trust $jane$. The assumption of transitivity (and it is a big assumption, as discussed in [8]) then raises additional issues.

The first of these is the context in which $john$ trusts $jane$. As many authors have pointed out, not least [8], trust is highly context dependent. $john$ may trust $jane$ in some domains, but not in others. To use a common example, $john$ may trust $jane$, a car mechanic, for information on cars and how to fix them. However, just because $john$ trusts $jane$ about cars does not mean that he will trust her to recommend a restaurant or someone to babysit his daughter. Indeed, as [17] points out, the semantics of the links between $john$ and $mary$ and between $john$ and $jane$ are rather different. Consider that $john$ is, as in the example from which Figure 1 is taken in [20], looking for recommendations about which film to watch. If $john$ solicits information from his friend $mary$, he is trusting her to recommend films he would like to watch. However, if he accepts an indirect recommendation from $jane$, whom he trusts because $mary$ trusts her, he is relying on $jane$ being a good recommender of films but is relying on $mary$ being a good recommender of film recommenders. In other words, $john$ is not trusting $mary$'s film knowledge in this second case, but her knowledge of people who can recommend films. [17] distinguishes between *functional* trust, the trust in an individual to carry out some task (such as recommending a film), and *referral* trust, the trust in an individual's recommendation of another individual. In our work, we ignore these distinctions, assuming that the trust networks capture a context in which transitivity holds. (We can easily imagine an agent having different trust networks for different contexts, each composed of functional and referral trust links.)

3 How We Use Argumentation in Handling Trust

In this section we describe how our work makes use of argumentation to handle trust.

3.1 How Argumentation Can Help

Some models of trust, for example the influential model from [6], start from the position that trust in an individual is only ever important when the trusting party needs that individual to perform an action. While it is clear that in such cases trust is important — if we are going to construct a plan where a critical action is to be performed by individual I then we certainly need to trust I to do what we need them to — we, like [23] for example, believe that trust in individuals also has a role to play when those individuals provide us with information.

In particular, we believe that knowing the sources of the information and the way in which it is used, that is the *provenance* of the information, is important when the sources of the information may not be completely trustworthy. As [26] points out,

> By knowing the provenance of data, users can often better understand, trust, reproduce, and validate it.

The relationship between trust and provenance has been explored by a number of authors, for example [10,32,40,41]. As we argued in [29], since argumentation records the data from which conclusions are drawn, it provides a natural mechanism to capture provenance. This is especially the case when the argument records not only the data from which conclusions are drawn, but also the full derivation of those conclusions (as, for example, in [9,31]).

3.2 Our Contribution so Far

We are working on two related lines of research. In both, we use argumentation, as described above, to make explicit the connection between sources of data and the conclusions derived from that data. In one line of research, we use argumentation to carry out reasoning about the trustworthiness of sources — that is to compute trust propagation — with the aim of permitting reasoning about the validity of different forms of propagation. In the other line of research, we use a graphical representation of arguments as a means of communicating provenance information, as will be explained below.

In using argumentation to compute trust propagation [30], we assume that each agent Ag_i has a knowledge base, $\Delta_i^{tr} \subseteq \Sigma_i$, containing information about who trusts whom. Table 1 contains Δ_{john}^{tr}, the knowledge base for $john$, constructed from the example in Figure 1. Each element of Δ_{john}^{tr} has the form:

$$(\langle index \rangle : \langle data \rangle : \langle value \rangle)$$

The first is a means of referring to the element, the second is a formula, and the third is the degree of trust between the individuals.

Table 1. Knowledge base containing $john$'s beliefs

$$\begin{array}{l}\Delta^{tr}_{john}\ (t1 : trusts(john, mary) : 0.9)\\ \qquad\quad (t2 : trusts(mary, jane) : 0.7)\\ \qquad\quad (t3 : trusts(mary, dave) : 0.8)\\ \qquad\quad (t4 : trusts(alice, jane) : 0.6)\\ \qquad\quad (t5 : trusts(alice, paul) : 0.4)\end{array}$$

Arguments can then be constructed from Δ^{tr}_{john} using the rules in Figure 2[1]. For example, using the first two rules, from Figure 2, Ax^{tr} and dp, it is possible to construct the argument:

$$\Delta^{tr}_{john} \vdash_{tr} (trusts(john, jane) : \{t1, t2\} : \{Ax^{tr}, Ax^{tr}, dp\} : \tilde{t})$$

where all arguments in our approach take the form:

$$(\langle conclusion\rangle : \langle grounds\rangle : \langle rules\rangle : \langle value\rangle)$$

The $\langle conclusion\rangle$ is inferred from the $\langle grounds\rangle$ using the rules of inference $\langle rules\rangle$ and with degree $\langle value\rangle$. In this case the argument says $john$ trusts $jane$ with degree $\tilde{t}$ (which is $0.9 \otimes^{tr} 0.7$), through two applications of the rule Ax^{tr} and one application of the rule dp to the two facts indexed by $t1$ and $t2$[2]. Using just Ax^{tr} and dp captures the transitivity of trust, and the crucial inference step is what [13] calls "direct propagation" (hence the name of the rule dp). Figure 3a shows the result of establishing all the indirect links possible using direct propagation where $\otimes^{tr}$ is taken to be $\min$, as we do in [30].

The reason for constructing arguments about trust is primarily so that it is possible to tell on what basis the conclusion to trust a particular source has been drawn. We do this because there are a variety of reasons that one might have for trusting a source, and it may be necessary to identify which reasons(s) have been used in a particular case in order to be able to dispute or defend them. Making the reasons explicit, as our approach does, unleashes this possibility. As an example of an alternative to transitivity as a form of trust propagation, consider the rule cc from Figure 2. This captures a form of reasoning that [13] calls *co-citation*. [13] describes this as:

> For example, suppose i_1 trusts j_1 and j_2, and i_2 trusts j_2. Under co-citation, we would conclude that i_2 should also trust j_1.

In our example (see Figures 1 and 3b), therefore, co-citation suggests that since $alice$ trusts $jane$ and $paul$, and $mary$ trusts $jane$, then $mary$ should trust $paul$. The idea is that since $alice$ and $mary$ agree on the trustworthiness of $jane$, $mary$ should trust $alice$'s opinion about $paul$. [13] also tells us how trust values should be combined in

[1] Note that the consequence relation in Figure 2 is not intended to be comprehensive. There are many other ways to construct arguments about trust — for some examples see [28] — which could be included in the definition of $\vdash_{tr}$.

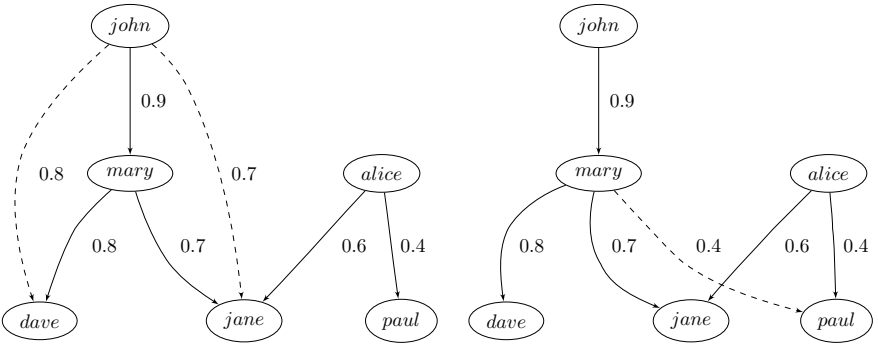
[2] As mentioned above, there are good reasons for using the formulae themselves in the grounds and factoring the whole proof into the set of rules (as we do in [29]) to obtain structured arguments like those in [9, 31]. However, for simplicity, here we use the relevant indices.

$$Ax^{tr} \quad \frac{(n : trusts(x, y) : \tilde{d}) \in \Delta_i^{tr}}{\Delta_i^{tr} \vdash_{tr} (trusts(x, y) : \{n\} : \{Ax^{tr}\} : \tilde{d})}$$

$$dp \quad \frac{\Delta_i^{tr} \vdash_{tr} (trusts(x, y) : G : R : \tilde{d}) \text{ and } \Delta_i^{tr} \vdash_{tr} (trusts(y, z) : H : S : \tilde{e})}{\Delta_i^{tr} \vdash_{tr} (trusts(x, z) : G \cup H : R \cup S \cup \{dp\} : \tilde{d} \otimes^{tr} \tilde{e})}$$

$$cc \quad \frac{\Delta_i^{tr} \vdash_{tr} (trusts(x, y) : G : R : \tilde{d}) \text{ and } \Delta_i^{tr} \vdash_{tr} (trusts(x, z) : H : S : \tilde{e}) \text{ and } \Delta_i^{tr} \vdash_{tr} (trusts(w, z) : K : T : \tilde{f})}{\Delta_i^{tr} \vdash_{tr} (trusts(w, y) : G \cup H \cup K : R \cup S \cup T \cup \{cc\} : \tilde{d} \otimes^{tr} \tilde{e} \otimes^{tr} \tilde{f})}$$

Fig. 2. Some rules for propagating trust



(a) Direct propagation (b) Co-citation

Fig. 3. Two types of trust propagation — in each figure the dashed lines show the indirect trust relationships that are inferred

this case — $mary$'s trust in $paul$ is just the combination of trust values along the path from $mary$ to $jane$ to $alice$ to $paul$. Figure 3b shows the direct trust link that co-citation implies in our example where the values are again combined using $\min$, with $mary$ trusting $paul$ to degree 0.4.

Combining the application of cc with the use of dp, as above, allows the construction of the argument:

$$\Delta^{tr}_{john} \vdash_{tr} (trusts(john, paul) : \{t1, t2, t4, t5\} : rules_1 : \tilde{r})$$

indicating that $john$ trusts $paul$ (based on $john$'s knowledge base as in Table 1), where $rules_1$ is:

$$\{Ax^{tr}, Ax^{tr}, Ax^{tr}, Ax^{tr}, cc, dp\}$$

and $\tilde{r}$ is $0.9 \otimes^{tr} 0.7 \otimes^{tr} 0.6 \otimes^{tr} 0.4$ (which also comes to 0.4 when $\otimes^{tr}$is $\min$).

Once these trust values have been established, $john$ can then apply Equation 1 to establish a degree of belief for anything the other agents in the trust network tell him, and then combine this information with what he already knows. [30] describes formally how this can be done.

The description above concerns the first of the two lines of work we are pursuing: the use of argumentation to capture trust propagation. Our second line of work employs a graphical model of arguments as the basis of communicating provenance information. The model we have developed [38, 37] is best explained through the use of an instance of a graphical argument of the kind generated by the model. Such a graph is given in Figure 4. In short, the graph contains three components. First, there is a trust graph. In this case, it is a subset of the trust graph from Figure 1 — exactly that bit of the graph from Figure 1 which contains the agents from whom $john$ can infer trust using direct propagation. Second, there are arguments. These are proof trees where the conclusions of the trees are formulae of interest, and each formula that is not an inference is linked to the agent that supplies the information. In this case, the arguments — taken from the example in [20] — concern whether or not to watch the Pedro Almodovar film "Hable con ella" (abbreviated "hce" in Figure 4). Again Equation 1 tells us how

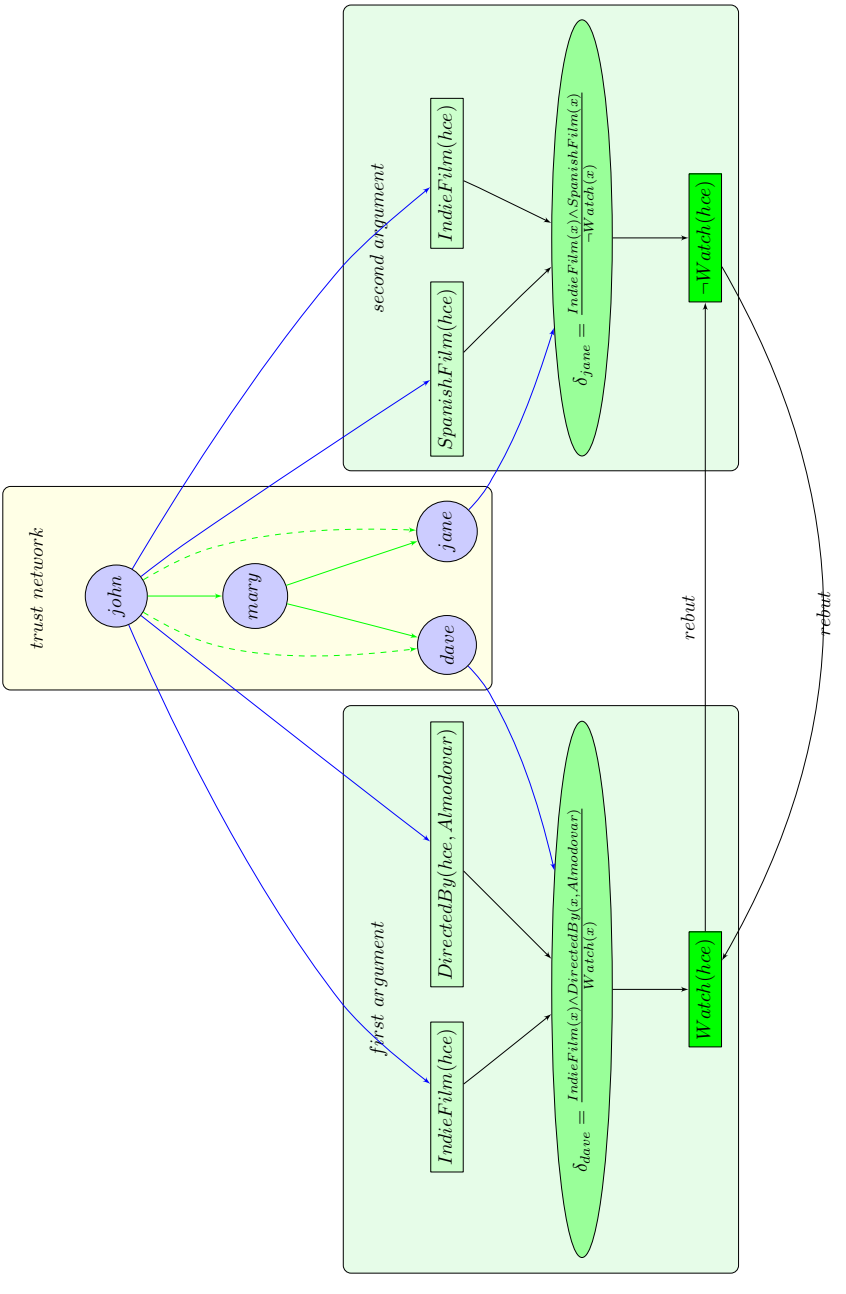


Fig. 4. An example of a trust-extended argumentation graph

to establish belief values for information that comes from different agents in a way that depends on trust in the source agent(s). Third, there are arcs that identify conflicts between arguments. In Figure 4, these are just between the conclusions of arguments, but more general conflicts are also identified in the full model [37]. Together these three components make up a *trust-extended argumentation graph*.

The full model [37] not only includes a formal description of each of these components, but also translates standard approaches for evaluating arguments [5] into criteria that can be evaluated on the graph. In addition, we have identified algorithms for building the graph and then evaluating the arguments that it contains.

3.3 Current and Future Work

Our long term aim is to build tools that help users to reason with information from sources of varying trustworthiness. Our hypothesis is that the graphical argumentation model we illustrated above is a useful way to do this. Now that we have a formal model which makes it possible to automate reasoning and link sources to conclusions, we are working towards testing this hypothesis. To do this, we need an interactive tool, and implementing such a tool is one of our current foci. One important aspect of the implementation is the interface. While we believe that it is useful to show users how sources of information relate to conclusions, we do not think that users will respond well to complete graphs of the kind in Figure 4 — there is too much information for this to be easily digested by the average user. Instead we are looking at ways of allowing users to navigate sections of the graph, zooming in on areas of interest. To help us understand the best way to do this, we are running user studies where human subjects develop argument graphs for some simple problem scenarios.

4 Conclusions

We have described our work on using argumentation to handle trust. We focus on the situation in which some entity uses information that comes from sources of varying trustworthiness, and look at how argumentation can be used to capture the provenance of the information used to derive answers to queries of interest. There are two main aspects of our work to date. One is the use of argumentation as a means of capturing different mechanisms for propagating trust. The other is the generation of a graphical model that can be used to communicate provenance information to users. Our current work is concentrating on implementing the second model and experimentally determining how best to present results to users.

Finally we should note that though the particular combinations of argumentation and trust that we are studying are novel, the idea of combining trust and argumentation is not. Four lines of work on trust and argumentation that are complementary to ours are those of Harwood [14,15], Matt *at al.*, Hunter [16], [24], and Stranders [36]. For a detailed comparison of this work and ours, see [37]. In addition, argumentation has been used in the past to reason about risk [18,25], a subject closely related to trust; though the cited work looks at risk of carcinogenicity given chemical structure rather than risk due to untrustworthiness.

Acknowledgments

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

1. Abrams, Z., McGrew, R., Plotkin, S.: Keeping peers honest in EigenTrust. In: Proceedings of the 2nd Workshop on the Economics of Peer-to-Peer Systems (2004)
2. Adler, B.T., de Alfaro, L.: A content-driven reputation system for the Wikipedia. In: Proceedings of the 16th International World Wide Web Conference, Banff, Alberta (May 2007)
3. Artz, D., Gil, Y.: A survey of trust in computer science and the semantic web. Journal of Web Semantics 5(2), 58–71 (2007)
4. Dong, X.L., Berti-Equille, L., Srivastava, D.: Integrating conflicting data: The role of source dependence. In: Proceedings of the 35th International Conference on Very Large Databases, Lyon, France (August 2009)
5. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77, 321–357 (1995)
6. Falcone, R., Castelfranchi, C.: Social trust: A cognitive approach. In: Castelfranchi, C., Tan, Y. (eds.) Trust and Deception in Virtual Societies, pp. 55–99. Kluwer Academic Publishers, Dordrecht (2001)
7. http://trust.mindswap.org/FilmTrust/
8. Francone, R., Castelfranchi, C.: Transitivity in trust: A discussed property. In: Proceedings of the Undicesimo Workshop Nazionale "Dagli Oggetti agli Agenti", Rimini (September 2010)
9. García, A.J., Simari, G.: Defeasible logic programming: an argumentative approach. Theory and Practice of Logic Programming 4(1), 95–138 (2004)
10. Golbeck, J.: Combining provenance with trust in social networks for semantic web content filtering. In: Moreau, L., Foster, I. (eds.) IPAW 2006. LNCS, vol. 4145, pp. 101–108. Springer, Heidelberg (2006)
11. Golbeck, J., Hendler, J.: Filmtrust: Movie recommendations using trust in web-based social networks. In: Proceedings of the IEEE Consumer Communications and Networking Conference (2006)
12. Grandison, T., Sloman, M.: A survey of trust in internet applications. IEEE Communications Surveys and Tutorials 4(4), 2–16 (2000)
13. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: Proceedings of the 13th International Conference on the World Wide Web (2004)
14. Harwood, W.T., Clark, J.A., Jacob, J.L.: Networks of trust and distrust: Towards logical reputation systems. In: Gabbay, D.M., van der Torre,L. (eds.) Logics in Security, Copenhagen, Denmark (2010)
15. Harwood, W.T., Clark, J.A., Jacob, J.L.: A perspective on trust, security and autonomous systems. In: Proceedings of the New Security Paradigms Workshop, Concord, MA (2010)
16. Hunter, A.: Reasoning about the appropriateness of propoents for arguments. In: Proceedings of the 23rd AAAI Conference on Artificial Intelligence, Chicago, Illinois (July 2008)

17. Jøsang, A., Gray, E., Kinateder, M.: Simplification and analysis of transitive trust networks. Web Intelligence and Agent Systems 4(2), 139–161 (2006)
18. Judson, P.N., Fox, J., Krause, P.J.: Using new reasoning technology in chemical information systems. Journal of Chemical Information and Computer Sciences 36, 621–624 (1996)
19. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in P2P networks. In: Proceedings of the 12th World Wide Web Conference (May 2004)
20. Katz, Y., Golbeck, J.: Social network-based trust in prioritzed default logic. In: Proceedings of the 21st National Conference on Artificial Intelligence (2006)
21. Lang, J., Spear, M., Wu, S.F.: Social manipulation of online recommender systems. In: Bolc, L., Makowski, M., Wierzbicki, A. (eds.) SocInfo 2010. LNCS, vol. 6430, pp. 125–139. Springer, Heidelberg (2010)
22. Lerman, K., Galstyan, A.: Analysis of social voting patterns on Digg. In: Proceedings of the 1st Workshop on Online Social Networks, Seattle (August 2008)
23. Liau, C.-J.: Belief, information acquisition, and trust in multi-agent systems — a modal logic formulation. Artificial Intelligence 149, 31–60 (2003)
24. Matt, P.-A., Morge, M., Toni, F.: Combining statistics and arguments to compute trust. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagents Systems, Toronto, Canada (May 2010)
25. McBurney, P., Parsons, S.: Dialectical argumentation for reasoning about chemical carcinogenicity. Logic Journal of the IGPL 9(2), 191–203 (2001)
26. Miles, S., Groth, P., Munroe, S., Moreau, L.: PrIMe: A methodology for developing provenance-aware applications. ACM Transactions on Software Engineering and Methodology (2010) (to appear)
27. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bringing order to the Web. Technical Report 1999-66, Stanford InfoLab (1999)
28. Parsons, S., Haigh, K., Levitt, K., Rowe, J., Singh, M., Sklar, E.: Arguments about trust. Technical report, Collaborative Technology Alliance (2011)
29. Parsons, S., McBurney, P., Sklar, E.: Reasoning about trust using argumentation: A position paper. In: Proceedings of the Workshop on Argumentation in Multiagent Systems, Toronto, Canada (May 2010)
30. Parsons, S., Sklar, E., McBurney, P.: Using argumentation to reason with and about trust. In: Proceedings of the 8th International Workshop on Argumentation in Multiagent Systems, Taipei, Taiwan (2011)
31. Prakken, H.: An abstract framework for argumentation with structured arguments. Argument and Computation 1, 93–124 (2010)
32. Rajbhandari, S., Wootten, I., Ali, A., Rana, O.F.: Evaluating provenance-based trust for scientific workflows. In: Proceedinsg of the Sixth IEEE International Symposium on Cluster Computing and the Grid, pp. 365–372. IEEE Computer Society Press, Singapore (2006)
33. 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, pp. 127–157. Elsevier Science, Amsterdam (2002)
34. Resnick, P., Zeckhauser, R., Friedman, E., Kuwabara, K.: Reputation systems: Facilitating trust in internet interactions. Communications of the ACM 43, 45–48 (2000)
35. Sabater, J., Sierra, C.: Review on computational trust and reputation models. AI Review 23(1), 33–60 (2005)
36. Stranders, R., de Weerdt, M., Witteveen, C.: Fuzzy argumentation for trust. In: Sadri, F., Satoh, K. (eds.) CLIMA VIII 2007. LNCS (LNAI), vol. 5056, pp. 214–230. Springer, Heidelberg (2008)
37. Tang, Y., Cai, K., McBurney, P., Sklar, E., Parsons, S.: Using argumentation to reason about trust and belief. Journal of Logic and Computation (to appear, 2011)

38. Tang, Y., Cai, K., Sklar, E., McBurney, P., Parsons, S.: A system of argumentation for reasoning about trust. In: Proceedings of the 8th European Workshop on Multi-Agent Systems, Paris, France (December 2010)
39. Teng, C.-Y., Lauterbach, D., Adamic, L.: I rate you. You rate me. Should we do so publicly? In: Proceedings of the 3rd Workshop on Online Social Networks, Boston (June 2010)
40. Victor, P., Cornelis, C., De Cock, M., Pinheiro da Silva, P.: Towards a provenance-preserving trust model in agent networks. In: Proceedings of the Workshop on Models of Trust for the Web (2006)
41. Wang, X., Govindan, K., Mohapatra, P.: Provenance-based information trustworthiness evaluation in multihop networks. In: Proceedings of the 53rd Annual IEEE Global Communications Conference, Miami, FL (December 2010)
42. Ye, S., Wu, S.F.: Measuring message propagation and social influence on twitter.com. In: Bolc, L., Makowski, M., Wierzbicki, A. (eds.) SocInfo 2010. LNCS, vol. 6430, pp. 216–231. Springer, Heidelberg (2010)
43. Yin, X., Han, J., Yu, P.S.: Truth discovery with multiple conflicting information providers on the web. In: Proceedings of the Conference on Knowledge and Data Discovery (2007)
44. Zhong, S., Chen, J., Yang, Y.R.: Sprite: A simple cheat-proof, credit-based system for mobile ad-hoc networks. In: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (2003)

Transitivity and Propagation of Trust in Information Sources: An Analysis in Modal Logic

Robert Demolombe

Institut de Recherche en Informatique de Toulouse, France
robert.demolombe@orange.fr

Abstract. The paper is about trust in information sources in the context of Multi Agents Systems and it is focused on information and trust propagation. Trust definition is inspired from Cognitive Science and it is seen as a truster's belief in some trustee's properties which are called: sincerity, competence, vigilance, cooperativity, validity and completeness. These definitions are formalized in Modal Logic and it is shown that even if trust, in that sense, is not transitive, we can find interesting sufficient conditions based on trust that guarantee that the truth of an information is propagated along a chain of information sources.

1 Introduction

In the context of Multi Agent Systems trust plays a quite significant role with respect to interactions between agents. That is because in many applications agents only have a partial knowledge of the agents they have to interact with. For instance, in the context of electronic commerce the buyer has to trust the seller in the properties of the goods he wants to buy and the seller has to trust the buyer in the fact that he will be paid for the goods he wants to sell.

In the context of information retrieval the agents have to trust the information sources in the fact that the transmitted information is true or in the fact that information sources are competent or sincere or both competent and sincere. Moreover, in many cases the information transmitted by the information sources is not supported by direct observations but by information reported by other information sources. Then, information may be propagated along a chain of information sources and it is definitely not easy to decide wether we can trust such or such information source. That is the main issue which is investigated in this paper.

The analysis of this problem is not easy because the notion of trust itself is not simple. Even if some authors do not analyse this notion in detail and define trust by a relationship of the kind: agent i trusts agent j, we consider here that this is an over simplification and that trust is a complex mental attitude as people in the field of Cognitive Science [4,5] or in Philosophy [1,13,12] have pointed out.

In this approach concepts definitions and reasoning rules related to trust have to be carefully defined. That is why a formalism based on mathematical logic is

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 13–28, 2011.

adopted in this work in order to investigate trust transitivity and propagation. Thanks to this formalism it will be shown that even if trust is not transitive, other properties related to information propagation and to trust propagation hold.

The paper is structured as follows. In section 2 are informally compared several trust definitions and the definition adopted in the rest of the paper is presented. The logical framework which is used for formal definitions and reasoning rules is defined in section 3. This formalism is used in section 4 to define different kinds of trust. Since properties that guarantee the truth of transmitted information or of trust propagation are quite complex a case study is presented in section 5 to give to the reader an intuitive understanding of these properties. The main results of the paper are presented in section 6 in terms of theorems. In section 7 our work is compared with related works and the last section presents our conclusions.

2 Informal Trust Definitions

As mentioned above there are many different trust definitions. In [4,5] Castelfranchi and Falcone mention that they have found around 60 different definitions. However, their own definition is one of the most popular in this field. It is informally presented below.

The main feature is that trust is a truster's belief about some trustee's properties. This belief is motivated by truster's goal and the properties he ascribes to the trustee are such that by doing some action α the trustee will reach the truster's goal. More precisely trust is defined by the fact that the truster has the following beliefs:

- truster's goal is to reach a situation where the proposition ϕ holds
- the action α has the effect that ϕ holds
- the trustee has the ability and opportunity to do the action α
- the trustee has the intention to do α

This approach has been expressed by Lorini and Demolombe in Modal Logic in [14,15] and it has been shown that a logical consequence of truster's beliefs is that the truster believes that his goal will be reached after α performance by the trustee.

In this paper the trust definition we have adopted has some similarities with the above definition. However, there are significant differences. The first one is that truster's goal is not involved in trust definition itself even if this goal may be a motivation for the truster to adopt some beliefs about the trustee. The second one is that the properties ascribed to the trustee are all in a conditional form. That is, the truster believes that some fact entails another fact, and at least one of these facts is about a trustee's property.

In the specific context of trust in information sources these facts may be that (1) the trustee has informed the truster about some proposition or that (2) the trustee believes that some proposition is true or the fact that (3) it is the case that

some proposition is true. The combination of these three kinds of facts in terms of the entailment relationship leads to six different kinds of trustee's properties and six kinds of trust in the trustee's properties [7,8,9]. These properties are presented below.

Trust in sincerity: the truster believes that if he is informed by the trustee about some proposition, then the trustee believes that this proposition is true.

Trust in competence: the truster believes that if the trustee believes that some proposition is true, then this proposition is true.

Trust in vigilance: the truster believes that if some proposition is true, then the trustee believes that this proposition is true.

Trust in cooperativity: the truster believes that if the trustee believes that some proposition is true, then he is informed by the trustee about this proposition.

Trust in validity: the truster believes that if he is informed by the trustee about some proposition, then this proposition is true.

Trust in completeness: the truster believes that if some proposition is true, then he is informed by the trustee about this proposition.

We can informally see how these trust definitions and the definition proposed by Castelfranchi and Falcone are related. Let us assume, for example, that the truster's goal is to be informed about the truth of some proposition ϕ. Then, if the truster trusts the trustee in his completeness about ϕ and about $\neg\phi$, the truster believes that if ϕ is true he will be informed by the trustee about the fact ϕ is true and if ϕ is false, he will be informed by the trustee about the fact that ϕ is false. If, in addition, the truster trusts the trustee in his validity about ϕ and $\neg\phi$, from the fact that he has been informed that ϕ is true, he will believe that ϕ is true, and, in the same way, if he has been informed that ϕ is false, he will believe that ϕ is false. Therefore, whatever ϕ is true or false, the truster goal will be achieved.

To express these definitions and derivations in formal terms we present in the next section an appropriate logical framework.

3 Logical Framework

The logical framework is defined by its formal language L and its axiomatics.

We have adopted the following notations:

$ATOM$: set of atomic propositions denoted by p, q, r...

$AGENT$: set of agents denoted by i, j, k, l,...

The language L is the set of formulas defined by the following BNF:

$$\phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid Bel_i\phi \mid Inf_{j,i}\phi$$

where p ranges over $ATOM$ and i and j range over $AGENT$.

The intuitive meaning of the modal operators is:

- $Bel_i\phi$: the agent i believes that ϕ is the case.
- $Inf_{j,i}\phi$: the agent j has informed the agent i about ϕ.

The axiomatics of the logic is the axiomatics of a Propositional multi Modal Logic (see Chellas in [6]).

In addition to the axiomatics of Classical Propositional Calculus we have the following axiom schemas an inference rules.

(K) $Bel_i(\phi \rightarrow \psi) \rightarrow (Bel_i\phi \rightarrow Bel_i\psi)$
(D) $\neg(Bel_i\phi \wedge Bel_i\neg\phi)$
(Nec) If $\vdash \phi$, then $\vdash Bel_i\phi$

For the modal operator $Inf_{j,i}$ we have the inference rule and axiom schemas:

(EQV) If $\vdash \phi \leftrightarrow \psi$, then $\vdash Inf_{j,i}\phi \leftrightarrow Inf_{j,i}\psi$
(CONJ) $Inf_{j,i}\phi \wedge Inf_{j,i}\psi \rightarrow Inf_{j,i}(\phi \wedge \psi)$
(OBS) $Inf_{j,i}\phi \rightarrow Bel_iInf_{j,i}\phi$
(OBS') $\neg Inf_{j,i}\phi \rightarrow Bel_i\neg Inf_{j,i}\phi$

According to Chellas's terminology, modalities of the kind Bel_i obey a normal system KD and modalities of the kind $Inf_{j,i}$ obey a particular kind of classical system which is not monotonic. Axioms (OBS) and (OBS') show how these two kinds of modalities interact.

The justification of (OBS) and (OBS') is that it is assumed that if an agent j informs or does not inform an agent i about ϕ, then i is aware of this fact. Roughly speaking, it is assumed that we have perfect communication channels.

The adoption of axiom schema (CONJ) is questionable. It can be adopted or rejected depending on the fact that performance of both actions $Inf_{j,i}\phi$ and $Inf_{j,i}\psi$, on one hand, and performance of the action $Inf_{j,i}(\phi \wedge \psi)$, on the other hand, have "equivalent" effects with respect to the issues we want to analyze. If an effect of the action $Inf_{j,i}\phi$ is denoted by $Effect(\phi)$, this "equivalence" holds as long as we have: (CLOS) $Effect(\phi) \wedge Effect(\psi) \rightarrow Effect(\phi \wedge \psi)$. In the context of our trust definitions the effects we are interested in, which are mentioned in the following section, are either $Effect(\phi) \stackrel{\text{def}}{=} \phi$ or $Effect(\phi) \stackrel{\text{def}}{=} Bel_j\phi$. For both of them the property of closure under conjonction (CLOS) holds. Therefore, the axiom schema (CONJ) does not lead to any counter intuitive consequence in our context. Nevertheless, it is worth noting that it is not needed to prove any theorem in this paper and, then, it could be rejected as well. Last comment: if in an other context (for example, if we are interested in resources that have been used for communication) we have to count the number of communication actions that have been performed, then the (CLOS) property does not hold any more and (CONJ) should be rejected.

We have **not** accepted the following inference rule:

(CLOS) If $\vdash \phi \rightarrow \psi$, then $\vdash Inf_{j,i}\phi \rightarrow Inf_{j,i}\psi$

The reason is that it could lead to consequences that are counter intuitive. Let's consider, for example, a situation where, if j informs i about $p \vee q$, then $p \vee q$ is true, while the fact that j informs i about p does not entail that p is true. That could be the case, for instance, if p means that it is snowing and q means that

it is raining, because there are much more situations where it is raining than situations where it is snowing.

If we accept (CONS), in a situation where j has informed i about p, from (CONS) we could infer that it is also true that j has informed i about $p \vee q$ and that $p \vee q$ is true.

4 Formal Trust Definitions

The different kinds of trust which have been informally presented in section 2 are formally represented in this framework as follows.

Sincerity.
$Sinc(j, i, \phi)$: if agent j has informed agent i about ϕ, then j believes ϕ.
$Sinc(j, i, \phi) \stackrel{\text{def}}{=} Inf_{j,i}\phi \rightarrow Bel_j\phi$
$TrustSinc(i, j, \phi)$: agent i trusts agent j in his sincerity about ϕ.
$TrustSinc(i, j, \phi) \stackrel{\text{def}}{=} Bel_i(Inf_{j,i}\phi \rightarrow Bel_j\phi)$
Competence.
$Comp(j, \phi)$: if agent j believes ϕ, then ϕ is true.
$Comp(j, \phi) \stackrel{\text{def}}{=} Bel_j\phi \rightarrow \phi$
$TrustComp(i, j, \phi)$: agent i trusts agent j in his competence about ϕ.
$TrustComp(i, j, \phi) \stackrel{\text{def}}{=} Bel_i(Bel_j\phi \rightarrow \phi)$
The "dual" of these properties are formally represented below.
Cooperativity.
$Coop(j, i, \phi)$: if agent j believes ϕ, then j informs agent i about ϕ.
$Coop(j, i, \phi) \stackrel{\text{def}}{=} Bel_j\phi \rightarrow Inf_{j,i}\phi$
$TrustCoop(i, j, \phi)$: agent i trusts agent j in his cooperativity about ϕ.
$TrustCoop(i, j, \phi) \stackrel{\text{def}}{=} Bel_i(Bel_j\phi \rightarrow Inf_{j,i}\phi)$
Vigilance.
$Vigi(j, \phi)$: if ϕ is true, then agent j believes ϕ.
$Vigi(j, \phi) \stackrel{\text{def}}{=} \phi \rightarrow Bel_j\phi$
$TrustVigi(i, j, \phi)$: agent i trusts agent j in his vigilance about ϕ.
$TrustVigi(i, j, \phi) \stackrel{\text{def}}{=} Bel_i(\phi \rightarrow Bel_j\phi)$
We also have the following less specific kinds of trust.
Validity.
$Val(j, i, \phi)$: if agent j has informed agent i about ϕ, then ϕ is true.
$Val(j, i, \phi) \stackrel{\text{def}}{=} Inf_{j,i}\phi \rightarrow \phi$
$TrustVal(i, j, \phi)$: agent i trusts agent j in his validity about ϕ.
$TrustVal(i, j, \phi) \stackrel{\text{def}}{=} Bel_i(Inf_{j,i}\phi \rightarrow \phi)$
The "dual" of validity is completeness.
Completeness.
$Cmp(j, \phi)$: if ϕ is true, then agent j informs i about ϕ.
$Cmp(j, \phi) \stackrel{\text{def}}{=} \phi \rightarrow Inf_{j,i}\phi$
$TrustCmp(i, j, \phi)$: agent i trusts agent j in his completeness about ϕ.

$TrustCmp(i,j,\phi) \stackrel{\text{def}}{=} Bel_i(\phi \rightarrow Inf_{j,i}\phi)$

Some of these properties are not independent. We can easily show that we have:

(V) $\vdash TrustSinc(i,j,\phi) \wedge TrustComp(i,j,\phi) \rightarrow TrustVal(i,j,\phi)$
(C) $\vdash TrustVigi(i,j,\phi) \wedge TrustCoop(i,j,\phi) \rightarrow TrustCmp(i,j,\phi)$

These definitions can be used to characterize the effects of the fact that an agent has informed or does not have informed another agent, depending on different kinds of assumption about the trust relationship between these two agents. We have the following properties.

(E1) $\vdash TrustSinc(i,j,\phi) \rightarrow (Inf_{j,i}\phi \rightarrow Bel_i Bel_j \phi)$
(E2) $\vdash TrustVal(i,j,\phi) \rightarrow (Inf_{j,i}\phi \rightarrow Bel_i \phi)$
(E3) $\vdash TrustCoop(i,j,\phi) \rightarrow (\neg Inf_{j,i}\phi \rightarrow Bel_i \neg Bel_j \phi)$
(E4) $\vdash TrustCmp(i,j,\phi) \rightarrow (\neg Inf_{j,i}\phi \rightarrow Bel_i \neg \phi)$

Property (E2) (resp. (E4)) shows sufficient conditions about trust that guarantee that performing (resp. not performing) the action $Inf_{j,i}\phi$ has the effect that i believes that ϕ is true (resp. false).

5 From Case Studies to Generalization

Before to show in formal terms in which situations information and/or trust can be propagated along a chain of information sources a case study is presented in order to give an intuitive understanding of the general results which are presented in the next section.

John wants to invest money in stocks and bonds. Peter who is a journalist in the field of finance has told to John that Franck, who is a trader, told him that the stock value of the company AXB is going to increase. John trusts Peter and Peter trusts Franck. The question is: *can we infer that John trusts Franck?*.

It is tempting to answer: "*yes*". However, to answer this question we have to make more precise the kind of trust we have in mind.

Let us assume that we are thinking to trust in validity. Then, if the sentence: "the stock value of the company AXB is going to increase" is denoted by p, the sentence "*John trusts Peter*" means: "*John believes that if Peter tells to John that p, then p is true*", the sentence "*Peter trusts the Franck*" means: "*Peter believes that if Franck tells to Peter that p, then p is true*", and the sentence "*John trusts Franck*" means: "*John believes that if Franck tells to John that p, then p is true*".

According to this definition of trust, from the fact that John trusts Peter (i.e. $TrustVal(John, Peter, p)$) and Peter trusts Franck (i.e. $TrustVal(Peter, Franck, p)$)) we cannot infer that John trusts Franck (i.e. $TrustVal(John, Franck, p)$). The first reason is that John is not necessarily aware of what believes Peter, and in particular he may not be aware of the fact that Peter trusts Franck. In formal terms we may have: $\neg Bel_{John}(TrustVal(Peter, Franck, p))$.

Even if it is assumed that Peter has told to John that he trusts Franck (i.e. $Inf_{Peter,John}(TrustVal(Peter, Franck, p))$) it is not necessarily the case that

John believes that what Peter told him is true. In other words, it may be that John does not trust Peter in his validity about the fact that Peter trusts Franck in his validity about p (i.e. $\neg TrustVal(John, Peter, Val(Peter, Franck, p))$).

Indeed, it may be that John trusts Peter as a reliable information source about the value of the stocks but that he does not trust Peter as an evaluator of other information sources.

The reason why John does not trust Peter as an evaluator may be, for example, that John believes that Peter has the capacity to evaluate the trader's competence about p but Peter does not have the capacity to evaluate the trader's sincerity about p and it could happen that the trader Franck tells to Peter that the stock value of the company AXB is going to increase while Franck believes that it is going to decrease.

This simple example shows that trust in informations sources' validity, as it has been defined, is not transitive. In formal terms, it can easily be checked that the set of sentences: $TrustVal(John, Peter, p)$, $TrustVal(John, Franck, p)$ and $\neg TrustVal(Peter, Franck, p)$ is consistent.

Nevertheless, we can try to find if there are other kinds of trust relative to information transmission that "guarantees" the truth of the information which is transmitted.

Let us use q to denote the sentence: "Franck has told p to Peter and, if Franck has told p to Peter, then p is true".

Now, it is assumed that John trusts Peter in his validity about q.

If John is aware of the fact that the trader Franck has told p to Peter and of the fact that Peter has told q to him, we can infer that John believes that p is true.

To check the validity of this derivation and to understand how it can be generalized to an unlimited number of information sources we introduce the following notations: i: John, i_1: Peter, i_2: Franck and:

$q \stackrel{\text{def}}{=} Inf_{i_2,i_1} p \wedge (Inf_{i_2,i_1} p \rightarrow p)$

In that example we have: $Inf_{i_2,i_1} p$, $Inf_{i_1,i} q$ and $Bel_i(Inf_{i_1,i} q \rightarrow q)$.

If it is assumed that i (John) is aware of what i_1 (Peter) told him q, we also have: $Bel_i Inf_{i_1,i} q$. Then, we can infer $Bel_i q$ and, by definition of q, we have: $Bel_i(Inf_{i_2,i_1} p \wedge (Inf_{i_2,i_1} p \rightarrow p))$, which entails $Bel_i p$.

Let us consider now a variant of the previous examples where some agents play the role of information sources and others play the role of evaluators of the information sources. Let us assume, for example, that Franck has told to Peter that if another agent, who is called Carlo, tells to John that p, then p is true. The reason why Franck ascribes this property to Carlo may be, for example, that he knows that Carlo is a manager of the company AXB. Let us assume in addition that Jack, who is a consultant in stocks and bonds, has told to John that Franck is a reliable evaluator of Carlo, and John believes that Jack is a reliable evaluator of Franck. Can we infer in that case that John trusts Carlo in p and that John believes p?

To find a formal answer to this question we adopt the following notations: e_2: Jack, j: Carlo and:

$p' \stackrel{\text{def}}{=} Inf_{j,i}p \rightarrow p$
$r \stackrel{\text{def}}{=} Inf_{i_2,i_1}p' \rightarrow p'$
$s \stackrel{\text{def}}{=} (Inf_{i_2,i_1}p') \wedge (Inf_{e_2,i_1}r) \wedge (Inf_{e_2,i_1}r \rightarrow r)$

It is assumed that John trusts Peter in his validity about s and John is aware of the fact that Peter has told him s and Carlo has told him p. Then, we have:

$Bel_i Inf_{j,i}p$, $Bel_i Inf_{i_1,i}s$ and $Bel_i(Inf_{i_1,i}s \rightarrow s)$.

From these assumptions we can infer: $Bel_i(s)$, which, by definition of s means: $Bel_i((Inf_{i_2,i_1}p') \wedge (Inf_{e_2,i_1}r) \wedge (Inf_{e_2,i_1}r \rightarrow r))$, which entails: $Bel_i((Inf_{i_2,i_1}p') \wedge r)$, which, by definition of r, means: $Bel_i((Inf_{i_2,i_1}p') \wedge (Inf_{i_2,i_1}p' \rightarrow p'))$, which entails: $Bel_i p'$, which, by definition of p', means: $Bel_i(Inf_{j,i}p \rightarrow p)$, and from $Bel_i Inf_{j,i}p$ we can infer: $Bel_i p$. That gives a positive answer to the above question.

Before to present a general analysis of trust and information propagation we introduce the following general notations.

In the case of the second example the information transmitted by an agent i_k to another agent i_{k-1} is denoted by $\Phi_{k,k-1}$. Then, we have: $\Phi_{2,1} = p$ and $\Phi_{1,0} = Inf_{i_2,i_1}\Phi_{2,1} \wedge Val(i_2, i_1, \Phi_{2,1})$. Roughly speaking we can say that the information transmitted by i_1 to i_0 represents the fact that i_1 has received the piece of information represented by $\Phi_{2,1}$ from i_2 (i.e. $Inf_{i_2,i_1}\Phi_{2,1}$) and the fact that i_2 is a valid information source for i_1 with respect to $\Phi_{2,1}$ (i.e. $Val(i_2, i_1, \Phi_{2,1})$).

If we have three information sources: i_3, i_2 and i_1, we have (see figure 1):

$\Phi_{3,2} = p$, $\Phi_{2,1} = Inf_{i_3,i_2}\Phi_{3,2} \wedge Val(i_3, i_2, \Phi_{3,2})$ and $\Phi_{1,0} = Inf_{i_2,i_1}\Phi_{2,1} \wedge Val(i_2, i_1, \Phi_{2,1})$

The fact that i trusts i_1 in his validity about $\Phi_{1,0}$ is represented by: $Bel_i(Inf_{i_1,i}\Phi_{1,0} \rightarrow \Phi_{1,0})$.

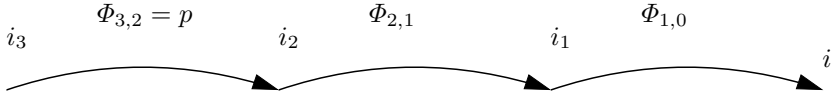


Fig. 1. Chain of information sources

For the third example $\Psi_{k,k-1}$ is used to denote the information transmitted by an agent i_k to another agent i_{k-1}. Then, we have: $\Psi_{2,1} = Val(j, i, p)$ and $\Psi_{1,0} = (Inf_{i_2,i_1}\Psi_{2,1}) \wedge (Inf_{e_2,i_1}Val(i_2, i_1, \Psi_{2,1})) \wedge Val(e_2, i_1, Val(i_2, i_1, \Psi_{2,1}))$. In that case we can say that the information transmitted by i_1 to i represents the fact that i_1 has received the piece of information $\Psi_{2,1}$ from i_2 (i.e. $Inf_{i_2,i_1}\Psi_{2,1}$) and i_1 has received from the evaluator e_2 a piece of information which means that i_2 is a valid information source for i_1 with respect to $\Psi_{2,1}$ (i.e. $Inf_{e_2,i_1}Val(i_2, i_1, \Psi_{2,1})$) and the fact that the evaluator e_2 is a valid information source for i_1 with respect to the fact that i_2 is a valid information source for i_1 with respect to $\Psi_{2,1}$ (i.e. $Val(e_2, i_1, Val(i_2, i_1, \Psi_{2,1}))$).

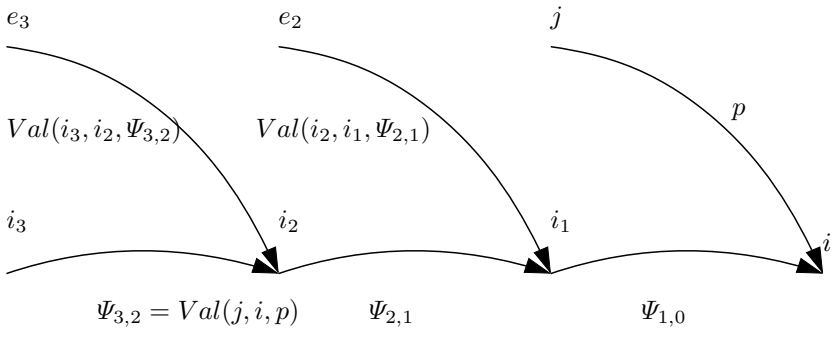


Fig. 2. Chain of information sources and of of their evaluators

In the case of a generalization of the third example to three information sources, the information transmitted by an information source to another one is represented by the following sentences (see figure 2).
$\Psi_{3,2} = Val(j, i, p)$, $\Psi_{2,1} = (Inf_{i_3,i_2}\Psi_{3,2}) \wedge (Inf_{e_3,i_2}Val(i_3, i_2, \Psi_{3,2})) \wedge Val(e_3, i_2, \Psi_{3,2})$ and $\Psi_{1,0} = (Inf_{i_2,i_1}\Psi_{2,1}) \wedge (Inf_{e_2,i_1}Val(i_2, i_1, \Psi_{2,1})) \wedge Val(e_2, i_1, \Psi_{2,1})$

The fact that i trusts i_1 in his validity about $\Psi_{1,0}$ is represented by:
$Bel_i(Inf_{i_1,i}\Psi_{1,0} \rightarrow \Psi_{1,0})$.

6 Trust in Information Sources Propagation

In this section general results about information and trust propagation are presented in the form of theorems. We first prove the following Lemma 1.

Lemma 1. *If ϕ is in L and i and j are in $AGENT$, we have:*
$\vdash Inf_{j,i}\phi \wedge Bel_i Val(j, i, \phi) \rightarrow Bel_i\phi$
$\vdash \neg Inf_{j,i}\phi \wedge Bel_i Cmp(j, i, \phi) \rightarrow Bel_i\neg\phi$

Proof. *This Lemma is a direct consequence of (OBS), (OBS') and of validity and completeness definitions.* **QED.**

The Theorem 1 shows a property which allows us to infer from an assumption about the fact that i trusts j in his competence about k's competence and i believes that j trusts k in his competence a conclusion about the fact i trusts k in his competence. However, this is not a property of transitivity because the assumptions do not have exactly the same form as the conclusion.

Theorem 1. *If ϕ is in L and i, j and k are in $AGENT$, we have:*
$\vdash TrustComp(i, j, Comp(k, \phi)) \wedge Bel_i TrustComp(j, k, \phi)$
$\rightarrow TrustComp(i, k, \phi)$

Proof. *From $TrustComp(i, j, Comp(k, \phi))$ definition we have:*
(1) $Bel_i(Bel_j Comp(k, \phi) \rightarrow Comp(k, \phi))$
From $Bel_i TrustComp(j, k, \phi)$ we have: (2) $Bel_i Bel_j Comp(k, \phi)$.

Then, from (1) and (2) we have: (3) $Bel_i Comp(k, \phi)$.
By definition of $TrustComp$, *(3) is:* $TrustComp(i, k, \phi)$. **QED.**

The Theorem 2 shows that if agent i_1 has informed agent i about the fact that agent i_2 has informed agent i_1 about the fact that...agent i_n has informed agent i_{n-1} about ϕ, and agent i believes that these agents are valid for what they have told each other, then agent i believes ϕ.

This theorem guarantees some kind of propagation of the proposition ϕ through a chain of information sources from i_n to i, provided agent i believes that they are all valid information sources for what they have told to the next information source in that chain.

Theorem 2. *If* ϕ *is in* L *and* i, i_1, i_2, ... ,i_n *are in* $AGENT$ *and we adopt the following notations:*

$\Phi_{n,n-1} \stackrel{def}{=} \phi$ *and for* k *in* $[1, n-1]$: $\Phi_{k,k-1} \stackrel{def}{=} Inf_{i_{k+1},i_k} \Phi_{k+1,k}$
we have:

$\vdash (Inf_{i_1,i} \Phi_{1,0}) \wedge Bel_i(Val(i_1, i, \Phi_{1,0}) \wedge Val(i_2, i_1, \Phi_{2,1}) \wedge \ldots \wedge Val(i_n, i_{n-1}, \Phi_{n,n-1})) \rightarrow Bel_i \phi$

Proof. *From* Val *definition definition* $Val(i_1, i, \Phi_{1,0}) \wedge Val(i_2, i_1, \Phi_{2,1})$ *is:* $(Inf_{i_1,i} \Phi_{1,0} \rightarrow \Phi_{1,0}) \wedge (Inf_{i_2,i_1} \Phi_{2,1} \rightarrow \Phi_{2,1})$, *and from* $\Phi_{k,k-1}$ *definition* $\Phi_{1,0}$ *is* $Inf_{i_2,i_1} \Phi_{2,1}$. *Then,* $Val(i_1, i, \Phi_{1,0}) \wedge Val(i_2, i_1, \Phi_{2,1})$ *is logically equivalent to:* $Inf_{i_1,i} \Phi_{1,0} \rightarrow \Phi_{2,1}$, *and we can easily prove by induction that* $Val(i_1, i, \Phi_{1,0}) \wedge Val(i_2, i_1, \Phi_{2,1}) \wedge \ldots \wedge Val(i_n, i_{n-1}, \Phi_{n,n-1})$ *is logically equivalent to* $Inf_{i_1,i} \Phi_{1,0} \rightarrow \Phi_{n,n-1}$.

Then, from $Bel_i(Val(i_1, i, \Phi_{1,0}) \wedge Val(i_2, i_1, \Phi_{2,1}) \wedge \ldots \wedge Val(i_n, i_{n-1}, \Phi_{n,n-1}))$ *we infer that: (1)* $Bel_i(Inf_{i_1,i} \Phi_{1,0}) \rightarrow \Phi_{n,n-1})$, *and from (1),* $Inf_{i_1,i} \Phi_{1,0}$ *and Lemma 1 we infer:* $Bel_i(\Phi_{n,n-1})$, *that is* $Bel_i \phi$. **QED.**

The Theorem 3 shows that if it not the case that agent i_1 has informed agent i about the fact that agent i_2 has informed agent i_1 about the fact that...agent i_n has informed agent i_{n-1} about ϕ, and agent i believes that these agents are complete for what they might have told each other, then agent i believes $\neg\phi$.

This theorem can be seen as the dual of Theorem 2 where completeness plays a similar role as validity.

Theorem 3. *If* ϕ *is in* L *and* i, i_1, i_2, ... ,i_n *are in* $AGENT$ *and we adopt the following notations:*

$\Phi_{n,n-1} \stackrel{def}{=} \phi$ *and for* k *in* $[1, n-1]$: $\Phi_{k,k-1} \stackrel{def}{=} Inf_{i_{k+1},i_k} \Phi_{k+1,k}$
we have:

$\vdash (\neg Inf_{i_1,i} \Phi_{1,0}) \wedge Bel_i(Cmp(i_1, i, \Phi_{1,0}) \wedge Cmp(i_2, i_1, \Phi_{2,1}) \wedge \ldots$
$\wedge Cmp(i_n, i_{n-1}, \Phi_{n,n-1}) \rightarrow Bel_i \neg\phi$

Proof. *proof is very similar as the proof of Theorem 2.*

The difference between the assumptions in Theorem 4 and those in the Theorem 2 is that each agent i_k informs i_{k-1} about the validity of agent i_{k+1} for the information $\Phi_{k+1,k}$ transmitted by i_{k+1} to him and agent i only trusts agent i_1 in his validity. Roughly speaking, here it is not required that i knows the agents i_2, i_3, ... ,i_n.

Theorem 4. *If ϕ is in L and i, i_1, i_2, ... ,i_n are in AGENT and we adopt the following notations:*

$\Phi_{n,n-1} \stackrel{def}{=} \phi$ *and*

for k in $[1, n-1]$: $\Phi_{k,k-1} \stackrel{def}{=} (Inf_{i_{k+1},i_k}\Phi_{k+1,k}) \wedge Val(i_{k+1}, i_k, \Phi_{k+1,k})$

we have:

$\vdash (Inf_{i_1,i}\Phi_{1,0}) \wedge TrustVal(i, i_1, \Phi_{1,0}) \rightarrow Bel_i\phi$

Proof. *Let us call (H) the formula:* $(Inf_{i_1,i}\Phi_{1,0}) \wedge TrustVal(i, i_1, \Phi_{1,0})$.

We prove by induction that for every l in $[1, n]$: (H) entails $Bel_i\Phi_{l,l-1}$.

For $l = 1$, from (H) definition and Lemma 1 we have: $Bel_i\Phi_{1,0}$.

Induction hypothesis: (H) entails $Bel_i\Phi_{l,l-1}$.

From $\Phi_{l,l-1}$ definition and the induction hypothesis (H) entails:

(1) $Bel_iInf_{i_{l+1},i_l}\Phi_{l+1,l}$ and (2) $Bel_iVal(i_{l+1}, i_l, \Phi_{l+1,l})$

From (1) and (2) we have: $Bel_i\Phi_{l+1,l}$. Therefore, (H) entails $Bel_i\Phi_{l+1,l}$.

Then for every l in $[1, n]$ we have: $Bel_i\Phi_{l,l-1}$, and in particular we have: $Bel_i\Phi_{n,n-1}$ which is, by definition of $\Phi_{n,n-1}$, $Bel_i\phi$. **QED.**

The following Theorem 5 is the dual of Theorem 4. The main difference is that here the meaning of the proposition $\Phi_{k,k-1}$ is that if i_{k+1} is a complete information source, then i_{k+1} informs i_k about $\Phi_{k+1,k}$.

Theorem 5. *If ϕ is in L and i, i_1, i_2, ... ,i_n are in AGENT and we adopt the following notations:*

$\Phi_{n,n-1} \stackrel{def}{=} \phi$ *and*

for k in $[1, n-1]$: $\Phi_{k,k-1} \stackrel{def}{=} Cmp(i_{k+1}, i_k, \Phi_{k+1,k}) \rightarrow Inf_{i_{k+1},i_k}\Phi_{k+1,k}$

we have:

$\vdash (\neg Inf_{i_1,i}\Phi_{1,0}) \wedge TrustCmp(i, i_1, i, \Phi_{1,0}) \rightarrow Bel_i\neg\phi$

Proof. *The proof is very similar as the proof of Theorem 4.*

If we call (H) the formula: $(\neg Inf_{i_1,i}\Phi_{1,0}) \wedge TrustCmp(i, i_1, \Phi_{1,0})$ *we prove by induction that (H) entails $Bel_i\neg\Phi_{l,l-1}$.*

We just have to notice that $\neg\Phi_{l,l-1}$ is equivalent to: $Cmp(i_{l+1}, i_l, \Phi_{l+1,l}) \wedge \neg Inf_{i_{l+1},i_l}\Phi_{l+1,l}$, which entails $\neg\Phi_{l+1,l}$. **QED.**

The following Theorem 6 has a similar meaning as Theorem 4.

The difference is that for each information source i_k there is another information source e_k who plays the role of an evaluator of i_k's validity and e_k himself is considered by i_{k-1} as a valid information source for this evaluation.

Theorem 6. *If ϕ is in L and j, i, i_1, i_2, ... ,i_n are in AGENT and we adopt the following notations:*

$\Psi_{n,n-1} \stackrel{def}{=} \phi$ *and*

for k in $[1, n-1]$: $\Psi_{k,k-1} \stackrel{def}{=} (Inf_{i_{k+1},i_k}\Psi_{k+1,k}) \wedge (Inf_{e_{k+1},i_k}Val(i_{k+1}, i_k, \Psi_{k+1,k})) \wedge Val(e_{k+1}, i_k, Val(i_{k+1}, i_k, \Psi_{k+1,k}))$

we have:

$\vdash (Inf_{i_1,i}\Psi_{1,0}) \wedge TrustVal(i, i_1, \Psi_{1,0}) \rightarrow Bel_i\phi$

Proof. *The proof is similar as the proof of Theorem 4.*

Let us call (H) the formula $(Inf_{i_1,i}\Psi_{1,0}) \wedge TrustVal(i_1, i, \Psi_{1,0})$.

We prove by induction on l *that for every* l *in* $[1, n]$ *(H) entails* $Bel_i\Psi_{l,l-1}$.

For $l = 1$, *from* $Bel_iVal(i_1, i, \Psi_{1,0})$ *and* $Inf_{i_1,i}\Psi_{1,0}$, *by Lemma 1 we have:* $Bel_i\Psi_{1,0}$.

Induction hypothesis: (H) entails $Bel_i\Psi_{l,l-1}$.

From (H) and $\Psi_{l,l-1}$ *definition we infer: (1)* $Bel_i(Inf_{e_{l+1},i_l}Val(i_{l+1}, i_k, \Psi_{l+1,l}))$ *and (2)* $Bel_iVal(e_{l+1}, i_l, Val(i_{l+1}, i_l, \Psi_{l+1,l}))$.

Then, from (1) and (2), we infer: (3) $Bel_i(Val(i_{l+1}, i_l, \Psi_{l+1,l}))$.

From (H) and $\Psi_{l,l-1}$ *definition we also infer: (4)* $Bel_i(Inf_{i_{l+1},i_l}\Psi_{l+1,l})$.

Then, from (3) and (4), we infer: $Bel_i\Psi_{l+1,l}$. *Therefore, (H) entails* $Bel_i\Psi_{l+1,l}$.

Then, by definition of $\Psi_{n,n-1}$, *(H) entails* $Bel_i\phi$. **QED.**

The following Theorem 7 is the dual of Theorem 6 in the sense that the evaluator e_{k+1} is an evaluator of i_{k+1}'s completeness instead of i_{k+1}'s validity.

Theorem 7. *If* ϕ *is in* L *and* j, i, i_1, i_2, ... ,i_n *are in* $AGENT$ *and we adopt the following notations:*

$\Psi_{n,n-1} \stackrel{def}{=} \phi$ *and*

for k *in* $[1, n-1]$: $\Psi_{k,k-1} \stackrel{def}{=} ((Inf_{e_{k+1},i_k}Cmp(i_{k+1}, i_k, \Psi_{k+1,k})) \wedge Val(e_{k+1}, i_k, Cmp(i_{k+1}, i_k, \Psi_{k+1,k}))) \rightarrow (Inf_{i_{k+1},i_k}\Psi_{k+1,k})$

we have:

$\vdash (\neg Inf_{i_1,i}\Psi_{1,0}) \wedge TrustCmp(i, i_1, \Psi_{1,0}) \rightarrow Bel_i\neg\phi$

Proof. *The proof is similar as the proof of Theorem 6.*

Here (H) is $(\neg Inf_{i_1,i}\Psi_{1,0}) \wedge TrustCmp(i, i_1, \Psi_{1,0})$ *and we prove by induction that (H) entails* $Bel_i\neg\Psi_{n,n-1}$.

Indeed, if it assumed that (H) entails $Bel_i\neg\Psi_{l,l-1}$, *from* $\Psi_{l,l-1}$ *definition* $\neg\Psi_{l,l-1}$ *is equivalent to (1)* $(Inf_{e_{l+1},i_l}Cmp(i_{l+1}, i_l, \Psi_{l+1,l})) \wedge Val(e_{l+1}, i_l, Cmp(i_{l+1}, i_l, \Psi_{l+1,l})) \wedge \neg(Inf_{i_{l+1},i_l}\Psi_{l+1,l})$. *We can easily show that (1) entails: (2)* $Cmp(i_{l+1}, i_l, \Psi_{l+1,l})$ *and (3)* $\neg(Inf_{i_{l+1},i_l}\Psi_{l+1,l})$. *Since (2) and (3) entail (4)* $\neg\Psi_{l+1,l}$, *we have* $Bel_i\neg\Psi_{l+1,l}$. **QED.**

In the following Theorem 8 the assumptions are similar as the assumptions in Theorem 6. The first difference is that here each information source i_k trusts the evaluator e_{k+1}. That is, the information transmitted by i_k to i_{k-1} expresses i_k's opinion. The second one is that agent i trusts all the information sources in their competence about the information they have transmitted to the other information sources.

Theorem 8. *If* ϕ *is in* L *and* j, i, i_1, i_2, ... ,i_n *are in* $AGENT$ *and we adopt the following notations:*

$\Psi_{n,n-1} \stackrel{def}{=} \phi$ *and*

for k *in* $[1, n-1]$: $\Psi_{k,k-1} \stackrel{def}{=} (Inf_{i_{k+1},i_k}\Psi_{k+1,k}) \wedge (Inf_{e_{k+1},i_k}Val(i_{k+1}, i_k, \Psi_{k+1,k})) \wedge TrustVal(i_k, e_{k+1}, Val(i_{k+1}, i_k, \Psi_{k+1,k}))$

$TrustCompAll_i \stackrel{def}{=} \bigwedge_{k\in[1,n-1]} TrustComp(i, i_k, \Psi_{k+1,k})$

we have:

$\vdash TrustVal(i, i_1, \Psi_{1,0}) \wedge (Inf_{i_1,i}\Psi_{1,0}) \wedge TrustCompAll_i \rightarrow Bel_i\phi$

Proof. *The proof is similar as the proof of Theorem 6.*

Let us call (H) the formula: $TrustVal(i, i_1\Psi_{1,0}) \wedge (Inf_{i_1,i}\Psi_{1,0}) \wedge TrustCompAll_i$.

We prove by induction on l *that for every* l *in* $[1, n]$ *(H) entails* $Bel_i\Psi_{l,l-1}$.

For $l = 1$, *from (H) we have: (1)* $TrustVal(i, i_1, \Psi_{1,0}) \wedge (Inf_{i_1,i}\Psi_{1,0})$.

From (1) and Lemma 1 we have: $Bel_i\Psi_{1,0}$.

Induction hypothesis: (H) entails $Bel_i\Psi_{l,l-1}$.

From (H) and $\Psi_{l,l-1}$ *definition we have:*

(2) $Bel_i(Inf_{e_{l+1},i_l}Val(i_{l+1}, i_l, \Psi_{l+1,l})) \wedge TrustVal(i_l, e_{l+1}, Val(i_{l+1}, i_l, \Psi_{l+1,l}))$.

From (2) and Lemma 1 we have: (3) $Bel_iBel_{i_l}Val(i_{l+1}, i_l, \Psi_{l+1,l})$.

From $Bel_i\Psi_{l,l-1}$ *and* $\Psi_{l,l-1}$ *definition we also have: (4)* $Bel_iInf_{i_{l+1},i_l}\Psi_{l+1,l}$.

Then, from (3), (4) and Lemma 1 we have: (5) $Bel_iBel_{i_l}\Psi_{l+1,l}$.

From (H) and $TrustCompAll_i$ *definition we have:* $TrustComp(i, i_l, \Psi_{l+1,l})$, *and by* $TrustComp$ *definition we have: (6)* $Bel_i(Bel_{i_l}\Psi_{l+1,l} \rightarrow \Psi_{l+1,l})$.

Therefore, from (5) and (6) we have: $Bel_i\Psi_{l+1,l}$.

Then, for $l = n - 1$ *we have:* $Bel_i\Psi_{n,n-1}$, *that is* $Bel_i\phi$. **QED.**

The following Theorem 9 is the dual of Theorem 8.

Theorem 9. *If* ϕ *is in* L *and* j, i, i_1, i_2, ... ,i_n *are in* $AGENT$ *and we adopt the following notations:*

$\Psi_{n,n-1} \stackrel{def}{=} \phi$ *and*

for k *in* $[1, n-1]$: $\Psi_{k,k-1} \stackrel{def}{=} ((Inf_{e_{k+1},i_k}Cmp(i_{k+1}, i_k, \Psi_{k+1,k})) \wedge TrustVal(i_k, e_{k+1}, Cmp(i_{k+1}, i_k, \Psi_{k+1,k}))) \rightarrow (Inf_{i_{k+1},i_k}\Psi_{k+1,k})$

$TrustCompAll_i \stackrel{def}{=} \bigwedge_{k\in[1,n-1]} TrustComp(i, i_k, \neg\Psi_{k+1,k})$

we have:

$\vdash TrustCmp(i, i_1, \Psi_{1,0}) \wedge \neg(Inf_{i_1,i}\Psi_{1,0}) \wedge TrustCompAll_i \rightarrow Bel_i\neg\phi$

Proof. *The proof is similar as the proof of Theorem 8.*

Here (H) is the formula $TrustCmp(i, i_1, \Psi_{1,0}) \wedge \neg(Inf_{i_1,i}\Psi_{1,0}) \wedge TrustCompAll_i$ *and we prove by induction that (H) entails* $Bel_i\neg\Psi_{l,l-1}$.

It is assumed that (H) entails $Bel_i\neg\Psi_{l,l-1}$. *From* $\Psi_{l,l-1}$ *definition* $\neg\Psi_{l,l-1}$ *is equivalent to (1)* $(Inf_{e_{l+1},i_l}Cmp(i_{l+1}, i_l, \Psi_{l+1,l})) \wedge TrustVal(i_l, e_{l+1}, Cmp(i_{l+1}, i_l, \Psi_{l+1,l})) \wedge \neg(Inf_{i_{l+1},i_l}\Psi_{l+1,l})$.

We can easily show that (1) entails: (2) $Bel_{i_l}Cmp(i_{l+1}, i_l, \Psi_{l+1,l})$ *and (3)* $\neg(Inf_{i_{l+1},i_l}\Psi_{l+1,l})$. *From Lemma 1, (2) and (3) entail (4)* $Bel_{i_l}\neg\Psi_{l+1,l}$. *Therefore,* $Bel_i\neg\Psi_{l,l-1}$ *entails (5)* $Bel_iBel_{i_l}\neg\Psi_{l+1,l}$.

From $TrustCompAll_i$, *(H) entails (6)* $TrustComp(i, i_l, \neg\Psi_{l+1,l})$ *and (5) and (6) entail* $Bel_i\neg\Psi_{l+1,l}$. **QED.**

In the Theorems 2, 4, 6, 8 the information represented by ϕ is propagated under some assumptions from the first information source i_n in the chain until the agent i. It is worth noting that the proposition ϕ may be about some other information source j. For example, ϕ may be of the form: $Val(j, i, \theta)$ (resp.

$Cmp(j, i, \theta)$). In that case the conclusions of these theorems express that i trusts j in his validity (resp. his completeness) about θ that is: $TrustVal(i, j, \theta)$ (resp. $TrustVal(i, j, \theta)$).

The Theorems 3, 5, 7 and 9 respectively are the dual of 2, 4, 6 and 8 and from the fact that agent i has not been informed i can infer that ϕ is false.

7 Related Works

In [17] trust is represented by a probability associated to a binary relation between two agents. It is also assumed *a priori* that the trust relationship is transitive. These simplifications are assumed by the authors in order to be able to define a mathematical model to compute the "percolation" of trust in a graph of agents.

In [3] the authors have also considered that trust is just a binary relation between the truster and the trustee and the weight associated to this relation represents trust level. Then, the objective is to define a method to evaluate this weight. The method is based on the operators: aggregation, concatenation, and selection of information coming from different sources.

The authors in [16] investigate opinion propagation in order to determine agent's reputation level instead of agent's trust. It has similar objectives as our work with respect to the analysis of propagation. However, these opinions are not analyzed in detail and they are not explicitly considered as agents' beliefs.

The work presented in [2] is the work which is the closest to our work we have found in the literature. In an informal analysis trust is decomposed into several elements: the truster, the trustee and the purpose of trust. Then, these notions are formalized in the Josiang's Subjective Logic which, roughly speaking, can be seen as a combination of probability theory and epistemic logic. However, in this work trust purpose is represented by atomic propositions and no nested modal operator is used for reasoning about agents' beliefs. For example, it is not possible to represent the fact that some agent has some beliefs about the agents' beliefs in a chain of information sources as we did. The main contribution is to propose a technique to evaluate the level of trust in a context of trust propagation.

The common feature of these works is that it is assumed that trust is transitive and the goal is to find a method to compute how the trust level is propagated along a network of agents. Also, they all implicitly assume that all the agents are informed about the trust network, which is not really the case in many real applications. In [2] the trust purpose is explicit but this purpose is not analyzed in detail in the case where the purpose is to propagate information. For example, agents' properties like sincerity or competence are ignored.

8 Conclusion

Trust in information sources has been defined in terms of truster's belief about an entailment relation about some trustee's properties. It has been formally represented by formulas of the form: $Bel_i(Ant_j \rightarrow Cons_j)$, where the antecedent

Ant_j and the consequent $Cons_j$ can be a communication action $Inf_{j,i}\phi$ or a belief $Bel_j\phi$ or a fact ϕ.

These trust definitions have been used to define sufficient conditions which guarantee that information is propagated along a chain of information sources: i_n, i_{n-1}, ... , i_k, ... , i_1 until agent i. A particular case we have analyzed is when this information is about another information source j and, in that case, the effect of information propagation is that i trusts j in some properties.

The conditions that guarantee propagation can be about the fact that each information source is valid or complete or about the fact that each information source validity or completeness is evaluated by a valid evaluator or about the fact that each information source trusts the previous information source in the chain about his validity or completeness.

An original feature of these properties is that the agent i can draw the conclusion that a proposition ϕ is false from the fact that he did not receive an information about ϕ.

We have presented the proofs of the theorems because we think that they can help to understand the meaning of the information $\Phi_{k,k-1}$ or $\Psi_{k,k-1}$ transmitted between each information source in the cases where this information is represented by complex formulas. Indeed, the proofs are constructive and they are by induction on the rank of the information sources. Then, we can imagine that the agent i when he is reasoning about the information sources does the same proofs.

The results which have been presented in the theorems could be used as specifications to implement automated reasoning techniques in order to apply them to specific applications. It could be that to find efficient implementations we have to restrict the expressive power of the proposition ϕ which is propagated. That should deserve further works.

A possible direction for further works is to investigate whether the assumptions in these theorems are minimal in the sense that they are not only sufficient conditions to guarantee such or such kind of propagation, but that they also are necessary conditions.

Another direction is to consider a more general structure for information sources than a linear structure. For instance, if agent i infers that agent j is valid from the fact that i has been informed by i_1 about j's sincerity and by i_2 about j's competence, the structure of information sources is a tree.

Another direction could also be to consider graded trust instead of "yes/no" trust as we did in [11,10]. For example, in this approach graded validity is represented by: $Bel_i^g(Inf_{j,i}\phi \Rightarrow^h \phi)$, where h is the regularity level of the fact that $Inf_{j,i}\phi$ entails ϕ and g is the uncertainty level of agent i about this entailment level. That could be relevant to express that trust level decreases a long a chain of information sources.

References

1. Bacharach, M., Gambetta, D.: Trust as type detection. In: Castelfranchi, C., Tan, Y.-H. (eds.) Trust and Deception in Virtual Societies. Kluwer Academic Publisher, Dordrecht (2001)

2. Bhuiyan, T., Josang, A., Xu, Y.: An analysis of trust transitivity taking base rate into account. In: Proceeding of the Sixth International Conference on Ubiquitous Intelligence and Computing, Brisbane (2009)
3. Singh, M., Hang, C., Wang, Y.: Operators for propagating trust and their evaluation in social networks. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (2009)
4. Castelfranchi, C., Falcone, R.: Social trust: a cognitive approach. In: Castelfranchi, C., Tan, Y.-H. (eds.) Trust and Deception in Virtual Societies. Kluwer Academic Publisher, Dordrecht (2001)
5. Castelfranchi, C., Falcone, R.: Trust Theory: A Socio-Cognitive and Computational Model. Wiley, Chichester (2010)
6. Chellas, B.F.: Modal Logic: An introduction. Cambridge University Press, Cambridge (1988)
7. Cholvy, L., Demolombe, R., Jones, A.J.I.: Reasoning about the safety of information: from logical formalization to operational definition. In: Proc. of 8th International Symposium on Methodologies for Intelligent Systems (1994)
8. Demolombe, R.: To trust information sources: a proposal for a modal logical framework. In: Castelfranchi, C., Tan, Y.-H. (eds.) Trust and Deception in Virtual Societies. Kluwer Academic Publisher, Dordrecht (2001)
9. Demolombe, R.: Reasoning about trust: A formal logical framework. In: Jensen, C., Poslad, S., Dimitrakos, T. (eds.) iTrust 2004. LNCS, vol. 2995, pp. 291–303. Springer, Heidelberg (2004)
10. Demolombe, R.: Graded Trust. In: Falcone, R., Barber, S., Sabater-Mir, J., Singh, M. (eds.) Proceedings of the Trust in Agent Societies Workshop at AAMAS 2009 (2009)
11. Demolombe, R., Liau, C.-J.: A logic of graded trust and belief fusion. In: Castelfranci, C., Falcone, R. (eds.) Proc. of 4th Workshop on Deception, Fraud and Trust (2001)
12. Jones, A.J.I.: On the concept of trust. Decision Support Systems, 33 (2002)
13. Jones, A.J.I., Firozabadi, B.S.: On the characterisation of a trusting agent. Aspects of a formal approach. In: Castelfranchi, C., Tan, Y.-H. (eds.) Trust and Deception in Virtual Societies. Kluwer Academic Publisher, Dordrecht (2001)
14. Lorini, E., Demolombe, R.: Trust and norms in the context of computer security: A logical formalization. In: van der Meyden, R., van der Torre, L. (eds.) DEON 2008. LNCS (LNAI), vol. 5076, pp. 50–64. Springer, Heidelberg (2008)
15. Lorini, E., Demolombe, R.: From trust in information sources to trust in communication systems:An analysis in modal logic. In: Meyer, J.-J., Broersen, J. (eds.) KRAMAS 2008. LNCS(LNAI), vol. 5605, pp. 81–98. Springer, Heidelberg (2009)
16. Osman, N., Sierra, C., Sabater-Mir, J.: Propagation of opinions in structural graphs. In: 19th European Conference on Artificial Intelligence, ECAI 2010 (2010)
17. Richters, O., Peixoto, T.P.: Trust transitivity in social networks. Technical report, Darmstadt Technical University (2010)

The Functional Dependence Relation on Hypergraphs of Secrets

Sara Miner More and Pavel Naumov

Department of Mathematics and Computer Science
McDaniel College, Westminster, Maryland 21157, USA
{smore,pnaumov}@mcdaniel.edu

Abstract. The paper considers interdependencies between secrets in a multiparty system. Each secret is assumed to be known only to a certain fixed set of parties. These sets can be viewed as edges of a hypergraph whose vertices are the parties of the system. In previous work, the authors investigated properties of interdependencies that are expressible through a multi-argument relation called *independence*, which is a generalization of a binary relation also known as nondeducibility. This work studies properties expressible through functional dependence. The main result is a complete and decidable logical system that describes interdependencies on a fixed hypergraph.

1 Introduction

In this paper, we study properties of interdependencies between pieces of information. We call these pieces *secrets* to emphasize the fact that they might be known to some parties and unknown to the others. Below, we first describe two relations for expressing interdependencies between secrets. Next, we discuss these relations in the context of collaboration networks which specify the available communication channels for the parties establishing the secrets.

Relations on Secrets. If there is no interdependence at all between two secrets, then we will say that the two secrets are *independent*. In other words, secrets a and b are independent if any possible value of secret a is compatible with any possible value of secret b. We denote this relation between two secrets by $[a, b]$. This relation was introduced by Sutherland [1] and is also known as *nondeducibility* in the study of information flow. Halpern and O'Neill [2] proposed a closely-related notion called f-secrecy. In earlier work [3,4], we generalized independence to a relation $[a_1, \dots, a_n]$ between an arbitrary set of secrets.

Another natural relation between two secrets is *functional dependence*, which we denote by $a \rhd b$. It means that the value of secret a reveals the value of secret b. A more general and less trivial form of functional dependence is functional dependence between sets of secrets. If A and B are two sets of secrets, then $A \rhd B$ means that, together, the values of all secrets in A reveal the values of all secrets in B. Armstrong [5] presented the following sound and complete axiomatization of this relation:

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 29–40, 2011.

1. *Reflexivity*: $A \rhd B$, if $A \supseteq B$,
2. *Augmentation*: $A \rhd B \to A, C \rhd B, C$,
3. *Transitivity*: $A \rhd B \to (B \rhd C \to A \rhd C)$,

where here and everywhere below A, B denotes the union of sets A and B. The above axioms are known in database literature as Armstrong's axioms [6, p. 81]. Beeri, Fagin, and Howard [7] suggested a variation of Armstrong's axioms that describe properties of multi-valued dependence. A logical system that combines independence and functional dependence predicates was described by Kelvey, More, Naumov, and Sapp [8].

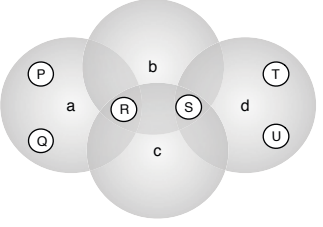


Fig. 1. Network H_0

Secrets in Networks. So far, we have assumed that the values of secrets are determined a priori. In the physical world, however, secret values are often generated, or at least disseminated, via interaction between several parties. Quite often such interactions happen over a network with fixed topology. For example, in social networks, interaction between nodes happens along connections formed by friendship, kinship, financial relationship, etc. In distributed computer systems, interaction happens over computer networks. Exchange of genetic information happens along the edges of the genealogical tree. Corporate secrets normally flow over an organization chart. In cryptographic protocols, it is often assumed that values are transmitted over well-defined channels. On social networking websites, information is shared between "friends". Messages between objects on an UML interaction diagram are sent along connections defined by associations between the classes of the objects.

In this paper, we will use the notion of *collaboration network* to refer to the topological structure that specifies which secrets are known to which parties. An example of such network is given in Figure 1. In this network, parties P, Q and R share[1] secret a; parties R and S share secrets b and c; and parties S, T and U share secret d. If different secrets are established completely independently, then possession of one or several of these secrets reveals no information about the other secrets. Assume, however, that secrets are not picked completely

[1] In this paper, the "sharing of a secret" between parties means that all parties know the entire secret in question; this is not to be confused with cryptographic secret-sharing [9].

independently. Instead, each party with access to multiple secrets may enforce some desired interdependence between the values of these secrets. These "local" interdependencies between secrets known to a single party may result in a "global" interdependence between several secrets, not all of which are known to any single party. Given the fixed topology of the collaboration network, we study what global interdependencies between secrets may exist in the system.

We will say that the local interdependencies define a *protocol*. For the collaboration network H_0 depicted in Figure 1, for example, we can imagine the following protocol. Parties P, Q and R together pick a random value a from set $\{0, 1\}$. Next, party R chooses values b and c from $\{0, 1\}$ in such a way that $a = b + c \mod 2$ and sends both of these values to party S. Party S computes $d = b + c \mod 2$ and shares value d with parties T and U. In this protocol, it is clear that the values of a and d will always match. Hence, for this specific protocol, we can say that $a \rhd d$ and $a, b \rhd c, d$, but at the same time, $[a, b]$ and $[a, c]$.

The functional dependence and independence examples above are for a single protocol, subject to a particular set of local interdependencies between secrets. If the network remains fixed, but the protocol is changed, then secrets which were previously interdependent may no longer be so, and vice versa. For example, for network H_0 above, the claim $a \rhd d$ will no longer be true if, say, party s switches from enforcing the local condition $d = b + c \mod 2$ to enforcing the local condition $d = b$. In this paper, we study properties of relations between secrets that follow from the topological structure of the collaboration network, no matter which specific protocol is used. Examples of such properties for network H_0 are $a \rhd d \to b, c \rhd d$ and $[a, b, c] \to [a, d]$.

In our previous CLIMA paper [4], we gave a complete axiomatization of all properties of independence between sets of secrets over an arbitrary collaboration network. In this work, we give a similar axiomatization for the properties of functional dependence. It consists of the above-mentioned Armstrong axioms and an additional *Gateway* axiom that captures properties specific to the topology of the collaboration network.

Although the proposed logical system captures properties of functional dependence that are not specific to any protocol, this logic could potentially be used as a framework for reasoning about specific protocols in the same way, for example, as the first order logic is used for reasoning about specific mathematical theories.

2 Hypergraphs

A collaboration network where a single secret can be shared between multiple parties can be described mathematically as a hypergraph in which vertices are parties and (hyper)edges are secrets. In this section, we introduce the hypergraph terminology that is used later in the article.

Definition 1. *A hypergraph is pair $H = \langle V, E\rangle$, where*

1. *V is a finite set, whose elements are called "vertices".*
2. *E is a finite multiset of subsets of V. Elements of E are called "edges". Elements of an edge are called the "ends" of the edge.*

Note that we use "mulitisets" in the above definition to allow for multiple edges between the same set of ends.

A *path* in a hypergraph is a sequence of edges in which adjacent edges share at least one end. Paths will be assumed to be simple, in the sense that no edge is repeated in a path.

Definition 2. *A gateway between sets of edges A and B is a set of edges G such that every path from A to B contains at least one edge from G.*

For instance, set $\{b, c\}$ is a gateway between single-element sets $\{a\}$ and $\{d\}$ on the hypergraph H_0 from Figure 1. Note also that in the definition above, sets A, B, and G are not necessarily disjoint. Thus, for example, for any set of edges A, set A is a gateway between A and itself. Also, note that the empty set is a gateway between any two components of the hypergraph that are not connected one to another.

3 Protocol: A Formal Definition

Definition 3. *A protocol over a hypergraph $H = \langle V, E\rangle$ is a pair $\mathcal{P} = \langle Val, Loc\rangle$ such that*

1. *$Val(e)$ is an arbitrary set of "values" for each edge $e \in E$,*
2. *$Loc = \{Loc(v)\}_{v\in V}$ is a family of relations, indexed by vertices (parties) of the hypergraph H, which we call "local conditions". If $Inc(v)$ is the set of all edges incident with vertex v, then $Loc_v \subseteq \prod_{e\in Inc(v)} Val(e)$.*

Definition 4. *A run of a protocol $\langle Val, Loc\rangle$ is a function r such that*

1. *$r(e) \in Val(e)$ for any edge $e \in E$,*
2. *$\langle r(e)\rangle_{e\in Inc(v)} \in Loc(v)$.*

The set of all runs of a protocol $\mathcal{P}$ is denoted by $\mathcal{R}(\mathcal{P})$.

Definition 5. *A protocol $\mathcal{P} = \langle Val, Loc\rangle$ is called finite if the set $Val(e)$ is finite for every edge e of the hypergraph.*

We conclude this section with the key definition of this paper. It is the definition of functional dependence between sets of edges.

Definition 6. *A set of edges A functionally determines a set of edges B, with respect to a fixed protocol $\mathcal{P}$, if*

$$\forall r, r' \in \mathcal{R}(\mathcal{P}) \left(\bigwedge_{a\in A} r(a) = r'(a) \to \bigwedge_{b\in B} r(b) = r'(b) \right).$$

We find it convenient to use the notation $f \equiv_X g$ if functions f and g are equal on every argument from set X. Using this notation, we can say that a set of edges A functionally determines a set of edges B if

$$\forall r, r' \in \mathcal{R}(\mathcal{P})\ (r \equiv_A r' \rightarrow r \equiv_B r').$$

4 Language of Secrets

By $\Phi(H)$, we denote the set of all collaboration network properties specified by hypergraph H that are expressible through the functional dependence predicate. More formally, $\Phi(H)$ is the minimal set of formulas defined recursively as follows: (i) for any finite subsets A and B of the set of all edges of hypergraph H, formula $A \rhd B$ is in $\Phi(H)$, (ii) the false constant $\perp$ is in $\Phi(H)$, and (iii) for any formulas ϕ and $\psi \in \Phi(H)$, the implication $\phi \rightarrow \psi$ is in $\Phi(H)$. As usual, we assume that conjunction, disjunction, and negation are defined through $\rightarrow$ and $\perp$.

Next, we define a relation $\vDash$ between a protocol and a formula from $\Phi(H)$. Informally, $\mathcal{P} \vDash \phi$ means that formula ϕ is true under protocol $\mathcal{P}$.

Definition 7. *For any protocol $\mathcal{P}$ over a hypergraph H, and any formula $\phi \in \Phi(H)$, we define the relation $\mathcal{P} \vDash \phi$ recursively as follows:*

1. $\mathcal{P} \nvDash \perp$,
2. $\mathcal{P} \vDash A \rhd B$ *if the set of edges A functionally determines set of edges B under protocol $\mathcal{P}$,*
3. $\mathcal{P} \vDash \phi_1 \rightarrow \phi_2$ *if* $\mathcal{P} \nvDash \phi_1$ *or* $\mathcal{P} \vDash \phi_2$.

In this article, we study the formulas $\phi \in \Phi(H)$ that are true under *every* protocol $\mathcal{P}$ over a fixed hypergraph H. Below we describe a formal logical system for such formulas. This system, like earlier systems defined by Armstrong [5], More and Naumov [10,3,4] and by Kelvey, More, Naumov, and Sapp [8], belongs to the set of deductive systems that capture properties of secrets. In general, we refer to such systems as *logics of secrets*. Since this article is focused on only one such system, here we call it simply the *Logic of Secrets* of hypergraph H.

5 Axioms

For a fixed hypergraph H, the Logic of Secrets, in addition to propositional tautologies and the Modus Ponens inference rule, contains the following axioms:

1. *Reflexivity*: $A \rhd B$, if $A \supseteq B$,
2. *Augmentation*: $A \rhd B \rightarrow A, C \rhd B, C$,
3. *Transitivity*: $A \rhd B \rightarrow (B \rhd C \rightarrow A \rhd C)$,
4. *Gateway* : $A \rhd B \rightarrow G \rhd B$, if G is a gateway between sets A and B in hypergraph H.

Recall that the first three of these axioms were introduced by Armstong [5]. The soundness of all four axioms will be shown in Section 7. We use the notation $X \vdash_H \Phi$ to state that formula Φ is derivable from the set of formulas X in the Logic of Secrets for hypergraph H.

6 Examples of Proofs

In this section, we give four examples of proofs in the Logic of Secrets. Our first example refers to the square hypergraph H_1 depicted in Figure 2.

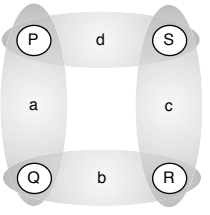


Fig. 2. Hypergraph H_1

Proposition 1. $\vdash_{H_1} (a \rhd c) \wedge (b \rhd d) \rightarrow (a \rhd d) \wedge (b \rhd c)$.

Proof. Due to the symmetry of the hypergraph, it is sufficient to show that $(a \rhd c) \wedge (b \rhd d) \rightarrow a \rhd d$. Note that $\{a, c\}$ is a gateway between sets $\{b\}$ and $\{d\}$. Thus, by the Gateway axiom, $b \rhd d$ implies $(a, c \rhd d)$. On the other hand, by the Augmentation axiom, the assumption $a \rhd c$ yields $(a \rhd a, c)$. By the Transitivity axiom, $(a \rhd a, c)$ and $(a, c \rhd d)$ imply $a \rhd d$. □

For the second example, consider the linear hypergraph H_2 shown in Figure 3.

Fig. 3. Hypergraph H_2

Proposition 2. $\vdash_{H_2} (a \rhd d) \wedge (e \rhd c) \rightarrow b \rhd c$.

Proof. We begin with the assumption that $e \rhd c$. Since $\{d\}$ is a gateway between sets $\{e\}$ and $\{c\}$, by the Gateway axiom, $d \rhd c$. Next, using the assumption that $a \rhd d$, the Transitivity axiom yields $a \rhd c$. Finally, we note that $\{b\}$ is a gateway between $\{a\}$ and $\{c\}$, and apply the Gateway axiom once again to conclude that $b \rhd c$. □

Note that the second hypothesis in the example above is significant. Indeed, imagine a protocol on H_2 where $V(d) = \{0\}$, the set of values allowed on all other edges is $\{0, 1\}$, and the local condition at each vertex v is always true, or, formally, $L(v) \equiv \prod_{e \in Inc(v)} Val(e)$. Under this protocol, $a \rhd d$ since the value of a on any run trivially determines the (constant) value of d. However, the value of b is of no help in determining the value of c, so the conclusion $b \rhd c$ does not hold.

Next, consider the "olympic rings" hypergraph H_3 shown in Figure 4.

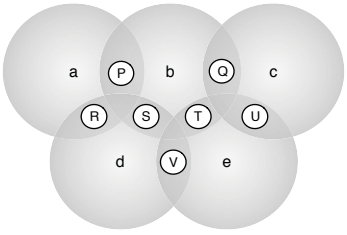


Fig. 4. Hypergraph H_3

Proposition 3. $\vdash_{H_3} (a \rhd e) \wedge (c \rhd d) \rightarrow b, c \rhd e$.

Proof. Assume that $a \rhd e$ and $c \rhd d$. Note that set $\{d, b\}$ is a gateway between sets $\{a\}$ and $\{e\}$. Thus, by the Gateway axiom, from assumption $a \rhd e$ we can conclude that $d, b \rhd e$. The assumption $c \rhd d$, by the Augmentation axiom, implies that $b, c \rhd b, d$. Therefore, by the Transitivity axiom, $b, c \rhd e$. □

As our final example, we prove a property of the hexagonal hypergraph H_3 shown in Figure 5.

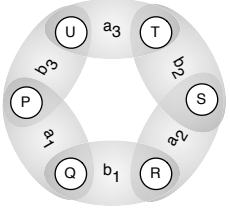


Fig. 5. Hypergraph H_4

Proposition 4. $\vdash_{H_4} (a_1, a_2 \rhd a_3) \wedge (a_2, a_3 \rhd a_1) \wedge (a_3, a_1 \rhd a_2) \rightarrow b_1, b_2, b_3 \rhd a_1, a_2, a_3$.

Proof. Note that $\{b_1, b_3\}$ is a gateway between sets $\{a_2, a_3\}$ and $\{a_1\}$. Thus, by the Gateway axiom, $a_2, a_3 \rhd a_1 \rightarrow b_1, b_3 \rhd a_1$. Hence, by the assumption, $a_2, a_3 \rhd a_1$, we have that $b_1, b_3 \rhd a_1$. Similarly one can show that $b_1, b_2 \rhd a_2$ and $b_2, b_3 \rhd a_3$ using the assumptions $a_3, a_1 \rhd a_2$ and $a_1, a_2 \rhd a_3$.

Consider statements $b_1, b_3 \rhd a_1$ and $b_1, b_2 \rhd a_2$. By the Augmentation axiom, they, respectively, imply that $b_1, b_2, b_3 \rhd a_1, b_1, b_2$ and $a_1, b_1, b_2 \rhd a_1, a_2$. Thus, by the Transitivity axiom, $b_1, b_2, b_3 \rhd a_1, a_2$.

Now consider $b_1, b_2, b_3 \rhd a_1, a_2$ and statement $b_2, b_3 \rhd a_3$, established earlier. By the Augmentation axiom, they, respectively, imply that $b_1, b_2, b_3 \rhd a_1, a_2, b_2, b_3$ and $a_1, a_2, b_2, b_3 \rhd a_1, a_2, a_3$. Thus, by the Transitivity axiom, $b_1, b_2, b_3 \rhd a_1, a_2, a_3$. □

7 Soundness

In this section, we demonstrate the soundness of each of the four axioms in the Logic of Secrets.

Theorem 1 (Reflexivity). *$\mathcal{P} \vDash A \rhd B$, for any protocol $\mathcal{P}$ and any $B \subseteq A$.*

Proof. Consider any two runs $r, r' \in \mathcal{R}(\mathcal{P})$ such that $r \equiv_A r'$. Thus $r \equiv_B r'$ for any $B \subseteq A$. □

Theorem 2 (Augmentation). *$\mathcal{P} \vDash A \rhd B \to A, C \rhd B, C$, for any protocol $\mathcal{P}$ and any sets of edges A, B, and C.*

Proof. Assume $\mathcal{P} \vDash A \rhd B$ and consider any two runs $r, r' \in \mathcal{R}(\mathcal{P})$ such that $r \equiv_{A,C} r'$. By our assumption, $r \equiv_B r'$. Therefore, $r \equiv_{B,C} r'$. □

Theorem 3 (Transitivity). *$\mathcal{P} \vDash A \rhd B \to (B \rhd C \to A \rhd C)$, for any protocol $\mathcal{P}$ and any sets of edges A, B, and C.*

Proof. Assume $\mathcal{P} \vDash A \rhd B$ and $\mathcal{P} \vDash B \rhd C$. Consider any two runs $r, r' \in \mathcal{R}(\mathcal{P})$ such that $r \equiv_A r'$. By the first assumption, $r \equiv_B r'$. By the second assumption, $r \equiv_C r'$. □

Theorem 4 (Gateway). *$\mathcal{P} \vDash A \rhd B \to G \rhd B$, for any protocol $\mathcal{P}$ and any gateway G between sets A and B.*

Proof. Assume $\mathcal{P} \vDash A \rhd B$ and consider any two runs $r_1, r_2 \in \mathcal{R}(\mathcal{P})$ such that $r_1 \equiv_G r_2$. We will show that $r_1 \equiv_B r_2$. Consider the hypergraph H' obtained from H by removal of all edges in set G. By the definition of a gateway, no single connected component of hypergraph H' can contain edges from set $A \setminus G$ and set $B \setminus G$ at the same time. Let us divide all connected components of H' into two subgraphs H'_A and H'_B such that H'_A contains no edges from $B \setminus G$ and H'_B contains no edges from $A \setminus G$. Components that do not contain edges from either $A \setminus G$ or $B \setminus G$ can be arbitrarily assigned to either H'_A or H'_B.

Next, define a function r on each $c \in E$ as follows:

$$r(c) = \begin{cases} r_1(c) & \text{if } c \in H'_A, \\ r_1(c) = r_2(c) & \text{if } c \in G, \\ r_2(c) & \text{if } c \in H'_B. \end{cases}$$

We will prove that r is a run of protocol $\mathcal{P}$. We need to show that r satisfies the local conditions of protocol $\mathcal{P}$ at each vertex v. The connected component of H' containing a vertex v either belongs to H'_A or H'_B. Without loss of generality, assume that it belongs to H'_A. Thus, $Inc(v)$, the set of all edges in H incident with vertex v, is a subset of $H'_A \cup G$. Hence, $r \equiv_{Inc(v)} r_1$. Therefore, r satisfies the local condition at vertex v simply because r_1 does.

By the definition of r, we have $r \equiv_A r_1$ and $r \equiv_B r_2$. Together, the first of these statements and the assumption that $\mathcal{P} \vDash A \rhd B$ imply that $r \equiv_B r_1$. Thus, due to the second statement, $r_1 \equiv_B r \equiv_B r_2$. □

8 Completeness

In this section, we demonstrate that the Logic of Secrets is complete with respect to the semantics defined above. To do so, we first describe the construction of a protocol called $\mathcal{P}_0$, which is implicitly parameterized by a hypergraph and a set of formulas.

8.1 Protocol $\mathcal{P}_0$

Throughout this section, we will assume that $H = \langle V, E\rangle$ is a fixed hypergraph, and $X \subseteq \Phi(H)$ is a fixed set of formulas.

Definition 8. *For any $A \subseteq E$, we define A^* to be the set of all edges $c \in E$ such that $X \vdash_H A \rhd c$.*

Theorem 5. *$A \subseteq A^*$, for any $A \subseteq E$.*

Proof. Let $a \in A$. By the Reflexivity axiom, $\vdash_H A \rhd a$. Hence, $a \in A^*$. □

Theorem 6. *$X \vdash_H A \rhd A^*$, for any $A \subseteq E$.*

Proof. Let $A^* = \{a_1, \ldots, a_n\}$. By the definition of A^*, $X \vdash_H A \rhd a_i$, for any $i \leq n$. We will prove, by induction on k, that $X \vdash_H (A \rhd a_1, \ldots, a_k)$ for any $0 \leq k \leq n$.
Base Case: $X \vdash_H A \rhd \varnothing$ by the Reflexivity axiom.
Induction Step: Assume that $X \vdash_H (A \rhd a_1, \ldots, a_k)$. By the Augmentation axiom,

$$X \vdash_H A, a_{k+1} \rhd a_1, \ldots, a_k, a_{k+1}. \tag{1}$$

Recall that $X \vdash_H A \rhd a_{k+1}$. Again by the Augmentation axiom, $X \vdash_H (A \rhd A, a_{k+1})$. Hence, $X \vdash_H (A \rhd a_1, \ldots, a_k, a_{k+1})$, by (1) and the Transitivity axiom. □

We now proceed to define our protocol $\mathcal{P}_0$. We will first specify the set of values $Val(c)$ for each edge $c \in E$. In this construction, the value of each edge c on a particular run will be a function from the set 2^E into the set $\{0, 1\}$. Thus, for any $c \in E$ and any $F \subseteq E$, we have $r(c)(F) \in \{0, 1\}$. We will find it more convenient, however, to think about r as a two-argument boolean function: $r(c, F) \in \{0, 1\}$.

Furthermore, we will not allow the value of a edge on a particular run to be just *any* function from the set 2^E into $\{0, 1\}$. Instead, for any edge c, we will restrict set $Val(c)$ so that, for any run r, if $c \in F^*$, then $r(c, F) = 0$.

To complete the description of protocol $\mathcal{P}_0$, we will specify the local conditions for each vertex in the hypergraph. At each vertex v, we define the local condition $Loc(v)$ in such away that run $r(c, F)$ satisfies $Loc(v)$ if and only if

$$\forall F \subseteq E\ \forall c, d \in (Inc(v) \setminus F^*)\ (r(c, F) = r(d, F)).$$

That is, when two edges are incident with a vertex v and neither edge is in F^*, the values of the functions assigned to those edges on argument F must match on any given run.

Now that the definition of protocol $\mathcal{P}_0$ is complete, we make the following two claims about its relationship to the given set of formulas X.

Theorem 7. *If* $\mathcal{P}_0 \vDash A \rhd B$, *then* $X \vdash_H A \rhd B$.

Proof. Assume $\mathcal{P}_0 \vDash A \rhd B$ and consider two specific runs of $\mathcal{P}_0$. The first of these two runs will be the constant run $r_1(c, F) = 0$. The second run is defined as

$$r_2(c, F) = \begin{cases} 1 \text{ if } c \notin A^* \text{ and } F = A, \\ 0 \text{ if } c \in A^* \text{ or } F \neq A. \end{cases} \tag{2}$$

Run r_1 trivially satisfies the local condition at every vertex v. To show that r_2 satisfies the local condition at a vertex v, consider any $F \subseteq E$ and any $c, d \in Inc(v) \setminus F^*$. If $F \neq A$, then $r_2(c, F) = 0 = r_2(d, F)$. If $F = A$, then, since $c, d \in Inc(v) \setminus F^*$, we have $c, d \notin A^*$. Thus, $r_2(c, F) = 1 = r_2(d, F)$. Therefore, r_2 is a run of protocol $\mathcal{P}_0$.

Notice that by Theorem 5, $A \subseteq A^*$. Thus, by equality (2), $r_2(a, F) = 0$ for any $a \in A$ and any $F \subseteq E$. Hence, $r_1(a, F) = 0 = r_2(a, F)$ for any $a \in A$ and $F \subseteq E$. Thus, by the assumption that $\mathcal{P}_0 \vDash A \rhd B$, we have $r_1(b, F) = r_2(b, F)$ for any $b \in B$ and $F \subseteq E$. In particular, $r_1(b, A) = r_2(b, A)$ for any $b \in B$. Since, by definition, $r_1(b, A) = 0$, we get $r_2(b, A) = 0$ for any $b \in B$. By the definition of r_2, this means that $B \subseteq A^*$. By the Reflexivity axiom, $\vdash_H A^* \rhd B$. By Theorem 6 and the Transitivity axiom, $X \vdash_H A \rhd B$. □

Theorem 8. *If* $X \vdash_H A \rhd B$, *then* $\mathcal{P}_0 \vDash A \rhd B$.

Proof. Assume that $X \vdash_H A \rhd B$, but $\mathcal{P}_0 \nvDash A \rhd B$. Thus, there are runs r_1 and r_2 of $\mathcal{P}_0$ such that $r_1(a, F) = r_2(a, F)$ for any $a \in A$ and any $F \subseteq E$, yet there are $b_0 \in B$ and $F_0 \subseteq E$ such that

$$r_1(b_0, F_0) \neq r_2(b_0, F_0). \tag{3}$$

First, assume that hypergraph H', obtained from H by the removal of all edges in set F_0^*, contains a path π connecting edge b_0 with a edge $a_0 \in A$. This case implicitly assumes that $b_0, a_0 \notin F_0^*$. Let functions f_1 and f_2 on the edges of hypergraph H be defined as $f_1(c) = r_1(c, F_0)$ and $f_2(c) = r_2(c, F_0)$. Due to the local conditions of protocol $\mathcal{P}_0$, all edges along path π must have the same value of function f_1. The same is also true about function f_2. Therefore, $r_1(b_0, F_0) = f_1(b_0) = f_1(a_0) = r_1(a_0, F_0) = r_2(a_0, F_0) = f_2(a_0) = f_2(b_0) = r_2(b_0, F_0)$. This is a contradiction with statement (3).

Next, suppose that there is no path in H' connecting b_0 with a edge in A. Thus, set F_0^* is a gateway between sets A and $\{b_0\}$. By the Gateway axiom,

$$\vdash_H A \rhd b_0 \to F_0^* \rhd b_0. \tag{4}$$

By the Reflexivity axiom, $\vdash_H B \rhd b_0$. Recall the assumption $X \vdash_H A \rhd B$. Thus, by the Transitivity axiom, $X \vdash_H A \rhd b_0$. Taking into account (4), $X \vdash_H F_0^* \rhd b_0$. By Theorem 6, $\vdash F_0 \rhd F_0^*$. Hence, again by Transitivity, $X \vdash_H F_0 \rhd b_0$. Thus, by Definition 8, $b_0 \in F_0^*$. Hence, by the definition of protocol $\mathcal{P}_0$, $r(b_0, F_0)$ has value 0 for any run r. Therefore, $r_1(b_0, F_0) = 0 = r_2(b_0, F_0)$. This is a contradiction with statement (3). □

8.2 Main Result

Now, we are ready to finish the proof of completeness.

Theorem 9. *If* $\nvdash_H \phi$*, then there is a finite protocol* $\mathcal{P}$ *such that* $\mathcal{P} \not\models \phi$*.*

Proof. Assume $\nvdash_H \phi$. Let X be a maximal consistent set of formulas such that $\neg\phi \in X$. Consider the finite protocol $\mathcal{P}_0$ parameterized by hypergraph H and set of formulas X. For any formula ψ, we will show that $X \vdash_H \psi$ if and only if $\mathcal{P}_0 \models \psi$. The proof is by induction on the structural complexity of formula ψ. The base case follows from Theorems 7 and 8. The induction case follows from the maximality and consistency of set X. To finish the proof of the theorem, select ψ to be $\neg\phi$. □

Corollary 1. *Binary relation* $\vdash_H \phi$ *is decidable.*

Proof. This statement follows from the completeness of the Logic of Secrets with respect to *finite* protocols and the recursive enumerability of all theorems in the logic. □

9 Conclusion

We have presented a complete axiomatization of the properties of the functional dependence relation over secrets on hypergraphs. In light of previous results capturing properties of the independence relation in the same setting [4], it would be interesting to describe properties that connect these two predicates on hypergraphs.

An example of such a property for the hypergraph H_6 in Figure 6 is given in the following theorem.

Fig. 6. Hypergraph H_6

Theorem 10. *For any protocol* $\mathcal{P}$ *over hypergraph* H_6*,*

$$\mathcal{P} \models (a, b \rhd c) \wedge [a, b] \rightarrow b \rhd c.$$

Proof. For any two runs $r_1, r_2 \in \mathcal{R}(\mathcal{P})$ where $r_1(b) = r_2(b)$, we must show that $r_1(c) = r_2(c)$. The assumption $[a, b]$ guarantees that values $r_1(a)$ and $r_2(b)$ coexist in some run in $\mathcal{R}(\mathcal{P})$; call this run r_3. Thus, we have $r_3(a) = r_1(a)$ and $r_3(b) = r_2(b)$.

Next, we create a new function r_4 which "glues" together runs r_3 and r_2 at vertex q. Formally, we define r_4 as

$$r_4(x) = \begin{cases} r_3(x) & \text{if } x = a, \\ r_2(x) & \text{if } x \in \{b, c\}. \end{cases}$$

We claim that function r_4 satisfies the local conditions of protocol $\mathcal{P}$, since at each vertex in H_5, it behaves locally like an existing run. Indeed, at vertex p, r_4 matches run r_3, and at parties r and s, r_4 matches run r_2. At vertex q, r_4 matches r_2 exactly, since $r_4(b) = r_2(b)$. Thus, $r_4 \in \mathcal{R}(\mathcal{P})$. To complete the proof, we note that $r_1(a) = r_3(a) = r_4(a)$ and $r_1(b) = r_2(b) = r_4(b)$. By the assumption that $(a, b \rhd c)$, we have $r_1(c) = r_4(c)$. The definition of r_4 is such that $r_4(c) = r_2(c)$, so $r_1(c) = r_2(c)$, as desired. □

A complete axiomatization of properties that connect the functional dependence relation and the independence relation between secrets on a hypergraph remains an open problem.

Acknowledgment

The authors would like to thank Andrea Mills and Benjamin Sapp for discussions of the functional dependence relation on sets of secrets during earlier stages of this work.

References

1. Sutherland, D.: A model of information. In: Proceedings of Ninth National Computer Security Conference, pp. 175–183 (1986)
2. Halpern, J.Y., O'Neill, K.R.: Secrecy in multiagent systems. ACM Trans. Inf. Syst. Secur. 12(1), 1–47 (2008)
3. Miner More, S., Naumov, P.: On interdependence of secrets in collaboration networks. In: Proceedings of 12th Conference on Theoretical Aspects of Rationality and Knowledge, pp. 208–217. Stanford University, Stanford (2009)
4. Miner More, S., Naumov, P.: Hypergraphs of multiparty secrets. In: Dix, J., Leite, J., Governatori, G., Jamroga, W. (eds.) CLIMA XI. LNCS(LNAI), vol. 6245, pp. 15–32. Springer, Heidelberg (2010)
5. Armstrong, W.W.: Dependency structures of data base relationships. In: Information Processing, Proc. IFIP Congress, Stockholm, pp. 580–583. North-Holland, Amsterdam (1974)
6. Garcia-Molina, H., Ullman, J., Widom, J.: Database Systems: The Complete Book, 2nd edn. Prentice-Hall, Englewood Cliffs (2009)
7. Beeri, C., Fagin, R., Howard, J.H.: A complete axiomatization for functional and multivalued dependencies in database relations. In: SIGMOD 1977: Proceedings of the 1977 ACM SIGMOD International Conference on Management of Data, pp. 47–61. ACM, New York (1977)
8. Kelvey, R., Miner More, S., Naumov, P., Sapp, B.: Independence and functional dependence relations on secrets. In: Proceedings of 12th International Conference on the Principles of Knowledge Representation and Reasoning, Toronto, pp. 528–533. AAAI, Menlo Park (2010)
9. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery 22(11), 612–613 (1979)
10. More, S.M., Naumov, P.: An independence relation for sets of secrets. In: Ono, H., Kanazawa, M., de Queiroz, R. (eds.) WoLLIC 2009. LNCS(LNAI), vol. 5514, pp. 296–304. Springer, Heidelberg (2009)

Three Steps

Hans van Ditmarsch and Fernando Soler–Toscano

University of Sevilla, Spain
{hvd,fsoler}@us.es

Abstract. Given is a deal of ten cards over three players, such that two players each get four cards and the remaining player (the 'eavesdropper') two cards. We show that there does not exist a protocol of two steps for the four-card players to inform each other safely of their hands of cards, and we then present a protocol of three steps that achieves that goal. We verify the properties of that protocol by combinatorial and, mainly, logical (model checking) means. No such three-step protocol for cards was known. The method can be generalized. This will advance the characterization of card deals for which such exchanges of secrets are possible.

1 Knowledge-Based Protocols for Card Players

From a pack of seven known cards two players A and B each draw three cards and a third player C gets the remaining card. How can A and B openly (publicly) inform each other about their cards, without the third player learning from any of their cards who holds it?

This problem is often known as the Russian Cards Problem [16] and goes back to [9]. Such issues in cards cryptography have been investigated in the logical and model checking community [19,4,17,11], including related issues in epistemic puzzles, and also in combinatorics and theoretical computer science [7,14,10,20], including related issues in bit-exchange protocols.

One solution for the riddle is as follows. Suppose that the actual deal of cards is that agent A has $\{0, 1, 2\}$, B has $\{3, 4, 5\}$ and C has $\{6\}$.

- A says: My hand is one of 012, 046, 136, 145, 235.
- B says: C's card is 6.

After this, it is common knowledge to the three agents that A knows the hand of B, that B knows the hand of A, and that C is ignorant of the ownership of any card not held by herself.

We can also see these two sequences as the execution of a knowledge-based protocol. Given A's hand of cards, there is a (non-deterministic) way to produce her announcement, and given her announcement, B always responds by announcing C's card. The protocol is knowledge-based, because the agents initially only know their own hand of cards, and have public knowledge of the deck of cards and how many cards each agent has drawn from the pack. We can imagine agent A producing her announcement as follows from her initial knowledge (so, indeed, it is a function of her local state to her action):

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 41–57, 2011.

> Let my hand of cards be ijk and let the remaining cards be $lmno$. Choose one from ijk, say, w.l.o.g. i, and choose two from $lmno$, say lm. Three of the hands in my announcement are ijk, ilm, and ino. From lm choose one, say l, and from no choose one, say n. The two remaining hands are jln and kmo. Announce these five hands in random order.

This first step of such a protocol extends to a trivial second step wherein B announces C's card. It can be viewed as an *unconditionally secure* protocol, as C cannot learn any of the cards of A and B, no matter her computational resources. The security is therefore not conditional on the high complexity of some computation.

The Russian Card problem can be seen as the $(3, 3, 1)$ instance of the general (a, b, c) case, where A, B, C hold a, b, c cards, respectively. When can A and B communicate their hands of cards to each other, and when not? And how many steps are needed in a protocol realizing this?

Some results are found in [7,16,1,2]. An announcement by a player in a protocol is always equivalent to the announcement of a set of alternative hands including the actual hand. There are two-step protocols for a wide range of cases, for example, merely to mention some instantiations of more general patterns, for $(4, 2, 1)$, for $(22, 22, 1)$, and for $(10, 18, 3)$ (an instantation of [2, Theorem3] for a deck of $p^2 + p + 1$ cards, for prime $p = 5$).

The protocol above is a two-step protocol: B's answer depends on hearing A's announcement; he does not know what C's card is initially. Surprisingly, there are 'one-step protocols' of a kind: a different way to produce the sequence 012, 046, 136, 145, 235 is to observe that the sum of the cards in every triple is 3 modulo 7. So A could execute the protocol wherein she announces the sum of her cards, modulo 7. Agent B could also have done that, and even *at the same time* as A (alternatively to announcing C's card, *after* A's announcement) [2]. We require that A and B make their announcement in some order and therefore keep calling that a two-step protocol as well. In [2] we have proved that for $(n, n, 1)$ with $n > 2$ there is a two-step protocol, with A announcing the sum of his cards modulo $2n + 1$ and B announcing C's card. Recently we have generalized this result to provide two-step protocols for $(n, m, 1)$ with $n, m > 2$.

There does not always exist a protocol for (a, b, c), for two players to exchange their hands of cards. For example, this is not possible when each player holds one card [7]. If there exists a protocol for (a, b, c), is it always possible to make the exchange in two steps? Other works [7,14,10] give protocols of various length, three, four, or more steps—but without a proof of minimality. The answer to that question is: no. This contribution presents a three-step protocol for $(4, 4, 2)$ and a proof that no shorter protocol exists.

2 Logical Preliminaries

Protocols for card deals consist of public announcements. Public announcement logic [13,3] is an extension of multi-agent epistemic logic. Its language, structures, and semantics are as follows.

Given are a finite set of agents $\mathcal{A}$ and a countable set of propositional variables P. The *language of public announcement logic* is inductively defined as

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_A\varphi \mid C_{\mathcal{A}'}\varphi \mid [!\varphi]\psi$$

where $p \in P$, $A \in \mathcal{A}$, and $\mathcal{A}' \subseteq \mathcal{A}$. For $K_A\varphi$, read 'agent A knows formula φ'. For $C_{\mathcal{A}'}\varphi$, read 'group of agents $\mathcal{A}'$ commonly know formula φ'. For $[!\varphi]\psi$, read 'after truthful public announcement of φ, formula ψ (is true)'.

An *epistemic model* $M = \langle S, \sim, V\rangle$ consists of a *domain* S of *states* (or 'worlds'), an *accessibility function* $R : \mathcal{A} \to \mathcal{P}(S \times S)$, where each $\sim_A$ is an equivalence relation, and a *valuation* $V : P \to \mathcal{P}(S)$. The accessibility relation $\sim_{\mathcal{A}'}$ is defined as $(\bigcup_{A\in\mathcal{A}'} \sim_A)^*$. For $s \in S$, (M, s) is an *epistemic state*, also known as a pointed Kripke model.

Assume an epistemic model $M = \langle S, \sim, V\rangle$.

$$\begin{array}{ll}
M, s \models p & \text{iff } s \in V(p) \\
M, s \models \neg\varphi & \text{iff } M, s \not\models \varphi \\
M, s \models \varphi \wedge \psi & \text{iff } M, s \models \varphi \text{ and } M, s \models \psi \\
M, s \models K_A\varphi & \text{iff for all } t \in S : s \sim_A t \text{ implies } M, t \models \varphi \\
M, s \models C_{\mathcal{A}'}\varphi & \text{iff for all } t \in S : s \sim_{\mathcal{A}'} t \text{ implies } M, t \models \varphi \\
M, s \models [!\varphi]\psi & \text{iff } M, s \models \varphi \text{ implies } M|\varphi, s \models \psi
\end{array}$$

where the model restriction $M|\varphi = \langle S', \sim', V'\rangle$ is defined as $S' = \{s' \in S$ such that $M, s' \models \varphi\}$, $\sim'_A = \sim_A \cap\ (S' \times S')$ and $V'(p) = V(p) \cap S'$. Complete proof systems for this logic are presented in [13,3].

Our definition of a knowledge-based protocol consisting of announcements is a special case of the *knowledge-based program* à la Fagin *et al.* [6]. Instead of each agent choosing an action conditional on her knowledge, each agent chooses an announcement conditional on her knowledge. Although as a concept it is fairly obvious, it has not been singled out in the literature so far.

Definition 1 (Knowledge-based protocol). *A* knowledge-based protocol *for public announcement logic is a finite sequence of instructions determining sequences of announcements. Each agent A chooses an announcement $K_A\psi$ conditional on that agent's knowledge $K_A\varphi$. The chosen announcements are uttered simultaneously, i.e., an $|\mathcal{A}|$-tuple of preconditions of form $K_A\varphi_A$ determines an announcement $!\bigwedge_{A\in\mathcal{A}} K_A\psi_A$. The protocol is assumed common knowledge between all agents.*

In this work, we further assume that only one agent makes an announcement at the same time, that announcements are alternating between agents A and B, and that agent A starts the protocol. Given an initial epistemic model M, a *protocol execution* is a sequence of announcements determined by a knowledge-based protocol effecting successive model changes. The set of all such execution sequences is the extensional notion of protocols as in [12].

The protocol execution above consists of an announcement by A followed by announcement by B that can be modelled in public announcement logic as

$$\texttt{a\_announc} = K_A(012_A \vee 046_A \vee 136_A \vee 145_A \vee 235_A) \tag{1}$$

$$\texttt{b\_announc} = K_B 6_C \tag{2}$$

where n_i, for $0 \leq n \leq 6$ represents that agent i has the card n, and where nmk_i is an abbreviation for $n_i \wedge m_i \wedge k_i$. Note that A's announcement is indeed a function of A's knowledge, because $K_A 012_A$ entails $K_A(012_A \vee 046_A \vee 136_A \vee 145_A \vee 235_A)$, and that also B's announcement is a function of his knowledge.

We need some more card deal terminology in the continuation. Given players or agents A, B and C and a, b and c cards, with $a + b + c = d$. The cards are shuffled and dealt to the agents. A gets a cards, etc. This is a *card deal of size* (a, b, c). A condition holds for a protocol if it holds after every execution sequence of the protocol. A protocol is safe if it preserves common knowledge of ignorance of C. A protocol is A-informative if after termination it is common knowledge that A knows the card deal. A protocol is B-informative if after termination it is common knowledge that B knows the card deal. These three conditions are:

$$\texttt{b\_knows\_as} = \bigwedge_{n=0}^{d-1} \big(K_B n_A \vee K_B \neg n_A\big) \tag{3}$$

$$\texttt{a\_knows\_bs} = \bigwedge_{n=0}^{d-1} \big(K_A n_B \vee K_A \neg n_B\big) \tag{4}$$

$$\texttt{c\_ignorant} = \bigwedge_{n=0}^{d-1} \big(\neg K_C n_A \wedge \neg K_C n_B\big) \tag{5}$$

The protocol for $(3, 3, 1)$ in the introductory section is safe, A-informative, and B-informative because after A's public announcement of (1) it is true that

$$C_{ABC}\big(\texttt{b\_knows\_as} \wedge \texttt{c\_ignorant}\big) \tag{6}$$

and after B's public announcement of (2) it is true that

$$C_{ABC}\big(\texttt{b\_knows\_as} \wedge \texttt{a\_knows\_bs} \wedge \texttt{c\_ignorant}\big) \tag{7}$$

where we write C_{ABC} for $C_{\{A,B,C\}}$. Note that $C_{ABC}\big(\texttt{b\_knows\_as} \wedge \texttt{c\_ignorant}\big)$ holds whenever one of the hands in (1) is the actual hand. It is therefore sufficient to check that $\big(\texttt{b\_knows\_as} \wedge \texttt{c\_ignorant}\big)$ is a model validity after the announcement.

The model checker DEMO (for 'a Demo of Epistemic MOdelling') has been developed by van Eijck [18]. It is written in Haskell. DEMO allows the representation of epistemic models, performing updates with epistemic actions such as public announcements, and evaluating epistemic formulas in epistemic models or epistemic models resulting from update execution. The syntax of dynamic epistemic logic in DEMO is fairly similar to the standard language defined above, e.g., $K_B(8_B \wedge 9_B)$ is represented by `K b (Conj[Prop (R 8),Prop (R 9)]` (DEMO syntax only allows atoms called P, Q, or R, with number labels) and C_{ABC}`c_ignorant` by `(CK [a,b,c] c_ignorant)`. Instead of the single-pointed epistemic states (M, s) defined above, DEMO allows multi-pointed epistemic states: the set of designated points of the model stands for the current uncertainty about the actual state. If all points of the model are designated, checking

the truth of a formula in that model means checking if it is a model validity (we will use this in the continuation). In DEMO, the names of states must be natural numbers, starting with 0. Let the model M be `rus` and the actual state be `0`, then the verification that $M, s \models C_{ABC}$`c_ignorant` is represented in DEMO by

```
Main> isTrue (rus 0) (CK [a,b,c] c_ignorant)
True
```

The model resulting from updating `(rus 0)` with a public announcement of the formula `b_knows_as` is represented by `(upd (rus 0) (public b_knows_as))`. A multi-pointed model has a list of states instead of a single state, as in `(rus [0..224])`. The DEMO script employed in this paper is similar to the one in [17] that treats the $(3, 3, 1)$ case (but the protocol is very different).

3 There Is No Two-Step Protocol for $(4, 4, 2)$

We show that there does not exist a protocol in two steps, consisting of one announcement by A and one announcement by B, for $(4, 4, 2)$. We prove this using the upper and lower bounds for the number of hands in a safe announcement as in [1], namely by showing that the minimum number of hands is greater than the maximum number of hands. We recall that any announcement whatsoever must be equivalent to an announcement of alternative hands (see [16] — this is for the simple reason that the denotation of an announcement known to be true by the agent making it, is a union of equivalence classes for that agent; and an equivalence class is characterized by a hand of cards for that agent).

Consider an announcement consisting of alternative hands and containing the actual hand. An announcement is *more informative* if it consists of *fewer* hands. The most informative announcement consists of announcing the actual hand. The least informative announcement consists of announcing all hands. However, an announcement is *safer* if is consists of *more* hands. Clearly, the safest announcement consists of announcing all hands. But that announcement is uninformative. In [1] lower and upper bounds are given for the number of hands in an announcement. In the following, 'good' means 'safe and B-informative'.

- [**1, Prop.1**] The number of hands in a good announcement is at least

$$\frac{(a+b+c)(c+1)}{a}$$

- [**1, Prop.2**] The number of hands in a good announcement is at least

$$\frac{(a+b)(a+b+c)}{b(b+c)}$$

- [**1, Prop.3**] The number of hands in a good announcement is at most

$$\frac{(a+b+c)!(c+1)!}{(b+c)!(c+a+1)!}\left\lfloor\frac{a+c+1}{c+1}\right\rfloor$$

– [1, **Prop.4**] The number of hands in a good announcement is at most

$$\frac{(a+b+c)!(c+1)!}{a!(b+2c+1)!}\left\lfloor\frac{b+2c+1}{c+1}\right\rfloor$$

The two propositions for lower bounds should read 'at least the ceiling of', and those for higher bounds 'at most the floor of'. We obviously need integers. Which of the two lower bounds is sharper, depends on the card deal (a, b, c); and similarly for the two upper bounds. For $(4, 4, 2)$, [1, Prop.2] delivers a lower bound of 4 and [1, Prop.3] a higher bound of 12. That is not problematic. However, the other two propositions prove that:

Proposition 1. *There is no two-step protocol for* $(4, 4, 2)$.

Proof. For a lower bound, apply [1, Prop.1]. For $(a, b, c) = (4, 4, 2)$, the number of hands required is at least

$$\left\lceil\frac{(a+b+c)(c+1)}{a}\right\rceil = \left\lceil\frac{10\cdot 3}{4}\right\rceil = \lceil 7.5\rceil = 8.$$

For a higher bound, apply [1, Prop.4]. For $(a, b, c) = (4, 4, 2)$, the number of hands required is at most

$$\left\lfloor\frac{(a+b+c)!(c+1)!}{a!(b+2c+1)!}\lfloor\frac{b+2c+1}{c+1}\rfloor\right\rfloor = \left\lfloor\frac{10!3!}{4!9!}\lfloor\frac{9}{3}\rfloor\right\rfloor = \lfloor\frac{10}{4}\cdot 3\rfloor = \lfloor 7.5\rfloor = 7.$$

As a safe and B-informative announcement for $(4, 4, 2)$ should contain at least eight and at most seven hands, it cannot exist.

4 A Safe and Informative Announcement

The first step in the three-step protocol that we propose employs a *block design* [15]. We recall the definition of a t-design with parameters (v, k, λ) (also called a t–(v, k, λ) design). Relative to a v-set X (i.e., a set with $|X| = v$ elements) this is a collection of k-subsets of X called *blocks* with the property that every t-subset of X is contained in exactly λ blocks.

Announcements in protocols for card deals may be, but do not have to be, block designs. Given a deal with parameters (a, b, c), we take $v = d = a + b + c$, and $k = a$ (the blocks are A-hands). For $(3, 3, 1)$, the announcement 012 034 056 135 246 is *not* a design. The announcement 012 034 056 135 146 236 245 is a 2–$(7, 3, 1)$ design—each number pair occurs once. And it is a 1–$(7, 3, 3)$ design—each of the numbers 0 to 6 occurs three times.

Now for the card deal of size $(4, 4, 2)$. Consider the following 2–$(10, 4, 2)$ design $\mathcal{L}_A$ listed in the database Design DB [5].

0123 0145 0267 0389 0468 0579 1289 1367 1479 1568 2345 2478 2569 3469 3578

The set $\mathcal{L}_A$ is a 2–$(10, 4, 2)$ design, because each pair occurs twice. It is also a 1–$(10, 4, 6)$ design: each card occurs 6 times; but it is not a 3–design, e.g., triple 012 is in, but triple 018 is out. The block size 4 corresponds to the number of A's cards. Would this make a suitable announcement for A? Not every quadruple occurs in the design. What if A's hand of cards is 0124? Consider the following protocol.

Definition 2 (Protocol OneStep for A's announcement). *Let $\{i, j, k, l\}$ be A's hand of cards. Select a member $\{m, n, o, p\}$ from $\mathcal{L}_A$. (We do not assume that i, j, k, l or m, n, o, p are ordered by size, as in 0123.) Select a permutation π of 0..9 such that $\pi(i) = m$, $\pi(j) = n$, $\pi(k) = o$, $\pi(l) = p$. Let $\pi(\mathcal{L}_A)$ be the collection induced by permutation π. Player A announces $\pi(\mathcal{L}_A)$. (I.e., A says: "My hand of cards is one of $\pi(\mathcal{L}_A)$.")*

This is a knowledge-based protocol, because A knows her hand of cards. If 0123 is A's actual hand, the selected element of $\mathcal{L}_A$ is also 0123, and the selected permutation is the identity, then A's announcement corresponds to the following public announcement.

$$K_a(0123_a \vee 0145_a \vee 0267_a \vee 0389_a \vee 0468_a \vee 0579_a \vee 1289_a \vee 1367_a \vee 1479_a \vee 1568_a \vee 2345_a \vee 2478_a \vee 2569_a \vee 3469_a \vee 3578_a) \tag{8}$$

We now show that this is a safe announcement. There are two ways to go about this: a combinatorial proof, and more logical proof, namely by means of dynamic epistemic model checking.

Proposition 2. *Protocol* OneStep *is safe.*

Proof. The combinatorial proof uses the two combinatorial safety requirements formulated in [1]. They are:

- **CA2.** For every c-set X the members of $\mathcal{L}$ avoiding X have empty intersection.
- **CA3.** For every c-set X the members of $\mathcal{L}$ avoiding X have union consisting of all cards except those of X.

For $(4, 4, 2)$, and $\mathcal{L} = \mathcal{L}_A$, CA2 guarantees that, if C holds the two cards in X, then after A announces $\mathcal{L}_A$, C does not learn one of A's cards (if a card occurs in all A-hands that C still considers possible, then A *must* have that card). CA3 guarantees that C does not learn one of B's cards (if a card does not occur in all A-hands that C still considers possible, and C also does not have that card, then B *must* have that card).

Let $X = \{0, 1\}$ ($X = 01$). From the fifteen hands in $\mathcal{L}_A$, there are six containing 0 and six containing 1 (it's a 1–design with $\lambda = 6$)—of which two contain the pair 01 (it's a 2-design with $\lambda = 2$): ten hands contain 0 or 1. Consider the five remaining hands 2345 2478 2569 3469 3578. Some numbers occur three times (2, 3, 4, and 5) and some twice only (6, 7, 8, and 9). But all numbers occur at least once—so they are contained in the union of these five hands—and at least once not—so they are not in the intersection of these five hands. Therefore, for $X = 01$ the conditions CA2 and CA3 are satisfied.

This holds not just for 01 but for any ij. There are always five hands that do not contain i and j, because of the design properties. If a card $k \neq i, j$ were to occur in all those five hands, it would only occur once in the remaining ten (as $\mathcal{L}_A$ is a 1–design wherein each card occurs 6 times). But there must be two occurrences of the pair ik, as $\mathcal{L}_A$ is a 2–design wherein each pair occurs twice. So no card k occurs in all remaining hands. However, if a card $k \neq i, j$ were to occur in none of those five hands, it would occur six times in the ten hands containing i or j. One can easily see that the maximum number of allowed k-occurrences is four, otherwise there would be more than two ik or more than two jk pairs. So it can also not be the case that k does not occur in any of the remaining hands.

Alternatively, we can achieve this result by model checking. The requirement is then that after $\mathcal{L}_A$ it is common knowledge, given the actual deal of cards, that C is ignorant. This requirement is satisfied if it is a model validity that C is ignorant. Therefore, the proof is:

```
Main> isTrue (rus [0..224]) c_ignorant
True
```

This succinct line hides the computation behind it. Also, the structure of the epistemic model resulting from $\mathcal{L}_A$ will serve us in extending Protocol OneStep to a full protocol—we did not mention so far what B learns from $\mathcal{L}_A$. The next subsection presents this model, and some statistics on the model checking.

5 The Model before and after the First Announcement

The initial model has $\binom{10}{4}\binom{6}{4} = 3150$ states/deals. After A's announcement of 15 hands it is reduced to $15 \cdot \binom{6}{4} = 225$ states, that are labeled from 0 to 224. Appendix A shows the equivalence classes for the three agents in this model. For example, it indicates that in the deals 0..14 agent A has the same hand (another part of the DEMO specification, not shown, nails down which hand).

Figure 1 depicts a part of the epistemic model after A's announcement. This is a $\sim_{BC}$-equivalence class (we write $\sim_{BC}$ for $\sim_{\{B,C\}}$). All states in this class can be distinguished by A. The model consists of 15 such subgraphs (an A-class also consisting of 15 states, one in each of these subgraphs). In the picture, the solid lines represent accessibility for C and the dashed lines accessibility for B. Note that in 12 of the 15 states B knows the deal, and that in the remaining three ($\{0, 59, 104\}$) B considers three possible hands for C. The numbers in the points refer to the deals that appear in the equivalence classes of Appendix A. The state named 0 represents the deal 0123|4567|89, i.e., where A's hand is $\{0, 1, 2, 3\}$, B's hand is $\{4, 5, 6, 7\}$ and C's hand is $\{8, 9\}$.

The verification in epistemic logic of Proposition 2 was represented by

```
Main> isTrue (rus [0..224]) c_ignorant
True
```

and to similar effect we can check

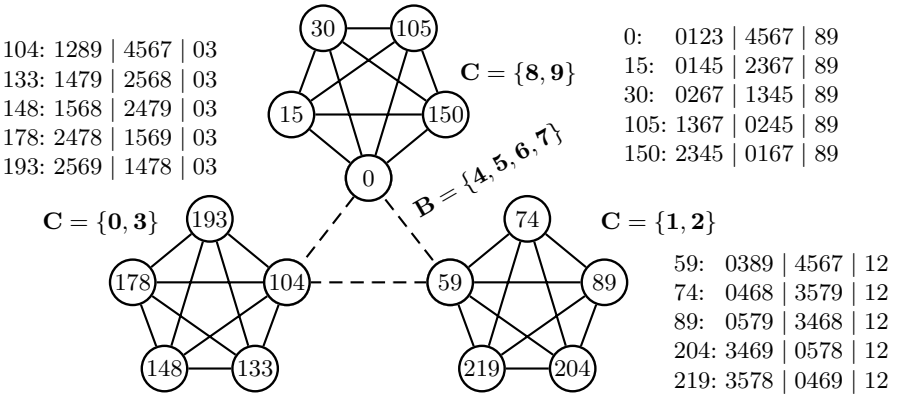


Fig. 1. The epistemic model for B and C after A's announcement

```
Main> isTrue (rus 0) CK [a,b,c] c_ignorant
True
```

The computational cost of this check depends on some inessential but for practical purposes important parameters. Table 1 shows the runtime required to check Proposition 2 with DEMO in a 2.26 GHz processor, running GHCi, version 6.12.3 over Ubuntu Linux 9.10. Two formulas that represent the ignorance of C about card n are provided. The first is the one used in (5). The second checks C's ignorance only for cards that C does not hold. Both formulas are equivalent in the model. There are also two ways for checking C's ignorance. The first one (central column) is by checking that ignorance of C is common knowledge at some point in the model (`0` in this case). The second one (right) is by checking that C's ignorance is a model validity. As the model is connected, both ways are equivalent in the model. The minimum time corresponds to checking that (5) is a model validity. The improvement for the lower, conditional formula for C's ignorance is only when checking common knowledge. We have also checked `c_ignorant` (5) in the (initial) model prior to A's announcement, and the cost of announcing $\mathcal{L}_A$ (8) in that initial model. This takes 13.2 seconds.

Table 1. DEMO's runtime in milliseconds to check Proposition 2 in several ways

C's ignorance about n	(`rus`[0]) $\models C_{ABC}$`c_ignorant`	(`rus`) $\models$ `c_ignorant`
$\neg K_C n_A \wedge \neg K_C n_B$	629	256
$\neg n_C \rightarrow \neg K_C n_A \wedge \neg K_C n_B$	544	399

We have used the Haskell script in Appendix B to demonstrate that the epistemic model after executing Protocol OneStep indeed consists of fifteen $\sim_{BC}$-connected subgraphs like the one in Figure 1, where all points in one such subgraph are all different for A. Function `(goodGraph n)` checks that every point

n in the epistemic model after the execution of Protocol OneStep belongs to a $\sim_{BC}$–graph like that of Figure 1.

```
Main> foldl1 (&&) (map goodGraph [0..224])
True
```

By showing the structure of this model we have demonstrated (or rather corroborated) two aspects of Protocol OneStep. Firstly, given an actual hand of A, the actual announcement is not $\mathcal{L}_A$ but $\pi(\mathcal{L}_A)$. That does not change the model structure, as this merely involves renaming atoms. In fact, we have done this anyway: 'B holds 8' is not an atom 8_B in DEMO, but some `R 8` (see the previous section). So this involves only further renaming. Secondly, the point of the structure need not be card deal 0123|4567|89, as here, but can be another deal of cards wherein A holds another hand of cards in $\mathcal{L}_A$. But the model looks the same from any such different perspective. This is as expected: for any two hands X and Y of the design, there are (many) permutations π such that $\pi(X) = Y$ and $\pi(\mathcal{L}_A) = \mathcal{L}_A$, in other words, these permutations induce automorphisms of the design. The remaining sections will not refer to design theory phenomena — the further extensions of the protocol are not designs.

6 A Three Step Protocol for (4, 4, 2)

We now present a three-step protocol for (4, 4, 2) that extends Protocol OneStep.

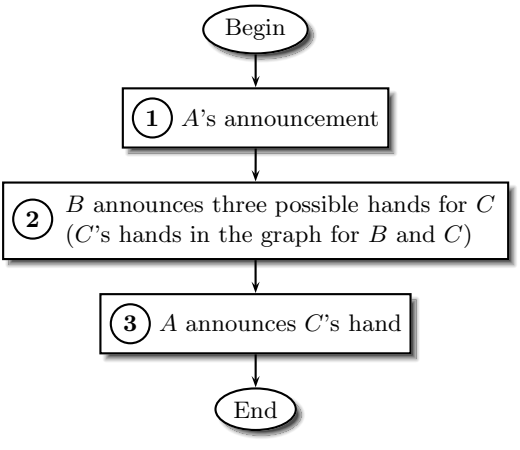


Fig. 2. Protocol ThreeSteps for (4,4,2)

Definition 3 (Protocol ThreeSteps).

Step 1. *A announces (a permutation of)* (8)*. Call this* $\mathcal{L}_A$*, as before.*
Step 2. *B announces that C has one of three hands. These are the three C hands in the* $\sim_{BC}$*–connected part of the model* $M|\mathcal{L}_A$*.*

Step 3. *A announces the hand of C.*

Proposition 3. *Protocol* ThreeSteps *is safe.*

Proof. The safety of Step 1 is guaranteed by Proposition 2. The effect of Step 2 is to reduce the epistemic model to just one $\sim_{BC}$-class like the one depicted in Figure 1. As it contains three complete $\sim_C$–classes, the ignorance of C remains common knowledge. Step 3 reduces the model to a single $\sim_C$–class. So, again, common knowledge of ignorance is preserved.

Proposition 4. *Protocol* ThreeSteps *is A-informative.*

Proof. Agent A knows the deal after Step 2. All $\sim_A$-classes are then singletons.

Proposition 5. *Protocol* ThreeSteps *is B-informative.*

Proof. Agent B knows the deal after Step 3: a $\sim_C$-class consists of five singleton $\sim_B$-classes.

7 A Probabilistic Protocol

Consider a 15-state $\sim_{BC}$-class, as in Figure 1. It seems appealing for B to announce the card deal in the twelve cases where he knows the card deal, and only to continue the protocol in the three cases where he does not know that. This is not safe (Figure 3), but we can make it safe (Figure 4 and Definition 4).

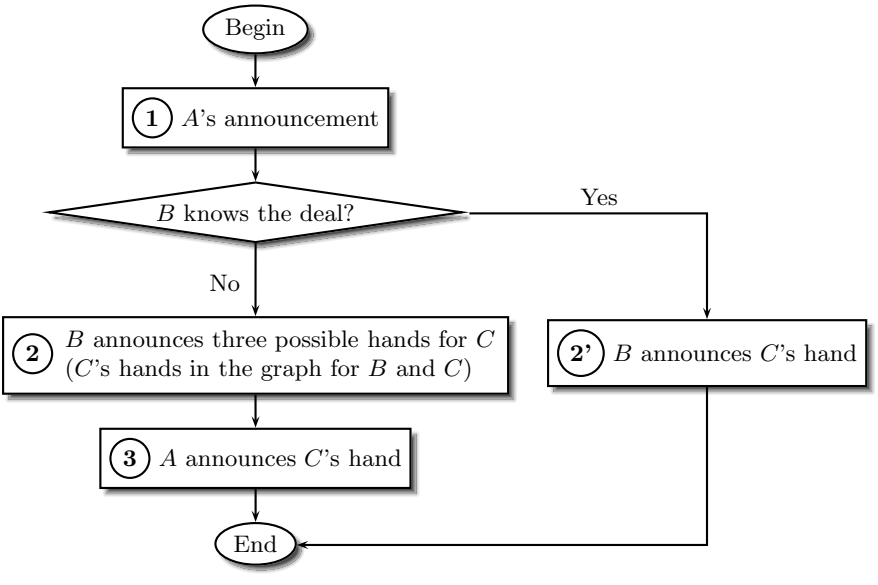


Fig. 3. An unsafe protocol for (4,4,2)

Look at Figure 3. If B knows the deal after A's announcement at Step 1, B announces C's cards and the protocol finishes in just two steps. It is Step 2' in Figure 3. Otherwise, the protocol continues as in Figure 2 and takes three steps.

After every step in Figure 3 agent C remains ignorant about any card of A and B, but this becomes different when C also knows the protocol. Kerckhoffs's Principle [8] says that the protocol should be common knowledge: the design of a security protocol should be done under the assumption that an intruder knows everything about the protocol except the key. For cards communication this means that we may assume that C knows everything about the protocol except the hands of A and B; i.e., C knows the flowchart in Figure 3. If B announces C's cards in Step 2 he implicitly announces `b_knows_as` and if B announces some alternative hands for C he implicitly announces the negation of that. Fortunately, we have that

```
Main> isTrue (upd (rus [0..224]) (public b_knows_as)) c_ignorant
True
```

This is not trivial (there remain four indistinguishable card deals for C, and that is enough in this case to keep her ignorant) but unfortunately, we also have that

```
Main> isTrue (upd (rus [0..224]) (public (Neg b_knows_as)))
   (Neg c_ignorant)
True
```

When C is informed about B's ignorance, she learns that the actual deal is the only one in C's pentagon (see Figure 1) where B does not know the deal: A gains full knowledge of the card deal.

The protocol in Figure 3 is unsafe. The problem is that B performs Step 2 *only* if he does not know the card deal. The link is broken in Protocol ProbSteps—see Figure 4.

Definition 4 (Protocol ProbSteps).

Step 1. *A announces (a permutation of) $\mathcal{L}_A$.*
Step 2'. *If B knows the deal then with probability $p < 1$ B announces C's hand.*
Step 2. *If B does not know the deal after Step 1 or if B did not execute Step 2', then B announces the three C-hands in the $\sim_{BC}$–class.*
Step 3. *If Step 2 was executed, A announces the hand of C.*

Proposition 6. *Protocol* ProbSteps *is safe.*

Proof. In case that the protocol follows the sequence of Steps 1, 2, 3, it is like Protocol ThreeSteps, so it is safe (Proposition 3). Note that when B performs Step 2, he is not implicitly announcing his ignorance of C's cards, because there is non-zero probability that B performs Step 2 when he knows the deal. When the protocol consists of Steps 1 and 2' it is also safe, because (as before)

```
Main> isTrue (upd (rus [0..224]) (public b_knows_as)) c_ignorant
True
```

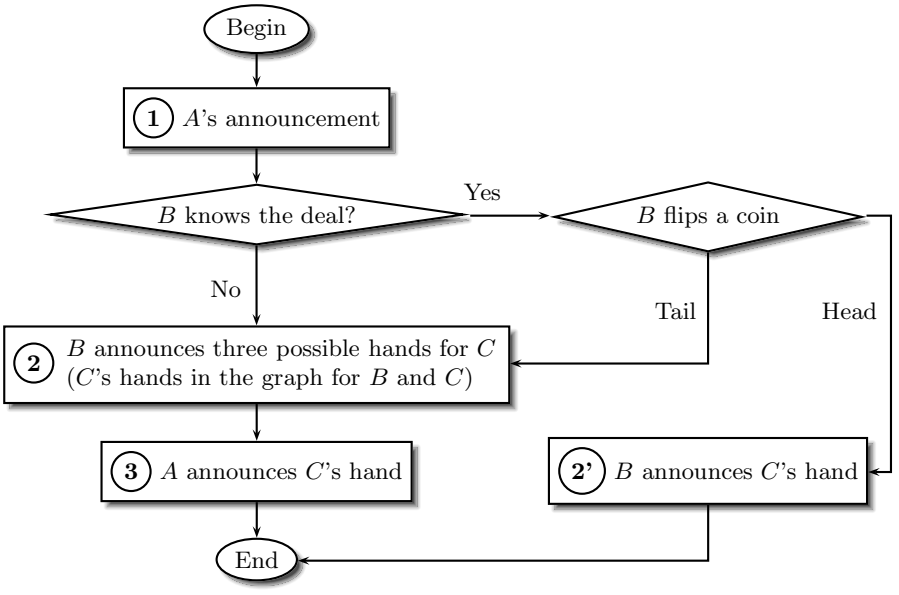


Fig. 4. A probabilistic version of the protocol

The following two propositions are obvious.

Proposition 7. *Protocol* ProbSteps *is A-informative.*

Proposition 8. *Protocol* ProbSteps *is B-informative.*

Proposition 9. *The average length of Protocol* ProbSteps *is* $3 - 0.8p$.

Proof. Given that the permutation of (8) announced by A at Step 1 is randomly chosen, then on the further assumption that all cards deals are equally likely (random probability distribution), the probability that B knows the deal after A's announcement is 0.8. (B knows the deal in four out of five states in every $\sim_C$-pentagon.) As p is the probability of then announcing the deal, the protocol has length 2 with probability $0.8p$, and it has length 3 with probability $1 - 0.8p$. So the average length of the protocol is

$$3(1 - 0.8p) + 2(0.8p) = 3 - 0.8p$$

When p approaches 1, this probability approaches 2.2. (It cannot be 1, as the protocol is then unsafe again.)

The value of p in Proposition 9 is common knowledge. Agent C can apply Bayes' Theorem when B performs Step 2, to calculate the probability for B not knowing the deal:

$$P(\neg b\_knows\_all|Step\ 2) = \frac{P(Step\ 2|\neg b\_knows\_all)P(\neg b\_knows\_all)}{P(Step\ 2)} = \\ = \frac{1 \times 0.2}{0.2 + 0.8(1-p)} = \frac{0.2}{1 - 0.8p}$$

This is the probability for C to correctly guess the deal when Step 2 is performed with B is still ignorant.

It remains safe 'in principle' to have a large p value but that also increases the probability that the eavesdropper C correctly *guesses* the deal. We can reduce the probability of guessing if we consider a parallel execution of n instances of the protocol, where the secret is the conjunction of the n deals. The probability of guessing correctly is then multiplied to the power of n, and can therefore be reduced as much as we want.

8 Conclusion

For a card deal of size $(4, 4, 2)$ we have shown that there does not exist a protocol of two steps for the four-card players to inform each other safely of their hands of cards. We have presented a three-step Protocol ThreeSteps that achieves that goal. We verified the properties of that protocol by combinatorial means, using properties of block designs, and also by model checking in DEMO. Future work involves investigating other designs, in order to find protocols of at least three steps for card deals of size (a, b, c) for $c \geq 2$ (for $c = 1$ and two steps, a full characterization is known). This will advance the characterization of card deals for which such exchanges of secrets are possible.

Acknowledgment

We thank three anonymous CLIMA reviewers for their comments. Hans van Ditmarsch is also affiliated to IMSC (Institute of Mathematical Sciences Chennai), India, as associated researcher. We thank the participants of the Sevilla logic seminar for their interaction.

References

1. Albert, M.H., Aldred, R.E.L., Atkinson, M.D., van Ditmarsch, H., Handley, C.C.: Safe communication for card players by combinatorial designs for two-step protocols. Australasian Journal of Combinatorics 33, 33–46 (2005)
2. Albert, M.H., Cordón-Franco, A., van Ditmarsch, H., Fernández-Duque, D., Joosten, J.J., Soler-Toscano, F.: Secure communication of local states in interpreted systems. In: Abraham, A., Corchado, J.M., González, S.R., De Paz Santana, J. (eds.) International Symposium on Distributed Computing and Artificial Intelligence. Advances in Intelligent and Soft Computing, vol. 91, pp. 117–124. Springer, Heidelberg (2011)
3. Baltag, A., Moss, L.S., Solecki, S.: The logic of public announcements, common knowledge, and private suspicions. In: Gilboa, I. (ed.) Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 1998), pp. 43–56 (1998)
4. Dixon, C.: Using temporal logics of knowledge for specification and verification–a case study. Journal of Applied Logic 4(1), 50–78 (2006)

5. Dobcsányi, P.: Design db (2011), http://batman.cs.dal.ca/~peter/designdb/
6. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
7. Fischer, M.J., Wright, R.N.: Bounds on secret key exchange using a random deal of cards. Journal of Cryptology 9(2), 71–99 (1996)
8. Kerckhoffs, A.: La cryptographie militaire. Journal Des Sciences Militaires IX, 5–38, 161–191 (1883)
9. Kirkman, T.: On a problem in combinations. Camb. and Dublin Math. J. 2, 191–204 (1847)
10. Koizumi, K., Mizuki, T., Nishizeki, T.: Necessary and sufficient numbers of cards for the transformation protocol. In: Chwa, K.-Y., Munro, J.I. (eds.) COCOON 2004. LNCS, vol. 3106, pp. 92–101. Springer, Heidelberg (2004)
11. Luo, X., Su, K., Sattar, A., Chen, Y.: Solving sum and product riddle via bdd-based model checking. In: Web Intelligence/IAT Workshops, pp. 630–633. IEEE, Los Alamitos (2008)
12. Parikh, R., Ramanujam, R.: A knowledge based semantics of messages. Journal of Logic, Language and Information 12, 453–467 (2003)
13. Plaza, J.A.: Logics of public communications. In: Emrich, M.L., Pfeifer, M.S., Hadzikadic, M., Ras, Z.W. (eds.) Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems: Poster Session Program, pp. 201–216. Oak Ridge National Laboratory (1989)
14. Stiglic, A.: Computations with a deck of cards. Theoretical Computer Science 259(1-2), 671–678 (2001)
15. Stinson, D.R.: Combinatorial Designs – Constructions and Analysis. Springer, Heidelberg (2004)
16. van Ditmarsch, H.: The Russian cards problem. Studia Logica 75, 31–62 (2003)
17. van Ditmarsch, H., van der Hoek, W., van der Meyden, R., Ruan, J.: Model checking russian cards. Electronic Notes in Theoretical Computer Science 149, 105–123 (2006); Presented at MoChArt 2005, Model Checking in Artificial Intelligence
18. van Eijck, J.: DEMO — a demo of epistemic modelling. In: van Benthem, J., Gabbay, D., Löwe, B. (eds.) Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop. Texts in Logic and Games, vol. 1, pp. 305–363. Amsterdam University Press, Amsterdam (2007)
19. van Otterloo, S., van der Hoek, W., Wooldridge, M.: Model checking a knowledge exchange scenario. Applied Artificial Intelligence 18(9-10), 937–952 (2004)
20. Wang, Y.: Epistemic Modelling and Protocol Dynamics. PhD thesis, Universiteit van Amsterdam (2010)

A Equivalence Classes after *A*'s Announcement

```
cA = [[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14],[15,16,17,18,19,20,21,22,23,
24,25,26,27,28,29],[30,31,32,33,34,35,36,37,38,39,40,41,42,43,44],[45,46,
47,48,49,50,51,52,53,54,55,56,57,58,59],[60,61,62,63,64,65,66,67,68,69,
70,71,72,73,74],[75,76,77,78,79,80,81,82,83,84,85,86,87,88,89],[90,91,92,
93,94,95,96,97,98,99,100,101,102,103,104],[105,106,107,108,109,110,111,
112,113,114,115,116,117,118,119],[120,121,122,123,124,125,126,127,128,
129,130,131,132,133,134],[135,136,137,138,139,140,141,142,143,144,145,
146,147,148,149],[150,151,152,153,154,155,156,157,158,159,160,161,162,
163,164],[165,166,167,168,169,170,171,172,173,174,175,176,177,178,179],
[180,181,182,183,184,185,186,187,188,189,190,191,192,193,194],
[195,196,197,198,199,200,201,202,203,204,205,206,207,208,209],
[210,211,212,213,214,215,216,217,218,219,220,221,222,223,224]]

cB = [[0,59,104],[1],[2],[3],[4],[5,44,119],[6],[7],[8],[9],[10],[11],
[12],[13],[14,29,164],[15],[16,87,132],[17],[18],[19,72,147],[20],[21],
[22],[23],[24],[25],[26],[27],[28],[30],[31,82,191],[32],[33],[34,67,
176],[35],[36],[37],[38],[39],[40],[41],[42],[43],[45],[46,78,220],[47],
[48],[49,63,205],[50],[51],[52],[53],[54],[55],[56],[57],[58],[60],[61],
[62],[64],[65],[66],[68],[69],[70],[71],[73],[74],[75],[76],[77],[79],
[80],[81],[83],[84],[85],[86],[88],[89],[90],[91],[92,141,186],[93,126,
171],[94],[95],[96],[97],[98],[99],[100],[101],[102],[103],[105],[106],
[107,139,217],[108,124,202],[109],[110],[111],[112],[113],[114],[115],
[116],[117],[118],[120],[121],[122],[123],[125],[127],[128],[129],[130],
[131],[133],[134],[135],[136],[137],[138],[140],[142],[143],[144],[145],
[146],[148],[149],[150],[151],[152,170,215],[153,185,200],[154],[155],
[156],[157],[158],[159],[160],[161],[162],[163],[165],[166],[167],[168],
[169],[172],[173],[174],[175],[177],[178],[179],[180],[181],[182],[183],
[184],[187],[188],[189],[190],[192],[193],[194],[195],[196],[197],[198],
[199],[201],[203],[204],[206],[207],[208],[209],[210],[211],[212],[213],
[214],[216],[218],[219],[221],[222],[223],[224]]

cC = [[0,15,30,105,150],[1,16,60,135,151],[2,17,152,180,195],[3,18,153,
165,210],[4,19,75,120,154],[5,20,45,90,155],[6,31,61,106,166],[7,32,107,
121,196],[8,46,62,91,197],[9,47,92,122,167],[10,33,108,136,211],[11,34,
76,109,181],[12,48,93,137,182],[13,49,77,94,212],[14,35,50,95,110],[21,
36,63,138,168],[22,37,78,123,183],[23,64,96,139,184],[24,79,97,124,169],
[25,66,111,141,213],[26,81,112,126,198],[27,51,67,142,199],[28,52,82,127,
214],[29,69,84,129,144],[38,65,98,125,170],[39,80,99,140,185],[40,70,156,
171,216],[41,85,157,186,201],[42,57,72,173,203],[43,58,87,188,218],[44,
73,88,174,189],[53,68,113,128,200],[54,83,114,143,215],[55,71,158,187,
202],[56,86,159,172,217],[59,74,89,204,219],[100,146,162,191,206],[101,
131,163,176,221],[102,117,132,177,207],[103,118,147,192,222],[104,133,
148,178,193],[115,145,160,175,220],[116,130,161,190,205],[119,134,149,
208,223],[164,179,194,209,224]]
```

B A Haskell Script

```
import List
goodGraph :: Integer -> Bool
goodGraph n =
  let
    -- cc1: C-class with n (a pentagon)
    cc1 = head (take 1 (filter (\x -> (elem n x)) cC))
    -- bc1: points B-accessible with cc1 (4 singlet. + 1 triang.)
    bc1 = foldl1 (++)
          (take 5 (filter (\x -> (intersect x cc1) /= []) cB))
    -- cc2: C-classes with points in bc1 (three pentagons)
    cc2 = filter (\x -> (intersect x bc1) /= []) cC
    -- cc3: points in the three pentagons
    cc3 = foldl1 (++) cc2
    -- bc2: B-classes with points in cc2 (12 singlet. + 1 triang.)
    bc2 = filter (\x -> (intersect x cc3 /= [])) cB
    -- na: number of A-classes with elements in the graph
    na  = length (filter (\x -> (intersect x cc3 /= [])) cA)
  in
   -- 1: all the A-classes are represented (1 point per class)
   na == 15 &&
   -- 2: in the B-classes there is just one triangle
   length (filter (\x -> (length x) == 3) bc2) ==  1 &&
   -- 3: in the B-classes there are 12 singletons
   length (filter (\x -> (length x) == 1) bc2) == 12 &&
   -- 4: in the C-classes there are 15 points (3 pentagons)
   length cc3 == 15
```

A Modal Framework for Relating Belief and Signed Information

Emiliano Lorini[1], Laurent Perrussel[2], and Jean-Marc Thévenin[2]

[1] IRIT
Toulouse - France
emiliano.lorini@irit.fr
[2] IRIT - Université de Toulouse
Toulouse - France
{laurent.perrussel,jean-marc.thevenin}@univ-tlse1.fr

Abstract. The aim of this paper is to propose a modal framework for reasoning about signed information. This modal framework allows agents to keep track of information source as long as they receive information in a multi-agent system. Agents gain that they can elaborate and justify their own current belief state by considering a reliability relation over the sources of information. The belief elaboration process is considered under two perspectives: (i) from a static point of view an agent aggregates received signed information according to its preferred sources in order to build its belief and (ii) from a dynamic point of view as an agent receives information it adapts its belief state about signed information. Splitting the notions of beliefs and signed statement is useful for handling the underlying trust issue: an agent believes some statement because it may justify the statement's origin and its reliability.

1 Introduction

An agent embedded in a multi-agent system gets information from multiple origins; it captures information from its own sensors or, it may receive messages issued by other agents through some communication channels. Based on this set of basic information the agent then defines its belief state and performs actions [25]. As long as it gets information, the agent has to decide what it should believe and also which beliefs are no longer available [14,16,4]. In order to decide which beliefs should hold, the agent needs some criteria. A common criterion consists of handling a reliability relation about information origins [10,6]. According to its opinion about the reliability of the information source, the agent decides to adopt the received piece of information or not.

Keeping track of information and its origin is a key issue for trust characterization. Agents can justify their beliefs: agent a believes φ because agent b has provided φ and b is reliable [19]. In addition, keeping track of agents involved in information broadcasting enables agents to evaluate, from their own point of view, whether they are all reliable, i.e. believable [21].

Although several works have been made in order to show how an agent can merge information issued from multiple origins [27,23], very few works have focused on the explicit representation of the origins of information [24,19] in the context of BDI-based

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 58–73, 2011.

systems with communication actions. This explicit representation is necessary since it represents the underlying rationale of agents' beliefs.

The aim of this article is to propose a modal framework to describe agent's belief state while preserving information source. The underlying purpose is to avoid the syntax dependency of the work proposed in [24]. We formalize the transition from information to belief and consider the dynamics of information, that is how the agent adapts its belief state about signed information with respect to new incoming information (messages).

The dynamics is usually described in terms of performative actions based on KQML performatives [13] or speech acts [7,8]. Hereafter, we propose to consider a *tell* action as a private announcement from one agent (the sender of the message) to a second agent (the receiver of the message). A private announcement enables to stress up how an agent "restricts" its belief state as it receives information. More precisely, it shrinks the space of possible information states with respect to information sent by its sources and then according to that space, it builds up its beliefs.

The article is structured as follows: In section 2, we present the intuitive meaning of signed information and belief state. Next in section 3, we present the technical details of the modal logic framework. In section 4, we then formalize two intuitive and common policy for relating signed information and belief which consist in the adoption as belief of (i) information commonly signed by some set of agents and (ii) all consistent signed information. In section 5, we extend the modal framework with actions of the form "agent a tells to agent b that a certain fact φ is true". In section 6, we apply our results on an example. We conclude the paper in section 7 by summing up the contribution and considering some open issues.

2 Setting the Framework

Handling the source of information leads to the notion of *signed statement*, that is some statement is true according to some source. From a semantics perspective, we want to be able to represent for an agent, what are the possible states according to information received from each source (w.r.t. some initial state of affairs).

2.1 Representing Signed Statements

Signed information can be represented through Kripke models using one accessibility relation per source of information. Relation denoted S_b describes all the states reachable from some initial state according to information issued from source b. The belief states of each agent, can in turn be represented using one accessibility relation per agent. Relation denoted B_a provides all the possible belief states agent a can reach from some initial state. Figure 1 represents signed states that can be reached from the four possible belief states of agent a, namely w_1, w_2, w_3 and w_4, according to information issued from agents b and c.

Modal statement $\mathsf{Sign}(b, p)$ stands for statement p is true in some state according to source b. Intuitively $\mathsf{Sign}(b, p)$ is true in state w if statement p holds in all states reachable from w through relation S_b. Let us have a look at the example presented Figure 1. Statements p or $\neg p$ and q or $\neg q$ are mentioned between brackets above each

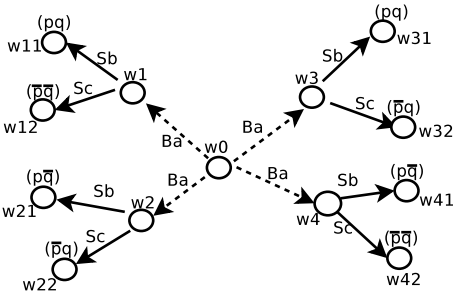


Fig. 1. Relating belief state and signatures

world reachable through S_b or S_c in which they hold. Following S_b in state w_1 agent a can only reach state w_{11} and statement p holds in this state. Consequently $\mathsf{Sign}(b, p)$ is true for in state w_1. Following S_c from state w_1, agent a can only reach state w_{12} where statement $\neg p$ hold. Consequently, in state w_1 $\mathsf{Sign}(c, \neg p)$ is true.

In this framework, signed statements represent the rationales for beliefs. It is possible to interpret formulas such as $\mathsf{Bel}(a, \mathsf{Sign}(b, p))$ which stands for agent a believes that agent b signs statement p by checking that in all possible belief states of agent a $\mathsf{Sign}(b, p)$ is true. According to Figure 1, we get $\mathsf{Bel}(a, \mathsf{Sign}(b, p) \wedge \mathsf{Sign}(c, \neg p))$ since $\mathsf{Sign}(b, p)$ and $\mathsf{Sign}(c, \neg p)$ hold in w_1, w_2, w_3 and w_4.

2.2 Preferences over Information Sources

In order to handle possibly mutually inconsistent signed statements, agents consider extra information stating which signed statement they prefer. Agents may determine themselves their preferences by considering the sources of information [10,23], temporal aspects or the topics of the statements [9].

In this paper, for the sake of conciseness and following numerous contributions such as [6], we propose to consider extra information representing a preorder relation denoted $\leqslant$ defined over the reliability of sources of information: $a \leqslant b$ stands for a is at least as reliable as b and $a < b$ stands for a is more reliable than b. We assume that the agents consider information about only one topic and handling competencies or different kinds of reliability (such as suggested in [5]) is out of the scope of this paper.

According to the example presented Figure 1 agent a believes that agent b signs p and agent c signs $\neg p$. Considering extra information $\mathsf{Bel}(a, c < b)$ standing for agent a believes that agent c is more reliable than agent b, agent a should adopt statement $\neg p$ as a belief. In semi-formal terms, we get that:

$$\mathsf{Bel}(a, (\mathsf{Sign}(b, p) \wedge \mathsf{Sign}(c, \neg p) \wedge c < b)) \Rightarrow \mathsf{Bel}(a, \neg p) \qquad \textbf{(Adpt)}$$

Using extra-information on the reliability of sources and considering signed statements rather than statements, the problem of belief change [14] is almost rephrased in terms close to the ones used in belief merging [20,18]. Reliability order over sources of information enables us to stratify signed information and then by merging this stratified information in a consistent way the agents get "justified" beliefs [3].

By splitting the two concepts of beliefs and signed statements, we are not limited to axiom schema (**Adpt**). It is possible to define multiple policies to define agent's beliefs: lexicographic aggregation; focus on statements signed by some specific agents; evaluating the truthfulness of signed information (does the receiver believes that the sender believes the signed statement). This dichotomy also avoids defining belief state (private state) only by considering belief about other agent's belief (also a private state). We adopt a different principle which enables to show the transition from a public characteristic (signed information and tell actions) to a private one (belief).

2.3 Representing Tell Statements

Let us consider performative $Tell(b, a, q)$ which stands for agent b tells to agent a that q is true. We interpret this performative as a private announcement [29] rather than with help of actions and transitions between states. After performative $Tell(b, a, q)$ agent a believes that b signs q. Consequently, all belief states for which $\mathsf{Sign}(b, q)$ is false are no more reachable for agent a after $Tell(b, a, q)$.

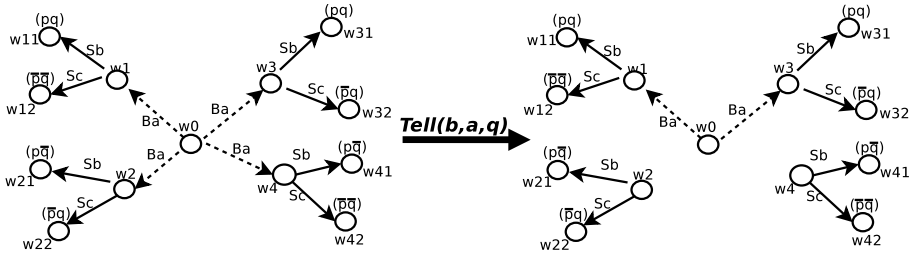


Fig. 2. Agent b tells q to agent a

Figure 2 illustrates how agent a's belief state changes after receiving performative $Tell(b, a, q)$. In the initial situation (the left part of the figure), agent a believes $\mathsf{Sign}(b, p)$ but does not believe $\mathsf{Sign}(b, q)$ since q does not hold in states w_{21} and w_{41} that can be reached through S_b from belief states w_2 and w_4. After receiving agent b's message (right part of the figure), belief state w_2 and w_4 are no longer reachable through B_a and agent a believes $\mathsf{Sign}(b, p)$ and $\mathsf{Sign}(b, q)$. Notice that agent a still believes $\mathsf{Sign}(c, \neg p)$ and still does not believe $\mathsf{Sign}(c, q)$ and $\mathsf{Sign}(c, \neg q)$. In other words, agent a's beliefs about information signed by c have not changed. At this stage, using extra information $\mathsf{Bel}(a, c < b)$ agent a can aggregate its signed beliefs as follows: $\mathsf{Bel}(a, \neg p)$ and $\mathsf{Bel}(a, q)$. Indeed, statement q issued from b is not contradicted by the more reliable source c.

Private announcements stress up the information gathering aspect: possible worlds accessible through relation B_a represent the ignorance of agent B_a and by shrinking this set of possible believable worlds, we represent how agent a gains information. Notice that this way of handling the dynamics entails as a drawback that agent's belief cannot always be consistent: updating a model might lead to a model where seriality cannot be guaranteed [29].

3 Formal Framework

In this section, we focus on *signatures*, *beliefs* and *preferences*; *tell* actions will be introduced later. The proposed language for reasoning about these three notions is a restricted first order language which enables quantification over agent ids. Quantification allows agents to reason about anonymous signatures. Logical language $\mathcal{L}$ is based on doxastic logic. Modal operator Bel represents beliefs: $\mathsf{Bel}(a, \varphi)$ means agent a believes $\mathcal{L}$-formula φ. Modal operator Sign represents signed statements: $\mathsf{Sign}(t, \varphi)$ means t (an agent id or a variable of the agent sort) signs statement φ. In order to represent an agent's opinion about reliability, we introduce the notation $a \leqslant b$ which stands for: agent a is said to be at least as reliable as b.

Definition 1 (Syntax of $\mathcal{L}$). *Let $\mathcal{P}$ be a finite set of propositional symbols. Let $\mathcal{A}$ be a finite set of agent ids. Let $\mathcal{V}$ be a set of variables s.t. $\mathcal{A} \cap \mathcal{V} = \emptyset$. Let $\mathcal{T} = A \cup \mathcal{V}$ be the set of agent terms. The set of formulas of the language $\mathcal{L}$ is defined by the following BNF:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{Sign}(t, \varphi) \mid \mathsf{Bel}(a, \varphi) \mid \forall x\varphi \mid t \leqslant t'$$

where $p \in \mathcal{P}$, $t \in \mathcal{T}$, $t' \in \mathcal{T}$, $a \in \mathcal{A}$ and $x \in \mathcal{V}$.

Writing $a < b$ stands for a is strictly more reliable than b: $a \leqslant b \wedge \neg(b \leqslant a)$. Operators $\rightarrow$ and $\exists$ are used according to their usual meaning.

3.1 Axiomatics

Axiomatization of logic $\mathcal{L}$ includes all tautologies of propositional calculus. Table 1 details the axioms and inference rules describing the behavior of belief, signed statement and reliability. These axioms and inference rules are standard and follow $KD45$

Table 1. Logic $\mathcal{L}$ axioms and inference rules

(K_S)	$\mathsf{Sign}(a, \varphi \rightarrow \psi) \rightarrow (\mathsf{Sign}(a, \varphi) \rightarrow \mathsf{Sign}(a, \psi))$
(D_S)	$\mathsf{Sign}(a, \varphi) \rightarrow \neg\mathsf{Sign}(a, \neg\varphi)$
(K_B)	$\mathsf{Bel}(a, \varphi \rightarrow \psi) \rightarrow (\mathsf{Bel}(a, \varphi) \rightarrow \mathsf{Bel}(a, \psi))$
(4_B)	$\mathsf{Bel}(a, \varphi) \rightarrow \mathsf{Bel}(a, \mathsf{Bel}(a, \varphi))$
(5_B)	$\neg\mathsf{Bel}(a, \varphi) \rightarrow \mathsf{Bel}(a, \neg\mathsf{Bel}(a, \varphi))$
$(R_{\leqslant})$	$t \leqslant t$
$(Tr_{\leqslant})$	$t \leqslant t' \wedge t' \leqslant t'' \rightarrow t \leqslant t''$
$(T_{\leqslant})$	$t \leqslant t' \vee t' \leqslant t$
$(To_{\leqslant})$	$\mathsf{Bel}(a, t \leqslant t') \vee \mathsf{Bel}(a, t' \leqslant t)$
(MP)	From φ and $\varphi \rightarrow \psi$ infer ψ
(G)	From φ infer $\forall t\varphi$
(N_S)	From φ infer $\mathsf{Sign}(t, \varphi)$
(N_B)	From φ infer $\mathsf{Bel}(a, \varphi)$

logic for the behavior of signed statement and $K45$ logic for the behavior of belief; let us again stress that axiom D could not hold because of public announcement (see section 5). Notice axiom schema $(To_{\leqslant})$ which reflects that reliability relations have to be believed as total. Let $\vdash$ denotes the proof relation.

3.2 Semantics

The semantics of $\mathcal{L}$-formulas is defined in terms of possible states and relations between states [12]. Those relations respectively represent the notion of signatures and beliefs as discussed in section 2. In each state, propositional symbols are interpreted and total preorders representing agents' reliability are set.

Definition 2 (Model). *Let M be a model defined as a tuple:*

$$\langle W, \bigcup_{i \in \mathcal{A}} S_i, \bigcup_{i \in \mathcal{A}} B_i, I, \preceq \rangle$$

where W is a set of possible states. $S_i \in W \times W$ is an accessibility relation representing signatures, $B_i \in W \times W$ is an accessibility relation representing beliefs. I is an interpretation function of the propositional symbols w.r.t. each possible state, $I : W \times \mathcal{P} \mapsto \{0, 1\}$. $\preceq$ is a function which represents total preorders; these preorders are specific to each state, that is $\preceq$: $W \mapsto 2^{A \times A}$.

A variable assignment is a function v which maps every variable x to an agent id. A t-alternative v' of v is a variable assignment similar to v for every variable except t. For $t \in \mathcal{T}$, $[\![t]\!]_v$ belongs to A and refers to the assignment of agent terms w.r.t. variable assignment v, such that:

$$\text{if } t \in A \text{ then } [\![t]\!]_v = t \qquad \text{if } t \in \mathcal{V} \text{ then } [\![t]\!]_v = v(t)$$

We define the satisfaction relation $\models$ with respect to some model M, state w and variable assignment v as follows.

Definition 3 ($\models$). *Let M be a model and v be a variable assignment: $v : \mathcal{V} \to A$. M satisfies an $\mathcal{L}$-formula φ w.r.t. a variable assignment v and a state w, according to the following rules:*

- $M, v, w \models t \leqslant t'$ *iff* $([\![t]\!]_v, [\![t']\!]_v) \in \preceq(w)$.
- $M, v, w \models p$ *iff* $p \in \mathcal{P}$ *and* $I(w, p) = 1$.
- $M, v, w \models \mathsf{Sign}(t, \varphi)$ *iff* $M, v, w' \models \varphi$ *for all* w' *s.t.* $(w, w') \in S_{[\![t]\!]_v}$
- $M, v, w \models \mathsf{Bel}(a, \varphi)$ *iff* $M, v, w' \models \varphi$ *for all* w' *s.t.* $(w, w') \in B_a$
- $M, v, w \models \forall t \varphi$ *iff for every t-alternative v',* $M, v', w \models \varphi$.

We write $\models \varphi$ iff for all M, w and v, we have $M, v, w \models \varphi$. The semantics for operators $\neg$, $\rightarrow$, $\vee$, $\wedge$ and $\exists$ is defined in the standard way. Let us now detail the constraints that should operate on the model. We require that signature has to be consistent which entails that all relations S_i have to be serial. Belief operator as well as signature operator are *K45* operator and thus all B_i and S_i are transitive and euclidean. Interwoven relations between signatures and beliefs are detailed in the next section.

Constraining the Reliability Relations. We assume that every agent holds belief about reliability without any uncertainty. That is, agent's beliefs about reliability can be represented as a total preorder. We propose to handle multiple preorders by indexing reliability with worlds. However, the aggregation of the preorders associated to all the believable worlds of one agent (which are total) must lead to a total preorder. This will then help the agent to aggregate all signed statements. In other words, we require that the integration (or merging) of signed statements should be based on an underlying total preorder over statements (as it is commonly assumed in the belief revision and merging areas—see [14,17,18]). Formally we introduce the two following constraints:

1. for all states w, $t \preceq(w) t'$ or $t' \preceq(w) t$ and,
2. suppose w' s.t. $(w, w') \in B_i$ and $t \preceq(w') t'$, then for all states w'' s.t. $(w, w'') \in B_i$, $t \preceq(w'') t'$.

The first constraint enforces that preorders are total in all states, which reflects axiom schema $(T_{\leqslant})$. The second constraint expresses that totality should hold in all the belief states of one agent, which reflects axiom schema $(To_{\leqslant})$. Moreover, preorder definition entails that reflexivity and transitivity hold.

We conclude the section by giving the results about soundness and completeness.

Theorem 1. *Logical system $\mathcal{L}$ is sound and complete.*

Proof. Soundness is straightforward. The completeness proof is mainly based on [12]. The proof is based on the definition of a canonical model which is itself built upon the definition of maximal consistent sets. A maximal consistent set is defined as follows: let $\varphi^0, \varphi^1, \cdots$ be an infinite sequence of $\mathcal{L}$-formulas. W.r.t. the sequence $\varphi^0, \varphi^1, \cdots$, a maximal and consistent set T is built s.t. $T = \cup_{i \in 0 \cdots \infty} T^i$ where: (i) if $\varphi^i \neq \forall t \varphi(t)$ then $T_i = T_{i-1} \cup \{\varphi_i\}$ if $T_{i-1} \cup \{\varphi_i\}$ is consistent and $T_i = T_{i-1} \cup \{\neg\varphi_i\}$ otherwise and (ii) if $\varphi^i = \forall t \varphi(t)$ then $T_i = T_{i-1} \cup \{\varphi_i\}$ if $T_{i-1} \cup \{\varphi_i\}$ is consistent and $T_i = T_{i-1} \cup \{\neg\varphi_i\} \cup \{\neg\varphi(\bar{t})\}$ otherwise ($\bar{t}$ is a new variable). Using the set of maximal consistent sets, we then define the canonical model M^c: $M^c = \langle W^c, \cup_{i \in \mathcal{A}} S_i, \cup_{i \in \mathcal{A}} B_i, I, \preceq \rangle$ s.t. (i) W^c is the set of maximal consistent sets, (ii) $I(w, p) = 1$ if $p \in w$ and $I(w, p) = 0$ otherwise, (iii) $a \preceq(w) b$ iff $a \leqslant b \in w$, (iv) $(w, w') \in B_a$ iff $\{\varphi \mid \mathsf{Bel}(a, \varphi) \in w\} \subseteq w'$ (idem for S_a). Using that canonical model, it is routine to prove that $M^c, w \models \varphi$ iff $\varphi \in w$ by assuming that property $M^c, w \models \varphi'$ iff $\varphi' \in w$ is satisfied for every subformula φ' of φ. Next, we conclude the proof by showing that M^c is a model for logic $\mathcal{L}$, that is all S_a are serial, all B_a are transitive and euclidean and all $\preceq$ are total preorders (because of the axioms $(R_{\leqslant})$, $(Tr_{\leqslant})$ and $(To_{\leqslant})$). Completeness is then proved.

4 Linking Signatures and Beliefs

There are multiple ways to switch from information to beliefs. These different ways may follow principles issued from the belief merging principle [20,18,6] or epistemic attitudes such as trust [21,19]. As previously mentioned, we do not require that an agent has to believe that other agents believe in information they provide. This is a key issue when information is propagated from one agent to another. At some stage, an agent

may just broadcast some information without committing to that information in terms of belief. Hereafter, we present two different policies showing how an agent switches from signed information to belief: the first one consists of aggregating in an incremental way signed statements that are commonly signed by a subset of agents which are equally reliable; the second policy consists of accepting as beliefs statements which are signed by one agent and not contradicted by the other agents.

4.1 Ranking Agents

The two policies for aggregating signed statements require that these signed statements are considered in an incremental way; that is "from the most reliable to the less reliable statements". Agents can be ranked since we always consider a total preorder. The agents which are equally reliable are gathered in the same group and the groups can then be ranked. Agents are ranked as follows. At first, we characterize the most reliable set of agents denoted as set C_1 with the help of the following formula:

$$a \in \mathsf{C}_1 =_{def} \forall t(a \leqslant t)$$

The formula characterizing members of C_1 can then be used for characterizing membership to a set C_i such that $i > 1$.

$$a \in \mathsf{C}_i =_{def} \Big(\big(\bigwedge_{j \in 1 \ldots (i-1)} \neg(a \in \mathsf{C}_j) \big) \wedge \forall t \big(\big(\bigwedge_{j \in 1 \ldots (i-1)} \neg(t \in \mathsf{C}_j) \big) \rightarrow (a \leqslant t) \big) \Big)$$

It means that all agents belonging to a set C_i are equally reliable and for all $a \in \mathsf{C}_i, b \in \mathsf{C}_j$ if $i <_{\mathbb{N}} j$ then $a < b$.

Let us now rephrase this ranking in terms of semantics. Consider agent a, an initial state w_0 and the reliability relations associated to the belief states of agent a that can be reached using relation B_a. These reliability relations can be aggregated to produce a total preorder representing the reliability relation believed by agent a at w_0. Notice that the constraints shown in section 3.2 ensure that this preorder is total and thus we can build a partition of the set of agent ids as follows.

Let $\mathcal{C}(w_0)$ be a partition of $\mathcal{A}$ at w_0 such that in every set $\mathcal{C}_i(w_0)$ of $\mathcal{C}(w_0)$, all agents are equally reliable:

$$a, b \in \mathcal{C}_i(w_0) \iff a \preceq(w_0) b \text{ and } b \preceq(w_0) a$$

For all $a \in \mathcal{C}_i(w_0), b \in \mathcal{C}_j(w_0)$ if $i <_{\mathbb{N}} j$ then $a \prec(w_0) b$.

Proposition 1. *For any M, w_0 and v: $a \in \mathcal{C}_i(w_0)$ iff $M, w_0, v \models a \in \mathsf{C}_i$.*

4.2 Careful Aggregation

The first policy for aggregating signed statement is a skeptical (or careful) policy, close to the classical notion of universal belief (i.e. everybody in a group believes a certain

proposition p to be true) [11]: only statements which are universally signed by a group of agents are considered as beliefs.

Let us consider the formal definition. The following shortcut stands for the fact that all agents belonging to some set C_i sign information φ:

$$\mathsf{Sign}(\mathsf{C}_i, \varphi) =_{def} \forall t\big(t \in \mathsf{C}_i \rightarrow \mathsf{Sign}(t, \varphi)\big)$$

The following definition of aggregation policy states that statement φ is believed by agent a if agent a (i) believes that statement φ is a common signed statement w.r.t. some C_i (line 1 left part) and (ii) believes that $\neg\varphi$ is not signed by all other agents which are more reliable than agents belonging to C_i (line 1 right part).

$$\mathsf{Careful}(a,\varphi) =_{def} \bigvee_i \Big(\mathsf{Bel}\big(a, \mathsf{Sign}(\mathsf{C}_i, \varphi)\big) \wedge \mathsf{Bel}\big(a, \bigwedge\nolimits_{j\in 1\ldots(i-1)} \neg\mathsf{Sign}(\mathsf{C}_j, \neg\varphi)\big)\Big) \rightarrow \mathsf{Bel}(a, \varphi)$$

Note that $\mathsf{Careful}(a,\varphi)$ stands for: agent a aggregates information about φ in a careful way.

4.3 A More Confident Aggregation

The first proposed policy for switching from signed statements to belief is a restrictive one. A less careful policy is to elaborate beliefs by considering statements that are signed by some agent(s) belonging to a group $\mathcal{C}_i$ while other agents of the same level of preference don't have contradictory opinion on this statement. As previously, these statements have to be aggregated with respect to the reliability of the groups of agents.

This aggregation attitude can be expressed in our logic. We first redefine the notion of group signature as follows:

$$\mathsf{C\text{–}Sign}(\mathsf{C}_i, \varphi) =_{def} \exists t\big(t \in \mathsf{C}_i \wedge \mathsf{Sign}(t, \varphi)\big) \wedge \forall t'(t' \in \mathsf{C}_i \rightarrow \neg\mathsf{Sign}(t', \neg\varphi))$$

This shortcut stands for: there is a formula φ that is believed by some agent belonging to the group C_i and all the agents of that group can sign this statement φ. Again, the aggregation stage has to be redefined in order to consider this confident group signature:

$$\mathsf{Conf}(a,\varphi) =_{def} \bigvee_i \Big(\mathsf{Bel}\big(a, \mathsf{C\text{–}Sign}(\mathsf{C}_i, \varphi)\big) \wedge \mathsf{Bel}\big(a, \bigwedge\nolimits_{j\in 1\ldots(i-1)} \neg\mathsf{C\text{–}Sign}(\mathsf{C}_j, \neg\varphi)\big)\Big) \rightarrow \mathsf{Bel}(a, \varphi)$$

Note that $\mathsf{Conf}(a,\varphi)$ stands for: agent a aggregates information about φ in a confident way.

As we can see, facing information sent by other agents does not entail that the receiver has only one way to handle them. By specifying multiple policies for elaborating its belief state, we allow a more flexible definition of multi-agent system: each agent of the system can adopt a specific policy. This characteristic also enables to handle the trust issue in a sophisticated way: when an agent believes that some other agents sign information it can at first ask them (or check) if they believe information they sign and

second ask them (or check) how these agents elaborate their belief state. It then offers new opportunities for the definition of more sophisticated policies for elaborating beliefs which take into account the two previous issues.

5 Acquiring Information

In the previous section, we have detailed two policies for building belief based on signed information. These policies consider belief and signed information from a static point of view. Let us now consider a more dynamic view by introducing actions of the form "agent b tells to agent a that a certain fact φ is true" (*alias* tell actions). This *tell* action ensures that agent a will believe that agent b signs φ, that is, a *tell* action is responsible for updating an agent's beliefs about other agents' signatures. A *tell* action of agent b (the sender) towards agent a (the receiver) that φ is true is considered as a *private announcement* in the sense of [15], so that if agent a's belief state should change, the belief states of the other agents should not change. We note tell actions by $\texttt{Tell}(b, a, \varphi)$. Let $\mathcal{L}_T$ be the extended language which embeds *tell* statements.

Definition 4 (Syntax of $\mathcal{L}_T$). *The set of formulas of the language $\mathcal{L}_T$ is defined by the following BNF:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{Sign}(t, \varphi) \mid \mathsf{Bel}(a, \varphi) \mid$$
$$\forall x\varphi \mid t \leqslant t' \mid [\texttt{Tell}(b, a, \varphi)]\varphi$$

where $p \in \mathcal{P}$, $t \in \mathcal{T}$, $t' \in \mathcal{T}$, $a \in \mathcal{A}$, $b \in \mathcal{A}$ and $x \in \mathcal{V}$.

$\mathcal{L}_T$ extends $\mathcal{L}$ with dynamic operator $[\texttt{Tell}(b, a, \varphi)]$. The intuitive meaning of statement $[\texttt{Tell}(b, a, \varphi)]\varphi'$ is after b tells φ to a, φ' holds.

Let us first focus on the axiomatics of the logic $\mathcal{L}_T$. Table 2 details the reduction axioms describing the behavior of the operator $[\texttt{Tell}(a, b, \varphi)]$. (T_{AP}) denotes the atomic permanence, (T_N) denotes negation handling, and (T_C) denotes conjunction handling. (T_B) describes the interplay between a *tell* action and the beliefs of the message receiver. $(T_{B_{\neq}})$ describes the interplay between a *tell* action and the beliefs of all agents different from the message receiver. In particular, $(T_{B_{\neq}})$ highlights the permanence of the beliefs of all agents different from the message receiver. (T_S) describes signature permanence, $(T_{\leq})$ describes preferences permanence, and $(T_{\forall})$ describes the interplay between *tell* action and quantification over variable assignments.

We write $\vdash_T$ to denote the proof relation for the logic $\mathcal{L}_T$ determined by the principles of the logic $\mathcal{L}$, the schemata in table 2 and the rule of replacement of proved equivalence.

The result bellow captures the essential aspect of the *tell* action. It says that after agent b tells to agent a information φ, agent a believes that agent b signs φ:

Proposition 2. $\vdash_T [\texttt{Tell}(b, a, \varphi)]\mathsf{Bel}(a, \mathsf{Sign}(b, \varphi))$

Once agent a starts to believe that agent b signs φ (as an effect of b's act of telling to a that φ), agent a might also start to believe that φ. As we have shown above, this depends on the reliability of agent b according to agent a and on policies linking signatures with beliefs such as those described in Section 4.

Table 2. Logic $\mathcal{L}_T$ axioms

$$
\begin{array}{ll}
(T_{AP}) & [\texttt{Tell}(b,a,\varphi)]p \leftrightarrow p \\
(T_N) & [\texttt{Tell}(b,a,\varphi)]\neg\varphi \leftrightarrow \neg[\texttt{Tell}(b,a,\varphi)]\varphi \\
(T_C) & [\texttt{Tell}(b,a,\varphi)](\varphi \wedge \varphi') \leftrightarrow \\
 & \quad ([\texttt{Tell}(b,a,\varphi)]\varphi \wedge [\texttt{Tell}(b,a,\varphi)]\varphi') \\
(T_B) & [\texttt{Tell}(b,a,\varphi)]\mathsf{Bel}(a,\varphi) \leftrightarrow \\
 & \quad \mathsf{Bel}(a,(\mathsf{Sign}(b,\varphi) \rightarrow [\texttt{Tell}(b,a,\varphi)]\varphi)) \\
(T_{B_{\neq}}) & [\texttt{Tell}(b,a,\varphi)]\mathsf{Bel}(i,\varphi) \leftrightarrow \mathsf{Bel}(i,\varphi) \quad \text{if } i \neq a \\
(T_S) & [\texttt{Tell}(b,a,\varphi)]\mathsf{Sign}(t,\varphi') \leftrightarrow \mathsf{Sign}(t,\varphi') \\
(T_{\leqslant}) & [\texttt{Tell}(b,a,\varphi)](t \leqslant t') \leftrightarrow (t \leqslant t') \\
(T_{\forall}) & [\texttt{Tell}(b,a,\varphi)]\forall x\varphi \leftrightarrow \forall x[\texttt{Tell}(b,a,\varphi)]\varphi
\end{array}
$$

We define the semantics of a *tell* statement as follows. The truth conditions are those given above for the formulas p, $\neg\varphi$, $\varphi\wedge\varphi'$, $\mathsf{Sign}(t,\varphi)$, $\mathsf{Bel}(a,\varphi)$, $\forall x\varphi$, $t \leqslant t'$. The truth condition for $[\texttt{Tell}(b,a,\varphi)]\varphi'$ is defined in a way which is close to the semantics of dynamic epistemic logic [29]. More precisely, after agent b tells to agent a information φ, agent a removes from its belief state all states in which agent b does not sign φ so that agent a believes that agent b signs φ.

Definition 5 (Announcement Semantics). *Let M be a model s.t. $M = \langle W, \bigcup_{i\in\mathcal{A}} S_i, \bigcup_{i\in\mathcal{A}} B_i, I, \preceq\rangle$ and let w be a state in W. We have:*

- $M,v,w \models [\texttt{Tell}(b,a,\varphi)]\varphi'$ *iff* $M|_{\langle a,b,\varphi\rangle}, v, w_1 \models \varphi'$.

where $M|_{\langle b,a,\varphi\rangle} = \langle W^*, \bigcup_{i\in\mathcal{A}} S_i^*, \bigcup_{i\in\mathcal{A}} B_i^*, I^*, \preceq^*\rangle$ is defined as follows:

- $W^* = W_1 \cup W_2$ *such that* $W_1 = \{w_1 | w \in W\}$ *and* $W_2 = \{w_2 | w \in W\}$;
- $B_a^* = \{(w_1, w_1') | (w,w') \in B_a$ *and* $M,v,w' \models \mathsf{Sign}(b,\varphi)\} \cup \{(w_2,w_2') | (w,w') \in B_a\}$;
- $B_i^* = \{(w_1,w_2') | (w,w') \in B_i\} \cup \{(w_2,w_2') | (w,w') \in B_i\}$ *for all* $i \in \mathcal{A}$ *such that* $i \neq a$;
- $S_i^* = \{(w_1,w_2') | (w,w') \in S_i\} \cup \{(w_2,w_2') | (w,w') \in S_i\}$ *for all* $i \in \mathcal{A}$;
- $\preceq^*(w_1) = \preceq^*(w_2) = \preceq(w)$ *for all* $w \in W$;
- $I^*(w_1,p) = I^*(w_2,p) = I(w,p)$ *for all* $w \in W$.

Basically, the effect of b's action of telling to a that φ is to shrink the set of belief accessible states for a to the states in which b signs φ, while keeping constant the set of accessible belief states for all other agents (see Figure 3). To get this result, we proceed as follows: (i) all states w are duplicated in W_1 and W_2; (ii) we restrict the set of possible belief state of agent a w.r.t. set of states W_1. It can be the case that some agent c has some belief about agent a's belief; in such configuration, changing the set of possible belief state for agent a in set of states W_1 should have no influence on belief of agent c about agent a's belief. We prevent this problem using the set of duplicated states W_2

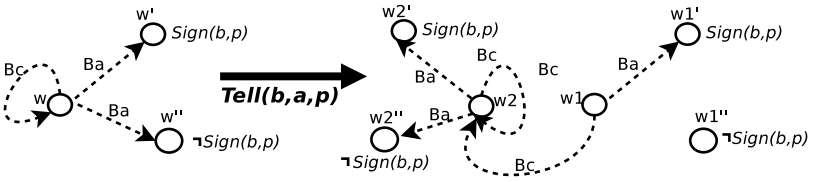


Fig. 3. Updating belief state

preserving all initial relations B and S so that each agent except agent a has a link between W_1 and W_2. Figure 3 illustrates this problem: suppose that b tells to a statement p; we should then restrict belief of agent a: state w'' should not be a possible belief state. If we do not duplicate states and relations, the consequence is that agent c's belief will be changed. The duplication keeps state w'', represented by w''_2, as a possible belief state for agent a when this belief state is considered from agent c's possible belief state.

Note that $Tell(b, a, \varphi)$ also keeps constant other agents' signatures and the reliability order over agents: S_i^* and $\preceq^*$ are full copies of initial S_i and $\preceq$.

Theorem 2. *If M is a $\mathcal{L}$-model then $M|_{\langle a,b,\varphi\rangle}$ is also a $\mathcal{L}$-model.*

Proof. It is straightforward to prove that, after action $\texttt{Tell}(b, a, \varphi)$, preorders $\preceq^*(w)$ are total for every state w (i.e. for all states w, $t \preceq^*(w) t'$ or $t' \preceq^*(w) t$), and totality holds in all possible states for an agent (i.e. if wB_i^*w' and $t \preceq^*(w') t'$, then for all states w'' s.t. wB_i^*w'', $t \preceq^*(w'') t'$). Let us prove that the action $\texttt{Tell}(b, a, \varphi)$ preserves the seriality of every S_i and the transitivity of every B_i. Suppose there is w' such that $(w, w') \in S_i$. Then, $(w_1, w'_2) \in S_i^*$ and $(w_2, w'_2) \in S_i^*$. Thus, S_i^* is serial.

Suppose $(w, w'), (w', w'')$ and $(w, w'') \in B_i$ s.t. $i \neq a$. Then $(w_1, w'_2) \in B_i^*$ and $(w'_2, w''_2) \in B_i^*$ and $(w_1, w''_2) \in B_i^*$. Then, transitivity is preserved for all agent $i \neq a$. Now, let us focus on agent a. Suppose $(w, w'), (w', w'')$ and $(w, w'') \in B_a$, we have to consider different cases w.r.t. the truth value of $\mathsf{Sign}(b, \varphi)$ in w' and w''. If $M, v, w' \models \mathsf{Sign}(b, \varphi)$ and $M, v, w'' \models \mathsf{Sign}(b, \varphi)$, then transitivity is preserved; if $M, v, w' \models \mathsf{Sign}(b, \varphi)$ and $M, v, w'' \not\models \mathsf{Sign}(b, \varphi)$ then (w_1, w'_1) and $(w'_1, w''_1) \notin B_a^*$ and transitivity is preserved; if $M, v, w' \not\models \mathsf{Sign}(b, \varphi)$ and $M, v, w'' \models \mathsf{Sign}(b, \varphi)$ then $(w_1, w'_1) \notin B_a^*$ and transitivity is preserved; finally if $M, v, w' \not\models \mathsf{Sign}(b, \varphi)$ and $M, v, w'' \not\models \mathsf{Sign}(b, \varphi)$ then $(w_1, w'_1), (w_1, w''_1)$ and $(w'_1, w''_1) \notin B_a^*$ and transitivity is also preserved.

In a similar way we can prove that action $\texttt{Tell}(b, a, \varphi)$ preserves the euclidianity of every B_i.

We conclude this section by considering soundness and completeness.

Theorem 3. *All formulas presented in table 2 are valid.*

We skip the proof since it is routine to prove these validities. We then state the theorem about completeness of the logic $\mathcal{L}_T$.

Theorem 4. *The logic $\mathcal{L}_T$ is completely axiomatized by principles of the logic $\mathcal{L}$ together with the schemata in Table 2 and the rule of replacement of proved equivalence.*

Proof. The theorem is an immediate consequence of Theorem 3. That is, each axiom allows to eliminate the tell operator in an incremental way; it enable to write formulas in a particular normal form: starting from the innermost modal operator, the tell operator is pushed inside the formulas such that it only occurs in front of atomic formulas, the tell operator can then be eliminated [29].

6 Example

Suppose a car accident involving three cars which are blue (bc), red (rc) and yellow (yc). Let po be the police detective who is interviewing the three eyewitnesses of the accident e_1, e_2, e_3. In this scenario we have therefore $\mathcal{A} = \{po, e_1, e_2, e_3\}$. The first eyewitness e_1 tells the police detective that the blue car is responsible of the accident while the second one (e_2) states that the red car has caused the collision; the third eyewitness (e_3) can only state information about the yellow car. The three eyewitnesses tell the police detective that yc is not responsible of the accident.

In that context of information gathering, the police detective does not need to assume that the eyewitnesses tell the truth or believe in information they provide. The police detective just needs to assume that e_1 provides or signs information $bc \wedge \neg rc \wedge \neg yc$, e_2 provides or signs information $\neg bc \wedge rc \wedge \neg yc$ and e_3 provides information $rc \wedge \neg yc$. Next, based on these pieces of information, the police detective will build his opinion, i.e. his belief about the accident. The police detective faces contradicting information about the blue and red cars, but because the witnesses all agree about the yellow one, the police detective should believe that the yellow car is not the responsible of the collision. That is, the detective is willing to root his belief w.r.t. the set of signed statements he handles and the aggregation policy he considers.

Let us formally translate this belief elaboration stage in formal terms. Suppose at first, the following preferences for the police detective: eyewitnesses 1 and 3 are equally reliable and are more reliable than eyewitness 2. That is:

$$\vdash_T \mathsf{Bel}(po, e_1 \leqslant e_3 \wedge e_3 \leqslant e_1) \wedge \mathsf{Bel}(po, e_1 < e_2 \wedge e_3 < e_2)$$

We also suppose that the police detective considers himself less reliable than e_2, because he was not present when the car accident occurred:

$$\vdash_T \mathsf{Bel}(po, e_2 < po)$$

Furthermore, we consider information provided by the three witnesses. Tell actions are represented by the following actions:

$$T_1 =_{def} \mathtt{Tell}(e_1, po, bc \wedge \neg rc \wedge \neg yc)$$
$$T_2 =_{def} \mathtt{Tell}(e_2, po, \neg bc \wedge rc \wedge \neg yc)$$
$$T_3 =_{def} \mathtt{Tell}(e_3, po, rc \wedge \neg yc)$$

Proposition 2 entails that the police detective believes received information:

$$\vdash_T [T_1][T_2][T_3]\mathsf{Bel}(po, \mathsf{Sign}(e_1, bc \wedge \neg rc \wedge \neg yc) \wedge \mathsf{Sign}(e_2, \neg bc \wedge rc \wedge \neg yc) \wedge \mathsf{Sign}(e_3, rc \wedge \neg yc))$$

Theorems 5–7 entail that after the announcements, preferences are unchanged:

$$\vdash_T [T_1][T_2][T_3]\mathsf{Bel}(po, e_1 \leqslant e_3 \wedge e_3 \leqslant e_1) \wedge \mathsf{Bel}(po, e_1 < e_2 \wedge e_3 < e_2)$$

Thirdly, we consider the aggregation stage. According to the detective's preferences, we have the following groups of equally reliable agents: $\mathsf{C}_1 = \{e_1, e_3\}$ and $\mathsf{C}_2 = \{e_2\}$. Let us focus on the definition of belief based on a careful approach. There is one statement, namely $\neg yc$, signed by the group of most reliable agents C_1. Moreover, $\neg bc \wedge rc \wedge \neg yc$ is signed by the group C_2. Finally, it is not the case that the group of most reliable agents signs $\neg rc$ because e_1 and e_3 provide conflicting information about rc (in particular e_3 says that rc):

$$\vdash_T [T_1][T_2][T_3]\mathsf{Bel}(po, \mathsf{Sign}(\mathsf{C}_1, \neg yc) \wedge \mathsf{Sign}(\mathsf{C}_2, \neg bc \wedge rc \wedge \neg yc) \wedge \neg\mathsf{Sign}(\mathsf{C}_1, \neg rc))$$

Hence, if the police detective aggregates information in a careful way, he will believe that the red car is the responsible one:

$$\vdash_T [T_1][T_2][T_3](\mathsf{Careful}(po,rc) \rightarrow \mathsf{Bel}(po, rc))$$

Let us now focus on belief involved by the confident aggregation policy. Since e_1 signs statement bc and e_3 tells nothing about bc, it is then the case that bc is signed in a confident way by the group C_1. That is, at least one agent in C_1 signs bc and no agent in C_1 signs $\neg bc$:

$$\vdash_T [T_1][T_2][T_3]\mathsf{Bel}(po, \mathsf{C\text{–}Sign}(\mathsf{C}_1, bc))$$

Hence, if the police detective aggregates information in a confident way, he will aggregate signed information in a different way and will conclude that the blue car is the responsible one:

$$\vdash_T [T_1][T_2][T_3](\mathsf{Conf}(po,rc) \rightarrow \mathsf{Bel}(po, bc))$$

As we can see, the key issue is the aggregation policy since it leads to different conclusions. It also leads to stress up the interest of defining belief as a non-primitive concept.

Finally, we can notice that both policies lead to the conclusion that yellow car is not responsible because of the unanimity of witnesses.

7 Conclusion

In this paper we have shown how information and its source can be processed by an agent so that at first, it just acquires information from sensors or other agents and second, it builds its belief state by considering signed information. By splitting information and belief, an agent is able to handle clear rationales to construct its belief state both from a static and dynamic perspectives. From a static perspective we have applied our formal framework to characterize two possible policies for agents in the process of building their belief state from the basic signed information they hold. From this perspective this work is close to what has been done in belief merging [20,18,6]. The key difference with existing work in the belief merging area is the introduction of merging

in a modal based framework at first (this is also a common characteristic with [6]); second a clear distinction between belief and signed statement and third a dynamic view on belief construction. These last two characteristics differ in two ways from existing work [20,18,6]: (i) it is usually assumed that belief and information are almost similar; we have shown that we do not have to assume this hypothesis; (ii) beliefs are almost not viewed as a primitive concepts but rather as the result of some information processing which gives a flexible framework (see Section 4). Our work is also related to the work of [22] in which agents' mental attitudes and agent's ostensible (expressed) attitudes are distinguished and a formalism capturing this distinction is proposed. In particular, our notion of signed information is close to the notion of ostensible belief of Nickles et al. However, Nickles et al. do not consider reliability of information sources. Moreover, their approach does not deal with dynamics of information by means of communicative actions. The latter is a central aspect of our proposal (see Section 5).

Concerning the dynamic perspective we have shown how the basic signed information held by an agent may change as it receives tell statements from another agent processed in a similar way to private announcements in the sense of dynamic epistemic logic (DEL) [29,15].

Our short term goal is to consider more sophisticated ways to set the reliability relations. The idea is to consider the agent skills [5] so that agents can consider multiple reliability relations at the same time. At this time, even if agents can consider multiple alternative reliability relations, they cannot mixed them. Our goal is to avoid this limit.

Another interesting direction of future research is to refine the dynamic part of our approach. Indeed, the approach presented in Section 5 only deals with belief expansion in the sense of AGM theory [1] but it does not deal with belief revision. In particular, our logic does not allow to model a situation in which an agent believes that a certain fact p is true and, after revising his beliefs in the light of a new information received by an information source, he starts to believe $\neg p$. In order to model this scenario, we will have to extend our formalism with a belief revision mechanism [28,2,26].

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: partial meet contraction and revision functions. The Journal of Symbolic Logic 50(2), 510–530 (1985)
2. Aucher, G.: A combined system for update logic and belief revision. In: Barley, M.W., Kasabov, N. (eds.) PRIMA 2004. LNCS (LNAI), vol. 3371, pp. 1–17. Springer, Heidelberg (2005)
3. Benferhat, S., Garcia, L.: Handling locally stratified inconsistent knowledge bases. Studia Logica 70, 77–104 (2002)
4. Booth, R., Meyer, T.: How to revise a total preorder. Journal of Philosophical Logic 40(2), 193–238 (2011)
5. Cholvy, L.: Automated reasoning with merged contradictory information whose reliability depends on topics. In: Froidevaux, C., Kohlas, J. (eds.) ECSQARU 1995. LNCS, vol. 946, pp. 125–132. Springer, Heidelberg (1995)
6. Cholvy, L.: A modal logic for reasoning with contradictory beliefs which takes into account the number and the reliability of the sources. In: Godo, L. (ed.) ECSQARU 2005. LNCS (LNAI), vol. 3571, pp. 390–401. Springer, Heidelberg (2005)

7. Cohen, P., Levesque, H.: Rational Interaction as the Basis for Communication. In: Cohen, P., Morgan, J., Pollack, M. (eds.) Intentions in Communication, pp. 221–256. MIT Press, Cambridge (1990)
8. Cohen, P., Levesque, H.: Communicative actions for artificial agents. In: Lesser, V., Gasser, L. (eds.) Proceedings of the First International Conference on Multi-Agent Systems (ICMAS 1995), pp. 65–72. The MIT Press, San Francisco (1995)
9. Fariñas del Cerro, L., Herzig, A., Longin, D., Rifi, O.: Belief reconstruction in cooperative dialogues. In: Giunchiglia, F. (ed.) AIMSA 1998. LNCS (LNAI), vol. 1480, pp. 254–266. Springer, Heidelberg (1998)
10. Dragoni, A., Giorgini, P.: Revising beliefs received from multiple sources. In: Frontiers of Belief Revision, Applied Logic. Kluwer, Dordrecht (1999)
11. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
12. Fagin, R., Halpern, Y.M.J.: Reasoning About Knowledge. MIT Press, Cambridge (1995)
13. Finin, T., Labrou, Y., Mayfield, J.: KQML as an agent communication language. In: Bradshaw, J. (ed.) Software Agents. MIT Press, Cambridge (1997)
14. Gärdenfors, P.: Knowledge in flux: Modeling the Dynamics of Epistemic States. MIT Press, Cambridge (1988)
15. Gerbrandy, J., Groeneveld, W.: Reasoning about information change. J. of Logic, Language and Information 6(2) (1997)
16. Herzig, A., Longin, D.: Belief dynamics in cooperative dialogues. J. of Semantics 17(2) (2000) (vol. published in 2001)
17. Katsuno, H., Mendelzon, A.: Propositional knowledge base revision and minimal change. Artificial Intelligence 52(3), 263–294 (1991)
18. Konieczny, S., Pérez, R.: Propositional belief base merging or how to merge beliefs/goals coming from several sources and some links with social choice theory. European Journal of Operational Research 160(3), 785–802 (2005)
19. Liau, C.: Belief, information acquisition, and trust in multi-agent systems–a modal logic formulation. Artificial Intelligence 149(1), 31–60 (2003)
20. Liberatore, P., Schaerf, M.: Arbitration (or how to merge knowledge bases). IEEE Transactions on Knowledge and Data Engineering 10(1), 76–90 (1998)
21. Lorini, E., Demolombe, R.: From Binary Trust to Graded Trust in Information Sources: A Logical Perspective. In: Falcone, R., Barber, S.K., Sabater-Mir, J., Singh, M.P. (eds.) Trust 2008. LNCS (LNAI), vol. 5396, pp. 205–225. Springer, Heidelberg (2008)
22. Nickles, M., Fischer, F., Weiss, G.: Communication attitudes: A formal approach to ostensible intentions, and individual and group opinions. In: Proc. of LCMAS 2005. Electronic Notes in Computer Science, vol. 157(4), pp. 95–115. Elsevier, Amsterdam (2005)
23. Perrussel, L., Thévenin, J.-M.: (Dis)Belief Change Based on Messages Processing. In: Dix, J., Leite, J. (eds.) CLIMA IV 2004. LNCS (LNAI), vol. 3259, pp. 201–217. Springer, Heidelberg (2004)
24. Perrussel, L., Thévenin, J.: A logical approach for describing (dis)belief change and message processing. In: Proc. of AAMAS 2004, pp. 614–621. IEEE C.S, Los Alamitos (2004)
25. Rao, A., Georgeff, M.: Modeling rational agents within a bdi-architecture. In: Proc. of KR 1991, pp. 473–484 (1991)
26. Segerberg, K.: Belief revision from the point of view of doxastic logic. Logic Journal of IGPL 3(4), 535–553 (1995)
27. van der Hoek, W., Wooldridge, M.: Towards a logic of rational agency. Journal of the IGPL 11(2), 133–157 (2003)
28. van Ditmarsch, H.: Prolegomena to dynamic logic for belief revision. Synthese 147(2), 229–275 (2005)
29. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Synthese Library, vol. 337. Springer, Heidelberg (2007)

On the Definability of Simulability and Bisimilarity by Finite Epistemic Models

Hans van Ditmarsch, David Fernández-Duque, and Wiebe van der Hoek

University of Sevilla, Spain
{hvd,dfduque}@us.es
University of Liverpool, United Kingdom
Wiebe.Van-Der-Hoek@liverpool.ac.uk

Abstract. We explore when finite epistemic models are definable up to simulability or bisimulation, either over the basic multi-agent epistemic language L or over its extension L^C with common knowledge operators. Our negative results are that: simulability is not definable in general in L^C, and finite epistemic states (i.e., pointed models) are not definable up to bisimulation in L. Our positive results are that: finite epistemic states are definable up to bisimulation by *model validity* of L-formulas, and there is a class of epistemic models we call *well-multifounded* for which simulability is definable over L. From our method it also follows that finite epistemic models (i.e., not-pointed models) are definable up to bisimulation using model validity in L. Our results may prove useful for the logical specification of multi-agent systems, as it provides justification for the ubiquitous but often unjustified claims of the form 'suppose action a can only be performed in state s': we show when such preconditions exist. An application are characteristic formulae for interpreted systems. They have a special form wherein factual knowledge, positive knowledge, and ignorance can be separated.

1 Introduction

Modal logic is *the* framework for formalising knowledge representation and areas in artificial intelligence as diverse as distributed computing, reasoning about programs, verifying temporal properties of systems, game theoretic reasoning, reasoning about knowledge and belief, and specifying and verifying multi-agent systems. One of the reasons of the popularity of modal logic in such diverse fields is its semantics: the notion of *states*, or *worlds*, together with that of a relation between them, is the key concept in Kripke models, on which modal logics are interpreted. Such worlds may model the state of a distributed system, a processor, or a machine, or a situation in a game or a protocol, and the binary relations indicate for instance a possible transition (in time, or by a computation) between states, or they may represent some attitude of an agent: some state may be desired by an agent, some states may form the goal of an agent, or, as is the interpretation for epistemic logic, states may be conceived as indistinguishable by the agent.

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 74–87, 2011.

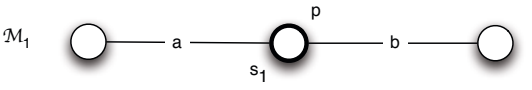


Fig. 1. A two-agent one-fact scenario

Kripke models are a great tool for designing and modelling complex situations, and a modal language provides a perfect way to verify properties about the systems obtained. To give an example, consider the simplest multi-agent system one can envisage to reason about information: we assume to have two agents, a and b, and one atomic fact, say p. Suppose the information about the scenario to be modelled is the following: *p is true, although neither a nor b knows it.* One model that comes to mind to represent this situation is $\mathcal{M}_1$ depicted in Figure 1, the 'actual state' is s_1. Given this state where p is true, each agent considers another state possible were it is false.

However, this raises many questions: is this *the* model of the scenario, or are there alternative models, and how do we tell they are different? For instance, would the models of Figure 2 be 'equally good'? It of course depends on what is meant by that criterion: yes, all three models satisfy the description of the scenario ($p \wedge \neg K_a p \wedge \neg K_b p$ is true in all of their designated states s_i), yet each model verifies some additional and *different* information about which the scenario description stayed ambiguous, or, rather, under-specified.

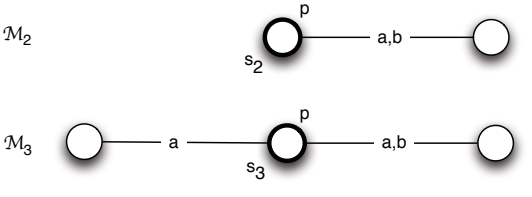


Fig. 2. Two other two-agent one-fact scenarios

The notion of *bisimulation* [3] tells us when two structures are essentially the same, at least as far as modal logic is concerned; in this sense it plays a role analogous to, say, an isomorphism between algebraic structures. Following this analogy a *simulation* [1] would play the role of a homomorphism; simulations are defined like bisimulations without the 'back' clause, and as in the case of isomorphisms, bisimulations can be defined as simulations whose inverses are also simulations. These relations preserve a fragment of the modal language known as the *positive existential formulas* [3].

Referring back to our examples, we want to specify that we are actually in model $\mathcal{M}_2$, in state s_2: Can we specify that in our object language? Because

the modal language is bisimulation-invariant we can hope at most to describe models up to bisimulation. However, it is not immediate that this can be done and, in fact, it is usually not the case. For example, finite unimodal K-states (i.e., arbitrary finite Kripke frames) cannot be defined up to bisimulation over the basic modal language, but they can over PDL [9,2].

Over the class of transitive models we do have that finite pointed models are definable up to bisimulation in the basic modal language, but greater expressive power is needed to define simulability [6]. Over $\mathsf{S5}$ we should expect the situation to become much simpler because accessibility is an equivalence relation [8] – indeed, both simulability by and bisimilarity to finite models are modally definable in this class – but once we consider multiagent models, things become trickier.

Our goal is, for $\mathsf{S5}$, to explore when finite epistemic models are definable up to simulability or bisimulation, either over the basic modal language L or the language L^C enriched with common knowledge operators. Our main results are

1. simulability is not definable in general in L^C (Theorem 2);
2. finite epistemic states are not definable up to bisimulation in L (Theorem 1);
3. finite epistemic models are defined up to bisimulation by *model validity* in L (Theorem 3), and
4. there is a class of epistemic models we call *well-multifounded* for which simulability is definable over L (Theorem 5).

Note that finite epistemic states are definable up to bisimulation in L^C [10]; our third result is a variant of this, exploiting the fact that the common knowledge operator is used in a somewhat shallow fashion there.

2 Epistemic Logic

We consider the basic language of epistemic logic $\mathsf{L} = \mathsf{L}^A$, where A is a non-empty finite set of 'agents' (in this paper mainly a and b) and whose formulas are built from propositional variables in a *finite* set PV using the Boolean connectives $\wedge$ and $\neg$ (all other connectives are to be defined in terms of these) and the unary modal operator K_a for each $a \in A$. We write M_a as a shorthand for $\neg K_a \neg$. The reason we work with a finite set PV of propositional variables is that, evidently, one cannot define models up to simulation or bisimulation in the presence of infinitely many variables using a finite formula.

The language L^C is an extension of L which introduces an operator C_B ('common knowledge') for each $B \subseteq A$.

We are interested in interpreting L and L^C over *epistemic models*, which are tuples

$$\mathcal{M} = \langle |\mathcal{M}|, \sim_{\mathcal{M}}, [\![\cdot]\!]_{\mathcal{M}} \rangle$$

where $|\mathcal{M}|$ is a non-empty set, $\sim_{\mathcal{M}}$ a tuple of equivalence relations $\overset{a}{\sim}_{\mathcal{M}}$ for each $a \in A$ and

$$[\![\cdot]\!]_{\mathcal{M}} : \mathsf{PV} \to 2^{|\mathcal{M}|}.$$

We will usually omit the subindex on $\sim_{\mathcal{M}}$ unless this may lead to confusion. The valuation $[\![\cdot]\!]_{\mathcal{M}}$ is extended to arbitrary formulas in the standard way for Booleans. For the epistemic modal operator we have

$$[\![K_a\varphi]\!]_{\mathcal{M}} = \{s \in |\mathcal{M}| : \forall t \overset{a}{\sim} s, t \in [\![\varphi]\!]_{\mathcal{M}}\},$$

and $[\![C_B\varphi]\!]_{\mathcal{M}}$ is the largest subset F of $|\mathcal{M}|$ such that, if $s \in F$, $b \in B$ and $t \overset{b}{\sim} s$, then $t \in [\![\varphi]\!]_{\mathcal{M}}$. For $t \in [\![\varphi]\!]_{\mathcal{M}}$ we also write $\langle \mathcal{M}, t\rangle \models \varphi$, and if $\langle \mathcal{M}, t\rangle \models \varphi$ for all $t \in |\mathcal{M}|$ we write $\mathcal{M} \models \varphi$, and we say φ is valid on model $\mathcal{M}$. An *epistemic state* (or *pointed epistemic model*) is a pair $\langle \mathcal{M}, s\rangle$ where $\mathcal{M}$ is an epistemic model and $s \in |\mathcal{M}|$.

3 Simulation and Bisimulation

In this section we define *simulations* and *bisimulations*, and some notions and results linking these to the logical language.

Definition 1. *If $\mathcal{M}$, $\mathcal{N}$ are epistemic models, a* simulation *between $\mathcal{M}$ and $\mathcal{N}$ is a binary relation*

$$S \subseteq |\mathcal{M}| \times |\mathcal{N}| \quad \textit{such that}$$

Atoms *for every x S y and every propositional variable p, $x \in [\![p]\!]_{\mathcal{M}} \Leftrightarrow y \in [\![p]\!]_{\mathcal{N}}$ and*

Forth *if $x' \overset{a}{\sim}_{\mathcal{M}} x$ and x S y, there is $y' \overset{a}{\sim}_{\mathcal{N}} y$ with x' S y'.*

The relation S is a bisimulation *if it further satisfies*

Back *if x S y and $y' \overset{a}{\sim}_{\mathcal{N}} y$, there is $x' \overset{a}{\sim}_{\mathcal{M}} x$ with x' S y'.*

Given models $\mathcal{M}$ and $\mathcal{N}$, a point $x \in |\mathcal{M}|$ *simulates* $y \in |\mathcal{N}|$ if there exists a simulation $S \subseteq |\mathcal{M}| \times |\mathcal{N}|$ such that x S y; we will write $\langle \mathcal{M}, x\rangle \trianglelefteq \langle \mathcal{N}, y\rangle$. We also say that $\langle \mathcal{M}, x\rangle$ simulates $\langle \mathcal{N}, y\rangle$, or that $\langle \mathcal{N}, y\rangle$ is simulated by $\langle \mathcal{M}, x\rangle$.

If a bisimulation B exists between $\mathcal{M}$ and $\mathcal{N}$ such that x B y, we will write $\langle \mathcal{M}, x\rangle \leftrightarrows \langle \mathcal{N}, y\rangle$ or (unless confusion results) $x \leftrightarrows y$. We write $\mathcal{M} \leftrightarrows \mathcal{N}$ (in words, $\mathcal{M}$ and $\mathcal{N}$ are *bisimilar* to each other) if there is a bisimulation between them with domain $|\mathcal{M}|$ and range $|\mathcal{N}|$. This is called a *total bisimulation.*

A well-known fact [3] is that:

Proposition 1. *If $\langle \mathcal{M}, x\rangle \leftrightarrows \langle \mathcal{N}, y\rangle$, then x and y satisfy the same formulas of* L.

Although we are mainly interested in "full" simulations and bisimulations, we occasionally need an approximation to a bisimulation given by an *k-bisimulation*, defined as follows:

Definition 2. *Let $\mathcal{M}, \mathcal{N}$ be epistemic models, $x \in |\mathcal{M}|$ and $y \in |\mathcal{N}|$ and $k \geq 0$. We define $x \leftrightarrows_k y$ (x is* k-bisimilar to *y) if x and y satisfy the same set of atoms and either $k = 0$ or the following variant of the 'back and forth' conditions holds:*

Forth' *if* $x' \overset{a}{\sim}_{\mathcal{M}} x$, *there is* $y' \overset{a}{\sim}_{\mathcal{N}} y$ *with* $x' \leftrightarrows_{k-1} y'$ *and*
Back' *if* $y' \overset{a}{\sim}_{\mathcal{N}} y$, *there is* $x' \overset{a}{\sim}_{\mathcal{M}} x$ *with* $x' \leftrightarrows_{k-1} y'$.

Then we have the following, also found in [3]:

Proposition 2. *If* $\langle \mathcal{M}, x\rangle \leftrightarrows_k \langle \mathcal{N}, y\rangle$, *then* x *and* y *satisfy the same formulas of* L *of modal depth at most* k.

Recall that the *modal depth* of a formula φ is the nesting number of modal operators appearing in φ.

We continue by defining some relevant types of formulas. Let an epistemic state $\langle \mathcal{M}, s\rangle$ be given, and a logical language LAN, where LAN is either L or L^C.

The *factual description* $\sigma_s \in \mathsf{LAN}$ of state s in $\mathcal{M}$ is the conjunction of the values of all variables in s: Let $\sigma_p = p$ if $s \in [\![p]\!]_{\mathcal{M}}$ and else $\sigma_p = \neg p$, then $\sigma_s := \bigwedge \sigma_p$.

A *characteristic formula* $\chi_{\langle \mathcal{M},s\rangle}$, or, alternatively, a *description of the epistemic state* $\langle \mathcal{M}, s\rangle$, is a LAN formula that is true in $\langle \mathcal{M}, s\rangle$ and such that $\langle \mathcal{N}, t\rangle \models \chi_{\langle \mathcal{M},s\rangle}$ implies $\langle \mathcal{N}, t\rangle \leftrightarrows \langle \mathcal{M}, s\rangle$. Similarly a *description of an epistemic model* $\mathcal{M}$ is a LAN formula $\chi_{\mathcal{M}}$ true in the model $\mathcal{M}$ and such that for all $\mathcal{N}$, $\mathcal{N} \models \chi_{\mathcal{M}}$ implies $\mathcal{M} \leftrightarrows \mathcal{N}$.

A *distinguishing formula* between two subsets S, S' of an epistemic model $\mathcal{M}$ is a LAN formula $\delta_{S,S'}$ such that $S \subseteq [\![\delta_{S,S'}]\!]_{\mathcal{M}}$ whereas $S' \cap [\![\delta_{S,S'}]\!]_{\mathcal{M}} = \emptyset$. If $S = \{x\}$ and $S' = \{y\}$ we write $\delta_{x,y}$ and we say that $\delta_{x,y}$ distinguishes x from y in $\mathcal{M}$; and for a formula that distinguishes a state x from all other (non-bisimilar) states in the model $\mathcal{M}$ we write δ_x. For a distinguishing δ_x we have that for all t in $\mathcal{M}$, $\langle \mathcal{M}, t\rangle \models \delta_x$ implies $\langle \mathcal{M}, t\rangle \leftrightarrows \langle \mathcal{M}, x\rangle$.

4 Undefinability

Some (meta-) property Φ is definable over logical language LAN (where LAN is L or L^C) iff there is some $\varphi \in \mathsf{LAN}$ such that for all epistemic states $\langle \mathcal{M}, s\rangle$ we have $\langle \mathcal{M}, s\rangle \models \varphi$ iff $\langle \mathcal{M}, s\rangle$ satisfies Φ. In particular, to say that 'being bisimilar to a finite epistemic state' is definable over LAN means that for every $\langle \mathcal{M}, x\rangle$ with $|\mathcal{M}|$ being finite, there is some formula $\varphi_{\langle \mathcal{M},x\rangle} \in \mathsf{LAN}$ such that, for every epistemic state $\langle \mathcal{N}, y\rangle$,

$$\langle \mathcal{N}, y\rangle \models \varphi_{\langle \mathcal{M},x\rangle} \Leftrightarrow \langle \mathcal{M}, x\rangle \leftrightarrows \langle \mathcal{N}, y\rangle\,. \tag{1}$$

Replacing $\leftrightarrows$ by $\trianglelefteq$ in (1) we have definability of 'being simulated by a finite epistemic state'.

Theorem 1. *The property of being bisimilar to a finite epistemic state is not definable over* L.

Proof. See also Figure 3. Given $N < \omega$, consider the model $\mathcal{S}_N$ consisting of $2N + 1$ states

$$|\mathcal{S}_N| = \{s_n : -N \leq n \leq N\}$$

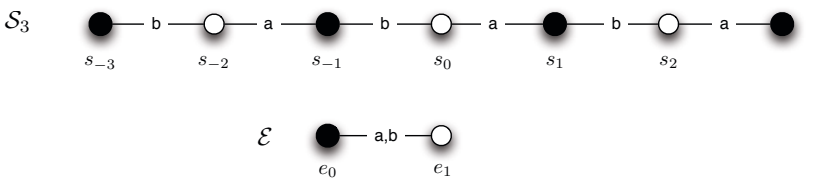


Fig. 3. The models $\mathcal{E}$ and S_3. p-states are black

such that $s_n \overset{a}{\sim} s_{n+1}$ if n is even, $s_n \overset{b}{\sim} s_{n+1}$ if n is odd and

$$[\![p]\!]_{\mathcal{S}_N} = \{s_{2n} \in |\mathcal{S}_N| : n < N\}\,.$$

Let $\mathcal{E}$ be a model with two worlds e_0, e_1, both indistinguishable to both agents and with $[\![p]\!]_{\mathcal{E}} = \{e_0\}$.

Then, clearly, for all N

$$\langle \mathcal{S}_N, s_0\rangle \not\leftrightarrows \langle \mathcal{E}, e_0\rangle \tag{2}$$

yet an easy induction shows that

$$\langle \mathcal{S}_N, s_0\rangle \leftrightarrows_N \langle \mathcal{E}, e_0\rangle \tag{3}$$

Now suppose there would be a formula $\varphi_{\langle \mathcal{E}, e_0\rangle}$ such that for all states $\langle \mathcal{M}, x\rangle$, we would have $\langle \mathcal{M}, x\rangle \models \varphi_{\langle \mathcal{E}, e_0\rangle}$ iff $\langle \mathcal{M}, x\rangle \leftrightarrows \langle \mathcal{E}, e_0\rangle$. Let k be the modal depth of $\varphi_{\langle \mathcal{E}, e_0\rangle}$. Since every model is bisimilar to itself, we would have $\langle \mathcal{E}, e_0\rangle \models \varphi_{\langle \mathcal{E}, e_0\rangle}$. Also, by (3) we would conclude $\langle \mathcal{S}_k, s_0\rangle \models \varphi_{\langle \mathcal{E}, e_0\rangle}$, which would then yield, by our assumption, that $\langle \mathcal{S}_k, s_0\rangle \leftrightarrows \langle \mathcal{E}, e_0\rangle$, which contradicts (2).

Theorem 2. *The property of being simulated by a finite epistemic state is not definable over* L^C.

Proof. Let $\mathcal{E}$ be as in the proof of Theorem 1. We will define two epistemic states $\langle \mathcal{H}, 0\rangle$ and $\langle \mathcal{H}', 0\rangle$ satisfying the following properties:

1. for all $\varphi \in \mathsf{L}^C$: $\langle \mathcal{H}, 0\rangle \models \varphi$ iff $\langle \mathcal{H}', 0\rangle \models \varphi$.
2. $\langle \mathcal{E}, e_0\rangle \trianglelefteq \langle \mathcal{H}', 0\rangle$
3. not $\langle \mathcal{E}, e_0\rangle \trianglelefteq \langle \mathcal{H}, 0\rangle$.

Consider the two variants of a 'spider' (See also Figure 4.) The first of those is $\mathcal{H}$ given by

$$|\mathcal{H}| = \{0\} \cup \{\langle n, m\rangle : 0 < |n| < m\}$$

with $\langle n, m\rangle \overset{a}{\sim} \langle n+1, m\rangle$ if n is odd, $\langle n, m\rangle \overset{b}{\sim} \langle n+1, m\rangle$ if n is even, $0 \overset{a}{\sim} \langle 1, m\rangle$ and $0 \overset{b}{\sim} \langle -1, m\rangle$ for all m and

$$[\![p]\!]_{\mathcal{H}} = \{\langle n, m\rangle : n \text{ is odd}\}\,.$$

Then consider the spider $\mathcal{H}'$, defined as $\mathcal{H}$ but with $\{\langle n, \omega\rangle : n \in \mathbb{Z} \setminus \{0\}\}$ added (this set represents two 'infinite legs' of the spider).

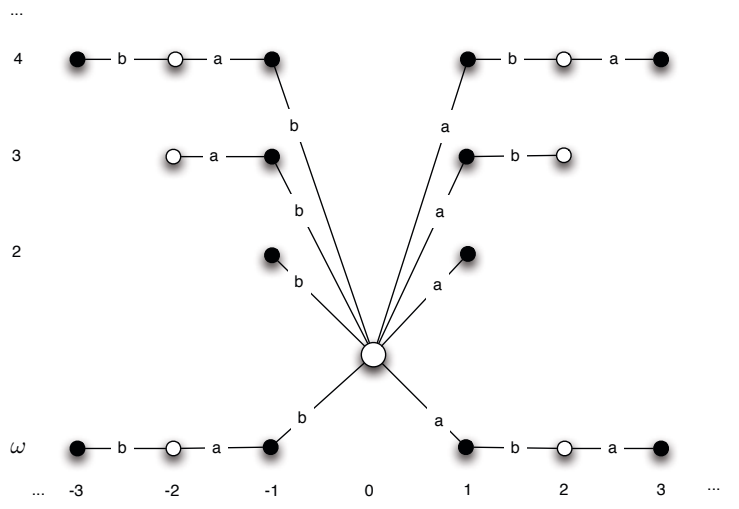


Fig. 4. Part of the spider $\mathcal{H}'$ ($0 < |n| < 5$ and $m \in \{2, 3, 4, \omega\}$). p-states are black, $\neg p$-states are white. The large state at the center is 0. The two bottom legs are infinite; $\mathcal{H}$ lacks them.

Then, $\langle \mathcal{H}, 0\rangle$ and $\langle \mathcal{H}', 0\rangle$ satisfy the same set of L^C formulas; we omit the proof, which uses common knowledge games (cf. [11]).

Now, being simulated by $\langle \mathcal{E}, e_0\rangle$ is equivalent to there being an infinite sequence $\langle s_n : n \in \mathbb{Z}\rangle$ of possibly repeating states such that $s_0 = 0$, $s_{n+1} \overset{a}{\sim} s_n$ if n is even, $s_{n+1} \overset{b}{\sim} s_n$ if n is odd, and s_n satisfies p if and only if n is odd. To see this, suppose that $\mathcal{N}$ is some model, and $S \subseteq |\mathcal{E}| \times |\mathcal{N}|$ is a non-empty simulation. Then, there is some $s_0 \in |\mathcal{N}|$ such that $e_0\ S\ s_0$. Now, because S is a simulation and $e_1 \overset{a}{\sim}_{\mathcal{E}} e_0$, there is $s_1 \overset{a}{\sim} s_0$ such that $e_1\ S\ s_1$; similarly, $e_1 \overset{b}{\sim}_{\mathcal{E}} e_0$, so there is $s_{-1} \overset{b}{\sim} s_0$ such that $e_1\ S\ s_{-1}$. Now, to find s_2 we note that $e_1 \overset{b}{\sim}_{\mathcal{E}} e_0$, so because S is a simulation there must be $s_2 \overset{b}{\sim} s_1$ with $e_0\ S\ s_2$.

Continuing in this fashion (in both directions) we obtain s_n for each n. Note that the actual point may be repeated, but this is not important.

Now, some inspection will show that $\mathcal{H}'$ provides such a sequence (on its ω-branch). More precisely, we define a simulation $S \subseteq |\mathcal{E}| \times |\mathcal{H}'|$ with $e_0\ S\ 0$ and $e_i\ S\ \langle n, \omega\rangle$ if and only if $i \equiv n \pmod 2$.

However, $\mathcal{H}$ does not have such an infinite sequence, so $\langle \mathcal{E}, e_0\rangle \trianglelefteq \langle \mathcal{H}', 0\rangle$ but $\langle \mathcal{E}, e_0\rangle \not\trianglelefteq \langle \mathcal{H}, 0\rangle$, as claimed.

5 Definability

When describing a state s in a modal structure using a formula χ_s, what matters is the factual description σ_s of s, and a description χ_t of every accessible state t together with a statement that nothing else is accessible. So the basic pattern for state s is $\sigma_s \wedge \bigwedge_{Rst} M\chi_t \wedge K \bigvee_{Rst} \chi_t$, where, in turn, each χ_t has a pattern

similar to χ_s. This is the basis of the 1970s Jankov-Fine construction to describe finite modal tree structures [3]. The unwinding of an $S5$-structure is an infinite tree, where this does not work, unless we have infinitary modal operators or an infinitary language at our disposition. In [2] the authors show that any model $\langle M, s\rangle$ has a characteristic formula in the modal language with infinitary conjunctions and disjunctions. In the subsequent [9] an alternative route is taken via infinitary modal operators. In an older setting, [4] tackle the issue for CTL and CTL* and prove characterization of finite models in CTL. The modus operandi in [2,9] is to introduce fresh variables p_s for a given modal structure, one for each state s, and describe the fixpoint using these fresh variables p_s. This serves to characterize modal structures but at the price of going to an extended language with an infinite stock of variables. The *explicit* purpose of these fresh variables is to make each state unique. The *implicit* justification for this procedure is that it does not matter if we change a model structure in the value of 'irrelevant' variables (in an $S5$ setting one could imagine a variable to be irrelevant if it is commonly known to be true or false, so that—psychologizing—it can be removed from the vocabulary in which agents reason about their uncertainty). But this approach does not serve our present goals, for two reasons: firstly, we want to give a characteristic formula in a given logical language, i.e., with a given set of propositional variables, and secondly, the procedure of introducing fresh variables makes bisimilar states s and t non-bisimilar, even at the **Atoms** level.

So, to distinguish a state from other states, state descriptions σ_s are too weak but fresh variables p_s are too strong. There is a way in between. Given a finite multi-agent $S5$ model, iterating the (multi-agent version of the) Jankov-Fine construction above up to the number of states in a model, for each s we can construct a distinguishing formula δ_s in $\mathcal{L}$. Clearly, this construction may be costly. Van Benthem also mentions the distinguishing formulae δ_s in [9], in the setting of PDL. Another result in [9] is that every finite $\langle M, s\rangle$ has a characteristic formula in PDL with the Kleene *, using the existence of such distinguishing formulas δ_s. In [9] in fact only the unimodal case is considered, over arbitrary K-models. This can be generalized to a set of PDL-action labels $a, b, \ldots$; we fill in the details here (see also [10,12]). Given a finite set of action labels $a_1, \ldots, a_n = A$, we can see the corresponding dynamic modal operators $[a_1], \ldots, [a_n]$ also as epistemic modal operators $K_{a_1}, ..., K_{a_n}$; and instead of the Kleene-* applied to the *choice* between *all* these operators (a crucial detail!), i.e., $(a_1 + ... + a_n)^*$, interpreted by the accessibility relation $(R_{a_1} \cup \cdots \cup R_{a_n})^*$, we take the common knowledge operator C_A. Given that translation, we can adapt [9, Proposition 3].

Lemma 1. *Each finite epistemic state $\langle \mathcal{M}, s\rangle$ is distinguished from all other (non-bisimilar) points in $\mathcal{M}$ by a formula δ_s in epistemic logic* **without** *common knowledge.*

Proof. For any finite model $\mathcal{M}$, the following informal algorithm partitions its domain $S_{\mathcal{M}}$ into its bisimulation equivalence classes, with a distinguishing formula for each class. We give the construction for a single agent and for a finite set

of variables P, the multi-agent case is similar. The algorithm consists of a finite iteration. For step 0, consider the state descriptions σ_s for all states $s \in S_{\mathcal{M}}$, and refine the domain into subsets with the same state description. Before we show step $n+1$ given step n, let us show step 1. For step 1, we proceed with the non-singleton subsets. Consider some such subset $S' \subseteq S_{\mathcal{M}}$. For each $s \in S'$, consider the formulas $\sigma_s \wedge \bigwedge_{t \sim s} M\sigma_t \wedge K \bigvee_{t \sim s} \sigma_t$, and refine S' into subsets satisfying the same such formula. Formally, for step 0, let $\delta_s^0 := \sigma_s$; and for step $n+1$, we get $\delta_s^{n+1} := \sigma_s \wedge \bigwedge_{t \sim s} M\delta_t^n \wedge K \bigvee_{t \sim s} \delta_t^n$. Proceed iteratively until all sets are singletons or until a maximum of $|S_{\mathcal{M}}|$ steps is reached. As the outcome of this process, each state is accompanied by a distinguishing formula δ_s, where $\delta_s = \delta_s^m$ for some $m \leq |\mathcal{M}|$. Note that all δ_s are in $\mathcal{L}$. In this construction, bisimilar states will end up in the same subset.[1]

Theorem 3. *Each finite epistemic model $\mathcal{M}$ is characterised by a formula in epistemic logic* **without** *common knowledge; that is, given finite $\mathcal{M}$ there exists a formula $\chi_{\mathcal{M}} \in \mathsf{L}$ such that, for every (possibly infinite) epistemic model $\mathcal{N}$, $\mathcal{N} \models \chi_{\mathcal{M}}$ if and only if $\mathcal{M} \leftrightarroweq \mathcal{N}$.*

Proof. Let $\mathcal{M}$ be finite, and without loss of generality, suppose it contraction-minimal, i.e., $\mathcal{M}$ is not bisimilar to a smaller model. Also assume that it is generated. Define

$$\mathrm{Forth}_x = \bigwedge_{a \in A} \bigwedge_{y \overset{a}{\sim} x} M_a \delta_y \text{ and } \mathrm{Back}_x = \bigwedge_{a \in A} K_a \bigvee_{y \overset{a}{\sim} x} \delta_y;$$

$$\chi_{\mathcal{M}} = \bigvee_{x \in |\mathcal{M}|} \delta_x \wedge \bigwedge_{x \in |\mathcal{M}|} (\delta_x \to \mathrm{Forth}_x \wedge \mathrm{Back}_x) \tag{4}$$

Now the following are equivalent, for any $\langle \mathcal{N}, t \rangle$:

- $\mathcal{M} \leftrightarroweq \mathcal{N}$
- $\mathcal{N} \models \chi_{\mathcal{M}}$.

Since $\mathcal{M}, s \models \chi_{\mathcal{M}}$, any bisimilar model $\mathcal{N}$ also satisfies $\chi_{\mathcal{M}}$. For the converse, If $\mathcal{N} \models \chi_{\mathcal{M}}$, then we can define a relation

$$B \subseteq |\mathcal{M}| \times |\mathcal{N}|$$

given by $x \; B \; y$ if and only if $y \in [\![\delta_x]\!]_{\mathcal{N}}$. Then, it remains to be checked that B is a bisimulation; indeed, if $x \; B \; y$ and $x' \overset{a}{\sim} x$, then we have that y satisfies δ_x and, since

$$\langle \mathcal{N}, t \rangle \models \delta_x \to \mathrm{Forth}_x$$

(use $\chi_{\mathcal{M}}$), it follows that y satisfies $M_a \delta_{x'}$, and thus there is $y' \overset{a}{\sim} y$ such that y' satisfies $\delta_{x'}$, i.e., $x' \; B \; y'$. Thus B satisfies **Forth**.

A similar argument shows that B satisfies **Back** as well, and hence it is a bisimulation.

[1] The maximum $|S_{\mathcal{M}}|$ of the iteration is also mentioned as Facts 7 and 8 in [9, p.30]; but no actual construction is given there. A similar maximum can be construed in the CTL setting of the older publication [4].

Compare this to the following, proven in [10]:

Theorem 4. *Given a finite model $\mathcal{M}$ and $x \in |\mathcal{M}|$, there is a formula $\chi_{\langle \mathcal{M},x\rangle} \in \mathsf{L}^C$ such that, given any finite state $\langle \mathcal{N}, y\rangle$, $\langle \mathcal{N}, y\rangle \models \chi_{\langle \mathcal{M},x\rangle}$ if and only if $\langle \mathcal{N}, y\rangle \leftrightarrows \langle \mathcal{M}, x\rangle$.*

Note that, by Theorem 1, the use of common knowledge is essential. Indeed, common knowledge is used as follows:

$$\chi_{\langle M,s\rangle} = \delta_s \wedge C_A \bigwedge_{x\in|\mathcal{M}|} (\delta_x \rightarrow \mathrm{Forth}_x \wedge \mathrm{Back}_x) \tag{5}$$

We observe that common knowledge is used rather weakly, namely in the form of a single occurrence of that operator. Compare this with Theorem 3 above, that does not have the common knowledge operator. We further point out the resemblance with the distinguishing formula construction in the proof of Lemma 1. There, we use the Yankov-Fine construction iteratively up to the size of the finite model, and with building units the state descriptions. Whereas for the characteristic formula we use the Yankov-Fine construction as a conjunction over all states in the model, closed by common knowledge, and with building units the distinguishing formulas.

Now we turn our attention to a setting in which simulability does become definable:

Definition 3 (Well-multifounded models). *An epistemic model $\mathcal{M}$ is* well-multifounded *if there does not exist an infinite sequence of states $s_n \overset{a(n)}{\sim} s_{n+1}$ such that for all n, $s_n \neq s_{n+1}$ and $a(n) \neq a(n+1)$.*

The model $\mathcal{M}_1$ from Figure 1 is well-multifounded, the models $\mathcal{M}_2$ and $\mathcal{M}_3$ from Figure 2 are not.

Theorem 5. *If $\mathcal{M}$ is well-multifounded and finite and $s \in |\mathcal{M}|$, then simulability by $\langle \mathcal{M}, s\rangle$ is definable over L.*

Proof. Suppose that $\mathcal{M}$ is well-multifounded and finite. We first note that given two distinct states $s, t \in |\mathcal{M}|$, there is at most one agent $a \in A$ such that $s \overset{a}{\sim} t$; otherwise, this would immediately give us an infinite loop $s, t, s, t, s...$ violating well-multifoundedness.

Now, pick $s_* \in |\mathcal{M}|$. We will show, by induction on the size of $\mathcal{M}$, that there is a formula $\mathrm{Sim}(\langle \mathcal{M}, s_*\rangle)$ defining the property of being simulated by $\langle \mathcal{M}, s_*\rangle$.

The base case, when $\mathcal{M}$ has no states, is vacuously true.

Suppose, then, that a formula $\mathrm{Sim}(\langle \mathcal{Y}, s\rangle)$ exists defining the property of being simulated by $\langle \mathcal{Y}, t\rangle$ whenever $\mathcal{Y}$ is well-multifounded and has strictly less states than $\mathcal{M}$.

Consider

$$\mathcal{N} = \mathcal{M} \restriction (|\mathcal{M}| \setminus \{s_*\}),$$

i.e., the model obtained by deleting s_* from $\mathcal{M}$.

Because $\mathcal{N}$ is smaller than $\mathcal{M}$, for every $t \in |\mathcal{N}|$ there is a formula $\mathrm{Sim}(\langle \mathcal{N}, t\rangle)$ which defines simulability by $\langle \mathcal{N}, t\rangle$; that is, such that given any epistemic model $\mathcal{X}$ and $x \in |\mathcal{X}|$, $\langle \mathcal{N}, t\rangle \trianglelefteq \langle \mathcal{X}, x\rangle$ if and only if $\mathcal{X}, x \models \mathrm{Sim}(\langle \mathcal{N}, t\rangle)$.

Recall that $\sigma(s)$ is the conjunction of all literals (propositional variables or their negation) which are true on s. Define

$$\mathrm{Sim}(\langle \mathcal{M}, s_*\rangle) = \sigma(s_*) \wedge \bigwedge_{a \in A} \bigwedge_{t \overset{a}{\sim}_{\mathcal{M}} s_*} \mathrm{Sim}(\mathcal{N}, t).$$

We claim that $\mathrm{Sim}(\langle \mathcal{M}, s_*\rangle)$ defines being simulated by $\langle \mathcal{M}, s_*\rangle$ over the class of epistemic models, that is, given any state $\langle \mathcal{X}, x\rangle$ we have that

$$\langle \mathcal{M}, s_*\rangle \trianglelefteq \langle \mathcal{X}, x\rangle \Leftrightarrow \langle \mathcal{X}, x\rangle \models \mathrm{Sim}(\langle \mathcal{M}, s_*\rangle).$$

We consider each direction separately.

Left-to-right. Assume that $\langle \mathcal{M}, s_*\rangle \trianglelefteq \langle \mathcal{X}, x\rangle$, so that there is a simulation

$$S \subseteq |\mathcal{M}| \times |\mathcal{X}|$$

with $s_* \; S \; x$. We need to show that $\mathcal{X}, x \models \mathrm{Sim}(\langle \mathcal{M}, s_*\rangle)$.

Clearly $x \in [\![\sigma(s_*)]\!]_{\mathcal{X}}$. Now, if $t \overset{a}{\sim}_{\mathcal{M}} s_*$, because S is a simulation there is $y \overset{a}{\sim}_{\mathcal{X}} x$ such that $t \; S \; y$.

We need to show that $y \in [\![\mathrm{Sim}(\langle \mathcal{N}, t\rangle)]\!]_{\mathcal{X}}$ to establish our claim; but by our induction hypothesis, it suffices to observe that $S \restriction |\mathcal{N}|$ is a simulation (the 'forth' condition is clearly preserved by submodels), and by assumption the existence of such a simulation implies that y satisfies $\mathrm{Sim}(\mathcal{N}, t)$.

Looking at the definition of $\mathrm{Sim}(\langle \mathcal{M}, s_*\rangle)$, it follows that $x \in [\![\mathrm{Sim}(\langle \mathcal{M}, s_*\rangle)]\!]_{\mathcal{X}}$, as desired.

Right-to-left. Suppose that $x_* \in [\![\mathrm{Sim}(\langle \mathcal{M}, s_*\rangle)]\!]_{\mathcal{X}}$; we need to construct a simulation $S \subseteq |\mathcal{M}| \times |\mathcal{X}|$ such that $s_* \; S \; x_*$. In fact, for our inductive argument to work it is convenient to define S so that it is a function; let us call a simulation which is a function an *embedding*. Let B be the set of all agents a such that there is $t \neq s_*$ with $t \overset{a}{\sim}_{\mathcal{M}} s_*$. For each $a \in B$ pick a fixed representative $d(a)$ with the property that $d(a) \overset{a}{\sim}_{\mathcal{M}} s_*$.

Let $\mathcal{N}_a$ define the submodel of $\mathcal{N}$ generated by $d(a)$. We claim:

1. each $\mathcal{N}_a$ is well-multifounded.
   (Since well-multifoundedness is preserved under submodels);
2. $\mathcal{N}_a$ does not depend on the specific choice of $d(a)$; that is, for all $t \overset{a}{\sim}_{\mathcal{M}} s_*$, the model $\mathcal{N}_a$ equals the submodel of $\mathcal{N}$ generated by t.
   (If $t \overset{a}{\sim}_{\mathcal{M}} s_*$, by transitivity we have that $t \overset{a}{\sim} d(a)$, so $t \in |\mathcal{N}_a|$; similarly, $d(a)$ is contained in the submodel generated by t, so the two are equal.)
3. $|\mathcal{N}_a| \cap |\mathcal{N}_b| = \varnothing$ whenever $a \neq b$.
   (If $t \in |\mathcal{N}_a| \cap |\mathcal{N}_b|$, we have a path connecting $d(a)$ to t and another connecting $d(b)$ to t (this is the definition of lying in the generated submodel). 'Pruning'

the path if necessary, we can assume that points do not repeat and the agents connecting consecutive points are distinct.
These two paths give us a 'loop' which begins on s_* and passes through $d(a)$, then t, then $d(b)$ and back to s_*; we can then run this loop infinitely many times to obtain a sequence violating well-multifoundedness. We conclude that there is no such t.)

For every $a \in B$ and $t \overset{a}{\sim} s_*$, we have $x_* \in [\![M_a \mathrm{Sim}(\langle \mathcal{N}, t\rangle)]\!]_{\mathcal{X}}$ (by $\mathrm{Sim}(\langle \mathcal{M}, s_*\rangle)$); in particular,

$$x_* \in [\![M_a \mathrm{Sim}(\langle \mathcal{N}, d(a)\rangle)]\!]_{\mathcal{X}}\,.$$

By induction we can assume that there is $y(a) \overset{a}{\sim}_{\mathcal{X}} x_*$ and an embedding $S_a \subseteq |\mathcal{N}_a| \times |\mathcal{X}|$ such that $d(a)\ S_a\ (d(a)) = y(a)$[2]. Define

$$S = \{\langle s_*, x_*\rangle\} \cup \bigcup_{a \in B} S_a.$$

We claim that S is an embedding: clearly it preserves propositional variables; it remains to check that the 'forth' condition holds. It is easy to see that S is indeed a function, since we have only added the pair $\langle s_*, x_*\rangle$ to a disjoint union of functions (with disjoint domains). Thus the 'forth' condition becomes: if $s \overset{a}{\sim}_{\mathcal{M}} t$ then $S(s) \overset{a}{\sim}_{\mathcal{X}} S(t)$.

To prove this, pick $s \overset{a}{\sim}_{\mathcal{M}} t$ and suppose that $x = S(s)$. There are essentially three cases we must consider. (1) $s, t \in \mathcal{N}$. They both lie in some $\mathcal{N}_a$, since these models are generated; but then S_a is an embedding, so $S_a(s) \overset{a}{\sim}_{\mathcal{X}} S_a(t)$. (2) $s = s_*$, so that $S(s) = x_*$. By transitivity of $\overset{a}{\sim}_{\mathcal{M}}$ we have that $t \overset{a}{\sim}_{\mathcal{M}} d(a)$, hence $S_a(t) \overset{a}{\sim}_{\mathcal{X}} y(a)$ and, by transitivity of $\overset{a}{\sim}_{\mathcal{X}}$, we have that $S_a(t) \overset{a}{\sim}_{\mathcal{X}} x_*$, which is what we needed. (3) $t = s_*$. Then $s \in |\mathcal{N}_a|$, and because these sets are disjoint it is the only agent for which this holds. Thus $S(s) = S_a(s)$. Now, $s \overset{a}{\sim}_{\mathcal{M}} s_* \overset{a}{\sim}_{\mathcal{M}} d(a)$, so by transitivity $s \overset{a}{\sim}_{\mathcal{M}} d(a)$ and, because S_a is a simulation, $S_a(s) \overset{a}{\sim}_{\mathcal{X}} y(a) \overset{a}{\sim}_{\mathcal{X}} x_*$, as desired.

6 Conclusion and Further Research

When building multi-agent systems applications, in case you want to execute some dynamics, you often want to apply the action 'right here in the model'. Unless you have nominals, you can enforce this by preconditions that distinguish that state from other states. This can be done in logic without common knowledge. However, there are different cases wherein you want to pin down the current state including all its epistemic aspects. 'Only under *exactly* these conditions, this action will apply'. Then you need a characteristic formula and *that* can only be done in epistemic logic with common knowledge. This

[2] Properly speaking, we begin with an embedding S'_a between $\mathcal{N}$ and $\mathcal{X}$ and restrict it to $\mathcal{N}_a$.

significant distinction may not always be observed. We hope that our results, already known in the communities of PDL and CTL, may therefore contribute to logical hygiene in epistemic applications.

Computing characteristic formulas employing distinguishing formulas may be costly. For specific model classes the characteristic formulas may be simpler. Static interpreted systems [5] consist of a domain of global states, where each global state is a collection of local states, one for each agent. If agents only know their local state, the description of the global state (i.e., the description of the valuation) is a distinguishing formula in the Kripke model for that interpreted system. (See also [7] on the relation between interpreted systems wherein agents *only* know their local state, and Kripke models, by way of 'hypercubes'.) Also, for such systems, the characteristic formula has a special shape wherein factual (purely propositional), positive, and negative knowledge can be separated as different parts of the formula. The factual part says that one of all global states in the system must be the case (this is therefore a large disjunction of state descriptions); the positive part says that each agent knows her local state, and the negative part sums up the ignorance. This approach is followed in [12] for interpreted systems modelling card deals—but it can clearly be generalized. The constructions in the previous section, instead, consist of many conjuncts in each of which all these three types of formula are mixed. The modular form of characteristic formulas for such interpreted systems is an advantage, because when there are dynamic developments in the system it allows us to focus on the changing parts of the formula: factual and positive knowledge are always preserved, and only the negative knowledge, the ignorance, is reduced.

Here our results on simulability are also useful, since simulation is intimately related with epistemic actions. Roughly speaking, epistemic actions are events by which agents learn new information, and given an epistemic model $\mathcal{M}$, any submodel $\mathcal{N}$ of $\mathcal{M}$ (and even more: if $\mathcal{N}$ results from executing whatever action model in $\mathcal{M}$, then $\mathcal{N}$ is simulated by $\mathcal{M}$) models a situation where agents have more positive knowledge than they had on $\mathcal{M}$, since each new world on $\mathcal{M}$ represents a new source of uncertainty for the agents. *Simulation* is the bisimulation-invariant analogue of being a submodel; that is, $\mathcal{N} \trianglelefteq \mathcal{M}$ is only slightly more general than to say that $\mathcal{N}$ is bisimilar to a submodel of $\mathcal{M}$.

Thus defining submodels up to simulability helps us interiorize dynamics into the syntax by fully describing the effect of certain epistemic actions on a model. This could be used to turn a static description of a model into a dynamic one, by describing which epistemic states may result after executing those actions which are available to the agents.

For future research, it is also worthwhile to determine (and lower) the complexity of the computation of distinguishing formulas, the building stones of the characteristic formulas. Complexity worries were already uttered in [4]—but we do not know of any subsequent resolution. In the case of static interpreted systems where state descriptions are already distinguishing, this is clearly more efficient.

Acknowledgment

We thank the CLIMA reviewers for comments. Hans van Ditmarsch is also affiliated to IMSC (Institute of Mathematical Sciences Chennai), India, as associated researcher.

References

1. Aczel, P.: Non-Well-Founded Sets. CSLI Lecture Notes, vol. 14. CSLI Publications, Stanford (1988)
2. Barwise, J., Moss, L.S.: Vicious Circles. CSLI Publications, Stanford (1996)
3. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press, Cambridge (2001)
4. Browne, M., Clarke, E., Grümberg, O.: Characterizing Kripke structures in temporal logic. In: Ehrig, H., Levi, G., Montanari, U., Kowalski, R. (eds.) CAAP 1987 and TAPSOFT 1987. LNCS, vol. 249, pp. 256–270. Springer, Heidelberg (1987)
5. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning About Knowledge. MIT Press, Cambridge (1995)
6. Fernández-Duque, D.: On the modal definability of simulability by finite transitive models. In: Studia Logica (forthcoming, 2011)
7. Lomuscio, A.R., Ryan, M.D.: On the relation between interpreted systems and kripke models. In: Wobcke, W.R., Pagnucco, M., Zhang, C. (eds.) Agents and Multi-Agent Systems Formalisms, Methodologies, and Applications. LNCS (LNAI), vol. 1441, pp. 46–59. Springer, Heidelberg (1998)
8. Meyer, J.-J.C., van der Hoek, W.: Epistemic Logic for AI and Computer Science. Cambridge Tracts in Theoretical Computer Science, vol. 41. Cambridge University Press, Cambridge (1995)
9. van Benthem, J.: Dynamic odds and ends. ILLC Technical Report ML (1998)
10. van Benthem, J.: 'One is a lonely number': on the logic of communication. In: Chatzidakis, Z., Koepke, P., Pohlers, W. (eds.) Logic Colloquium 2002. Lecture Notes in Logic, vol. 27. Association for Symbolic Logic (2002)
11. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Springer, Berlin (2007)
12. van Ditmarsch, H., van der Hoek, W., Kooi, B.P.: Descriptions of game states. In: Mints, G., Muskens, R. (eds.) Logic, Games, and Constructive Sets. CSLI Lecture Notes, vol. 161, pp. 43–58. CSLI Publications, Stanford (2003)

Applications of Logic in Social Choice Theory
(Invited Talk)

Ulle Endriss

Institute for Logic, Language and Computation (ILLC)
University of Amsterdam

Abstract. Social choice theory studies of how groups of people should and do make collective decisions. In this talk I will argue that modern logic can contribute to the study of social choice theory in many different ways, and I will substantiate this claim with examples from recent work by members of my group at the University of Amsterdam.

1 Social Choice Theory

Social choice theory (SCT) is the formal study of mechanisms for collective decision making. As a scientific discipline, it is usually considered to be part of Economic Theory, although it also plays an important role in Political Science and Philosophy. In recent years, furthermore its fundamental significance for work in Multiagent Systems has become to be widely recognised.

The archetypal problem in the field is *preference aggregation:* given the preferences of a number of agents over a set of alternatives, how should we aggregate these individual preferences so as to arrive at a single collective preference order? To see that this is not a trivial question, consider the following example. There are three alternatives, called A, B and C, and five agents. The preferences of each agent are modelled as a linear order over the set of alternatives:

Agent 1:	$A \succ B \succ C$
Agent 2:	$B \succ C \succ A$
Agent 3:	$C \succ A \succ B$
Agent 4:	$C \succ A \succ B$
Agent 5:	$B \succ C \succ A$

The most obvious approach for obtaining a collective preference order is to use the *majority rule:* rank X above Y if and only if a majority of the agents do. If we follow this rule, then we must adopt $A \succ B$ (as three of the agents do), $B \succ C$ (as again three of the agents do), and $C \succ A$ (as four of the agents do). But this means that we get a collective preference order that is cyclic! This surprising outcome is an instance of the *Condorcet Paradox*, named after the 18th century political scientist and mathematician who first discussed it at length.

The question now arises whether there is a better aggregation rule than the majority rule, one that does not suffer from this paradox. Social choice theorists have approached this question using the so-called "axiomatic method": by

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 88–91, 2011.

formulating desirable properties of aggregation rules as "axioms" in a mathematically rigorous manner, they have been able to obtain results that show that it is impossible to find a rule that satisfies a certain combination of desirable properties or that a certain combination of such properties uniquely characterises a particular rule. Famous examples include Arrow's Theorem (showing that there exists no preference aggregation rule for three or more alternatives that respects unanimous choices made by the individuals, that ranks pairs of alternatives independently from how other alternatives are ranked, and that is not dictatorial); the Gibbard-Satterthwaite Theorem (showing that there exists no voting rule for three or more alternatives that does not exclude an alternative from winning *a priori*, that does not allow for situations in which a voter can benefit from submitting a ballot that does not truthfully reflect her actual preferences, and that is not dictatorial); and May's Theorem (showing that for two alternatives the majority rule is the only aggregation rule that treats all agents and alternatives symmetrically and that respects a basic monotonicity condition).

2 Applications of Logic

Logic has long played an important role in SCT: for instance, some properties of aggregation rules *entail* other properties, and impossibility theorems establish the *inconsistency* of certain sets of properties.

However, this use of "logic" is rather informal in nature. While it does refer to logical concepts such as "consistency", it does not make use of a formal language. In the sequel I will argue that logic, including formal symbolic logic, has many more applications in SCT, and I will substantiate this claim with examples from recent work by members of my group at the University of Amsterdam.[1]

2.1 Representation of Preferences

Before we can tackle the problem of aggregating preferences, we need to decide how to model the preferences themselves. In classical SCT, preferences are taken to be linear (or weak) orders over the set of alternatives, but other types of preference structures are also of interest (see e.g. [4]).

How to actually *represent* preferences, using a formal language, becomes critical when alternatives have a combinatorial structure, e.g., when agents are asked to express their preferences over all combinations of assigning truth values to, say, ten variables. (For an introduction to the field of preference modelling for *social choice in combinatorial domains*, see the expository article authored jointly with Chevaleyre et al. [1].) One family of languages proposed is based on *weighted goals:* agents describe their preferences in terms of propositional formulas they would like to see satisfied, together with numerical weights indicating their

[1] This is not intended to be a comprehensive review of the field; in particular, references are restricted to my own work. For extensive references to the use of logic in SCT, as well as to work in classical SCT and modern computational social choice more generally, please refer to the bibliographies of the cited papers.

relative importance. Besides their application in SCT, also the study of the properties of such languages themselves, e.g., their expressive power, has lead to interesting research questions [11,12].

Much of this work requires only classical propositional logic, but nonclassical logic also has a role to play. For instance, for resource allocation problems where there may be multiple copies of the same type of resource available, *linear logic* turns out to be the right kind of formalism to represent preferences [9].

2.2 Characterisation and Impossibility Results

A natural application of logic in SCT is to attempt to fully formalise parts of the field. For instance, we have been able to give a full formalisation, in classical first-order logic, of the framework of preference aggregation introduced by Arrow and we have shown that Arrow's Theorem corresponds to the claim that a particular set of first-order formulas does not have a finite model [6].

In another line of work we have shown that paradoxes of social choice, such as the Condorcet Paradox, can be explained in terms of the violation of an integrity constraint, expressed in propositional logic, that characterises the domain of aggregation [7,8]. This approach provides a new way of characterising aggregation rules, namely in terms of the types of integrity constraints it respects, as well as a new proof technique for deriving old and new impossibility theorems in SCT.

2.3 Automated Reasoning in Social Choice Theory

One motivation for seeking to formalise SCT is that it opens up the possibility of using automated theorem provers to verify known results. For instance, a by-product of our work on the formalisation of the Arrovian framework of preference aggregation is a specification of that framework in the language of `Prover9`, the automated theorem prover formerly known as `Otter` [6]. Further optimisation may one day lead to a fully automated proof of Arrow's Theorem.

In another area of SCT, known as *ranking sets of objects*, we already have been able to obtain results in a fully automated manner. Here the choice-theoretic problem is how to extend an agent's preferences over individual objects to a preference order over sets of such objects. Building on a model-theoretic result, we have been able to use a satisfiability checker to not only verify known impossibility theorems but also to discover nontrivial new results [5].

2.4 Judgment Aggregation

Logic is not only useful for the analysis of aggregation problems, but information expressed in terms of logic may itself be subject to aggregation. This kind of problem is studied in the field of *judgment aggregation* (JA). Suppose three agents each have to judge which of the formulas φ, ψ and $\varphi \wedge \psi$ are true:

	φ	ψ	$\varphi \wedge \psi$
Agent 1:	true	true	true
Agent 2:	true	false	false
Agent 3:	false	true	false

Observe that each agent provides a logically consistent set of judgments. How should we aggregate this information? If we use the *majority rule* for each proposition, then φ should be collectively accepted, ψ should also be collectively accepted, and $\varphi \wedge \psi$ should be collectively rejected—that is, we obtain an inconsistent judgment set! Over the past decade or so, this paradox of JA has given rise to a fast moving area of research, spanning Legal Theory, Philosophy, Economic Theory and AI. For instance, we have recently begun to analyse the *computational complexity* of a number of problems that naturally arise in JA [2,3].

While originally associated with problems in legal reasoning, it is not hard to see that JA can have a range of significant applications in other fields as well. One of them I believe to be the Semantic Web, and more specifically the aggregation of knowledge distributed over a number of different ontologies [10].

References

1. Chevaleyre, Y., Endriss, U., Lang, J., Maudet, N.: Preference handling in combinatorial domains: From AI to social choice. AI Magazine 29(4), 37–46 (2008)
2. Endriss, U., Grandi, U., Porello, D.: Complexity of judgment aggregation: Safety of the agenda. In: Proc. 9th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS-2010 (2010)
3. Endriss, U., Grandi, U., Porello, D.: Complexity of winner determination and strategic manipulation in judgment aggregation. In: Proc. 3rd International Workshop on Computational Social Choice, COMSOC-2010 (2010)
4. Endriss, U., Pini, M.S., Rossi, F., Venable, K.B.: Preference aggregation over restricted ballot languages: Sincerity and strategy-proofness. In: Proc. 21st International Joint Conference on Artificial Intelligence, IJCAI-2009 (2009)
5. Geist, C., Endriss, U.: Automated search for impossibility theorems in social choice theory: Ranking sets of objects. Journal of Artificial Intelligence Research (JAIR) 40, 143–174 (2011)
6. Grandi, U., Endriss, U.: First-order logic formalisation of arrow's theorem. In: He, X., Horty, J., Pacuit, E. (eds.) LORI 2009. LNCS, vol. 5834, pp. 133–146. Springer, Heidelberg (2009)
7. Grandi, U., Endriss, U.: Lifting rationality assumptions in binary aggregation. In: Proc. 24th AAAI Conference on Artificial Intelligence, AAAI-2010 (2010)
8. Grandi, U., Endriss, U.: Binary aggregation with integrity constraints. In: Proc. 22nd International Joint Conference on Artificial Intelligence, IJCAI-2011 (2011)
9. Porello, D., Endriss, U.: Modelling combinatorial auctions in linear logic. In: Proc. 12th International Conference on the Principles of Knowledge Representation and Reasoning, KR-2010 (2010)
10. Porello, D., Endriss, U.: Ontology merging as social choice. In: Leite, F., Torroni, P., Ågotnes, T., Boella, G., van der Torre, L. (eds.) CLIMA XII 2011. LNCS (LNAI), vol. 6814, pp. 157–170. Springer, Heidelberg (2011)
11. Uckelman, J., Chevaleyre, Y., Endriss, U., Lang, J.: Representing utility functions via weighted goals. Mathematical Logic Quarterly 55(4), 341–361 (2009)
12. Uckelman, J., Endriss, U.: Compactly representing utility functions using weighted goals and the max aggregator. Artificial Intelligence 174(15), 1222–1246 (2010)

A Geometric Look at Manipulation

Jan van Eijck

CWI, Amsterdam

Abstract. We take a fresh look at voting theory, in particular at the notion of manipulation, by employing the geometry of the Saari triangle. This yields a geometric proof of the Gibbard/Satterthwaite theorem, and new insight into what it means to manipulate the vote. Next, we propose two possible strengthenings of the notion of manipulability (or weakenings of the notion of non-manipulability), and analyze how these affect the impossibility proof for non-manipulable voting rules.

1 Introduction

We start with fixing some terminology, mostly following the conventions of [11].

Let A be a finite set of goods, with $|A| > 2$. An *A-ballot* is a linear ordering of A. Let $\{1, \ldots, n\}$ be a set of voters. An (A, n)-*profile* is an n-tuple of A-ballots. If $\mathbf{P}$ is an (A, n)-profile, then $\mathbf{P}$ can be written as $(>_1, \ldots, >_n)$. $>_i$, the i-th component of profile $(>_1, \ldots, >_n)$, is the ballot of voter i. $>_i$ expresses "what voter i wants."

Let $\mathbf{P}(A)$ be the set of all (A, n)-profiles, for given $n \in \mathbb{N}$. A function $V : \mathbf{P}(A) \to A$ is a *resolute voting rule* for A. A function $V : \mathbf{P}(A) \to \mathcal{P}^+(A)$ is a *voting rule* for A. A function $V : \mathbf{P}(A) \to \mathcal{P}^+(A) \to \mathcal{P}^+(A)$ with the property that

$$V(\mathbf{P})(v) \subseteq v$$

is a *social choice function* for A. Let $\mathbf{ord}(A)$ be the set of all linear orderings of A. A function

$$V : \mathbf{P}(A) \to \mathbf{ord}(A)$$

is a *social welfare function* for A. A social welfare function transforms a sequence of ballots into a single ballot.

When a linear preference order $>$ on A is mentioned, we will use $<$ for $\{(x, y) \mid y > x\}$, $\geq$ for $\{(x, y) \mid x > y \vee x = y\}$, and $\leq$ for $\{(x, y) \mid x < y \vee x = y\}$.

Let $\mathbf{P} \sim_i \mathbf{P}'$ express that $\mathbf{P}$ and $\mathbf{P}'$ differ only in the ballot of voter i.

A voting rule satisfies *Pareto* if no x is winning if there is some y that every voter prefers to x:

$$\forall \mathbf{P} \forall x \in V(\mathbf{P}) \forall y \in A(y \neq x \longrightarrow \exists i \in N : \ x \geq_i y).$$

A resolute voting rule V is *non-manipulable (NM)* (or: *strategy-proof*) if $\mathbf{P} \sim_i \mathbf{P}'$ implies $V(\mathbf{P}) \geq_i V(\mathbf{P}')$.

Note that in $\mathbf{P}$, "what voter i wants" is expressed by $>_i$, and in $\mathbf{P}'$, "what voter i wants" is expressed by $>'_i$. If changing the ballot from $>_i$ to $>'_i$ gives a better outcome (better, given $>_i$) than sticking to $>_i$, then the voting rule invites strategic voting.

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 92–104, 2011.

A voting rule V is *non-imposed* if any candidate can be a winner: $\forall a \in A \exists \mathbf{P} : a \in V(\mathbf{P})$. We will use a slightly weaker property. A resolute voting rule V is *weakly non-imposed (NI)* if at least three outcomes are possible: $|\{x \mid \exists \mathbf{P} : V(\mathbf{P}) = x\}| \geq 3$.

A resolute voting rule V is a *dictatorship* if there is some k such that $V : \mathbf{P}(A) \to A$ maps any $\mathbf{P}$ to the top ranking item in $>_k$.

A voter i is *effective* (or: *pivotal*) for V and $\mathbf{P}$ if there is some $\mathbf{P}'$ with $\mathbf{P} \sim_i \mathbf{P}'$ and $V(\mathbf{P}) \neq V(\mathbf{P}')$.

Here is what the famous 'Gibbard-Satterthwaite Theorem [5,10] states:

GS. Any resolute voting rule that is NM and that is NI is a dictatorship.

In this paper we will reflect on the theorem by starting out with giving a new easy proof, and then analyzing what makes the proof so easy.

> The more important a theorem is, the harder we should look for explanations why it is true. And the Gibbard-Satterthwaite theorem is extremely important. [11]

The Gibbard/Satterthwaite theorem is closely related to Arrow's Theorem [1], although the GS theorem is about (resolute) voting rules, while Arrow's theorem is about social welfare functions. A proof in dialogue form of Arrow's theorem, with comments on the connection with Gibbard/Satterthwaite, is in [4]. Geometric proofs of Arrow's theorem exist [9,7]. The geometric approach to paradoxes of preference aggregation from [9] is extended in [3] to paradoxes of judgement aggregation. In jugdgement aggregation not all outcomes are possible because the judgements are logically interconnected.

In this paper we give a geometric proof of the Gibbard-Satterthwaite Theorem. Our first aim is to make (still) clearer *why the GS Theorem is true*. It will turn out that our proof is easy because the notion of manipulability is strong. Next, we will analyse what the proof tells us about the notion of manipulability, and study what some slight modifications of this notion would do to the proof.

2 Geometry of Voting: The Saari Triangle

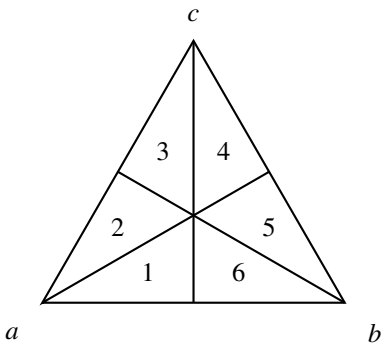
If a voter i changes his ballot from $>_i$ to $>_i'$, then this change can be decomposed into a sequence of adjacent transpositions. E.g., the change from *abcd* to *cbad* can be decomposed into

$$abcd \to bacd \to bcad \to cbad.$$

First a and b are swapped, then c and a, and finally b and c.

We call such an adjacent transposition where x and y are swapped from a situation where x is preferred over y to a situation where y is preferred over x an $x : y$ crossing. Geometrically, an $x : y$ crossing crosses the line between the set of ballots where x is preferred over y to the set of ballot where y is preferred over x.

For the case of three alternatives, this is made clear in the geometry of profiles given by the Saari Triangle from [9]. Following [9], we call call the voters that have ballot $a > b > c$ voters of type 1. The voters of type 2 are those that have ballot $a > c > b$. The voters of type 3 have ballot $c > a > b$. The voters of type 4 have ballot $c > b > a$. The voters of type 5 have ballot $b > c > a$. The voters of type 6 have ballot $b > a > c$.



Here is how to read this. The closer a region is to a vertex, the more preferred the vertex. Now the six regions represent the six types of voters. The subtriangle marked 1 is closest to the a vertex and farthest from the c vertex, so this area represents the $a > b > c$ voters. Note that every time a boundary between regions in the triangle gets crossed, one binary preference gets swapped. E.g., in crossing from the 1 into the 2 region, $b < c$ gets swapped to $c < b$.

Geometrically, $\mathbf{P} \sim_i \mathbf{P}'$ gets represented as a single voter i changing type by moving from one region in the triangle to another.

To allow draws in the ballots one can allow voters positioned on the lines. It is easy to extend all that follows to cover such cases as well; it is left to the reader to check that the argument is not affected by this.

3 A 'Geometric' Proof of the Gibbard Satterthwaite Theorem

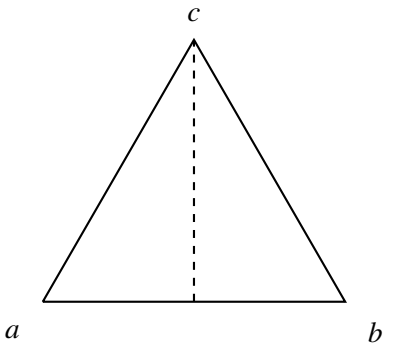
We will now chart the possible effects of minimal type changes, on the assumption that the voting rule satisfies NM. It will turn out that the constraints on vote changing that follow from NM are very strong.

The Crossing Lemma describes the possible effects of $x : y$ crossings on the outcome of the vote, given that the voting rule satisfies NM. Note that the lemma holds for any number of alternatives.

Lemma 1 (Crossing Lemma). *Let V be NM, and let $\mathbf{P} \sim_i \mathbf{P}'$ be such that $>_i$ and $>_i'$ are related by an adjacent transposition that exchanges x and y (with $x >_i y$). Then $V(\mathbf{P}) \neq V(\mathbf{P}')$ implies $V(\mathbf{P}) = x$ and $V(\mathbf{P}') = y$.*

Proof. Suppose V, $\mathbf{P}$, $\mathbf{P}'$, i, x, y are as in the statement of the Lemma. Assume $V(\mathbf{P}) \neq V(\mathbf{P}')$. By NM we have that $V(\mathbf{P}) \geq_i V(\mathbf{P}')$ and $V(\mathbf{P}) \leq_i' V(\mathbf{P}')$. Since the two orderings differ only in the positions of x and y, and x and y are adjacent, it follows that $V(\mathbf{P}) = x$ and $V(\mathbf{P}') = y$. (This continues to hold if ballots are allowed to have ties.) □

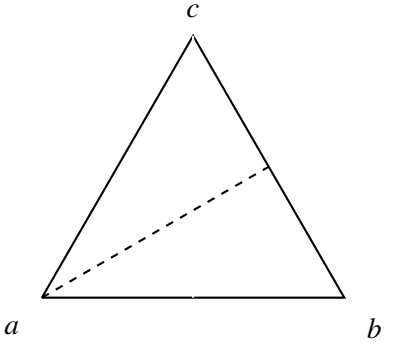
For the Saari triangle, where the regions are arranged by adjacent transpositions, we can visualize the constraints on crossing the $a \sim b$ divide as follows:



$a \sim b$: dividing line between a and b regions

The only vote change that can take place when this line is crossed is from a to b if the line is crossed from the a region into the b region, and from b to a if it is crossed in the other direction.

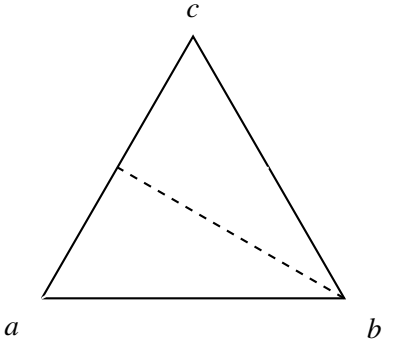
Similarly for the consequences of NM for crossing the $b \sim c$ divide:



$b \sim c$: dividing line between b and c regions

The only shift in the vote (given NM) that can take place when this line is crossed from the b to the c region is from b to c, and vice versa.

Finally, the consequences of NM for crossing the $a \sim c$ divide:



$a \sim c$: dividing line between a and c regions

Crossing from a to c by a single voter can only cause a vote shift from a to c, and crossing in the other direction can only cause a vote shift from c to a.

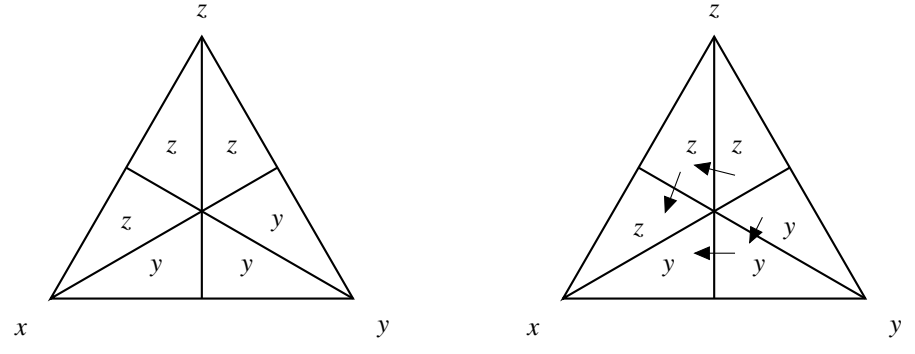
Summing up, we get the following **crossing rule** for crossings in the Saari triangle:

Fact 1. *In crossing the $x \sim y$ divide, from the x region into the y region (call this an $x : y$ crossing) the only value change that is allowed is from x to y.*

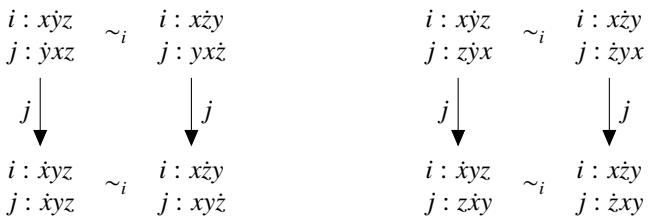
The Crossing Lemma is our main tool to prove the following Effectiveness Lemma:

Lemma 2 (Effectiveness Lemma). *If V is NM and NI, and i is effective for* **P***, then $V(\mathbf{P})$ is equal to the top of the i-ballot in* **P***.*

Proof. Assume V is NM and i is effective for **P**, and suppose for a contradiction that $V(\mathbf{P}) <_i x$, where x is the top element of the i-ballot in **P**. Since i is effective, there is a profile $\mathbf{P}'$ with $\mathbf{P} \sim_i \mathbf{P}'$ and $V(\mathbf{P}) \neq V(\mathbf{P}')$. By NM, $V(\mathbf{P}) >_i V(\mathbf{P}')$. Let $y = V(\mathbf{P})$ and $z = V(\mathbf{P}')$. Then the i-ballot in **P** has the pattern $x \cdots \dot{y} \cdots z$, with $\dot{y}$ indicating that y is the outcome of the vote. Now change all ballots in **P** by pushing values different from x, y, z below the third position, while keeping the order of x, y, z. Let the result be **Q**. Then by the Crossing Lemma, $V(\mathbf{Q}) = V(\mathbf{P}) = y$. So i-ballot and result of the vote in **Q** are given by $x\dot{y}z$. All ballots different from those with one of the six permutations of x, y, z on top are irrelevant for the argument that follows, and we can disregard them. By the Crossing Lemma, the only consistent configuration for how the vote can change as i moves through the relevant ballots is given by: $x\dot{y}z \sim_i x\dot{z}y \sim_i \dot{z}xy \sim_i \dot{z}yx \sim_i \dot{y}zx \sim_i \dot{y}xz$ $(\sim_i x\dot{y}z)$. See the lefthand side picture below.



Suppose some other agent j is able to influence the vote. Then all j can do is make the vote switch to x. The only way of doing that is by moving x up in his ballot. This ballot change can be decomposed into adjacent transposition steps. According to the Crossing Lemma, this is what j could do: move from zxy to xzy to make the vote switch from z to x, move from zyx to zxy to make the vote switch from z to x, move from yxz to xyz to make the vote switch from y to x, or move from yzx to yxz to make the vote switch from y to x. See the righthand side picture above. Here are two of the cases:



In both cases, $\dot{x}yz \sim_i x\dot{z}y$ in the bottom line contradicts the Crossing Lemma. The other two cases are similar. So, x cannot be forced, and contradiction with NI. This proves the Lemma. □

Theorem 1 (Gibbard-Satterthwaite). *Any resolute voting rule that is NM and NI is a dictatorship.*

Proof. Let V be a resolute voting rule that is NM and NI. Then by NI, there has to be a profile $\mathbf{P}$ with at least one effective voter i.

Suppose $\mathbf{P}$ has another effective voter j. By the Effectiveness Lemma, i determines the vote for every $\mathbf{P}'$ with $\mathbf{P} \sim_i \mathbf{P}'$, and j determines the vote for every $\mathbf{P}''$ with $\mathbf{P} \sim_j \mathbf{P}''$. By the Effectiveness Lemma, $V(\mathbf{P}) = x$ is the favourite of both i and j in $\mathbf{P}$. By the Effectiveness Lemma, $V(\mathbf{P}') = y \neq x$ is the favourite of i in $\mathbf{P}'$, while the favourite of j in $\mathbf{P}'$ is still x. By the Effectiveness Lemma, $V(\mathbf{P}'') = z \neq x$ is the favourite of j in $\mathbf{P}''$, while the favourite of i in $\mathbf{P}''$ is still x. We may assume that $z \neq y$, for by NI there is some z different from both x and y that can be the outcome of the vote, and if j moves z to the top of her ballot, z will be the outcome of the vote by the Effectiveness Lemma. Let $\mathbf{Q}$ be the result of both i and j changing their ballots from those in $\mathbf{P}$, i to his $\mathbf{P}'$ ballot and j to her $\mathbf{P}''$ ballot. Then $\mathbf{P}' \sim_j \mathbf{Q} \sim_i \mathbf{P}''$. Suppose $V(\mathbf{Q}) \neq V(\mathbf{P}')$. Then j is effective in $\mathbf{P}'$, so by the Effectiveness Lemma, $V(\mathbf{P}')$ should equal the favourite of j in $\mathbf{P}'$, which is not the case. So $V(\mathbf{Q}) = V(\mathbf{P}')$. This means that i is effective in $\mathbf{P}''$. By the Effectiveness Lemma, $V(\mathbf{P}'')$ should equal the favourite of i in $\mathbf{P}''$, which is not the case. Contradiction.

So i is the one and only effective voter for $\mathbf{P}$. But then i must be the one and only effective voter for *any* $\mathbf{P}$, and therefore i is the dictator. □

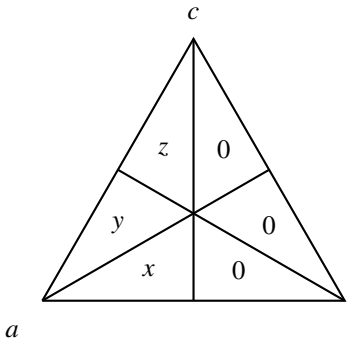
4 Some Other Properties of Resolute Voting Rules

The following theorem provides another example of use of the Saari triangle for proving simple properties of voting rules.

Theorem 2. *For any resolute voting rule V that satisfies NM it holds that V satisfies Pareto iff V satisfies NI.*

Proof. $\Rightarrow$: We will show that any resolute voting rule V that satisfies NM but not Pareto is imposed.

Suppose V satisfies NM, but not Pareto. Then (wlog) there is a profile $\mathbf{P}$ that has everywhere a above b, but $V(\mathbf{P}) = b$.



Check that type changes to the empty regions, moving clockwise, can never produce a V-value different from b. So V is imposed. (In fact, no outcome other than b is possible!)

$\Leftarrow$: Let V satisfy NM and Pareto. Suppose for a contradiction that V is imposed, i.e., for any profile $\mathbf{P}$, either $V(\mathbf{P}) = x$ or $V(\mathbf{P}) = y$. Since we assume that $|A| > 2$ there is some z different from both x and y. Consider some $\mathbf{P}$ with $V(\mathbf{P}) = x$. Move any alternative different from x, y, z down below any of these three alternatives in all ballots of $\mathbf{P}$. Call the resulting profile $\mathbf{Q}$. By Pareto, $V(\mathbf{Q}) = x$. Let $\mathbf{Q}'$ be the result of all voters changing their ballots in $\mathbf{Q}$ by moving z to the top position of their ballot. Then by Pareto, $V(\mathbf{Q}') = z$ and contradiction with the assumption that V is imposed. □

A resolute voting rule V is *monotonic* if whenever $V(\mathbf{P}) = a$ and $\mathbf{P}'$ is the result of changing each $>_i$ to $>_i'$ in such a way that for all $b \in A$, $a >_i b$ implies $a >_i' b$, then $V(\mathbf{P}') = a$.

Theorem 3. *If V is NM then V is monotonic.*

Proof. It follows immediately from the crossing lemma that any adjacent transposition that does not move a b up past a winning a will not make the vote change from a to b. □

Theorem 4. *If V is mononotic then V is NM.*

Proof. Assume V is monotonic. Suppose for a contradiction that V can be manipulated. Then there has to be a pair $\mathbf{P} \sim_i \mathbf{P}'$ such that $V(\mathbf{P}') >_i V(\mathbf{P})$. Since any individual vote change can be decomposed into adjacent transpositions, there has to be a pair $\mathbf{P} \sim_i \mathbf{P}_i$ with $V(\mathbf{P}') >_i V(\mathbf{P})$ and such that an adjacent transposition relates $>_i$ to $>_i'$. Let $V(\mathbf{P}') = a$ and $V(\mathbf{P}) = b$. Then $>_i = \alpha ab\beta$ and $>_i' = \alpha ba\beta$. By monotonicity, changing $>_i'$ to $>_i$ by means of moving a up over b should not change the vote, and contradiction with $V(\mathbf{P}) = b$. □

Theorem 5 (Muller-Satterthwaite [6]). *Any resolute voting rule V that is* monotonic *and satisfies Pareto is a dictatorship.*

Proof. Let V be a resolute voting rule that is *monotonic* and satisfies Pareto. Then by Theorem 4, V is NM. By Theorem 2, V also is NI. It follows from Theorem 1 that V is a dictatorship. □

This ends our discussion of the proof of the Gibbard-Satterthwaite theorem and related results.

5 Reflections on Manipulation

In this section we will argue that the notion of manipulation that was used for proving the Gibbard-Satterthwaite theorem is too general to serve as a useful classifier of voting rules. If we define our concept of sin in such manner that every human (including saints and law-abiding citizens) is a sinner, then we can take this as a condemnation of humanity, but we can also conclude that there may be something wrong with our definition of sin. To escape from the uniform condemnation of humanity, we could ask ourselves if some sins are perhaps worse than others.

So let us ask ourselves some questions about possible manipulations. Which of the following is worse?

champion rearrangement. My preferences are $a > b > c$, and the V-value is c. I switch my preferences to $b > a > c$ and the value becomes b.

champion bashing. My preferences are $a > b > c$, and the V-value is c. I switch my preferences to $b > c > a$ and the value becomes b.

In the first case, the only thing that happens is that a does not beat c, but I have another candidate b for beating c. So I push b, and it turns out b is better for the job.

One might think about vote manipulation from the perspective of reasoning about other minds, as follows. Taking an epistemic perspective on manipulation, we see:

- I *know* that the outcome of the voting process if I stick to ordering $a > b > c$ is c. This in information about *what the others think*.
- I *know* that the outcome if I change the order of my two most favoured candidates is b. This is information about *how others would react if I readjust*.
- Who, in his right mind, would not readjust? Not adjusting would be worse than a crime: it would be a stupidity.
- I can even explain it to a: "Sorry, in the circumstances you are not the right choice. If I insist on you, I will not be able to beat c. But I will not deny that you are way better than that abject c."

These considerations show that 'manipulation' is simply too coarse for making useful distinctions. To illustrate that it is possible to do better, we propose a couple of notions that are stronger than manipulation.

Definition 1. *The* knights *of a voter i, given profile* $\mathbf{P}$ *and resolute voting rule V, are the goods that are above* $V(\mathbf{P})$ *on the i-ballot. The* knaves *of a voter i, given profile* $\mathbf{P}$ *and resolute voting rule V, are the goods that are below* $V(\mathbf{P})$ *on the i-ballot.*

For example, if i has ballot $a > b > c > d$ in $\mathbf{P}$, and the outcome of the vote is c, then a and b are knights of i in $\mathbf{P}$, and d is a knight of i in $\mathbf{P}$.

Definition 2. *A resolute voting rule V is* demotion pervertible *(DP) if there exists an i-minimal pair of profiles* $\mathbf{P}$, $\mathbf{P}'$ *such that*

- $V(\mathbf{P}) <_i V(\mathbf{P}')$, *and*
- $\exists x : \; V(\mathbf{P}) <_i x <_i' V(\mathbf{P})$.

A resolute voting rule V is NDPe *(non-demotion-pervertible) if V is not DP.*

Note: the demotion of knight x from above $V(\mathbf{P})$ to a new position below $V(\mathbf{P})$ is the perversion.

For example, suppose i has ballot *abcd* in $\mathbf{P}$, and the outcome of the vote is c, and $\mathbf{P} \sim_i \mathbf{P}'$ where i has ballot *bcad* in $\mathbf{P}'$, and the outcome of the vote in $\mathbf{P}'$ is b. Then V is demotion pervertible, for we have that $V(\mathbf{P}) = c <_i b = V(\mathbf{P}')$, and $V(\mathbf{P}) = c <_i a <_i' c = V(\mathbf{P})$, that is to say, a was demoted from a knight to a knave position (from the perspective of $\mathbf{P}$).

Definition 3. *A resolute voting rule V is* promotion pervertible *(PP) if there exists an i-minimal pair of profiles $\mathbf{P}$, $\mathbf{P}'$ such that*

- $V(\mathbf{P}) <_i V(\mathbf{P}')$*, and*
- $\exists x : V(\mathbf{P}) <_i' x <_i V(\mathbf{P})$.

A resolute voting rule V is NPPe *(non-promotion-pervertible) if V is not PP.*

Note: the promotion of knave x from below $V(\mathbf{P})$ to a new position above $V(\mathbf{P})$ is the perversion.

For example, suppose i has ballot *abcd* in $\mathbf{P}$, and the outcome of the vote is c, and $\mathbf{P} \sim_i \mathbf{P}'$ where i has ballot *abdc* in $\mathbf{P}'$, and the outcome of the vote in $\mathbf{P}'$ is b. Then V is promotion pervertible, for we have that $V(\mathbf{P}) = c <_i b = V(\mathbf{P}')$, and $V(\mathbf{P}) = c <_i' d <_i c = V(\mathbf{P})$, that is to say, d was promoted from a knave to a knight position (from the perspective of $\mathbf{P}$).

Definition 4. *An i-minimal pair of profiles $\mathbf{P}$, $\mathbf{P}'$* invites decency towards knights *if the following hold:*

- $\exists x : V(\mathbf{P}) <_i x <_i' V(\mathbf{P})$ *implies* $V(\mathbf{P}) \geq_i V(\mathbf{P}')$,
- $\exists x : V(\mathbf{P}') <_i' x <_i V(\mathbf{P}')$ *implies* $V(\mathbf{P}) \leq_i' V(\mathbf{P}')$.

Conversely:

- $V(\mathbf{P}) <_i V(\mathbf{P}')$ *implies* $\forall x : V(\mathbf{P}) <_i x \Rightarrow V(\mathbf{P}) \leq_i' x$,
- $V(\mathbf{P}) >_i' V(\mathbf{P}')$ *implies* $\forall x : V(\mathbf{P}') <_i' x \Rightarrow V(\mathbf{P}') \leq_i x$.

"If the shift from $\geq_i$ to $\geq_i'$ is an improvement, then no knight was demoted", and similarly in the other direction.

Definition 5. *An i-minimal pair of profiles $\mathbf{P}$, $\mathbf{P}'$* invites decency towards knaves *if the following hold:*

- $\exists x : V(\mathbf{P}) <_i' x <_i V(\mathbf{P})$ *implies* $V(\mathbf{P}) \geq_i V(\mathbf{P}')$,
- $\exists x : V(\mathbf{P}') <_i x <_i' V(\mathbf{P}')$ *implies* $V(\mathbf{P}) \leq_i' V(\mathbf{P}')$.

Conversely:

- $V(\mathbf{P}) <_i V(\mathbf{P}')$ *implies* $\forall x : V(\mathbf{P}) <_i' x \Rightarrow V(\mathbf{P}) \leq_i x$,
- $V(\mathbf{P}) >_i' V(\mathbf{P}')$ *implies* $\forall x : V(\mathbf{P}') <_i x \Rightarrow V(\mathbf{P}') \leq_i' x$.

Lemma 3 (Non-Perversion Lemma: the Meaning of Perversion). *A resolute voting rule V is NDPe iff V invites decency towards knights for every i-minimal pair of profiles. A resolute voting rule V is NPPe iff V invites decency towards knaves for every i-minimal pair of profiles.*

Proof. By an easy check on the definitions. □

We can use the notion of decency-inviting pairs to work out what the decent V-value switches are for adjacent transpositions (in the case of three alternatives: crossings in the Saari triangle). It then turns out that the constraints on the crossings change. Here is the new crossing lemma for decent behaviour towards both knights and knaves:

Lemma 4 (Lemma For Decent Crossings). *Let V be NDP and NPP, and let $\mathbf{P} \sim_i \mathbf{P}'$ be such that $>_i$ and $>_i'$ are related by an adjacent transposition that exchanges x and y (with $x >_i y$). Then $V(\mathbf{P}) \neq V(\mathbf{P}')$ implies:*

- *if $V(\mathbf{P}) = x$ then $V(\mathbf{P}') <_i x$,*
- *if $V(\mathbf{P}) = y$ then $V(\mathbf{P}') <_i' y$.*

Proof. Assume V is NDP and NPP, let $\mathbf{P} \sim_i \mathbf{P}'$ be such that $>_i$ and $>_i'$ are related by an adjacent transposition that exchanges x and y (with $x >_i y$). Assume $V(\mathbf{P}) \neq V(\mathbf{P}')$.

Suppose $V(\mathbf{P}) = x$. Then $V(\mathbf{P}') \geq_i x$ and $V(\mathbf{P}) \neq V(\mathbf{P}')$ imply $V(\mathbf{P}) <_i V(\mathbf{P}')$. Moreover, $V(\mathbf{P}) = x <_i y <_i' x$. Contradiction with the given that V is NDP. Therefore $V(\mathbf{P}') <_i x$.

Suppose $V(\mathbf{P}) = y$. Then $V(\mathbf{P}') \geq_i' y$ and $V(\mathbf{P}) \neq V(\mathbf{P}')$ imply $V(\mathbf{P}) <_i V(\mathbf{P}')$. Moreover $V(\mathbf{P}) = y <_i' x <_i y$. Contradiction with the given that V is NPP. Therefore $V(\mathbf{P}') <_i' y$. □

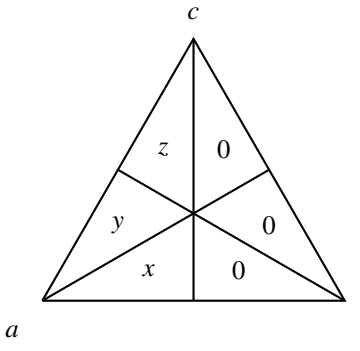
What the new crossing lemma says is that in decent $x : y$ crossing a shift in the vote from x has to be to a position that is above x on the original ballot, and a shift of the vote from y has to be to a position that is above y on the new ballot (in particular, a shift to x is forbidden).

Working this out we find what the decent V-value switches are for walking through the Saari triangle, moving in clockwise direction.

- From 1 to 2: all except (c, b).
- From 2 to 3: all except (a, c).
- From 3 to 4: all except (b, a).
- From 4 to 5: all except (b, c).
- From 5 to 6: all except (a, c).
- From 6 to 1: all except (a, b).

Armed with this, we can turn back to the manipulability proof, to see where it breaks down. It is clear, then, that the Effectiveness Lemma cannot be proved anymore with the new, much weaker version of the Crossing Lemma.

Other theorems also break down. The equivalence of the Pareto condition and non-imposition no longer holds for resolute voting rules that satisfy NPP and NDP. This can be seen by analyzing the picture again:



It remains to show that the notion of pervertibility allows for useful distinctions. For that, notice that these notions can also be applied to voting rules (functions from profiles to non-empty subsets of alternatives), as follows. A voting rule V is single winner demotion pervertibible if there is a pair of profies $\mathbf{P} \sim_i \mathbf{P}'$ with $V(\mathbf{P}) = \{x\}$ and $V(\mathbf{P}') = \{y\}$, and $x <_i y$, and moreover $>_i'$ is the result of demoting at least one knight in $>_i$. Similarly for promotion pervertibility.

The following example shows that the Borda Count rule (the set of alternatives with the highest Borda counts tie for a win, where the Borda count of an alternative a in ballot $>_i$ is given by the number of alternatives strictly below a according to $>_i$) is promotion pervertible in this sense. Let $\mathbf{P} \sim_1 \mathbf{P}_i$ be given by:

1	2	3	4
a	b	d	c
b	d	c	a
c	c	a	b
d	a	b	d

$\sim_1$

1	2	3	4
b	b	d	c
a	d	c	a
d	c	a	b
c	a	b	d

The BC scores in $\mathbf{P}$ are given by $a : 6, b : 6, c : 7, d : 5$, so the outcome of the Borda vote in $\mathbf{P}$ is $\{c\}$. The BC scores in $\mathbf{P}'$ are given by $a : 5, b : 7, c : 6, d : 6$, so the outcome of the Borda vote in $\mathbf{P}'$ is $\{b\}$. The vote change involves a promotion of the 1-knave d in $\mathbf{P}$; moreover, this promotion is necessary to get b to win. (This example is used in [11] to show that the Borda Count rule is single winner manipulable.)

The Borda Count rule is not single winner demotion pervertible, because of the following fact:

Theorem 6. *The Borda Count Rule has the following property: Although it is* possible *to pervert the Borda Count rule by knight demotion, this is never* profitable*. For any knight demoting manipulation there is a rearrangement alternative that does not demote knights and that works just as well.*

Proof. Let $\mathbf{P} \sim_i \mathbf{P}'$ be an i-minimal pair of profiles with $V(\mathbf{P}) = \{x\}$, and $V(\mathbf{P}') = \{y\}$, with $x <_i y$. To make y the winner in $\mathbf{P}'$, either the Borda count of y must have gone *up*, or the Borda count of x must have gone *down*, or both. Moving y up does not involve knight demotion, and moving x down does not involve knight demotion either. □

6 Conclusion

Don Saari analyzes Arrow's impossibility theorem using geometry, and argues that the principle of IIA (Indepence of Irrelevant Alternatives) is to blame (a very brief summary is in [8]). Saari shows how modifying IIA can turn the impossibility theorem into a useful possibility result.

The same seems possible (and necessary) for the notion of manipulability. Our analysis of proof of the Gibbard/Satterthwaite theorem highlights the severity of the constraints that NM imposes on how the value of the vote can change.

Our notions of *pervertibility* are just examples of possible ways out. Hopefully, the distinction between pervertible and non-pervertible voting rules will turn out more useful than that between manipulable and non-manipulable voting rules. Classifying the pervertible voting rules is future work.

An earlier proposal for modifying the notion of manipulability is in [2]. The criticism in that paper of the notion of manipulability is two-fold: the authors argue that manipulations can be sincere, and they argue that the non-transparancy that results from manipulability can be a boon. Let's ignore the second criticism, and focus on the first.

> We do not think that someone has revealed a preference for beer over champagne when they buy beer rather than champagne, when we know their finances will not stretch to a bottle of the bubbly. [2]

The cited paper calls a manipulation of V in $\mathbf{P}$ given by $\mathbf{P}'$ *sincere* if $\mathbf{P}'$ is the result of a subset S of the voters moving some y that they all prefer to $V(\mathbf{P})$ to the top of their ballots, while leaving the rest of the ballot unchanged. (Actually, the definition is stated in game-theoretic terms; this is my paraphrase.) Clearly, this is a special case of our proposal: y is among the knights of all voters in S, so no knave is promoted and no knight demoted in the switch from $\mathbf{P}$ to $\mathbf{P}'$. But the proposal is less general than ours, for a ballot change from $xyz\dot{w}$ to $x\dot{z}yw$ (with the dots indicating the outcome of the vote) is not sincere in the sense of [2], but it is decent in our sense.

Acknowledgement. Thanks to Krzysztof Apt, Vince Conitzer, Ulle Endriss, Floor Sietsma and Sunil Simon for enlightening discussions about the topic of this paper. Three anonymous CLIMA XII reviewers also gave useful feedback.

References

1. Arrow, K.: Social Choice and Individual Values, 2nd edn. Wiley, New York (1951)
2. Dowding, K., van Hees, M.: In praise of manipulation. British Journal of Political Science 38, 1–15 (2008), http://dx.doi.org/10.1017/S000712340800001X
3. Eckert, D., Klamler, C.: A geometric approach to paradoxes of majority voting: From Anscombe's paradox to the discursive dilemma with Saari and Nurmi. Homo Oeconomicus 26(3/4), 471–488 (2009)
4. van Eijck, J.: On Social Choice Theory. In: Discourses on Social Software, pp. 71–85. Amsterdam University Press, Amsterdam (2009), www.cwi.nl/~jve/books/pdfs/justOSCT.pdf
5. Gibbard, A.: Manipulation of voting schemes: A general result. Econometrica 41, 587–601 (1973)

6. Muller, E., Satterhwaite, M.: The equivalence of strong positive association and strategy proofness. Journal of Economic Theory 14, 412–418 (1977)
7. Perote-Peña, J., Piggins, A.: Geometry and impossibility. Economic Theory 20, 831–836 (2002)
8. Saari, D.: Arrow impossibility theorem. In: Encyclopaedia of Mathematics. Springer Online Reference Works. Springer, Heidelberg (2001), http://eom.springer.de/a/a110710.htm
9. Saari, D.G.: Basic Geometry of Voting. Springer, Heidelberg (1995)
10. Satterthwaite, M.A.: Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. Journal of Economic Theory 10, 187–217 (1975)
11. Taylor, A.D.: Social Choice and the Mathematics of Manipulation. Mathematical Association of America and Cambridge University Press (2005)

Alternating-Time Temporal Announcement Logic

Tiago de Lima

CRIL – University of Artois and CNRS
Rue Jean Souvraz, SP 18
F-62307 Lens Cedex, France

Abstract. We propose a formalism that we call Alternating-time Temporal Announcement Logic (ATAL). It can be seen as an extension of the Coalition Announcement Logic (CAL) proposed by Ågotnes et al. As well as CAL, ATAL has modal operators enabling to express sentences like 'there is an action α by group of agents G after which consequence φ is true, in spite of what the other agents do'. One of the differences here, is that such action α can also be a physical action, and not only public announcements, as in CAL. Based on the latter kind of operator, ATAL also presents operators similar to those in Alternating-time Temporal Logic, which enable to express agents abilities. For instance, ATAL has operators enabling to express sentences like 'the group of agents G is able to enforce that φ is true from the next step on until ψ becomes true'. We also provide a sound and complete axiomatization for ATAL and draw comparisons with several other logics, such as Public Announcement Logic with Assignment, Arbitrary Public Announcement Logic, Coalition Logic and Alternating-time Temporal Logic.

Keywords: Logics for coalitional ability; Epistemic Logic; Dynamic Epistemic Logic; Coalition Logic; Alternating-time Temporal Logic.

1 Introduction

Recently, several formalisms aiming at modeling multi-agent systems have been proposed. The most known examples are perhaps Sees-To-It-That logic (STIT) [5,7], Coalition Logic (CL) [20] and Alternating-time Temporal Logic (ATL) [3,17]. These formalisms allow reasoning about the abilities of the agents, i.e., about what states the agents are able to achieve. In ATL for example, one can write the formula $\langle\!\langle G \rangle\!\rangle \varphi$, which means 'the group of agents G is able to enforce an outcome satisfying φ'. However, these logics do not enable reasoning about how the group G is able to enforce such outcomes. In other words, these logics do not enable reasoning about the actions the agents actually perform in order to enforce the outcome satisfying φ.

In the literature, we can find formalisms allowing reasoning about what outcomes agents are able to achieve and about how the agents achieve such outcomes. But frequently, they do not allow reasoning about individual actions of the agents. In other words, the actions are either exogenous or always executed jointly by all the agents of the scenario. For example, in Public Announcement Logic (PAL) [21], one can write the formula $\langle \varphi \rangle \mathrm{K}_i \psi$, which means 'agent i knows ψ after the announcement of φ'. But the announcement of φ is not "enacted" by any agent of the scenario. It is either interpreted as executed by all the agents together, or as an exogenous event. To this category

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 105–121, 2011.

also belong logic ES [18] as well as some logics of the family known as Dynamic Epistemic Logics (DEL), such as the BMS framework [4], the already mentioned Public Announcement Logic (PAL) [21], and Public Announcement Logic with Assignment (PALA) [10].

Formalisms allowing reasoning about agents abilities and individual actions also exist. But their focus is on epistemic actions, i.e., actions only able to change the epistemic state of the agents. To this category belong Group Announcement Logic (GAL) and Coalition Announcement Logic (CAL) [2,1]. Both are extensions of Epistemic Logic (EL) with "enacted" public announcement operators and with group announcement operators. In GAL, one can write, e.g., the formula $\langle K_i\varphi\rangle K_j\psi$, which means 'agent j knows that ψ after the announcement of φ by agent i'. In addition, the formula $\langle G\rangle\varphi$ means 'there is an announcement by group G after which φ, where the other agents remain silent'. The group announcement operator in CAL is different. There, the formula $\langle\!\langle G\rangle\!\rangle\varphi$ means 'there is an announcement by group G after which φ is true, in spite of what the other agents announce'. In both formalisms though, the only kind of action present is public announcement. Such actions are a specific kind of communication actions enabling to change the epistemic state of the agents.

Here, a new formalism called Alternating-time Temporal Announcement Logic (ATAL) is proposed. This logic can be seen as an extension of CAL. As well as in CAL, in ATAL, formula $\langle\!\langle G\rangle\!\rangle\varphi$ is true if and only if there is an action by group G after which φ is true, in spite of what the other agents do, which can also be read as 'the group G is able to enforce that φ is true in the next step'. But it brings some improvements. First, ATAL also contains physical actions. These are actions that also change the actual state of the world (and not only the epistemic state of the agents). Second, it contains temporal operators. In ATAL, the formula $\langle\!\langle G\rangle\!\rangle^*\varphi$ means 'the group G is able to enforce that φ is true from now on' and formula $\langle\!\langle G,\psi\rangle\!\rangle\varphi$ means 'the group G is able to enforce that eventually φ will be true, while meanwhile enforcing that ψ is true'. Third, a complete axiomatization and a decidability result are achieved (those were missing for CAL). The logic is shown to be decidable when the set of available actions is finite.

The remainder of the article is organized as follows. Section 2 presents the way we address an issue related to conflicting physical actions. Section 3 presents the formalism: its syntax, semantics, axiomatization, expressivity and a decidability result. Section 4 presents two examples which show how ATAL can be used to model multi-agent systems. Section 5 discusses related work. And finally, Section 6 concludes.

2 Formalizing Conflicting Actions

Before presenting the entire formalism, we show how we intend to solve an issue related to conflicting physical actions.

For our logic ATAL, we will assume a countable set P of propositional variables and a countable set A of labels denoting actions. (Actually, there will be more than one set of actions. But this is not important for the moment.) Then, inspired by other work on reasoning about actions [8,9], we will also assume that every action $a \in A$ has an 'action description'. Such action descriptions are an alternative way to implement the successor

state axioms (which are proposed, e.g., in [22,18]). It enables us to represent the actions in a simple way, which then permits system specifications with reasonable size.

An action description is a triple $\mathrm{D}(a) = \langle \mathrm{pre}(a), \mathrm{con}^+(a), \mathrm{con}^-(a) \rangle$, where $\mathrm{pre}(a)$ is a formula describing the executability pre-condition of action a, and $\mathrm{con}^+(a)$ and $\mathrm{con}^-(a)$ are partial functions from a finite subset of P to formulae. Formula $\mathrm{con}^+(a)(p)$ is the positive condition of p (i.e., the condition to make p true). Formula $\mathrm{con}^-(a)(p)$ is the negative condition of p (i.e., the condition to make p false).

For example, consider a scenario with a light bulb that can be turned on and off using two different buttons. If button a is pressed, the light will turn on; and if button b is pressed, the light will turn off. Now, let the proposition p represent the state of the light bulb: it is true if and only if the light bulb is on. The description of the actions of pressing the buttons are:

$$\mathrm{D}(a) = \langle \top, \{(p \mapsto \varphi)\}, \{(p \mapsto \bot)\} \rangle$$
$$\mathrm{D}(b) = \langle \top, \{(p \mapsto \bot)\}, \{(p \mapsto \psi)\} \rangle$$

In this description, $\mathrm{pre}(a) = \top$ means that the action of pressing button a is always executable; $\mathrm{con}^+(a)(p) = \varphi$ means that the light should turn on if φ is true; $\mathrm{con}^-(a)(p) = \bot$ means that the light does not turn off if this button is pressed; and analogously for b. We do not bother with the contents of φ and ψ. They could, for instance, describe the state where the mechanism linking the corresponding button to the light bulb is working properly.

The formalism will be constructed in such a way that, if action a is executed, the truth value of p is set to true if $\mathrm{con}^+(a)(p)$ is true, and it is set to false if $\mathrm{con}^-(a)(p)$ is true. To avoid problems, we can also impose the restriction that both $\mathrm{con}^+(a)(p)$ and $\mathrm{con}^-(a)(p)$ cannot be true at the same time for all a and p. Then, let $\mathrm{pos}(a)(p)$ represent the truth value of p after the execution of action a, we could use:

$$\mathrm{pos}(a)(p) = \mathrm{con}^+(a)(p) \vee (p \wedge \neg \mathrm{con}^-(a)(p))$$

which correspond to the successor state axiom suggested in [22]. However, because we intend to use such descriptions to treat multi-agent scenarios, we now ask the question: 'What should be the truth value of p if one agent presses button a and another agent presses the button b, both at the same time?' Note that the two mechanisms may very well be working properly, i.e., φ and ψ may be true at the same time, which means that we cannot use the idea based on successor state axioms as above.

There are several possible answers to this question. We may take the approach where p is set to true if at least one agent decides to do so. In such approach, we say that agents have 'positive control' of the propositional variables. But we can also take the approach where p is set to false if at least one among the agents decides to do so. In such approach, we say that agents have 'negative control' of the propositional variables. And finally, we can also take the approach where agents have 'shared control' of the propositional variables. That is, the truth value of an atom is calculated taken all agents actions into account. The latter approach is the one we take here. More precisely, let $a_1 \sqcup \cdots \sqcup a_n$ denote the execution of a_1 to a_n in parallel, we define:

$$
\begin{aligned}
\mathrm{con}^{+} &= \mathrm{con}^{+}(a_1)(p) \vee \cdots \vee \mathrm{con}^{+}(a_n)(p) \\
\mathrm{con}^{-} &= \mathrm{con}^{-}(a_1)(p) \vee \cdots \vee \mathrm{con}^{-}(a_n)(p) \\
\mathrm{pos}(a_1 \sqcup \cdots \sqcup a_n)(p) &= (\mathrm{con}^{+} \wedge \neg \mathrm{con}^{-}) \vee \\
&\quad\ (p \wedge \neg \mathrm{con}^{-}) \vee \\
&\quad\ (p \wedge \mathrm{con}^{+} \wedge \mathrm{con}^{-})
\end{aligned} \tag{1}
$$

This means that the condition to make p true (noted con^{+}) is true if and only if at least one of its conditions $\mathrm{con}^{+}(a_i)(p)$ is true. Analogously, the condition to make p false (noted con^{-}) is true if and only if at least one of its conditions $\mathrm{con}^{-}(a_i)(p)$ is true. In addition, the truth value of p after the execution of action a will:

- be true if con^{+} is true and con^{-} is false,
- be false if con^{-} is true and con^{+} is false, and
- remain the same if both con^{+} and con^{-} have the same truth value.

Indeed, returning to our example of the light bulb, we have:

- in the situation where both φ and ψ are true, the light bulb will not change its state, because $\mathrm{pos}(a \sqcup b)(p)$ is equivalent to p;
- in the situation where φ is true and ψ is false, the light bulb will be on, because $\mathrm{pos}(a \sqcup b)(p)$ is equivalent to $\varphi \vee p$; and
- in the situation where φ is false and ψ is true, the light bulb will be off, because $\mathrm{pos}(a \sqcup b)(p)$ is equivalent to $p \wedge \neg\psi$.

Now, we may ask a similar question about the executability pre-condition of actions taken in parallel. But this matter seems much less controversial. We thus take the following standard approach:

$$
\mathrm{pre}(a_1 \sqcup \cdots \sqcup a_n) = \mathrm{pre}(a) \wedge \cdots \wedge \mathrm{pre}(a) \tag{2}
$$

This means that the execution of a_1 to a_n in parallel possible if and only if all the individual actions are executable.

In what follows, we assume that these approaches for pos and pre are taken. However, most of the results of this article also hold with other approaches. For instance, with the already mentioned positive and negative control.

3 The Logic

Once a suitable way of handling conflicting physical actions is found, we can build up our logic. As said before, it will be called Alternating-time Temporal Announcement Logic (or simply ATAL).

3.1 Syntax

The vocabulary of the language of ATAL contains a countable set P of propositional variables, a finite set N of labels denoting agents and, for each $i \in N$, a countable set

A_i of labels denoting the actions available for the agent i. We assume that each set A_i contains the special action ϵ, which stands for the 'no-operation action'.

We use A_N to denote the set of all joint actions available for the entire group of agents, i.e., A_N is the set of functions from N to $\bigcup_{i\in N} A_i$, each of them returning, for each $i \in N$, an action from A_i. Thus, each $\alpha \in A_N$ is a set $\{(i,a) : i \in N \text{ and } a \in A_i\}$. Let $G \subseteq N$ and $\alpha \in A_N$, we use α_G to denote α with its domain restricted to G, i.e., $\alpha_G = \{(i,a) : (i,a) \in \alpha \text{ and } i \in G\}$. (We note that $\alpha_N = \alpha$ and $\alpha_\emptyset = \emptyset$).

The language $\mathcal{L}$ of our logic is the set of formulae φ defined by the following BNF:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathrm{K}_i\varphi \mid [\alpha_G]\varphi \mid \langle\!\langle G\rangle\!\rangle\varphi \mid \langle\!\langle G\rangle\!\rangle^*\varphi \mid \langle\!\langle G,\varphi\rangle\!\rangle\varphi$$

where p ranges over P, α ranges over A_N, and G ranges over 2^N. The language fragment without operators $\langle\!\langle G,\varphi\rangle\!\rangle$ and $\langle\!\langle G\rangle\!\rangle^*$ is called the 'next-fragment of ATAL and is noted $\mathcal{L}_\mathrm{X}$. The fragment formed by $\mathcal{L}_\mathrm{X}$ without operators $[\alpha_G]$ and $\langle\!\langle G\rangle\!\rangle$ is the language of epistemic logic and is noted $\mathcal{L}_\mathrm{el}$. In what follows, the common abbreviations for $\wedge$, $\rightarrow$, $\leftrightarrow$ and $\bot$ are used. We also use the abbreviations for the duals of $[\alpha_G]$ and $[\![G]\!]$. They are defined by $\langle\alpha_G\rangle\varphi \stackrel{\text{def}}{=} \neg[\alpha_G]\neg\varphi$ and $[\![G]\!]\varphi \stackrel{\text{def}}{=} \neg\langle\!\langle G\rangle\!\rangle\neg\varphi$, respectively.

As usual, the intended meaning of formula $\mathrm{K}_i\varphi$ is 'agent i knows that φ is true'. The intended meaning of a joint action $\alpha_G = \{(i_1,a_1),\ldots,(i_{|G|},a_{|G|})\}$ is 'all the agents in $\{i_1,\ldots,i_{|G|}\}$ execute their corresponding actions in $\{a_1,\ldots,a_{|G|}\}$ simultaneously (and we do not consider what the other agents are doing at the same time)'. The intended meaning of formula $[\alpha_G]\varphi$ is 'after every possible occurrence of α_G φ is true'. The intended meaning of formula $\langle\!\langle G\rangle\!\rangle\varphi$ is 'group G is able to enforce that φ is true in the next step'. Thus, the intended meaning of its dual $[\![G]\!]\varphi$ is 'it is not the case that group G is able to enforce that $\neg\varphi$ is true in the next step', or, equivalently, 'group G is not able to avoid that φ is true in the next step'. Moreover, the intended meaning of formulae of the form $\langle\!\langle G\rangle\!\rangle^*\varphi$ is 'group G is able to enforce that φ is true from now on', while formulae of the form $\langle\!\langle G,\psi\rangle\!\rangle\varphi$ is intended to mean 'group G is able to enforce that eventually φ will be true, while meanwhile enforcing that ψ is true'.

3.2 Semantics

To interpret joint actions, we assume, for each $a \in A_i$ and each $i \in N$, an action description $\mathrm{D}(a) = \langle \mathrm{pre}(a), \mathrm{con}^+(a), \mathrm{con}^-(a)\rangle$, where $\mathrm{pre}(a) \in \mathcal{L}_\mathrm{el}$, each $\mathrm{con}^+(a)(p) \in \mathcal{L}_\mathrm{el}$, and each $\mathrm{con}^-(a)(p) \in \mathcal{L}_\mathrm{el}$.[1] In addition, the 'no-operation' action ϵ has an action description formed by $\mathrm{pre}(\epsilon) = \top$ and $\mathrm{con}^+(\epsilon) = \mathrm{con}^-(\epsilon) = \emptyset$.

Formulae in $\mathcal{L}$ are interpreted using Kripke structures, also called epistemic models (or simply models), consisting of triples $M = \langle W, R, V\rangle$, where W is a non-empty set of possible worlds; R is a function from N to $W \times W$ which, for every agent $i \in N$, returns an equivalence relation over the set of worlds; and V is a function from P to 2^W which, for every propositional variable $p \in P$, returns the set of worlds where p is true.

Dynamic operators of ATAL are interpreted using 'model updates'. The update by α_N modifies M in two ways: the worlds not satisfying its executability pre-condition

[1] Their restriction of these formulae to $\mathcal{L}_\mathrm{el}$ prevents the definition of the satisfaction relation given below to be circular.

are removed and the truth value of propositional variables is changed according to their positive and negative conditions. Formally, it is defined as follows. The update of the model $M = \langle W, R, V\rangle$ by the joint action $\alpha_N \in A_N$ is the new epistemic model $M|\alpha_N = \langle W|\alpha_N, R|\alpha_N, V|\alpha_N\rangle$ where:

$$W|\alpha_N = \{w : M, w \models \text{pre}(\alpha_N)\}$$
$$R|\alpha_N(i) = R(i) \cap (W|\alpha_N \times W|\alpha_N)$$
$$V|\alpha_N(p) = \{w : M, w \models \text{pos}(\alpha_N)(p)\} \cap W|\alpha_N$$

and where $\text{pre}(\alpha_N)$ is as defined in Equation 2 (Section 2) and $\text{pos}(\alpha_N)(p)$ is as defined in Equation 1 (Section 2).

Let an epistemic model $M = \langle W, R, V\rangle$ and a possible world $w \in W$ be given. The satisfaction relation $\models$ between pairs (M, w), which are called pointed epistemic models, and formulae from $\mathcal{L}$ is defined as usual for Boolean operators, plus:

$M, w \models \mathrm{K}_i\varphi$ iff for all $v \in W$, if $(w, v) \in R(i)$ then $M, v \models \varphi$

$M, w \models [\alpha_G]\varphi$ iff for all $\beta \in A_N$, if $M, w \models \text{pre}(\alpha_G \cup \beta_{N\setminus G})$ then $M|(\alpha_G \cup \beta_{N\setminus G}), w \models \varphi$

$M, w \models \langle\!\langle G\rangle\!\rangle\varphi$ iff there is $\alpha \in A_N$ such that $M, w \models [\alpha_G]\varphi$

$M, w \models \langle\!\langle G\rangle\!\rangle^*\varphi$ iff for all $n \in \mathbb{N}$, if $n \geq 0$ then $M, w \models \langle\!\langle G\rangle\!\rangle^n\varphi$

$M, w \models \langle\!\langle G, \psi\rangle\!\rangle\varphi$ iff there is $n \in \mathbb{N}$ s.t. $n \geq 0$ and $M, w \models \langle\!\langle G\rangle\!\rangle^n\varphi$ and for all $m \in \mathbb{N}$, if $0 \leq m < n$ then $M, w \models \langle\!\langle G\rangle\!\rangle^m(\neg\varphi \wedge \psi)$

where $\langle\!\langle G\rangle\!\rangle^n$, for $n \geq 0$, stands for a sequence of n operators $\langle\!\langle G\rangle\!\rangle$, i.e., $\langle\!\langle G\rangle\!\rangle^0\varphi \stackrel{\text{def}}{=} \varphi$ and $\langle\!\langle G\rangle\!\rangle^{n+1}\varphi \stackrel{\text{def}}{=} \langle\!\langle G\rangle\!\rangle\langle\!\langle G\rangle\!\rangle^n\varphi$.

As usual, a formula $\varphi \in \mathcal{L}$ is valid in ATAL, noted $\models \varphi$, if and only if every pointed model (M, w) satisfies φ.

Remark 1. Following the definition given above, the satisfaction relation for the operators $[\alpha_N]$ is:

$M, w \models [\alpha_N]\varphi$ iff for all $\beta \in A_N$, if $M, w \models \text{pre}(\alpha_N \cup \beta_\emptyset)$ then $M|(\alpha_N \cup \beta_\emptyset), w \models \varphi$
iff if $M, w \models \text{pre}(\alpha_N)$ then $M|\alpha_N, w \models \varphi$

because $\beta_\emptyset = \emptyset$ for all $\beta \in A_N$. We note its similarity with the semantics of operators $[\varphi]$ of PAL.

Remark 2. The satisfaction relation for operators $[\alpha_\emptyset]$ is:

$M, w \models [\alpha_\emptyset]\varphi$ iff for all $\beta \in A_N$, if $M, w \models \text{pre}(\alpha_\emptyset \cup \beta_N)$ then $M|(\alpha_\emptyset \cup \beta_N), w \models \varphi$
iff for all $\beta \in A_N$, if $M, w \models \text{pre}(\beta_N)$ then $M|\beta_N, w \models \varphi$

We note its similarity with the semantics of operator $\Box$ of APAL.

3.3 The Next-Fragment of ATAL

The next-fragment of ATAL has some interesting properties that are worth to be analyzed separately. We do so in this section.

Axiomatization. Two of the inference rules of ATAL are formulated using 'necessity forms' η. These are defined by the following BNF:

$$\eta ::= \sharp \mid \varphi \to \eta \mid \mathrm{K}_i\eta \mid [\alpha_G]\eta$$

where $\sharp$ is a special symbol (that appears only once in a necessity form), φ ranges over $\mathcal{L}$, i ranges over N, α ranges A and G ranges over 2^N.

If η is a necessity form, then $\eta(\varphi)$ is obtained from η by substituting $\sharp$ in η for formula φ.

Table 1. Axiomatization of the next-fragment of ATAL

(TAU)	All instantiations of propositional tautologies	
(KT5)	All instantiations of axioms K, T and 5 for each operator K_i	
(AA)	$[\alpha_N]p \leftrightarrow (\mathrm{pre}(\alpha_N) \to \mathrm{pos}(\alpha_N)(p))$	(action and atoms)
(AN)	$[\alpha_N]\neg\varphi \leftrightarrow (\mathrm{pre}(\alpha_N) \to \neg[\alpha_N]\varphi)$	(action and negation)
(AC)	$[\alpha_N](\varphi \wedge \psi) \leftrightarrow ([\alpha_N]\varphi \wedge [\alpha_N]\psi)$	(action and conjunction)
(AK)	$[\alpha_N]\mathrm{K}_i\varphi \leftrightarrow (\mathrm{pre}(\alpha_N) \to \mathrm{K}_i[\alpha_N]\varphi)$	(action and knowledge)
(AS)	$([\alpha_G]\varphi \wedge [\beta_H]\psi) \to [\alpha_G \cup \beta_H](\varphi \wedge \psi)$ (if $G \cap H = \emptyset$)	(action superadditivity)
(AG)	$[\alpha_G]\varphi \to \langle\!\langle G \rangle\!\rangle\varphi$	(action and group)
(RMP)	From φ and $\varphi \to \psi$ infer ψ	(modus ponens)
(RNA)	From φ infer $[\alpha_N]\varphi$	(action necessitation)
(RNK)	From φ infer $\mathrm{K}_i\varphi$	(knowledge necessit.)
(RA)	From $\eta([\alpha_G \cup \beta_H]\varphi)$ for all $\beta \in A_N$ infer $\eta([\alpha_G]\varphi)$	(deriving action)
(RG)	From $\eta(\langle\alpha_G\rangle\varphi)$ for all $\alpha \in A_N$ infer $\eta([\![G]\!]\varphi)$	(deriving group)

Table 1 displays the axiomatization of the next-fragment of ATAL. The principles axiomatizing the full language of ATAL are shown later. Principles TAU, KT5, RMP and RNK are standard. Principles AA, AN, AC and AK are similar to the reduction axioms of public announcement logic [21], and principle RNA is the necessitation rule for operators $[\alpha_N]$. These principles follow directly from the semantics of ATAL (c.f. Remark 1). Principle AS is sometimes called 'superadditivity'. It captures the intuition that, if a group G enforces φ by executing action α, and group H enforces ψ by executing action β, then, by working together, the two groups enforce outcomes satisfying both φ and ψ. Principle RA is formulated using necessity forms. It captures the intuition that, if a group G enforces φ by executing α in spite of what other agents do, in particular, in spite of what H does, then G enforces φ by executing α. And finally, Principles RA and RG capture the intuition that, if group G enforces φ by executing action α, then G is able to enforce φ.

Theorem 1 (Soundness). *All principles in Table 1 are valid in* ATAL.[2]

[2] The proofs of this and some other theorems are omitted to not exceed the page limit.

As usual, a formula $\varphi \in \mathcal{L}$ is a theorem of ATAL, noted $\vdash \varphi$, if and only if φ is an instantiation of some axiom from the axiomatization of ATAL, or it is generated by the application of some inference rule from the axiomatization of ATAL to theorems of ATAL.

In the sequel, some interesting properties of ATAL are derived. Some of them are used to prove completeness of the axiomatization. Moreover, this exercise helps to illustrate how to correctly use the non-standard inference rules RA and RG.

Proposition 1

1. *If* $\vdash \varphi$ *then* $\vdash [\alpha_G]\varphi$ *(necessitation for* $[\alpha_G]$*)*
2. $\vdash [\alpha_G]\varphi \rightarrow [\alpha_G \cup \beta_H]\varphi$ *(if* $G \cap H = \emptyset$*)* *(outcome monotonicity)*
3. $\vdash [\alpha_G](\varphi \wedge \psi) \leftrightarrow ([\alpha_G]\varphi \wedge [\alpha_G]\psi)$ *(action and conjunction for* $[\alpha_G]$*)*
4. $\vdash \mathrm{K}_i[\alpha_G]\varphi \rightarrow [\alpha_G]\mathrm{K}_i\varphi$ *(perfect recall)*
5. $\vdash \langle\!\langle G \rangle\!\rangle \top$ *(group activity)*
6. $\vdash \neg\langle\!\langle \emptyset \rangle\!\rangle \neg\varphi \rightarrow \langle\!\langle N \rangle\!\rangle \varphi$ *(joint determinism)*
7. $\vdash (\langle\!\langle G \rangle\!\rangle \varphi \wedge \langle\!\langle H \rangle\!\rangle \psi) \rightarrow \langle\!\langle G \cup H \rangle\!\rangle (\varphi \wedge \psi)$ *(if* $G \cap H = \emptyset$*)* *(group superadditivity)*
8. *If* $\vdash \varphi \rightarrow \psi$ *then* $\vdash \langle\!\langle G \rangle\!\rangle \varphi \rightarrow \langle\!\langle G \rangle\!\rangle \psi$ *(monotonicity)*

Proof. We derive each property using the axiomatization of ATAL:

1. 1. $\vdash \varphi$ (hypothesis)
   2. for all $\beta \in A_N$, $\vdash [\alpha_G \cup \beta_{N\setminus G}]\varphi$ (from 1 with RNA)
   3. $\vdash [\alpha_G]\varphi$ (from 2 with RA, because $\sharp$ is a necessity form)

2. Assume $G \cap H = \emptyset$.
   1. $\vdash [\beta_H]\top$ (from Prop. 1.1)
   2. $\vdash ([\alpha_G]\varphi \wedge [\beta_H]\top) \rightarrow [\alpha_G \cup \beta_H](\varphi \wedge \top)$ (AS)
   3. $\vdash [\alpha_G]\varphi \rightarrow [\alpha_G \cup \beta_H]\varphi$ (from 1 and 2)

3. First, we derive the implication from the right to the left:
   1. for all $\beta \in A_N$, $\vdash ([\alpha_G]\varphi \wedge [\alpha_G]\psi) \rightarrow ([\alpha_G \cup \beta_{N\setminus G}]\varphi \wedge [\alpha_G \cup \beta_{N\setminus G}]\psi)$ (from Prop. 1.2)
   2. for all $\beta \in A_N$, $\vdash ([\alpha_G]\varphi \wedge [\alpha_G]\psi) \rightarrow [\alpha_G \cup \beta_{N\setminus G}](\varphi \wedge \psi)$ (from 1 and AC)
   3. $\vdash ([\alpha_G]\varphi \wedge [\alpha_G]\psi) \rightarrow [\alpha_G](\varphi \wedge \psi)$ (from 2 with RA, because $([\alpha_G]\varphi \wedge [\alpha_G]\psi) \rightarrow \sharp$ is a necessity form)

   Now, we derive the implication from the left to the right:
   1. for all $\beta \in A_N$, $\vdash [\alpha_G](\varphi \wedge \psi) \rightarrow [\alpha_G \cup \beta_{N\setminus G}](\varphi \wedge \psi)$ (Prop. 1.2)
   2. for all $\beta \in A_N$, $\vdash [\alpha_G \cup \beta_{N\setminus G}](\varphi \wedge \psi) \rightarrow ([\alpha_G \cup \beta_{N\setminus G}]\varphi \wedge [\alpha_G \cup \beta_{N\setminus G}]\psi)$ (from AC)
   3. for all $\beta \in A_N$, $\vdash [\alpha_G \cup \beta_{N\setminus G}](\varphi \wedge \psi) \rightarrow [\alpha_G \cup \beta_{N\setminus G}]\varphi$ (from 2)
   4. for all $\beta \in A_N$, $\vdash [\alpha_G](\varphi \wedge \psi) \rightarrow [\alpha_G \cup \beta_{N\setminus G}]\varphi$ (from 1 and 3)
   5. $\vdash [\alpha_G](\varphi \wedge \psi) \rightarrow [\alpha_G]\varphi$ (from 4 with RA, because $[\alpha_G](\varphi \wedge \psi) \rightarrow \sharp$ is a necessity form)

   Analogously, we obtain $\vdash [\alpha_G](\varphi \wedge \psi) \rightarrow [\alpha_G]\psi$. From this and 5, we obtain: $\vdash [\alpha_G](\varphi \wedge \psi) \rightarrow ([\alpha_G]\varphi \wedge [\alpha_G]\psi)$.

4. 1. for all $\beta \in A_N$, $\vdash ([\alpha_G]\varphi \wedge [\beta_{N\setminus G}]\top) \rightarrow [\alpha_G \cup \beta_{N\setminus G}](\varphi \wedge \top)$ (AS)
   2. for all $\beta \in A_N$, $\vdash [\alpha_G]\varphi \rightarrow [\alpha_G \cup \beta_{N\setminus G}]\varphi$ (from 1)
   3. for all $\beta \in A_N$, $\vdash \mathrm{K}_i([\alpha_G]\varphi \rightarrow [\alpha_G \cup \beta_{N\setminus G}]\varphi)$ (from 2 with RNK)
   4. for all $\beta \in A_N$, $\vdash \mathrm{K}_i[\alpha_G]\varphi \rightarrow \mathrm{K}_i[\alpha_G \cup \beta_{N\setminus G}]\varphi$ (from 3 with KT5)
   5. for all $\beta \in A_N$, $\vdash \mathrm{K}_i[\alpha_G \cup \beta_{N\setminus G}]\varphi \rightarrow [\alpha_G \cup \beta_{N\setminus G}]\mathrm{K}_i\varphi$ (AK)
   6. for all $\beta \in A_N$, $\vdash \mathrm{K}_i[\alpha_G]\varphi \rightarrow [\alpha_G \cup \beta_{N\setminus G}]\mathrm{K}_i\varphi$ (from 4 and 5)
   7. $\vdash \mathrm{K}_i[\alpha_G]\varphi \rightarrow [\alpha_G]\mathrm{K}_i\varphi$ (from 6 with RA, because $\mathrm{K}_i[\alpha_G]\varphi \rightarrow \sharp$ is a necessity form)

5. 1. $\vdash [\alpha_G]\top$ (from Prop. 1.1)
   2. $\vdash \langle\!\langle G \rangle\!\rangle \top$ (from 1 with RA and RMP)

6. We show its contrapositive:
   1. for all $\alpha \in A_N$ $\vdash [\![N]\!]\varphi \rightarrow \langle \alpha_N \rangle \varphi$ (from AG)
   2. for all $\alpha \in A_N$ $\vdash [\![N]\!]\varphi \rightarrow [\alpha_N]\varphi$ (from 1 with AN)
   3. $\vdash [\![N]\!]\varphi \rightarrow [\alpha_\emptyset]\varphi$ (from 2 with RA)
   4. $\vdash [\![N]\!]\varphi \rightarrow \langle\!\langle \emptyset \rangle\!\rangle \varphi$ (from 2 with AG)

7. Again, we show its contrapositive. Let $G \cap H = \emptyset$:
   1. for all $\alpha \in A_N$ $\vdash [\![G \cup H]\!](\varphi \vee \psi) \rightarrow \langle \alpha_{G\cup H} \rangle (\varphi \vee \psi)$ (from AG)
   2. for all $\alpha \in A_N$ $\vdash [\![G \cup H]\!](\varphi \vee \psi) \rightarrow (\langle \alpha_G \rangle \varphi \vee \langle \alpha_H \rangle \psi)$ (from 1 with AS)
   3. $\vdash [\![G \cup H]\!](\varphi \vee \psi) \rightarrow ([\![G]\!]\varphi \vee [\![H]\!]\psi)$ (from 2 with RG)

8. 1. $\vdash \varphi \rightarrow \psi$ (hypothesis)
   2. for all $\alpha \in A_N$ $\vdash [\alpha_G](\varphi \rightarrow \psi)$ (from 1 with Prop. 1.1)
   3. for all $\alpha \in A_N$ $\vdash [\alpha_G]\varphi \rightarrow [\alpha_G]\psi$ (from 2 with Prop. 1.3)
   4. for all $\alpha \in A_N$ $\vdash \langle \alpha_G \rangle \neg\psi \rightarrow \langle \alpha_G \rangle \neg\varphi$ (from 3)
   5. $\vdash \langle \alpha_G \rangle \neg\psi \rightarrow [\![G]\!]\neg\varphi$ (from 3 with RG)
   6. $\vdash \langle\!\langle G \rangle\!\rangle \varphi \rightarrow [\alpha_G]\psi$ (from 5)
   7. $\vdash \langle\!\langle G \rangle\!\rangle \varphi \rightarrow \langle\!\langle G \rangle\!\rangle \psi$ (from 6 with AG)

□

Propositions 1.1 and 1.3 together show that operators $[\alpha_G]$ are normal modal operators. Proposition 1.4 corresponds to what is called 'perfect recall' in [11]. It captures the intuition that the knowledge of the agents either increases or remains the same after the execution of an action. This means that agents never loose information, i.e., once an agent knows something, this agent will never forget it. Together with the fact that each $R(i)$ is an equivalence relation, Proposition 1.4 implies that action occurrences are perceived by all agents, which implies that the agents also perceive the passage of time.

Propositions 1.5–8 are the principles satisfied by operators $\langle G \rangle$ of Coalition Logic (CL). We conjecture that ATAL is at least as expressive as CL. But we leave the full proof of this claim for future work.

Theorem 2 (Completeness). *Every valid formula $\varphi \in \mathcal{L}_X$ is a theorem of* ATAL.

Expressivity. The next-fragment of ATAL is at least as expressive as Public Announcement Logic (PAL). The latter logic is an extension of multi-agent epistemic logic with public announcements. This logic contains formulae of the form $[\varphi]\psi$, which mean 'after the public announce of φ, ψ is true'.

We show that ATAL is at least as expressive as PAL by encoding PAL into the version of ATAL satisfying the following condition:

(a) For some agent $i \in N$ and for each $\varphi \in \mathcal{L}_{\text{el}}$, there is an action $a^{\varphi} \in A$ such that $\text{pre}(a^{\varphi}) = \varphi$ and $\text{con}^{+}(a^{\varphi}) = \text{con}^{-}(a^{\varphi}) = \emptyset$.

Now, it follows that, for each $\varphi \in \mathcal{L}_{\text{el}}$, there is a joint action $\alpha^{\varphi} \in A_N$ such that $\alpha^{\varphi}(i) = a^{\varphi}$ and $\alpha^{\varphi}(i') = \epsilon$ for all $i' \neq i$. The idea is that agent i is the only one who acts in the system. Using this observation, we can define the translation from PAL to ATAL recursively, as follows:

$$\begin{aligned} \text{tr}(\varphi) &= \varphi \qquad\qquad (\text{if } \varphi \in \mathcal{L}_{\text{el}}) \\ \text{tr}([\varphi]\psi) &= [\alpha_N^{\varphi}]\text{tr}(\psi) \end{aligned}$$

It is routine to show that φ is valid in PAL if and only if $\text{tr}(\varphi)$ is valid in ATAL. The formal proof uses Remark 1.

The next-fragment of ATAL is also at least as expressive as Public Announcement Logic with Assignment (PALA). The latter logic is an extension of PAL with public assignments. This logic contains formulae of the form $[p{:=}\varphi]\psi$, which mean 'after the assignment of the truth value of φ to p, ψ is true'.

The translation tr can be extended to a translation from PALA to ATAL, as follows. We take the version of ATAL where, in addition to (a) above, we have:

(b) For each assignment σ of PALA there is an action $a^{\sigma} \in A$ such that $\text{pre}(a^{\sigma}) = \top$; and for each $p \in \text{dom}(\sigma)$ we have $\text{con}^{+}(a^{\sigma})(p) = \sigma(p)$ and $\text{con}^{-}(a^{\sigma})(p) = \neg\sigma(p)$.

Similarly as before, it follows that, for each assignment σ, there is a joint action $\alpha^{\sigma} \in A_N$ such that $\alpha^{\sigma}(i) = a^{\sigma}$ and $\alpha^{\sigma}(i') = \epsilon$ for all $i' \neq i$. Then, the translation tr from PALA to ATAL is the one defined above plus:

$$\text{tr}([\sigma]\varphi) = [\alpha_N^{\sigma}]\text{tr}(\varphi)$$

Again, it is routine to show that φ is valid in PALA if and only if it is valid in ATAL. Here, we use again Remark 1 with the addition that, since $\text{pre}(a_N^{\sigma}) = \top$, the semantics of operators $[\alpha_N^{\sigma}]$ is equivalent to the semantics of operators $[\sigma]$ of PALA.

The next-fragment of ATAL is also at least as expressive as Arbitrary Public Announcement Logic (APAL), which is an extension of PAL with the arbitrary announcement operator. This logic contains formulae of the form $\Box\psi$, which mean 'after every public announcement, ψ is true'.

The translation tr from PAL to ATAL is extended by the addition of:

$$\text{tr}(\Box\psi) = [\alpha_{\emptyset}]\text{tr}(\psi)$$

By using in addition Remark 2, it is easy to see that φ is valid in APAL if and only if it is valid in ATAL.

Finally, it is easy to see that our operator $\langle\!\langle G \rangle\!\rangle$ has the same semantics as the coalition announcement operator in Coalition Announcement Logic (CAL). The only difference is that in CAL, the set of available announcements is restricted to a specific kind. In ATAL, every formula $\varphi \in \mathcal{L}_{\text{el}}$ can be announced.

(Un)decidability. Validity checking in ATAL is not decidable. It follows immediately from the non-decidability of validity checking in APAL [12], since ATAL is at least as expressive as APAL. However, when the set A_i of available actions for each agent i is finite, so is the set A_N of available joint actions. In this case, the infinitary rules RA and RG can be replaced by the following two axioms:

(RA') $$\bigwedge_{\beta \in A_N} [\alpha_G \cup \beta_{N \setminus G}]\varphi \rightarrow [\alpha_G]\varphi$$

(RG') $$\bigwedge_{\alpha \in A_N} \langle \alpha_G \rangle \varphi \rightarrow [\![G]\!]\varphi$$

Axioms RG' and AG together imply the following reduction axiom:

$$\langle\!\langle G \rangle\!\rangle \varphi \leftrightarrow \bigvee_{\alpha \in A_N} [\alpha_G]\varphi$$

This means that operators $\langle\!\langle G \rangle\!\rangle$ can be eliminated from formulae by successive applications of this equivalence (and the rule of substitution of equivalences RSE).

Similarly, Axiom RA' and Proposition 1.2 together imply the following reduction axiom:

$$[\alpha_G]\varphi \leftrightarrow \bigwedge_{\beta \in A_N} [\alpha_G \cup \beta_{N \setminus G}]\varphi$$

In this case, successive applications of this equivalence (and rule RSE) replace operators $[\alpha_G]$ for operators $[\alpha'_N]$, i.e., formulae containing actions executed by a group G can be replaced by formulae containing actions executed by the entire set of agents N. And finally, using Axioms AA, AN, AC and AK, operators $[\alpha'_N]$ can be eliminated. The result is a formula in $\mathcal{L}_{\text{el}}$.

All this means that the next-fragment of ATAL with a finite set of actions is reducible to epistemic logic. And since validity checking in the latter logic is decidable, so is validity checking in such fragment.

3.4 Adding Always and Until

Now, we analyze some properties of the entire logic ATAL.

Axiomatization. Table 2 displays the axiomatization of ATAL. The principles therein are standard for logics containing operators always and until. For instance, they are analogous to the principles present in the axiomatization of ATL, given in [13]. They are proved to be sound in Theorem 3 below.

Table 2. Axiomatization of ATAL with a finite number of actions

	All principles in Table 1 (of Page 111)	
(FPA)	$\langle\!\langle G\rangle\!\rangle^* \varphi \rightarrow (\varphi \wedge \langle\!\langle G\rangle\!\rangle\langle\!\langle G\rangle\!\rangle^* \varphi)$	(fixed-point for always)
(FPU)	$\langle\!\langle G, \psi\rangle\!\rangle \varphi \leftrightarrow (\varphi \vee (\psi \wedge \langle\!\langle G\rangle\!\rangle\langle\!\langle G, \psi\rangle\!\rangle \varphi))$	(fixed-point for until)
(RIA)	From $\chi \rightarrow (\varphi \wedge \langle\!\langle G\rangle\!\rangle \chi)$ infer $\chi \rightarrow \langle\!\langle G\rangle\!\rangle^* \varphi$	(induction for always)
(RIU)	From $(\varphi \vee (\psi \wedge \langle\!\langle G\rangle\!\rangle \chi)) \rightarrow \chi$ infer $\langle\!\langle G, \psi\rangle\!\rangle \varphi \rightarrow \chi$	(induction for until)

Theorem 3 (Soundness (cont.)). *All principles in Table 2 are valid in* ATAL.

However, to prove completeness, we need to add the assumption that all sets A_i are finite! This is so because the technique used (similar to the one used in [14] for the common knowledge operator) requires a finite canonical model. The complete axiomatization for ATAL with infinite sets A_i is left as an open question.

Theorem 4 (Completeness (cont.)). *Every formula $\varphi \in \mathcal{L}$ which is valid in* ATAL *with a finite number of actions is also a theorem of* ATAL.

Expressivity. We conjecture that the version of ATAL with a finite number of actions is, nonetheless, at least as expressive as Vanilla ATL, i.e., the fragment of ATL where a group operator is always followed by a temporal operator, and a temporal operator is always immediately preceded by a group operator.[3] The translation tr from Vanilla ATL to ATAL would be the trivial one for Boolean formulae, plus:

$$\begin{aligned} \mathrm{tr}(\langle\!\langle G\rangle\!\rangle \bigcirc \varphi) &= \langle\!\langle G\rangle\!\rangle \varphi \\ \mathrm{tr}(\langle\!\langle G\rangle\!\rangle \Box \varphi) &= \langle\!\langle G\rangle\!\rangle^* \varphi \\ \mathrm{tr}(\langle\!\langle G\rangle\!\rangle (\varphi U \psi)) &= \langle\!\langle G, \varphi\rangle\!\rangle \psi \end{aligned}$$

The translation of Vanilla ATL models would also be simple. Because the former does not have epistemic modalities, the actual possible world in the Vanilla ATL model corresponds to an epistemic state where all agents have complete information, i.e., an epistemic model with only one possible world and reflexive arrows for all agents. In addition, each choice in the Vanilla ATL model corresponds to a different possible update of this epistemic model. Also note that there can only be physical actions in Vanilla ATL.

Decidability. ATAL with a finite number of actions is decidable. This can be shown as follows.

Theorem 5 (Finite Model Theorem). *Every satisfiable formula $\varphi \in \mathcal{L}$ is satisfiable in a finite model.*

The proof is based on the fact that the filtrated canonical model for ATAL is constructed using a finite set of formulae $\mathrm{cl}^+(\varphi)$. This enables us to show the following theorem.

[3] For instance, let $\bigcirc$ mean "next". In Vanilla ATL, $\langle\!\langle G\rangle\!\rangle \bigcirc p$ is a formula, but $\langle\!\langle G\rangle\!\rangle (p \wedge \bigcirc p)$ is not, nor is $\bigcirc p$ or $\langle\!\langle G\rangle\!\rangle p$. For more details, please, consult [3].

Theorem 6 (Decidability). *Validity checking in* ATAL *with a finite number of actions is decidable.*

Proof. Recall from the discussion in Section 3.3 that a finite set of actions enables us to provide a finitary axiomatization for ATAL. Then, there is a finite proof for each valid formula. Because the set of such proofs is enumerable, if φ is valid, one can find its proof in such enumeration. On the other hand, there is a finite model satisfying every satisfiable formula. Because the set of such models is enumerable, if φ is not valid, one can find the model satisfying $\neg\varphi$ in such enumeration. Altogether means that there is an algorithm that decides whether φ is valid or not. □

4 Applications

We show in this section two examples of scenarios which can be modeled using ATAL.

Light bulb and light switch. In ATAL, one can reason about collaborative agency. To see it, we consider a scenario containing two agents: Alice (agent i) and Betty (agent j). They live in a strange house: its interior is illuminated by a light bulb, but the corresponding switch is located outside the house. In our scenario, Alice is inside the house and Betty is outside it, close to the switch. Thus, Alice can see whether the light bulb is on (noted p) or off (noted $\neg p$) and tell (or rater shout) it to Betty, but she cannot toggle the switch. Betty, on the other hand, can toggle the switch (action *tog*) but she cannot see whether the light is on or off. If she toggles the switch with the light on, it will turn off, and if she toggles it with the light off, it will turn on. So, if Alice and Betty want to reach a state where the light is on and they know it, i.e., if they want to reach a state satisfying $\mathrm{K}_i p \wedge \mathrm{K}_j p$, they must put their efforts together.

To formalize this in ATAL, let the set of agents be $N = \{i, j\}$ and the actions be described with:

$$\begin{aligned}
\mathrm{D}(tog) &= \langle \top, \{(p \mapsto \neg p)\}, \{(p \mapsto p)\} \rangle \\
\mathrm{D}(on) &= \langle p, \emptyset, \emptyset \rangle \\
\mathrm{D}(off) &= \langle \neg p, \emptyset, \emptyset \rangle
\end{aligned}$$

Action *tog* is available only for Betty while actions *on* and *off* are available only for Alice. Note that the latter two actions work as the public announcements that the light bulb is on and off, respectively. Thus, we stipulate $A_j = \{\epsilon, tog\}$ and $A_i = \{\epsilon, on, off\}$.

The set of available joint actions A_N is formed by all combinations of these actions. For the considerations below, we will use the following ones:

$$\begin{aligned}
\alpha_{\{i,j\}} &= \{(i \mapsto on), (j \mapsto \epsilon)\} \\
\alpha'_{\{i,j\}} &= \{(i \mapsto off), (j \mapsto \epsilon)\} \\
\beta_{\{i,j\}} &= \{(i \mapsto \epsilon), (j \mapsto tog)\} \\
\beta'_{\{i,j\}} &= \{(i \mapsto \epsilon), (j \mapsto \epsilon)\}
\end{aligned}$$

It is easy to check that every pointed model (M, w) satisfies $p \to [\alpha_{\{i,j\}}][\beta'_{\{i,j\}}](\mathrm{K}_i p \wedge \mathrm{K}_j p)$, which means that, if the light is on, then both agents will know that the light will be on after Alice telling that it is on and Betty not toggling the switch. Therefore, we have that every pointed model also satisfies $p \to \langle\!\langle \{i, j\} \rangle\!\rangle \langle\!\langle \{i, j\} \rangle\!\rangle (\mathrm{K}_i p \wedge \mathrm{K}_j p)$, which means that, if the light is on, then after two steps Alice and Betty are able to enforce an outcome where both of them know that the light is on. Analogously, it is easy to check that every pointed model also satisfies $\neg p \to [\alpha'_{\{i,j\}}][\beta_{\{i,j\}}](\mathrm{K}_i p \wedge \mathrm{K}_j p)$ which implies that we also have $\neg p \to \langle\!\langle \{i, j\} \rangle\!\rangle \langle\!\langle \{i, j\} \rangle\!\rangle (\mathrm{K}_i p \wedge \mathrm{K}_j p)$, Altogether means that every pointed model satisfies: $\langle\!\langle \{i, j\} \rangle\!\rangle \langle\!\langle \{i, j\} \rangle\!\rangle (\mathrm{K}_i p \wedge \mathrm{K}_j p)$, And finally, it implies that every pointed model satisfies $\langle\!\langle \{i, j\}, \top \rangle\!\rangle (\mathrm{K}_i p \wedge \mathrm{K}_j p)$, which in words means that whatever the initial situation is, Alice and Betty are able to enforce that eventually both know that the light is on.

Tic tac toe. To illustrate that ATAL can also be used to model competitive agency, such as in game-like scenarios, we consider a formalization of the game tic tac toe. We use capital letters to name each cell in the grid from the left to the right and from the top to the bottom (i.e., A names the rightmost top cell, B names the middle top cell, ..., and I names the leftmost bottom cell). Then, we assume some propositional variables describing the situation of the game, e.g., p_{XA} means 'there is a X in cell A', and two propositional variables expressing which player has the right to play: q_X means 'it is X's turn to play' and q_O means 'it is O's turn to play'. Finally, we assume some action labels describing the possible plays, e.g., a_{OB} means 'plays O in cell B'. Their description can be given as follows. For all $x \in \{X, O\}$ and $y \in \{A, \ldots, I\}$:

$$\begin{aligned} \mathrm{pre}(a_{xy}) &= q_x \wedge \neg p_{Xy} \wedge \neg p_{Oy} \\ \mathrm{con}^+(a_{xy}) &= \{(p_{xy} \mapsto \top), (q_{\overline{x}} \mapsto \top)\} \\ \mathrm{con}^-(a_{xy}) &= \{(q_x \mapsto \top)\} \end{aligned}$$

where $\overline{x}$ means 'the opposite player', i.e., $\overline{X} = O$ and $\overline{O} = X$. For instance, it follows from this description that player x can play x in cell y if and only if it is x's turn and there is no X nor O in the cell y. The actions a_{Xy} are only available for player X, while actions a_{Oy} are only available for player O. Finally, let some available joint actions be:

$$\begin{aligned} \alpha_{\{X,O\}} &= \{(X \mapsto a_{XI}), (O \mapsto \epsilon)\} \\ \alpha'_{\{X,O\}} &= \{(X \mapsto a_{XD}), (O \mapsto \epsilon)\} \\ \beta_{\{X,O\}} &= \{(X \mapsto \epsilon), (O \mapsto a_{OI})\} \end{aligned}$$

Now, let us suppose an already started match which looks like the following picture, where it is X's turn to play:

X		O
	X	O
O		

Assume a pointed model (M, w) satisfying this situation, i.e., assume:

$$\begin{aligned} M, w \models\ & q_X \wedge p_{XA} \wedge \neg p_{XB} \wedge \neg p_{XC} \wedge \\ & \neg p_{XD} \wedge p_{XE} \wedge \neg p_{XF} \wedge \\ & \neg p_{XG} \wedge \neg p_{XH} \wedge \neg p_{XI} \wedge \\ & \neg p_{OA} \wedge \wedge p_{OB} \wedge p_{OC} \wedge \\ & \neg p_{OD} \wedge p_{OE} \wedge p_{OF} \wedge \\ & p_{OG} \wedge \neg p_{OH} \wedge \neg p_{OI} \end{aligned}$$

It is easy to check that (M, w) also satisfies $[\alpha_{\{X\}}]p_{XI}$, which means that p_{XB} becomes true after such action and, therefore, player X wins. Note that such formula implies $\langle\!\langle\{X\}\rangle\!\rangle p_{XI}$, which means that player X can win in one step. But we also have that this model satisfyes $[\alpha'_{\{X\}}][\beta_{\{O\}}]p_{OI}$, which means that after some other play by player X, player O can win. Note that this implies $\langle\!\langle X\rangle\!\rangle\langle\!\langle O\rangle\!\rangle p_{OI}$, which means that X can put O in a position where O can win the game.

5 Related Work

Apart from the formalisms mentioned in Sections 3.3 and 3.4, there are some other containing group actions and group modalities in their languages that are worth to be mentioned here.

The first of those formalisms is the Group Announcement Logic (GAL) [2,1]. Its differences from ATAL are also mentioned in the introduction: actions in GAL are only public announcements and its operator $\langle G\rangle$ has a different meaning. It is not clear to us whether ATAL is as expressive as GAL or vice-versa.

There is also the Coalition Action Logic [6], the Alternating-time Temporal Logic with Explicit Strategies [23], the Dynamic Logic of Agency (DDL) [16] and the Coalition Epistemic Dynamic Logic (CEDL) [19]. The first two differ from ATAL in several aspects. The most important of them is perhaps the fact that they do not model the knowledge of the agents and, thus, also does not have epistemic actions. The third and fourth ones model the knowledge of agents and their languages look very similar to that of ATAL. But, their semantics is completely defined in terms of Kripke structures, instead of model updates. This difference is reflected on their axiomatization. Both, DDL and CEDL do not validate reduction axioms as ATAL does (i.e., Axioms AA, AN, AC and AK). One can argue that those formalisms are able to model "more actions", instead of only public announcements and physical actions. To this we can respond by saying that, first, these two kinds of actions are very useful in practice. The examples in Section 4 give a strong support for this argument. Second, because of its action descriptions, the formalization of a dynamic scenarios in ATAL is generally more succinct than in DDL and CEDL. This is the case because one must deal with the so-called frame axioms in DDL and CEDL. Compare, for example, the formalizations in Section 4, and the formalization in Section 4 of [19].

6 Conclusion

In this work, we propose the Alternating-time Temporal Announcement Logic (ATAL), wherein one may write formulae of the form $\langle\!\langle G\rangle\!\rangle\varphi$, which mean 'there is an action

by group G after which φ is true, in spite of what the other agents do' (as well as always and until versions of it). Differently from the previously existent Coalition Announcement Logic, ATAL also contains physical actions and is equipped with a sound and complete axiomatization. We show that ATAL subsumes several logics in Dynamic Epistemic Logics family. ATAL is also shown to be useful in modeling multi-agent systems through examples of collaborative and competitive scenarios.

Possible future work include a couple of questions and improvements. First of all, we intend to address the issue on non-finite sets A_i. Second, we intend to establish whether ATAL is at least as expressive as CL, ATL and also Group Announcement Logic (GAL).

For the improvements part, we expect to be able to enrich ATAL language with complex actions, in the same spirt as in Dynamic Logic [15]. This would probably enable reasoning about conditional plans and repetition, thus, making ATAL also a useful formalism for multi-agent planning. Another possible improvement would be the incorporation of other actions, such as the private announcements of the BMS framework [4].

References

1. Ågotnes, T., Balbiani, P., van Ditmarsch, H., Seban, P.: Group announcement logic. Journal of Applied Logic 8(1), 62–81 (2010)
2. Ågotnes, T., van Ditmarsch, H.: Coalitions and announcements. In: Padgham, et al. (eds.) Proc. of the AAMAS 2008. IFAAMAS, pp. 673–680 (2008)
3. Alur, R., Henzinger, T., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM 5(49), 672–713 (2002)
4. Baltag, A., Moss, L.: Logics for epistemic programs. Synthese 139, 165–224 (2004)
5. Belnap, N.D., Perloff, M., Xu, M.: Facing the future: agents and choices in our indeterminist world. Oxford University Press, Oxford (2001)
6. Borgo, S.: Coalitions in action logic. In: Veloso, M. (ed.) Proc. of IJCAI 2007, pp. 1822–1827 (2007)
7. Broersen, J.: A complete STIT logic for knowledge and action, and some of its applications. In: Baldoni, M., Son, T.C., van Riemsdijk, M.B., Winikoff, M. (eds.) DALT 2008. LNCS (LNAI), vol. 5397, pp. 47–59. Springer, Heidelberg (2009)
8. Demolombe, R., Herzig, A., Varzinczak, I.: Regression in modal logic. Journal of Applied Non-classical Logics 13(2), 165–168 (2003)
9. van Ditmarsch, H., Herzig, A., de Lima, T.: Optimal regression for reasoning about knowledge and actions. In: Proc. of AAAI, pp. 1070–1075. AAAI Press, Menlo Park (2007)
10. van Ditmasch, H., van der Hoek, W., Kooi, B.: Dynamic epistemic logic with assignment. In: Dignum, F., et al. (eds.) Proc. of AAMAS 2005, pp. 141–148 (2005)
11. Fagin, R., Halpern, J., Vardi, Y.M.M.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
12. French, T., van Ditmarsch, H.: Undecidability for arbitrary public announcement logic. In: Areces, C., Goldblatt, R. (eds.) Proc. of AiML 2008, pp. 23–42. College Publications, London (2008)
13. Goranko, V., van Drimmelen, G.: Complete axiomatization and decidability of alternating-time temporal logic. Theoretical Computer Science 353, 93–117 (2006)
14. Halpern, J.Y., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artificial Intelligence 54, 311–379 (1992)
15. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)

16. Herzig, A., Lorini, E.: A dynamic logic of agency I: STIT, abilities and powers. Journal of Logic, Language and Information 19, 89–121 (2009)
17. van der Hoek, W., Wooldridge, M.: Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. Studia Logica 75, 125–157 (2003)
18. Lakemeyer, G., Levesque, H.: Semantics for a useful fragment of the situation calculus. In: Proc. of IJCAI 2005, pp. 490–496. Professional Book Center (2005)
19. de Lima, T., Royakkers, L., Dignum, F.: A logic for reasoning about responsibility. Logic Journal of the IGPL 18(1), 99–117 (2010)
20. Pauly, M.: A modal logic for coalitional power in games. Journal of Logic and Computation 12(1), 149–166 (2002)
21. Plaza, J.: Logics of public communication. In: Proc. of ISMIS 1989 (1989)
22. Reiter, R.: The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In: Lifschitz, V. (ed.) Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, pp. 359–380. Academic Press, New york (1991)
23. Walther, D., van der Hoek, W., Wooldridge, M.: Alternating-time temporal logic with explicit strategies. In: Samet, D. (ed.) Proc. of TARK XI, pp. 269–278. Presses Universitaires de Louvain (2007)

Synthesizing Strategies for Homogenous Multi-Agent Systems with Incomplete Information

Jan Calta[1] and Dmitry Shkatov[2]

[1] Humboldt University, Berlin, Germany
calta@informatik.hu-berlin.de
[2] University of the Witwatersrand, Johannesburg, South Africa
dmitry@cs.wits.ac.za

Abstract. We present an algorithm for synthesizing strategies for multi-agent systems composed of homogeneous agents possessing incomplete information about the system as a whole. The algorithm finds all maximal strategies for such agents that enforce a certain property of the system. In contrast to other algorithms known from the literature, our algorithm can be used for automated program synthesis for systems in which agents are required to be homogeneous (i.e., every agent has to follow the same strategy), which is a more restrictive setting.

1 Introduction

Multi-agent systems have, over the last decade, emerged as an active research area on the borderline between game theory, logic, computer science, and artificial intelligence. The two most common computational tasks associated with multi-agent systems are checking if the system conforms to a given specification (technically, this is a model-checking problem for multi-agent systems) and synthesizing strategies for agents that enforce a certain property (technically, this is a synthesis problem). In the present paper, we are concerned with the latter problem.

In the case of systems where agents are heterogeneous—that is, are not required to behave in the same way—every strategy for a group of agents is a set of strategies for individual agents (an agent's strategy is a set of "rules" prescribing the agent how to act given particular circumstances) so that, when agents follow their respective strategies, they bring about a desired outcome. In the case, considered here, of systems where agents are homogeneous, a strategy for a group of agents is a bunch of identical individual strategies since all agents are required to follow the same rules; therefore, a single individual strategy automatically gives us a strategy for the group.

Another distinction commonly made when considering multi-agent systems is the one between those in which agents have complete information about the current state of the whole system and those in which agents have only partial—or, incomplete—information about the current state of the system. In the present

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 122–138, 2011.

paper, we consider the case of systems made up of agents possessing incomplete information.

As alluded to above, in this paper, we present an algorithm for automatically synthesizing strategies for multi-agent systems of homogeneous agents with incomplete information. In practice, such a synthesis allows us to automatically generate an algorithm for a system in which all software components must execute the same program and, moreover, the components have no access to the complete information about the system.

While the synthesis problem for systems with imperfect information is well studied (see [3] for an overview), the traditional approaches are automata-theoretic and we are not aware of any framework for synthesizing strategies specifically for homogenous systems.

2 Formal Model

In this section, we introduce the formal framework for the synthesis of the strategies; namely, structures used to model multi-agent systems and the language for expressing the desired system properties.

2.1 Modular Models

We use modular models (inspired by modular interpreted systems from [6]) to formally model homogeneous multi-agent systems with incomplete information and a fixed topology. More precisely, in such systems,

- all agents have the same set of local states and the same set of available actions;
- each agent has (complete) information about its local state and a limited information about the states of its "neighbors"; thus, an agent may not be able to distinguish between separate system states (a system state is a tuple of local states, one for each agent);
- the neighborhood relation does not change during the runtime (the connections between the agents are fixed throughout the computation).

An example of such system is a multi-core processor, where agents are the cores, agents' actions are processor instructions, the local states of the cores are given by the content of their registers, and the topology of the system is a grid.

The above assumptions allow us to model an agent as a *module*. Since our agents are homogenous, modules for all agents are identical. Thus, the entire system can be represented by a single agent module and the topology of the system. The use of modular models allows us to possibly save a considerable amount of storage space in comparison to the traditional model — a transition system built upon the global statespace of the entire system (such models are referred to in the literature as "concurrent epistemic game structures").

Definition 1. *A modular model* $\mathfrak{M}$ *is a tuple*

$$\mathfrak{M} = \langle Agt, Act, St, \Pi, \pi, neig, k, \Sigma, man, tran \rangle, \text{ where}$$

- $Agt = \{1, \ldots, n\}$ *is a set of agents;*
- *Act is a finite nonempty set of actions; arbitrary actions will be denoted by lower-case Greek letters from the beginning of the alphabet, such as* $\alpha, \beta, \ldots$*;*
- *St is a finite nonempty set of agent's states (the set of system states is then* St^n*); we denote the members of St using lower-case Latin letters such as q and members of* St^n *using upper-case Latin letters such as Q;*
- $\Pi = \Pi_1 \cup \cdots \cup \Pi_n$ *is the union of sets of atomic propositions, one for each agent,*
- $\pi : \Pi \to \mathcal{P}(St)$ *is a valuation function,*
- k *is the maximal number of neighbors an agent can have,*
- $neig : Agt \times \{1, \ldots, k\} \to Agt \cup \{\sharp\}$ *is a neighborhood function (thus, if* $a \in Agt$ *and* $1 \leq i \leq k$*, then* $neig(a, i)$ *is the* i^{th} *neighbor of a;* $neig(a, i) = \sharp$ *means that a does not have the* i^{th} *neighbor),*
- Σ *is a finite nonempty set of manifestation symbols,*
- $man : St \to \Sigma$ *is a manifestation function; for technical reasons,* $man(Q[\sharp]) = \lambda$[1] *where* $\lambda \in \Sigma$ *is a manifestation of an "absent" neighbor;*
- $tran : St \times \Sigma^k \times Act \to St$ *is a transition function.*

In our example of the multi-core processor (Fig. 1), there are four cores with identifiers 1 through 4 ($Agt = \{1, \ldots, 4\}$). Each of them uses two Boolean registers, `l` ("low") and `h` ("high"). The content of each register is represented by one atomic proposition ($\Pi_a = \{\mathtt{l}_a, \mathtt{h}_a\}$ for every $a \in Agt$). Thus, each core has four possible states: $St = \{q_0, q_1, q_2, q_3\}$ and $\pi(\mathtt{l}_a) = \{q_1, q_3\}, \pi(\mathtt{h}_a) = \{q_2, q_3\}$ for every $a \in Agt$ (i is the binary value of the combination of registers `hl` for every $q_i \in St$). The cores are arranged in square, so that $k = 2$ and $neigh(a, 1) = a - 1$ for $a \in \{2, 3, 4\}$, $neigh(a, 2) = a + 1$ for $a \in \{1, 2, 3\}$, $neigh(1, 1) = 4$ and $neigh(4, 2) = 1$. Each core makes his register `l` observable to its neighbors, that is, $\Sigma = \{0, 1\}$, $man(q_i) = 0$ for $i \in \{0, 2\}$ and $man(q_i) = 1$ for $i \in \{1, 3\}$. The agents are capable of two actions ($Act = \{nop, add\}$). Action *nop* has no effect on the state of any agent and action *add* adds to the binary value of the agent's registers the binary value of the observable registers of the agent's neighbors. The transition function defines the effects of the actions as follows:

$$\begin{aligned} &\text{For every } i \in \{0, \ldots, 3\} \text{ and every } \sigma_1, \sigma_2 \in \Sigma \\ &\qquad tran(q_i, \sigma_1, \sigma_2, nop) = q_i \text{ and} \\ &\qquad tran(q_i, \sigma_1, \sigma_2, add) = q_{i+\sigma_1\sigma_2 \bmod 4}\,. \end{aligned} \tag{1}$$

Manifestation function models the incomplete information available to the agents; if $man(q_1) = man(q_2)$ for two different states of agent a then these states can not be distinguished by a's neighbors and no other agent beside a's neighbors has any information about a's state. Notice that all of a's neighbors have the same partial view of a's state (this is part of our assumption that the system being modeled is homogenous).

[1] Throughout the paper, we write $t[i]$ for the i^{th} element of a tuple t.

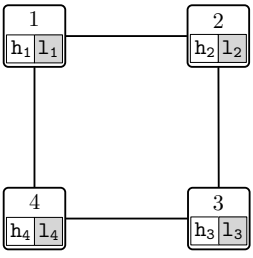


Fig. 1. A four-core processor with cores $1, \ldots, 4$, each of the cores has two boolean registers — $\mathtt{h}$ and $\mathtt{l}$ — where the value of $\mathtt{l}$ is observable by the neighboring cores

In what follows, we refer to the combinations of agent's state and manifestations of her neighbors as her *perception*. Then we can think of the transition function as assigning a's state to a combination of a's perception and action (thus, given a's perception and her action, her state changes according to the transition function).

2.2 Logical Formalism

Over the recent years, the logic of choice for reasoning about multi-agent systems has been ATL, introduced in [1]. ATL has pretty elaborate syntax, as it allows to reason about strategic ability of coalitions of agents, which requires the use of the so-called coalition modalities. As all the agents in the systems we consider in this paper are under our control, and we can model the presence of environment as a choice of what systems states the system is in at the beginning of the computation, we do not need the full syntactic machinery of ATL. Moreover, as all our computations are meant to be deterministic, it is sufficient for our purposes to use the syntax of the Linear-Time Temporal Logic LTL. (On the other hand, the semantics we are going to propose will differ substantially from the standard LTL semantics, to reflect the particularities of the models we consider.) Our syntax reflects the fact that propositional symbols are evaluated at agents', not system, states:

$$\gamma ::= \top \mid \mathtt{p_a} \mid \neg \mathtt{p_a}, \text{where } a \in Agt \text{ and } \mathtt{p_a} \in \Pi_a \tag{2}$$

$$\varphi ::= \gamma \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \mathsf{G}\varphi \mid \varphi \,\mathsf{U}\, \varphi, \text{ where } \gamma \text{ is defined as in (2).} \tag{3}$$

Such syntax only allows LTL formulas in positive normal form, that is, the ones where nothing but the statements about agent states can be negated. Every LTL formula can be translated into an equivalent formula in positive normal form.

Definition 2. *A* homogenous strategy *is a function* $S : St \times \Sigma^k \to Act$ *assigning actions to perceptions.*

For now, we only consider total functions. Following function S is one possible homogenous strategy for the model of our four-core processor:

$$\begin{aligned}\text{For every } i \in \{0,\dots,3\} \text{ and } \sigma_1,\sigma_2 \in \Sigma\\ S(q_i,\sigma_1\sigma_2) = \mathit{add} \text{ if } \sigma_1 = \sigma_2 \text{ and}\\ S(q_i,\sigma_1\sigma_2) = \mathit{nop} \text{ if } \sigma_1 \neq \sigma_2.\end{aligned}$$

Definition 3. *The* perception function $per : St^n \times Agt \to St \times \Sigma^k$ *assigns to a system state and an agent her perception of that system state:* $per(Q,a) = \langle Q[a], man(Q[neig(a,1)]),\dots,man(Q[neig(a,k)])\rangle$ *where* Q *is a system state,* $a \in Agt$ *and* k *is the neighborhood size.*

For example, $per(\langle q_0,q_2,q_3,q_3\rangle,3)$, that is, agent's 3 perception of system state $Q = \langle q_0,q_2,q_3,q_3\rangle$, would be $\langle q_3,0,1\rangle$, because the state of agent 3 is q_3, the manifestation of the state of its left neighbor (agent 2) is 0 and the manifestation of the state of its right neighbor (agent 4) is 1.

Definition 4. *An* action vector *is an* n*-tuple of actions from* Act*, one for each* $a \in Agt$*.*

Given system states Q and Q', we say that action vector A leads from Q to Q' if $tran(per(Q,a),A[a]) = Q'[a]$ for every $a \in Agt$.

Definition 5. *A* path *is an infinite sequence of system states* $\Lambda = Q_1,Q_2,Q_3\dots$ *that can be effected by subsequent action vectors; that is, for every* $j \geq 1$*, there exists an action vector* $A \in Act^n$ *leading from* Q_j *to* Q_{j+1}*. The* j^{th} *component of* Λ *is denoted by* $\Lambda[j]$*.*

Definition 6. *The* outcome *of a homogenous strategy* S *at system state* Q*, denoted by* $out(Q,S)$*, is the path* Λ *such that* $\Lambda[1] = Q$ *and*

$$\Lambda_{j+1} = \prod_{a\in Agt} tran(per(\Lambda_j,a),S(per(\Lambda_j,a))),$$

for every $j \geq 1$*.*

Given a set of system states $\mathbb{Q}$, we use notation $out(\mathbb{Q},S)$ as a shorthand for $\bigcup_{Q\in\mathbb{Q}} out(Q,S)$.

Informally, given a modular model $\mathfrak{M}$ and a set of system states $\mathbb{Q}$, an LTL formula φ holds at $\mathbb{Q}$ if there exists a homogenous strategy S such that for every system state $Q \in \mathbb{Q}$ the outcome of S at Q satisfies φ. Formally:

$\mathfrak{M},\mathbb{Q} \models \varphi$ iff there is a strategy S such that $\Lambda \Vdash_{\mathrm{LTL}} \varphi$ for every $\Lambda \in out(\mathbb{Q},S)$,

where $\Lambda \Vdash_{\mathrm{LTL}} \varphi$ holds iff Λ satisfies φ according to the standard LTL semantics. From the semantics above it follows that,

- $\mathfrak{M},\mathbb{Q} \models \mathtt{p_a}$ iff $Q[a] \in \pi(\mathtt{p_a})$ for every $Q \in \mathbb{Q}$

– $\mathfrak{M}, \mathbb{Q} \models \neg \mathtt{p_a}$ iff $Q[a] \notin \pi(\mathtt{p_a})$ for every $Q \in \mathbb{Q}$

where $a \in Agt$ and $\mathtt{p_a} \in \Pi_a$.

If the outcome of a strategy at some state satisfies a formula φ then we say that the strategy *enforces* φ. For a given strategy enforcing a formula φ we want to know the set of *all* system states at which the outcome of the strategy satisfies φ; we refer to this set as the *domain* of the strategy. This is the reason why we evaluate formulas at sets of system states rather than at states themselves. We took the idea of evaluating formulas at sets of states from [5].

Note that in our model, all the information available for an agent's decision what action to perform is stored in the current local states of the agent and her neighbors, including a (finite) section of history if required. Since the strategies are based on the perceptions of current states, they can be seen as memoryless strategies.

3 Synthesis of All Maximal Homogenous Strategies

Our aim in this paper is, given a modular model $\mathfrak{M}$ and an LTL formula φ, to synthesize *all maximal homogenous strategies enforcing* φ, i.e., all homogenous strategies enforcing φ whose domains cannot be extended.[2]

3.1 Naive Solution

The naive solution (Alg. 1) is to generate all the possible strategies for a given modular model, and then, for each of those strategies, to check at which system states its outcome satisfies φ. Finally, we have to check the resulting strategies for maximality, excluding all non-maximal strategies from our solution.

We now briefly outline this "naive" approach. The algorithm below uses function *mc*, which for a given LTL formula φ and a transition system T returns the set of all states in T at which φ holds. The transition system T passed to *mc* represents all outcomes of a strategy; therefore, there exists exactly one outgoing transition from each state of T. Therefore, the number of transitions in T equals the number of system states in T, i.e., $|St|^n$. The model checking of φ on the outcomes of a strategy is thus polynomial to the number of system states and length of the formula.

The naive solution outlined above has two major drawbacks. First, we have to go through all possible (total) strategies, whose number equals $|Act|^{|St| \cdot |\Sigma|^k}$, which is a pretty large number. Second, we have, prior to the model checking stage, generated the whole transition system corresponding to our modular model (i.e., the transition system containing every possible system state; the number of such states, as we have seen, is $|St|^n$). In other words, following this "naive" approach, we do not take advantage of the fact that, from the very beginning, we know what formula we want to enforce—we blindly generate all the possible total strategies, and only then check which of them enforce φ.

[2] From now on, we only consider homogenous strategies and omit the word "homogenous".

Algorithm 1. Naive Synthesis

Data an LTL formula φ to enforce and a model $\mathfrak{M}$
Result a set of all maximal strategies enforcing φ, together with their domains
1: $Out := \emptyset$
2: *//generate the set of all homogenous strategies Str*
3: **for all** $S \in Str$ **do**
//generate the entire transition system T by following S at every system state
4: $T := \{\langle Q, Q'\rangle \mid Q'[i] = tran(per(Q,i), S(per(Q,i)))$ for every $1 \leq i \leq n\}$
//do global model checking for $T \models \varphi$
5: $D = mc(\varphi, T)$ *//D is the set of all system states from T where* φ *holds*
6: **if** $D \neq \emptyset$ **then**
7: $Out := Out \cup \{\langle S, D\rangle\}$
//remove non-maximal solutions
8: **for all** $\langle S, D\rangle \in Out$ **do**
9: **if** $\exists\langle S', D'\rangle \in Out$ such that $D \subset D'$ **then** $Out := Out \setminus \{\langle S, D\rangle\}$
return Out

The more sophisticated approach that we are going to consider in the rest of the paper immediately takes advantage of such knowledge; namely, we only ever generate strategies enforcing φ.

3.2 Incremental solution

In this subsection, we explore the solution that only ever considers strategies that enforce the desired property φ (as opposed to exploring all the strategies).

Since we construct the strategies for a formula φ incrementally, we work with partial functions defined only on a subset of all possible perceptions. We can see a partial strategy S as a set of pairs "perception—assigned action": $S \subseteq \{\langle q, m_1, \ldots, m_k, \alpha\rangle \in St \times \Sigma^k \times Act\}$.

Definition 7. *The outcome of a* partial *strategy S at a system state Q, denoted by $out(Q, S)$, is the (possibly finite) sequence Λ of system states such that $\Lambda[1] = Q$ and $\Lambda_{j+1} = \prod_{a\in Agt} tran(per(\Lambda_j, a), S(per(\Lambda_j, a)))$ for every $1 \leq j \leq |\Lambda|$. If Λ is finite, then it ends with a system state Q' such that S is not defined for $per(Q', a)$ for at least one $a \in Agt$.*

Given a set of system states $\mathbb{Q}$, we use $out(\mathbb{Q}, S)$ as a shorthand for $\bigcup_{Q\in\mathbb{Q}} out(Q, S)$. We use Λ^i for the suffix of Λ starting at $\Lambda[i]$. We use $\overline{\Lambda}$ for the set of all paths with the prefix Λ. If Λ is infinite then $\overline{\Lambda} = \{\Lambda\}$. We say that a sequence of states Λ satisfies an LTL formula φ if $\Lambda' \Vdash_{\text{LTL}} \varphi$ for every $\Lambda' \in \overline{\Lambda}$.

Definition 8 (Strategy for a formula). *A partial strategy* for an LTL formula *φ with domain $\mathbb{D}$ is a partial strategy S such that $\Lambda' \Vdash_{LTL} \varphi$ for every $\Lambda' \in \overline{\Lambda}$ and every $\Lambda \in out(\mathbb{D}, S)$.*

Definition 9 (Compatibility). *Two partial strategies S and S' are* compatible *if they assign the same actions to the same perceptions, i.e., if both S and S' are defined for $\langle q, m_1, \ldots, m_k\rangle$ then $S(q, m_1, \ldots, m_k) = S'(q, m_1, \ldots, m_k)$.*

If partial strategies S' and S'' are compatible, we define the joint strategy $S = S' \cup S''$ at every state at which either S' or S'' is defined. We can join compatible strategies into a *maximal* strategy:

Definition 10 (Maximal strategy). *A partial strategy S for an LTL formula φ with domain $\mathbb{D}$ is* maximal *if:*

1. *there is no other partial strategy $S' \supset S$ for φ with domain $\mathbb{D}' \supset \mathbb{D}$ and*
2. *there is no other partial strategy $S'' \subset S$ for φ with domain $\mathbb{D}$*

A *minimal* strategy for an LTL formula φ and a system state Q is defined exactly for all the agent states that occur in the computation starting at Q until φ is satisfied. Formally:

Definition 11 (Minimal strategy). *Let S be a partial strategy for an LTL formula φ with a system state Q in its domain, let Λ be the outcome of S at Q and let Λ_p be the shortest prefix of Λ such that $\Lambda' \Vdash_{LTL} \varphi$ for every $\Lambda' \in \overline{\Lambda_p}$. S is a* minimal *strategy for φ and Q if S is defined exactly for the set $\{\langle q, m_1, \cdots m_k\rangle \in St \times \Sigma^k \mid \langle q, m_1, \cdots m_k\rangle = per(Q, a)$ for some $Q \in \Lambda_p$ and some $a \in Agt\}$*

We construct all maximal strategies for an LTL formula φ in two steps:

1. We iteratively find every system state Q from which φ can be enforced and every minimal strategy for φ and Q.
2. We find every combination of the compatible minimal strategies such that the joint strategy is a maximal strategy for φ.

To accomplish the first step, for every subformula φ' of φ, we construct a directed graph representing the outcomes of minimal strategies enforcing φ'. Every vertex $\langle Q, S\rangle$ of such graph represents a system state Q and a minimal strategy S that enforces φ' at Q. Every edge directed from vertex v to vertex u represents an action vector given by the strategy from v, that leads from the system state from v to the system state from u, that is, given $v = \langle Q, S\rangle$ and $u = \langle Q', S'\rangle$, if an edge is directed from v to u then $\Lambda[2] = Q'$ where $\Lambda = out(Q, S)$ and $S' \subseteq S$. Given a subgraph $F = \langle V_F, E_F\rangle$ of such graph, we refer with $[F]$ to the set $\{Q \in St^n \mid \langle Q, S\rangle \in V_F$ for some $S\}$.

Informally, for each subformula φ' of φ, we obtain a *directed forest* $F_{\varphi'}$ (see Def. 12). A vertex $\langle Q, S\rangle$ is included in $F_{\varphi'}$ iff S is a minimal strategy for φ' and Q.[3] Maximal strategies for φ consist of partial strategies given by the directed forests for the subformulas with incompatible branches cut off.

Definition 12. *A* directed tree *is a connected directed graph $\langle V, E\rangle$ such that $|V| = |E| + 1$ and every vertex has at most one outgoing edge. The* root *of a directed forest $\langle V, E\rangle$ is $v \in V$ such that there is a path from u to v for every $u \in V$. A* directed forest *is a set of directed trees.*

[3] If $S = \emptyset$ then *any* strategy enforces φ' at Q.

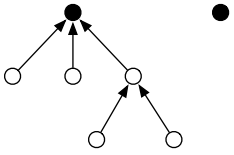


Fig. 2. A directed forest consisting of two trees. The black vertices are the roots.

Remark 1. Let F and F' be directed forests, let $\langle Q_1, S_1\rangle$ be a non-root vertex in F with its direct successor being $\langle Q_2, S_2\rangle$. Then $S_2 \subseteq S_1$ and if there is a non-root vertex $\langle Q_1, S_1' \supseteq S_1\rangle$ in F' then its direct successor is $\langle Q_2, S_2'\rangle$ where $S_2 \subseteq S_2' \subseteq S_1'$.

Consequently, given a vertex $\langle Q, S\rangle$ in a tree of directed forest F with root $\langle Q_R, S_R \subseteq S\rangle$, whenever there is a vertex $\langle Q, S' \supseteq S\rangle$ in another directed forest F', then for the branch $\langle Q, S\rangle, \ldots, \langle Q_R, S_R\rangle$ of F there is a sequence of vertices $\langle Q, S'\rangle, \ldots, \langle Q_R, S_R'\rangle$ such that $S_R \subseteq S_R' \subseteq S'$ in F'.

The basic operation in constructing a directed forest for a formula φ is computing the set of all predecessors of a vertex $v = \langle Q, S\rangle$, using function *pre*:

$$\begin{aligned} pre(Q,S) = \{\langle Q', S'\rangle \mid\ & S' \text{ is the smallest strategy such that} \\ & \Lambda[2] = Q \text{ where } \Lambda = out(Q', S') \text{ and } S \subseteq S'\} \end{aligned}$$

To compute $pre(Q, S)$, we construct a colored undirected graph $C_{(Q,S)} = \langle V, E\rangle$ where V is partitioned into sets of vertices $V_1, \ldots, V_n$ $(n = |Agt|)$, with every vertex from V_a representing a perception and an action of agent a:

$$\begin{aligned} V_a = \{\langle s, m_1, \ldots, m_k, \alpha\rangle \in St \times \Sigma^k \times Act \mid\ & \\ tran(s, m_1, \ldots, m_k, \alpha) = Q[a] \text{ and } & \\ S(s, m_1, \ldots, m_k) = \alpha \text{ or is undefined } \} & \end{aligned}$$

We say that every vertex in V_a has color a. Two vertices from V are connected iff they represent mutually non-conflicting perceptions of the neighboring agents:

$$\begin{aligned} E = \{&\langle s, m_1, \ldots, m_k, \alpha\rangle \in V_a, \langle s', m_1', \ldots, m_k', \alpha'\rangle \in V_b \mid \\ &\exists 1 \leq l \leq k \text{ such that } neig(a, l) = b \text{ and } m_l = man(s') \text{ and} \\ &\exists 1 \leq l' \leq k \text{ such that } neig(b, l') = a \text{ and } m_{l'}' = man(s)\} \end{aligned}$$

Consider a subgraph $\langle V', E'\rangle$ of $C_{(Q,S)}$ such that:

1. V' contains, for every $a \in Agt$, exactly one vertex v_a from V_a (let $v_a = \langle s, m_1, \ldots m_k, \alpha\rangle$),
2. for every two $a, a' \in Agt$, $\langle v_a, v_{a'}\rangle \in E'$ iff a and a' are neighbors and
3. for every $v_a, v_{a'} \in V'$, if v_a and $v_{a'}$ represent the same perceptions then they also represent the same actions.

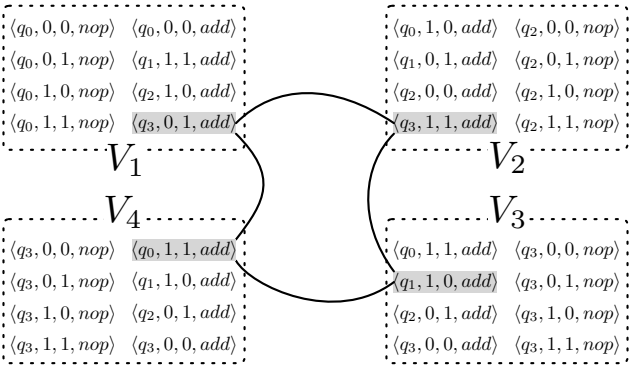


Fig. 3. Graph $C_{Q,S}$ where $Q = \langle q_0, q_2, q_3, q_3\rangle$ and $S = \emptyset$ (each tuple represents one vertex, edges of $C_{Q,S}$ are not depicted). The dashed squares represent the colored partitions of the vertices for the respective agents. One subgraph representing a predecessor of Q is depicted, with grey vertices and solid edges. The subgraph represents state $Q' = \langle q_3, q_3, q_1, q_0\rangle$ and strategy S' composed of the vertices of the subgraph. The second state of the outcome of S' at Q' is Q.

Every such subgraph represents one system state Q' where $Q'[a] = s$ and one partial strategy $S' = S \cup \bigcup_{a \in Agt} v_a$. Thus, $\langle Q', S'\rangle \in pre(Q, S)$ and the subgraphs of $C_{(Q,S)}$ satisfying the conditions above represent all predecessors of $\langle Q, S\rangle$. An example of such subgraph for $pre(Q = \langle q_0, q_2, q_3, q_3\rangle, S = \emptyset)$ and for our model with transition function *tran* (1) is depicted in Fig. 3.

Other auxiliary functions used for construction of the directed trees are:

- *roots*(F), returns all roots of the directed forest F;
- *predecessors*$(\langle Q, S\rangle, F)$, returns all predecessors of the vertex $\langle Q, S\rangle$ in the directed forest F;
- *traverse*$(\langle Q, S\rangle, F)$, returns all vertices of the (sub)tree rooted in $\langle Q, S\rangle$ in the directed forest F; and
- *prune*(T, R, S'), in directed tree T cuts off all subtrees rooted in a vertex representing state R or representing a strategy incompatible with strategy S' and extends the strategies in the remaining vertices with S' (T may be empty after this operation).

Using the functions described above, we construct, for each subformula φ' of φ, the following directed forest $F_{\varphi'} = \langle V_{\varphi'}, E_{\varphi'}\rangle$:

- **case** $\varphi' = \mathtt{p_a}$:
  $V_{\varphi'} = \{\langle Q, \emptyset\rangle \mid \{Q\} \models \mathtt{p_a}\}$; $E_{\varphi'} = \emptyset$;
- **case** $\varphi' = \neg\mathtt{p_a}$:
  $V_{\varphi'} = \{\langle Q, \emptyset\rangle \mid \{Q\} \models \neg\mathtt{p_a}\}$; $E_{\varphi'} = \emptyset$;
- **case** $\varphi' = \psi_1 \vee \psi_2$:
  $F_{\varphi'} = F_{\varphi_1} \cup F_{\varphi_2}$;
- **case** $\varphi' = \psi_1 \wedge \psi_2$:

$$V_{\varphi'} = \{\langle Q, S\rangle \mid \text{ either } \langle Q, S\rangle \in F_{\psi_1} \text{ and } \langle Q, S'\rangle \in F_{\psi_2} \text{ for some } S' \subseteq S \text{ or } \langle Q, S\rangle \in F_{\psi_2} \text{ and } \langle Q, S'\rangle \in F_{\psi_1} \text{ for some } S' \subseteq S\};$$
$$E_{\varphi'} = \emptyset;$$

– **case** $\varphi' = \mathsf{X}\psi$
We alter a copy of F_ψ as follows: For every vertex $\langle Q, S\rangle$ in F_ψ (obtained by function *traverse*), we find all its predecessors by $pre(Q, S)$ and append to $\langle Q, S\rangle$ those that are not preceding $\langle Q, S\rangle$ in F_ψ. Moreover, we remove every vertex that is a root in F_ψ. Thus, the predecessors of the removed roots become roots themselves. The resulting directed forest is $F_{\varphi'}$.

$$V_{\varphi'} = (V_\psi \cup \{\langle Q, S\rangle \mid \langle Q, S\rangle \in pre(Q', S') \text{ for some } \langle Q', S'\rangle \in F_\psi\}) \setminus \{\langle Q, S\rangle \mid \langle Q, S\rangle \in roots(F_\psi)\};$$
$$E_{\varphi'} = \{\langle \langle Q, S\rangle, \langle Q', S'\rangle\rangle \mid \langle Q, S\rangle \in pre(Q', S')\};$$

– **case** $\varphi' = \psi_1 \mathsf{U} \psi_2$ (Fig. 4):
We use function $reach(\langle Q, S\rangle, R, F_1, F_2)$ to construct $F_{\varphi'}$. The function appends to every vertex $\langle Q, S\rangle$ from the forest F_2, directly or indirectly, every branch of the forest F_1 from which $\langle Q, S\rangle$ is reachable. Where possible, the function extends the strategy S' of a vertex $\langle Q', S'\rangle$ from the directed forest F_1 so that, when following the extended strategy at Q', after reaching the root of $\langle Q', S'\rangle$ in F_1, vertex $\langle Q, S\rangle$ in F_2 is reached solely through the system states represented by vertices of F_1 (the function uses *pre*, *roots* and *prune*, see Alg. 2). In case that Q' is identical with the system state R represented by the root of $\langle Q, S\rangle$, the entire subtree rooted in $\langle Q', S'\rangle$ is excluded from the result. Thus, every system state is represented in any given branch of the resulting tree at most once and unfolding of potential cycles is prevented. At the end, altered forest F_2 is $F_{\varphi'}$.

Algorithm 2. $reach(\langle Q, S\rangle, R, F_1, F_2)$

Data a directed forest F_2, a vertex $\langle Q, S\rangle$ from F_2, the state R represented by the root of the tree containing $\langle Q, S\rangle$ and a directed forest F_1
1: **for all** $\langle Q', S'\rangle \in roots(F_1)$ such that $Q' = Q$ and S' is compatible with S **do**
2: copy the tree rooted in $\langle Q', S'\rangle$ as T;
3: $prune(T, R, S \cup S')$;
4: append every subtree of T rooted in a predecessor of the root of T to $\langle Q, S\rangle$ in F_2;
5: **for all** $\langle Q', S'\rangle \in roots(F_1)$ such that $Q' \neq Q$ and there is $\langle Q', S''\rangle \in pre(Q, S)$ where S' is compatible with S'' **do**
6: copy the tree rooted in $\langle Q', S'\rangle$ as T;
7: $prune(T, R, S'')$;
8: append T to $\langle Q, S\rangle$ in F_2;
9: **for all** $\langle Q', S'\rangle \in predecessors(\langle Q, S, \rangle, F_2)$ **do**
10: $reach(\langle Q', S'\rangle, R, F_1, F_2)$;

(a). Forests for ψ_1 and ψ_2.

(b). Forest for $\psi_1 \mathsf{U} \psi_2$.

Fig. 4. An example of $F_{\psi_1 \mathsf{U} \psi_2}$. The dashed vertex is excluded from $F_{\psi_1 \mathsf{U} \psi_2}$ by $reach(\langle Q, S\rangle, R, F_{\psi_1}, F_{\psi_2})$ because its strategy is incompatible with S or because it represents the same state as the root of F_{ψ_2}.

– **case** $\varphi' = \mathsf{G}\,\psi$:
We represent with vertices of $F_{\varphi'}$ all strategies that eventually enforce cycles visiting only the states from $[F_\psi]$. We use function $cycle(\langle Q, S\rangle, R, \{\langle Q, S\rangle\}, F_\psi, F_{\varphi'})$ to add to $F_{\varphi'}$ all such strategies that, in addition, include state $\langle Q, S\rangle$ in the cycle, where R is the state represented by the root of $\langle Q, S\rangle$ in F_ψ (see Alg. 3). Since we use $cycle(\langle Q, S\rangle, R, \{\langle Q, S\rangle\}, F_\psi, F_{\varphi'})$ for every vertex $\langle Q, S\rangle$ that is a root in F_ψ, the resulting forest is $F_{\varphi'}$.

Algorithm 3. $cycle(\langle Q, S\rangle, R, T, F_1, F_2)$

Data a directed forest F_1, a tree T, a vertex $\langle Q, S\rangle$ from T, the state R represented by the root of the tree in F_1 containing $\langle Q, S\rangle$, a directed forest F_2 to add the results.

```
1: ℙ = {⟨Q', S'⟩ ∈ pre(Q, S) | ∃⟨Q', S''⟩ ⊆ S'⟩ ∈ F1};
2: append every ⟨Q', S'⟩ ∈ ℙ to ⟨Q, S⟩ in T;
3: for all ⟨Q', S'⟩ ∈ ℙ do
4:     if Q' = R then
5:         cut off ⟨Q', S'⟩ from T;
6:         copy T to F2 as T';
7:         prune(T', R, S');
8:     else
9:         cycle(⟨Q', S'⟩, R, T, F1, F2);
```

Now we show, how the directed forest F_φ for a formula φ represents all minimal strategies for φ.

Claim. Let φ be an LTL formula and let F_φ be the directed forest for φ. S is a minimal homogenous strategy for φ and a system state Q iff $\langle Q, S\rangle$ is a vertex of F_φ.

Proof. We only proof the nontrivial cases:

– **case** $\varphi = \mathsf{X}\psi$:
S_ψ is a minimal strategy for ψ and state Q_ψ iff there is $\langle Q_\psi, S_\psi\rangle \in V_\psi$. Let S be a strategy, let Q be a state and let $\Lambda = out(Q, S)$. Vertex $\langle Q, S\rangle$ is

present in F_φ iff $\langle Q, S\rangle \in pre(Q', S')$ for some $\langle Q', S'\rangle \in V_\psi$ iff, for some $\langle Q', S'\rangle \in V_\psi$, $\Lambda[2] = Q'$ and $S \supseteq S'$ is defined for the same agent states as S' and, moreover, for the agent states from Q iff S is a minimal strategy for φ and Q.

– **case** $\varphi' = \psi_1 \mathsf{U} \psi_2$:
($\Leftarrow$): Let $\langle Q, S\rangle$ be a vertex from $F_{\varphi'}$. Either (i) $\langle Q, S\rangle \in V_{\psi_2}$ or (ii) $\langle Q, S\rangle$ is added to $F_{\varphi'}$ by function *reach* from F_{ψ_1} and thus some $\langle Q, S_{\psi_1} \subseteq S\rangle \in V_{\psi_1}$. $\langle Q, S\rangle \in V_{\psi_2}$ iff S is a minimal strategy for ψ_2 and Q. Every minimal strategy for ψ_2 and Q is also a minimal strategy for φ' and Q. Thus, if (i) then S is a minimal strategy for φ' and Q. If (ii) then S is the union of compatible strategies S_r and S_{ψ_2} where (1) the last state of $\Lambda = out(Q, S_r)$ is Q' such that $\langle Q', S_{\psi_2}\rangle \in V_{\psi_2}$ and (2) for every other state $Q_\Lambda \in \Lambda$ there is some vertex $\langle Q_\Lambda, S_\Lambda \subseteq S_r\rangle \in F_{\psi_1}$. Thus, the outcome $\Lambda_{\varphi'}$ of $\langle Q, S\rangle$ consists of Λ followed by $\Lambda_2 = out(Q', S_{\psi_2})$. Since Λ_r is composed of the outcomes of minimal strategies for ψ_1 and ends with Q', every path starting before Q' and following $\Lambda_{\varphi'}$ satisfies ψ_1 and every path starting from Q' on and following $\Lambda_{\varphi'}$ satisfies ψ_2. Thus, there is $i \geq 1$ such that $\Lambda' \Vdash_{\mathrm{LTL}} \psi_2$ for every $\Lambda' \in \overline{\Lambda^i_{\varphi'}}$ and $\Lambda' \Vdash_{\mathrm{LTL}} \psi_1$ for every $\Lambda' \in \overline{\Lambda^j_{\varphi'}}$ and every $1 \leq j < i$, namely such i that $\Lambda_{\varphi'}[i] = Q'$. Thus, S is a strategy for φ' and Q.

Strategy S_r is defined for the same agent states as the minimal strategies for ψ_1 from which S_r is composed and, moreover, for the agent states from the last system states of the outcomes of the minimal strategies for ψ_1. Thus, every agent state for which S_r is defined is present at some system state from the outcome of S_r at Q. Since S_{ψ_2} is a minimal strategy for ψ_2 and Q', every for which S_{ψ_2} is defined is present at some system state from the outcome of S_{ψ_2} at Q'. Since $S = S_r \cup S_{\psi_2}$, every agent state for which S is defined is present at some system state from the outcome of S at Q. Let $\Lambda_{\varphi'} = out(S, Q)$ with length l and $Q' = \Lambda_{\varphi'}[i]$ for some i. Since every path satisfying φ' must contain a state from $[F_{\psi_2}]$ and $\Lambda_{\varphi'}[i]$ is the first such state in $\Lambda_{\varphi'}$, there is no $j < i$ such that $\Lambda' \Vdash_{\mathrm{LTL}} \varphi'$ for every $\Lambda' \in \overline{\Lambda^j}$. Since S_{ψ_2} is a minimal strategy for ψ_2 and Q', there is no prefix of the outcome of S_{ψ_2} such that all paths starting with this prefix satisfy ψ_2 and thus, there is no $i \leq j < l$ such that $\Lambda' \Vdash_{\mathrm{LTL}} \varphi'$ for every $\Lambda' \in \overline{\Lambda^j}$. Hence, S is a minimal strategy for φ' and Q.
($\Rightarrow$): Let S be a minimal strategy for ψ and state Q. Then there is $i \geq 1$ such that, for every path Λ' with prefix $\Lambda = out(Q, S)$, $\Lambda'^i \Vdash_{\mathrm{LTL}} \psi_2$ and $\Lambda'^j \Vdash_{\mathrm{LTL}} \psi_1$ for every $1 \leq j < i$. Thus, some $S_{\psi_2} \subseteq S$ is a minimal strategy for ψ_2 and $\Lambda[i]$ and therefore there is a vertex $\langle \Lambda[i], S_{\psi_2} \subseteq S\rangle$ in F_{ψ_2}. If $i = 1$ then S itself is a minimal strategy for ψ_2 and Q and thus, vertex $\langle Q, S\rangle \in F_{\psi_2}$. Since $F_{\psi_2} \subseteq F_{\varphi'}$, $\langle Q, S\rangle$ is a vertex of $F_{\varphi'}$. If $i > 1$ then for every $\Lambda[j]$ there is a vertex representing $\Lambda[j]$ in F_{ψ_1}. Since $\Lambda[i]$ is reachable from every $\Lambda[j]$ by visiting only states from $[F_{\psi_1}]$, $\Lambda[j]$ is found by function *reach* for every $1 \leq j < i$ and vertex $\langle \Lambda[j], S_j\rangle$ is added to $F_{\varphi'}$. Strategy S_j contains S_{ψ_2} and, in addition, is defined for every agent state present in some $\Lambda[l]$ where $j \leq l < i$ so that the outcome of S_j at $\Lambda[j]$ is Λ^j. Thus,

$out(\Lambda[1], S_1) = \Lambda^1 = \Lambda$ where $\Lambda = out(Q, S)$. Since S is a minimal strategy for φ' and Q and the outcomes of S_1 and S at Q are identical, $S^1 = S$. Thus, $\langle \Lambda[1], S_1 \rangle = \langle Q, S \rangle$ and $\langle Q, S \rangle \in F_{\varphi'}$.

– **case** $\varphi' = \mathsf{G}\,\psi$:
($\Leftarrow$): Let $\langle Q, S \rangle$ be a vertex from $F_{\varphi'}$ contained in a tree T. Tree T with root $\langle R, S_R \rangle$ is added by *cycle* to $F_{\varphi'}$ only if $out(R, S_R)[2] = Q_R$ for some $\langle Q_R, S_R \rangle \in T$. For every vertex $\langle Q', S' \rangle$ from T, the outcome of S' at Q' contains R and thus, also the outcome of S_R at Q_R contains R. Therefore, $\Lambda_R = out(R, S_R)$ contains R infinitely often and thus, Λ_R is infinite. Since $S_R \subseteq S'$ and $R \in out(Q', S')$ for every $\langle Q', S' \rangle \in T$, Λ_R is a suffix of $out(Q', S')$ for every $\langle Q', S' \rangle \in T$. Thus, $out(Q, S)$ is infinite.
As we have shown, $\Lambda = out(Q, S)$ is a path (i.e., an infinite sequence) consisting solely of Q followed by the system states represented by the successors of $\langle Q, S \rangle$ in T. Since T is constructed by function *cycle* from F_ψ, for every $\langle Q', S' \rangle \in T$ there is some $\langle Q', S_\psi \subseteq S' \rangle \in F_\psi$. Thus, for every $i \geq 1$, Λ^i has a prefix $out(\Lambda[i], S_i \subseteq S)$ where $\langle \Lambda[i], S_i \rangle$ is a vertex from F_ψ. Since S_i is a minimal strategy for ψ and $\Lambda[i]$, every path starting with $out(\Lambda[i], S_i)$ satisfies ψ and thus, also Λ^i satisfies ψ. Therefore, S is a strategy for $\varphi' = \mathsf{G}\,\psi$ with Q in its domain.

Strategy S is composed of strategies S_i such that $\langle \Lambda[i], S_i \rangle \in F_\psi$ where $\Lambda = out(Q, S)$ and $i \geq 1$. Since every S_i is a minimal strategy for ψ and $\Lambda[i]$, it is defined solely for the agent states present in all system states in $out(\Lambda[i], S_i)$ except the last one. Additionally, S is defined also for all agent states present in the last system state of $out(\Lambda[i], S_i)$ for every $i \geq 1$ so that Λ is infinite. Thus, S is defined exactly of the agent states present in the system states from $\Lambda = out(Q, S)$ and since Λ satisfies φ', S is a minimal strategy for φ' and Q.
($\Rightarrow$): Let S be a minimal strategy for φ' and state Q. Then $\Lambda = out(Q, S)$ is infinite and $\Lambda^i \Vdash_{\mathrm{LTL}} \psi$ for every $i > 0$. Thus, for every $i > 0$ there is some minimal strategy S_i for ψ and $\Lambda[i]$ such that $out(\Lambda[i], S_i)$ is a prefix of Λ^i and therefore $\langle \Lambda[i], S_i \rangle \in F_\psi$ for every $i > 0$ and some $S_i \subseteq S$.

First, consider the case that $Q = \Lambda[1]$ occurs in Λ infinitely often. Thus, Q is part of a cycle and can be reached from Q through the sequence of predecessors of Q. Since $\langle Q, S_1 \rangle \in F_\psi$, function *cycle* looks for vertices $\langle Q', S' \rangle$ such that $\Lambda'[i] = Q$ for some $i > 0$ and $\Lambda' = out(Q', S')$ and $\langle \Lambda'[j], S'_j \subseteq S' \rangle \in F_\psi$ for every $0 < j < i$ until $Q' = Q$. S' is always defined exactly for the agent states occurring at system states from $out(Q', S')$. In case that $Q' = Q$, outcome of S' at Q' contains Q infinitely often. Since S is also defined exactly for the for the agent states occurring at system states from $out(Q, S)$, vertex $\langle Q, S \rangle$ is found be *cycle* and a tree with root $\langle Q, S \rangle$ is added to $F_{\varphi'}$.

Second, consider that $Q = \Lambda[1]$ does not occur in Λ infinitely often (that implies that it occurs in Λ only once). Since Λ is infinite, there is $Q' \neq Q$ that occurs in Λ infinitely often and a vertex $\langle Q', S' \subseteq S \rangle$ was found by *cycle* such that $out(Q', S')$ is a suffix of Λ. Function *cycle* adds to F_ψ every vertex

$\langle Q'', S'' \supseteq S' \rangle$ such that $Q' \in out(Q'', S'')$ and S'' is only defined for the agent states occurring at the system states from $out(Q'', S'')$. Thus, $\langle Q, S \rangle$ is found as well and added to $F_{\varphi'}$. □

Definition 13. *A* branch *is a path in a directed tree that starts with a leaf and ends with the root. A* subbranch *is a suffix of a branch.*

Definition 14. *A* maximal compatible subforest *M of a directed forest F is a set of subbranches of F such that:*

1. *all strategies from the vertices of M are mutually compatible and*
2. *no other subbranch of F can be added to M without violating condition 1.*

In the second step, all maximal compatible subforests of F_φ are generated during depth-first search through F_φ.

Claim. Let F_φ be the directed forest for an LTL formula φ. M is a maximal strategy for φ iff M is a maximal compatible subforest of F_φ.

Proof. Straightforward. □

Hence, after completing the second step, all maximal strategies for φ are found and the task of the strategy synthesis is fulfilled.

4 Complexity and Comparison

The naive approach described in Section 3.1 involves model checking on every transition system generated by a total strategy. Since a total strategy assigns $\alpha \in Act$ to every possible perception from $St \times \Sigma^k$, where k is the maximal size of the neighborhood, there exist $|Act|^{|St| \cdot |\Sigma|^k}$ total strategies. As the number of manifestations ($|\Sigma|$) is in the worst case equal to the number of agent states ($|St|$), the number of all strategies is always less or equal $S = |Act|^{|St|^{k+1}}$. As we have shown in Section 3.1, the model checking of one total strategy for a formula φ of length l involves at most $l \cdot |St|^n$ steps. Thus, the worst case complexity of finding all total strategies for φ is $O(l \cdot |St|^n \cdot |Act|^{|St|^{k+1}})$. To keep only *maximal* strategies for φ, we have to drop strategies with non-maximal domains. Thus, we have to mutually compare the domains of all total strategies for φ and the overall complexity is $O(|Act|^{2 \cdot |St|^{k+1}})$.

The worst case complexity of the incremental approach described in Section 3.2 is worse than that of the naive approach. To show this, we estimate the upper bound on the size of the directed forest F_φ from which all maximal strategies for formula φ are computed. We denote the number of vertices of F_φ with $|F_\varphi|$. This number is given by the maximal number of the predecessors of a vertex in the forest (denoted by deg_{in}) to the power of the maximal depth of the tree. Since every branch represents in its vertices each global state at most once, the maximal depth of the tree is the number of global states, which is

$|St|^n$. Thus, $O(|F_\varphi|) = O({deg_{in}}^{|St|^n})$. Assume that number of agents is bigger than the size if the neighborhood ($n > k + 1$) and that deg_{in} is at least two (which is usually the case), and therefore, regardless of the complexity of *pre* function used for finding predecessors in F_φ, the construction of F_φ already may take more steps than synthesizing all maximal strategies for φ with the naive approach ($O(2^{|St|^n}) > O(|Act|^{2|St|^{k+1}})$).

The incremental approach may take more time than the naive one for example if $\varphi = \mathsf{G}\top$. While any total strategy is a maximal strategy for $\mathsf{G}\top$, the non-trivial approach would still look for all predecessors of all global states while iteratively generating strategies that eventually enforce cycles. This computation would be in this case more expensive than simple generation of all total strategies. However, we believe that in case of non-trivial formulas, which usually cannot be enforced by every arbitrary strategy, incremental constructing of only such strategies that enforce the formula takes substantially fewer steps that following the naive approach. This is a phenomenon known in other areas. For example, as shown in [4], a 2-EXPTIME algorithm for deciding satisfiability in LTL vastly outperforms in practice an algorithm whose worst-time theoretical complexity is singly exponential.

5 Conclusion

In this paper, we described a technique for the synthesis of strategies for homogenous multi-agent systems with incomplete information. While there exists a general solution for this problem in the context of heterogenous systems [2], it is not directly applicable in scenarios where the agents are homogenous and each of them must follow the same strategy.

We proposed a modular system model that takes the advantage of the assumption on the homogeneity of the agents and allows for a compact representation of the systems, especially in comparison to the traditional models based on the global system statespace. We used the language of Linear Temporal Logic (LTL) to express the system properties that the synthesized strategies shall enforce. We proposed alternative semantics for LTL that enables us to express these properties intuitively in our context. We described a naive solution to the stated problem and then we proposed a non-trivial algorithm, which constructs the strategies incrementally. Although the latter algorithm has higher worst case complexity than the naive solution, we argue that in practice, the incremental strategy synthesis works much better than the naive solution. Experimental evaluation of the incremental method is left for future work.

Acknowledgements. This work is supported by the grant SUA 08/034 from the National Research Foundation of South Africa and the Deutsches Zentrum für Luft- und Raumfahrt, as well as by the grant from the Deutsche Forschungsgemeinschaft, Graduiertenkolleg METRIK (GRK 1324/1).

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM 49(5), 672–713 (2002)
2. Calta, J., Shkatov, D., Schlingloff, B.H.: Finding Uniform Strategies for Multi-agent Systems. In: Dix, J., Leite, J., Governatori, G., Jamroga, W. (eds.) CLIMA XI. LNCS, vol. 6245, pp. 135–152. Springer, Heidelberg (2010)
3. Gastin, P., Sznajder, N., Zeitoun, M.: Distributed synthesis for well-connected architectures. Formal Methods in System Design 34(3), 215–237 (2009)
4. Goranko, V., Shkatov, D.: Tableau-based decision procedures for logics of strategic ability in multiagent systems. ACM Trans. Comput. Logic 11(1), 1–51 (2009)
5. Jamroga, W., Ågotnes, T.: Constructive knowledge: what agents can achieve under imperfect information. Journal of Applied Non-classical Logics 17(4), 423–475 (2007)
6. Jamroga, W., Ågotnes, T.: Modular interpreted systems. In: AAMAS 2007: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1–8. ACM, New York (2007)

Reasoning about Joint Action and Coalitional Ability in K_n with Intersection

Thomas Ågotnes[1] and Natasha Alechina[2]

[1] Department of Information Science and Media Studies,
University of Bergen, Norway
thomas.agotnes@infomedia.uib.no
[2] School of Computer Science
University of Nottingham, UK
natasha.alechina@nottingham.ac.uk

Abstract. In this paper we point out that standard PDL-like logics with intersection are useful for reasoning about game structures. In particular, they can express coalitional ability operators known from coalition logic and ATL. An advantage of standard, normal, modal logics is a well understood theoretical foundation and the availability of tools for automated verification and reasoning. We study a minimal variant, multi-modal K with intersection of modalities, interpreted over models corresponding to game structures. There is a restriction: we consider only game structures that are injective. We give a complete axiomatisation of the corresponding models, as well as a characterisation of key complexity problems. We also prove a representation theorem identifying the effectivity functions corresponding to injective games.

1 Introduction

Logics interpreted in *game structures*, enabling automated reasoning and verification of game theoretic properties of multi-agent systems, have received considerable interest in recent years. For computational reasons most such logics are modal logics [17]. One of the most popular approaches is reasoning about *coalitional ability*. Examples of logics in this category include *Coalition Logic* (CL) [14] and *Alternating-time Temporal Logic* (ATL) [2], and many extensions of these, which are interpreted in *game structures*. These logics have *coalition operators* of the form $[C]$ where C is a set of agents (a *coalition*), and $[C]\phi$ means that C can make ϕ true by choosing some joint action. On the other hand, van Benthem [15,16] has pointed out that standard propositional dynamic logic (PDL) [9] is natural for reasoning about games. An advantage of using PDL-like languages is that they are theoretically well understood, with a range of mathematical and computational tools available.

In this paper we point out that standard PDL-like logics with *intersection* are useful for reasoning about games. In particular, they can express coalition operators. We study a minimal variant, multi-modal K with intersection of modalities ($K_n^\cap$), interpreted in game structures, and define an embedding of CL into $K_n^\cap$. $K_n^\cap$ is a fragment of Boolean Modal Logic [6] which has been extensively studied (and implemented) as a variant

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 139–156, 2011.

of propositional dynamic logic with intersection and also by researchers in description logic (see for example [12]).

Although other logics that are normal modal logics and/or have PDL-type operators and can express coalition operators have been studied recently [4,11], these typically have non-standard syntactic operators and/or non-standard semantics (see Section 7). The focus in the current paper is on reasoning about joint action in game structures using a minimal language with the intersection operator.

The main contributions of the paper are, in our view, threefold. First, we give an interpretation of $K_n^\cap$ over models corresponding to game structures and give a sound and complete axiomatisation and characterisations of key complexity problems. There is a snag: this restricted model class does in fact not correspond to *all* game structures, only to *injective* game structures [7], i.e., game structures where two different strategy profiles never lead to the same outcome state. However, we argue that this is a minor limitation:

- Coalition Logic cannot discern between injective and non-injective game structures.
- Injectivity is a very common assumption in game theory. Indeed, the notion of outcome states is often (as in the standard textbook [13]) dispensed with altogether, and preferences defined directly over strategy profiles – implicitly defining an injective game.

Second, we show that the coalition logic operator can be expressed in $K_n^\cap$. Third, we prove a variant of Pauly's *representation theorem* [14,8] for injective games: we characterise the class of effectivity functions that α-correspond to injective games.

We argue that the logic we study is interesting for several reasons. As a normal modal logic it has a well understood theoretical foundation, and it is well supported by tools for automated verification and reasoning as a fragment of standard computer science logics such as PDL. For example, model checking can be done using standard model checkers for PDL (with intersection). In contrast, CL is a non-normal modal logic, with only special purpose tool support (mainly tools developed for ATL) available. Our logic can also be seen as providing an additional set of model-checking and theorem-proving tools for CL. Finally, from a theoretical viewpoint, the logic establishes new connections between coalition logic and normal modal logics.

The paper is organised as follows. We start with introducing CL and $K_n^\cap$. We then, in Section 3, discuss the restriction of CL to injective game structures and prove the representation theorem. In Section 4 we define the interpretation of $K_n^\cap$ in game structures, together with translations from CL, before axiomatisation and complexity are discussed in Sections 5 and 6. We discuss related work and conclude in Section 7. Some of the proofs are found in a technical appendix.

2 Background

2.1 Coalition Logic

We give a brief introduction to Coalition Logic (CL) [14]. Let $N = \{1, \ldots, g\}$ be a finite set of agents, and Θ a set of propositional variables. The language of $\mathcal{L}_{CL}(N, \Theta)$

of CL [14] is defined as follows:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid [C]\phi$$

where $p \in \Theta$ and $C \subseteq N$. Derived propositional connectives are defined as usual. We write $\overline{C}$ for $N \setminus C$ and sometimes abuse notation and write a singleton $\{i\}$ as i. The language can be interpreted over *concurrent game structures* (CGSs) [2]. A CGS over N and Θ is a tuple $M = \langle S, V, Act, d, \delta\rangle$ where

- S is a set of *states*;
- V is a *valuation function*, assigning a set $V(q) \subseteq \Theta$ to each state $q \in S$;
- Act is a set of *actions*;
- For each $i \in N$ and $q \in S$, $d_i(q) \subseteq Act$ is a non-empty set of actions available to agent i in q. $D(q) = d_1(q) \times \cdots \times d_g(q)$ is the set of *full joint actions* in q. When $C \subseteq N$, $D_C(q) = \times_{i\in C} d_i(q)$ is the set of C*-actions* in q. If $\mathbf{a} \in D_C(q)$, a_i ($i \in C$) denotes the action taken from $d_i(q)$.
- δ is a *transition function*, mapping each state $q \in S$ and full joint action $\mathbf{a} \in D(q)$ to a state $\delta(q, \mathbf{a}) \in S$.

Let $\mathcal{M}_{cgs}(N, \Theta, Act)$ be the class of CGSs over N and Θ having Act as the set of actions.

A CGS can be seen as a state-transition system where the edges are labelled with joint actions, but also, alternatively, as an assignment of a *strategic game form* to each state. A *strategic game form* is a tuple $\langle N, \{\Sigma_i : i \in N\}, S, o\rangle$ where Σ_i is the *strategies* for $i \in N$, and $o : \times_{j\in N}\Sigma_j \to S$ is the *outcome function*. We write Σ_C for $\times_{i\in C}\Sigma_i$. When $\sigma_C \in \Sigma_C$, we use $(\sigma_C)_i$ to denote the strategy for i. A CGS can be seen as an assignment of a game $G(s) = \langle N, \{\Sigma_i^s : i \in N\}, S, o^s\rangle$ to each state $s \in S$, where $\Sigma_i^s = d_i(s)$ and $o^s(\mathbf{a}) = \delta(s, \mathbf{a})$.

Intuitively, the CL expression $[C]\phi$ means that a group of agents (*coalition*) C is *effective* for formula ϕ, i.e., that they can ensure that ϕ holds in the next state no matter what the other agents do. Formally, a formula ϕ is interpreted in a state s of CGS M as follows:

$$\begin{array}{l} M, s \models p \Leftrightarrow p \in V(s) \\ M, s \models \neg\phi \Leftrightarrow M, s \not\models \phi \\ M, s \models (\phi_1 \wedge \phi_2) \Leftrightarrow (M, s \models \phi_1 \text{ and } M, s \models \phi_2) \\ M, s \models [C]\psi \Leftrightarrow \\ \quad \exists \mathbf{a}_C \in D_C(s) \forall \mathbf{a}_{\overline{C}} \in D_{\overline{C}}(s),\ M, \delta(s, (a_1, \ldots, a_g)) \models \psi \end{array}$$

As effectivity is the only property of games relevant for the interpretation of coalition logic, we can in fact abstract away all other aspects of game structures. An *effectivity function* [14] over N and a set of states S is a function E that maps any coalition $C \subseteq N$ to a set of sets of states $E(C) \subseteq 2^S$. Given a strategic game form G, the *(α-)effectivity function* E_G of G is defined as follows:

$$X \in E_G(C) \text{ iff } \exists \sigma_C \in \Sigma_C \forall \sigma_{\overline{C}} \in \Sigma_{\overline{C}}, o(\sigma_C, \sigma_{\overline{C}}) \in X.$$

Which effectivity functions are the effectivity functions of strategic game forms? In [14] it is claimed that an effectivity function E is the α-effectivity function of a strategic game form iff E is *playable*:

1. $X \in E(C) \& X \subseteq Y \& Y \subseteq S \Rightarrow Y \in E(C)$ (*outcome monotonicity*);
2. $S \setminus X \notin E(\emptyset) \Rightarrow X \in E(N)$ (*N-maximality*);
3. $\emptyset \notin E(C)$ (*Liveness*);
4. $S \in E(C)$ (*Safety*);
5. $C \cap D = \emptyset \& X \in E(C) \& Y \in E(D) \Rightarrow X \cap Y \in E(C \cup D)$ (*superadditivity*).

However, it has recently been showed [8] that this claim is in fact not correct: there are playable effectivity functions over infinite sets, which are not α-effectivity functions of any strategic game forms. In [8] the result is also corrected: an effectivity function E is said to be *truly playable* iff it is playable and $E(\emptyset)$ has a *complete nonmonotonic core*. The *nonmonotonic core* $E^{nc}(C)$ of $E(C)$, for $C \subseteq N$, is defined as follows: $E^{nc}(C) = \{X \in E(C) : \neg\exists Y(Y \in E(C) \text{ and } Y \subset X)\}$. $E^{nc}(C)$ is *complete* if for every $X \in E(C)$ there exists $Y \in E^{nc}(C)$ such that $Y \subseteq X$. The corrected representation theorem [8] shows that E is the α-effectivity function of a strategic game form iff E is truly playable. A *coalition model* is a tuple $\mathcal{M} = \langle S, E, V\rangle$ where E gives a truly playable effectivity function $E(s)$ for each state $s \in S$, and V is a valuation function. The coalition logic language can alternatively be interpreted in a coalition model, as follows:

$$\mathcal{M}, s \models [C]\phi \text{ iff } \phi^{\mathcal{M}} \in E(s)(C)$$

where $\phi^{\mathcal{M}} = \{t \in S : \mathcal{M}, t \models \phi\}$. It is easy to see that the two semantics coincide: $M, s \models \phi$ iff $\mathcal{M}_\alpha, s \models \phi$ for all ϕ, where $M = (S, V, Act, d, \delta)$ and $\mathcal{M}_\alpha = (S, E_\alpha, V)$ and $E_\alpha(s) = E_{G(s)}$.

2.2 Multi-modal K with Intersection of Modalities

Given a finite set of atomic modalities Π^0 of cardinality n and a countably infinite set of propositional atoms Θ, the formulae $\phi \in \mathcal{L}^{\cap}_K(\Pi^0, \Theta)$ and modalities $\pi \in \Pi$ of multi-modal K with intersection of modalities ($K^{\cap}_n$) are defined as follows:

$$\phi ::= p \in \Theta \mid \neg\phi \mid \phi \wedge \phi \mid [\pi]\phi \qquad \pi ::= a \mid \pi \cap \pi$$

where $a \in \Pi^0$. As usual, $\langle\pi\rangle\phi$ is defined as $\neg[\pi]\neg\phi$, and derived propositional connectives are defined as usual.

A (Kripke) model for the language $\mathcal{L}^{\cap}_K(\Pi^0, \Theta)$ is a tuple $M = \langle S, V, \{R_\pi : \pi \in \Pi\}\rangle$ where

- S is a set of *states*;
- $V : \Theta \to 2^S$ is a *valuation function*;
- For each $\pi \in \Pi$, $R_\pi \subseteq S \times S$
- $R_{\pi_1 \cap \pi_2} = R_{\pi_1} \cap R_{\pi_2}$ (INT)

The interpretation of a formula in a state of a model is defined as follows (other clauses as usual):

$$M, s \models [\pi]\phi \text{ iff } \forall(s, s') \in R_\pi, M, s' \models \phi$$

3 Injective Games

The idea of interpreting $K_n^\cap$ in game structures is very simple: interpret a full joint action $\langle a_1, \ldots, a_g \rangle$ in a CGS as a set of g different ("atomic") edges, one for each agent-action combination. This gives us a $K_n^\cap$ model, where the atomic modalities are agent-action pairs. Full joint actions can be recovered by taking the intersection between the relations for two or more atomic modalities for different agents.

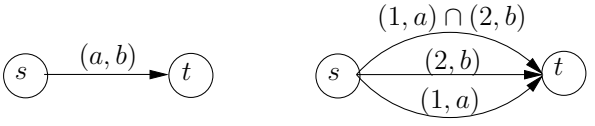


Fig. 1. CGS (left) and $K_n^\cap$ model (right)

For example, consider the CGS on the left in Figure 1. The corresponding $K_n^\cap$ model is shown to the right. Coalition operators can now be captured approximately as follows. If we for example want to say that there exists a joint action by agents 1 and 2, all executions of which result in the outcome p, in such a $K_n^\cap$ model, we can say something like $\bigvee_{a,b \in Act}[(1, a) \cap (2, b)]p$ (in addition we must check that the actions a and b are actually available in the current state, but that is straightforward).

However, there is a problem with this approach. Consider the CGS to the left in Figure 2. The approach above gives us the $K_n^\cap$ model to the right in the figure. This model has four atomic transitions from s to t: two labelled $(1, a_1)$ and $(2, b_1)$ which correspond to the full joint action (a_1, b_1), and two labelled $(1, a_2)$ and $(2, b_2)$ which correspond to (a_2, b_2). The full joint actions can be recovered by intersection of the atomic transitions, but the problem is that too much is "recovered" in this way: we get the spurious transitions (a_1, b_2) and (a_2, b_1) which are not present between these states in the original model.

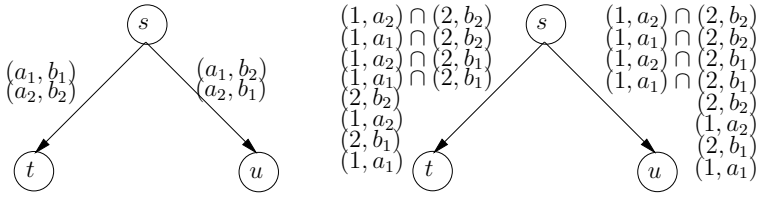


Fig. 2. CGS (left) and $K_n^\cap$ model (right). An arrow with more than one label represents a transition for each label.

The problem is that by decomposing a full joint action into individual actions, we lose information about which *combinations* of actions relate the two states. That information is crucial, e.g., for the interpretation of coalition operators. We call a CGS

without two or more different full joint actions between the same two states, i.e., with an injective δ, *injective* (following [7]). Injective CGSs do not suffer this problem.

It is relatively straightforward to see that any CGS is equivalent, in the sense of satisfying the same coalition logic formulae, to an injective CGS: take the *tree-unfolding* of the model (see [1] for relevant definitions of tree-unfoldings, bisimulations, and invariance under bisimulation, for the ATL language which contains the CL language). The tree-unfolding, however, is a model with infinitely many states, which may be a problem, e.g., if we want to do model checking. Fortunately it turns out that every finite CGS (finite state space) is CL-equivalent to a finite, and even "small", injective CGS. The following theorem follows immediately from a result by Goranko [7, Proposition 12] (with some minor changes and amendments).

Theorem 1. *For every* CGS $M = \langle S, V, Act, d, \delta \rangle$ *there is an injective* CGS M' *with states* S' *such that* $S \subseteq S'$ *and for all* CL *formulae* ϕ *and states* $s \in S$, $M, s \models \phi$ *iff* $M', s \models \phi$. *Moreover, if* M *is finite, then* $|S'| \leq |S| + |\delta|$.

This makes it possible to translate a CGS into a $K_n^\cap$ model, such that we can recover a CGS that is CL-equivalent to the former from the latter. Before formally defining the translation in Section 4 we take a closer look at injective games; the reader mainly interested in the translation can skip directly to that section.

3.1 Effectivity Functions and Representation

Although coalition logic cannot discern between injective games and non-injective games, there is still another pertinent question if we want to restrict our attention to injective games: the question of representation using effectivity functions. Which truly playable effectivity functions correspond to injective games? The answer is not necessarily "all": this is similar to the relationship between playable and truly playable effectivity functions [8]; the latter is a proper subset of the former while coalition logic cannot discern between the two. Indeed, not all truly playable effectivity functions are the α-effectivity functions of injective games:

Example 1. Let $N = \{1, 2\}$ and E be defined as follows:

$$E(\emptyset) = E(1) = E(2) = \{\{s, t\}\} \quad E(\{1, 2\}) = \{\{s\}, \{t\}, \{s, t\}\}$$

(where $s \neq t$). The reader can check that E is truly playable. However, it is not the α-effectivity function of an injective game. For assume it is, that $E = E_G$ for some G.. Because of Safety, the game has exactly two states s and t. Together with the fact that $\{s\}, \{t\} \in E(\{1, 2\})$, that means that one of the agents must have exactly one strategy, and the other exactly two: all other combinations violates injectivity of a two-state game. Wlog. assume that $\Sigma_1 = \{\sigma_1\}$ and $\Sigma_2 = \{\sigma'_1, \sigma'_2\}$. $\{s\} \in E_G(\{1, 2\})$ implies that $o(\sigma_1, \sigma'_1) = s$ or $o(\sigma_1, \sigma'_2) = s$; wlog. assume the former. Then $\{t\} \in E_G(\{1, 2\})$ implies that $o(\sigma_1, \sigma'_2) = t$. But that means that $\{s\}, \{t\} \in E_G(2)$, which is not the case.

We now state and prove a representation theorem (Theorem 2) for injective games. An effectivity function is *injectively playable* iff it is playable and (for all C, i, j, X, Y):

$$E(C) \text{ has a complete nonmonotonic core} \tag{1}$$

$$E^{nc}(C) = \{\bigcap_{i \in C} X_i : X_i \in E^{nc}(i)\} \quad C \neq \emptyset \tag{2}$$

$$X, Y \in E^{nc}(i) \text{ and } X \neq Y \Rightarrow X \cap Y = \emptyset \tag{3}$$

$$X \in E^{nc}(j) \text{ and } x \in X \Rightarrow \exists Y \in E^{nc}(i), x \in Y \tag{4}$$

Injective playability extends the true playability requirement of a complete nonmonotonic core [8] from the empty coalition to *all* coalitions. As a result, $E(C)$ is completely determined by its nonmonotonic core (stated formally in the following Lemma). In addition, there are some restrictions on the structure of the core. None of the additional properties of injective playability, (1)–(4), hold in general for truly playable effectivity functions (in particular, true playability does not imply complete nonmonotonic core for non-empty coalitions).

Lemma 1. *Let E be an outcome monotonic effectivity function. $E(C)$ has a complete nonmonotonic core iff $E(C) = \{X : Y \subseteq X, Y \in E^{nc}(C)\}$.*

Lemma 2. *If E is injectively playable, then:*

$$(\forall_{i \in N} X_i \in E^{nc}(i)) \Rightarrow |\bigcap_{i \in N} X_i| = 1 \tag{5}$$

$$E^{nc}(\emptyset) = \{Z\} \textit{ where } Z = \bigcup E^{nc}(N) \tag{6}$$

Before proving the main result (Th. 2) we need the following lemma.

Lemma 3. *If E_G is the α-effectivity function of an injective game $G = (N, \{\Sigma_i : i \in N\}, o, S)$, then for all $C \subseteq N$:*

$$E_G^{nc}(C) = \{\ \{o(\sigma_C, \sigma_{\overline{C}}) : \sigma_{\overline{C}} \in \Sigma_{\overline{C}}\} : \sigma_C \in \Sigma_C\}$$

Theorem 2. *An effectivity function E is injectively playable iff it is the α-effectivity function of some injective game G.*

Proofs are found in the appendix.

4 Multi-modal K with Intersection for Games

We now show how $K_n^{\cap}$ formulae can be interpreted in game structures, by identifying a class of $K_n^{\cap}$ models corresponding to (injective) game structures.

4.1 Joint Action Models

Let Act be a finite set of actions and N a set of g agents. Define a set of atomic modalities as follows:

$$\Pi^0_{Act\,N} = N \times Act$$

Intuitively, an atomic modality is a pair (i, a) intended to represent an action and the agent that executes that action. We will call an atomic modality in Π^0_{ActN} an *individual action*, and a composite modality $\pi = \pi_1 \cap \pi_2$ a *joint action*. Since the intersection operation is associative, we can write any joint action π as an intersection of a set of individual actions: $\pi = (i_1, a_1) \cap \cdots \cap (i_k, a_k)$. Joint actions of the form $(1, a_1) \cap \ldots \cap (g, a_g)$ with one individual action for *every* agent in N will be called *complete (joint) actions*.

The following are some properties of $K^\cap_n$ models over Π^0_{ActN} that will be of particular interest. We say that an action $a \in Act$ is *enabled* for agent i in a state s iff there is a state s' such that $(s, s') \in R_{(i,a)}$.

Seriality (SER). For any state s and agent i, at least one action is enabled in s for i.

Independent Choice (IC). For any state s, agents $C = \{i_1, \ldots, i_k\}$ and actions $a_1, \ldots, a_k \in Act$, if for every j a_j is enabled for i_j in s, then there is a state s' such that $(s, s') \in R_{(i_1,a_1)\cap\cdots\cap(i_k,a_k)}$.

Deterministic Joint Actions (DJA). For any complete joint action α and states s, s_1, s_2, $(s, s_1), (s, s_2) \in R_\alpha$ implies that $s_1 = s_2$.

Unique Joint Actions (UJA). For any complete joint actions α and β and states s, t, if $(s, t) \in R_\alpha \cap R_\beta$ then $\alpha = \beta$.

A $K^\cap_n$ model over Π^0_{ActN} (where Act is finite) is a *joint action model* if it satisfies the properties SER, IC, DJA, and UJA.

Given (finite) Act and N, we now translate any CGS over Act and N into a joint action model.

Definition 1. *Given an injective* CGS $M = \langle S, V, Act, d, \delta\rangle \in \mathcal{M}_{cgs}(N, \Theta, Act)$ *where Act is finite, the corresponding joint action model $\hat{M} = \langle S, V, \{R_\pi : \pi \in \Pi\}\rangle$ over Θ and Π^0_{ActN} is defined as follows:*

- $R_{(i,a)} = \{(s, s') : \exists \mathbf{a} \in D(s)$ *s.t.* $a_i = a$ *and* $s' = \delta(s, \mathbf{a})\}$, *when* $(i, a) \in \Pi^0_{ActN}$
- $R_{\pi_1\cap\pi_2} = R_{\pi_1} \cap R_{\pi_2}$

We use the following property to show that M^N is indeed a joint action model.

Lemma 4. *Let $M = \langle S, V, Act, d, \delta\rangle$ be injective and $\hat{M} = \langle S, V, \{R_\pi : \pi \in \Pi\}\rangle$ be the corresponding joint action model, and let $\pi = (i_1, a_1) \cap \ldots \cap (i_k, a_k)$. Then $(s, t) \in R_\pi$ iff there is an $\mathbf{a}' \in D(s)$ such that $a'_{i_j} = a_j$ for all $1 \leq j \leq k$ and $\delta(s, \mathbf{a}') = t$.*

Proof. $(s, t) \in R_\pi$ iff there are $\mathbf{a}^1, \ldots, \mathbf{a}^k \in D(s)$ such that for all $1 \leq j \leq k$: $a^j_{i_j} = a_j$ and $t = \delta(s, \mathbf{a}^j)$. Since M is injective, it must be the case that $\mathbf{a}^1 = \cdots = \mathbf{a}^k$.

Lemma 5. *$\hat{M}$ is a joint action model.*

Proof. $\hat{M}$ is a proper $\mathcal{L}^\cap_K(\Pi^0_{ActN}, \Theta)$ model by definition. SER holds because $d_i(s)$ is always non-empty. IC holds because $D(s)$ is defined as the cross product of $d_i(s)$ for all i. For DJA, $(s, s_1), (s, s_2) \in R_\alpha$ implies that $\delta(s, \mathbf{a}) = s_1 = s_2$ by Lemma 4. For UJA, $(s, t) \in R_\alpha \cap R_\beta$ implies that $\delta(s, \mathbf{a}) = t$ and $\delta(s, \mathbf{b}) = t$ by Lemma 4, which by injectivity implies that $\mathbf{a} = \mathbf{b}$ which again implies that $\alpha = \beta$.

Thus, we get a direct interpretation of formulae $\phi \in \mathcal{L}^\cap_K(\Pi^0_{ActN}, \Theta)$ in injective CGSs $M \in \mathcal{M}_{cgs}(N, \Theta, Act)$: $M, s \models \phi$ iff $\hat{M}, s \models \phi$.

4.2 Embedding of CL

Given a coalition logic formula $\phi \in \mathcal{L}_{CL}(N, \Theta)$ and a finite set of actions Act, we define the translation $\phi' \in \mathcal{L}_K^{\cap}(\Pi^0_{ActN}, \Theta)$ as follows:

$$\begin{array}{ll} p' & \equiv p \\ (\neg\phi)' & \equiv \neg\phi' \\ (\phi_1 \wedge \phi_2)' & \equiv \phi_1' \wedge \phi_2' \\ ([\{i_1, \ldots, i_k\}]\phi)' & \equiv \bigvee_{a_1,\ldots,a_k \in Act} \bigwedge_{1 \leq j \leq k} \langle (i_j, a_j) \rangle \top \\ & \wedge [(i_1, a_1) \cap \ldots \cap (i_k, a_k)]\phi' \end{array}$$

The translation of the CL formula $[C]\phi$ says that there is an action for each agent in C, such that (i) the actions are enabled and (ii) for all possible states resulting from executing the actions at the same time (the translation of) ϕ holds. The translation assumes that the set of possible actions Act is given and that it is finite. In model checking this can be obtained directly from the model.

Theorem 3. *Let Θ, N and Act (finite) be fixed. For any $\phi \in \mathcal{L}_{CL}(N, \Theta)$ and any injective* CGS *$M \in \mathcal{M}_{cgs}(N, \Theta, Act)$ and any state s in M:*

$$M, s \models \phi \textit{ iff } \hat{M}, s \models \phi'$$

Proof. Let Θ, N and Act (finite) be given, $\phi \in \mathcal{L}_{CL}(N, \Theta)$ and $M \in \mathcal{M}_{cgs}(N, \Theta, Act)$ be injective and s in M. The proof is by induction on the structure of ϕ. The only interesting case is when $\phi = [C]\psi$ where $C = \{i_1, \ldots, i_k\}$. By Lemma 4 $\hat{M}, s \models ([C]\psi)'$ iff there is a $\langle a_1, \ldots, a_k \rangle \in D_C(s)$ such that for all $(s,t) \in R_{(i_1,a_1)\cap\ldots(i_k,a_k)}$ $\hat{M}, t \models \psi'$ iff there is a $\langle a_1, \ldots, a_k \rangle \in D_C(s)$ such that for all $\mathbf{a}' \in D(s)$ with $\mathbf{a}'_{i_j} = a_j$ $(1 \leq j \leq k)$ $\hat{M}, \delta(s, \mathbf{a}') \models \psi'$ iff by the induction hypothesis there is a $\langle a_1, \ldots, a_k \rangle \in D_C(s)$ such that for all $\mathbf{a}' \in D(s)$ with $\mathbf{a}'_{i_j} = a_j$ $(1 \leq j \leq k)$ $M, \delta(s, \mathbf{a}') \models \psi$ iff $M, s \models [\{i_1, \ldots, i_k\}]\psi$.

The following follows immediately from Theorems 3 and 1.

Corollary 1. *Let Θ, N and Act (finite) be fixed. For any $\phi \in \mathcal{L}_{CL}(N, \Theta)$ and any* CGS *$M \in \mathcal{M}_{cgs}(N, \Theta, Act)$ there is a joint action model $\overline{M}$ over Π^0_{ActN} such that for any state s in M:*

$$M, s \models \phi \textit{ iff } \overline{M}, s \models \phi'$$

5 Axiomatisation of Joint Action Models

We now give an axiomatisation for the language $\mathcal{L}_K^{\cap}(\Pi^0_{ActN}, \Theta)$ and prove that it is sound and complete with respect to the class of all joint action models over Π^0_{ActN} and Θ. The axiom system $\mathcal{S}$ is defined as follows:

K $[\pi](\phi \to \psi) \to ([\pi]\phi \to [\pi]\psi)$
A1 $\bigvee_{a \in Act} \langle (i, a) \rangle \top$
A2 $\langle \pi \rangle \phi \to \bigvee_{a \in Act} \langle \pi \cap (i, a) \rangle \phi$

A3 $\bigwedge_{i \in N} \langle (i, a_i) \rangle \top \rightarrow \langle (1, a_1) \cap \ldots \cap (g, a_g) \rangle \top$
A4 $\langle (1, a_1) \cap \cdots \cap (g, a_g) \rangle \phi \rightarrow [(1, a_1) \cap \ldots \cap (g, a_g)] \phi$
A5 $[\pi]\phi \rightarrow [\pi \cap \pi']\phi$
A6 $[(i, a) \cap (i, b)] \bot$ when $a \neq b$
MP From $\phi \rightarrow \psi$ and ϕ infer ψ
G From ϕ infer $[\pi]\phi$

K, **MP** and **G** says that the $[\pi]$ modalities are normal. **A1** says that at least one action is enabled for each agent in every state, **A2** says that if there is a joint action for some agents that can ensure ϕ, then any agent can do some action at the same time such that ϕ is still ensured, **A3** says that all joint actions composed of enabled individual actions are enabled, **A4** says that complete joint actions are deterministic, **A5** is the standard axiom for intersection, and **A6** says that an agent cannot do two actions simultaneously.

Theorem 4. *The axiom system $\mathcal{S}$ is sound and complete wrt. all joint action models.*

Proof. Soundness is straightforward. To prove completeness, we introduce some conventions and auxiliary concepts and show some intermediate properties. When $\pi = (i_1, a_1) \cap \cdots \cap (i_k, a_k)$ and $\pi' = (j_1, b_1) \cap \cdots \cap (j_l, b_l)$ are joint actions, we write $\pi \leq \pi'$ to denote that for every $1 \leq u \leq k$ there is a $1 \leq v \leq l$ such that $i_u = j_v$ and $a_u = b_v$. Recall that a complete joint action is an expression of the form $(1, a_1) \cap \cdots \cap (g, a_g)$ where $a_i \in Act$, giving one action for each agent in the system. Let JA denote the (finite) set of complete joint actions. We use $\alpha, \beta, \ldots$ to denote complete joint actions.

We will make use of a notion of *pseudomodels*, which only have transition relations for complete joint actions. Formally, a pseudomodel is a tuple $(S, \{R_\alpha : \alpha \in JA\}, V)$ where: S is a set of states, $R_\alpha \subseteq S \times S$ for each $\alpha \in JA$ and $V : \Theta \rightarrow 2^S$.

First, we construct the *canonical* pseudomodel $M^c = (S^c, \{R^c_\alpha : \alpha \in JA\}, V^c)$ as follows:

- S^c is the set of $\mathcal{L}^{\cap}_K(\Pi^0_{ActN}, \Theta)$-maximal $\mathcal{S}$-consistent sets Γ
- $V^c(p) = \{\Gamma : p \in \Gamma\}$
- $R^c_\alpha \Gamma \Gamma'$ iff for any ψ, if $\psi \in \Gamma'$ then $\langle \alpha \rangle \psi \in \Gamma$

Lemma 6 (Existence Lemma). *For any $s \in S^c$, if $\langle \alpha \rangle \gamma \in s$ for some $\alpha \in JA$, then there is an s' such that $(s, s') \in R^c_\alpha$ and $\gamma \in s'$.*

The proof of the existence lemma is as in standard normal modal logic.

Now let ϕ be a consistent formula; we show that it is satisfied in some joint action model. Let $x \in S^c$ be such that $\phi \in x$. We now take the *unravelling* of the canonical pseudomodel around x. The pseudomodel $M^x = (S^x, \{R^x_\alpha : \alpha \in JA\}, V^x)$ is defined as follows:

- S^x is the set of all finite sequences $(s_0, R^c_{\alpha_0}, s_1 \ldots, R^c_{\alpha_{k-1}}, s_k)$ such that $s_0, \ldots, s_k \in S^c$, $s_0 = x$, and $(s_i, s_{i+1}) \in R^c_{\alpha_i}$ for all $0 \leq i \leq k-1$.
- $(s, u) \in R^x_\alpha$ iff $s = (x, R^c_{\alpha_0}, s_1 \ldots, R^c_{\alpha_{k-1}}, s_k)$ and $u = (x, R^c_{\alpha_0}, s_1 \ldots, R^c_{\alpha_{k-1}}, s_k, R^c_\alpha, s_{k+1})$ for some $s_{k+1} \in S^c$.
- $V^x(p) = \{(x, R^c_{\alpha_0}, s_1 \ldots, R^c_{\alpha_{k-1}}, s_k) : s_k \in V^c(p)\}$

We now transform the pseudomodel M^x into a (proper) model $M = (S^x, \{R_\pi : \pi \in \Pi\}, V^x)$ as follows:

- $R_{(i,a)} = \bigcup_{\alpha \in JA, (i,a) \leq \alpha} R^x_\alpha$
- $R_{\pi_1 \cap \pi_2} = R_{\pi_1} \cap R_{\pi_2}$

Lemma 7. *$R_\alpha = R^x_\alpha$, for any complete joint action $\alpha \in JA$.*

Proof. Let $\alpha = (1, a_1) \cap \cdots \cap (g, a_g)$, and observe that

$$R_\alpha = (\bigcup_{(1,a_1) \leq \alpha'} R^x_{\alpha'}) \cap \cdots \cap (\bigcup_{(g,a_g) \leq \alpha'} R^x_{\alpha'}).$$

First, assume that $(s,t) \in R^x_\alpha$. It follows immediately that $(s,t) \in \bigcup_{(i,a_i) \subseteq \alpha'} R^x_{\alpha'}$ for all i by taking $\alpha' = \alpha$. Second, assume that $(s,t) \in R_\alpha$, i.e., that there are $\alpha_1, \ldots, \alpha_g$ such that $(i, a_i) \leq \alpha_i$ for all i and $(s,t) \in R^x_{\alpha_1} \cap \cdots \cap R^x_{\alpha_g}$. By construction of M^x, that implies that $\alpha_1 = \cdots = \alpha_g$. So $\alpha_1 = \cdots = \alpha_g = \alpha$.

Let $last(s)$ denote the last element $s_k \in S^c$ in a sequence $s \in S^x$.

Lemma 8 (Truth Lemma). *For any s and ψ, $M, s \models \psi$ iff $\psi \in last(s)$.*

Proof. The proof is by induction on the structure of ψ. The cases for propositional atoms and Boolean connectives are straightforward, so let $\psi = \langle \pi \rangle \gamma$. Let $\pi = (i_1, a_1) \cap \cdots \cap (i_k, a_k)$ for some $1 \leq k \leq n$.

First assume that $i_l = i_m = i$ and $a_l \neq a_m$ for some $l \neq m$. Observe that $R_\pi = (\bigcup_{(i_1,a_1) \leq \alpha'} R^x_{\alpha'}) \cap \cdots \cap (\bigcup_{(i_k,a_k) \leq \alpha'} R^x_{\alpha'})$. Let α_1 and α_2 be arbitrary complete joint actions such that $(i_l, a_l) \leq \alpha_1$ and $(i_m, a_m) \leq \alpha_2$. $\alpha_1 \neq \alpha_2$ since $i_l = i_m$ and $a_l \neq a_m$. By construction of M^x, $R^x_{\alpha_1} \cap R^x_{\alpha_2} = \emptyset$. Thus, $R_\pi = \emptyset$ (since α_1 and α_2 were arbitrary), and $M, s \not\models \langle \pi \rangle \gamma$ for any γ. On the other hand, by **A6** $\neg \langle (i, a_l) \cap (i, a_m) \rangle \gamma \in last(s)$ for any γ, and it follows by **A5** that $\neg \langle \pi \rangle \gamma \in last(s)$. Thus, the Lemma holds in this case and we henceforth assume that $a_l = a_m$ whenever $i_l = i_m$.

For the implication to the right, assume that $M, s \models \langle \pi \rangle \gamma$, i.e., that $M, t \models \gamma$ for some $(s,t) \in R_\pi = \bigcup_{(i_1,a_1) \leq \alpha'} R^x_{\alpha'} \cap \cdots \cap \bigcup_{(i_k,a_k) \leq \alpha'} R^x_{\alpha'}$. Thus, for each $1 \leq j \leq k$, there is an α_j such that $(s,t) \in R^x_{\alpha_j}$ and $(i_j, a_j) \in \alpha_j$. By construction of M^x, $\alpha_1 = \cdots = \alpha_k = \alpha$ (the state t has only one "incoming" transition). Thus, $(s,t) \in R^x_\alpha$ with $\pi \leq \alpha$. Assume that $s = (x, R^c_{\alpha_0}, s_1, \ldots, R^c_{\alpha_{k-1}}, s_k)$. Then $t = (x, R^c_{\alpha_0}, s_1, \ldots, R^c_{\alpha_{k-1}}, s_k, R^c_\alpha, s_{k+1})$ for some s_{k+1} such that $(s_k, s_{k+1}) \in R^c_\alpha$. By the induction hypothesis $\gamma \in last(t) = s_{k+1}$. By construction of R^c_α, $\langle \alpha \rangle \gamma \in s_k = last(s)$. By (repeated applications of) **A5**, $\langle \pi \rangle \gamma \in s_k = last(s)$.

For the implication to the left, let $s = (x, R^c_{\alpha_0}, s_1, \ldots, R^c_{\alpha_{k-1}}, s_k)$ and let $\langle \pi \rangle \gamma \in last(s) = s_k$. Let $X = \{l_0, \ldots, l_m\}$ be the agents not mentioned in π, i.e., $X = \{l \in N : l \neq i_j, 1 \leq j \leq n\}$. Let $\pi_0 = \pi$, and $\pi_{j+1} = \pi_j \cap (l_j, a_j)$ for each $0 \leq j \leq m$ where $a_j \in Act$ is such that $\langle \pi_j \cap (l_j, a_j) \rangle \gamma \in s_k$. The existence of such a_js are ensured by axiom **A2**. Finally, let $\alpha = \pi_{m+1}$. This construction together with the assumption that $a_l = a_m$ whenever $i_k = i_m$ ensures that α is a complete joint action. By the fact that $\langle \alpha \rangle \gamma \in s_k$ and the existence lemma, there is a state $s_{k+1} \in S^c$ such that $(s_k, s_{k+1}) \in R^c_\alpha$ and $\gamma \in s_{k+1}$. Let $t = (x, R^c_{\alpha_0}, s_1, \ldots, R^c_{\alpha_{k-1}}, s_k, R^c_\alpha, s_{k+1})$; $t \in S^x$

and $(s,t) \in R^x_\alpha$ by definition of M^x. By Lemma 7 $(s,t) \in R_\alpha$, and from the fact that $\pi \leq \alpha$ it is easy to see that $R_\alpha \subseteq R_\pi$ by definition of R_π, so $(s,t) \in R_\pi$. Since $\gamma \in last(t)$, by the induction hypothesis $M,t \models \gamma$, and thus $M,s \models \langle\pi\rangle\gamma$.

Lemma 9. *M is a joint action model.*

Proof. INT: Immediate from the definition.

SER: Let s be a state and i an agent. From **A1** and **A3** there is some $\alpha = (1,a_1) \cap \cdots \cap (g,a_g)$ such that $\langle\alpha\rangle\top \in s$. From the truth lemma and Lemma 7 there is a s' such that $(s,s') \in R_\alpha = R^x_\alpha$. From the definition of $R_{(i,a_i)}$, $R^x_\alpha \subseteq R_{(i,a_i)}$, and thus $(s,s') \in R_{(i,a_i)}$.

IC: Let s be a state, $C = \{i_1,\ldots,i_k\}$ a coalition, and assume that for each j, (i_j,a_{i_j}) is enabled for i_j in s. Let $\pi = (i_1,a_{i_1}) \cap \cdots \cap (i_k,a_{i_k})$. By the truth lemma, $\bigwedge_{i_j \in C}\langle(i_j,a_{i_j})\rangle\top \in last(s)$. For each $i_j \in N \setminus C$, let a_{i_j} be such that $\langle(i_j,a_{i_j})\rangle\top \in last(s)$ – existing by **A1**. Let $\alpha = (1,a_1) \cap \cdots \cap (g,a_g)$. By **A3** $\langle\alpha\rangle\top \in last(s)$, and by **A5** $\langle\pi\rangle\top \in last(s)$. By the truth lemma, there is an s' such that $(s,s') \in R_\pi$.

DJA: Assume that $s = (x, R^c_{\alpha_0}, s_1, \ldots, R^c_{\alpha_{k-1}}, s_k)$. Let $(s,s_1),(s,s_2) \in R_\alpha$, where α is complete. By Lemma 7, there are $s^1_{k+1}, s^2_{k+1} \in S^c$ such that $s_1 = (x, R^c_{\alpha_0}, s_1, \ldots, R^c_{\alpha_{k-1}}, s_k, R^c_\alpha, s^1_{k+1})$, $s_2 = (x, R^c_{\alpha_0}, s_1, \ldots, R^c_{\alpha_{k-1}}, s_k, R^c_\alpha, s^2_{k+1})$, and $(s_k,s^1_{k+1}),(s_k,s^2_{k+1}) \in R^c_\alpha$. Assume that $s_1 \neq s_2$, i.e., since s_1 and s_2 are identical up to the last state, that $s^1_{k+1} \neq s^2_{k+1}$. By the definition of S^c, there must be a formula $\psi \in s^1_{k+1}$ such that $\neg\psi \in s^2_{k+1}$. By the truth lemma, $M,s_1 \models \psi$ and $M,s_2 \models \neg\psi$ and thus $M,s \models \langle\alpha\rangle\psi \wedge \langle\alpha\rangle\neg\psi$. By the truth lemma again, $\langle\alpha\rangle\psi, \langle\alpha\rangle\neg\psi \in last(s)$. By **A4**, $[\alpha]\psi, [\alpha]\neg\psi \in last(s)$. But $\langle\alpha\rangle\psi, [\alpha]\neg\psi \in last(s)$ contradicts, via standard modal reasoning, the fact that $last(s)$ is consistent. Thus, $s_1 = s_2$.

UJA: Immediate by Lemma 7 and construction of M^x.

Since $\phi \in x = last(x)$, ϕ is satisfied in a joint action model by Lemma 8 and 9. This concludes the completeness proof.

6 Complexity

We show that the complexity of deciding satisfiability in joint action models of $\mathcal{L}^{\cap}_K(\Pi^0_{ActN}, \Theta)$ formulae is in PSPACE. The proof uses ideas from [12].

In what follows, we assume wlog. that formulas do not contain diamond modalities and disjunctions. Given a set of formulas X, we use $Cl(X)$ to denote the smallest set containing all subformulas of formulas in X such that:

(a) for each agent i and action a, $[(i,a)]\bot \in Cl(X)$
(b) for every complete joint action $\alpha \in JA$, $[\alpha]\bot \in Cl(X)$
(c) if $\neg[(1,a_1),\ldots,(g,a_g)]\psi \in Cl(X)$, then $[(1,a_1),\ldots,(g,a_g)] \sim \psi \in Cl(X)$, where $\sim\psi = \neg\psi$ if ψ is not of the form $\neg\chi$, and $\sim\psi = \chi$ otherwise
(d) for each i and $a \neq b$, $[(i,a) \cap (i,b)]\bot \in Cl(X)$
(e) if $\psi \in Cl(X)$, then $\sim\psi \in Cl(X)$

The following procedure Tab is based on the $K^{\cap\cup}_\omega$-World proc. of [12]. For sets of formulas Δ and S where S is closed as above, $Tab(\Delta,S)$ returns true iff

(A) Δ is a maximally propositionally consistent subset of S, that is, for each $\neg\psi \in S$, $\psi \in \Delta$ iff $\neg\psi \notin \Delta$ and for each $\psi_1 \wedge \psi_2 \in S$, $\psi_1 \wedge \psi_2 \in \Delta$ iff $\psi_1 \in \Delta$ and $\psi_2 \in \Delta$.
(B) There is a partition of the set $\{\neg[\pi]\psi : \neg[\pi]\psi \in \Delta\}$ into sets W_α (at most one for each $\alpha \in JA$) such that if $\neg[\pi]\psi \in W_\alpha$ then $\pi \leq \alpha$ and
  (i) $\neg\psi \in \Delta_\alpha$
  (ii) for each π' and ψ', if $[\pi']\psi' \in \Delta$ and $\pi' \leq \alpha$, then $\psi' \in \Delta_\alpha$
  (iii) $Tab(\Delta_\alpha, S')$ returns true, where $S' = Cl(\{\psi' : [\pi']\psi' \in \Delta \text{ and } \pi' \leq \alpha\} \cup \{\neg\psi : \neg[\pi]\psi \in W_\alpha\})$
(C) for each $i \in N$, $\neg[(i,a)]\bot \in \Delta$ for some $a \in Act$
(D) if $\neg[(1,a_1)]\bot, \ldots, \neg[(g,a_g)]\bot \in \Delta$, then $\neg[(1,a_1) \cap \ldots \cap (g,a_g)]\bot \in \Delta$
(E) if $\neg[\alpha]\psi \in \Delta$, then $[\alpha] \sim \psi \in \Delta$
(F) for every $i \in N$ and $a, b \in Act$ such that $a \neq b$, $[(i,a) \cap (i,b)]\bot \in \Delta$

We require $Tab(\Delta, S)$ to terminate when the only modal formulas in S are those introduced by the clauses (a), (b) and (d) of the definition of $Cl(X)$. Note that otherwise formulas of the form $\neg[\alpha]\bot$ will continue triggering new calls to $Tab(\Delta, S)$.

Lemma 10. *A formula ϕ is satisfiable in a joint action model iff there exists $\Delta \subseteq Cl(\phi)$ with $\phi \in \Delta$ such that $Tab(\Delta, Cl(\phi))$ returns true.*

Proof. One direction is easy. For the other direction, we will show how to construct a model for ϕ if there exists $\Delta \subseteq Cl(\phi)$ with $\phi \in \Delta$ such that $Tab(\Delta, Cl(\phi))$ returns true. Suppose for ϕ such Δ exists, and let us call it Δ_0. The model M and state s_0 satisfying ϕ are constructed as follows. Each Δ in successive recursive calls of $Tab(\Delta, S)$ corresponds to a (partial specification of a) state. The existence of propositional assignment satisfying formulas in Δ is ensured by clause (A). The initial state s_0 corresponds to Δ_0. In each Δ, each formula of the form $\neg[\pi]\psi$ by the clause (B) belongs to a set W_α and has a 'witness' Δ_α for $\neg\psi$ accessible by a complete joint action α such that $\pi \leq \alpha$ and $\neg\psi \in \Delta_\alpha$. In the model we stipulate $R_\alpha(s_\Delta, s_{\Delta_\alpha})$ holds together with $R_{\pi'}(s_\Delta, s_{\Delta_\alpha})$ for every $\pi' \leq \alpha$. The rest of clause (B) makes sure that Δ_α contains all formulas ψ' such that $[\pi']\psi' \in \Delta$, which makes sure both that the truth definition for $[\pi]$ and the semantics of intersection work as expected. This part is almost identical to the proof for $K_\omega^\cap$ (apart from requiring a unique α-successor). All we need to prove is that in addition, the resulting model satisfies the properties of seriality, independent choice, determinism for complete joint actions, and uniqueness of joint actions.

SER is trivial by clause (C). When we terminate the procedure, to ensure seriality we add one more successor state for each α with an α loop to itself. This modification will not affect the truth of ϕ in s_0 because it is at a modal distance from s_0 which is greater than the modal depth of ϕ. IC is ensured by clause (D). DJA is ensured by (B); the existence of partition is enabled by (E) which makes the set of formulas $\sim \psi$ for $\neg[\alpha]\psi \in \Delta$ consistent. UJA is ensured by (F); namely there is no Δ' accessible by $\alpha \cap \alpha'$ from Δ, where $\alpha, \alpha' \in JA$ and $\alpha \neq \alpha'$; otherwise by clause (F) for some agent i which performs a different action in α and α', $[(i,a) \cap (i,b)]\bot \in \Delta$ and hence $\bot \in \Delta'$, but by the definition of the procedure then it cannot return true for Δ'.

Theorem 5. *The complexity of satisfiability problem of formulas in joint action models is PSPACE-complete.*

Proof. Satisfiability is decided by $Tab(\Delta, Cl(\phi))$ by the previous lemma. To see that $Tab(\Delta, Cl(\phi))$ requires polynomial space, consider the size of $Cl(\phi)$. The set of subformulas of ϕ is clearly polynomial in $|\phi|$. The number of formulas added to $Cl(\phi)$ by clause (a) is gm, the number of formulas added by clause (b) is m^g, the number of formulas added by clause (d) is gm^2, and (c) and (e) at most double the number of formulas in $Cl(\phi)$. Note that g and m are constant factors, hence the size of $Cl(\phi)$ and Δ is polynomial in $|\phi|$. PSPACE-hardness follows from K being PSPACE-complete.

The following is an immediate consequence of the result for model checking complexity of PDL with intersection [10]:

Theorem 6. *Model checking the $\mathcal{L}^{\cap}_K(\Pi^0_{ActN}, \Theta)$ language in joint action models is in PTIME.*

The complexity results above are encouraging from the point of view of using the logic of joint actions for verifying properties of game structures using standard theorem-proving and model-checking tools for normal modal logic. However, verification of properties involving coalitional ability comes at the price of performing a translation from CL to the language of $K^{\cap}_n$. The size of the translation may grow exponentially in the size of the input formula (nested coalition modalities give rise to nested disjunctions over all possible actions).

7 Discussion

In this paper we defined and studied a class of $K^{\cap}_n$ models corresponding to the class of concurrent game structures that are (1) injective and (2) parameterised by a fixed and finite set of actions, and showed that on this model class coalition modalities can be expressed in the $K^{\cap}_n$ language. Along the way we proved a representation theorem for injective games (this result holds also games with infinite sets of actions).

As mentioned in the introduction, the idea of interpreting PDL-like languages in games is not new. However, we are not aware of existing completeness or complexity results for K_n with intersection interpreted in game structures, nor on using intersection to capture the coalition operator.

[15] uses propositional dynamic logic (PDL) interpreted directly in extensive-form games, and also suggests extending the language with a "forcing" operator $\{G, i\}\phi$, with the meaning that agent i has a strategy in game G which forces a set of outcomes that all will satisfy ϕ. However, the forcing operator is not defined in terms of intersection, and the operator is only defined for singleton coalitions. [4] have already shown that coalition logic can be embedded in a normal modal logic, namely in a variant of STIT (*seeing-to-it-that*) logic [3]. While this is a valuable result for several reasons, we argue that embedding in $K^{\cap}_n$ is of additional interest because the latter is a more standard logic (see the introduction). A closely related work is [11], which sets out from a similar starting point as the current paper: defining a "minimalistic" logical framework based on PDL that is interpreted in models where agents perform joint actions. *Deterministic Dynamic Logic of Agency* ($\mathcal{DDLA}$) [11] has modalities of the form $\langle i : a \rangle$ where i is an agent and a is an action, very similar to the modalities in the current paper

in other words, and is shown to embed coalition logic. The interpretation of the modalities is slightly different: $\langle i : a \rangle \phi$ informally means that "i performs action a and ϕ holds afterwards". The formal interpretation is not a standard PDL interpretation. Also the language is not standard PDL; it includes a modality $\diamond$ that quantifies over actions. The language does not use intersection. In contrast, the current paper has focused on reasoning about joint action using only standard PDL modalities and operators, in particular intersection. We leave the precise relationship between the two logics to future work.

In this paper we studied a "minimal" language with intersection, sufficient to capture the coalition operators. For future work, extensions of the language with other PDL operators would be of interest, building on existing results on PDL with intersection such as [5].

Acknowledgments

We thank the anonymous reviewers for comments that helped us improve the paper. We also thank Pål Grønås Drange who commented on a draft of the paper.

References

1. Ågotnes, T., Goranko, V., Jamroga, W.: Alternating-time temporal logics with irrevocable strategies. In: Samet, D. (ed.) Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK XI), June 2007, pp. 15–24. Presses Universitaires de Louvain, Brussels, Belgium (2007)
2. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM 49, 672–713 (2002)
3. Belnap, N., Perloff, M.: Seeing to it that: a canonical form for agentives. Theoria 54, 175–199 (1988)
4. Broersen, J., Herzig, A., Troquard, N.: A normal simulation of coalition logic and an epistemic extension. In: Samet, D. (ed.) Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2007), Brussels, Belgium, June 25-27, pp. 92–101 (2007)
5. Danecki, S.: Nondeterministic propositional dynamic logic with intersection is decidable. In: Skowron, A. (ed.) SCT 1984. LNCS, vol. 208, pp. 34–53. Springer, Heidelberg (1985)
6. Gargov, G., Passy, S.: A note on boolean modal logic. In: Proc. of The Summer School and Conf. on Mathematical Logic "Heyting 1988", pp. 311–321. Plenum Press, New York (1988)
7. Goranko, V.: Coalition games and alternating temporal logics. In: Proceeding of the Eighth Conference on Theoretical Aspects of Rationality and Knowledge (TARK VIII), pp. 259–272. Morgan Kaufmann, San Francisco (2001)
8. Goranko, V., Jamroga, W., Turrini, P.: Strategic games and truly playable effectivity functions. In: Tumer, Yolum, Sonenberg, Stone (eds.) Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, pp. 727–734 (2011)
9. Harel, D.: Dynamic logic. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic. Synthese Library, vol. 165, pp. 497–604. D. Reidel Publishing Co., Dordrecht (1984)

10. Lange, M.: Model checking propositional dynamic logic with all extras. J. Applied Logic 4(1), 39–49 (2006)
11. Lorini, E.: A dynamic logic of agency II: Deterministic DLA, coalition logic, and game theory. Journal of Logic, Language and Information 19, 327–351 (2010)
12. Lutz, C., Sattler, U.: The complexity of reasoning with boolean modal logics. In: Wolter, F., Wansing, H., de Rijke, M., Zakharyaschev, M. (eds.) Advances in Modal Logic, vol. 3, pp. 329–348. World Scientific, Singapore (2002)
13. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. The MIT Press, Cambridge (1994)
14. Pauly, M.: A modal logic for coalitional power in games. Journal of Logic and Computation 12(1), 149–166 (2002)
15. van Benthem, J.: Games in dynamic-epistemic logic. Bulletin of Economic Research 53(4), 219–248 (2001)
16. van Benthem, J.: Extensive games as process models. J. of Logic, Lang. and Inf. 11, 289–313 (2002)
17. van der Hoek, W., Pauly, M.: Modal logic for games and information. In: van Benthem, J., Blackburn, P., Wolter, F. (eds.) The Handbook of Modal Logic, pp. 1152–1180. Elsevier, Amsterdam (2006)

Appendix: Some Proofs

Proof (Lemma 1). For the implication to the right, assume that $E(C)$ has a complete nonmonotonic core. $E(C) \subseteq \{X : Y \subseteq X, Y \in E^{nc}(C)\}$ is immediate. If $Y \subseteq X$ and $Y \in E^{nc}(C)$, then $X \in E(C)$ by outcome monotonicity. The implication to the left is immediate.

Proof (Lemma 2). (5) Let $X_i \in E^{nc}(i)$ for each $i \in N$. By (2), $\bigcap_{i\in N} X_i \in E^{nc}(N)$, and by true playability and [8, Proposition 5] there is an $x \in \bigcap_{i\in N} X_i$ such that $\{x\} \in E(N)$. That means that $\{x\} = \bigcap_{i\in N} X_i$; because $\bigcap_{i\in N} X_i \neq \{x\}$ contradicts the facts that $\bigcap_{i\in N} X_i \in E^{nc}(N)$, $\{x\} \in E(N)$ and $x \in \bigcap_{i\in N} X_i$.

(6) Let $Z = \bigcup E^{nc}(N)$. Since injective playability implies true playability, we know that $E^{nc}(\emptyset) = \{Z'\}$ for some Z' [8, Proposition 5]. We show that $Z' = Z$. We have that $S \setminus Z \notin E(N)$; otherwise there would be a $X \subseteq S \setminus Z$ such that $X \in E^{nc}(N)$ (by (1)) and thus $X \subseteq Z$ by definition of Z, which together with the fact that $X \neq \emptyset$ (Liveness) is a contradiction. By N-maximality, $S \setminus (S \setminus Z) = Z \in E(\emptyset)$. Thus, $Z' \subseteq Z$ by (1). Assume, towards a contradiction, that $Z \not\subseteq Z'$, i.e., that there is an $x \in Z$ such that $x \notin Z'$. That $x \in Z$ means that there is an $X \in E^{nc}(N)$ with $x \in X$. Let $X' = X \setminus \{x\}$. That $Z' \in E(\emptyset)$ and $X \in E(N)$ implies by superadditivity that $Z' \cap X \in E(N)$, and by the fact that $x \notin Z'$ we have that $Z' \cap X \subseteq X'$. By outcome monotonicity, $X' \in E(N)$. But that contradicts the fact that $X \in E^{nc}(N)$. Thus, $Z = Z'$.

Proof (Lemma 3). $X \in E^{nc}_G(C)$ iff $\exists \sigma_C \forall \sigma_{\overline{C}} o(\sigma_C, \sigma_{\overline{C}}) \in X$ and there is no $Y \in E_G(C)$ such that $Y \subset X$. Let $P = \{ \{o(\sigma_C, \sigma_{\overline{C}}) : \sigma_{\overline{C}} \in \Sigma_{\overline{C}}\} : \sigma_C \in \Sigma_C\}$.

First, let $X \in E^{nc}_G(C)$ and let σ_C be as above (a witness for X). Let $Y = \{o(\sigma_C, \sigma_{\overline{C}}) : \sigma_{\overline{C}} \in \Sigma_{\overline{C}}\}$. $Y \in P$. $Y \subseteq X$. We have that $Y \in E_G(C)$ (by definition of α-effectivity), so by the fact that $X \in E^{nc}_G(C)$ it follows that $Y \not\subset X$ and thus that $Y = X$.

Second, let $X = \{o(\sigma_C, \sigma_{\overline{C}}) : \sigma_{\overline{C}} \in \Sigma_{\overline{C}}\} \in P$ for some σ_C. $X \in E_G(C)$. Assume towards a contradiction that there is a $Y \in E_G(C)$ such that $Y \subset X$. Thus there is a $\sigma'_C \in \Sigma_C$ such that for all $\sigma'_{\overline{C}} \in \Sigma_{\overline{C}}$, $o(\sigma'_C, \sigma'_{\overline{C}}) \in Y$. $\sigma_C \neq \sigma'_C$; otherwise $X \subseteq Y$, a contradiction. $Y \neq \emptyset$, so there is a $y \in Y \cap X$. In other words, there are $\sigma_{\overline{C}}$ and $\sigma'_{\overline{C}}$ such that $o(\sigma_C, \sigma_{\overline{C}}) = o(\sigma'_C, \sigma'_{\overline{C}}) = y$. But this contradicts the fact that G is injective, since $\sigma_C \neq \sigma'_C$. Thus, there is no such Y, and $X \in E^{nc}_G(C)$.

Proof (Theorem 2). First, let E_G be the α-effectivity function of some injective game G. We show that E_G is injectively playable. It is immediate from [14] that E_G is playable.

In order to show (1), let $X \in E_G(C)$, i.e., there is a σ_C such that for all $\sigma_{\overline{C}}$ $o(\sigma_C, \sigma_{\overline{C}}) \in X$. Let $Y = \{o(\sigma_C, \sigma_{\overline{C}}) : \sigma_{\overline{C}} \in \Sigma_{\overline{C}}\}$. $Y \subseteq X$, and $Y \in E^{nc}_G(C)$ by Lemma 3.

In order to show (2), assume that $|C| \geq 2$, since (2) holds trivially for $|C| = 1$. For one direction, let $X \in E^{nc}_G(C)$. By Lemma 3, $X = \{o(\sigma_C, \sigma_{\overline{C}}) : \sigma_{\overline{C}} \in \Sigma_{\overline{C}}\}$ for some σ_C. Let, for each $i \in C$, $\sigma_i = (\sigma_C)_i$ and $X_i = \{o(\sigma_i, \sigma_{\overline{i}}) : \sigma_{\overline{i}} \in \Sigma_{\overline{i}}\}$. $X_i \in E^{nc}_G(i)$ by Lemma 3. $X \subseteq \bigcap_{i \in C} X_i$. We must show that $\bigcap_{i \in C} X_i \subseteq X$. Let $x \in \bigcap_{i \in C} X_i$. For each $i \in C$, there exists some $\sigma_{\overline{i}}$ such that $x = o(\sigma_i, \sigma_{\overline{i}})$. For any arbitrary $i, j \in C$, $i \neq j$, from $o(\sigma_j, \sigma_{\overline{j}}) = x = o(\sigma_i, \sigma_{\overline{i}})$ we get that $(\sigma_{\overline{j}})_i = \sigma_i$ by injectivity. Thus, $o(\sigma_C, \sigma_{\overline{C}}) = o(\sigma_j, \sigma_{\overline{j}})$, for all $j \in C$ and some $\sigma_{\overline{C}}$, and thus $x \in X$.

For the other direction of (2), let $X = \bigcap_{i \in C} X_i$ with $X_i \in E^{nc}_G(i)$. Again, for each $i \in C$, $X_i = \{o(\sigma_i, \sigma_{\overline{i}}) : \sigma_{\overline{i}} \in \Sigma_{\overline{i}}\}$ for some σ_i. Let σ_C be defined by $(\sigma_C)_i = \sigma_i$. Let $Y = \{o(\sigma_C, \sigma_{\overline{C}}) : \sigma_{\overline{C}} \in \Sigma_{\overline{C}}\}$. $Y \in E^{nc}_G(C)$ by Lemma 3. We show that $Y = X$. First, let $\sigma_{\overline{C}} \in \Sigma_{\overline{C}}$ be arbitrary. Since $(\sigma_C)_i = \sigma_i$, $o(\sigma_C, \sigma_{\overline{C}}) \in X_i$ for all $i \in C$, and $o(\sigma_C, \sigma_{\overline{C}}) \in X$. Thus, $Y \subseteq X$. Let $x \in X$. For each $i \in C$, there is some $\sigma_{\overline{i}}$ such that $o(\sigma_i, \sigma_{\overline{i}}) = x$. We can now reason as above. Let $i, j \in C$, $i \neq j$. From $o(\sigma_j, \sigma_{\overline{j}}) = x = o(\sigma_i, \sigma_{\overline{i}})$, we get that $(\sigma_{\overline{j}})_i = \sigma_i$ by injectivity. Thus, $o(\sigma_j, \sigma_{\overline{j}}) = o(\sigma_C, \sigma_{\overline{C}})$, for some arbitrary $j \in C$ and some $\sigma_{\overline{C}}$, and thus $x \in Y$. Thus, $X \subseteq Y$.

In order to show that (3) holds, let $X \neq Y \in E^{nc}_G(i)$. By Lemma 3, there are σ_i, σ'_i such that $X = \{o(\sigma_i, \sigma_{\overline{i}}) : \sigma_{\overline{i}} \in \Sigma_{\overline{i}}\}$ and $Y = \{o(\sigma'_i, \sigma_{\overline{i}}) : \sigma_{\overline{i}} \in \Sigma_{\overline{i}}\}$. Assume that $x \in X \cap Y$, i.e., that $o(\sigma_i, \sigma_{\overline{i}}) = o(\sigma'_i, \sigma'_{\overline{i}})$ for some $\sigma_{\overline{i}}$ and $\sigma'_{\overline{i}}$. Since the game is injective that means that $\sigma_i = \sigma'_i$, but that contradicts the fact that $X \neq Y$. Thus, $X \cap Y = \emptyset$.

In order to show that (4) holds, let $X \in E^{nc}_G(j)$ and $x \in X$. By Lemma 3, there is a σ_j such that $X = \{o(\sigma_j, \sigma_{\overline{j}}) : \sigma_{\overline{j}} \in \Sigma_{\overline{j}}\}$. In particular, $x = o(\sigma_j, \sigma_{\overline{j}})$ for some $\sigma_{\overline{j}}$. Let $\sigma_i = (\sigma_{\overline{j}})_i$, and let $Y = \{o(\sigma_i, \sigma_{\overline{i}}) : \sigma_{\overline{i}} \in \Sigma_{\overline{i}}\}$. $x \in Y$, and $Y \in E^{nc}_G(i)$ by Lemma 3.

Second, let E be an injectively playable effectivity function over N and S. We construct a game $G = (N, \{\Sigma_i : i \in N\}, o, S)$ as follows:

$$\Sigma_i = E^{nc}(i) \qquad o(X_1, \ldots, X_g) = x \quad \text{where} \quad \{x\} = \bigcap_{i \in N} X_i$$

The property (5) (Lemma 2) ensures that the game is well defined. To see that G is injective, assume that $o(X_1, \ldots, X_g) = o(X'_1, \ldots, X'_g) = x$. That means that, for each i, $x \in X_i \cap X'_i$, and by (3) it follows that $X_i = X'_i$. Thus, G is injective.

We must show that $E_G = E$. By (1), outcome monotonicity and Lemma 1 it suffices to show that $E^{nc}_G(C) = E^{nc}(C)$ for all $C \subseteq N$.

First assume that $C \neq \emptyset$. For any X, $X \in E^{nc}_G(C)$ iff (by Lemma 3) $\exists \sigma_C$ such that $X = \{o(\sigma_C, \sigma_{\overline{C}}) : \sigma_{\overline{C}} \in \Sigma_{\overline{C}}\}$ iff $\exists \{X_i \in E^{nc}(i) : i \in C\}$ such that $X = \{x : \{x\} = \bigcap_{i \in N} X_i, X_j \in E^{nc}(j), j \in N \setminus C\}$. On the other hand, $X \in E^{nc}(C)$ iff $\exists \{X_i \in E^{nc}(i) : i \in C\}$ such that $X = \bigcap_{i \in C} X_i$, by (2). Thus, let, for each $i \in C$, $X_i \in E^{nc}(i)$. It suffices to show that $\{x : \{x\} = \bigcap_{i \in N} X_i, X_j \in E^{nc}(j), j \in N \setminus C\} = \bigcap_{i \in C} X_i$. For inclusion towards the left, assume that $x \in \bigcap_{i \in C} X_i$. If $C = N$ we are done. Otherwise, from (4) it follows that there is a $X_j \in E^{nc}(j)$ such that $x \in X_j$, for every $j \in N \setminus C$. Thus, $x \in \bigcap_{i \in N} X_i$. For inclusion towards the right, let $\{x\} = \bigcap_{i \in N} X_i$ for some $\{X_i \in E^{nc}(i) : i \in N \setminus C\}$. It immediately follows that $x \in \bigcap_{i \in C} X_i$.

Second, consider the case that $C = \emptyset$. $X \in E^{nc}_G(\emptyset)$ iff (by Lemma 3) $X = \{o(\sigma_N) : \sigma_N \in \Sigma_N\}$ iff $X = \{x : \{x\} = \bigcap_{i \in N} X_i, X_i \in E^{nc}(i)\}$ iff (by (2)) $X = \bigcup E^{nc}(N)$ iff (by Lemma 2) $X \in E^{nc}(\emptyset)$.

Ontology Merging as Social Choice

Daniele Porello and Ulle Endriss

Institute for Logic, Language and Computation (ILLC)
University of Amsterdam

Abstract. The problem of merging several ontologies has important applications in the Semantic Web, medical ontology engineering, and other domains where information from several distinct sources needs to be integrated in a coherent manner. We propose to treat ontology merging as a problem of social choice, i.e., as a problem of aggregating the input of a set of individuals into an adequate collective decision, and we show how to apply the methodology of social choice theory in this new domain. We do this for the case of ontologies that are modelled using description logics. Specifically, we formulate a number of desirable properties for ontology merging procedures, we identify the incompatibility of some of these properties, and we define and analyse several concrete procedures.

1 Introduction

Merging a number of ontologies originating from different sources is a pressing problem in applications ranging from medical informatics to the Semantic Web [13,6]. We propose to add a new perspective to this problem by treating it as a problem of *social choice*. Social choice theory (SCT) is a branch of economic theory that deals with the design and analysis of mechanisms for aggregating opinions of individual agents to arrive at a basis for a collective decision [7]. A typical example is voting. In the context of ontology merging, we may think of the provider of each ontology as a voter, and these voters try to "elect" a collective ontology that adequately and fairly represents the information provided by each of them.

As an example, imagine a possible Semantic Web scenario. Suppose several sources provide different encyclopedia entries of the same word. Naturally, encyclopedias might differ with respect to the information provided, the degree of exhaustiveness attained, or the aspects chosen as relevant. Of course, there might be conflicts about the views provided by the different sources. We might imagine an agent who is searching the web for a given definition who is interested in knowing an answer that best represents the *class* of encyclopedias he has access to, rather than checking each source by itself. This problem is clearly related to the problem of aggregating several points of view into a collective point of view, where we do not have enough information to discriminate the reliability of the various sources. With respect to such a scenario, the kind of axioms usually discussed in SCT are relevant, because they provide precise definitions of the idea of collective information.

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 157–170, 2011.

Our aim in this paper is to make the idea of viewing ontology merging as a problem of social choice precise by providing a suitable formal framework for its analysis and to propose a number of simple procedures that fit this framework, together with an initial analysis of some of their most fundamental properties. We concentrate on high-level properties that are broadly related to "fairness" and we restrict attention to what one might want to call "coarse" merging: the ontology to be constructed will be a list of some of the formulas included in the individual ontologies. We do not deal with "fine" merging, where we might also want to construct entirely new formulas from those provided by the individuals. We use ontologies expressed in a simple description logic [1] as an example, although the choice of logic is in fact not critical to our proposal.

In the remainder of this paper we shall use the term *ontology aggregation* to refer to our specific approach based on SCT, to distinguish it from the broader and established research area of *ontology merging*.

What we propose is closely related to *judgment aggregation* (JA), a branch of SCT that deals with the aggregation of individual judgments regarding the truth or falsehood of a set of interrelated propositions modelled as formulas of propositional logic [10]. The main points of interest of our proposal from the viewpoint of the JA literature are the following:

(1) First, the agenda, i.e., the set of formulas which may or may not be accepted by individuals, is not closed under complementation (which is a standard assumption in JA).
(2) Second, we operate under an *open world assumption*, meaning that an agent's failure to explicitly include a formula in her ontology does not necessarily mean that she rejects the truth of that formula.
(3) Third, description logical ontologies make a separation between terminological and assertional knowledge, and this conceptual distinction can guide the aggregation process (cf. the discussion of "premises" and "conclusions" in the JA literature).

The problem of modelling ontology change is of course a very general and protean task, dealing with a vast number of interrelated phenomena such as updating after new information has arrived, revision, or debugging for inconsistencies [6]. Contributions to ontology merging range from sophisticated engineering solutions (see e.g. [13]) to works in mathematical logic. Applications of AGM belief revision to ontology merging and debugging have been discussed, for instance, by [14]. However, even though the connections between SCT and belief merging are clearly recognised in AI [8], this methodology seems not yet to have been applied to ontology merging.

The remainder of the paper is organised as follows. In Section 2, we define a formal framework for ontology aggregation in description logics. In Section 3, we then define a number of axioms (i.e., desirable properties) that a specific aggregation procedure may or may not satisfy. Finally, in Section 4, we present a number of such procedures based on simple principles and discuss to what extent they satisfy the axioms defined earlier. We conclude with a brief discussion of possible directions for future work.

2 A Framework for Ontology Aggregation

We first define our framework for aggregating ontologies expressed in a description logic with a common alphabet. We begin by recalling some basic notation and terminology from description logics.

2.1 Preliminaries: Description Logics

Description logics are languages for knowledge representation with a formal syntax and semantics that balance expressive power as dictated by applications with computational efficiency requirements. The best known and mostly widely used basic description logic is $\mathcal{ALC}$. Our approach is not tied to any particular description logic, but for reasons of clarity of exposition we shall restrict attention to $\mathcal{ALC}$. The following review of the basics of description logics and $\mathcal{ALC}$ is fairly succinct; for full details we refer to the literature [1].

The language of $\mathcal{ALC}$ is based on an alphabet consisting of *atomic concepts*, *role names*, and *object names*. The set of *concept descriptions* is generated by the following grammar (where A represents atomic concepts and R role names):

$$C ::= A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C$$

A *TBox* is a finite set of formulas of the form $A \sqsubseteq C$ and $A \equiv C$ (where A is an atomic concept and C a concept description). It is used to store terminological knowledge regarding the relationships between concepts. An *ABox* is a finite set of formulas of the form $A(a)$ ("object a is an instance of concept A") and $R(a, b)$ ("objects a and b stand to each other in the R-relation").[1] It is used to store assertional knowledge regarding specific objects. The semantics of $\mathcal{ALC}$ is defined in terms of *interpretations* that map each object name to an element of its domain, each atomic concept to a subset of the domain, and each role name to a binary relation on the domain. The truth of a formula in such an interpretation is defined in the usual manner [1]. For instance, $\forall R.C$ is true in a given interpretation at point a if all elements related to a via (the interpretation of) R belong to the (interpretation of) C. A set of (TBox and ABox) formulas is *satisfiable* if there exists an interpretation in which they are all true. A consequence relation $\models$ is defined on top of this semantics in the standard way.

2.2 Ontology Aggregators

Let us now fix a particular alphabet. This induces a fixed finite set of ABox formulas (but the set of TBox formulas is infinite). Let us fix a finite set $\mathcal{L}$ of $\mathcal{ALC}$ formulas over this alphabet that includes all ABox formulas that can be expressed.[2] We call $\mathcal{L}$ the *agenda* and any set $O \subseteq \mathcal{L}$ an *ontology*.[3] Any such

[1] Note that limiting the ABox to "atomic" formulas is not a restriction, as A may be given a complex definition in the TBox.

[2] The finite set of TBox formulas in $\mathcal{L}$ might be all TBox formulas of a certain maximum length or the union of all TBox formulas that a given population of agents choose to include in their TBoxes.

[3] In the literature, the term "ontology" is sometimes restricted to terminological knowledge; here we use it in this broader sense.

ontology O can be divided into a TBox O^T and an ABox O^A. We denote the set of all those ontologies that are *satisfiable* by $\mathrm{On}(\mathcal{L})$. Also recall that the *closure* of a set of formulas $\Phi \subseteq \mathcal{L}$ is the set of all formulas that logically follow from those in Φ. It is denoted by $\mathrm{Cl}(\Phi) := \{\varphi \in \mathcal{L} \mid \Phi \models \varphi\}$.

Let $\mathcal{N} = \{1, \ldots, n\}$ be a finite set of *agents* (or *individuals*, or *experts*). Each agent $i \in \mathcal{N}$ provides a satisfiable ontology $O_i \in \mathrm{On}(\mathcal{L})$. An *ontology profile* $\boldsymbol{O} = (O_1, \ldots, O_n) \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$ is a vector of such ontologies, one for each agent. We write $\mathcal{N}_\varphi^{\boldsymbol{O}} := \{i \in \mathcal{N} \mid \varphi \in O_i\}$ for the set of agents including φ in their ontology under profile $\boldsymbol{O}$.

The question we shall address in this paper is how to best aggregate an ontology profile into a single collective ontology. That is, our object of study are *ontology aggregators*.

Definition 1 (Ontology aggregators). *An ontology aggregator is a function* $F : \mathrm{On}(\mathcal{L})^{\mathcal{N}} \to 2^{\mathcal{L}}$ *mapping any profile of satisfiable ontologies to an ontology.*

Observe that, according to this definition, the ontology we obtain as the outcome of an aggregation process need not be satisfiable. Of course, we will be particularly interested in ontology aggregators that are *satisfiable*, i.e., aggregators F for which $F(O_1, \ldots, O_n)$ is satisfiable whenever all O_i are.

2.3 Example

A simplistic example for an ontology aggregator is F with $F(\boldsymbol{O}) := O_1 \cup \cdots \cup O_n$, which simply returns the union of the individuals ontologies. Of course, this will usually not be a good choice, as F clearly is not a satisfiable aggregator. Another simple natural choice is the *majority rule:* accept a formula if and only if a majority of the agents do. This can also lead to unsatisfiable outcomes, as we can easily simulate the *doctrinal paradox* familiar from JA [10]. Suppose three agents share a common TBox with two formulas:

$$C_3 \sqsubseteq C_1 \sqcap C_2 \qquad C_4 \sqsubseteq \neg C_3$$

Furthermore, suppose the three ABoxes are as follows:

	$C_1(a)$	$C_2(a)$	$C_3(a)$	$C_4(a)$
Agent 1	yes	yes	yes	no
Agent 2	yes	no	no	yes
Agent 3	no	yes	no	yes
Majority	yes	yes	no	yes

Even though all individual ontologies are satisfiable, the ontology obtained by applying the majority rule is not.

3 Properties of Ontology Aggregators

We now define a number of properties that a given ontology aggregator may or may not satisfy. Most of these properties relate, in one way or another, to the

"fairness" of the aggregation process and are directly inspired by properties of voting rules, JA rules, and other types of aggregators commonly defined in SCT [7,10]. As in SCT, we refer to these properties as *axioms*.

3.1 Syntactic Axioms

We first define a number of axioms that are "syntactic" in the sense that they relate to the formulas that occur explicitly in the ontologies of individual agents or in the collective ontology. We will later contrast this with "semantic" axioms that also make reference to the formulas that can be *inferred* from those ontologies.

The axiom of *unanimity* postulates then when all individual ontologies include φ, then so should the collective ontology. This clearly is a desirable property in any kind of domain. An aggregator F is *anonymous* if it is symmetric wrt. individual ontologies. This is appropriate if we have reasons to treat all agents equally. In the social choice literature the axiom of anonymity is usually motivated in terms of fairness considerations, which may or may not be relevant in the context of ontology aggregation, depending on the application at hand. But treating all agents equally is also justified, for instance, if we simply do not have any information regarding the reliability of individual agents. F is *independent* if inclusion of φ in the collective ontology only depends on the pattern of its inclusion in the individual ontologies and is independent from which other formulas may or may not have been included. Independence is a more demanding axiom that the previous two; whether or not it should be imposed certainly is debatable. Finally, F is *monotonic* if additional support for a collectively accepted formula will never lead to it being rejected. This, again, is a property that we would usually (though maybe not always) like to see satisfied, certainly in cases where it is reasonable to assume that every agent has at least some degree of reliability. The four axioms introduced so far are formalised as follows:

- **Unanimity:** F is called *unanimous* if $O_1 \cap \cdots \cap O_n \subseteq F(\boldsymbol{O})$ for every profile $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$.
- **Anonymity:** F is called *anonymous* if for any profile $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$ and any permutation $\pi : \mathcal{N} \to \mathcal{N}$ we have that $F(O_1, \ldots, O_n) = F(O_{\pi(1)}, \ldots, O_{\pi(n)})$.
- **Independence:** F is called *independent* if for any $\varphi \in \mathcal{L}$ and profiles $\boldsymbol{O}, \boldsymbol{O'} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$, we have that $\varphi \in O_i \Leftrightarrow \varphi \in O'_i$ for all $i \in \mathcal{N}$ implies $\varphi \in F(\boldsymbol{O}) \Leftrightarrow \varphi \in F(\boldsymbol{O'})$.
- **Monotonicity:** F is called *monotonic* if for any $i \in \mathcal{N}$, $\varphi \in \mathcal{L}$, and $\boldsymbol{O}, \boldsymbol{O'} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$ with $O_j{=}O'_j$ for all $j \neq i$, we have that $\varphi \in O'_i \setminus O_i$ and $\varphi \in F(\boldsymbol{O})$ imply $\varphi \in F(\boldsymbol{O'})$.

A further important axiom from the literature is *neutrality*, which, intuitively, requires all formulas to be treated symmetrically. In fact, there are a number of possible interpretations of this notion, including these:

- **Neutrality:** F is called *neutral* if for any $\varphi, \psi \in \mathcal{L}$ and $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$ we have that $\varphi \in O_i \Leftrightarrow \psi \in O_i$ for all $i \in \mathcal{N}$ implies $\varphi \in F(\boldsymbol{O}) \Leftrightarrow \psi \in F(\boldsymbol{O})$.

– **Acceptance-Rejection Neutrality:** F is called *acceptance-rejection neutral* if for any $\varphi \in \mathcal{L}$ and $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$ we have that $\varphi \in O_i \Leftrightarrow \psi \notin O_i$ for all $i \in \mathcal{N}$ implies $\varphi \in F(\boldsymbol{O}) \Leftrightarrow \psi \notin F(\boldsymbol{O})$.

The first notion of neutrality is the one that we shall adopt here. It says that if two formulas enjoy the same pattern of acceptance—in the same profile—then either both should be accepted or both should be rejected. The second axiom is closer to the original neutrality axiom in voting theory proposed by [11]. It says that if those patterns of acceptance are complementary, then exactly one of the two formulas should be accepted. The reason we do not believe that acceptance-rejection neutrality is appropriate for ontology aggregation is that it makes the implicit assumption that not explicitly including a formula into one's knowledge base amounts to actively rejecting the validity of that formula. This is an appropriate assumption in JA, but not here.[4]

We also propose three axioms that are specific to ontology aggregation and that do not have a counterpart in standard SCT or JA. The first is *groundedness:* a formula should only occur in the collective ontology if it is included in at least one of the individual ontologies, i.e., if it is an element of $O_1 \cup \cdots \cup O_n$, the *support* of a given profile $(O_1, \ldots, O_n)$. In standard JA, due to the assumption that agendas are closed under complementation (and that each agent will accept either φ or its complement), groundedness is implied by unanimity (with consistency) and does not require a separate axiom. The second axiom we propose is *exhaustiveness:* it should not be possible to add any formula from the support to the collective ontology without rendering the latter unsatisfiable. In other words, we should "exhaust" the supply of formulas in the support when building the collective ontology—as long as we do not create any inconsistencies this way. This axiom is desirable if we assume that all information supplied by individuals is (potentially) useful information and if we do not take an agent's omission of a particular formula in their ontology as a vote *against* that formula. That is, exhaustiveness is closely related to the open world assumption. Our third axiom is *group closure*, a weaker version of exhaustiveness: any formula in the support that is logically entailed by the collective ontology should in fact be part of that ontology. We now state these additional axioms formally:

– **Groundedness:** F is called *grounded* if $F(\boldsymbol{O}) \subseteq O_1 \cup \cdots \cup O_n$ for every profile $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$.
– **Exhaustiveness:** F is called *exhaustive* if there exists no satisfiable set $\Phi \subseteq O_1 \cup \cdots \cup O_n$ with $F(\boldsymbol{O}) \subset \Phi$ for any profile $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$.
– **Group Closure:** F is called *group-closed* if there exists no set $\Phi \subseteq O_1 \cup \cdots \cup O_n$ with $F(\boldsymbol{O}) \models \Phi$ and $F(\boldsymbol{O}) \subset \Phi$ for any profile $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$.

[4] Dietrich and List [3] use the name "acceptance-rejection neutrality" for a slightly different axiom: for any $\varphi \in \mathcal{L}$ and $\boldsymbol{O}, \boldsymbol{O}' \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$, we have that $\varphi \in O_i \Leftrightarrow \psi \notin O'_i$ for all $i \in \mathcal{N}$ implies $\varphi \in F(\boldsymbol{O}) \Leftrightarrow \psi \notin F(\boldsymbol{O}')$. Arguably, this is closer to an (in)dependence axiom, as it makes reference to *two* profiles.

All of the above axioms are natural requirements, but we stress that we do *not* impose them in general. Some may be more desirable than others for any given problem domain (but all should certainly be considered).

We are now in a position to make our objection to the axiom of acceptance-rejection neutrality more precise:

Proposition 1. *Any ontology aggregator that satisfies acceptance-rejection neutrality violates exhaustiveness.*

Proof. Any acceptance-rejection neutral aggregator cannot accept both φ and ψ when $\varphi \in O_i \Leftrightarrow \psi \notin O_i$ for all $i \in \mathcal{N}$. But if each is accepted by at least one agent, and if each is logically independent from all other formulas in the support, then an exhaustive aggregator must accept them both. □

3.2 Semantic Axioms

For many applications, the agents providing individual ontologies will not only be worried about the formulas included in the collective ontology but also about the formulas that can be *inferred* from that ontology. This distinction has also been discussed by Flouris et al. [5] in terms of implicitly and explicitly represented knowledge. We therefore formulate semantic variants of the axioms above in which we refer to the closures of ontologies rather than the ontologies themselves. Note that the existing literature on JA only deals with what we have called syntactic axioms above.

Here we only spell out the precise formulation of the semantic variants of the aforementioned axioms for some of them. The remaining ones can be adapted following the same pattern.

- **Semantic Unanimity:** F is called *semantically unanimous* if $\mathrm{Cl}(O_1) \cap \cdots \cap \mathrm{Cl}(O_n) \subseteq \mathrm{Cl}(F(\boldsymbol{O}))$ for every profile $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$.
- **Semantic Groundedness:** F is called *semantically grounded* if $\mathrm{Cl}(F(\boldsymbol{O})) \subseteq \mathrm{Cl}(O_1) \cup \cdots \cup \mathrm{Cl}(O_n)$ for every $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$.
- **Semantic Exhaustiveness:** F is called *semantically exhaustive* if there exists no satisfiable set $\Phi \subseteq \mathrm{Cl}(O_1) \cup \cdots \cup \mathrm{Cl}(O_n)$ with $\mathrm{Cl}(F(\boldsymbol{O})) \subset \Phi$ for any $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$.

That is, semantic unanimity, for instance, is satisfied if whenever each individual ontology suffices to infer some formula φ, then φ should also be derivable from the collective ontology. We believe that all of our semantic properties are generally desirable properties and system designers should be interested in satisfying these axioms—with one exception: semantic groundedness. This axiom postulates that only formulas derivable from at least one individual ontology should be derivable. This will rarely be a reasonable requirement. On the contrary, we would hope that by combining the information provided by several agents we are able to make new inferences that were not possible before aggregation. For comparison, note that *syntactic* groundedness is perfectly reasonable, at least for what we have called coarse merging above (for fine merging, we *do* want to be able to construct new formulas).

An interesting feature of our model is that it allows for stating precisely the relationship between implicitly and explicitly represented knowledge, namely by investigating relationship between syntactic and the semantic axioms. So, what is the relative strength of a syntactic axiom and its semantic variant? For unanimity, for instance, we can show that the syntactic version does not entail the semantic version, nor *vice versa*. First, consider this example, showing that there are syntactically unanimous aggregators that are not semantically unanimous: Suppose three agents share a common TBox including the formulas $C \equiv D$ and $D \equiv E$, and suppose the ABox of the first agent includes only $C(a)$, the second only $D(a)$, and the third only $E(a)$. Now the majority rule will produce an empty ABox. This violates semantic unanimity, as $C(a)$ can be inferred from all three individual ABoxes, but not from the collective ABox. However, the majority rule clearly is (syntactically) unanimous. Second, a trivial counterexample shows that semantically unanimous aggregators need not be syntactically unanimous: Consider the aggregator F mapping any input to a fixed unsatisfiable ontology, such as $\{C \equiv D \sqcap \neg D, C(a)\}$. F is not syntactically unanimous, but it is semantically unanimous (as we can infer anything from a contradictory ontology). Still, intuitively, semantic unanimity is the (much) stronger property. This intuition can be confirmed for "well-behaved" aggregators:

Proposition 2. *Any satisfiable and exhaustive ontology aggregator that is semantically unanimous is unanimous.*

Proof. Take any F that is satisfiable, exhaustive, and semantically unanimous. Now pick any formula φ and any profile $\boldsymbol{O}$ such that $\varphi \in O_1 \cap \cdots \cap O_n$. By satisfiability of F, the outcome $F(\boldsymbol{O})$ is satisfiable and so is its deductive closure. For the sake of contradiction, assume $\varphi \notin F(\boldsymbol{O})$. $\varphi \in O_1 \cap \cdots \cap O_n$ implies $\varphi \in \mathrm{Cl}(O_1) \cap \cdots \cap \mathrm{Cl}(O_n)$. Thus, by semantic unanimity, $\varphi \in \mathrm{Cl}(F(\boldsymbol{O}))$. That is, there exists a formula in the support (namely φ) that could be added to $F(\boldsymbol{O})$ without rendering the set unsatisfiable. But this violates exhaustiveness, and we are done. □

Similar connections between syntactic and semantic variants can be established for the other axioms.

4 Procedures for Ontology Aggregation

We now define a number of simple procedures for ontology aggregation and discuss some of their properties, including both the extent to which they can guarantee that collective ontologies will be satisfiable and the extent to which they satisfy some of the axioms introduced earlier. We stress that these procedures are not sophisticated enough to be employed for real-world ontology aggregation. Rather, our intent is to provide a catalogue of basic procedures that can serve as building blocks for constructing more sophisticated procedures in the future. Fully understanding the properties of these basic procedures is a necessary step towards designing more advanced procedures.

4.1 The Majority Rule

We have already introduced the majority rule informally. Formally, it is defined as follows:

Definition 2 (Majority rule). *The majority rule is the ontology aggregator* M *with* $\mathrm{M}(\boldsymbol{O}) = \{\varphi \in \mathcal{L} \mid |\mathcal{N}_\varphi^{\boldsymbol{O}}| > \frac{n}{2}\}$ *for all* $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$.

We have seen that the majority rule can produce unsatisfiable collective ontologies. Following Endriss et al. [4], we call $\mathcal{L}$ *safe* for a given aggregator F if $F(\boldsymbol{O})$ is satisfiable for any profile $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$. We will now identify necessary and sufficient conditions for the safety of $\mathcal{L}$ under the majority rule.

Adapting the terminology from JA [10], we recall that an agenda $\mathcal{L}$ satisfies the *median property* if and only if every unsatisfiable set $X \subseteq \mathcal{L}$ contains itself an unsatisfiable set Y with cardinality at most 2. Now a simple reformulation of a known result due to Nehring and Puppe shows that an agenda $\mathcal{L}$ is safe for the majority rule if and only if it satisfies the median property [12,10,4]. This result can be refined if we put restrictions on the range of profiles on $\mathcal{L}$ that we consider. Description logical ontologies suggest a natural restriction of this kind due to the division of knowledge into the TBox and the ABox. Suppose we restrict attention to *profiles with a common TBox:* all agents agree on the TBox but still need to aggregate their ABoxes. Fix such a TBox $\mathcal{T}$. We say that $\mathcal{L}$ satisfies the *$\mathcal{T}$-median property* if and only if for every set of ABox formulas $X \subseteq \mathcal{L}^A$ such that $\mathcal{T} \cup X$ is unsatisfiable there exists a set $Y \subseteq X$ with cardinality at most 2 such such $\mathcal{T} \cup Y$ is also unsatisfiable. We obtain the following characterisation:

Proposition 3. *The majority rule will return a satisfiable ontology for any profile with a common TBox $\mathcal{T}$ if and only if the agenda $\mathcal{L}$ satisfies the $\mathcal{T}$-median property.*

Proof. One direction is proved by the doctrinal paradox we have seen earlier. For the other direction, assume the $\mathcal{T}$-median property holds but $\mathrm{M}(\boldsymbol{O})$ is unsatisfiable. By definition of the majority rule, the TBox of $\mathrm{M}(\boldsymbol{O})$ is exactly the common TBox $\mathcal{T}$. Thus, by the $\mathcal{T}$-median property, there must be a set Y of ABox formulas in $\mathrm{M}(\boldsymbol{O})$ with $|Y| \leqslant 2$ such that $\mathcal{T} \cup Y$ is unsatisfiable. First, Y cannot be empty as that would mean that $\mathcal{T}$ is unsatisfiable, contradicting our assumption that individual ontologies are satisfiable. Second, $|Y| = 1$ is also not possible, as that would mean that at least one individual ontology must have included that one formula in Y (together with $\mathcal{T}$), which would again contradict our assumption that individual ontologies are satisfiable. So suppose that $|Y| = 2$ with $Y = \{\varphi, \psi\}$. These formulas could only have been accepted by M if $|\mathcal{N}_\varphi^{\boldsymbol{O}}| > \frac{n}{2}$ and $|\mathcal{N}_\psi^{\boldsymbol{O}}| > \frac{n}{2}$. But this means that at least one agent must have accepted both φ and ψ (and $\mathcal{T}$). This again contradicts the assumption that individual ontologies are satisfiable. □

In fact, from a purely technical point of view, we can prove the same kind of result for any division of the agenda into two disjoint sets: those formulas on which there is certain agreement (here the TBox) on those on which there is not (here

the ABox). For any such division we can formulate a weakened version of the median property (relative to the first) and prove a corresponding (strengthened) characterisation theorem. In the context of ontology aggregation, we argue, such a division is particularly natural.

4.2 Quota Rules

We can generalise the idea underlying the majority rule and accept a formula for the collective ontology whenever the number of agents who do so meet a certain quota. This gives rise to the family of quota rules:

Definition 3 (Quota rules). *Let $q \in [0,1]$. The quota rule with quota q is the ontology aggregator F_q with $F_q(\boldsymbol{O}) = \{\varphi \in \mathcal{L} \mid |\mathcal{N}^{\boldsymbol{O}}_{\varphi}| \geqslant q \cdot n\}$ for all $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$.*

We could also generalise further and allow different quotas for different formulas; Dietrich and List [2] make a distinction between general and *uniform* quota rules. Observe that we obtain the majority procedure for $q = \frac{1}{2} + \epsilon$ for any positive $\epsilon < \frac{1}{n}$. Also observe that for $q \leqslant \frac{1}{n}$ the aggregator F_q simply returns the union of all individual ontologies.

We have seen earlier that the majority rule violates semantic unanimity. In fact, any quota rule does, unless we lower the quota so far as to obtain the trivial union aggregator:

Proposition 4. *A quota rule with quota q for n agents is semantically unanimous if and only if $q \leqslant \frac{1}{n}$.*

Proof (sketch). First, it is easy to check that if the quota is at most $\frac{1}{n}$, then semantic unanimity holds. To see that the axiom does not hold as soon as $q > \frac{1}{n}$, consider the following example. All agents agree on the same TBox $\{C_1 \equiv C_2, C_2 \equiv C_3, \ldots, C_{n-1} \equiv C_n\}$ and, for each $i \in \mathcal{N}$, the ABox of agent i consists of the single formula $C_i(a)$. Then $C_1(a)$ can be inferred from each agent's ontology, but it will not be accepted if $q > \frac{1}{n}$. □

Quota-based rules are (syntactically) anonymous, neutral, independent and monotonic [2]. We can strengthen Proposition 4 and show that anonymity and independence together with semantic unanimity are sufficient to single out the trivial union aggregator:

Proposition 5. *If F is anonymous, independent and semantically unanimous, then $F(\boldsymbol{O}) = O_1 \cup \cdots \cup O_n$ for any $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$.*

Proof (sketch). By a standard argument [9,4], if F is anonymous and independent, then there exists a family of functions $\{g_\varphi : \mathbb{N} \to \{0,1\}\}_{\varphi \in \mathcal{L}}$ such that $\varphi \in F(\boldsymbol{O})$ if and only if $g_\varphi(|\{i \in \mathcal{N} \mid \varphi \in O_i\}|) = 1$. That is, whether of not φ is accepted only depends on the number of agents accepting φ. Now, using a similar construction as in the proof of Proposition 4, we can show that semantic unanimity forces us to accept a formula as soon as any positive number of individual agents do. □

4.3 A Support-Based Procedure

The next aggregation procedure we introduce works as follows: we order the formulas in terms of the number of agents supporting them; we then accept formulas in decreasing order, but drop any formula that would render the ontology constructed thus far unsatisfiable. To decide which of two formulas with the same number of agents supporting it to try first, we introduce a *priority rule* $\gg$ mapping each profile $\boldsymbol{O}$ to a strict linear order $\gg_{\boldsymbol{O}}$ on $\mathcal{L}$ such that $\varphi \gg_{\boldsymbol{O}} \psi$ implies $|\mathcal{N}^{\boldsymbol{O}}_{\varphi}| \geqslant |\mathcal{N}^{\boldsymbol{O}}_{\psi}|$ for all $\varphi, \psi \in \mathcal{L}$.

Definition 4 (Support-based procedure). *Given a priority rule $\gg$, the support-based procedure with $\gg$ is the ontology aggregator* $\mathrm{SBP}_{\gg}$ *mapping any profile $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$ to $\mathrm{SBP}_{\gg}(\boldsymbol{O}) := \Phi$ for the unique set $\Phi \subseteq \mathcal{L}$ for which $\varphi \in \Phi$ if and only if*

(i) $\mathcal{N}^{\boldsymbol{O}}_{\varphi} \neq \emptyset$ *and*
(ii) $\{\psi \in \Phi \mid \psi \gg_{\boldsymbol{O}} \varphi\} \cup \{\varphi\}$ *is satisfiable.*

We can also define an *irresolute* aggregator that returns the set of all ontologies obtained by *some* choice of priority rule: $\mathrm{SBP}(\boldsymbol{O}) := \{O \mid \mathrm{SBP}_{\gg}(\boldsymbol{O}) = O \text{ for some } \gg\}$.

The SBP clearly satisfies the axioms of anonymity, monotonicity, groundedness (due to condition (i)), and exhaustiveness (due to condition (ii)). Neutrality is violated by virtue of having to fix a priority rule $\gg$. Independence is also violated (because φ may cease to be accepted if a formula it is contradicting receives additional support).

Several variants and generalisations of the SBP are possible and interesting. For instance, we can replace $\gg$ as defined above with *any* other function mapping each profile $\boldsymbol{O}$ to a linear order $\gg_{\boldsymbol{O}}$ on $\mathcal{L}$. Each choice of $\gg$ corresponds to a different *greedy procedure* that attempts to accept as many formulas as possible without violating satisfiability in order of priority as specified by $\gg_{\varphi}$. For instance, a priority rule $\gg$ for which $\varphi \gg_{\boldsymbol{O}} \psi$ holds whenever $\mathcal{N}^{\boldsymbol{O}}_{\varphi} \supseteq \mathcal{N}^{\boldsymbol{O}}_{\psi}$ does but not necessarily whenever $|\mathcal{N}^{\boldsymbol{O}}_{\varphi}| \geqslant |\mathcal{N}^{\boldsymbol{O}}_{\psi}|$ does will be appropriate to aggregate ontologies from sources with different degrees of reliability (i.e., when the violation of anonymity is acceptable). Another attractive variant would be a *semantic* SBP, where we define $\gg$ in terms of $\{i \in \mathcal{N} \mid O_i \models \varphi\}$ instead of $\mathcal{N}^{\boldsymbol{O}}_{\varphi}$. That is, under this procedure we accept formulas (supported by at least one agent) in order of priority defined in terms of the number of agents who were able to *infer* those formulas from their own ontologies (but not necessarily included them explicitly).

4.4 A Distance-Based Procedure

In voting theory, many voting rules can be defined using a notion of distance. The well-known *Kemeny rule* is a natural example [7]. Similar ideas have also been used in belief merging [8].

We will now define an aggregation procedure that chooses from a class of acceptable ontologies (namely the satisfiable ones) that ontology that minimises

the sum of the distances to the individual ontologies. A common choice is the *Hamming distance:* the distance between two ontologies O and O' is the number of formulas that are included in one and only one of O and O'. In fact, the Hamming distance is not appropriate here, because it gives the same weight to a formula φ that an agent has stated but that will not be included in the collective ontology as to a formula ψ that she has omitted but that will be included (when in fact the former should be much worse; indeed, she may be entirely indifferent to the latter). That is, distances *stricto sensu*, which are symmetric, are not suitable for our purposes. With a slight abuse of terminology, we shall still call the function $d : (A, B) \mapsto |\{\varphi \mid \varphi \in A \text{ and } \varphi \notin B\}|$ a distance.

Definition 5 (Distance-based procedure). *The distance-based procedure is the (irresolute) ontology aggregator* DBP *mapping any profile* $\boldsymbol{O} \in \mathrm{On}(\mathcal{L})^{\mathcal{N}}$ *to the following set of satisfiable ontologies:*

$$\mathrm{DBP}(\boldsymbol{O}) = \mathrm{argmin}_{O \in \mathrm{On}(\mathcal{L})} \sum_{i \in \mathcal{N}} d(O_i, O)$$

To obtain a resolute aggregator, the DBP needs to be combined with a tie-breaking rule, which will violate either anonymity or neutrality. It also violates independence, because O does not range over all possible ontologies. On the other hand, it is satisfiable by construction. Note that if we choose a tie-breaking rule that selects a maximal set (wrt. set-inclusion), then the DBP will always return a maximally satisfiable set and thus satisfy the axiom of exhaustiveness.

4.5 Two-Stage Procedures

Finally, we briefly sketch an approach for two-stage procedures. Depending on the application, we may give priority to terminological knowledge over assertional knowledge, or *vice versa*, and define aggregation procedures accordingly. This idea is closely related to two classical procedures in JA, the *premise-based procedure*, where individuals vote on the premises by majority and then draw the conclusions, and the *conclusion-based procedure*, where each individual draws her own conclusions and then votes on them by majority [9]. The problem with these procedures is that we lack a convincing general approach for how to label a given proposition as either a premise or a conclusion. There is a significant difference in our case: when we aggregate ontologies, we have a clear separation between two classes of formulas by definition, namely the TBox and the ABox, so we can avoid the problem of splitting the agenda into premises and conclusions.

Definition 6 (Assertion-based procedures). *An (irresolute) assertion-based procedure maps each profile* $\boldsymbol{O}$ *to the set of ontologies obtained as follows:*

(1) Choose an aggregator F_A *restricted to ABox formulas, and let* $F_A(\boldsymbol{O})$ *be the outcome.*
(2) Then the TBox is defined as follows:

$$F_T(\boldsymbol{O}) = \mathrm{argmin}_{O \in \mathrm{On}(\mathcal{L})} \sum_{i \in \mathcal{N}} d(F_A(\boldsymbol{O}) \cup O_i^T, O)$$

An assertion-based procedure stresses the information coming from the ABox. A natural choice for the procedure used in the first step would be the majority rule. In the second step we then select a TBox that is satisfiable in view of the majority ABox and that minimises the cumulative distance to the individual TBoxes. Observe that it is possible that the collective TBox obtained in this manner is empty. An interesting variant of this approach may be to allow agents to revise their TBoxes themselves after the collective ABox has been fixed.

Similarly, we may want to give priority to TBox information and first aggregate TBoxes, then fix a TBox, and finally aggregate ABoxes.

5 Conclusion and Future Work

We have presented a framework for aggregating individual ontologies, consisting of both a TBox and an ABox, inspired by social choice theory. We have discussed axioms that are closely related to well-known fairness conditions and we have introduced new axioms defining a notion of efficiency for the aggregation of ontologies. We have then presented relevant results concerning those axioms and several ontology aggregation procedures we introduced, discussing how they balance fairness and efficiency. We have concentrated on coarse ontology merging, since we wanted to model the aggregation of the information actually provided by agents, as explicitly reflected by our groundedness axiom.

Concerning future work, we believe that the social choice approach provides useful insights also for fine merging. For example, support-based procedures and distance-based procedures can potentially be adapted to deal with *concept merging* (i.e., the construction of new TBox definitions out of definitions stemming from different individual ontologies), providing further qualitative *desiderata* that can be used to select among several possible ways of building concept definitions. We also believe that our work can provide an interesting starting point for future research in judgment aggregation and social choice theory. Ontologies suggest a very rich notion of agent, since they allow for representing the preferences an agent might have over a given set of alternatives together with her information on such alternatives and her criteria for choosing. In this sense, our approach to ontology aggregation can lead to a richer model of collective information and choices.

Acknowledgments. We would like to thank the participants of the Computational Social Choice Seminar at the ILLC for their feedback on an early incarnation of this work, particularly Umberto Grandi, Szymon Klarman and Eric Pacuit.

References

1. Baader, F., Nutt, W.: Basic description logics. In: The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
2. Dietrich, F., List, C.: Judgment aggregation by quota rules: Majority voting generalized. Journal of Theoretical Politics 19(4), 391–424 (2007)

3. Dietrich, F., List, C.: Judgment aggregation with consistency alone. Working paper. London School of Economics (2009)
4. Endriss, U., Grandi, U., Porello, D.: Complexity of judgment aggregation: Safety of the agenda. In: Proc. AAMAS-2010 (2010)
5. Flouris, G., Huang, Z., Pan, J.Z., Plexousakis, D., Wache, H.: Inconsistencies, negations and changes in ontologies. In: Proc. AAAI-2006 (2006)
6. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: Classification and survey. Knowledge Engineering Review 23(2), 117–152 (2008)
7. Gaertner, W.: A Primer in Social Choice Theory. Oxford University Press, Oxford (2006)
8. Konieczny, S., Lang, J., Marquis, P.: DA^2 merging operators. Artificial Intelligence 157(1-2), 49–79 (2004)
9. List, C., Pettit, P.: Aggregating sets of judgments: An impossibility result. Economics and Philosophy 18(1), 89–110 (2002)
10. List, C., Puppe, C.: Judgment aggregation: A survey. In: Handbook of Rational and Social Choice. Oxford University Press, Oxford (2009)
11. May, K.O.: A set of independent necessary and sufficient conditions for simple majority decision. Econometrica 20(4), 680–684 (1952)
12. Nehring, K., Puppe, C.: The structure of strategy-proof social choice. Part I: General characterization and possibility results on median spaces. Journal of Economic Theory 135(1), 269–305 (2007)
13. Noy, N.F., Musen, M.A.: Algorithm and tool for automated ontology merging and alignment. In: Proc. AAAI-2000 (2000)
14. Ribeiro, M.M., Wassermann, R.: Base revision for ontology debugging. Journal of Logic and Computation 19(5), 721–743 (2009)

Social Commitment Delegation and Monitoring*

Özgür Kafalı[1] and Paolo Torroni[2]

[1] Department of Computer Engineering - Boğaziçi University
34342, Bebek, İstanbul - Turkey
ozgurkafali@gmail.com
[2] DEIS - University of Bologna
V.le Risorgimento, 2, 40136, Bologna - Italy
paolo.torroni@unibo.it

Abstract. The success of contract-based multiagent systems relies on agents complying with their commitments. When something goes wrong, it is important to understand what are the commitments' mutual relations as well as their individual states. Accordingly, we explore how commitments are related through the three-agent commitment delegation operation. We then propose exception monitoring based on such relations, and demonstrate it via a case study.

1 Introduction

A social commitment describes a contract between two agents: the debtor commits to satisfy a property for the creditor [14]. In a contract-based multiagent system, several such commitments are in effect, e.g., the merchant is committed to deliver the goods when the customer pays, the bank is committed to confirm the customer's payment in three days. The former is represented by a conditional commitment CC(merchant, customer, paid, delivered), while the latter is represented by a base-level temporal commitment C(bank, customer, paid(3)), e.g., with a deadline. Often, agents delegate their commitments to others for several reasons, e.g., they are not capable of satisfying the properties. C(courier, merchant, delivered) is a delegation of CC(merchant, customer, paid, delivered) in which the merchant delegates the task of delivery to the courier.

Usually, commitments formed between different agents are connected to each other; either explicitly (by delegation), or implicitly (other dependencies). The delegation operation is important in the sense that it extends the set of agents that a commitment is involved with. The merchant-courier example demonstrates a typical case of delegation, where the commitment of delivery between the merchant and customer agents is extended with the courier agent. The customer agent may not be aware of this extension until the delivery is completed, or something goes wrong (e.g., the deadline passes). In the case of a problem, this connection should be revealed so that if the problem is related to the courier, it can be identified. However, not all connections are explicitly identified as in the

* This paper extends the AAMAS '11 poster paper [11].

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 171–189, 2011.

delegation case. The bank-customer example is a typical case of implicit causal dependency; the bank's confirmation of the customer's payment affects the merchant's commitment of delivery towards the customer. Again, if something goes wrong, the commitments should be traced in a chain to identify the problem.

We say that an exception has occurred regarding a property of the system if there is something wrong in the chain of commitments that are formed to satisfy that property. Possible causes of such exceptions are:

- *Violation:* One of the commitments related to the subject property is violated by its debtor, e.g., the bank does not confirm the customer's payment in time. Often, an exception is considered identical to a commitment violation.
- *Improper delegation:* One of the commitments related to the subject property is delegated without respecting its previous deadline, e.g., the merchant delays the courier's delivery by handing over the goods late.
- *Misalignment:* Two or more commitments related to the subject property are not aligned with each other, possibly due to different observations of the participating agents, e.g., the bank confirms the customer's payment but fails to notify the merchant on time.

When there are many commitments in the system at hand, in order to identify an exception we need effective ways to explore the space of commitments. In particular, we need to identify links between commitments and exclude from our search the irrelevant instances. The process of tracking down individual commitment states is called commitment monitoring [1,16]. We extend monitoring to enable run-time tracking of exceptions via the links between agents' commitments. To this end, we propose similarity relations to connect commitments based on their components: the agents, the properties, and the temporal aspects. The properties of commitments have been studied before in [4]. Moreover, temporal constraints are used to compare commitments in [9]. However, both approaches mainly focus on two-agent interactions, or compare commitments one by one. In [5], delegation is also taken into account from the perspective of commitment alignment, but no temporal aspects are considered. In [1], commitments are tracked and reasoned upon centrally. Those work should be able to capture exceptions caused by improper delegation or misalignment. Still, they fail to reveal implicit commitment dependencies as we propose to identify here. Note that a relatively insignificant commitment violation may be the cause of an exception.

Accordingly, we propose a monitoring framework where the relations between agents' commitments are extracted to identify problems related to a commitment violation. The framework identifies all improper delegations that have occurred in the system.

The rest of the paper is structured as follows. Section 2 reviews commitments and introduces our formal model. Section 3, 4 and 5 describe commitment delegation and the similarity relations based on delegation. Section 6 describes how monitoring is performed, proposing an implementation. Section 7 demonstrates the case study. Finally, Section 8 concludes the paper with further discussion.

2 Formal Model

A commitment [14] is formed between two agents; the debtor commits to the creditor for satisfying a property. There are two types of commitments:

- *Conditional commitment:* CC(X, Y, Q, P(T)) represents a conditional commitment where X is the debtor, Y is the creditor, Q is the antecedent, and P(T) is the consequent. T is the *limit* of the consequent.
- *Base-level commitment:* When the antecedent Q is satisfied, a base-level commitment C(X, Y, P(T)) is formed. T is the *deadline* of the consequent.

Our definition of commitments extends the Singh's definition [14] with the notions of limit and deadline, following a recent line of research on reasoning with commitments in time [1,16]. The limit of a consequent of a conditional commitment is an existential temporal constraint indicating the the number of time units allowed to satisfy P, as of the moment the antecedent Q is satisfied (relative deadline). The deadline in the base-level commitment represents instead an absolute deadline.

For example, the commitment CC(merchant, customer, paid, delivered(7)) tells that the delivery should take place at most seven time units after the payment. If the payment is done at time 3, then the base-level commitment C(merchant, customer, delivered(10)) tells that the merchant has to deliver latest at time 10.

From a logical perspective, a base-level commitment can be seen as a special case of a conditional commitment, in which the antecedent Q is true.

The commitment properties (i.e., antecedent and consequent) are described by conjunctions of atomic propositions together with temporal constraints to express limits or deadlines. Currently, we do not support negation or disjunction of properties. A commitment is a live object and changes state through its lifecycle [17]. We use the following four states for commitments: (1) *conditional*, when Q is not yet satisfied, (2) *active*, when Q is satisfied and P is not yet satisfied, (3) *fulfilled*, when P is satisfied within T, and (4) *violated* when P is not satisfied within T.

We are now ready to define the monitoring framework.

Definition 1. Monitoring framework $\mathcal{F}$ is a five-tuple $\langle \mathcal{P}, \mathcal{R}, \mathcal{A}, \mathcal{T}, \mathcal{M} \rangle$, where

- $\mathcal{P}$ is a set of conditional commitment templates, representing an abstract contract or protocol [17,3],
- $\mathcal{R}$ is a set of roles, each consisting of a subset of $\mathcal{P}$'s commitments and a set of actions,
- $\mathcal{A}$ is a set of agents, each enacting a role in $\mathcal{R}$,
- $\mathcal{T}$ is a trace of events, consisting of a set of actions performed at specific time points, and
- $\mathcal{M}$ is the monitoring process. ∎

Elements of $\mathcal{P}$ are abstract entities, i.e., *templates* which include roles from $\mathcal{R}$ in place of agents. When the agents in $\mathcal{A}$ are bound to the roles in the protocol [6], the commitments become real. The trace of events $\mathcal{T}$ describes a specific protocol execution, by which commitments change state accordingly. The monitoring process $\mathcal{M}$ consistently checks for improper delegations during the protocol's execution. Similarly to diagnosis, which looks for assumptions over executions of activities that classifies these executions either as correct (an activity behaves as intended) or faulty (an activity does not behave as intended) [7], monitoring seeks to detect faulty activity executions. It is important that monitoring is carried out at runtime, in reaction to events that bring about properties characterizing a faulty state. We propose to identify exceptions through the monitoring of agents' commitments. We describe $\mathcal{M}$ in more detail in Section 6.

3 Delegation

Previous work has looked at commitments and their relations from different angles. Chopra and Singh [5,4] compare commitments via a *strength* relation using the commitments' properties, Kafalı *et al.* [9] focus on the temporal aspects of commitments and provide similarity relations based on the commitments' deadlines. In particular, we combine both approaches, and propose similarity relations based on the three-agent commitment delegation operation. First, we describe a delegation.

Definition 2. A *delegation* of a commitment *CC (X, Y, Q, P)*, called *primary*, is a new commitment where either X or Y plays the role of the creditor or debtor, and a new agent Z is responsible for bringing about the antecedent Q or the consequent P. ∎

Six types of delegation are particularly meaningful. Only some of them have been considered previously in the literature. Let us define and illustrate them one by one, considering *CC(merchant, customer, paid, delivered)* as our *primary*, like in Figure 1.

Definition 3. (Explicit delegation) The primary is canceled and a new commitment *CC (Z, Y, Q, P)* is created. That is, a new debtor is committed to the same creditor. ∎

This delegation operation was proposed by Yolum and Singh [17]. A possible explicit delegation of the primary is *CC(courier, customer, paid, delivered)*. The new debtor *courier* replace the old debtor *merchant*, and the primary is canceled. Note that the antecedent (*paid*) is unchanged, but that may not necessarily be the case in an extended version of explicit delegation (see below).

Definition 4. (Weak explicit delegation) The primary is canceled and a new commitment *CC (Y, Z, P, Q)* is created. That is, the creditor Y of the primary is now the debtor of the new commitment, and Y wishes to achieve P via a new creditor Z. This is a weak delegation to achieve P since there is no obligation for Z to satisfy P unless Z needs Q satisfied. ∎

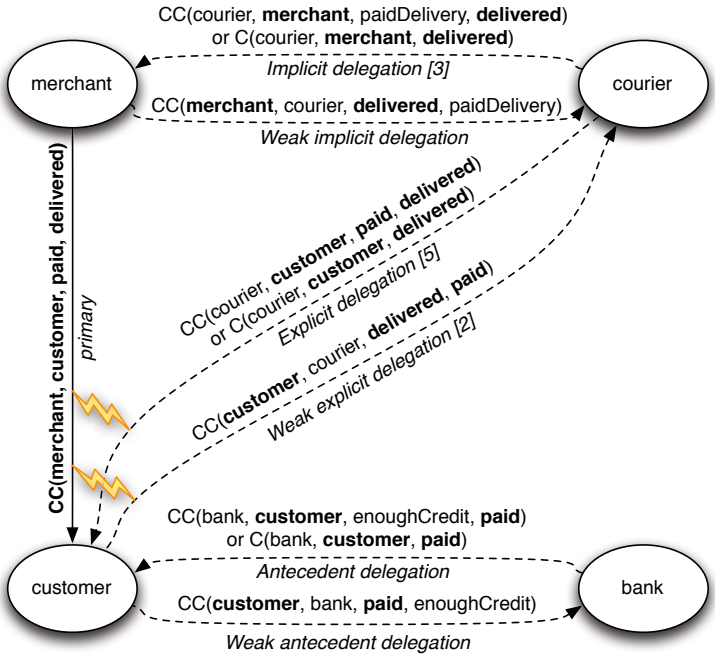


Fig. 1. Sample Delegations

The concept of weak delegation is inspired by Chopra *et al.* 's work [3]. A possible weak explicit delegation of the primary is *CC(customer, courier, delivered, paid)*. Note that the roles of creditor and debtor are reversed, and accordingly also antecedent and consequent are reversed.

Definition 5. (Implicit delegation) While the primary is still active, a new commitment *CC (Z, X, R, P)* is created. That is, the debtor *X* of the primary is now the creditor of a new commitment for the same consequent *P*. ∎

This type of delegation chain (e.g., two dependent commitments) was proposed by Kafalı *et al.* [9]. A possible implicit delegation of the primary is *CC(courier, merchant, paidDelivery, delivered)*. Note that the creditor is the *merchant*, which is the primary's debtor. For that reason, to maintain a commitment to the initial creditor (the *customer*), the primary is not canceled, but it remains.

Definition 6. (Weak implicit delegation) While the primary is still active, a new commitment *CC (X, Z, P, R)* is created. That is, the debtor *X* of the primary also becomes the debtor of a new commitment where the antecedent *P* is the primary's consequent. ∎

A possible weak implicit delegation of the primary is *CC(merchant, courier, delivered, paidDelivery)*.

Definition 7. (Antecedent delegation) While the primary is still active, a new commitment *CC (Z, Y, R, Q)* is created. That is, the creditor *Y* of the primary also becomes the creditor of a new commitment for the antecedent *Q* of the primary. ∎

A possible antecedent delegation of the primary is *CC(bank, customer, enoughCredit, paid)*. Note that the initial consequent (*delivered*) does not appear in the antecedent delegation. For that reason, to maintain a commitment about the initial consequent, the primary is not canceled, but it remains.

Definition 8. (Weak antecedent delegation) While the primary is still active, a new commitment *CC (Y, Z, Q, R)* is created. That is, the creditor *Y* of the primary is now the debtor of a new commitment which has the same antecedent *Q* as the primary. ∎

A possible weak antecedent delegation of the primary is *CC(customer, bank, paid, enoughCredit)*.

Most of the above definitions can be extended to base-level commitments. In addition, (weak) explicit delegation can be extended to have an antecedent R different from Q. Also note that a special case of (weak) implicit delegation is where R equals Q. Figure 1 summarizes the examples of commitment delegation given above.

We gave an exhaustive account of how a commitment can be *rationally* delegated, i.e., by preserving the responsibilities of roles in relation with the primary's properties. To see this, let us enumerate all the rational delegation possibilities for a commitment, and then show that they are covered by Definitions 3-8.

Let us first consider rational delegations that include the consequent *P* of a primary *CC(X, Y, Q, P)*. The secondary will have *P* either as the consequent or the antecedent. Moreover, by Definition 2, a new agent *Z* should replace either *X* or *Y*.

Case 1: *P* is the secondary's consequent. In the primary, the debtor is responsible for *P*. Therefore we need a new debtor *Z* responsible for *P*. There are three alternatives: (1.1) *CC(Z, Y, R, P)*, which is an (extended) explicit delegation; (1.2) *CC(Z, X, R, P)*, which is an implicit delegation; or (1.3) *CC(Z, W, R, P)*, with *W* also different from *X* and *Y*. However, the latter case is just a different commitment about *P*, but it cannot be considered a delegation, since there is no common agent between primary and secondary.

Case 2: *P* is the secondary's antecedent. In that case, *Z* will be new creditor expecting *P*. There are again three alternatives: (2.1) *CC(Y, Z, P, R)*, which is a weak explicit delegation, (2.2) *CC(X, Z, P, R)*, which is a weak implicit delegation; and (2.3) *CC(W, Z, P, R)*, which cannot be considered a delegation for the same reasons as in Case 1.

Let us now consider rational delegations that include the antecedent *Q*. There are again two possibilities:

Case 3: *Q* is the secondary's consequent. Rationally, the antecedent should be different from *Q* and *P*, and the creditor should be the same *Y*, since *Y* is the

agent who expects Q satisfied. The only option for a delegation is to have a new debtor Z, other than X or Y. That is the definition of antecedent delegation, *CC(Z, Y, R, Q)*.

Case 4: Q is the antecedent. Rationally, the secondary's debtor should be Y, and the secondary should have a new creditor Z, other than X or Y. This defines a weak antecedent delegation, *CC(Y, Z, Q, R)*.

4 Similarity

We say that a commitment is *delegation-similar* to another commitment if one is a delegation of the other according to Definitions 3-8. We showed that our account of commitment delegation is exhaustive (if we restrict ourselves to rational delegations). Now, we shift the focus to commitments that are similar *via* other commitments. We will therefore need to define ternary similarity relations, which include two commitments similar to each other (we call them "secondary"), and a "primary" commitment which connects them. For the sake of presentation, we omit here the temporal constraints of the commitments, which will be the subject of the next section.

Definition 9. Commitment $CC_1(X_1, Y_1, Q_1, P_1)$ is consequent-delegation similar to commitment $CC_2(X_2, Y_2, Q_2, P_2)$ via commitment $CC_3(X_3, Y_3, Q_3, P_3)$ iff

1. $P_3 \models P_1 \wedge P_2$ (conjunction), and
2. $Y_1 = Y_2 = X_3$ (delegation).

We call CC_3 the *primary*, and CC_1, CC_2 *secondary*. ∎

In Definition 9, the debtor X_3 of a primary about a complex commitment (in which the consequent P_3 is a conjunction of several parts, $P_3 = P_1 \wedge P_2 \wedge \ldots$) negotiates two implicit delegations of two such parts, with two new agents. As a result, we have two delegations, and therefore two secondaries, connected by the same primary. This corresponds to an X_3 establishing one or more sub-contracts to achieve some of the objectives. Note that the conditional commitments in the definition can also be base-level commitments since the antecedents are not part of the similarity relation.

Consider for instance the commitments in $\mathcal{C}_{consequent}$:

$$\mathcal{C}_{consequent} = \begin{cases} C_3(\text{merchant, customer, delivered} \wedge \text{invoiced}) \\ C_1(\text{courier, merchant, delivered}) \\ C_2(\text{accountant, merchant, invoiced}) \end{cases}$$

According to C_3, the merchant is committed to deliver the item and provide the invoice. Now, assume that the merchant delegates the delivery of the item to the courier, and the preparation of the invoice to his accountant. Thus, we have the commitments (C_1 and C_2). Accordingly, C_1 and C_2 are consequent-delegation similar via C_3. C_3 is primary, and C_1, C_2 are secondary. The converse case (weak delegation) is also possible:

Definition 10. Commitment $CC_1(X_1, Y_1, Q_1, P_1)$ is weak consequent-delegation similar to commitment $CC_2(X_2, Y_2, Q_2, P_2)$ via commitment $CC_3(X_3, Y_3, Q_3, P_3)$ iff

1. $P_3 \models Q_1 \wedge Q_2$ (conjunction), and
2. $X_1 = X_2 = X_3$ (delegation).

Let us now focus on the antecedent. Consider the following example:

$$\mathcal{C}_{antecedent} = \begin{cases} CC_3(\text{merchant, customer, paid} \wedge \text{confirmed, delivered}) \\ C_1(\text{accountant, customer, paid}) \\ C_2(\text{bank, customer, confirmed}) \end{cases}$$

According to CC_3, once the customer pays and her card is confirmed, the merchant will deliver the item. Now, the customer delegates the payment to her accountant, and the confirmation to the bank. This brings us two new commitments, C_1 and C_2, respectively. The following definitions capture this type of similarity between C_1 and C_2.

Definition 11. Commitment $CC_1(X_1, Y_1, Q_1, P_1)$ is antecedent-delegation similar to commitment $CC_2(X_2, Y_2, Q_2, P_2)$ via commitment $CC_3(X_3, Y_3, Q_3, P_3)$ iff

1. $Q_3 \models P_1 \wedge P_2$ (conjunction), and
2. $Y_1 = Y_2 = Y_3$ (delegation).

As usual, CC_3 is the *primary*, and CC_1, CC_2 *secondary*.

Definition 11 revises Definition 9 for conjunction of antecedents. This also represents a causal relation between the commitments, since CC_3 cannot be discharged until CC_1 and CC_2 are both fulfilled. The secondary commitments in the definition can also be base-level commitments. However, CC_3 cannot be a base-level commitment anymore since Q_3 is part of the relation. Antecedent delegation is not actively used in monitoring, but acts as a connective in a chain of delegations. In the previus example, C_1 and C_2 are antecedent-delegation similar via CC_3. A weaker version is possible:

Definition 12. Commitment $CC_1(X_1, Y_1, Q_1, P_1)$ is weak antecedent-delegation similar to commitment $CC_2(X_2, Y_2, Q_2, P_2)$ via commitment $CC_3(X_3, Y_3, Q_3, P_3)$ iff

1. $Q_3 \models Q_1 \wedge Q_2$ (conjunction), and
2. $X_1 = X_2 = Y_3$ (delegation).

Let us finally consider the following situation:

$$\mathcal{C}_{causal} = \begin{cases} CC_1(\text{bank, client, requested, delivered}) \\ CC_3(\text{courier, bank, printed, delivered}) \\ C_2(\text{office, bank, printed}) \end{cases}$$

According to CC_1, once the client requests a credit card, the bank will deliver the card. Now, the bank delegates the delivery to the courier via CC_3. However, in order to deliver, the courier needs the card printed. Thus, the bank makes another delegation with the office via C_2. There is a causal relation between commitments here: in order to satisfy CC_1's secondary CC_3, a new commitment C_2 is in place. In particular, CC_3 is the link between two otherwise seemingly unrelated commitments.

We will say that CC_1 and C_2 are causal-delegation similar via CC_3. Put formally:

Definition 13. Commitment $CC_1(X_1, Y_1, Q_1, P_1)$ is causal-delegation similar to commitment $CC_2(X_2, Y_2, Q_2, P_2)$ via commitment $CC_3(X_3, Y_3, Q_3, P_3)$ iff

1. $P_1 = P_3$ and $X_1 = Y_3$ (implicit-delegation), and
2. $P_2 = Q_3$ and $Y_2 = Y_3$ (antecedent-delegation).

Here, we call CC_1 *outcome*, CC_2 *cause*, and CC_3 *connective*. ∎

Definition 13 connects two commitments through two delegations; one consequent (implicit) and one antecedent delegation. Here, CC_1 and CC_2 can also be base-level commitments since Q_1 and Q_2 are not part of the relation. However, CC_3 cannot be a base-level commitment since Q_3 is part of the relation. We do not consider the weak delegation case for causal delegation. Because, if the *consequent delegation* is weak, then the causal structure between the *connective* and *cause* no longer exists.

The similarity relations described in Definitions 9-13 allow us to trace commitments in a chain-like structure, e.g., nested delegations. They are also exhaustive, in the sense that they describe all possible, rational cases of commitment delegation. Let us informally see why.

The primary commitment in (weak) explicit delegation no longer exists after delegation, thus we do not need to consider it in the similarity relations. That is, there is no longer a primary commitment to compare with. Implicit delegation is covered in Definition 13, between commitments *outcome* and *connective*. In addition, Definition 9 covers implicit delegation where conjunction of consequents is considered. Weak implicit delegation with conjunction of consequents is covered in Definition 10, the single consequents case is trivial (i.e., each consequent is identical).

Antecedent delegation is covered in Definition 13, between commitments *connective* and *cause*. In addition, Definition 11 covers antecedent delegation where conjunction of antecedents is considered. Weak antecedent delegation with

conjunction of antecedents is covered in Definition 12. Again, the single antecedents case is trivial.

5 Limits and Deadlines

We will now extend the ternary relations by taking into account the temporal constraints. Since we are interested in exceptional situations and in the possible reasons behind them, we will identify and define cases of delegation in which the deadline of the primary is not properly propagated onto the secondary. In particular, we will call *improper* a delegation that exceeds the deadline of the primary commitment.

Definition 14. An improper consequent delegation between a primary and a secondary commitment (as by Definition 9) occurs if either of the following holds:

1. All conditional: If the limit of a secondary is greater than the limit of the primary.
2. All base-level: If the deadline of a secondary is greater than the deadline of the primary.
3. Only primary conditional: If the deadline of a secondary is greater than the limit of the primary added to the current time point.
4. Only primary base-level: If the limit of a secondary added to the current time point is greater than the deadline of the primary.

Consider for instance the commitments in $\mathcal{C}_{improper-consequent}$:

$$\mathcal{C}_{improper-consequent} = \begin{cases} C_3(\text{merchant, customer, delivered}(12) \wedge \text{invoiced}(12)) \\ C_1(\text{courier, merchant, delivered}(12)) \\ C_2(\text{accountant, merchant, invoiced}(14)) \end{cases}$$

These are the same commitments in $\mathcal{C}_{consequent}$, modified with temporal constraints. Assume that the current time is 8. Now, this is an improper delegation. Because, the deadline of C_2 is greater than that of C_3. Note that the occurrence of an exception, although likely, is not inevitable since the accountant may still satisfy *invoiced* at time 12.

Note that we cannot have an improper antecedent delegation, since we do not consider time limits for the antecedent of a commitment in this work.

Definition 15. An improper causal delegation between a cause and an outcome commitment (as by Definition 13) occurs if either of the following holds:

1. All conditional: If the limit of the cause added to the limit of the connective is greater than the limit of the outcome.
2. Only connective conditional: If the deadline of the cause added to the limit of the connective is greater than the deadline of the outcome.

Consider the commitments in $\mathcal{C}_{improper-causal}$:

$$\mathcal{C}_{improper-causal} = \begin{cases} CC_1(\text{bank, client, requested, delivered}(7)) \\ CC_3(\text{courier, bank, printed, delivered}(5)) \\ CC_2(\text{office, bank, confirmed, printed}(3)) \end{cases}$$

These are the same commitments as in $\mathcal{C}_{causal}$, modified with temporal constraints. Obviously, there is a problem with this choice of conditional commitments. Because, in order for the card to be delivered, it has to be printed first, and the time requirements for those two processes exceed the time limit that the bank has towards the client. The bank should have gotten into other commitments that would have lead the fulfillment of its primary commitment towards the client [3]. However, note that CC_2 and CC_3 may still fulfill CC_1 since the debtors of those commitments may satisfy the consequents before the deadlines.

Definition 16. An improper delegation is an improper consequent delegation or an improper causal delegation, or a combination of them. ∎

Combinations of delegations can occur in a chain-like structure. In such cases, the deadlines or limits should be propagated for correct monitoring. Consider for instance the commitments in $\mathcal{C}_{joint}$:

$$\mathcal{C}_{joint} = \begin{cases} C_1(\text{bank, client, delivered}(10)) \\ CC_2(\text{courier, bank, printed} \wedge \text{tested, delivered}(5)) \\ C_3(\text{office, bank, printed}(5) \wedge \text{tested}(5)) \\ C_4(\text{operator, office, printed}(5)) \\ C_5(\text{tester, office, tested}(7)) \end{cases}$$

There is no problem when we consider C_1, CC_2 and C_3 only; the deadlines are consistent. However, the deadline of C_5 is greater than the deadline of C_3 which creates a problem for the C_3, C_4 and C_5 consequent-delegation group. This should be propagated up to C_1. That is, the expected deadline of C_3 is now extended, which further extends the expected deadline of C_1.

6 Monitoring

Given a monitoring framework $\mathcal{F} = \langle \mathcal{P}, \mathcal{R}, \mathcal{A}, \mathcal{T}, \mathcal{M} \rangle$, $\mathcal{M}$ identifies all the improper delegations that occurred up to the current time point T. It makes use of the current states of the commitments $\mathcal{C}_T$ in the system to produce $\mathcal{M}_T$; the set of improper delegations. Now, let us formally describe $\mathcal{M}$.

Definition 17. Given a monitoring framework $\mathcal{F}$, the current time point T, and a set of commitments $\mathcal{C}_T = \{C_1, ..., C_i, ..., C_n\}$ that describe the current system at T, the monitoring process $\mathcal{M}$ produces the monitoring outcome $\mathcal{M}_T = \{(C_i, C_j) \mid C_i, C_j \in \mathcal{C}_T \text{ and } C_i \text{ is an improper delegation of } C_j\}$ that contains the improper delegations among C_T. ∎

The purpose of monitoring is to identify the faults (e.g., improper delegations) among agents' commitments. Once we have identified the improper delegations, we can signal an exception based on $\mathcal{M}_T$. The following two rules describe when an exception occurs:

$$\frac{monitor(C_m) \wedge \exists C_i : (C_i, C_m) \in \mathcal{M}_T}{exception(C_i, C_m)}$$

The first rule states that if we are interested in the monitoring of commitment C_m, and there is a commitment C_i that is an improper delegation of C_m, then there is an exception.

$$\frac{monitor(C_m) \wedge \exists C_j : delegation(C_j, C_m) \wedge \exists C_i : (C_i, C_j) \in \mathcal{M}_T}{exception(C_i, C_m)}$$

The second rule extends the first rule by taking into account nested delegations. In particular, if a delegation (not necessarily improper) of the subject commitment C_m has an improper delegation, then there is an exception. Note that the level of nested delegations is indefinite. That is, C_m may have been delegated several times (e.g., C_m to C_x and C_x to C_y), and then the occurrence of an improper delegation triggers the exception. Note that we do not provide an algorithm for exception monitoring in this paper. Rather, we present the rules that describe how a certain case will be considered an exception in a declarative way. We plan to build a distributed procedure where agents will identify exceptions via an exchange of their local knowledge.

We implemented a proof-of-concept monitoring framework prototype using *j*$\mathcal{REC}$ [1,16,9], a tool for run-time monitoring of commitments that accepts specifications written in the $\mathcal{REC}$ ($\mathcal{R}$eactive $\mathcal{E}$vent $\mathcal{C}$alculus) language [2].[1] The input to a $\mathcal{REC}$ reasoner is the following:

- a *commitments model* that contains the rules for manipulation of commitments,
- a *domain model* that contains the protocol rules that describe the agents' domain,
- an *event trace* that contains the actions of the agents throughout time.

Listing 1.1 shows part of the commitments model. First, the states of the commitments are described. Then, the rules that describe the state transitions are defined[2]. In $\mathcal{REC}$, we can express that an event *initiates* (or *terminates*) a temporal *fluent*, by way of *initiates(Event, Fluent, Time)* relations. A commitment with its state is considered a temporal fluent.

[1] The *j*$\mathcal{REC}$ tool is freely available for download from `http://www.inf.unibz.it/~montali/tools.html#jREC`

[2] Note that lines starting with % are comments.

```
% commitment states
conditional(C, T):- holds_at(status(C, conditional), T).
...

% create as conditional
initiates(E, status(C, conditional), T) :- ccreate(E, C, T).

% create as active
initiates(E, status(C, active), T) :- create(E, C, T).

% conditional to active
terminates(E, status(C1, conditional), T) :-
                                        detach(E, C1, C2, T).

initiates(E, status(C1, detached), T) :- detach(E, C1, _, T).

initiates(E, status(C2, active), T) :-   detach(E, _, C2, T).

detach(E, cc(X, Y, Q, P, t(T1)), c(X, Y, P, t(T2)), T):-
 conditional(cc(X, Y, Q, P, t(T1)), T),
 initiates(E, Q, T), T2 is T + T1.
...
```

Listing 1.1. Commitments model

Listing 1.2 shows part of the rules that describe the example domain. Note that an offer from the bank to client creates a conditional commitment between the two agents. The event trace will be given for a sample execution when we present the case study.

```
% fluent manipulation
initiates(exec(pay(Client, Bank, Card)), paid(Card), _) :-
 isClient(Client), isBank(Bank), isCard(Card).
...

% commitment manipulation
ccreate(exec(offer(Bank, Client, Card)),
 cc(Bank, Client, paid(Card), delivered(Card), t(7)), _) :-
 isBank(Bank), isClient(Client), isCard(Card).
...
```

Listing 1.2. Domain model

Given these inputs, $\mathcal{REC}$ produces an outcome that demonstrates the agents' fluents through time. This is used to monitor the individual states of the commitments at run-time [1,16]. However, we are not limited to this. We provide

exception monitoring via the relations among those commitments. Thus, we extend the commitments model with a similarity model and an exception model.

Listing 1.3 shows part of the rules that describe delegation-based similarity, and how improper delegations occur (Section 4). Moreover, Listing 1.4 describes the rules for exceptions.

```
% delegation
explicitDelegation(c(Z, Y, P, _), c(X, Y, P, _)) :- X \= Z.
...

delegation(C1, C2) :- explicitDelegation(C1, C2).
...

% improper delegation
initiates(_, improperDelegation(
  c(X3, Y3, P3, t(T3)), c(X1, Y1, P1, t(T1))), T) :-
 active(c(X1, Y1, P1, t(T1)), T),
 conditional(cc(X2, Y2, Q2, P2, t(T2)), T),
 active(c(X3, Y3, P3, t(T3)), T),
 implicitDelegation(
  cc(X2, Y2, Q2, P2, t(T2)), c(X1, Y1, P1, t(T1))),
 antecedentDelegation(
  c(X3, Y3, P3, t(T3)), cc(X2, Y2, Q2, P2, t(T2))),
 (T2 + T3) > T1.
...
```

Listing 1.3. Similarity model

```
% Monitoring rule 1
initiates(_, exception(C1, C2), T):-
 holds_at(improperDelegation(C1, C2), T).

% Monitoring rule 2
initiates(E, exception(C1, C2), T):-
 holds_at(improperDelegation(C1, C), T),
 active(C2, T), delegation(C, C2).
```

Listing 1.4. Exception model

7 Case Study

Let us consider the protocol in Table 1, represented by 3 commitments. The bank must deliver the credit card within 7 days of the customer's request (see template CC_1^t in $\mathcal{P}_{card}$). When the card is requested, the bank notifies the office for printing the card (CC_3^t). Then, the courier delivers the card to the client (CC_2^t).

Table 1. Acquire credit card ($\mathcal{P}_{card}$)

$$\mathcal{P}_{card} = \begin{cases} CC_1^t(\text{bank, client, requested, delivered(7)}) \\ CC_2^t(\text{courier, bank, printed, delivered(3)}) \\ CC_3^t(\text{office, bank, confirmed, printed(3)}) \end{cases}$$

$$\mathcal{R}_{client} = \begin{cases} CC_1 \\ \text{request(client, bank)} \rightarrow \text{requested} \\ \text{deliver(X, client)} \rightarrow \text{delivered} \end{cases}$$

$$\mathcal{A} = \{\, \text{bank}(\textit{hsbc}),\ \text{client}(\textit{federico}),\ \text{courier}(\textit{ups}),\ \text{office}(\textit{office}) \,\}$$

Notice the client's role; it only includes CC_1^t and two actions, for requesting and getting the card delivered. The last row of Table 1 shows which agents enact the corresponding roles in the protocol. Consider now the following trace:

$$\mathcal{T} = \begin{cases} 4 \quad \text{request}(\textit{federico}, \textit{hsbc}) & \text{(the client requests the credit card from the bank on day 4)} \\ 7 \quad \text{confirm}(\textit{hsbc}, \textit{office}) & \text{(the bank confirms the request)} \\ 10 \ \text{print}(\textit{office}, \textit{ups}) & \text{(the office produces the card and passes it to the courier)} \end{cases}$$

The following commitments are in place at time 11:

$$\mathcal{C}_{11} = \begin{cases} C_1(\text{hsbc, federico, delivered(11)}) \\ CC_2(\text{ups, hsbc, printed, delivered(3)}) \\ C_3(\text{office, hsbc, printed(10)}) \end{cases}$$

Notice the pattern among these three commitments; CC_2 is an implicit delegation of C_1 (Definition 5), and C_3 is an antecedent delegation of CC_2 (Definition 7). Then C_3 is delegation-similar to C_1 via CC_2.

Now assume that no delivery has occurred until time 12. Figure 2 shows the output of j-$\mathcal{REC}$, the Java-based $\mathcal{REC}$ reasoner. The horizontal axis shows the timeline of events. The fluents are positioned vertically, and their truth values (e.g., states for commitments) are computed according to the events. C_1 is indeed violated since its deadline has passed. Because of the similarity relation, CC_2 and C_3's deadlines together affect C_1. Even though the printing of the card is completed at day 10, the courier has 3 more days for delivery, which will eventually exceed C_1's deadline. Indeed, notice in the figure that delivery is completed at time 13, which fulfills the commitment of UPS to HSBC (C_2). However, the commitment of HSBC to Federico (C_1) is violated. We have the fluents *improperDelegation* and *exception*, corresponding to each improper delegation of commitments (both conditional and base-level) in the system. Here, the bank should have confirmed the client's request earlier, and notified the office accordingly.

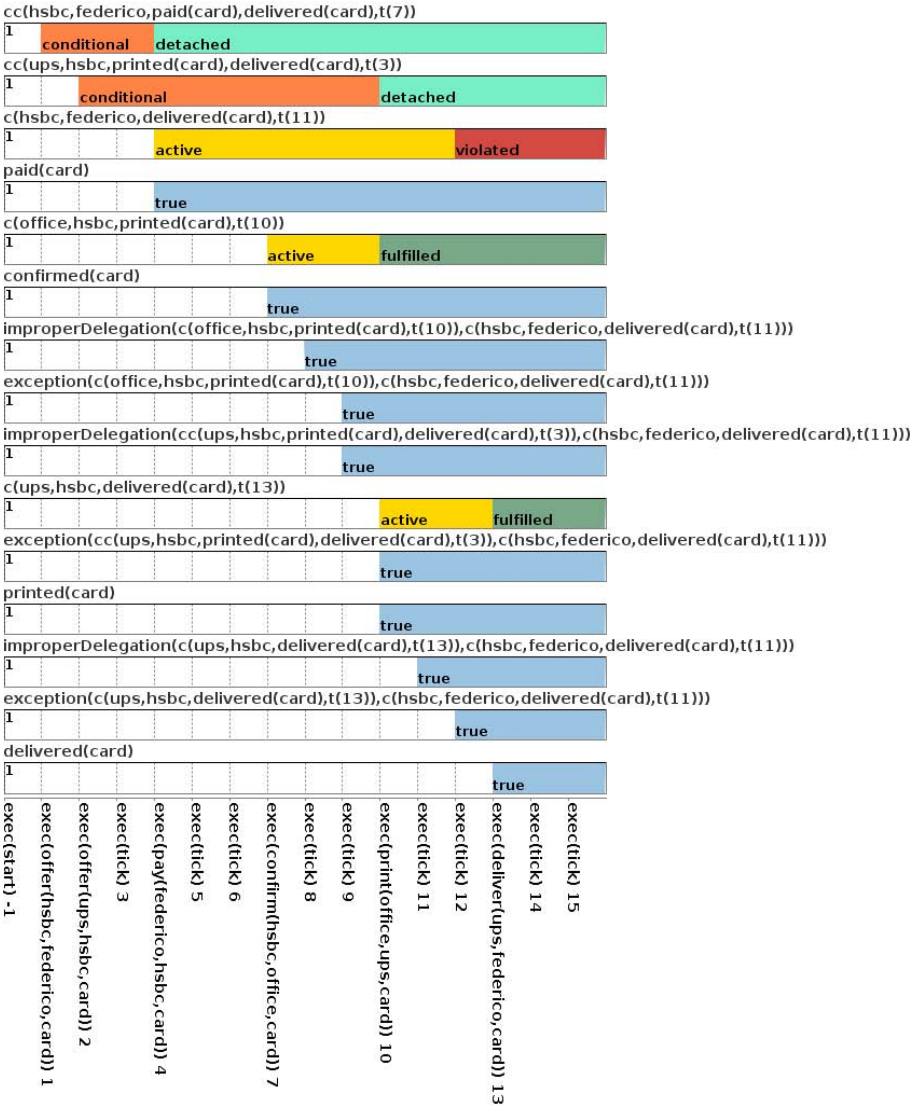


Fig. 2. $\mathcal{REC}$ output

8 Discussion

Work presented in this paper advances the state of the art in several directions. First, we identify the ways that a commitment can be extended with a third party (e.g., a delegatee agent), giving an exhaustive account. We use motivating examples inspired from an e-commerce scenario, to show that delegation can follow

meaningful patterns, other than the usual one or two considered in literature. To the best of our knowledge, no systematic classification of commitment delegation types has ever been done before.

Moreover, we provide similarity relations to connect commitments with one another. The relations we propose are again exhaustive, in the sense that they are capture all possible chains of rational delegation. The similarity relations are ternary relations, which connect two (otherwise unrelated) commitments via another existing commitment called primary. Nested delegations are captured (either explicit or implicit) through similarity. Similarity is fundamental to guide the monitoring process.

Finally, we apply such notions to the problem of handling exceptions in contract-regulated systems. We identify possible reasons of exceptions by considering time-related commitments and ways of delegating such commitments that may bring about inconsistent states. We call such delegations improper. We describe a framework that enables commitment monitoring, at run-time, given a specification of the current system state. The framework is able to identify the improper delegations that cause the exception. We implemented the framework in $\mathcal{REC}$ [1,16].

In principle, some of these notions, which we introduced for the purpose of monitoring, could also be used of auditing, or even at design time. For example, it may be useful to introduce design constraints, to prevent agents from causing improper delegations. We do not deal with design issues here, but as a future work it would be interesting to study the application of the improper delegation notion in contexts other than monitoring.

While we focussed here on the general framework, we did not run a formal analysis of the new notion of delegation. Therefore an important direction for future work would be the definition of a model-theoretic semantics of delegation, following and possibly extending recent results published by Lorini [12] and by others. For example, the similarity relations we defined in Section 4 describe how a single commitment can propagate into two separate delegations. It would be interesting to characterize this notion in terms of some reasoning postulates identified by Singh [15], for example L-DISJOIN and R-CONJOIN. Similarly, it would be important to understand the implications of our characterization of delegation in a normative context, following, e.g,. work done by Gelati et al. [8].

We plan to extend the language of commitment properties with negation and disjunction. Commitments with negated propositions are interesting in the sense that an agent commits to ensure something will not happen. This can be related with a maintenance goal [13,1], where a certain property should hold at all times during a specified interval.

Finally, we intend to design a distributed procedure where agents will collaboratively monitor improper delegations by exchanging their local knowledge. We plan to build on our recent work on commitment diagnosis via a distributed reasoning procedure [9] or to exploit higher-level forms of communication, such as dialogues, as we proposed in [10].

Acknowledgements

The first author is supported by Boğaziçi University Research Fund under grant BAP5694, and the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610. We thank Marco Montali for providing us with a working implementation of $j\mathcal{REC}$, which enabled us to run experiments. We are indebted to the anonymous reviewers for their valuable feedback.

References

1. Chesani, F., Mello, P., Montali, M., Torroni, P.: Commitment tracking via the reactive event calculus. In: IJCAI 2009: 21st International Joint Conference on Artificial Intelligence, pp. 91–96 (2009)
2. Chesani, F., Mello, P., Montali, M., Torroni, P.: A logic-based, reactive calculus of events. Fundamenta Informaticae 105(1-2), 135–161 (2010)
3. Chopra, A.K., Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Reasoning about agents and protocols via goals and commitments. In: AAMAS 2010: 9th International Conference on Autonomous Agents and Multiagent Systems, pp. 457–464 (2010)
4. Chopra, A.K., Singh, M.P.: Constitutive interoperability. In: AAMAS 2008: 7th International Conference on Autonomous Agents and Multiagent Systems, pp. 797–804 (2008)
5. Chopra, A.K., Singh, M.P.: Multiagent commitment alignment. In: AAMAS 2009: 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 937–944 (2009)
6. Dastani, M., Dignum, V., Dignum, F.: Role-assignment in open agent societies. In: AAMAS 2003: 2nd International Conference on Autonomous Agents and Multiagent Systems, pp. 489–496 (2003)
7. Friedrich, G.: Repair of service-based processes – an application area for logic programming. The ALP Newsletter (December 2010), http://www.cs.nmsu.edu/ALP
8. Gelati, J., Rotolo, A., Sartor, G., Governatori, G.: Normative autonomy and normative co-ordination: Declarative power, representation, and mandate. Artificial Intelligence & Law 12(1-2), 53–81 (2004)
9. Kafalı, Ö., Chesani, F., Torroni, P.: What happened to my commitment? Exception diagnosis among misalignment and misbehavior. In: Dix, J., Leite, J., Governatori, G., Jamroga, W. (eds.) CLIMA XI. LNCS, vol. 6245, pp. 82–98. Springer, Heidelberg (2010)
10. Kafalı, Ö., Toni, F., Torroni, P.: Reasoning about exceptions to contracts. In: Leite, F., Torroni, P., Ågotnes, T., Boella, G., van der Torre, L. (eds.) CLIMA XII 2011. LNCS, vol. 6814, pp. 225–242. Springer, Heidelberg (2011)
11. Kafalı, Ö., Torroni, P.: Diagnosing commitments: Delegation revisited (extended abstract). In: AAMAS 2011: 10th International Conference on Autonomous Agents and Multiagent Systems, pp. 1175–1176 (2011)
12. Lorini, E.: A logical analysis of commitment dynamics. In: Governatori, G., Sartor, G. (eds.) DEON 2010. LNCS, vol. 6181, pp. 288–305. Springer, Heidelberg (2010)
13. van Riemsdijk, M.B., Dastani, M., Winikoff, M.: Goals in agent systems: a unifying framework. In: AAMAS 2008: 7th International Conference on Autonomous Agents and Multiagent Systems, pp. 713–720 (2008)

14. Singh, M.P.: An ontology for commitments in multiagent systems: Toward a unification of normative concepts. Artificial Intelligence and Law 7, 97–113 (1999)
15. Singh, M.P.: Semantical considerations on dialectical and practical commitments. In: Fox, D., Gomes, C.P. (eds.) AAAI 2008: 23rd National Conference on Artificial Intelligence, pp. 176–181. AAAI Press, Menlo Park (2008)
16. Torroni, P., Chesani, F., Mello, P., Montali, M.: Social commitments in time: Satisfied or compensated. In: Baldoni, M., Bentahar, J., van Riemsdijk, M.B., Lloyd, J. (eds.) DALT 2009. LNCS, vol. 5948, pp. 228–243. Springer, Heidelberg (2010)
17. Yolum, P., Singh, M.P.: Flexible protocol specification and execution: applying event calculus planning using commitments. In: AAMAS 2002: 1st International Conference on Autonomous Agents and Multiagent Systems, pp. 527–534 (2002)

Verifying Team Formation Protocols with Probabilistic Model Checking*

Taolue Chen, Marta Kwiatkowska, David Parker, and Aistis Simaitis

Computing Laboratory, University of Oxford,
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK

Abstract. Multi-agent systems are an increasingly important software paradigm and in many of its applications agents cooperate to achieve a particular goal. This requires the design of efficient collaboration protocols, a typical example of which is *team formation*. In this paper, we illustrate how probabilistic model checking, a technique for formal verification of probabilistic systems, can be applied to the analysis, design and verification of such protocols. We start by analysing the performance of an existing team formation protocol modelled as a discrete-time Markov chain. Then, using a Markov decision process model, we construct optimal algorithms for team formation. Finally, we use stochastic two-player games to analyse the competitive coalitional setting, in which agents are split into cooperative and hostile classes. We present experimental results from these models using the probabilistic model checking tool PRISM, which we have extended with support for stochastic games.

1 Introduction

Multi-agent systems have become an important software paradigm. One of the key ideas behind this approach is that several different agents can cooperate to achieve certain goals. This requires the design of efficient collaboration protocols, of which *team formation* is a typical example. In this paper, we focus on a distributed team formation protocol introduced in [10]. There, the authors used it to analyse team performance in dynamic networks. The protocol has also been applied to coalition formation for data fusion in sensor networks [11]. In both cases it has been used as a basis for designing other algorithms, which makes it a compelling target for formal analysis.

The basic setting for the protocol of [10] consists of an *agent organisation*, i.e., a network of interconnected agents which have certain resources. These agents attempt to form teams in order to accomplish tasks which are generated periodically and globally advertised to the agent organisation. The topology of the network restricts the set of possible agent teams – for an agent to be on a team, the agent must have a connection with at least one other agent in that team. Tasks are generic in that they only require a team of agents with the necessary resources to accomplish the specific task. As in [10], we do not consider the solution process, but only the team formation.

As is typical for multi-agent algorithms, probabilities play a crucial role in team formation protocols. Firstly, agents are scheduled to act in a random order, following the

* This work is supported by the ERC Advanced Grant VERIWARE.

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 190–207, 2011.

approach of [10]; secondly, in our setting, tasks are drawn from a task pool, following some known probability distribution. This is particularly interesting for the *online* version of the algorithm (see Alg. 3), where tasks are generated after teams have formed. In this case, agents have to choose strategies to optimise against a set of tasks governed by a certain probability distribution rather than a particular task. Finally, probabilities are used to implement strategies of agents themselves, for example random selection of a team to join. These issues motivate the use of analysis techniques that take can take probabilistic behaviour into account.

Formal verification is an approach to check the correctness of a system using rigorous, mathematical reasoning. Fully automated verification techniques such as *model checking* have proved to be widely applicable, including to multi-agent systems [17]. In this paper, as described above, the systems that we study exhibit *probabilistic* behaviour. Thus, we use *probabilistic model checking*, an automated technique for the formal verification of *stochastic* systems.

Probabilistic model checking is based on the construction of a probabilistic model from a precise, high-level description of a system's behaviour. The model is then analysed against one or more formally specified *quantitative* properties, usually expressed in temporal logic. These properties capture not just the *correctness* of the system, but a wide range of measures such as *reliability* or *performance*. We can compute, for example, "the probability that the algorithm successfully terminates within k rounds". By augmenting the model with *rewards*, a further range of properties can be analysed.

In addition to offering convenient high-level formalisms for representing models and their properties, the strength of probabilistic model checking is that offers *exact*, *exhaustive* analysis techniques. Rather than, for example, discrete-event simulation (as it is done for team formation protocols in [10]), probabilistic model checking is based on an exhaustive exploration and numerical solution of the model, allowing best- and worst-case behaviour to be identified. This is particularly valuable for distributed protocols (like the ones in this paper), whose behaviour is notoriously difficult to understand precisely. Furthermore, efficient techniques and tools exist for this purpose.

In this paper, we use the PRISM probabilistic model checker [16] to analyse various agent organisations for the team formation protocol of [10]. We use several different types of probabilistic models and express quantitative performance properties of them in temporal logic. Firstly, we model the original version of the protocol using discrete-time Markov chains (DTMCs), where the behaviour of each agent is described entirely in a probabilistic (deterministic) way. Then, we extend the original algorithm by allowing agents to make decisions nondeterministically, instead of randomly, when forming teams; such systems are naturally modelled by Markov decision processes (MDPs). By analysing the MDP, we obtain the best- and worst-case performance of agent organisations. MDPs, however, can only model fully collaborative behaviour, whereas in many scenarios it is crucial to address hostile behaviour of some agents in the organisation. To cater for this we use stochastic two-player games (STPGs) as a model for the system containing two groups of agents – *collaborative* and *hostile* – which try to, respectively, maximise or minimise the performance of the organisation (i.e. this effectively becomes a zero-sum stochastic two-player game). Orthogonal to these, we consider two

different settings, namely *offline* and *online*, depending on whether the tasks are generated respectively *before* and *after* teams have formed (see Alg. 3).

Our experiments illustrate several aspects of agent organisation analysis. As a typical case, we choose four network topologies, each consisting of five agents, i.e., fully connected, ring, star, and a network having one isolated agent. For each one, we compute the expected performance of the organisation and find organisation-optimal resource allocation among agents. Then we show using MDP model checking what is the best performance that can be achieved by this organisation. Lastly, we take the model to the STPG setting to obtain the optimal coalitions of different sizes and evaluate their performance. For all of these cases, we consider the offline and online dichotomy.

In summary, the main contributions of this paper are as follows:

(1) We perform a comprehensive and *formal* analysis of the performance of the team formation protocol proposed in [10].
(2) We *extend* the original algorithm of [10], allowing agents to make decisions non-deterministically when forming teams. Then, by modelling and analysing as an MDP, we *synthesise* the best strategies for agents to achieve optimal performance, partially solving an open problem posed in [10].[1]
(3) We *extend* the PRISM model checker with support for modelling and automated analysis of STPGs and *address the competitive coalitional setting*, in which agents are split into cooperative and hostile classes, using *stochastic games* to synthesise optimal agent coalitions. To the best of our knowledge, this is the first work to perform a *fully-automated* probabilistic analysis of this kind.

We note that it would be difficult to achieve (2) and (3) using simulation-based approaches; this demonstrates the strength of formal verification.

Related work. Cooperative behaviour, which is one of the greatest advantages of agent-based computing, has been studied from many different angles over the years. Coalitional games have traditionally been analysed from a game-theoretic perspective [19], but in recent years have attracted a lot of attention from researchers in artificial intelligence, especially in cooperative task completion [20]. Several approaches for team formation and collaborative task solving have been considered including team formation under uncertainty using simple heuristic rules [13], reinforcement learning techniques [1] and methods using distributed graph algorithms [18]. To reason formally about cooperative games, several logics (e.g., Alternating Time Logic [3], Coalitional Game Logic [2], Strategy Logic [6]) and other formalisms (e.g., Cooperative Boolean Games [8]) have been introduced and used to analyse coalitional behaviour [5]. Model checking has been used to analyse (non-probabilistic) knowledge-based properties of multi-agent systems, using the tool MCMAS [17]. Probabilistic model checking was employed to analyse probabilistic agents in negotiation protocols [4] (but only for fixed strategies modelled as DTMCs) and to Byzantine agreement protocols [14].

[1] We quote: "the problem of developing or learning effective team initialising and team joining policies is also important, and is included in our on-going and future work".

2 Preliminaries

2.1 Probabilistic Models

We begin with a brief introduction to the three different types of probabilistic models that we will use in this paper.

Discrete-time Markov chains (DTMCs) are the simplest of these models. A DTMC $(S, \mathbf{P})$ is defined by a set of states S and a probability transition matrix $\mathbf{P} : S \times S \rightarrow [0, 1]$, where $\sum_{s' \in S} \mathbf{P}(s, s') = 1$ for all $s \in S$. This gives the probability $\mathbf{P}(s, s')$ that a transition will take place from state s to state s'.

Markov decision processes (MDPs) extend DTMCs by incorporating *nondeterministic choice* in addition to probabilistic behaviour. An MDP $(S, Act, Steps)$ comprises a set of actions Act and a (partial) probabilistic transition function $Steps : S \times Act \rightarrow Dist(S)$, which maps state-action pairs to probability distributions over the state space S. In each state $s \in S$, one or more distinct actions can be taken and, assuming that action $a \in Act$ is chosen, the distribution $Steps(s, a)$ gives the probability of making a transition to each state.

Stochastic two-player games (STPGs) generalise MDPs by allowing the nondeterministic choices in the model to be resolved by two distinct players. An STPG is a tuple $(S, (S_1, S_2), Act, Steps)$ where the set of states S is partitioned into two disjoint subsets S_1 and S_2. As for MDPs, $Steps : S \times Act \rightarrow Dist(S)$ is a function mapping state-action pairs to distributions over states. This is a *turn-based* game: in each state s of the game, either player 1 or player 2 selects an action $a \in Act$, depending on whether s is in set S_1 or S_2.

2.2 Probabilistic Model Checking and PRISM

Probabilistic model checking involves the construction and analysis of a probabilistic model. Usually, a high-level description language is used to model a system (here, we use the PRISM [16] modelling language). Then, one or more quantitative properties are formally specified and analysed on the model. Typically, *probabilistic temporal logics* are used to formalise properties. In this paper, we use PRISM's temporal logic-based query language, which is essentially the logic PCTL [12], extended to include reward-based properties [15, 9]. This can be used to express properties of both DTMCs and MDPs. We also generalise the logic to capture properties of stochastic games.

PCTL extends the well known temporal logic CTL with a probabilistic (P) operator. Informally, this places bounds on the probability of the occurrence of certain events in the model. We will illustrate the use of PRISM temporal logic queries, using some simple examples, referring the reader to [12, 15] for precise details of the syntax and semantics. For a DTMC, typical queries would be:

- $\mathtt{P}_{<0.01}[\,\Diamond\, fail\,]$ - "the probability of a failure occurring is less than 0.01"
- $\mathtt{P}_{\geq 0.95}[\,\Diamond\, end\,]$ - "the probability of the protocol terminating is at least 0.95".

For simplicity, we restrict our attention to *reachability* queries (in the examples above, $fail$ is a *label*, denoting a particular subset of the DTMC's states S and $\Diamond\, fail$ refers to the event in which a state from this set is reached). In practice, we often use a *quantitative* variant of the $\mathtt{P}$ operator, denoted $\mathtt{P}_{=?}$, which returns the actual probability of an event's occurrence, e.g.:

– $\mathtt{P}_{=?}[\,\Diamond\, fail\,]$ - "what is the probability of a failure occurring?"

Whereas in a DTMC it is relatively straightforward to define the probability of an event such as $\Diamond\, fail$, for MDPs we must also take account of the nondeterminism in the model. The standard approach is to use the notion of *strategies* (also referred to as *policies*, *schedulers*, etc.). A strategy resolves nondeterminism in an MDP (i.e. chooses an action in a state), based on its execution history. For a specific strategy, we *can* define the probability of an event. Thus, probabilistic model checking focuses on best- or worst-case analysis, quantifying over all possible strategies. We still employ quantitative properties, which now ask for *minimum* or *maximum* probabilities:

– $\mathtt{P}_{\max=?}[\,\Diamond\, fail\,]$ - "what is the maximum probability of a failure occurring?"

For a stochastic game, the same approach generalises naturally, but we require strategies for both players. Usually, we assume that the two players have opposing objectives, for example player 1 aims to minimise the probability of $\Diamond\, fail$ and player 2 tries to maximise it. Extending the notation from above, we write:

– $\mathtt{P}_{\min,\max=?}[\,\Diamond\, fail\,]$ - "what is the minimum probability of failure that player 1 can guarantee, assuming that player 2 tries to maximise it?"

For this simple class of (zero-sum) properties, these values are well defined [7].

Finally, we also use properties based on *rewards*, which capture a variety of additional quantitative measures. For consistency across all three types of models, we assume a simple state-based scheme, i.e., a *reward function* $\rho : S \to \mathbb{R}_{\geq 0}$. We consider the *expected total reward* accumulated until some target set of states is reached. Consider a DTMC with reward function $time$ and a label end denoting a set of target states. We write, for example:

– $\mathtt{R}^{time}_{=?}[\,\Diamond\, end\,]$ - "what is the expected time for the algorithm to complete?"

In exactly the same style as above, these queries generalise to MDPs and STPGs:

– $\mathtt{R}^{time}_{\max=?}[\,\Diamond\, end\,]$ - "what is the maximum expected algorithm completion time?"
– $\mathtt{R}^{time}_{\min,\max=?}[\,\Diamond\, end\,]$ - "what is the minimum expected time for algorithm completion that player 1 can guarantee, assuming player 2 tries to maximise it?"

PRISM [16] is a probabilistic model checker. It supports several different types of models, including DTMCs and MDPs (it also supports continuous-time Markov chains and probabilistic timed automata). On these models, a wide range of temporal logic-based properties can be checked, including all of those illustrated above. For the work presented here, we have built a prototype extension of PRISM that adds support for STPGs in the form of solving turn-based stochastic two-player zero-sum games (i.e. model-checking temporal formulae of the form described above). Models to be analysed by

PRISM are described in a high-level modelling language based on guarded command notation; we discuss this further in Section 4.1.

3 Definitions and Algorithms

The purpose of this section is to provide definitions of terminology used throughout this paper and then present the algorithms which will be analysed.

3.1 Definitions

We introduce definitions of *agent organisations*, *tasks*, *teams*, and formulae for computing *rewards* to measure the performance of both individual agents and agent teams.

Definition 1 (Agent Organisation). *An* agent organisation *is a tuple* $O = \langle A, N, R, R_A \rangle$ *where:*

- $A = \{a_1, a_2, ..., a_n\}$ *is a set of agents,*
- $N = \{\{a_i, a_j\} : \text{“}a_i \text{ and } a_j \text{ are neighbours”}\}$ *is a neighbourhood relation,*
- $R = \{r_1, r_2, \ldots, r_k\}$ *is a set of resource types, and*
- $R_A = \{R_{a_1}, R_{a_2}, \ldots, R_{a_n}\}$ *is a set of agent resources where* $r_j \in R_{a_i} \iff$ *“agent* a_i *has a resource* r_j*”.*

Definition 2 (Task). *A* task $T_i = \{r_i : \text{“}r_i \text{ is required by the task } i\text{”}\}$ *is a set of resources that are required to accomplish* T_i*. By* $T = \{T_1, T_2, \ldots, T_t\}$ *we denote a collection of tasks.*

Definition 3 (Team). *A* team *of agents is denoted by* $M_i = \{a_j : \text{“}a_j \text{ is a member of team } i\text{”}\}$*, and the set of all teams is* $M = \{M_1, M_2, \ldots, M_m\}$*. By* $\bar{M} = \bigcup_{1 \leq i \leq m} M_i$*, we denote the set of all agents that are committed to some team. For* $1 \leq i \leq m$*,* $R_{M_i} = \bigcup_{a \in M_i} R_a$ *is the set of resources the team* M_i *has. The team* M_i *is said to be able to accomplish the task* T_j *iff* $T_j \subseteq R_{M_i}$*.*

Definition 4 (Rewards). *For agent* a*, we define two types of* reward*:*

- *Type* W_1*, which rewards the* agent *with 1 point if it is in the team which was able to complete its task after team formation is over; and 0 otherwise. Formally,*

$$W_1(a) = \begin{cases} 1 & \text{if } \exists M_i . a \in M_i \wedge T_i \subseteq R_{M_i}, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

- *Type* W_2*, which rewards 1 point to the* team *which was able to complete its task, and 0 otherwise. The reward is shared equally between team members.*

$$W_2(a) = \begin{cases} \frac{1}{|M_i|} & \text{if } \exists M_i . a \in M_i \wedge T_i \subseteq R_{M_i}, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

For a set of agents A, the rewards are defined accordingly as the total reward achieved by its members, i.e.,

$$W_1(A) = \sum_{a \in A} W_1(a) \qquad W_2(A) = \sum_{a \in A} W_2(a). \tag{3}$$

The underlying idea of these two types of rewards is that W_2 provides incentives for agents to form smaller teams which can accomplish tasks, whereas W_1 motivates agents to be in a successful team. From the organisation's perspective, the W_1 reward should be used when resources are limited, whereas the W_2 reward will encourage agents to introduce resource redundancy into teams, but this may ensure that tasks are completed with higher probabilities.

3.2 Algorithms

In this section we provide pseudocode for the algorithms which we later analyse. During the team formation process, each agent performs as follows: when it is not committed to any team (meaning that it is available and not assigned to any task), it considers each task in a random order. If a task currently has no other agents committed to it, the agent can choose to initialise a team, and does so with the probability given in Eqn. (4) (i.e., the ratio between neighbours that are not committed to any team and total number of neighbours).

$$IP_a = \frac{|\{a' \in A \,:\, \{a, a'\} \in N \wedge a' \notin \bar{M}\}|}{|\{a' \in A \,:\, \{a, a'\} \in N\}|}. \tag{4}$$

For team joining, if an agent is eligible for a team, it always joins the team. Note that only uncommitted agents can commit to a new or partially filled task, and committed agents can not decommit from a given task.

In Alg. 1 we reproduce pseudocode for the JOINTEAM algorithm introduced in [10]. This combines team initialisation and team joining. This algorithm will be modelled and analysed as a DTMC, as we shall see in Sec. 5.1.

Algorithm 1. JOINTEAM algorithm [10] (probabilistic and deterministic)

```
procedure JoinTeam(a, T, M)
    for all T_i ∈ T in random order do
        if a ∉ M̄ then                                          ▷ agent is not committed
            if |M_i| = 0 then                                  ▷ team for task i is empty
                if R_a ∩ T_i ≠ ∅ then        ▷ agent has skill (replaced by true if called from Online, cf. Alg. 3)
                    with probability IP_a: M_i ← M_i ∪ {a}     ▷ initialise a team (see Eqn. (4))
                end if
            else if ∃{a, a'} ∈ N.a' ∈ M_i then                 ▷ there is neighbour in team for task i
                if R_a ∩ T_i \ R_{M_i} ≠ ∅ then  ▷ agent has a missing resource (replaced by true if called from Online)
                    M_i ← M_i ∪ {a}                            ▷ join team
                end if
            end if
        end if
    end for
end procedure
```

To tackle the problem of finding the best team initialisation and team joining strategy, we modify the original JOINTEAM algorithm by allowing agents to make decisions regarding what actions to take, instead of picking one randomly. Technically, the changes are as follows, which are highlighted in Alg. 2.

Algorithm 2. JOINTEAM algorithm (non-deterministic extension)

```
procedure JoinTeam(a, T, M)
    for all T_i ∈ T in arbitrary order do
        if a ∉ M̄ then                                          ▷ agent is not committed
            if |M_i| = 0 then                                  ▷ team for task i is empty
                if R_a ∩ T_i ≠ ∅ then        ▷ agent has skill (replaced by true if called from Online, cf. Alg. 3)
                    M_i ← M_i ∪ {a} or M_i ← M_i               ▷ initialise a team or do nothing
                end if
            else if ∃{a, a'} ∈ N.a' ∈ M_i then                 ▷ there is neighbour in team for task i
                if R_a ∩ T_i \ R_{M_i} ≠ ∅ then  ▷ agent has a missing resource (replaced by true if called from Online)
                    M_i ← M_i ∪ {a} or M_i ← M_i               ▷ join a team or do nothing
                end if
            end if
        end if
    end for
end procedure
```

– Allow agents to consider tasks in arbitrary order instead of randomly;
– Replace probabilistic choice to initialise the team by nondeterministic choice;
– Allow agent *not* to join a team even if it has a resource and neighbour in that team.

This algorithm allows analysis of the best-case performance that can be achieved by the protocol. It also allows us to analyse agent organisations with *hostile* agents, which aim to reduce organisation's performance. It will be modelled and analysed as an MDP and STPG respectively, as we shall see in Sec. 5.2 and Sec. 5.3. Furthermore, we observe that there are two natural ways to call JOINTEAM: the OFFLINE procedure first initialises the set of tasks and then sequentially calls the JOINTEAM procedures of every agent in random order, as described in Alg. 1. In contrast, the ONLINE routine calls JOINTEAM procedures for agents *before* selecting the tasks. (The JOINTEAM algorithm needs to be adapted slightly, see Alg. 1, the 5th line.) We investigate both the offline and online versions of the algorithms because they provide a nice comparison between optimisation against specific tasks (offline), and distribution of tasks (online). As we will see in Sec. 5.3, whether the offline or online version results in better performance depends on network topology.

Algorithm 3. Offline and online versions of JOINTEAM algorithm

```
procedure Offline(t)                                           ▷ t - number of tasks
    M = {M_i = ∅ : 1 ≤ i ≤ t}                                  ▷ initialise empty teams
    T = {T_i ≠ ∅ : T_i ⊆_random R, 1 ≤ i ≤ t}                   ▷ initialise tasks at random
    for all a ∈ A in random order do
        JoinTeam(a, T, M)
    end for
    perform tasks and compute rewards
end procedure

procedure Online(t)                                            ▷ t - number of tasks
    M = {M_i = ∅ : 1 ≤ i ≤ t}                                  ▷ initialise empty teams
    for all a ∈ A in random order do
        JoinTeam(a, T, M)
    end for
    T = {T_i ≠ ∅ : T_i ⊆_random R, 1 ≤ i ≤ t}                   ▷ initialise tasks at random
    perform tasks and compute rewards
end procedure
```

4 Models and Experimental Setup

4.1 PRISM Models

The PRISM model checker has been briefly described above. The purpose of this section is to explain how we model the algorithms from the previous section in PRISM. Due to space limitations we do not provide the source code of the models and properties used in this paper; instead, we have made them all available online[2]. Here we use a toy example from Fig. 1 to illustrate the design concepts. The system modelled in this example consists of two agents and a scheduling module which randomly generates the number of tasks to be performed (1 or 2, each with probability 0.5). Then, agents act in turn by choosing which team to join for each task. The reward structure *"total"* rewards 0.3 points for each task for which agents joined different teams and 1.0 points when agents cooperate. The choice of teams for an agent is nondeterministic (i.e. the underlying model will be either an MDP or STPG), but it could be made probabilistic in a way similar to the scheduler's generation of tasks (and thus become a DTMC).

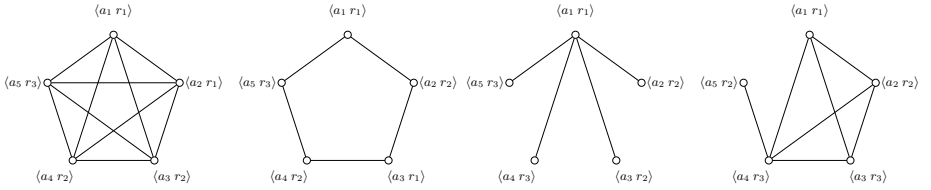
```
module scheduler
    turn : [1..3] init 1;
    num_tasks : [-1..2] init -1;
    [gen] num_tasks=-1 → 0.5 : (num_tasks'=1) + 0.5 : (num_tasks'=2);
    [go1] num_tasks>0 ∧ turn=1 → (turn'=2);
    [go2] num_tasks>0 ∧ turn=2 → (turn'=3);
    [do] num_tasks>0 ∧ turn=3 → (turn'=1) ∧ (num_tasks'=num_tasks − 1);
endmodule

module agent1
    team1 : [1..2] init 1;
    [go1] true → (team1'=1);
    [go1] true → (team1'=2);
endmodule

module agent2 = agent1 [go1=go2, team1=team2] endmodule

rewards "total"
    turn=3 ∧ team1≠team2 : 0.3;
    turn=3 ∧ team1=team2 : 1.0;
endrewards
```

Fig. 1. Example of a two agent system, described in PRISM's guarded command modelling language; see [16] for further details and links

The same principle has been applied to the models that we used for experiments. Each agent is modelled as a module with Alg. 1 and Alg. 2 encoded as guarded commands. There is a scheduler module which has Alg. 3 implemented as guarded commands. The reward structures are also described according to definitions in Eqn. (1)-(3).

From this high-level description of the algorithm, our extension of PRISM then constructs the corresponding models: DTMC, MDP, or STPG. For STPGs, we also need to specify the split of model states into those controlled by players 1 and 2. This is done with two expressions over PRISM model variables, describing these sets.

[2] See http://www.prismmodelchecker.org/files/clima11/



(a) Fully connected ($\mathbf{O}_{fc}$) (b) Ring ($\mathbf{O}_{r}$) (c) Star ($\mathbf{O}_{s}$) (d) Isolated agent ($\mathbf{O}_{ia}$)

Fig. 2. Experimental configurations of the agent organisations with optimal resource allocation (see Tab. 2). In parentheses is the abbreviation that we will use to refer to the organisation throughout this paper.

4.2 Experimental Setup

For our experiments we mainly consider organisations consisting of five agents which are organised into four networks: fully connected, ring, star, and a network having one isolated agent. Each agent is assigned one resource, and there are three different resources available. For each network, we find the optimal resource allocation with respect to task generation described[3] below using DTMC model checking (see Sec. 5). These organisations are then fixed and used in all experiments.

Agent Organisations. We run experiments with a set of five agents $A = \{a_1, a_2, a_3, a_4, a_5\}$ and a set of three resources $R = \{r_1, r_2, r_3\}$ arranged into four different agent organisations O_{fc}, O_r, O_s, O_{ia} (see Fig. 2 for graphical representations of these).

Tasks. We fix seven different tasks that will be used in experiments $T = \{\{r_1\}, \{r_2\}, \{r_3\}, \{r_1, r_2\}, \{r_1, r_3\}, \{r_2, r_3\}, \{r_1, r_2, r_3\}\}$. When running the algorithm, two tasks T_1 and T_2 are picked uniformly and independently at random (with replacement) and are advertised to the agent organisation. So, there are a total of 49 different combinations of T_1 and T_2 that can be generated.

5 Experimental Results

In this section, we present results obtained using three models: DTMC, MDP, and STPG. Tab. 1 compares model construction information for different sizes of fully connected agent organisations[4]. All experiments are performed on a 2.8GHz Intel Core 2 PC, 4Gb of RAM running the Fedora Core 13 operating system. Nondeterministic models (MDPs/STPGs) have a smaller state space because agent choices do not have to be resolved at the model construction stage. However, the model checking is generally more time consuming for MDPs and STPGs than for DTMCs. The time needed for model checking is in the range of 5-240 seconds.

[3] We have chosen this way of allocating resources in order to easily show how the performance can be improved by changing the strategy while keeping actions unchanged (i.e. compare DTMC and MDP models).

[4] We choose a fully connected agent organisation because it produces the largest models.

Table 1. Model comparison for different numbers of agents in a fully connected agent organisation for the offline version of Alg. 1

Agents	States	Transitions	Constr. Time (s)
2	1865	2256	0.1
3	17041	20904	0.3
4	184753	226736	3.4
5	2366305	2893536	74.4
6	35058241	42638400	2916.2

(a) DTMC

Agents	States	Transitions	Constr. Time (s)
2	1405	1846	0.1
3	9721	12474	0.2
4	76865	96664	1.1
5	731233	907992	5.1
6	8155873	10040112	29.7

(b) MDP and STPG

Table 2. Optimal resource allocations with respect to rewards defined in Eqn. (3). All satisfy the constraint $\forall i.|R_{a_i}| = 1 \wedge \forall i.1 \leq |\{R_{a_j} : r_i \in R_{a_j}(1 \leq j \leq 5)\}| \leq 2$.

Organisation O	Additional constraints	Example $\langle R_{a_1} R_{a_2} R_{a_3} R_{a_4} R_{a_5} \rangle$
O_{fc}	-	$R_A = \langle \{r_1\}\{r_1\}\{r_2\}\{r_2\}\{r_3\} \rangle$
O_r	$R_{a_1} \neq R_{a_5} \wedge \forall i < 5.R_{a_i} \neq R_{a_{i+1}}$	$R_A = \langle \{r_1\}\{r_2\}\{r_1\}\{r_2\}\{r_3\} \rangle$
O_s	$R_{a_1} = \{r\} \wedge \forall i > 1.r \notin R_{a_i}$	$R_A = \langle \{r_1\}\{r_2\}\{r_2\}\{r_3\}\{r_3\} \rangle$
O_{ia}	$R_{a_5} = \{r\} \wedge \exists i < 4.r \in R_{a_i}$	$R_A = \langle \{r_1\}\{r_2\}\{r_3\}\{r_3\}\{r_2\} \rangle$

As mentioned in Sec. 4.2, for each topology from Fig. 2 we obtain optimal resource allocations using probabilistic model checking on the DTMC model of the offline version of the algorithm (see Alg. 3 and Alg. 1). The following PRISM temporal logic queries are used to compute the expected rewards of the agent organisation under a particular resource allocation:

– $\mathtt{R}^{W_j}_{=?}[\,\Diamond\, \mathit{finished}\,]$ - "what is the expected total reward W_j of an agent organisation when execution terminates?" ($j \in \{1, 2\}$).

After obtaining the expected rewards for all possible resource allocations, we selected the one with the highest expected reward. The results are summarised in Tab. 2. The resource allocations given in column "Example" of Tab. 2 will be used for all future experiments and are shown in Fig. 2. We decided to fix resource allocations in this way in order to show how model-checking techniques can be used to improve algorithm performance by synthesising strategies (see discussion of MDP results in Sec. 5.2).

5.1 DTMC Analysis

In this section, we present the results for model checking the DTMC model of Alg. 1 for experimental agent organisations from Fig. 2, as well as offline and online versions of the algorithm (see Alg. 3 for details).

Tab. 3 shows the results obtained for the expected rewards of agent organisations in different settings, namely, using W_1 and W_2 reward structures (see Eqn. (3) for organisations and Eqn. (1) and Eqn. (2) for individual agents), and offline and online versions of Alg. 1. The following PRISM temporal logic queries were used to obtain the results:

Table 3. Model checking results for agent organisations from Fig. 2 with optimal resource allocations from Tab. 2 for offline and online versions of Alg. 3. Tables also show largest and smallest individual agent rewards. For a histogram view of the total reward data, see Fig. 3.

O	$W_1(O)$	$\min_{a\in A} W_1(a)$	$\max_{a\in A} W_1(a)$
O_{fc}	2.54906	0.44958	0.75073
O_r	2.30359	0.35494	0.63985
O_s	1.87278	0.28677	0.72568
O_{ia}	2.38529	0.28867	0.68769

(a) Offline. W_1 reward structure.

O	$W_2(O)$	$\min_{a\in A} W_2(a)$	$\max_{a\in A} W_2(a)$
O_{fc}	1.49125	0.26721	0.42238
O_r	1.42923	0.23531	0.38625
O_s	1.16649	0.18582	0.42321
O_{ia}	1.43599	0.20621	0.39907

(b) Offline. W_2 reward structure.

O	$W_1(O)$	$\min_{a\in A} W_1(a)$	$\max_{a\in A} W_1(a)$
O_{fc}	3.53645	0.64101	0.97239
O_r	3.48638	0.55089	0.91190
O_s	2.52500	0.41934	0.84761
O_{ia}	3.37359	0.41186	0.93601

(c) Online. W_1 reward structure.

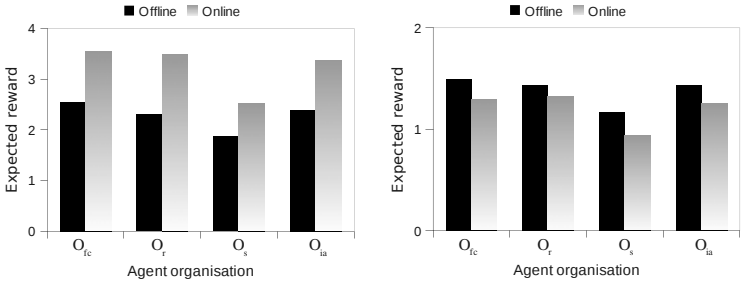
O	$W_2(O)$	$\min_{a\in A} W_2(a)$	$\max_{a\in A} W_2(a)$
O_{fc}	1.29743	0.24247	0.32657
O_r	1.31882	0.23157	0.31297
O_s	0.94404	0.16060	0.30158
O_{ia}	1.25560	0.17970	0.31990

(d) Online. W_2 reward structure.

- $\mathtt{R}^{W_j}_{=?}[\,\Diamond\, \mathit{finished}\,]$ - "what is the expected total reward W_j?" ($j \in \{1,2\}$),
- $\mathtt{R}^{W_j}_{=?}[\,\Diamond\, (\mathit{finished} \wedge a_i{\in}\bar{M})\,]$ - "what is the expected reward W_j for agent a_i?" ($j \in \{1,2\}, i \in \{1,2,3,4,5\}$).

As can be seen in Tab. 3, agents organised in O_s have the worst expected rewards in all settings, and also the largest disparity between the worst and best performing individual agents. Both of these characteristics are not surprising because agent a_1, which is placed in the middle, is most likely to be in a winning team, whereas the others do not have any choice but to join the team with a_1. Organisation O_{ia}, which has one isolated agent, shows a smaller differences between the worst and the best performing agents, but this is only because the performance of the "best" agents is lower, whereas the "worst" agent's performance is very close to that of O_s.

Fig. 3 compares total rewards of all organisations in offline and online settings. It can be seen that a fully connected organisation O_{fc} has the best overall performance in all but the online version using the W_2 reward structure, where it is outperformed by O_r. It is interesting to note that a more significant difference between O_{fc} and O_{ia} only emerges when moving to the online setting. This shows that having one agent which is isolated does not affect the ability of the organisation to efficiently respond to the generated tasks, but impacts its chances to organise before the tasks are generated. This is where the advantages of O_r emerge: in the online setting, it not only starts outperforming O_{ia} with respect to overall performance and disparity, but gets very close to the performance of O_{fc} using the W_1 reward structure, and produces a better total W_2 reward while keeping similar disparity levels. An attentive reader would have noticed that the online version produces larger expected W_1, but smaller expected W_2 rewards. This observation shows that, in an online version, the algorithm organises agents into teams that increase the expectation for more agents to be in a successful team, but decrease the expected total number of tasks completed. This is summarised in Tab. 4.



(a) Reward $W_1(O)$. (b) Reward $W_2(O)$.

Fig. 3. Expected rewards for agent organisations when using online and offline (see Alg. 3) versions of Alg. 1

Table 4. Task completion probabilities for optimal agent organisations using Alg. 1's offline and online versions (see Alg. 3)

O	T_1 compl.	T_2 compl.	T_1 and T_2 compl.
O_{fc}	0.74562	0.74562	0.49596
O_r	0.71461	0.71461	0.47062
O_s	0.58324	0.58324	0.23639
O_{ia}	0.71799	0.71799	0.44839

(a) Offline

O	T_1 compl.	T_2 compl.	T_1 and T_2 compl.
O_{fc}	0.64871	0.64871	0.31320
O_r	0.65941	0.65941	0.36712
O_s	0.47202	0.47202	0.07465
O_{ia}	0.62780	0.62780	0.29270

(b) Online

which shows the task completion probabilities. The following PRISM temporal logic queries were used to find the probabilities:

- $\mathtt{P}_{=?}[\,\Diamond\, T_j\_done\,]$ - "what is the probability to complete task T_j?" ($j \in \{1, 2\}$),
- $\mathtt{P}_{=?}[\,\Diamond\,(T_1\_done \wedge T_2\_done)\,]$ - "what is the probability to complete both tasks?".

Using formal analysis for DTMCs, we produced *exact* values of expectations for properties of the system (task completion probabilities and rewards were used as examples), so that even small differences between different organisations can be captured precisely. Also, we focused on one particular strategy defined in Alg. 1, but the PRISM code can be adapted easily to analyse other strategies and reward structures. In the next section we will explore how the performance can be improved by changing the strategy of the agents so that they collaborate to optimise the performance of the organisation.

5.2 MDP Analysis

In this section, we present the analysis of Alg. 2, which is modelled as an MDP. Using PRISM we find the maximum expected rewards and task completion probabilities for all agent organisations and compare the results with the strategy used in Alg. 1. This is achieved by synthesizing the optimal strategy for the MDP using PRISM and then model-checking the formulae on the resulting DTMC. Due to space limitations we do

Table 5. Maximum task completion probabilities for optimal agent organisations using Alg. 2's online and offline versions (see Alg. 3)

O	T_1 compl.	T_2 compl.	T_1 and T_2 compl.
O_{fc}	1.0	1.0	0.67346
O_r	1.0	1.0	0.67346
O_s	0.82857	0.82857	0.39183
O_{ia}	1.0	1.0	0.67346

(a) Offline

O	T_1 compl.	T_2 compl.	T_1 and T_2 compl.
O_{fc}	1.0	1.0	0.42857
O_r	1.0	1.0	0.42857
O_s	0.88571	0.88571	0.12653
O_{ia}	1.0	1.0	0.42857

(b) Online

not present the actual strategies here.[5] In Tab. 5 we can see the maximum expected task completion probabilities that can be achieved. All organisations except O_s can ensure that at least one task is completed with probability 1.0, no matter what the scheduling is. The following PRISM temporal logic queries were used to get the results:

- $\mathtt{P}_{\max=?}[\,\Diamond\, T_j\_done\,]$ - "what is the maximum probability to complete task T_j?" ($j \in \{1,2\}$),
- $\mathtt{P}_{\max=?}[\,\Diamond\,(T_1\_done \wedge T_2\_done)\,]$ - "what is the maximum probability to complete both tasks?".

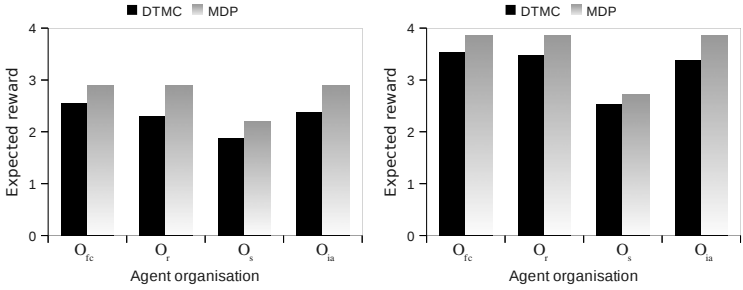
Fig. 4 compares maximum expected rewards for Alg. 1 that can be achieved by all agents collaborating. It is not very difficult to see that O_{fc}, O_r and O_{ia} have the same maximum reward, no matter whether W_1/W_2 reward or online/offline version is taken. These outperform the star organization O_s in all circumstances. More significant improvement can be obtained for the offline version for both rewards than for the online version. This result shows that there is more potential for collaboration for agents in the offline version. The small performance improvement for the online version suggests that the original strategy of Alg. 1 is close to optimal when teams are formed before tasks are advertised.

The PRISM temporal logic queries used to find the rewards were the following:

- $\mathtt{R}^{W_j}_{\max=?}[\,\Diamond\, finished\,]$ - "what is the maximum expected total reward W_j?" ($j \in \{1,2\}$),
- $\mathtt{R}^{W_j}_{\max=?}[\,\Diamond\,(finished \wedge a_i{\in}\bar{M})\,]$ - "what is the maximum expected reward W_j for agent a_i?" ($j \in \{1,2\}, i \in \{1,2,3,4,5\}$).

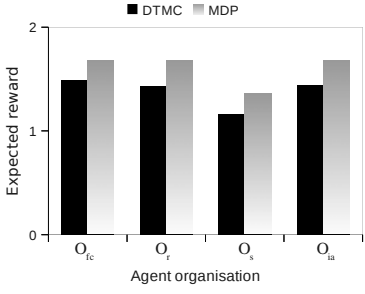
Using MDP model checking we have obtained the optimal strategy for the protocol and compared its performance to Alg. 1. The designer could choose to synthesize and use this strategy but it is worth noting that, even if the designer chooses not to implement this particular strategy, it still serves as a "best-case" benchmark for measuring other algorithms' performance. This analysis allows us to evaluate the effects of a fully collaborative behaviour of agents. However often only limited collaboration can be achieved. In order to facilitate analysis of such systems, one has to go beyond MDPs. In the next section we show how STPGs can be used for this purpose.

[5] The instructions on how to generate optimal strategies for MDPs can be found at: http://www.prismmodelchecker.org/manual/RunningPRISM/Adversaries

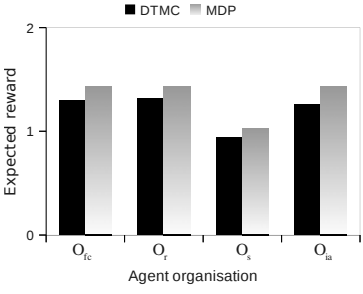


(a) Offline. Reward $W_1(O)$.

(b) Online. Reward $W_1(O)$.



(c) Offline. Reward $W_2(O)$.



(d) Online. Reward $W_2(O)$.

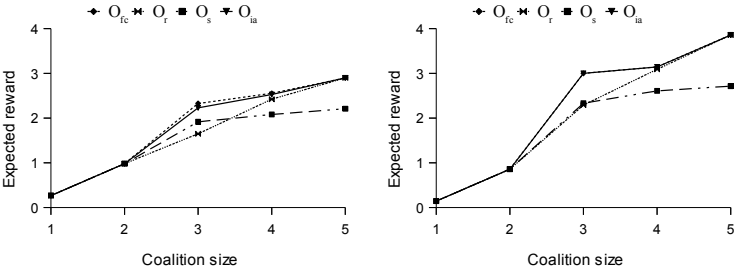
Fig. 4. Bar charts comparing offline and online versions of Alg. 1 analysed as a DTMC with the best-case performance of Alg. 2 analysed as an MDP

5.3 STPG Analysis

In this section we focus on the analysis of Alg. 2, but, in contrast to the previous section, we distinguish agents as either *cooperative* or *hostile*. This is naturally modelled as an STPG, by considering each class of agents (cooperative or hostile) as a separate player in a zero-sum game. Cooperative agents collaborate fully to act in the interest of the organisation, i.e., maximising rewards W_1 or W_2, whereas all hostile agents together take actions so as to minimise the expected rewards gained by the organisation. As a result of solving the STPG, we are able to obtain (synthesise) the optimal strategies for both cooperative and hostile agents, these can be used in further design of the system. Here we present the expected rewards achieved by the optimal strategies, but due to space limitations we omit the strategies themselves.

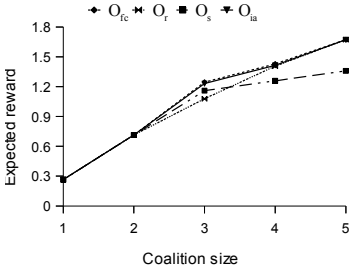
We are mainly interested in finding the optimal coalition. In general, a coalition $C \subseteq A$ is a subset of agents who cooperate to ensure certain goals, irrespective of how the other agents behave in the organization. We consider two criteria: the largest probability to accomplish tasks, and the largest reward (W_1 or W_2) achieved by the coalition. To this aim, we use the following PRISM temporal logic queries which, as mentioned earlier, have been extended for STPGs.

– $\mathtt{P}_{\max,\min=?}[\Diamond T_j\_done]$ - "what is the maximum probability for coalition C to complete task T_j when agents in $A \setminus C$ are hostile?" ($j \in \{1, 2\}$),

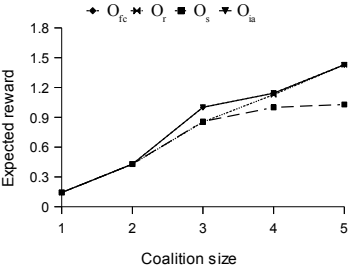


(a) Offline. Reward $W_1(O)$.

(b) Online. Reward $W_1(O)$.



(c) Offline. Reward $W_2(O)$.



(d) Online. Reward $W_2(O)$.

Fig. 5. Graphs comparing how the optimal coalition's performance depends on its size

- $\mathtt{P}_{\max,\min=?}[\,\Diamond\,(T_1\_done \wedge T_2\_done)\,]$ - "what is the maximum probability for coalition C to complete both tasks T_1 and T_2 when agents in $A \setminus C$ are hostile?",
- $\mathtt{R}^{W_j}_{\max,\min=?}[\,\Diamond\, finished\,]$ - "what is the maximum expected reward W_j for coalition C when agents in $A \setminus C$ are hostile?" ($j \in \{1, 2\}$).

For all agent organisations from Fig. 2, we enumerate all possible coalitions with different sizes and use PRISM to compute task completion probabilities (one task or both tasks) and rewards obtained (W_1 or W_2). These are done for both online and offline versions of the algorithm. It turns out that there exist coalitions of all sizes that are optimal with respect to all evaluated criteria; they are shown in Tab. 6. This result highlights the importance of positions in the network and resources held by the agents. For example, agent a_4 is in all optimal coalitions of sizes greater than 1 for O_{ia}. This is because it is connected to all agents, including agent a_5 which is isolated from other agents. For

Table 6. Optimal coalitions of all sizes for agent organisations from Fig. 2

O	1	2	3	4	5
O_{fc}	$\langle a_1 \rangle$	$\langle a_1, a_3 \rangle$	$\langle a_1, a_3, a_5 \rangle$	$\langle a_1, a_2, a_3, a_5 \rangle$	$\langle a_1, a_2, a_3, a_4, a_5 \rangle$
O_r	$\langle a_1 \rangle$	$\langle a_2, a_3 \rangle$	$\langle a_1, a_4, a_5 \rangle$	$\langle a_1, a_2, a_4, a_5 \rangle$	$\langle a_1, a_2, a_3, a_4, a_5 \rangle$
O_s	$\langle a_1 \rangle$	$\langle a_1, a_2 \rangle$	$\langle a_1, a_2, a_4 \rangle$	$\langle a_1, a_2, a_3, a_4 \rangle$	$\langle a_1, a_2, a_3, a_4, a_5 \rangle$
O_{ia}	$\langle a_1 \rangle$	$\langle a_1, a_4 \rangle$	$\langle a_1, a_2, a_4 \rangle$	$\langle a_1, a_2, a_4, a_5 \rangle$	$\langle a_1, a_2, a_3, a_4, a_5 \rangle$

O_r, however, the structure of the optimal coalition varies depending on coalition size. For example, for size 2 the optimal coalition consists of agents a_2 and a_3, but neither of them is in the optimal coalition of size 3.

Fig. 5 shows a comparison of agent organisations in terms of the maximum performance for different coalition sizes. O_{fc} outperforms others in all examples. This is interesting, because it suggests that having strong connectivity within the team outweighs the exposure to many hostile agents. Performance of O_r is the most consistent, as the maximum reward increases steadily with the coalition size. However, to be as effective as more connected networks like O_{fc}, the coalition has to contain most agents in the network. Better performance of O_s against O_r for coalition sizes up to 3 illustrates the importance of having a highly interconnected agent for small coalitions.

Important differences between the online and offline settings can be seen for reward W_1 in Fig. 5a and 5b. When going from coalition size 2 to 3, especially, for strongly connected O_{fc} and O_{ia}, coalitions can ensure that one task is completed with probability 1.0 , and thus guaranteeing reward of at least 3 for the coalition, which results in a jump of performance.

In this section, we have shown how an extension of PRISM to support STPGs can be used to verify properties of the multi-agent system that can be enforced by particular agent coalitions. This competitive scenario allowed us to analyse the team formation algorithm from the coalitional perspective, finding the optimal coalitions and comparing their performance on different network topologies.

6 Conclusion and Future Work

In this paper, we have presented a comprehensive, formal analysis of a team formation algorithm using probabilistic model checking. We believe this demonstrates the strong potential of these techniques to the formal analysis of multi-agent systems and hope that this case study will serve as a motivation for further work in this area.

As ongoing and future work, we plan to explore parallel execution of the JoinTeam algorithm for multiple agents, as here we only considered sequential execution, and consider the problem of synthesising optimal agent organisations for task distributions from a mechanism design perspective. We also plan to develop our prototype extension of PRISM into a fully-fledged model checker for stochastic games and to equip the analysis with abstraction techniques to allow for analysis of larger systems.

References

1. Abdallah, S., Lesser, V.R.: Organization-based cooperative coalition formation. In: IAT, pp. 162–168. IEEE, Los Alamitos (2004)
2. Ågotnes, T., van der Hoek, W., Wooldridge, M.: Reasoning about coalitional games. Artificial Intelligence 173(1), 45–79 (2009)
3. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM 49(5), 672–713 (2002)
4. Ballarini, P., Fisher, M., Wooldridge, M.: Uncertain agent verification through probabilistic model-checking. In: Barley, M., Mouratidis, H., Unruh, A., Spears, D., Scerri, P., Massacci, F. (eds.) SASEMAS 2004-2006. LNCS, vol. 4324, pp. 162–174. Springer, Heidelberg (2009)

5. Bonzon, E., Lagasquie-Schiex, M.-C., Lang, J.: Efficient coalitions in Boolean games. In: Texts in Logic and Games, vol. 5, pp. 293–297 (2008)
6. Chatterjee, K., Henzinger, T., Piterman, N.: Strategy logic. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 59–73. Springer, Heidelberg (2007)
7. Condon, A.: The complexity of stochastic games. Inf. Comput. 96(2), 203–224 (1992)
8. Dunne, P.E., van der Hoek, W., Kraus, S., Wooldridge, M.: Cooperative boolean games. In: Proc. of AAMAS 2008, pp. 1015–1022. ACM, New York (2008)
9. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated Verification Techniques for Probabilistic Systems. In: Proc. SFM 2011. Springer, Heidelberg (to appear, 2011)
10. Gaston, M.E., des Jardins, M.: Agent-organized networks for dynamic team formation. In: Proc. of AAMAS 2005, pp. 230–237. ACM, New York (2005)
11. Glinton, R., Scerri, P., Sycara, K.: Agent-based sensor coalition formation. In: Proc. Fusion 2008, pp. 1–7. IEEE, Los Alamitos (2008)
12. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
13. Kraus, S., Shehory, O., Taase, G.: Coalition formation with uncertain heterogeneous information. In: Proc. of AAMAS 2003, pp. 1–8. ACM, New York (2003)
14. Kwiatkowska, M., Norman, G.: Verifying randomized Byzantine agreement. In: Peled, D.A., Vardi, M.Y. (eds.) FORTE 2002. LNCS, vol. 2529, pp. 194–209. Springer, Heidelberg (2002)
15. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. In: Bernardo, M., Hillston, J. (eds.) SFM 2007. LNCS, vol. 4486, pp. 220–270. Springer, Heidelberg (2007)
16. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Proc. of CAV 2011. LNCS. Springer, Heidelberg (to appear, 2011)
17. Lomuscio, A., Raimondi, F.: MCMAS: A model checker for multi-agent systems. In: Hermanns, H. (ed.) TACAS 2006. LNCS, vol. 3920, pp. 450–454. Springer, Heidelberg (2006)
18. Manisterski, E., David, E., Kraus, S., Jennings, N.R.: Forming efficient agent groups for completing complex tasks. In: Proc. of AAMAS 2006, pp. 834–841. ACM, New York (2006)
19. Osborne, M.J., Rubinstein, A.: A course in game theory. The MIT press, Cambridge (1994)
20. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artificial Intelligence 101(1-2), 165–200 (1998)

Abduction-Based Search for Cooperative Answers

Samy Sá and João Alcântara

Universidade Federal do Ceará
samy@lia.ufc.br, jnando@lia.ufc.br

Abstract. We propose that agents may use abductive reasoning to adopt a Cooperative Answering behavior. In the event of failure or inadequacy of answers to a query (question), the agent should provide related answers by employing Query Relaxation. These answers are to be related to the query and might be useful to the querying agent. In order to achieve that, we consider agents with knowledge bases consisting of Abductive Logic Programs. Such agents are able to explain failure and guide the search for related answers. A distinctive aspect of our proposal is that in order to avoid undesirable results, one can select the parts of the query that cannot be relaxed.

1 Introduction

In a communication, agents can deliver or request information. When a request is made, it takes the form of a question (a query) and other agents will try to provide answers according to their knowledge bases. Cooperative Answering [5,7] is a form of cooperative behavior in which an effort is made so a deductive database can provide extra information (possibly useful) associated to a query. On its turn, deductive databases are a special kind of logic programs and queries have the same basic structure in both settings. Relaxation is presented in [5] as a method for expanding both deductive databases and logic programming queries. Just as well, logic programs are suitable to build intelligent agents and multi-agents systems, especially as an account for reasoning. Therefore, Cooperative Answering is particularly convenient when an agent whose knowledge base consists of a logic program fails to answer a query or the answer provided is not satisfactory to whoever made the question (its author), which may be either an user or another agent. In both cases, cooperative answering involves an effort to provide the best possible answer. This behavior can be of great use to Multiagent Systems (MAS) in most information sharing situations. For instance, consider the user agent *u01* asking for service in a library. Here is an example of question together with some possible answers:

u01 Do you have any books on MAS I have not yet borrowed?
(1) No. (or "None.")
(2) We only have these books on MAS, but you have borrowed them all.

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 208–224, 2011.

(3) There is a possibility. We have an unclassified publication on MAS never borrowed by you. It might be a book...

The first answer is the traditional one in databases, since there would be only an empty set of answers in case of failure. Clearly, the second and third options might be of more help to the asking agent. The effort to provide better answers characterizes this form of cooperative behavior.

Abduction is a form of non-monotonic reasoning capable of building explanations for newly observed, previously unknown data. In this paper, we attack the problem of maximizing the usefulness of related answers retrieved by relaxation of a query by employing abductive reasoning to search for answers closer to those of the original query. We will introduce criteria based on abductive reasoning such as those found in abductive logic programming (ALP) [4,10,14]. ALP has been combined with deductive databases in a variety of ways, such as for database integrity recovery [13], query optimization [16] and for assuring database integrity over view updates [14,1], just to name a few. In our approach, abduction is used to produce explanations for failure or give clues about the best way to expand a query in an attempt to retain its meaning. In that sense, an explanation is used to pinpoint which conditions of the query should be worked on to guide the relaxation process. We say such conditions are useful towards relaxation. Other contributions involve relating abductive explanations to Maximally Succeeding Subqueries [7] and providing preference criteria to rank explanations that relates to the Best-Small Plausibility Criterion [3]. We also consider the author of a query (either an user or another agent) can name the most important conditions and help the process. For most of the paper, we will only consider failed queries. In fact, we will argue that the expansion of the scope of a query can be considered as a special case of failure. In either case, we defend abduction to be the key to retrieve good related answers.

The paper is organized as follows: Section 2 introduces abductive logic programs, queries and relaxations. Section 3 presents the concepts we will use to guide the search process, which is discussed in Section 4. Section 5 considers the case in which the original query does not fail. We will evaluate our contributions in Section 6. Related work is discussed in Section 7 and Section 8 concludes the paper.

2 Background

In this paper we combine abductive reasoning and query relaxation to assist finding better related answers to queries. In this section we introduce Extended Disjunctive Programs (EDPs) [6], Abductive Logic Programs (ALPs) [14,15]; and define explanations, queries and different relaxation methods. We also present an example that will be used all along the paper.

2.1 Extended Disjunctive Programs

In this paper, we account for programs as intended in Extended Disjunctive Programs (EDP) [6] or Databases.

An EDP is defined over a *Herbrand Universe* HB, the set of all ground terms (terms without variables) a program might refer to. Such a program consists of a set of rules of the form

$$r: \ L_1; \ldots; L_k \leftarrow L_{k+1}, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$$

with $n \geq m \geq k \geq 0$. In this notation, ";" is the disjunction operator, each L_i is an objective literal, meaning it is either an atom (A) or its negation($\neg A$) and *not* is *negation as failure* (NAF). If L is an objective literal, *not* L is called a NAF-*literal*. We refer to both NAF-literals and objective literals as literals. In a rule r as above, we the disjunction $L_1; \ldots; L_k$ is the *head* of the rule and we use $head(r)$ to denote the set of objective literals $\{L_1, \ldots, L_k\}$. Similarly, the conjunction $L_{k+1}, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$ is the body of the rule, and $body(r)$ denotes the set of literals $\{L_{k+1}, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n\}$. We differ the objective literals from the positive and negative parts as $body^+(r)$ and $body^-(r)$ to refer to the sets $\{L_{k+1}, \ldots, L_m\}$ and $\{L_{m+1}, \ldots, L_n\}$, respectively. We also denote $not\_body^-(r)$ as the set of NAF-*Literals* $\{not\ L_{m+1}, \ldots, not\ L_n\}$. A rule may be written as $head(r) \leftarrow body^+(r), not\_body^-(r)$ or simply $head(r) \leftarrow body(r)$, since $body(r) = body^+(r) \cup not\_body^-(r)$. A rule is disjunctive if $head(r)$ has more than one literal and range restricted if all variables in $not\_body^-(r)$ also appear in $body^+(r)$. We also say a rule is an integrity constraint if $head(r) = \emptyset$ and that it is a fact if $body(r) = \emptyset$. Rules and literals can be differentiated through renaming their variables. A substitution $\theta = \{x_1/t_1, \ldots x_n/t_n\}$ is a mapping from variables to terms where $x_1, \ldots, x_n$ are all distinct variables and all t_i is a distinct term from x_i. If G is a conjunction of literals, $G\theta$ is the conjunction obtained from G through application of the substitution θ. Similarly, if r is a rule, $r\theta$ is an instance of r through the application of θ. We say that a program, rule or literal without variables is *ground*. A program with variables has a *ground instantiation* $ground(P)$ which consists of gathering all the substitutions of variables in P for elements of HB. We consider a program P with variables is identified by its ground instantiation $ground(P)$.

The semantics of an EDP is given by the Answer Sets Semantics [6]. Consider Lit_P as the set of all ground objective literals in the language of a program P and S one of its subsets. So P^S is the reduct of $ground(P)$ containing only all the ground instances $head(r) \leftarrow body^+(r)$ of rules of P such that $body^-(r) \cap S = \emptyset$. Given a NAF-*free* EDP P, S will be an Answer Set of P if it is a minimal subset of Lit_P such that (i) for every ground rule of P, if $body^+(r) \subseteq S$, then $head(r) \cap S \neq \emptyset$ and (ii) S is either consistent or $S = Lit_P$. An answer set S is consistent if, for every objective literal L, $\{L, \neg L\} \not\subseteq S$. A program might have zero, one or multiple answer sets. The program itself will be said consistent (inconsistent) if at least one (none) of its answer sets is consistent.

2.2 Abductive Logic Programs

Abduction is a special kind of non-deductive reasoning in which hypothesis are inferred to explain observable facts otherwise inconsistent to a theory. Abductive Logic Programming (ALP) brings this feature to standard logic programming

[10,4]. We will now introduce ALPs as in the abductive framework of *Extended Abduction* [14,15].

An abductive program is a pair $\langle P, H\rangle$, where P is an Extended Disjunctive Program (EDP) [6] with semantics given by the Answer Set Semantics [6] and H is a set of objective literals referred to as abducibles. If a literal $L \in H$ has variables, then all ground instances of L are abducibles and, consequently, elements of H. If P is consistent, then $\langle P, H\rangle$ is consistent. Throughout the paper we will assume only consistent programs. We say a conjunction $G = L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$ is range restricted if every variable in $L_{m+1}, \ldots, L_n$ also appears in $L_1, \ldots, L_m$. An *observation* over $\langle P, H\rangle$ is a conjunction G with all variables existentially quantified and range restricted. An observation $L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$ is satisfied by P if $\{L_1\theta, \ldots, L_m\theta\} \subseteq S$ and $\{L_{m+1}\theta, \ldots, L_n\theta\} \cap S = \emptyset$ for some substitution θ and answer set S of P. An observation is satisfied by $\langle P, H\rangle$ if it is satisfied by P.

Definition 1. *Let G be an observation over the ALP $\langle P, H\rangle$. A pair (E, F) is an* explanation *of G in $\langle P, H\rangle$ if:*

1. *$(P \setminus F) \cup E$ has an answer set which satisfies G*[1]*,*
2. *$(P \setminus F) \cup E$ is consistent,*
3. *E and F are sets of ground objective literals such that $E \subseteq H \setminus P$ and $F \subseteq H \cap P$.*

Intuitively, an explanation (E, F) means that by taking the literals in E as true while retracting (falsifying) the literals in F from P, the resulting P' satisfies G. If $(P \setminus F) \cup E$ has an answer set S satisfying all three conditions, S is called a *Belief Set* of $\langle P, H\rangle$ satisfying G with respect to (E, F). When an observation has an explanation, it means it can be made consistent with P. If the original program has an answer set satisfying G, then $(\emptyset, \emptyset)$ is an explanation and no changes are needed in P. An explanation (E, F) is minimal if, for any explanation (E', F') such that $E' \subseteq E$ and $F' \subseteq F$, then $E' = E$ and $F' = F$. In general, only the minimal explanations are of interest. The set H is related to the explanatory power of an agent. The more the elements in H, the more explanations the agent might be able to conceive.

2.3 Queries to an ALP

Informally, a query is a question to a knowledge base about its data and represents the intention of retrieving specific sets of facts from it. In the case of disjunctive programs, a query is a conjunction of conditions that describe the expected data. In fact, a query to an ALP consists of an observation over it, so it succeeds and fails in the same conditions (Section 2.2).

[1] This definition is for *credulous* explanations. Its choice over *skeptical* explanations [9] makes possible to have more explanations and gives us a better chance of finding good related answers to a query.

Definition 2. *A query is a conjunction $G = L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$ of literals (conditions) with all variables existentially quantified and range restricted. We write $Lit(G)$ to refer to the set of literals in a query G in contrast to the conjunction of such literals.*

Each literal represents a condition. We commonly refer to a query by writing either its set of literals $Lit(G)$ or the conjunction of those.

2.4 Query Relaxation

Cooperative query answering is a paradigm of query processing that tries to help the user by adding results closely related to the query or correcting misconceptions about a database [5]. By relaxing a query, we substitute it by another one that has a broader scope than the previous, so its result is a superset of the first. Relaxation is mainly employed when a query fails and allows to retrieve related results instead of failure. We consider the relaxation methods introduced in [15] as they are oriented to be used with ALPs.

Definition 3. *A query G can be relaxed to G' by any of the following methods:*

1. *Anti-Instantiation (AI): given a substitution θ if $G'\theta = G$, then G' is a relaxation of G by anti-instantiation.*
2. *Dropping Conditions (DC): if G' is a query and $Lit(G') \subset Lit(G)$ then G' is a relaxation of G where the conditions of $Lit(G) \setminus Lit(G')$ were dropped.*
3. *Goal Replacement (GR): if G is a conjunction G_1, G_2 and there is a rule $L \leftarrow G'_1$ in P with $G'_1\theta = G_1$, then $G' = L\theta, G_2$ is a relaxation of G by goal replacement.*

In each case, G' is also a query as previously defined.

Definition 4. *We say a literal (or condition) L of G was replaced in a relaxation G' if $L \in Lit(G) \setminus Lit(G')$ (in case of Dropping Condition or Goal Replacement) or L has some arguments with ground terms in G and variables in G' (in case of Anti-Instantiation).*

One can imagine the various ways of iteratively relaxing a query G in the manner of a Directed Graph with a single source in the original query and sink in the empty query ($G' = \emptyset$). In that case, each possible relaxation of G is represented as a node and every edge leads a node to a relaxation of its query obtained by a single application of just one of the methods presented above. Since a path in the graph is a sequence of relaxations, it will be a Directed Acyclic Graph (DAG) if no rule in the program has $head(r) \subseteq body(r)$.

2.5 Running Example

We will introduce an ALP, a query and some possible relaxations. This example will be referred back at the end of each section in order to illustrate any new concepts.

In what follows, we address the terms user (usr), book (bk), MAS, ALP, article ($artc$), borrowed ($brwd$) and publication (pub) and Artificial Intelligence (ai). We employ the character "." to separate rules in a program.

Example 1. Consider the ALP $\langle P, H\rangle$:

$$
\begin{array}{l}
P: usr(u01). \\
\quad bk(b11). \quad mas(b11). \quad brwd(u01,b11). \\
\quad bk(b12). \quad alp(b12). \\
\quad pub(b13). \quad alp(b13). \quad brwd(u01,b13). \\
\quad artc(b14). \quad mas(b14). \\
\quad artc(b15). \quad alp(b15). \\
\quad pub(b16). \quad mas(b16). \\
\\
\quad pub(X) \leftarrow bk(X). \qquad pub(X) \leftarrow artc(X). \\
\quad ai(X) \leftarrow mas(X). \qquad ai(X) \leftarrow alp(X). \\
\quad \leftarrow usr(X),\ bk(X). \\
\quad \leftarrow bk(X),\ artc(X). \\
\\
H: \{bk(X), mas(X), alp(X), brwd(X,Y), artc(X)\}
\end{array}
$$

The program P has a single answer set S that includes all the facts in the program together with the conclusions $pub(b11)$, $pub(12)$, $pub(14)$, $pub(15)$, $ai(11)$, $ai(12)$, $ai(13)$, $ai(14)$, $ai(15)$, $ai(16)$.

The query $G = bk(X), mas(X), not\ brwd(u01, X)$ from Section 1 fails in P. It has the following explanations:

$$
\begin{array}{l}
(E_1, F_1) = (\{\ \}, \{brwd(u01, b11)\}) \\
(E_2, F_2) = (\{mas(b12)\}, \{\ \}) \\
(E_3, F_3) = (\{bk(b13),\ mas(b13)\}, \{brwd(u01, b13)\}) \\
(E_4, F_4) = (\{bk(b14)\}, \{artc(b14)\}) \\
(E_5, F_5) = (\{bk(b15),\ mas(b15)\}, \{artc(b15)\}) \\
(E_6, F_6) = (\{bk(b16)\}, \{\ \})
\end{array}
$$

The possible relaxations of G are quite numerous. In the sequel, we present just a few examples to illustrate Definition 3:

$$
\begin{array}{ll}
G'_1 = bk(X), mas(X), not\ brwd(Y, X) & (AI) \\
G'_2 = mas(X), not\ brwd(u01, X) & (DC) \\
G'_3 = pub(X), mas(X), not\ brwd(u01, X) & (GR)
\end{array}
$$

The literal $not\ brwd(u01, X)$ was replaced in G'_1. Similarly, $bk(X)$ was replaced in both G'_2 and G'_3.

We will address to other relaxations of G throughout the paper.

3 Important Concepts

This section explores the concepts of useful literals and rational relaxations. We propose guiding the relaxation search by discarding options not based on these

concepts. Their formalization should be enough to convince of their usefulness to the problem and motivate our work. We first present a way for the author of the query to restrict relaxation attempts.

3.1 Useful Literals

Whenever a query G fails in a program $\langle P, H\rangle$ for a substitution θ, some literals in $Lit(G\theta)$ may be satisfied by $\langle P, H\rangle$. If any of these literals are replaced in a relaxation G', then $G'\theta$ will also fail. In a search for a successful relaxation, such literals are useless. In order to identify the useful ones, we introduce the following definition:

Definition 5. *Let G be a failed query and (E, F) an explanation for it in the program $\langle P, H\rangle$. Let S be an answer set of P. The set $U^S_{E,F}(G) = \{L \in Lit(G) \mid G\theta$ is satisfied by $P' = (P \setminus F) \cup E$ and either L is not a NAF-literal and $L\theta \notin S$ or L is a NAF-literal and $L\theta \in S$, for some $\theta\}$. We say $U^S_{E,F}(G)$ is the set of useful literals of G towards relaxation according to the explanation (E, F) and answer set S.*

Definition 6. *Let G be a failed query and (E, F) an explanation for it in the program $\langle P, H\rangle$. Let $S_1, \ldots, S_n$ be the answer sets of P. The set $U_{E,F}(G) = \{L \in Lit(G) \mid L \in U^{S_i}_{E,F}(G), 1 \leq i \leq n\}$ is the set of useful literals of G towards relaxation (or just useful, for short) according to the explanation (E, F). A literal is useful according to H if it is useful according to a minimal explanation in $\langle P, H\rangle$. The set of such literals is denoted by $U_H(G)$.*

Example 2. Next, we show which literals are useful in $U_{E,F}(G)$ according to each explanation (E, F) for the query $G = bk(X), mas(X), not\ brwd(u01, X)$ in the program from Example 1, that has a single answer set S:

$$
\begin{aligned}
&U_{E_1,F_1}(G) = U^S_{E_1,F_1}(G) = \{not\ brwd(u01, X)\}\\
&U_{E_2,F_2}(G) = U^S_{E_2,F_2}(G) = \{mas(X)\}\\
&U_{E_3,F_3}(G) = U^S_{E_3,F_3}(G) = \{bk(X),\ mas(X),\ not\ brwd(u01, X)\}\\
&U_{E_4,F_4}(G) = U^S_{E_4,F_4}(G) = \{bk(X)\}\\
&U_{E_5,F_5}(G) = U^S_{E_5,F_5}(G) = \{bk(X),\ mas(X)\}\\
&U_{E_6,F_6}(G) = U^S_{E_6,F_6}(G) = \{bk(X)\}
\end{aligned}
$$

Relying on useful literals is the first step to guide relaxation. Any attempts to relax a query by only replacing non-useful literals should fail. Next, we will restrict explanations to avoid wrongly taking a literal as useful.

3.2 Querying Agent's Choice

The author of a query might consider some of its conditions as crucial to the answer. We propose an extended notion of query in which the author might impose restrictions to relaxation. By extending a query with a set of non-replaceable literals, the querying agent can help the process with a better understanding of what is expected as answer for the query. This should be perceived as a way for the querying agent to indicate which neighborhood answers are acceptable.

Definition 7. *A restricted query is a pair* (G, B) *such that* $B \subseteq Lit(G)$ *and* G *is a query (as before).*

The literals in B are intended to identify those that cannot be replaced in a relaxation attempt nor used to build explanations. These restrictions bring change to the definitions of useful literals and explanations above, however these definitions can be easily revised: For useful literals, it suffices to replace $U_{E,F}(G) \subseteq Lit(G)$ with $U_{E,F}(G) \subseteq Lit(G) \setminus B$ in Definition 6. Explanations for restricted queries are the same as in Section 2.2, except that E and F are now such that $E \subseteq (H \setminus B) \setminus P$ and $F \subseteq (H \setminus B) \cap P$. We believe that making the proper corrections instead of presenting this version from start is better for clarity and presentation.

Example 3. Consider the restricted query (G, B) with G being the same as in Example 1 and $B = \{not\ brwd(u01, X)\}$. This restriction means that any related answers suggested should definitely not have been borrowed by the author of the query before. In that case, the explanations (E_1, F_1) and (E_3, F_3) would not be accepted, leaving only four explanations to be taken into account.

We will commonly address general queries instead of restricted ones. Whenever we do so, we will omit B. Any other changes promoted by a non-empty B should be easy to understand. Otherwise, we will be sure to comment on it.

3.3 Rational Explanations

A substitution θ' such that no literals in $Lit(G\theta')$ are satisfied by and answer set S of P suggests all literals as useful according to that answer set. Such explanations can be misleading as they consider ground instances of the query that fail in every condition. Any relaxation based on such explanations will likely produce answers far from those expected or also lead to failure. In fact, in case all possible relaxations of a query also fail, it is possible to still have explanations, but only of the kind that suggests all literals as useful. We will call such explanations to be non-rational and distinguish them from rational ones.

Definition 8. *An explanation* (E, F) *is called rational iff* $Lit(G) \setminus U^S_{E,F}(G) \neq \emptyset$*, for at least one answer set* S *of* P*. Otherwise, it is said to be a non-rational explanation.*

Example 4. The explanation (E_3, F_3) can be concluded to be non-rational from the beginning, since $U^S_{E_3,F_3}(G) = Lit(G)$. Then, this explanation should be discarded. All other explanations are taken as rational. If we consider the restricted query (G, B) from Example 3, these explanations are not produced. As a consequence, all explanations for (G, B) are rational.

Rational explanations state that the query is somehow partially satisfied, as opposed to non-rational explanations. In case all explanations are non-rational, it means that there are no good related answers or that the agent did not find rational explanations. That last case should be interpreted as if the agent does

not consider rational to relax the query. For the remainder of the paper, whenever we address explanations, we will be considering rational explanations.

Next, we present some results that encourage the choice of rational explanations.

Theorem 1. *If there is at least one rational explanation for the failure of G in $\langle P, H\rangle$, then every relaxation of G that fails will also have at least one explanation.*

Proof. The existence of an explanation (E, F) means G can be made consistent with P. If G' is a relaxation of G, then (E, F) is also an explanation for G' (possibly not minimal).

The existence of an explanation for the failure of a query G means no integrity constraints are violated by its potential answers, otherwise it would not be possible to make the query consistent with P by changing the set of facts in the program. Theorem 1 states that every relaxation of G also has at least one explanation, so we can assure no integrity constraints will be violated by such relaxed queries.

Theorem 2. *Every rational explanation can be effectively used to guide the search for a relaxed query that succeeds (is satisfied by $\langle P, H\rangle$).*

Proof. By Definition 8, $|Lit(G)| - |U_{E,F}(G)| \geq 1$. It suffices to drop the conditions consisting of useful literals. By doing so, the set of conditions left will consist of a query that succeeds.

Non-rational explanations might relax the query to the point it gets empty before succeeding.

Definition 9. *A relaxation G' of G is rational if (E, F) is rational and only literals $L \in U_{E,F}(G)$ are being replaced.*

4 Our Search Method

The search for answers related to a query is usually treated as search in a directed graph. As such, it can benefit of many search methods from AI. Our approach suggests significant cuts to the tree, guiding the search to make it local. Each explanation restricts the possible relaxations of the query by pinpointing which literals should be replaced. Another use of explanations is to describe the related results. In that sense, the less replacements an explanation suggests and the less changes to the program an explanation requires, the better. Therefore, we build a ranking of rational explanations and use it to guide the search towards the most promising paths in the graph. In this section we present our ranking criteria, define our approach and discuss how it deals with common issues related to search in graphs.

4.1 Ranking the Explanations

In cooperative answering, it would be helpful to identify adequate criteria to decide which explanations in a set should lead to the best related answers. An agent capable of pinpointing the best explanations will surely be able to provide answers as helpful as possible to any queries against its knowledge base. We now present the reduced numbers of useful literals and changes suggested to a program by an explanation as good leads to rank them.

Definition 10. *Let T be the set of all explanations for the failure of a query G. $T_{\preccurlyeq}$ is a reflexive and transitive preference relation over T, such that $(E, F) \preccurlyeq (E', F')$ iff $|U_{E,F}(G)| \leq |U_{E',F'}(G)|$.*

Definition 11. *An explanation (E, F) has a preference priority grade in $T_{\preccurlyeq}$, denoted by $N_{T_{\preccurlyeq}}(E, F)$, such that $N_{T_{\preccurlyeq}}(E, F) = |\{(E', F') \mid (E', F') \preccurlyeq (E, F)\}|$.*

We use the preference priority grades to decide in which order the explanations are taken to guide relaxation. As such, an explanation with lower priority grade is said to be of higher priority in the sense of being used to improve search.

Example 5. We refer back to the explanations taken as rational according to the previously presented Example 3. In that case, we have

$$T_{\preccurlyeq} = (E_2, F_2) \preccurlyeq (E_4, F_4) \preccurlyeq (E_6, F_6) \preccurlyeq (E_2, F_2) \text{ and } (E_2, F_2) \preccurlyeq (E_5, F_5).$$

The explanations $(E_2, F_2), (E_4, F_4), (E_6, F_6)$ can appear in any sequence, since $N_{T_{\preccurlyeq}}(E_2, F_2) = N_{T_{\preccurlyeq}}(E_4, F_4) = N_{T_{\preccurlyeq}}(E_6, F_6) = 2$. However, (E_5, F_5) has a lower preference priority when compared to the others ($N_{T_{\preccurlyeq}}(E_5, F_5) = 3$).

The explanations of lowest priority grade in $T_{\preccurlyeq}$ are related to the ground instances of G that are the closest of being satisfied by P and, consequently, to the Maximal Succeeding Subqueries (MSS) of G [7].

The changes required to P by an explanation can also be taken as clue to how far a query is from being satisfied according to it.

Definition 12. *Let T be the set of all explanations for the failure of a query G. $T_{\preceq}$ is a reflexive and transitive preference relation over T, such that $(E, F) \preceq (E', F')$ iff (i) $|U_{E,F}(G)| = |U_{E',F'}(G)|$ and $|E| + |F| \leq |E'| + |F'|$ or (ii) $|U_{E,F}(G)| \neq |U_{E',F'}(G)|$ and $(E, F) \preccurlyeq (E', F')$.*

Please note that the symbols $\preccurlyeq$ and $\preceq$ used in the preference relations above defined look alike, but are different. We chose such a pair of symbols because the concepts behind each preference relation are themselves similar.

Definition 13. *An explanation (E, F) has a preference priority grade in $T_{\preceq}$, denoted by $N_{T_{\preceq}}(E, F)$, such that $N_{T_{\preceq}}(E, F) = |\{(E', F') \mid (E', F') \preceq (E, F)\}|$.*

Example 6. Once again, we refer to the rational explanations according to Example 3. We then have

$$T_{\preceq} = (E_2, F_2) \preceq (E_6, F_6) \preceq (E_2, F_2) \text{ and } (E_2, F_2) \preceq (E_4, F_4) \preceq (E_5, F_5).$$

In $T_{\preceq}$ only the explanations (E_2, F_2) and (E_6, F_6) have minimal priority grade ($N_{T_{\preceq}}(E_2, F_2) = N_{T_{\preceq}}(E_6, F_6) = 1$). Also, the explanation (E_4, F_4) is only preferred over (E_5, F_5), since $N_{T_{\preceq}}(E_4, F_4) = 2$ and $N_{T_{\preceq}}(E_5, F_5) = 3$.

The explanations of minimal priority grade in $T_{\preceq}$ are those related to MSSs that require the lesser adaptation of P. The ranking criteria of $T_{\preceq}$ resembles the Best-Small Plausibility Criterion [3]. This criterion suggests that more plausible explanations are preferred over less plausible ones. In case two explanations present the same plausibility, the smaller explanation should be preferred.

Definition 14. *The* TOP-Explanations *for a query G are the explanations with minimal priority grade in the preference relation $T_{\preceq}$.*

The intuition behind TOP-Explanations is that they are the best-ranked explanations in our priority relations. Therefore, these should be the best candidates to be considered when searching for related answer of a query.

Example 7. (E_6, F_6) and (E_2, F_2) are TOP-Explanations in 1.

The explanations with minimal priority grade in $T_{\preccurlyeq}$ indicate, to some extent, which useful literals should be replaced first in relaxation attempts. Thus the use of TOP-Explanations can further determine precedence over useful literals.

4.2 Equivalent Explanations

Some explanations will lead to the exact same attempts to relax a query. Such a pair of explanations will suggest the exact same literals as useful.

Definition 15. *We say two explanations (E, F) and (E', F') are equivalent iff $U_{E,F}(G) = U_{E',F'}(G)$.*

This definition produces different equivalence classes. For each class, some of its elements can be discarded in order to further optimize search. The selected representatives should be those with minimal priority grade in the preference relation $T_{\preceq}$ when restricted to that class of equivalence.

Example 8. The explanations (E_4, F_4) and (E_6, F_6) are equivalent, so we can discard one of them safely. In this case, we opt to discard (E_4, F_4), resulting in the relation

$$(E_6, F_6) \preceq (E_2, F_2) \preceq (E_5, F_5).$$

4.3 The Search for Cooperative Answers

Our search method is based on the preference relation $T_{\preceq}$ presented in Section 4.1. This relation is conceived to minimize the changes to the query during the

relaxation process. This is done as an attempt to retain as much as possible of the query semantics.

In case a query (G, B) fails in P, the agent should search for explanations, discard the non-rational ones and build the preference relation $T_{\preceq}$. When this relation is not empty, then some rational explanations can be conceived and the query can be successfully relaxed. The agent should reduce equivalent explanations and then follow the preference relation, considering one explanation at a time and starting by the TOP-Explanations. For each explanation, the agent restricts the replacement of literals to those suggested as useful.

In order to assure completeness and efficiency, a search strategy implementation is bound to try to apply the relaxation methods in an specific order over each query. Taking this into consideration, we suppose any attempts to relax a query will take preference for Goal Replacement over Dropping Conditions and for Anti-Instantiation over both the other kinds. These preferences cope with the intuition that the set of predicates used in a query characterize it more than its ground arguments and that good related answers should be as close as possible to satisfying all such predicates.

Example 9. Most of these steps have been shown in the previous examples. The agent follows the preference relation built in Example 6 and relaxes the query G based on information from the TOP-Explanation (E_6, F_6). Relaxations should be made first by attempting anti-instantiation, then goal replacements and dropping conditions. The first relaxation is produced by goal replacement with the rule $pub(X) \leftarrow bk(X)$. The query $G' = pub(X), mas(X), not\ brwd(u01, X)$ succeeds with two results $X = b14$ and $X = b16$ and the related answer is returned.

A search using other explanations will only be necessary if all answers provided by the first explanation are not satisfactory to the query author. In that case, the agent takes the next explanation in the preference relation and so on.

4.4 Relaxation Trees and Issues

A relaxation tree relates a query (root) to its possible relaxations. The sons of a query node are its relaxations obtained by applying any of the methods from Definition 3 once. The leaves are nodes that can only be relaxed by dropping conditions and leading to an empty query. Given that tree, any search methods can be used. In any case, breadth-first methods are preferable over depth-first methods [5]. Some issues of the search in trees include multiple paths to the same answer and the size of neighborhood [5]. Those issues, especially when combined, result in a big search space and redundancy. Our approach reduces the search space by restricting possible relaxation attempts. As a result, the neighborhood of nodes and the number of alternative paths to relaxed queries are decreased.

The restrictions of explanatory power of an agent and from the author of the query in restricted queries make for alternative cuts in the search space. Differently, we use the best rational explanations (Definition 14) to possibly reduce changes to the original query. A set of useful literals will determine which relaxation methods can be applied. Each set refers to a part of the search space and

restricts search to a specific part of it. By rationally restricting the literals that can be replaced, we reduce the neighborhood of the nodes. This feature can be also perceived as pruning the tree. In that sense, our method works by reducing the neighborhood of nodes to be visited and, consequently, redundancies in the search. Redundancies are also reduced by eliminating equivalent explanations (Section 4.2).

5 Expanding the Scope of a Query

In case an answer (either relaxed or not) is not satisfactory to the querying agent, the cooperative behavior is to expand the scope of the query by relaxing it and looking for related answers. When a query fails in the first place, we have a preference relation over the explanations and the agent can follow it to guide relaxation. We now address the case when the query first succeeds. Explanations to guide relaxation can be found and ordered by applying the following steps:

1. Given $a_1, ..., a_n$ are the arguments of G, extend the language of $\langle P, H\rangle$ with a new relation $R\backslash n$.
2. Take the literal $R(a_1, ..., a_n)$ as an abducible and compose the query G_0 as $G \wedge R(a_1, ..., a_n)$.
3. Discard any explanations (E, F) that consists of nothing but an instance of $R\backslash n$ in E together with any non-rational ones (Definition 8).
4. Compute the preference relation $T_{\preccurlyeq}$ for G_0 (Definition 10).
5. Compute $T_{\preceq}$ (Definition 12).
6. Discard unnecessary explanations out of equivalence as in Section 4.2.
7. Use the preference relation $T_{\preceq}$ to guide the relaxations of G.

The first and second steps produce a query that will be guaranteed to fail in $\langle P, H\rangle$. As a consequence, for each substitution $\theta = \{X_1/a_1, ..., X_n/a_n\}$ for which G succeeded, there will be a minimal explanation $(\{R(a_1, ..., a_n)\}, \{\})$ in $T_{\preccurlyeq}$. After discarding these in the fourth step, the explanations with minimal priority grade are related to ground instances of G that failed in P minimally. The preference relation $T_{\preceq}$ is then built and optimized in the sixth step. As a consequence, all literals in U_H will be useful to relax the original query, except by $R(q)$.

Example 10. Consider, for instance, the query $G = bk(X), mas(X)$ which succeeds in $\langle P, H\rangle$ from example 1 with result $X = b11$. Suppose, then, the agent $u01$ wants to know about other books on MAS and replies that such answer is not satisfactory. The answering agent creates the query $G_0 = bk(X), mas(X), R(X)$, which fails, and takes $R(X)$ as an abducible. The query G_0 has the following explanations in $\langle P, H\rangle$:

$$
\begin{aligned}
(E_1, F_1) &= (\{R(b11)\}, \{\ \}) \\
(E_2, F_2) &= (\{mas(b12),\ R(b12)\}, \{\ \}) \\
(E_3, F_3) &= (\{bk(b13),\ mas(b13),\ R(b13)\}, \{\ \}) \\
(E_4, F_4) &= (\{bk(b14),\ R(b14)\}, \{artc(b14)\}) \\
(E_5, F_5) &= (\{bk(b15),\ mas(b15),\ R(b15)\}, \{artc(b14)\}) \\
(E_6, F_6) &= (\{bk(b16),\ R(b16)\}, \{\ \})
\end{aligned}
$$

The useful literals $U_{E,F}(G_0)$ according to each explanation (E, F) for the query $G_0 = bk(X), mas(X), R(X)$ are:

$$\begin{aligned}
U_{E_1,F_1}(G_0) &= U^S_{E_1,F_1}(G_0) = \{R(X)\} \\
U_{E_2,F_2}(G_0) &= U^S_{E_2,F_2}(G_0) = \{mas(X),\ R(X)\} \\
U_{E_3,F_3}(G_0) &= U^S_{E_3,F_3}(G_0) = \{bk(X),\ mas(X),\ R(X)\} \\
U_{E_4,F_4}(G_0) &= U^S_{E_4,F_4}(G_0) = \{bk(X),\ R(X)\} \\
U_{E_5,F_5}(G_0) &= U^S_{E_5,F_5}(G_0) = \{bk(X),\ mas(X),\ R(X)\} \\
U_{E_6,F_6}(G_0) &= U^S_{E_6,F_6}(G_0) = \{bk(X),\ R(X)\}
\end{aligned}$$

The agent discards the non-rational explanations (E_3, F_3) and (E_5, F_5), together with (E_1, F_1), since it consists of only the literal $R(X)$. Next, the agent computes the preference relation $T_{\preccurlyeq}$ (Definition 10) to find

$$T_{\preccurlyeq} = (E_2, F_2) \preccurlyeq (E_4, F_4) \preccurlyeq (E_6, F_6) \preccurlyeq (E_2, F_2).$$

The preference relation $T_{\preceq}$ (Definition 12) has the result

$$T_{\preceq} = (E_2, F_2) \preceq (E_6, F_6) \preceq (E_2, F_2) \text{ and } (E_2, F_2) \preceq (E_4, F_4).$$

Next, the agent discards explanation (E_4, F_4), since it is equivalent to (E_6, F_6) (Definition 15), and $(E_6, F_6) \preceq (E_4, F_4)$. Then, if the agent uses the explanation (E_2, F_2) to guide relaxation, it will consider $mas(X)$ and $R(X)$ as useful. Since the process is made over the original query G, only $mas(X)$ is actually useful. No relaxation by anti-instantiation is possible, so the agent attempts goal replacement. The agent finds a relaxed query $G' = bk(X), ai(X)$, which succeeds as more general than G and returns results $X = b11$ and $X = b12$, a superset of the results of G. In case the agent chooses to use the explanation (E_6, F_6), it will succeed with the relaxed query $G' = pub(X), mas(X)$ and results $X = b11$, $X = b14$ and $X = b16$.

6 Evaluation

First, we explore the trade-off perceived in searching for explanations before attempting relaxations. An extensive study about the complexity of abduction can be found in [3]. Their work deals with explanations that correct missing information, differently from those in [15], which are more expressive. The explanations of the latter relate the most to the class of Incompatibility Abduction Problems from the first. For such problems, finding a best explanation is NP-Hard [3]. For more specific results on the complexity of Extended Abduction, please refer to [14] (Theorem 6.3, item 1). For instance, the complexity of deciding if an observation as in Section 2.2 has a credulous explanation is $\sum_2^P - complete$.

We point out that explanations tend to be far less numerous than possible relaxations of a query. In fact, the number of explanations and possible relaxations depend on the knowledge base and the query itself, but we can control these numbers in a few ways. For instance, a restriction on the set of abducibles to only the literals of the query can reduce the number of explanations produced.

Also, one can reduce the number of possible relaxations of a query by naming conditions that should not be replaced as in Section 3.2. Another way to interfere with the number of possible relaxations is by orienting the design of the knowledge base to exhibit a cooperative answering behavior. In that case, possible relaxations should be previously conceived and made possible by the addition of rules or blocked by working out different integrity constraints. Either way, the use of explanations can drastically reduce redundancy in search.

As a final note, our approach is capable of significantly improving the quality of answers related to a query. Such an improvement in quality is achieved by finding minimal explanations to guide relaxation, which in turn are related to ground instances of data that satisfy the most conditions of the original query. Our approach reduce the search space because it considers relaxations that replace literals useful to one rational explanation at a time. Even though there is some reduction in the time spent exploring the tree, to build the preference relations from Section 4.1 changes the complexity of the process since we first need to find all the explanations for the failure of the query. However, we draw attention of the reader to the fact that instead of searching for related answers, we search for the *best* related answers.

7 Related Work

The approach of using sets of abducibles to support and filter relaxations is novel. Bylander et al. brings an extensive study of the computational complexity of abduction in [3]. The relation to our work lies at the discussion on explanations and plausibility. Their work considers comparing explanations to find the best ones. Our approach builds on criteria for rationality. Our explanations are partially ordered to indicate the best explanations in similarity to the Best-Small Plausibility Criterion introduced by these authors. This concept suggests that smaller explanations are preferable over bigger explanations of same plausibility. Their work, however, is not driven to cooperative query answering.

Godfrey motivates and presents several complexity results about minimal failing (MFS) and maximum succeeding subqueries (MSS) in [7]. These concepts introduce preference criteria over relaxation possibilities to produce answers as close as possible to what would satisfy the query. This approach is the closest to ours, but it is not based on abduction. The explanations with minimal priority grade in our preference relation $T_{\preccurlyeq}$ relate to maximum succeeding subqueries. Our filter is more suitable to agents, as it relies on the explanatory power of an agent and restrictions imposed by the author of a query.

An approach to cooperative answering based on Abstract Interpretation [8] has been proposed by Halder and Cortesi. In this approach, a query is rewritten into an abstract query as its strict values of arguments are replaced for concepts that generalize them. Such concepts are specified during the design of the database (or knowledge base) in an abstract database that consists of a copy of each data record with all arguments replaced by their corresponding abstract concepts. The abstract query is executed over the abstract database and the

corresponding records of the original database are retrieved as cooperative answers. This approach considers a single possibility for relaxation of queries and the quality of the answers retrieved is highly dependent on the design of the abstract database.

Pivert et al. propose replacing a failed query by a related one which has been previously processed and whose answer is known to be non-empty [12]. This approach uses fuzzy sets to model semantic approximation of queries. This concept is used as criteria to determine which cached query is the most similar to the one being replaced. This strategy is different from query relaxation, so related queries might not subsume the original. For that reason, this approach is only suitable to deal with failed queries.

There are also other approaches to hypothetical reasoning in logic programming that could be considered instead of Abductive Logic Programming. Inductive Logic Programming (ILP) [11] uses induction to construct clausal theories based on positive and negative examples. One difference is that abductive hypothesis consist of only ground terms, while an inductive hypothesis consists of a rule that generalizes the knowledge from the examples, therefore not ground. This kind of reasoning is commonly used for machine learning and could be applied to cooperative answering to improve the knowledge base and reduce failure over time. Abduction is more adequate to deal with failure on the spot. Another approach, CR-Prolog [2] introduces Constraint Restoring rules (CR-Rules) to extended programs. Such rules are only applied in case the regular program is inconsistent and are designed to deal with event exceptions. CR-Rules can also represent explanations which can be ordered by preference of application. However, in contrast to ALP, all explanations would need to be preconceived and encoded as cr-rules. While those explanations could then be ranked according to a preference relation, our approach makes it much simpler for the agents to discard relaxation possibilities and restrict the search space.

8 Conclusion and Future Work

Our goal is to improve the quality of the related answers obtained by relaxation. In order to attain this objective we build on explanations and established criteria to inform the search. Our contributions are manifold. First, our approach makes it possible for the author of the query to restrict relaxations by forbidding that some conditions are replaced. We also established criteria to rank and discard explanations, electing the best ones and using them to reduce redundancies and node neighborhood in the search. Finally, we cover both the cases of failure and non satisfaction by the author about the answers retrieved. We show how to do this by building a failing query from the original succeeding one, so the agent produces explanations related to neighborhood answers. This approach can improve cooperation between agents, as an effort is made to retrieve relevant data in face of a question. In the near future we plan to investigate how agents can use abduction-based cooperative answers to cooperatively work in group decision situations. Our focus will be to build a protocol for agents to solve impasses and seek consensus.

Acknowledgements

Research partially supported by CAPES (PROCAD).

References

1. Andreasen, T., Christiansen, H.: Flexible query-answering systems modelled in metalogic programming. In: ECAI 1996 Workshop Knowledge Representation Meets Databases, pp. 1–7 (1996)
2. Balduccini, M.: Computing answer sets of cr-prolog programs. Technical report, Texas Tech. University (2006)
3. Bylander, T., Allemang, D., Tanner, M.C., Josephson, J.R.: The computational complexity of abduction. Artif. Intell. 49, 25–60 (1991)
4. Denecker, M., Kakas, A.C.: Abduction in logic programming. In: Kakas, A.C., Sadri, F. (eds.) Computational Logic: Logic Programming and Beyond. LNCS (LNAI), vol. 2407, pp. 402–436. Springer, Heidelberg (2002)
5. Gaasterland, T., Godfrey, P., Minker, J.: Relaxation as a platform for cooperative answering. J. Intell. Inf. Syst. 1(3/4), 293–321 (1992)
6. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. 9(3/4), 365–386 (1991)
7. Godfrey, P.: Minimization in cooperative response to failing database queries. International Journal of Cooperative Information Systems 6, 95–149 (1997)
8. Halder, R., Cortesi, A.: Cooperative query answering by abstract interpretation. In: Černá, I., Gyimóthy, T., Hromkovič, J., Jefferey, K., Králović, R., Vukolić, M., Wolf, S. (eds.) SOFSEM 2011. LNCS, vol. 6543, pp. 284–296. Springer, Heidelberg (2011)
9. Inoue, K., Sakama, C.: Abductive framework for nonmonotonic theory change. In: IJCAI, pp. 204–210 (1995)
10. Kakas, A.C., Kowalski, R.A., Toni, F.: The role of abduction in logic programming. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 5, pp. 235–324. Oxford University Press, Oxford (1998)
11. Muggleton, S., De Raedt, L.: Inductive logic programming: Theory and methods. J. Log. Program. (19/20), 629–679 (1994)
12. Pivert, O., Jaudoin, H., Brando, C., Hadjali, A.: A method based on query caching and predicate substitution for the treatment of failing database queries. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 436–450. Springer, Heidelberg (2010)
13. Sadri, F., Toni, F.: Active behaviour in deductive databases. Technical report (1996)
14. Sakama, C., Inoue, K.: An abductive framework for computing knowledge base updates. Theory Pract. Log. Program. 3, 671–715 (2003)
15. Sakama, C., Inoue, K.: Negotiation by abduction and relaxation. In: AAMAS, pp. 1022–1029 (2007)
16. Wetzel, G., Toni, F.: Semantic query optimization through abduction and constraint handling. In: Andreasen, T., Christiansen, H., Larsen, H.L. (eds.) FQAS 1998. LNCS (LNAI), vol. 1495, pp. 149–366. Springer, Heidelberg (1998)

Reasoning about Exceptions to Contracts*

Özgür Kafalı[1], Francesca Toni[2], and Paolo Torroni[3]

[1] Department of Computer Engineering - Boğaziçi University
34342, Bebek, İstanbul - Turkey
ozgurkafali@gmail.com
[2] Department of Computing - Imperial College London
London, UK
ft@imperial.ac.uk
[3] DEIS - University of Bologna
V.le Risorgimento, 2, 40136, Bologna - Italy
paolo.torroni@unibo.it

Abstract. We show an application of Assumption-Based Argumentation for reasoning about and handling exceptions in multiagent contracts. We show that this solution enjoys interesting properties regarding the ABA semantics of results obtained and the determinism of diagnostic answers. As a case study, we present the workings of the framework on a delivery process from e-commerce.

1 Introduction

Open multiagent systems enable distributed business process execution using autonomous agents. Each agent executes a different part of the process. While this provides some advantages (e.g., privacy), it also makes the process vulnerable to *exceptions*. For example, if a buyer does not receive a merchandise that was scheduled for delivery, it can conclude that there must have been an exception in the workings of the entire process. Clearly, an agent's misbehaviour affects others. Thus when such an exception occurs, the agent facing the exception needs to identify and diagnose the problem behind it, so as to handle it properly and get back to normal execution.

Exception handling, and diagnosis in particular, is a hard and complicated task. This is especially true in a distributed setting, because reasoning is local but information is not, and agents need to spend considerable effort in gathering the missing bits from the other agents or from the environment.

According to Kalech and Kaminka [15], who studied social diagnosis in the context of agent teams, diagnosis has two phases: (i) selection of the diagnosing agents, and (ii) diagnosis of the team state (by the selected agent). An accurate selection of the diagnosing agents is in general a non-trivial task, but it is essential in limiting information flooding. In the case of commitment-regulated multiagent business process execution, however, selection becomes an elementary task, because commitments already contain a reference to the agent (the

* This paper extends the AAMAS '11 poster paper [12].

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 225–242, 2011.

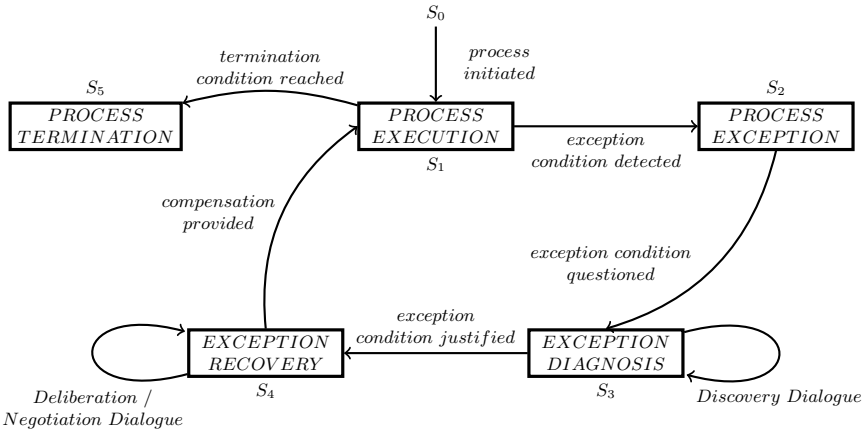


Fig. 1. Process Life-Cycle

so-called *debtor*) who may have caused a specific exception, or who may be able to find out what happened.

In [11], Kafalı *et al.* presented a distributed framework for handling exceptions in contract-based open multiagent systems. In their framework, agents perform monitoring tasks in order to detect possible misalignments and misbehaviours. However, their architecture does not tell how agents reason about exceptions and how they exchange the results of their reasoning.

This paper addresses precisely these two issues. We show that argumentation technologies are well suited to carry out such reasoning tasks and to support dialogues for collaborative diagnosis of multiagent contract exceptions. The dialogues provide the information exchange among the agents to enable diagnostic activities to step from agent to agent until the reason of the exception is found. Reasoning is based on the ABA argumentation framework [1,6,8,7]. Thanks to its grounding on consolidated argumentation theories, we are able to describe the diagnosis process in a high-level, declarative way, we can enable agents to construct hypotheses (arguments) about what went wrong and exchange such hypotheses between them, and we can ensure that the overall process is deterministic.

The diagnosis activities are embedded in an agent execution cycle, and they are performed whenever necessary. An agent executing a process assumes the role of diagnosing agents when he senses something wrong. Figure 1 shows the life-cycle of the multiagent system executing a process. The process begins execution as soon as it is initialized (e.g., the contracts between the agents are created), and it proceeds to normal execution (S_1), where it stays until it terminates or an exception condition is detected. In that case, the process enters the exception state (S_2) where the agent detecting the exception starts investigating the cause of the exception. This initiates the diagnosis process (S_3), which is carried out by local reasoning whenever the agent has sufficient information to

explain the exception, and by way of dialogues otherwise. When a valid justification is produced and agreed upon by the agents involved in the diagnosis, the process enters the recovery state (S_4). Ideally, if a reasonable compensation is found for the exception, e.g., after some negotiation, or by applying contractual clauses, the process goes back to normal execution state (S_1). Finally, the process terminates if a termination condition is reached (S_5). In this paper, we only focus on the diagnosis phase (S_3).

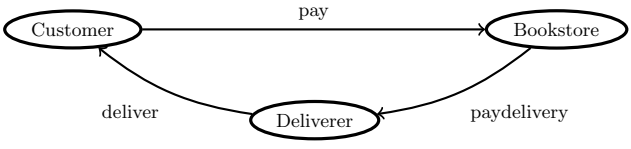


Fig. 2. Delivery Process

Throughout the paper we illustrate our proposal using an e-commerce scenario that describes a delivery process (see Figure 2). Such a scenario is inspired by the *MIT Process Handbook* [18]. It encompasses three business parties: a customer, a bookstore, and a deliverer. In a "normal" execution, the customer pays to the bookstore for an item. Then, the bookstore pays to the deliverer for the delivery of that item. Finally, the deliverer delivers the item to the customer. Social commitments [27] specify the mutual contractual obligations among the parties involved.

The rest of the paper is structured as follows: Section 2 reviews related work on multiagent diagnosis. Section 3 gives the necessary background on commitment-based contract modeling and outlines our diagnosis architecture. Section 4 introduces ABA, shows how to achieve diagnostic reasoning in ABA, and discusses properties of the formalization. Section 5 illustrates the workings of the framework on the delivery scenario. Section 6 discusses our contribution with respect to the state of the art and possible directions for future work.

2 Diagnosis in Multiagent Systems

Diagnosis is a process that starts from observing a deviation from the expected behaviour of a given system, and whose purpose is to identify the reason of the exception, i.e., to locate those subsystems whose abnormal behaviour accounts for the observed behaviour [20]. In multiagent systems, diagnosis is typically initiated by one specific agent, who detects an exception in the system, and interacts with other agents in order to isolate a problem in its own behaviour or in the behaviour of other agents.

The diagnosis problem is in general a hard one. For example, component-based diagnosis is in the worst case exponential in the number of annotated components [2]. Multiagent diagnosis presents additional problems, depending on the setting. In closed systems, such as teams, we may need to avoid information

flooding. In open domain, such as e-commerce settings, we may care for privacy of information and for the trust we put on autonomous, unknown agents.

There are two fundamentally different approaches to diagnostic reasoning: heuristic approaches, such as *fault-based diagnosis* (FBD) and diagnosis from the first principles or *model-based diagnosis* (MBD) [20]. In FBD, the idea is to encode the diagnostic reasoning of human experts in a given domain. The real-world system is not modeled. All known faults are instead modeled. Conversely, MBD starts from a model of the structure (components and their connections) and function (or behaviour) of the system, and a set of observations indicating an abnormal behaviour. A system is faulty if the observed behaviour of the system contradicts the behaviour predicted by assuming that all of its components are correct [3].

As pointed out by Kalech and Kaminka following Micalizio *et al.* [21], fault-based techniques [10,22,19,5], in which faults are modeled in advance, cannot be used for multiagent diagnosis, because the interactions in multiagent systems are unpredictable. For this reason, multiagent diagnosis is typically model-based. This is especially true in open systems, where the idea of social commitments is precisely to avoid enumerating all possible ways agents can interact in order to fulfill a contract, thus providing agents with more flexibility and opportunities [26].

Thus in recent years, the MBD approach has been applied to MAS diagnosis by several research groups, including Console *et al.* [4] with applications in the automotive industry [23], Roos *et al.* [30,25] for plan diagnosis, and Kalech and Kaminka [15,16] for coordination failures in agent teams. These are in general closed systems. For example, in [15], a coordination model is described by way of concurrency and mutual exclusion constraints. The approach assumes that each agent has knowledge of all the possible behaviours available to each team-member, i.e., their *behaviour library*. In this way, each observing agent creates a model of other agents in the team. Transitions between one behaviour to another are described in terms of pre-conditions and termination conditions. Then, a "social" diagnosis is initiated as a collaborative process aimed at finding causes of failures to maintain designer-specific social relationships [17]. More specifically to the case of team-work, a diagnosis is a set of contradictions in beliefs that accounts for the selection of different team behaviours by different agents [15].

Commitment exceptions are similar to coordination failures, and our diagnostic framework also follows an MBD approach. However, the setting and underlying assumptions are different, and so are the architecture, technology and, importantly, the concept of model.

In terms of assumptions, for one thing, we do not have a coordination model. Moreover, we do not restrict our work to cooperative teams, but we assume open systems, in which coordination is limited to social commitments about properties that must hold at some times (e.g., predefined deadlines). No assumptions are made on the behaviours of the agents that need to bring about such properties. Thus, we do not have an agent model, and we do not assume that agents share the same observations.

In terms of architecture, the idea is that an agent reasons locally, from the commitments and conditions that he is aware of, and progressively fills the gaps in his knowledge by way of inter-agent dialogues and local environmental sensing whenever needed. We do not have dedicated diagnosing agents. The agents that execute the process switch to diagnosis mode after they sense a commitment violation (see Figure 1), or if they are requested to do so by another agent (see Section 3). The underlying technology is computational argumentation, and in particular ABA, which can support both local diagnostic reasoning and dialogue-based interaction using the same language.

The main difference with the approaches mentioned above lays in our concept of model, which is at a more abstract level. At the agent level, the model is the set of domain-dependent rules about the process at hand. At the multiagent system level, the model is the set of commitments and their state. The notion of exception, or abnormal system behaviour, coincides with that of commitment violation. We are not after understanding what particular agent behaviour caused the exception, because an agent cannot know what causes or motivates other agents to behave in certain ways. The assumption of openness in the system prevents any assumption on the architecture of other agents. Our purpose is instead to explain the reason of an exception in terms of the state of the related commitments. Once we know that, responsible agents can be isolated, and we can proceed to the recovery phase.

3 Contracts, Commitments and Diagnosis Architecture

Social commitments [27] are artifacts that can be used to specify mutual contractual obligations among parties. A social commitment imposes an obligation on a "debtor" agent x towards a "creditor" agent y to bring about a certain property. Singh [27] considers two types of social commitments:

- *c(x, y, p(o))* is a **base-level commitment** between debtor x and creditor y to bring about the property described by proposition *p(o)*. When this commitment is created, it is said to be *active*, and debtor x is committed to creditor y for satisfying *p(o)*. *c(x, y, p(o))* stays *active* until its proposition gets either satisfied, in which case it becomes *fulfilled*, or violated, in which case it becomes *violated* [31].
- *cc(x, y, q(o), p(o))* is a **conditional commitment** between debtor x and creditor y to bring about the property described by proposition *p(o)* when condition *q(o)* holds. When this commitment is active, namely when *q(o)* is satisfied, x will become committed to y for satisfying *p(o)*. Thus a new base-level commitment *c(x, y, p(o))* is created, and we say that the conditional commitment is *fulfilled*. Note that a conditional commitment may never be violated.

We will use *c(X,Y,P(O))* and *cc(X,Y,Q(O),P(O))* as templates, whereby X and Y are variables standing for agents, O is a variable standing for an object, e.g., a

certain *book*, and *P(O)* and *Q(O)* are placeholders for atoms and literals (respectively) expressing properties about O, e.g., *delivered(book)* and $\neg$*delivered(book)*. All variables are implicitly universally quantified, and the actual commitments are instances of these templates. In the case of conditional commitments, when Y satisfies *Q(O)* with a specific substitute o for O, then X becomes committed to Y for satisfying *P(o)*.

We will assume that social commitments are represented with respect to an underlying *commitment language* $\mathcal{L}_c$ shared amongst all agents. In the remainder of the paper we will use the following conventions: as earlier in this section, uppercase letters denote variables; variables X, Y, Z are used to represent agents; P is used to represent an atomic property; Q is used to represent a property in the form of a literal (of the form P or $\neg P$); R is used to represent a conjunction of (literal) properties. As standard, $\neg\neg P$ is P. As in Prolog, _ represents an anonymous variable.

For simplicity, we will assume that there is at most one *cc(x, y, q(o), p(o))* for a given x, y, and *p(o)*. In other words, we do not consider those cases in which an agent x conditionally commits to the same agent y to bring about the very same property *p(o)* using two different contracts, which would unnecessarily complicate the diagnosis process. Such cases are left as an extension for future work. Also, note that, for the sake of simplicity and because we focus on diagnosis procedures, we will ignore (and not explicitly model) the *active* commitment state.

To describe the delivery process using contracts, we use two commitments. One tells that if *customer* pays for an *item*, then *bookstore* will be committed to deliver that item:

$$cc(bookstore, customer, paid(book), delivered(book))$$

The second one tells that if *bookstore* pays for the delivery of an *item*, then *deliverer* will be committed to deliver that item:

$$cc(deliverer, bookstore, paid\_delivery(book), delivered(book))$$

In real life, the delivery process does not always take its "normal" course. It may happen that the book does not get delivered. Usually, the customer is the first one to realize. We say that the customer detects an exception. Let us say we are now the customer. What shall we do? In order to recover the book, we should first understand what went wrong. We will consider several options: the bookstore may not have correctly place the order with the deliverer; the deliverer may be late; the bookstore may not have yet received the payment; and so on. A diagnosis process will tell us which of these possibilities truly explains what went wrong. The diagnosis process is initiated by the agent detecting an exception. During this process agents collect evidence from the environment and exchange information with each other.

The diagnosis architecture we rely upon to support the diagnosis process is that by Kafali *et al.* [11], in which agents sense and act upon an environment and exchange information about the state of their commitments. In this paper, such an exchange is carried out by way of dialogues.

Following the KGP agent model [14], we will assume that actions may be "physical" (e.g. *pay*), communicative (e.g. *justify*) or sensing actions on the environment (i.e. *question*). In the delivery process, we will consider the following physical actions:

- *pay(customer,bookstore,item):* a customer agent pays a bookstore agent for an item of interest.
- *paydelivery(bookstore,deliverer,item):* the bookstore pays a deliverer for the delivery of an item of interest.
- *deliver(deliverer,customer,item):* a deliverer delivers an item of interest to a customer.

The sensing actions amount to a particular type of database lookup for evidence gathering, that we call *evidence request exchange*, between an agent and some trustworthy element of the environment that can produce evidence. By doing so, we abstract away from the specific ways agents interact with the environment. Note that each agent has a partial view of the overall environment. Thus we call E_X the environment accessible to agent X. We use two types of sensing actions:

- *question*(X, E_X, P): agent X looks up E_X to check whether property P holds or not.
- *answer*(X, E_X, Q): agent X gets to know from E_X that Q holds.

Inter-agent dialogues are instead based on the following utterances (communicative actions):

- *explain*(X, Y, P): agent X sends a diagnosis request to Y, asking for a justification for a given property P.
- *justify*(X, Y, Q, P): agent X provides agent Y with a justification Q to why P holds.
- *rebut*(X, Y, Q, $\neg P$): agent X provides agent Y with a justification Q to why P does *not* hold.

In this paper we focus on the agents' reasoning to support the diagnosis process, and thus we do not discuss the agent execution model responsible for identifying actions to be performed. Several alternatives would be possible, e.g. using or devising control theories as in the KGP agent model [14].

4 Reasoning

We show how agents can reason about commitment exceptions based on assumption-based argumentation (ABA) [1,6,8,7], which we first briefly review below. We choose this framework because it can deal with inconsistent and incomplete information in general and in support of decision-making, it can generate justifications that can be communicated across agents, and because of its strong theoretical properties and the fact that it is equipped with provably

correct computational mechanisms, that will support any future deployment, e,g, using the publicly available CaSAPI ABA framework [9].[1]

Assumption-Based Argumentation (ABA) is a general-purpose argumentation framework where arguments and attacks between them are built from *ABA frameworks*, which are tuples $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ where:

- $(\mathcal{L}, \mathcal{R})$ is a *deductive system*, with $\mathcal{L}$ a language and $\mathcal{R}$ a set of inference rules,
- $\mathcal{A} \subseteq \mathcal{L}$, referred to as the set of *assumptions*,
- $\overline{\ }$ is a (total) mapping from $\mathcal{A}$ into $\mathcal{L}$, where $\overline{x}$ is referred to as the *contrary* of x.

In this paper, we assume that inference rules have the syntax $s_0 \leftarrow s_1, \ldots, s_n$ (for $n \geq 0$) where $s_i \in \mathcal{L}$. We refer to $s_1, \ldots, s_n$ as the *premises* and to s_0 as the *head* of the rule. If $n = 0$, we represent a rule simply by its head and we call the rule a *fact*. As in [6], we will restrict attention to *flat* ABA frameworks, such that no assumption occurs in the head of a rule.

Rules may be domain-dependent or not, and some of the premises of rules may be assumptions. These can be used to render the rules defeasible. In this setting, contraries of assumptions can be seen as representing "defeaters". Assumptions can also be used to fill gaps in incomplete knowledge/beliefs, and in this setting contraries are reasons for not making some gap-filling choices. Also, assumptions can be used to resolve inconsistencies (by making these depend upon assumptions that can be defeated).

An *(ABA) argument* in favour of a sentence $c \in \mathcal{L}$ supported by a set of assumptions $A \subseteq \mathcal{A}$ is a proof of c from A and (some of) the rules in $\mathcal{R}$. This proof can be understood as a tree (with root the claim and leaves the assumptions), as in [7], as a backward deduction, as in [6,8], or as a forward deduction, as in [1], equivalently. For the purposes of this paper, we will use the notation $A \vdash_R c$ to stand for an argument for c supported by A by means of rules $R \subseteq \mathcal{R}$. When the rules can be ignored, we write an argument $A \vdash_R c$ simply as $A \vdash c$. An argument $A \vdash c$ *attacks* an argument $A' \vdash c'$ if and only if $c = \overline{\alpha}$ for some $\alpha \in A'$.

Several "semantics" for ABA have been defined in terms of sets of assumptions fulfilling a number of conditions. These are expressed in terms of a notion of attack between sets of assumptions, where $A \subseteq \mathcal{A}$ attacks $A' \subseteq \mathcal{A}$ if and only if there is an argument $B \vdash c$, with $B \subseteq A$, attacking an argument $B' \vdash c'$, with $B' \subseteq A'$.

In this paper we will focus on the following notions:

- $A \subseteq \mathcal{A}$ is *conflict-free* if and only if A does not attack itself
- $A \subseteq \mathcal{A}$ is *admissible* if and only if A is conflict-free and attacks every $B \subseteq \mathcal{A}$ that attacks A
- $A \subseteq \mathcal{A}$ is *preferred* if and only if A is (subset) maximally admissible.

Note that these notions can be equivalently expressed in terms of arguments, rather than assumptions, as shown in [8].

[1] http://www.doc.ic.ac.uk/~dg00/casapi.html

Given an ABA framework $\mathcal{F}=\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ and a (conflict-free, admissible or preferred) set of assumptions $A \subseteq \mathcal{A}$ in $\mathcal{F}$, the (conflict-free, admissible or preferred, respectively) *extension* $\mathcal{E}_{\mathcal{F}}(A)$ is the set of all sentences supported by arguments with support a set of assumptions $B \subseteq A$:

$$\mathcal{E}_{\mathcal{F}}(A) = \{s \in \mathcal{L} | \exists B \vdash s \text{ with } B \subseteq A\}.$$

Note that conflict-free, admissible and preferred extensions are guaranteed to exist, for any ABA framework.

In the remainder of this section, we give an ABA framework supporting the reasoning of our agents. We will assume that each agent is equipped with an ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ such that the commitment language, $\mathcal{L}_c$, is contained in the internal language $\mathcal{L}$ underlying the ABA framework, namely $\mathcal{L}_c \subseteq \mathcal{L}$. We will leave this $\mathcal{L}$ implicit, and focus on rules, assumptions and contraries. Indeed, $\mathcal{L}$ will always consist of the set of all sentences occurring in all the given rules, as well as the given assumptions and contraries. We will give rules/assumptions/contraries as schemata, standing for all their ground instances over appropriate universes (for agents and properties). Until section 5 we will not focus on any specific agent and define rules, assumptions and contraries for a generic agent X. Assumptions will be of the form $\underline{asm(\_)}$. The contrary of $asm(a)$ will be of the form $c\_asm(a)$, for any a, formally: $\overline{asm(a)} = c\_asm(a)$.

4.1 Domain-Dependent Rules

These depend on the domain of application. In our example, these rules are related to the delivery process and include:

(F_1) *by_contract*(*cc*(*bookstore*, *customer*, *paid*(*book*), *delivered*(*book*))).

(F_2) *by_contract*(*cc*(*deliverer*, *bookstore*, *paid_delivery*(*book*), *delivered*(*book*))).

(F_3) *effect*(*pay*(*customer*, *bookstore*, *book*), *paid*(*book*)).

(F_4) *effect*(*paydelivery*(*bookstore*, *deliverer*, *book*), *paid_delivery*(*book*)).

(F_5) *effect*(*deliver*(*deliverer*, *customer*, *book*), *delivered*(*book*)).

(R_1) *justification*($\neg$*paid_delivery*(*book*), $\neg$*delivered*(*book*)) $\leftarrow$
$\neg$*paid_delivery*(*book*), $\neg$*delivered*(*book*).

The first two facts model the conditional commitment between the *customer* and *bookstore* agents (F_1) and between the *bookstore* and *deliverer* agents (F_2). The other facts $(F_3$–$F_5)$ describe the action-consequence relations. The rule R_1 represents that a problem in the delivery payment may be the reason for no delivery.

We will assume that the domain-dependent rules only define predicates *by_contract*(_), *effect*(_, _), and *justification*(_, _) as well as the predicates *believe*(_), *answer*(_, _), and *executed*(_) used below.

4.2 General-Purpose Reasoning Rules

These rules are held by agent X, independently of the specific scenario for exception diagnosis. They consist of belief rules (BR), commitment rules (CR) and action rules (AR).

Belief rules allow to "internalise" beliefs drawn from observations and expected effects of actions, unless there are reasons not to do so. They are required to avoid epistemic inconsistencies to arise, such as *believe(paid(book))* and *believe(¬paid(book))*.

(BR_1) $P \leftarrow believe(P), asm(P).$
(BR_2) $\neg P \leftarrow believe(\neg P), asm(\neg P).$
(BR_3) $believe(\neg P) \leftarrow asm(believe(\neg P)).$
(BR_4) $P \leftarrow answer(X, E_X, P).$
(BR_5) $\neg P \leftarrow answer(X, E_X, \neg P).$
(BR_6) $believe(Q) \leftarrow executed(A), effect(A, Q).$

Belief rules BR_1–BR_3 are defeasible, as represented by the presence of assumptions in their premises. The following rules for the contraries of the assumptions are their defeaters:

(BR_7) $c\_asm(P) \leftarrow \neg P.$
(BR_8) $c\_asm(\neg P) \leftarrow P.$
(BR_9) $c\_asm(believe(\neg P)) \leftarrow believe(P).$

Note that rules BR_1–BR_2 could be combined within a single rule

$$Q \leftarrow believe(Q), asm(Q).$$

Similarly for BR_4–BR_5 and BR_7–BR_8. However, we prefer to leave them separate for clarity, and to better underline the asymmetry in our treatment of positive and negative literals (properties). In particular, note that rules BR_3 and BR_9, together, model a form of closed-world assumption/negation-as-failure over properties, where $believe(\neg(\neg P))$ can be interpreted as the negation-as-failure of P.

Rules BR_4–BR_9 are strict, as there are no assumptions in their premises. Rules BR_4–BR_5 allow to turn observations (that a specific answer has been obtained after consulting the agent's environment E_X) into (internalised) beliefs. The fact that these rules are strict represents that we consider the environment as beyond doubt.

Note however that, should this not be the case, we could turn BR_4–BR_5 into defeasible rules by adding suitable assumptions and definitions for their contraries. Rule BR_6 allows to introduce (non-internalised) beliefs about the effects of executed actions. These beliefs may then become internalised by

applying rules BR_1–BR_2. Specific definitions of effects of actions belong to the domain-dependent part of the beliefs (see section 4.1).

As an illustration of the use of these rules extended with domain-dependent rules and under the notion of admissible sets of assumptions/arguments, consider the following cases:

- *believe(p)* is the only domain-dependent rule;
  then $\{asm(p)\}$ is the only (non-empty) admissible set of assumptions, supporting argument $\{asm(p)\} \vdash p$ in favour of p;
- *believe(¬p)* is the only domain-dependent rule;
  then $\{asm(\neg p)\}$ and $\{asm(believe(\neg p))\}$ and their union are the only (non-empty) admissible sets of assumptions, all supporting arguments in favour of $\neg p$;
- *believe(p)* and *believe(¬p)* are the only domain-dependent rules;
  then $\{asm(p)\}$ and $\{asm(\neg p)\}$ are the only (non-empty) admissible sets of assumptions, representing alternative choices for resolving the given inconsistency; the agent can choose whichever alternative;
- there are no domain-dependent rules;
  then $\{asm(\neg p), asm(believe(\neg p))\}$ is the only (non-empty) admissible set of assumptions, supporting an argument in favour of $\neg p$.

Note that (i) in no case an agent can assume both $asm(p)$ and $asm(\neg p)$, (ii) it can only derive p if it has some evidence for p, and (iii) it can only assume $asm(believe(\neg p))$ if it cannot derive *believe(p)*. Thus, the following result holds:

Property 1. Given an ABA framework with rules $BR_1 - BR_9$ and any set of domain-dependent rules as in section 4.1

1. every preferred extension $\mathcal{E}$ is *consistent*, namely there exists no property P such that both P and $\neg P$ belong to $\mathcal{E}$;
2. every preferred extension $\mathcal{E}$ is *total*, namely for every property P either P or $\neg P$ belongs to $\mathcal{E}$.

This result directly follows from our definition of belief rules and from the definition of preferred extensions in ABA.

Commitment rules model the evolution of commitments (atoms of the form *by_contract(...)*) and commitment states (atoms of the form *fulfilled*(...) and *violated*(...)) [2] during the agent's life-cycle.

(CR_1) $fulfilled(c(X,Y,P)) \leftarrow by\_contract(c(X,Y,P)), P,$
$asm(fulfilled(c(X,Y,P))).$

(CR_2) $by\_contract(c(X,Y,P)) \leftarrow by\_contract(cc(X,Y,Q,P)), Q,$
$asm(by\_contract(c(X,Y,P))).$

[2] As discussed in section 3, we ignore the *active* commitment state.

(CR_3) $violated(c(X,Y,P)) \leftarrow by\_contract(c(X,Y,P)), \neg P,$
$asm(violated(c(X,Y,P)))$.

(CR_4) $c\_asm(fulfilled(c(X,Y,P))) \leftarrow \neg P$.

(CR_5) $c\_asm(by\_contract(c(X,Y,P))) \leftarrow by\_contract(cc(X,Y,Q,P)), \neg Q$.

(CR_6) $c\_asm(violated(c(X,Y,P))) \leftarrow P$.

Note that, like belief rules, commitment rules may also be defeasible, since commitments change during the agent's life-cycle. In particular, $CR_1 - CR_3$ above are defeasible.

Property 2. Given an ABA framework with rules $CR_1 - CR_6$, $BR_1 - BR_9$ and any set of domain-dependent rules as in section 4.1, for any preferred extension $\mathcal{E}$, for any agent Y:

- there exists no property P such that $fulfilled(c(X,Y,P))$ and $violated(c(X,Y,P))$ belong to $\mathcal{E}$;
- for every property P such that $by\_contract(cc(X,Y,P))$ belongs to $\mathcal{E}$, either $fulfilled(c(X,Y,P))$ or $violated(c(X,Y,P))$ belongs to $\mathcal{E}$.

Part 1 follows from part 1 of result 1, since P and $\neg P$ are respectively conditions of CR_1 and CR_3. Part 2 follows from part 2 of result 1.

Action rules are of two types: for determining whether and how to consult the environment (action *question*) or for determining whether and how to conduct a *request explanation* dialogue. The first type of rules are:

(AR_1) $question(X, E_X, \neg P) \leftarrow violated(c(Y,X,P)), asm(question(X, E_X, \neg P))$.

(AR_2) $question(X, E_X, \neg Q) \leftarrow violated(c(Y,X,P)), by\_contract(cc(Y,X,Q,P)),$
$asm(question(X, E_X, \neg Q))$.

where contraries of assumptions are defined by rules:

(AR_3) $c\_asm(question(X, E_X, \neg P)) \leftarrow by\_contract(c(Y,X,P)),$
$answer(X, E_X, \neg P)$.

(AR_4) $c\_asm(question(X, E_X, \neg Q)) \leftarrow by\_contract(cc(Y,X,Q,P)),$
$answer(X, E_X, \neg P), asm(question(X, E_X, \neg P))$.

(AR_5) $c\_asm(question(X, E_X, \neg Q)) \leftarrow by\_contract(cc(Y,X,Q,P)),$
$answer(X, E_X, \neg Q)$.

AR_3 and AR_5 prevent a *question* by X to its environment on a violated property or a commitment condition if the question has already been answered. AR_4 forces a preference of a question about a violated commitment over a question about the condition of that commitment.

The second type of action rules regulate dialogues between agents. These are as follows:

(DR_1) $explain(X, Y, \neg P) \leftarrow violated(c(Y, X, P)), by\_contract(cc(Y, X, Q, P)),$
$answer(E_X, X, \neg P), answer(E_X, X, Q), asm(explain(X, Y, \neg P)).$

(DR_2) $c\_asm(explain(X, Y, \neg P)) \leftarrow violated(c(Y, X, P)), violated(c(Z, Y, P)).$

Namely, in case of a violation over P by X towards Y, and after having already checked within its environment, X asks Y for an explanation (DR_1), unless Y itself is object of a violation on P by some other agent Z (DR_2).

(DR_3) $justify(X, Y, R, \neg P) \leftarrow explain(Y, X, \neg P), justification(R, \neg P),$
$asm(justification(R, \neg P)).$

(DR_4) $rebut(X, Y, R, P) \leftarrow explain(Y, X, \neg P), justification(R, P),$
$asm(justification(R, P)).$

(DR_5) $c\_asm(justification(R, X)) \leftarrow justification(R, \neg X).$

5 Case Study

We present here two case studies of the diagnosis process, in the simple delivery scenario we have used throughout the paper.

5.1 Customer's Fault

This case presents the trace of the diagnosis process on the exception that is caused by the customer (the customer has not paid for the item correctly, although he thinks he did).

The customer's domain-dependent rules initially consist solely of $F_1, F_3 - F_5$ in section 4.1. The general rules are all rules in section 4.2. Let us refer to the resulting ABA framework as $\mathcal{F}$. After the customer pays for the book, one fact is added to $\mathcal{F}$: (F_6) $executed(pay(customer, bookstore, book))$ resulting in a new ABA framework $\mathcal{F}'$.

1. By F_6, F_3, BR_6 and BR_1, the property $paid(book)$ becomes supported in the unique preferred extension P of $\mathcal{F}'$, with argument (where we use p to stand for the property)
   - $\{asm(p)\} \vdash_{\{F_6, F_3, BR_6, BR_1\}} p$.
2. Moreover, using additionally CR_2 and F_1, a contract

   $$by\_contract(bookstore, customer, delivered(book))$$

   can also be derived in the context of P, supported by the argument (where we use c to stand for the contract and $\mathcal{R}_1$ to stand for $\{F_6, F_3, BR_6, BR_1\}$)
   - $\{asm(p), asm(c)\} \vdash_{\mathcal{R}_1 \cup \{CR_2, F_1\}} c$.
3. Then, the customer realizes that the book has not been delivered, supported by argument (where we use $\neg d$ to stand for $\neg delivered(book)$)

– $\{asm(\neg d)\} \vdash_{\{BR_3\}} \neg d$.

4. This causes an exception, since an argument for a violation can be supported in P using, additionally, CR_3 ($\mathcal{R}_2 = \mathcal{R}_1 \cup \{CR_2, F_1\} \cup \{BR_3\}$ and v stands for $violated(c(bookstore,\ customer, delivered(book)))$)
   – $\{\ asm(p), asm(c), asm(\neg d), asm(v)\ \} \vdash_{\mathcal{R}_2 \cup \{CR_3\}} v$.
5. When the time comes for the agent to think about the possible reasons of the exception, the action rules give support to a possible question for the environment of the customer (E_c). Namely, the following argument is supported (where q stands for $question(customer,\ E_c, \neg paid(book)$ and $\mathcal{R}_3 = \mathcal{R}_2 \cup \{CR_3\}$)
   – $\{\ asm(p), asm(c), asm(\neg d), asm(v), asm(q)\ \} \vdash_{\mathcal{R}_3 \cup \{AR_1\}} q$.

   Thus, since $q \in P \cap Actions$, the agent cycle selects to perform action q.
6. Assume, in return, the customer learns that he did not pay, namely

$$(F_7)\ answer(E_c, \neg paid(book))$$

   is observed and added to $\mathcal{F}'$, resulting in a new ABA framework $\mathcal{F}''$. Let P' be the preferred extension of $\mathcal{F}''$. P' supports the conclusion that the customer did not actually pay for the book, namely
   – $\{\} \vdash_{\{F_7, BR_5\}} \neg p$

   as well as the argument
   – $\{\} \vdash_{\{F_7, BR_5, BR_7\}} c\_asm(p)$.

   This attacks all arguments including $asm(p)$ in their support, e.g. the argument given at step 4 (as well as the arguments given at steps 1, 2 and 5). Thus, v does not belong to P' and therefore the customer is aware that no commitment is violated (even though no delivery has occurred). Moreover, the customer knows that it has not paid for the book correctly (first argument at this step).

At the end of this process, the customer has been able to remove the exception.

5.2 Bookstore's Fault

Now imagine that the bookstore has not paid for the delivery of the item correctly.

As in the previous case, the customer's domain-dependent rules initially are $F_1, F_3 - F_5$ in section 4.1. The general rules are all rules in section 4.2. Let us refer to the resulting ABA framework as $\mathcal{F}_c$. The bookstore's domain-dependent rules initially consist of $F_1 - F_5$ and R_1 in section 4.1. Again, the general rules are all rules in section 4.2. Let us refer to the resulting ABA framework as $\mathcal{F}_b$. The customer's reasoning starts and proceeds as before.

However, at step 6, the answer

$$(F_8)\ answer(E_c, paid(book))$$

is observed instead and added to the ABA framework of the customer. Then the dialogue rules lead the customer to ask the bookstore to explain why the book has not been delivered.

7. An argument exists in favour of

$$explain(customer, bookstore, \neg delivered(book))$$

using rule such as DR_1 to construct an argument for

$$violated(c(bookstore, customer, delivered(book))$$

(see step 4 in previous case), supported by

$$asm(explain(bookstore, customer, delivered(book)))$$

as well as all assumptions for the argument at step 4 in the previous case.
8. Upon receipt of such an explanation request, the bookstore also derives the conclusion that the book has not been delivered, similarly to step 3 in the previous case (but now the bookstore is constructing the arguments). Then, again similarly to the previous case, step 4, a violation is detected (by the bookstore now), that

$$violated(c(bookstore, customer, delivered(book)))$$

and, in addition, a further violation

$$violated(c(deliverer, bookstore, delivered(book)))$$

using rule F_2 instead of F_1.
9. The bookstore's ABA framework at this point supports an argument in favour of a new action of querying the environment of the bookstore (E_b):

$$question(bookstore, E_b, \neg paid\_delivery(book))$$

10. On observation of

$$answer(E_b, bookstore, \neg paid\_delivery(book))$$

the bookstore knows that he has not paid for delivery, therefore the set of conclusions supported by his ABA framework change. In particular,

$$violated(c(deliverer, bookstore, delivered(book)))$$

is not supported any more, while a *justify* for the failed delivery is now supported.
11. Using rule DR_4 and R_1, the bookstore can now answer the customer's request for explanation at step 7:

$$justify(bookstore, customer, \neg paid\_delivery(book), \neg delivered(book))$$

6 Conclusion and Future Work

Diagnosis of commitments exceptions is a major issue in multiagent process execution. However, it is a hard and complicated task, because the handling of an exception requires significant information exchange and because of the very nature of the information exchanged.

In this paper, we have shown that argumentation technologies, in particular ABA, can be used as a principled way to model agent reasoning about commitment exceptions, and as a basis to run diagnosis dialogues. In spite of the many works on argumentation-based negotiation [24], to the best of our knowledge, this is the first attempt to use argumentation theories in the context of multiagent contracts and exception handling.

Dialogues for diagnosis are also another novel contribution, as diagnosis does not seem to fit in any of the purposes of dialogues identified by Walton & Krabbe [29] and in other relevant literature.

This work distinguishes itself from some influential approaches to multiagent diagnosis, in that we provide a high-level model of the system (the commitments in place and their states), but not of the system behaviour, nor the individual models of agents. Since we employ commitment protocols, which enable flexible execution for agents, (e.g., agent behaviour is not significant as long as commitments are fulfilled), we identify abnormal behaviour (inconsistencies) with commitment violation. The idea is to explain the reason of an exception in terms of the state of the related commitments. Once we know that, responsible agents can be isolated, and we can take the necessary steps towards recovery.

Related research on handling commitment exceptions has been carried out by Kafalı *et al.* [11,13], but without saying anything about agent reasoning. Such an integration is enabled here by the underlying ABA argumentation logic. In this way we can express knowledge and reasoning in a declarative and modular way, and show properties about the overall diagnosis process.

In the future, we intend to work on a formal analysis of our framework. Many other important properties could be investigated. One interesting direction is the identification of (necessary and) sufficient conditions for the existence of solutions in diagnosis. Another one is a complexity analysis. Moreover, as we mentioned in Section 2, complexity is an issue in (multiagent) diagnosis. However, we believe that social commitments can mitigate this problem, because they reify that knowledge about the coordination model execution state, which typically needs to be inferred, thus reducing the computational cost of tasks such as monitoring, exception detection and diagnosis.

Finally, temporal reasoning has been recognized to be a very important aspect of commitment specification and handling [28]. Our framework does not accommodate such an aspect. To fill this gap, we plan to exploit the temporal reasoning capabilities of the KGP agent model [14], which we identified as a potential candidate for the embedding of this work.

Acknowledgements

The first author is supported by Boğaziçi University Research Fund under grant BAP5694, and the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610. We thank Paola Mello for her comments and suggestions, and the anonymous reviewers for their valuable feedback.

References

1. Bondarenko, A., Dung, P., Kowalski, R., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. Artificial Intelligence 93(1-2), 63–101 (1997)
2. Bylander, T., Allemang, D., Tanner, M.C., Josephson, J.R.: The computational complexity of abduction. Artificial Intelligence 49(1-3), 25–60 (1991)
3. Console, L., Dressler, O.: Model-based diagnosis in the real world: Lessons learned and challenges remaining. In: IJCAI 1999: 16th International Joint Conference on Artificial Intelligence, pp. 1393–1400 (1999)
4. Console, L., Dupré, D.T., Torasso, P.: Towards the integration of different knowledge sources in model-based diagnosis. In: Ardizzone, E., Sorbello, F., Gaglio, S. (eds.) AI*IA 1991. LNCS, vol. 549, pp. 177–186. Springer, Heidelberg (1991)
5. Dellarocas, C., Klein, M., Rodríguez-Aguilar, J.A.: An exception-handling architecture for open electronic marketplaces of contract net software agents. In: ACM Conference on Electronic Commerce, pp. 225–232 (2000)
6. Dung, P., Kowalski, R., Toni, F.: Dialectic proof procedures for assumption-based, admissible argumentation. Artificial Intelligence 170(2), 114–159 (2006)
7. Dung, P., Kowalski, R., Toni, F.: Assumption-based argumentation. In: Rahwan, I., Simari, G. (eds.) Argumentation in AI, pp. 199–218. Springer, Heidelberg (2009)
8. Dung, P., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artificial Intelligence 171(10-15), 642–674 (2007)
9. Gaertner, D., Toni, F.: Computing arguments and attacks in assumption-based argumentation. IEEE Intelligent Systems 22(6), 24–33 (2007)
10. Horling, B., Benyo, B., Lesser, V.R.: Using self-diagnosis to adapt organizational structures. In: Agents 2001: 5th International Conference on Autonomous Agents, pp. 529–536 (2001)
11. Kafalı, Ö., Chesani, F., Torroni, P.: What happened to my commitment? Exception diagnosis among misalignment and misbehavior. In: Dix, J., Leite, J., Governatori, G., Jamroga, W. (eds.) CLIMA XI. LNCS, vol. 6245, pp. 82–98. Springer, Heidelberg (2010)
12. Kafalı, Ö., Toni, F., Torroni, P.: Collaborative diagnosis of exceptions to contracts (extended abstract). In: AAMAS 2011: 10th International Conference on Autonomous Agents and Multiagent Systems. IFAAMAS, pp. 1167–1168 (2011)
13. Kafalı, Ö., Yolum, P.: Detecting exceptions in commitment protocols: Discovering hidden states. In: Dastani, M., El Fallah Segrouchni, A., Leite, J., Torroni, P. (eds.) LADS 2009. LNCS, vol. 6039, pp. 112–127. Springer, Heidelberg (2010)
14. Kakas, A.C., Mancarella, P., Sadri, F., Stathis, K., Toni, F.: Computational logic foundations of KGP agents. Journal of Artificial Intelligence Research 33, 285–348 (2008)
15. Kalech, M., Kaminka, G.A.: On the design of social diagnosis algorithms for multi-agent teams. In: IJCAI 2003: 18th International Joint Conference on Artificial Intelligence, pp. 370–375 (2003)

16. Kalech, M., Kaminka, G.A.: Towards model-based diagnosis of coordination failures. In: AAAI 2005: 20th National Conference on Artificial intelligence, pp. 102–107 (2005)
17. Kaminka, G.A., Tambe, M.: Robust agent teams via socially-attentive monitoring. Journal of Artificial Intelligence Research 12, 105–147 (2000)
18. Klein, M., Dellarocas, C.: A systematic repository of knowledge about handling exceptions in business processes. Tech. Rep. ASES-WP-2000-03. Massachusetts Institute of Technology, Cambridge, MA, USA (2000)
19. Lamperti, G., Zanella, M.: Eden: An intelligent software environment for diagnosis of discrete-event systems. In: Applied Intelligence, pp. 55–77 (2003)
20. Lucas, P.J.F.: Analysis of notions of diagnosis. Artificial Intelligence 105(1-2), 295–343 (1998)
21. Micalizio, R., Torasso, P., Torta, G.: On-line monitoring and diagnosis of a team of service robots: A model-based approach. AI Communications 19, 313–340 (2006)
22. Pencole, Y., Cordier, M.O., Roze, L.: Incremental decentralized diagnosis approach for the supervision of a telecommunication network. In: IEEE Conference on Decision and Control, pp. 435–440 (2002)
23. Picardi, C., Bray, R., Cascio, F., Console, L., Dague, P., Millet, D., Rehfus, B., Struss, P., Vallée, C.: Idd: Integrating diagnosis in the design of automotive systems. In: ECAI 2002: 15th European Conference on Artificial Intelligence, pp. 628–632. IOS Press, Amsterdam (2002)
24. Rahwan, I., Ramchurn, S.D., Jennings, N.R., Mcburney, P., Parsons, S., Sonenberg, L.: Argumentation-based negotiation. Knowledge Engineering Review 18, 343–375 (2003)
25. Roos, N., Witteveen, C.: Models and methods for plan diagnosis. Autonomous Agents and Multi-Agent Systems 19, 30–52 (2009)
26. Singh, M.P.: Agent communication languages: Rethinking the principles. IEEE Computer 31, 40–47 (1998)
27. Singh, M.P.: An ontology for commitments in multiagent systems: Toward a unification of normative concepts. Artificial Intelligence and Law 7, 97–113 (1999)
28. Torroni, P., Chesani, F., Mello, P., Montali, M.: Social commitments in time: Satisfied or compensated. In: Baldoni, M., Bentahar, J., van Riemsdijk, M.B., Lloyd, J. (eds.) DALT 2009. LNCS, vol. 5948, pp. 228–243. Springer, Heidelberg (2010)
29. Walton, D.N., Krabbe, E.C.W.: Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning. State University of New York Press, Albany (1995)
30. Witteveen, C., Roos, N., van der Krogt, R., de Weerdt, M.: Diagnosis of single and multi-agent plans. In: AAMAS 2005: 4th International Joint Conference on Autonomous Agents, pp. 805–812. ACM, New York (2005)
31. Yolum, P., Singh, M.P.: Flexible protocol specification and execution: applying event calculus planning using commitments. In: AAMAS 2002: 1st International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 527–534. ACM, New York (2002)

Probabilistic Rule Learning in Nonmonotonic Domains

Domenico Corapi[1], Daniel Sykes[1], Katsumi Inoue[2], and Alessandra Russo[1]

[1] Department of Computing
Imperial College London
180 Queen's Gate, SW7 2AZ
London, UK
{d.corapi,d.sykes,a.russo}@imperial.ac.uk
[2] National Institute of Informatics
Chiyoda-ku, 2-1-2, Hitotsubashi
Tokyo 101-8430, Japan
ki@nii.ac.jp

Abstract. We propose here a novel approach to rule learning in probabilistic nonmonotonic domains in the context of answer set programming. We used the approach to update the knowledge base of an agent based on observations. To handle the probabilistic nature of our observation data, we employ parameter estimation to find the probabilities associated with each of these atoms and consequently with rules. The outcome is the set of rules which have the greatest probability of entailing the observations. This ultimately improves tolerance of noisy data compared to traditional inductive logic programming techniques. We illustrate the benefits of the approach by applying it to a planning problem in which the involved agent requires both nonmonotonicity and tolerance of noisy input.

Keywords: Inductive Logic Programming, Probabilistic Logic Programming, Answer Set Programming, Hypothetical Reasoning, Planning.

1 Introduction

Traditional machine learning techniques assume preliminary knowledge about the domain elements which are critical to solve the learning task. This is often expressed as a task of finding a target function from a collection of examples, where the examples are assumed to be useful and relevant to the learned function. This is problematic in certain contexts. For example, a robot whose goal is to deliver an object to a location may fail to achieve this goal for a variety of reasons. Perhaps the motors failed, or the object was dropped along the way. It is not immediately obvious how to relate the observation (a failure) to the knowledge base of the robot. In other words, the extraction of features relevant to the learning task is itself a problem that must be considered.

Inductive logic programming (ILP) [16] is a technique, using logic programs as a representation language, that can be seen as a general-purpose method to find the relevant features amongst a large set of candidates [6] through a process of generalisation. The dependencies between features in the knowledge base and observations are then

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 243–258, 2011.

captured in a logic program. Such logic programs, when explicitly accounting for non-monotonicity, are expressive enough to be used for a wide range of applications, e.g. for planning tasks [29].

However, ILP techniques tend to produce poor results in domains where a logical representation alone cannot capture uncertainty since they are heavily reliant upon the concept of logical implication, which limits their capacity to cope with erroneous training examples and knowledge.

We can improve upon this dependency by recognising that the example data and the knowledge available about the domain naturally suffer from noise. Explicit probabilistic treatment of the observations and the knowledge also has the advantage of producing 'higher' abstraction as insignificant aspects of the examples are ignored.

In this work, we develop a novel rule learning approach, which builds on an existing technique that transforms ILP to abductive reasoning [4] (enabling us to handle nonmonotonic domains) and techniques for estimating probabilities of logic facts [10], [11]. Answer set programming (ASP) [17] is used to generate candidate rules for the ILP problem, which is to say, rules that logically entail some of the observations. Gradient descent is then used to estimate probabilities for the rules so that the error between the expected probability of the observations (as calculated) and the actual probability is minimised. The set of rules that most closely fits the observations is given as the result. Our main contribution is to show how through an encoding technique that represents logic rules as logic atoms, established techniques can be adapted to derive structured rules that improve a nonmonotonic theory, together with a probabilistic weight by estimating the probabilities of these logic atoms.

We have applied our approach in the context of learning rules to perform planning for a reactive agent. While reactive planning already has the benefit of making choices on the basis of sensing during execution, there remain situations when multiple actions can be performed in the same state. Standard reactive planning chooses one such action arbitrarily, but in the uncertain physical environments in which planning is often used, greater reliability can be achieved by choosing the actions which are most likely to lead to success of the overall plan.

The rest of the paper is organised as follows. Section 2 introduces some preliminary definitions. Section 3 describes the approach (NoMPRoL) in detail. Section 4 introduces our planning case study and the results gained thereof. Section 5 discusses the contribution and related work. Section 6 describes future work and concludes.

2 Preliminaries

We assume that the reader is familiar with logic programming [18]. Given a logic-based alphabet consisting of variables, constants and predicates, an *atom* is an expression of the form $p(t_1, .., t_n)$, where p is a predicate and t_i are terms (variable or ground) in the alphabet. An atom can be negated using *negation as failure*. *Literals* are atoms a or negated atoms $\text{not } a$. We say that $\text{not } a$ is true if we cannot find evidence supporting the truth of a. Atoms and literals are used to create *rules* of the form: $a \leftarrow b_1, ..., b_m, \text{not } c_1, ..., \text{not } c_n$, where a, b_i and c_j are atoms. Intuitively, this means *if all atoms b_i are known/true and no atom c_i is known/true, then a must be known/true*.

We refer to a as the head and $b_1, ..., b_m, \text{not } c_1, ..., \text{not } c_n$ as the body of the rule. A *(normal) logic program (or theory)* is a conjunction of rules and is also denoted by a set of rules. The semantics is defined in terms of *answer sets* [17], i.e. assignments of true and false to all atoms in the program that satisfy the rules in a minimal and consistent fashion. A program has zero or more answer sets, each corresponding to a solution.

Our approach makes use of abductive reasoning, i.e. reasoning about explanations for given observations. An *abductive logic program* is a pair $\langle \mathcal{T}, A \rangle$ where $\mathcal{T}$ is a theory, and A is set *abducible* atoms. An abductive solution Δ is the subset of abducibles that must be added to the theory in order to obtain a particular answer set. We refer to the answer sets M associated with some answer Δ as *generalised answer sets* [12].

Definition 1. *Let $\langle \mathcal{T}, A \rangle$ be an abductive logic program and $\Delta \subseteq A$ be an abductive solution. M is a* generalised answer set *for $\langle \mathcal{T}, A \rangle$ iff M is an answer set of $\mathcal{T} \cup \Delta$.*

We use the notation Δ_M to denote the abductive solution associated with generalised answer set M, i.e. $\Delta_M = A \cap M$. $AS(\mathcal{T}, A)$ denotes all the generalised answer sets of $\langle \mathcal{T}, A \rangle$. Given an abductive logic program $\langle \mathcal{T}, A \rangle$, we assume that every possible $\mathcal{T} \cup \Delta$ has at most one answer set[1].

3 Approach

We present a general methodology that, given an existing knowledge base in the form of a logic program $\mathcal{T}$, and a set of observations X subject to noise, finds the set of rules with estimated probabilities that explain the observations. The approach, which we call NoMPRoL, is divided into three stages as shown in Figure 1. In the first stage, the task of finding rules for the given knowledge base is encoded as an abductive reasoning problem. A transformed knowledge base (with the observations) is then presented to an answer set solver which finds all solutions (i.e. models) of this input, which is to say, in addition to atoms relating to the original knowledge base, answer sets contain abducibles representing learned rules that entail the observations. By construction, as clarified in Section 3.1, such abductive solutions can be transformed into a set of rules whilst preserving the semantics. The third part of the approach is to estimate probabilities for abducibles to find a maximum likelihood hypothesis comprising the set of rules H and a probability distribution P_0^θ. This is achieved by using gradient descent to minimise a mean squared error function.

We define a probability distribution over a set of abductive solutions, instantiating the framework of [27], treating negation in a similar way as in [23].

Definition 2. *A* probabilistic theory *is a tuple $\langle \mathcal{T}, A, P_0^\theta \rangle$ where $\mathcal{T}$ is a theory, A is a set of abducible atoms and P_0^θ is a probability distribution over 2^A.*

P_0^θ depends on $|A|$ independent variables θ_a associated with the probability that the abducible atom $a \in A$ is true. We call the set of all such variables θ. Note that the independence assumption is common in the existing frameworks for probabilistic logic.

[1] This is true whenever $\mathcal{T}$ is acyclic. Under this assumption the unique answer set is characterised by the Clark's completion [3] of the program.

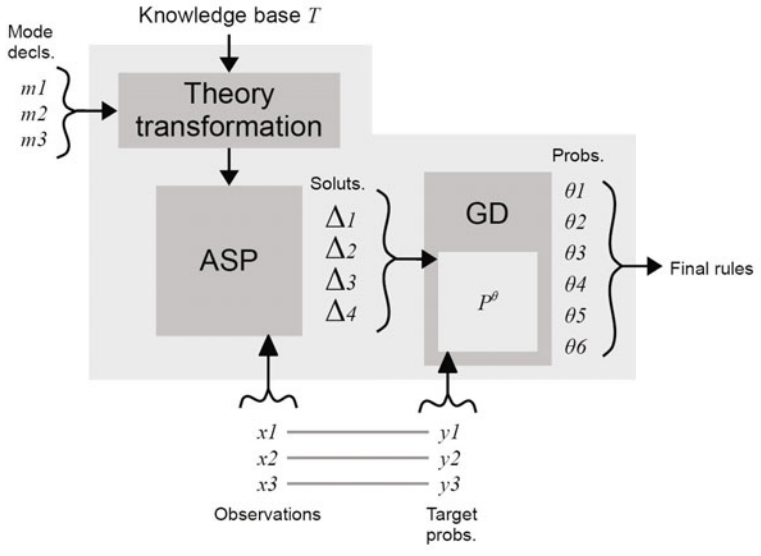


Fig. 1. Schematic view of the three phases in NoMPRoL

Mutual exclusivity of the variables can be modelled through integrity constraints (that in this work extend the *independent choices* in [23]). For any $\Delta \subseteq A$,

$$P_0^\theta(\Delta) = \prod_{a \in \Delta} \theta_a \prod_{a \in A \setminus \Delta} (1 - \theta_a)$$

defines the probability of sets of abducibles and indirectly of models that contain them. We define the probability of a logic literal as follows:

$$P^\theta(l|\langle \mathcal{T}, A \rangle) = \frac{\sum_{\{M : M \in AS(\mathcal{T},A), l \in M\}} P_0^\theta(\Delta_M)}{\sum_{\{M : M \in AS(\mathcal{T},A)\}} P_0^\theta(\Delta_M)} \quad (1)$$

The probability of a literal l is given by the sum of the probabilities of the abductive solutions that entail l normalised over the sum of the probabilities of all abductive solutions of the theory. The probabilities always implicitly refer to an underlying abductive logic program. If the abductive theory is clear from the context we use the notation $P^\theta(l)$.

3.1 ILP as Abductive Reasoning

ILP is used when a certain knowledge base must be enriched with rules that are also able to classify new examples. We follow the common practice of defining the space of acceptable rules (the *language bias*) with a set of mode declarations. *Head* and *body mode declarations* L [20] define the structure of the atoms (by means of a $schema$) that can appear in the body or in the head of the rules, hence defining the space $s(L)$ of possible hypotheses (the language bias).

Mode declarations are specified as $m : mode(t, r(m), s)$ where m is the label; the first argument specifies whether it is a head (h) or body (b) mode declaration; the second argument specifies the maximum number of occurrences of m; and the third argument specifies the *schema*. A schema s is a ground literal that contains one or more placemarkers. A *placemarker* is a ground function with one of the three symbols '+' for *input placemarkers*, '-' for *output placemarkers*, '#' for *constant placemarkers*. The only argument of the function is a constant called *type*.

Let r be a clause h :- $b_1, ..., b_n$ and L be a set of mode declarations. Then r is *compatible* with L (i.e. $r \in s(L)$) iff the following conditions are met:

1. h corresponds to the schema of a head mode declaration in L where all the input placemarkers and output placemarkers are replaced with variables and all the constant placemarkers are replaced with constants;
2. for each $b_i, i = 1, ..., n$, b_i corresponds to the schema of a body mode declaration in L where all the input placemarkers and output placemarkers are replaced with variables and all the constant placemarkers are replaced with constants;
3. every variable that replaces a input placemarker in any of the literals in the body either replaces a input placemarker in h or an output placemarker in some atom $b_j, j < i$.

We provide an example and refer to [20] for details.

Example 1. Consider the following mode declarations L:

$$m1 : mode(h, 1, daughter(+person, +person)).$$
$$m2 : mode(b, 1, mother(+person, +person)).$$
$$m3 : mode(b, 1, sex(+person, \#mf)).$$

Given a background theory $\mathcal{T}$

$$mf(m). \quad mf(f).$$
$$sex(ann, f). \quad sex(tom, m).$$
$$mother(mary, ann).$$

and a set of constraints $\mathcal{R} = \{$:- $not\ daughter(ann, mary)$, :- $daughter(tom, mary)\}$, the following labelled rules are compatible with L (i.e. $\{r1, r2\} \subseteq s(L)$):

$$r1 : daughter(X, Y) \text{ :- } sex(X, m).$$
$$r2 : daughter(X, Y) \text{ :- } mother(X, Y), sex(Y, m).$$

and the following rule:

$$r3 : daughter(X, Y) \text{ :- } mother(Y, X), sex(X, f).$$

is such that $\mathcal{T} \cup \mathcal{R} \cup \{r3\}$ has one generalised answer set (while $\mathcal{T} \cup \mathcal{R}$ has none) and $r3 \in s(L)$. Thus $r3$ is a possible solution and it defines the concept of $daughter$ based on the predicates $mother$ and sex, consistently with the constraints. Rules $r1$ and $r2$ are within the language bias but violate the given constraints (in turn they imply $daughter(tom, mary)$ and $not\ daughter(ann, mary)$).

We treat examples as constraints, so the task at hand is to find new rules that satisfy a set of given constraints. This is achieved by "lifting" the *abductive* reasoning to *inductive* reasoning using a similar representation to that in [4]. In contrast to [4], we use ASP mainly because it better supports the generation of all the solutions and the use of integrity constraints. To transform an ILP task into an abductive search task we represent all the rules within $s(L)$ as logical atoms that can be abduced using an answer set solver. Abductive solutions for the new theory can be transformed back into a solution for the equivalent ILP problem. The correctness and completeness of the transformations are obtained, as in [4], using logical equivalences after the truth values of the abducibles are defined.

For a given mode declaration, $s*$ denotes the atom created from the schema s by replacing the placemarkers with newly defined variables. $ins(s*)$ is shorthand for a list of all the variables that replace input placemarkers, $outs(s*)$ is a list of the variables that replace output placemarkers and $cons(s*)$ is a list of the variables that replace constant placemarkers in $s*$. All abducibles are marked by a \$ character.

Intuitively we want to transform the original theory into a "meta-theory" that can be used to reason about possible inductive solutions. The transformation is such that whenever certain abducibles are part of an abductive solution we must infer that a certain head is needed in a rule in inductive solution. Other abducibles are derived to define the body of such rules. Atoms of the type $\$head(h, id)$ are abduced whenever a rule (identified by id) is needed that uses the mode declaration h in the head. Atoms of the type $\$body(b, id, r, l, c)$ are abduced whenever a condition that uses the mode declaration b is required in the body of a rule that is part of a solution. In $\$body(b, id, r, l, c)$, r disambiguates amongst different uses of the same mode declaration in the body; l defines the bindings of the input variables; and c contains the constants used. For brevity we will not go into details of the type checking. Given a set of mode declarations L, we construct a set of rules $\mathcal{R}$ as follows:

– Let $b_1, ..., b_n$ be labels for all body mode declarations in L. For each head mode declaration $m : mode(h, r(m), s)$ the following clause is in $\mathcal{R}$:

$$
\begin{aligned}
s* \text{ :- } \quad & \$head(m, ID), \\
& not\ body(b_1, ID, 1, ins(s*), X_{1,1}), \\
& ... \\
& not\ body(b_1, ID, r(b_1), X_{1,(r(b_1)-1)}, X_{1,r(b_1)}), \\
& not\ body(b_2, ID, 1, X_{1,(r(b_1))}, X_{2,1}), \\
& ... \\
& not\ body(b_2, ID, r(b_2), X_{2,(r(b_2)-1)}, X_{2,r(b_2)}), \\
& ... \\
& not\ body(b(n), ID, r(b_n), X_{n,(r(b_n)-1)}, X_{n,r(b_n)}).
\end{aligned}
$$

These rules represent a skeleton for the rules that can be considered as inductive hypotheses. If the $\$head$ atom is not part of an abductive solution the rule is always true and has no effect on the rest of the theory. Otherwise the truth of the head depends on the other conditions. In each $body$ condition, except the first, the fourth argument is the same variable as the fifth argument of the previous $body$ condition (ordered left to right). This makes it possible to share variables between conditions

and bind input variables to output variables. Variables that substitute output placemarkers are this way shared amongst conditions. Note that an order is established *a priori* mainly for efficiency purposes and to remove redundancies. As shown in [4] this can be avoided with a different encoding.

– For each body mode declaration $m : mode(b, r(m), s)$, the following clause is in $\mathcal{R}$:

$$\begin{aligned} body(m, ID, R, V, X) :- \\ & link(V, ins(s*), E), \\ & append(V, outs(s*), X), \\ & \$body(m, ID, R, E, cons(s*)), \\ & not\ s*. \end{aligned}$$

Rules of this type only have effect on the transformed theory if the $\$body$ atom is abduced. The atom $link(v, i, e)$ produces a list i where all the elements are also in v and a list of e that contains the indexes of the elements i in v (e.g. $link((ann, mary, f, tom), (ann, f), (1, 3))$ is a true instance). The list in the third argument is used in the abducible $\$body$ to codify the binding of the variables. $append$ is a conventionally defined operator for appending lists.

The theory $\mathcal{R}$ is used within the learning process and processed together with a background theory and a set of constraints by the ASP solver. The final solution is transformed back into a set of rules.

Example 2. Let $\mathcal{T}$, $\mathcal{R}$ and L be the theories and mode declarations from the previous example. $\mathcal{R}$ is constructed from L as follows:

$$\begin{aligned} daughter(X, Y) :- & \$head(m1, ID), \\ & not\ body(m2, ID, 1, (X, Y), X_{m2,1}), \\ & not\ body(m3, ID, 1, X_{m2,1}, X_{m3,1}). \end{aligned}$$

$$\begin{aligned} body(m2, ID, 1, V, X) :- \\ & link(V, (Xl, Yl), L), \\ & \$body(m2, ID, 1, L, ()), \\ & not\ mother(Xl, Yl). \end{aligned}$$

$$\begin{aligned} body(m3, ID, 1, V, X) :- \\ & link(V, (Xl), L), \\ & \$body(m3, ID, 1, L, C), \\ & not\ sex(Xl, C). \end{aligned}$$

Using an ASP solver we can generate the following abductive solution that codifies rule $r3$ from the previous example $\Delta = \{\$head(m1, r3), \$body(m2, r3, 1, (2, 1), ()), \$body(m3, r3, 1, (1), (f))\}$. The first abducible specifies mode declaration $m1$ is used in the head of $r3$. The second specifies $m3$ is used in the body; the list $(2, 1)$ specifies that the first input variable (left to right in order of appearance in the mode declaration) is linked to the second variable in the head and that the second input variable is linked to the first variable in the head. The third abducible specifies $m3$ is used in the body, that the input variable is linked to the first variable in the head and that the constant is instantiated to f.

As made clear in Section 4.3, by defining a probability distribution over the abducibles we can instantiate a probability for the literals in the constraints. For example, let $\theta_{\$head(m1,r3)} = 1$, $\theta_{\$body(m2,r3,1,(2,1),())} = 0.8$, $\theta_{\$body(m3,r3,1,(1),(f))} = 1$ and $\theta_a = 0$ for all $a \in A \setminus \Delta$. In this case ${P_0}^{\theta}(\Delta) = 0.8$ and ${P_0}^{\theta}(\Delta \setminus \{\$body(m2, r3, 1, (2, 1), ())\}) = 0.2$. The two considered sets of abducibles together with $\mathcal{R}$ are respectively logically equivalent to $\{daughter(X, Y) : - mother(Y, X), sex(X, f)\}$ and $\{daughter(X, Y) : - sex(X, f)\}$ and implicitly define their probabilities.

3.2 Model Generation

Given a set of mode declarations L, the theory $\mathcal{R}$ generated from it, and a background theory $\mathcal{T}$, for each observation x we generate all the generalised answer sets that are needed to calculate $P^{\theta}(x)$. That is to say, we generate all generalised answer sets for the abductive theory $\langle \mathcal{T} \cup \{: -not\ x\} \cup \mathcal{R}, A\rangle$. A includes all the atoms with predicate $\$head$ and $\$body$ that correspond to meaningful literals in the final rule. The solutions $\Delta_1, ..., \Delta_q$ are used to construct the data structures that codify the sum of products in Equation (1), where each Δ_i corresponds to one element of the sum. Conceptually, the abductive solutions correspond to a boolean formula where all the abducibles within a solution are in conjunction (and negated if not part of the model) and all the solutions are in disjunction.

3.3 Parameter Estimation

The observations x_i are represented by literals and they are associated with a target probability y_i, i.e. the learned theory should entail each observation x_i with probability y_i. We assume that the true target value is altered by random Gaussian noise to account for errors in the background theory and noise in the observed literals as in [10]. Finding the maximum likelihood (abductive) hypothesis under this assumption corresponds to minimising the following mean squared error function [19]:

$$MSE(\theta) = \frac{1}{|X|} \sum_i (y_i - P^{\theta}(x_i | \langle \mathcal{T}, A \rangle))^2 \tag{2}$$

We use gradient descent[2] over equation (2) to estimate the probabilities of each abducible. In our particular case, positive and negative examples are modelled as (possibly negated) literals with target probability $y_i = 1$. Initially θ is given random values. Our implementation uses the algorithm described in [10], adopting binary decision diagrams (BDDs) to calculate the value of the gradient and of the probabilities of the observations at each iteration. The final θ_o that minimises the MSE can be mapped into a set of probabilistic hypotheses, thus providing rules that improve the original knowledge base by taking account of the probabilistic observations. This can be seen in more detail in Section 4.3.

[2] Other optimisation algorithms can be used.

4 Case Study: A Planning Agent

An ideal application for nonmonotonic rule learning is a planning problem [9] in which the initial knowledge base (from which plans are generated) is discovered to be incorrect or incomplete when the plans are generated and executed. Moreover, when the plans refer to a robotic agent situated in the physical world, with all its uncertainties, a probabilistic approach is required. Probabilities can also account for an incomplete model of the environment or of other agents acting in the same environment.

4.1 Knowledge Base

Figure 2 depicts the feedback loop between the knowledge base and the running system. The knowledge base (also known as a *domain model*) contains a description of the agent and the world it occupies, expressed as logic program in terms of conditions that hold and actions that can be performed (similar to Event Calculus [15]). Figure 3 shows part of the knowledge base for our specific example of a mobile robot, which states that a move action between two locations is possible if the places are connected and the robot is at the initial location.

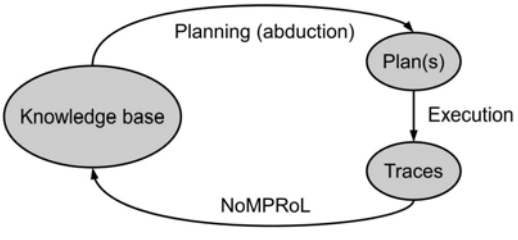


Fig. 2. Feedback between planning and rule learning

Typically there are many sequences of actions which the agent could perform to achieve a given goal. In a traditional linear plan, a single sequence is selected and the agent performs it blindly. *Reactive* or *universal* plans [28] improve robustness by having the agent sense the new world state after each action so that unexpected state changes—those not described by the knowledge base—can be handled. However, even for a given state there are often several actions which lead to the goal, and reactive planners make an almost arbitrary choice (often the shortest path). The practical truth is that each action has a different probability of leading to the goal, and it is on this basis that alternatives should be selected. The problem described is nonmonotonic as adding rules to the theory can invalidate previous consequences. The case study provided is a small instance of a class of problems that can involve complex interactions with the environment and rich contextual properties to be taken into account, e.g. position of other agents, speed, available power.

We generate plans given a goal condition such as $holdsAt(at(gps5), 3)$ by following an abductive procedure [29]. The resulting linear plans are merged to produce (a variant

```
:-  do(E1, T),
    do(E2, T),
    E1 != E2.

linked(gps1, gps2).
linked(gps1, gps3).
linked(gps1, gps4).
...

%move
possible(move(L1, L2), T) :- L1 != L2,
                             holdsAt(at(L1), T),
                             linked(L1, L2).
initiates(move(L1, L2), at(L2), T).
terminates(move(L1, L2), at(L1), T).
possible(move(L1, L2), T) :- L1 != L2,
                             holdsAt(at(L1), T),
                             linked(L1, L2).
...
happens(E, T) :- do(E, T),
                 succeeds(E,T).
```

Fig. 3. Knowledge base fragment

of) a reactive plan in which each state is associated with one or more actions, thus preserving the choice of paths present in the knowledge base. Each possible action for the current state has a probability (of achieving the goal) given by the sum over the probabilities of possible worlds (represented by generalised answer sets) that contain the action. The action with the highest probability is chosen at each iteration.

4.2 Trace Generation

Traces, from which the observations are derived, are generated by executing the reactive plan as follows:

1. the current environment state is sensed;
2. action(s) applicable to the current state are generated abductively for the given goal; and
3. one action is executed. Initially, when no probabilistic bias is available, where there is a choice of actions (multiple paths to the goal), one is selected randomly.

These steps are repeated until the goal condition is met, or a time-out occurs. At each iteration, the sensed state and the action chosen are recorded. The sequence of states and actions provides a trace. Figure 4 shows an example trace.

From each trace j we derive a set of observations $\{o_{1,j}, ..., o_{m,j}\}$ (one for each time the environment is sensed), and a set of *histories* $\{O_{1,j}, ..., O_{m,j}\}$. Each history $O_{i,j}$ includes all the states and actions in the execution up to the time point before the one

```
%trace 12
holdsAt(at(gps1),0).
do(move(gps1,gps2),0).
holdsAt(at(gps2),1).
do(move(gps2,gps5),1).
holdsAt(at(gps5),2).
do(pickup,2).
holdsAt(at(gps5),3).
```

Fig. 4. Execution trace

$o_{i,j}$ refers to. The training set X thus contains such pairs $(o_{i,j}, O_{i,j})$ and the target probability for each observation is set to 1. For example, the trace in Figure 4 gives a set of observations including:

$$o_{12,2} : \begin{cases} holdsAt(at(gps5), 2), \\ not\ holdsAt(holdingBall, 2). \end{cases}$$

$$O_{12,2} : \begin{cases} holdsAt(at(gps1), 0). \\ do(move(gps1, gps2), 0). \\ holdsAt(at(gps2), 1). \\ do(move(gps2, gps5), 1). \end{cases}$$

We are interested here in a revised theory that fits the observations for a given history. The probability associated with each observation thus involves the corresponding history. Consequently, the function we want to minimise is the following:

$$MSE(\theta) = \frac{1}{|X|} \sum_{i,j} (1 - P^{\theta}(o_{i,j} | \mathcal{T} \cup \Delta \cup O_{i,j}))^2 \tag{3}$$

4.3 Experiment

The practical experiment involves a mobile robot carrying a ball and a robotic arm which can remove the ball from the robot. The goal of the system is for the mobile robot to transport the ball to the arm whereupon the ball can be placed in a store next to the arm. This is expressed as $holdsAt(at(gps5), T)$, $holdsAt(holdingBall, T)$. The mobile robot is able to sense obstacles using short-range infra-red sensors and to detect when the ball is placed on top. In order to navigate through the environment, one or more cameras are placed on the ceiling which use simple computer vision techniques to determine the position and orientation of the mobile robot. The robotic arm is equipped with another camera to locate the ball, and pressure sensors to detect when it has successfully gripped an object.

As with any robotic system, there are many sources of noise. The infra-red sensors can detect obstacles incorrectly, causing erratic motion. Likewise, noise in the camera image can lead to incorrect localisation of the robot. Erratic motion may ultimately lead to complete failure of the plan, if for instance the robot collides with an obstacle or goes

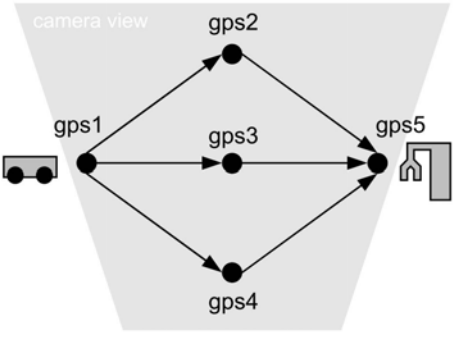


Fig. 5. The environment has three paths from the starting point to the arm

outside the area visible to the cameras. Noise in the camera and pressure sensors of the robotic arm can lead to the ball being dropped or not being identified at all.

To simplify navigation from the starting location to the target (the location of the arm), we create three waypoints, as shown in Figure 5, providing three alternative paths to the arm. The choice of a path leads to different rates of success, as a consequence of the sources of noise described above.

We executed the plan a total of 30 times, each starting in a random state and producing one trace. Each trace consists of the state observed and action taken at each time step, a flag indicating whether or not the goal was achieved and the time taken to execute the plan. Failure is detected explicitly (if the robot leaves the camera range) or otherwise through a time-out. Figure 4 shows one trace in which the robot started at location $gps1$, moved to $gps2$ followed by $gps5$. The absence of $holdsAt(holdingBall, 3)$ shows that the arm failed to pick up the ball, and this trace is recorded as a failure. Finally, we applied NoMPRoL, using the following mode declarations:

$$
\begin{aligned}
&mode(h, 2, succeeds(pickup, +time)).\\
&mode(h, 2, succeeds(move(+loc, +loc), +time)).\\
&mode(b, 2, +loc = \#loc).\\
&mode(b, 1, holdsAt(at(+loc, +time))).\\
&mode(b, 1, linked(+loc, +loc)).\\
&mode(b, 1, wasAt(+loc, +loc)).
\end{aligned}
$$

Results. The benefit of the approach will be shown by whether the robot improved its performance, measured as time taken to reach the goal in the real world and as the ratio of failures over many runs. We run NoMPRoL leaving one trace out for testing each time and using the remaining ones for the training. We then selected only those traces which contain the same actions as those which the agent would perform after learning, thus omitting those where learning has caused the behaviour to change. The average of those execution times and the number of time-outs for the initial and the trained agent are reported in Table 1.

Table 1. Performance measure of the initial and trained agents

	Average execution time (s)	Failure rate
Initial agent	85.2	0.3
Trained agent	63.6	0.1

We consider one representative training instance and discuss the process and results. After the GD algorithm converges each abducible in A is associated with a probability. For example the abducible $body(b\_not\_was\_at, (dp, 1), 1, (1), (gps2))$ has probability $\theta_{body(b\_not\_was\_at,(dp,1),1,(1),(gps2))} = 0.7315$. Intuitively this probability encodes the relevance of the condition within the rule labelled as $(dp, 1)$. Processing and approximating the probabilities for ease of exposition, we obtain the following rules:

$$
\begin{array}{lll}
r1 : succeeds(pickup, T) \text{ :-} & & \\
\qquad not\ wasAt(gps2, T). & & 0.7 \\
\\
r2 : succeeds(move(L1, L2), T) \text{ :-} & & \\
\qquad linked(L1, L2). & & 0.5 \\
\\
r3 : succeeds(move(L1, L2), T) \text{ :-} & & \\
\qquad not\ wasAt(gps3, T), & & 0.2 \\
\qquad L2 = gps3, & & 0.4 \\
\qquad holdsAt(L1, T), & & 1 \\
\qquad linked(L1, L2). & & 1
\end{array}
$$

The probability associated with a condition intuitively represents the probability that the condition is part of a sampled theory, resulting in a distribution over possible subtheories[3]. In general the final output will be chosen based on the interpreter or based on readability and accuracy requirements.

In our experiment the MSE for all the executions is lower than 1. Non-probabilistic logic rules would never result in a lower MSE, since at least one of the observations would not be entailed. Also, for systems where the estimation is performed on logic facts [11] the dependency between the final pickup and the robot's moving through location $gps2$ would not be detected.

[3] In fact, from these rules we can derive an equivalent distribution over theories, similarly to [5], that are obtained by dropping some of the conditions. For example the following theory has a probability $p = 0.168$ (causing every atom that is entailed by this theory to have a probability greater than p):

$$
\begin{array}{l}
succeeds(pickup, T) \text{ :- } not\ wasAt(gps2, T). \\
succeeds(move(L1, L2), T) \text{ :- } linked(L1, L2). \\
succeeds(move(L1, L2), T) \text{ :- } holdsAt(L1, T), linked(L1, L2).
\end{array}
$$

Intuitively, the probability is derived from the product of the probabilities associated with the conditions that are kept and the complement of the probabilities of the conditions that are discarded.

5 Discussion and Related Work

Much of the recent work in the field of Statistical Relational Learning [8] shares similar objectives to those addressed here. The work we present is, to the best of our knowledge, the first with a methodology for learning probabilistic rules in nonmonotonic domains. Under the assumption of monotonicity, the problem of learning probabilistic rules has been addressed separately as an estimation problem and as structure learning [22]. [21] concentrates on single-clause hypotheses; [10] applies gradient descent to estimate probabilities of logic facts in a logic program; [7] extends an established ILP search algorithm with a probability-based search heuristic. [31] addresses the problem of learning probabilistic planning rules.

Markov Logic Networks (MLN) [25] extend first-order logic with Markov networks and are successfully supported by a number of different learning techniques. Amongst these, [30] employs a gradient-driven estimation of the parameters and [14] proposes an integrated solution for structure learning. The main disadvantage of MLN compared to the type of representation used in this paper and, in general, to probabilistic logic representations based on distribution semantics is that formula weights in MLN are counterintuitive and their effect on the inference is not obvious.

Compared to solutions based on Markov Decision Processes (MDP) (e.g. [2]) the approach proposed employs a full first-order representation. An MDP could be used as a target representation for the learned output, with the caveat that maintainability is lost in a more constrained knowledge representation language. Furthermore in the case of planning the states and the effects of the actions can be modelled to take into account, as shown in the example, of relevant past events and states, thus defining the probability of an action at a time point t also as a function of states at $t' \leq t - 1$.

The methodology proposed is not a full solution to the problem of learning probabilistic nonmonotonic logic programs. Application in real scenarios would require a considerable computational effort. Nevertheless we provide a general mechanism to transform an inference problem into an equivalent learning task, thus enabling the use of established techniques, and a framework for further developments. ASP tools provide the most effective way to generate and check models for nonmonotonic theories and techniques for parameter estimation benefit from years of consolidation. To improve efficiency, we are currently working on an extension of NoMPRoL that makes use of stochastic sampling techniques as in [13].

We designed an integrated mechanism to solve the problem of generating inductive hypotheses that is based on ASP. Despite the interest in nonmonotonic ILP [26], there is a lack of available tools. [24], a nonmonotonic ILP system, does not provide the required support for the methodology presented in this paper. In particular, the system is not designed to tolerate noise in training examples and outputs only maximally compressive solutions.

6 Conclusions

We have presented an approach for rule learning in probabilistic nonmonotonic domains. The approach estimates probabilities for rules which, together with a knowledge

base, minimise the error between a target probability and the entailed probability over a set of observed atoms. Such rules can be used to improve the original knowledge base, reducing the gap between the current knowledge and the observations, as we have shown in a planning scenario where executing the learned rules reduces the overall rate of the robotic agent failing to reach its goal.

NoMPRoL has a potentially broad applicability, ranging from situations involving interactions with the external environment, such as is the case in self-adaptive systems, to cases where a rich logical representation is integrated with uncertainty (e.g. in software engineering [1], in which nonmonotonic ILP is used to extend a partial system specification with learned requirements). Our approach would enable such systems to revise their knowledge bases with probabilistic information that is difficult or impossible for the designer to provide. Currently the main limit of the approach is scalability. As the space of candidate hypotheses and the domain entities and relations grows, the answer set solver has to deal with exponentially large ground theories. We plan further experiments aimed at improving the scalability of the methodology and extensions towards an incremental online learning setting.

References

1. Alrajeh, D., Ray, O., Russo, A., Uchitel, S.: Extracting Requirements from Scenarios with ILP. In: Muggleton, S.H., Otero, R., Tamaddoni-Nezhad, A. (eds.) ILP 2006. LNCS (LNAI), vol. 4455, pp. 64–78. Springer, Heidelberg (2007)
2. Boutilier, C., Brafman, R.I., Geib, C.: Prioritized goal decomposition of markov decision processes: Toward a synthesis of classical and decision theoretic planning. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, pp. 1156–1162 (1997)
3. Clark, K.L.: Negation as failure. Logic and Data Bases, 293–322 (1977)
4. Corapi, D., Russo, A., Lupu, E.: Inductive logic programming as abductive search. In: Tec. Comm. of the 26th ICLP, LIPIcs, vol. 7, pp. 54–63, Dagstuhl (2010)
5. Dantsin, E.: Probabilistic logic programs and their semantics. In: Voronkov, A. (ed.) RCLP 1990 and RCLP 1991. LNCS, vol. 592, pp. 152–164. Springer, Heidelberg (1992)
6. De Raedt, L., Thomas, G., Getoor, L., Kersting, K., Muggleton, S. (eds.): Probabilistic, Logical and Relational Learning. Schloss Dagstuhl (2008)
7. De Raedt, L., Thon, I.: Probabilistic Rule Learning. In: Frasconi, P., Lisi, F.A. (eds.) ILP 2010. LNCS (LNAI), vol. 6489, pp. 47–58. Springer, Heidelberg (2011)
8. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press, Cambridge (2007)
9. Giunchiglia, F., Traverso, P.: Planning as model checking. In: Biundo, S., Fox, M. (eds.) ECP 1999. LNCS, vol. 1809, pp. 1–20. Springer, Heidelberg (2000)
10. Gutmann, B., Kimmig, A., Kersting, K., De Raedt, L.: Parameter learning in probabilistic databases: A least squares approach. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 473–488. Springer, Heidelberg (2008)
11. Inoue, K., Sato, T., Ishihata, M., Kameya, Y., Nabeshima, H.: Evaluating abductive hypotheses using an em algorithm on bdds. In: Boutilier, C. (ed.) IJCAI, pp. 810–815 (2009)
12. Inoue, K., Sakama, C.: Transforming abductive logic programs to disjunctive programs. In: Proc. 10th ICLP, pp. 335–353. MIT Press, Cambridge (1993)

13. Kimmig, A., Santos Costa, V., Rocha, R., Demoen, B., De Raedt, L.: On the efficient execution of probLog programs. In: Garcia de la Banda, M., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 175–189. Springer, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-89982-2_22
14. Kok, S., Domingos, P.: Learning the structure of markov logic networks. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 441–448. ACM Press, New York (2005)
15. Kowalski, R., Sergot, M.: A logic-based calculus of events. New Gen. Comput. 4(1), 67–95 (1986)
16. Lavrac, N., Dzeroski, S.: Inductive logic programming - techniques and applications. Ellis Horwood series in artificial intelligence. Ellis Horwood (1994)
17. Lifschitz, V.: Answer set programming and plan generation. Artificial Intelligence 138(1-2) (2002); Knowledge Representation and Logic Programming
18. Lloyd, J.W.: Foundations of Logic Programming, 2nd edn. Springer, Heidelberg (1987)
19. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
20. Muggleton, S.: Inverse entailment and progol. New Gen. Comp. 13(3&4), 245–286 (1995)
21. Muggleton, S.: Learning structure and parameters of stochastic logic programs. In: Matwin, S., Sammut, C. (eds.) ILP 2002. LNCS (LNAI), vol. 2583, pp. 198–206. Springer, Heidelberg (2003)
22. Muggleton, S.: Learning stochastic logic programs. Electron. Trans. Artif. Intell. 4(B), 141–153 (2000)
23. Poole, D.: Abducing through negation as failure: stable models within the independent choice logic. The Journal of Logic Programming 44(1-3), 5–35 (2000)
24. Ray, O.: Nonmonotonic abductive inductive learning. Journal of Applied Logic 7(3), 329–340 (2009), http://www.cs.bris.ac.uk/Publications/Papers/2001069.pdf
25. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006)
26. Sakama, C.: Nonmonotonic inductive logic programming. In: Eiter, T., Faber, W., Truszczyński, M. (eds.) LPNMR 2001. LNCS (LNAI), vol. 2173, p. 62. Springer, Heidelberg (2001)
27. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: ICLP, pp. 715–729 (1995)
28. Schoppers, M.: Universal plans for reactive robots in unpredictable environments. In: IJCAI, vol. 87, pp. 1039–1046 (1987)
29. Shanahan, M.: An abductive event calculus planner. The Journal of Logic Programming 44(1-3), 207–240 (2000)
30. Singla, P., Domingos, P.: Discriminative training of markov logic networks. In: Veloso, M.M., Kambhampati, S. (eds.) AAAI. pp. 868–873. AAAI Press / The MIT Press (2005)
31. Zettlemoyer, L.S., Pasula, H., Kaelbling, L.P.: Learning planning rules in noisy stochastic worlds. In: AAAI, pp. 911–918 (2005)

A Formal Semantics for Brahms*

Richard Stocker[1,**], Maarten Sierhuis[2,3], Louise Dennis[1], Clare Dixon[1], and Michael Fisher[1]

[1] Department of Computer Science, University of Liverpool, UK
[2] PARC, Palo Alto, USA
[3] Man-Machine Interaction, Delft University of Technology, Delft, NL
R.S.Stocker@liverpool.ac.uk

Abstract. The formal analysis of computational processes is by now a well-established field. However, in practical scenarios, the problem of how we can formally verify interactions with humans still remains. In this paper we are concerned with addressing this problem. Our overall goal is to provide formal verification techniques for human-agent teamwork, particularly astronaut-robot teamwork on future space missions and human-robot interactions in health-care scenarios. However, in order to carry out our formal verification, we must first have some formal basis for this activity. In this paper we provide this by detailing a formal operational semantics for Brahms, a modelling/simulation framework for human-agent teamwork that has been developed and extensively used within NASA. This provides a first, but important, step towards our overall goal by establishing a formal basis for describing human-agent teamwork, which can then lead on to verification techniques.

1 Introduction

Computational devices often need to interact with humans. These devices can range from mobile phones or domestic appliances, all the way to fully autonomous robots. In many cases all that the users care about is that the device works well most of the time. However, in mission critical scenarios we clearly require a more formal, and consequently much deeper, analysis. Specifically, as various space agencies plan missions to the Moon and Mars which involve robots and astronauts collaborating, then we surely need some form of *formal verification* for astronaut-robot teamwork. This is needed at least for astronaut safety (e.g. "the astronaut will never be out of contact with the base") but also for mission targets (e.g. "three robots and two astronauts can together build the shelter within one hour"). But: how are we to go about this? How can we possibly verify human behaviour? And how can we analyze teamwork?

In [2] a formal approach to the problem of human-agent (and therefore astronaut-robot) analysis was proposed, suggesting the model-checking of *Brahms* models [9, 17, 13]. Brahms is a simulation/modelling language in which complex human-agent work patterns can be described. Importantly for our purposes, Brahms is based on the concept of *rational agents* and the system continues to be successfully used within NASA for the sophisticated modelling of astronaut-robot planetary exploration teams [3, 12, 10].

* Work partially funded in the UK through EPSRC grants EP/F033567 and EP/F037201.
** Corresponding author.

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 259–274, 2011.

Thus, it seems natural to want to formally verify Brahms models [2] but, until now, the Brahms language had no *formal* semantics (unless you count the implementation code as this). So this paper describes the first formal operational semantics [15] for the Brahms language; in current work we are using the formal semantics to develop and apply model checking to Brahms.

2 Brahms

Brahms is a multi-agent modelling, simulation and development environment devised by Sierhuis [9] and subsequently developed at NASA Ames Research Center. Brahms is a modelling language designed to model human activity using *rational agents*.

An agent [18] essentially captures the idea of an autonomous entity, being able to make its own choices and carry out its own actions. Beyond simple autonomy, *rational agents* are increasingly used as a high-level abstraction/metaphor for building complex/autonomous space systems [5]. Rational agents can be seen as agents that make their decisions in a *rational* and *explainable* way (rather than, for example, purely randomly). The central aspect of the rational agent metaphor is that such agents are autonomous, but can react to changes in circumstance and can choose what to do based on their own agenda. In assessing such systems it may not be sufficient to consider *what* the agent will do, but we must often also consider *why* it chooses to do it. The predominant view of rational agents is that provided by the BDI (beliefs-desires-intentions) model [8, 7] in which we describe the goals the agent has and the choices it makes. Thus, in modelling a system in terms of rational agents, we typically describe each agent's *beliefs* and *goals* (*desires*), which in turn determine the agent's *intentions*.

Brahms follows a similar rational agent approach but, because it was developed in order to represent people's activities in real-world contexts, it also allows the representation of artifacts, data, and concepts in the form of classes and objects. Both agents and objects can be located in a model of the world (the geography model) giving agents the ability to detect objects and other agents in the world and have beliefs about the objects. Agents can move from one location in the world to another by executing a *move* activity, simulating the movement of people. For a more detailed description of the Brahms language we refer the reader to [9] and [10]. The *key aspects* of Brahms are:

- *activities*: actions an agent can perform, which typically consume simulation time;
- *facts*: state of the environment (which every agent/object can observe through the use of "detectables");
- *beliefs*: each agent's own personal perceptions;
- *detectables*: bring facts into the an agent's belief base and determine how the agent will react in response;
- *workframes*: sequences of events required to complete a task, together with any belief updates resulting from the task completion;
- *thoughtframes*: reasoning/thought processes, e.g. "I believe it is raining therefore I believe I need an umbrella";
- *time*: central to Brahms as the output is represented in the form of a time-line displaying every belief change and event that occurs.

In summary, the Brahms language was originally devised to model the contextual situated activity behaviour of groups of people. It has now evolved into a language for modelling both people *and* robots/agents. As such it is ideal for describing human-agent/robot teamwork.

2.1 Brahms Example

Orbital Communications Adaptor (OCA) officer flight controllers in NASA's International Space Station Mission Control Center use different computer systems to uplink, downlink, mirror, archive, and deliver files to and from the International Space Station (ISS) in real time. The OCA Mirroring System (OCAMS) is a multi-agent software system operational in NASA's Mission Control Center [14], replacing the OCA officer flight controller with an agent system that is based on the behavior of the human operator. NASA researchers developed a detailed human-behavioral agent model of the OCA officers' work practice behaviour in Brahms. The agent model was based on work practice observations of the OCA officers and the observed decision-making involved with the current way of doing the work. In the system design and implementation phases, this model of the human work practice behaviour was made part of the OCAMS multi-agent system, enabling the system to behave and make decisions as if it were an OCA officer.

Here is a short scenario of how the OCAMS system is used in mission control: The On-board Data File and Procedures Officer (ODF) sends a request to the OCAMS (personal) agent via their email system. The OCAMS agent parses the request and understands that the ODF has dropped a zip file to be uplinked to the ISS on the common server. The OCAMS agent needs to identify the type of file that is being delivered and decide, based on this, what uplink procedure needs to be executed. Having done so, the OCAMS agent chooses the procedure and starts executing it, as if it were an OCA officer. The OCAMS agent first transfers the file and performs a virus scan, and then continues to uplink the file to the correct folder on-board the ISS. The OCAMS agent applies the same procedure that an OCA officer would do.

The OCAMS system has been extended over three years [4]. With the latest release the OCAMS system will have completely taken over all routine tasks from the OCA officer, about 80% of the workload. Other flight controllers in mission control will interact with the OCAMS agent as if it were an OCA officer. With every new release (indeed with every increase in functionality) the system developers are required to perform complete testing of the system. Increases in functionality mean that there is now not enough time to test every possible case. The ability to carry out formal verification and validation of this human-agent system would enable more comprehensive analysis.

3 Overview of Semantics

Rather than presenting the full semantics in detail (see [16]), we consider the core elements of the semantics here and then work through an example Brahms scenario in Section 4.

3.1 Time Keeping and Scheduling

An important aspect of Brahms is a shared system clock; this creates the simulation time line and is used as a global arbiter of when activities start and end. Agents are not explicitly aware of the system clock even though it controls the duration of activities and can be referred to in the selection of workframes and thoughtframes. As a result, many applications also involve a clock "object" that agents are explicitly aware of.

The Brahms execution model involves updating the system clock, then examining each agent in turn to see what internal state changes take place at that time step, and then updating the system clock once more. The system clock does not update by a fixed amount in each cycle but makes a judgment about the next time something "interesting" is going to happen in the system and jumps forward to that point.

3.2 Running Workframes and Thoughtframes

Workframes and *thoughtframes* represent the plans and thought processes in Brahms. A workframe contains a sequence of activities and belief/fact updates which the agent/object will perform and a guard which determines whether the workframe is applicable or not. A thoughtframe is a restricted version of this which only contains sequences of belief/fact updates. Workframes may take time to complete (depending on the activities involved) while thoughtframes are assumed to run instantaneously. A workframe can also detect changes in its environment (facts), bring these changes into the agent's belief base and then decide whether or not to continue executing in the light of the changes. Essentially, workframes represent the work processes involved in completing a task and thoughtframes represent the reasoning process upon the current beliefs, e.g. "I perform a workframe to go the shops; on leaving the house I detect it is raining so I suspend my workframe and update my belief that it is raining, which then triggers a thoughtframe stating that, since it is raining and I want to go the shops, then I need a raincoat".

3.3 Priority and Suspension of Workframes and Thoughtframes

Workframes and thoughtframes have a priority (which can be assigned by the programmer or derived from their list of activities). Thoughtframe and workframe priorities are independent, but an agent will execute all thoughtframes first in any given time step before moving on to examine workframes. At any point in time, the workframe that an agent is currently working on can be suspended if another, higher priority, workframe becomes available. Thoughtframes are *never* suspended because they have no duration and so always complete before any higher priority thoughtframe becomes available. When several workframes have the same priority, Brahms considers the currently executing workframe to have the highest priority, any other suspended workframes have second highest, impassed[1] workframes will have third highest, and then any other workframe will follow. Priorities in Brahms are integers but to model this priority order in our semantics we assign suspended workframes an increased priority of "0.2" and

[1] Workframes suspended because of changes in detectable facts.

impassed workframes "0.1". Workframes and thoughtframes of equal status and with joint highest priority will be selected at random. When a workframe is suspended, everything is stored, even the duration into its current activity, so the agent can resume exactly from where it left off.

3.4 Executing Plans: Activities and Communication

When an agent is assigned a workframe/thoughtframe all the instructions contained within it are stored on one of two stacks; one for the current thoughtframe and one for the current workframe. When an agent executes an instruction it is 'popped' off the top of the stack. *Primitive* activities and *move* activities all have time associated with them — when they are at the top of a stack the duration of the task is decreased by an appropriate amount each time the system clock updates. If the duration remaining reaches zero then the activity is finished and popped off the stack. When the activity is a *move* activity the belief base of the agent, together with the global facts, are changed to reflect the new position of the agent/object; this update occurs when the time of the activity reaches zero.

Communications are also activities and may have a duration. When the communication ends, a belief update is performed on the target agent's (the receiver's) belief set.

3.5 Detectables

Detectables are contained within workframes and can only be executed if the workframe is currently active. Detected facts are imported into the agent's belief base and then either: *aborts* - deletes all elements from the workframe's stack; *impasses* - suspends the current workframe, *continues* - carries on regardless; or *completes* - deletes only activities from the workframe's stack but allows it to make all (instantaneous) belief updates.

3.6 Variables

Variables provide a method of quantification within Brahms. If there are multiple objects or agents which can match the specifications in a work- or thoughtframe's guard condition then the variable can either perform: *forone* — just select one; *foreach* — work on all, one after another; or *collectall* — work on all simultaneously. This is handled by recursive sets of workframes, e.g.

"Set of Workframes" = {W1,W2,W3:{W3.1,W3.2},W4}

where W3 is a workframe with variables and W3.1 and W3.2 are instantiations of W3 but with objects/agents in the place of the variables. When a workframe with variables is empty e.g. W3:{} Brahms will invoke a selection function to make instantiations based on the conditions.

3.7 Brahms Syntax

Brahms has a complex syntax for creating systems, agents and objects, although there is no space here to cover the full specification[2]. As an example Fig. **??** shows the definition of an agent's workframe showing where variables, detectables and the main body of the workframe are placed. Guards are specified by `precondition-decl`.

```
workframe ::=   workframe workframe-name
                {
                { display : ID.literal-string ; }
                { type : factframe | dataframe ; }
                { repeat : ID.truth-value ; }
                { priority : ID.unsigned ; }
                { variable-decl }
                { detectable-decl }
                { [ precondition-decl workframe-body-decl ] |
                   workframe-body-decl }
                }
```

3.8 Semantics: Notation

In the rest of this paper we use the following conventions to refer to components of the system, and agent and object states.

Agents: ag represents one *agent*, while Ag represents the *set* of all agents.
Beliefs: b represents one *belief*, while B represents a *set* of beliefs. In Brahms the overall system may have beliefs which are represented by B_ξ.
Facts: f represents one *fact*, while F represents a *set* of facts.
Workframes: β represents the *current workframe* being executed, WF represents a *set* of workframes, while *W* is any arbitrary workframe.
Thoughtframes: $\mathcal{T}$ represents any arbitrary *thoughtframe*, while TF represents a *set* of thoughtframes.
Activities: $\texttt{Prim\_Act}^t$ is a primitive activity of duration t.
Environment: ξ represents the *environment*
Time: T represents the time in general, while a specific duration for an activity is represented by t. The time maintained by the system clock is T_ξ.
Stage: The semantics are organised into "stages". Stages refer to the names of the operational semantic rules that may be applicable at that time, wild cards ($*$) are used to refer to multiple rules with identical prefixes. There is also a "*fin*" stage which indicates an agent/object is ready for the next cycle, and an "$idle$" stage which means it currently has no applicable thoughtframes or workframes.

Since the data structures for workframes are fairly complex we will treat these as a tuple, $\langle W_d, W_{ins} \rangle$ where W_d is *workframe header data* and W_{ins} is the *workframe instruction stack*. The *workframe header data* includes

[2] For the full syntax (with an informal semantics) see [11].

- W^r is the workframe's repeat variable.
- W^{pri} is the workframe's priority.
- W^V is the variable declaration.
- W^D is the workframe's detectables
- W^g is the workframe's guard.

Here we are considering any possible workframe (W), for the current workframe we would use β (or the name of the workframe). Thoughtframes are structured in a similar way.

3.9 Semantics: Structure

The system configuration is a 5-tuple: Ags - set of all agents; ag_i - current agent under consideration; B_ξ - belief base of the system, used to synchronise the agents (not used in simulations) e.g. agent i's next event finishes in 1000 seconds; F - set of facts in the environment, e.g. temperature is 20 degrees celsius; and T_ξ - current time of the system:

System's tuple = $\langle Ags, ag_i, B_\xi, F, T_\xi \rangle$

The agents and objects within a system have a 9-tuple representation: ag_i - the identification of the agent; $\mathcal{T}$ - the current thoughtframe; W - current workframe; $stage$ - stage the agent is at; B - set of beliefs the agent has; F - set of facts of the world; T - time of the agent; TF- set of thoughtframes the agent has; and WF - agent's set of workframes. The stage explains which set of rules the agent is currently considering or if the agent is in a finish (*fin*) or idle ($idle$) stage.

Agents tuple = $\langle ag_i, \mathcal{T}, W, stage, B, F, T, TF, WF \rangle$

The semantics are represented as a set of transition rules of the form

$$\langle StartingTuple \rangle \xrightarrow[ConditionsRequiredForActions]{ActionsPerformed} \langle ResultingTuple \rangle$$

Here, '$ConditionsRequiredForActions$' refers to any conditions which must hold before the rule can be applied. '$ActionsPerformed$' is used to represent change to the agent, object or system state which, for presentational reasons, can not be easily represented in the tuple.

Finally, it is assumed that all agents and objects can see and access everything in the environment's tuple, e.g. T_ξ.

4 Running Example of Brahms Semantics

As explained above, rather than giving all the semantics in detail (see [16] for the full semantics), we here work through a small Brahms scenario. Though simple, this scenario involves many of the aspects available within Brahms and so utilises a wide variety of semantic rules.

This example scenario is based on that provided in the Brahms tutorial which can be downloaded from `http://www.agentisolutions.com/download`. The scenario models two students: *Alex*, who will sit and study in the library until he becomes

hungry; and *Bob* who sits idly until the other student suggests going for food. When Alex becomes hungry he will decide to message Bob and venture out for food. Once Alex arrives at the restaurant he will wait for Bob to arrive and then he will eat, pay for the food and return to the library to study. The scenario also contains an explicit hourly clock object (separate from the system clock) which announces how many hours have passed, providing Alex with his beliefs about the current time.

Initialisation and parsing of the program code assigns the agents all their initial beliefs, determines the initial world facts and the geography of the area (distances between each location etc.). The agent Bob will be ignored in the discussion until his role becomes active. The initial beliefs of our student (Alex) are:

$$\left\{ \begin{array}{l} hungry = 15, \quad Loc = Library, \quad Bob.Loc = Home, \\ perceivedtime = 0, \quad Clock.time = 0, \quad desiredRestaurant = \emptyset \end{array} \right\} \tag{1}$$

Alex starts in the "*fin*" (finish) stage. Starting in the finished stage appears counterintuitive, however this "*fin*" indicates the agents have finished their previous events and are ready for the next cycle. On system initiaition we assume the agents previous events have been completed, even though they were empty.

Below is example Brahms code showing one of Alex's thoughtframes and one of his workframes. The thoughtframe represents Alex's thought process for increasing his hunger as time progresses (Campanile_Clock refers to the clock object) and Alex becomes hungrier when he sees that the clock object's time is later than he believes it is. He then updates his internal belief about the time and his hunger. The workframe tells Alex to perform the study activity when both the time and his level of hunger are less than 20.

```
agent Alex
{
  ...
  thoughtframes:
    thoughframe tf_FeelHungry
    {
       when(knownval(Campanile_Clock.time >
            current.perceivedtime)
        do
        {
            conclude((current.perceivedtime =
                      Campanile_Clock.time), bc: 100);
            conclude((current.hungry =
                      current.hungry + 3), bc: 100);
        }
    }

  workframes:
    workframe wf_Study
    {
        repeat: true;
        priority: 1;
```

```
        when(knownval(Campanile_Clock.time < 20)
             and current.hungry < 20)
        do
        {
            Study();
        }
    }
}
```

Here, "`bc: 100`" describes a percentage probability value associated with the belief. For simplicity we omit further discussion of this in the remainder of the paper.

In representations of thoughtframes and workframes in the semantics, the thoughtframe tf_FeelHungry will appear as

$$\langle\ FeelHungry_d,\ [conclude(perceivedtime = Clock.time);\ conclude(hungry = hungry + 3)]\rangle$$

where the thoughtframe data $FeelHungry_d$ contains the guard, $FeelHungry^g$ with the value $Clock.time > perceivedtime$. Similarly, the `wf_Study` workframe above will later appear as $\langle Study_d, [\texttt{Prim\_Act}^{3000}]\rangle$. Note that `Study()` is a primitive activity with duration 3000 seconds and we translate it directly into the primitive activity representation in the workframe's instruction stack. $Study_d$ includes the repeat variable $Study^r = true$, the priority $Study^{pri} = 1$ and the guard $Study^g = (Clock.time < 20) \wedge (hungry < 20)$

The **thoughtframes** Alex uses are:

$tf\_FeelHungry$ - increases Alex's hunger;
$tf\_ChooseBlakes$ - tells Alex to choose Blakes restaurant;
$tf\_ChooseRaleighs$ - tells Alex to choose Raleighs restaurant.

The **workframes** Alex uses are:

wf_Study - tells Alex to study;
$wf\_MoveToRestaurant$ - tells Alex to move to his desired restaurant;
wf_Wait - tells Alex to do nothing if he is in the restaurant and Bob isn't present;
wf_Eat - tells Alex to order and eat his food.

The **workframes** the Clock uses are:

$wf\_AsTimeGoesBy$ - increases the clock's time by 1 hour.

4.1 System Initiation

Initially the tuples for the Alex agent and the Clock object are

Alex: $\langle ag_{Alex}, \emptyset, \emptyset, fin, B_{Alex}, F, 0, TF_{Alex}, WF_{Alex}\rangle$
Clock: $\langle ob_{Clock}, \emptyset, \emptyset, fin, B_{Clock}, F, 0, \emptyset, WF_{Clock}\rangle$

where B_{Alex} is as in (1) and

$$\begin{aligned} TF_{Alex} &= \{tf\_FeelHungry, tf\_ChooseBlakes, tf\_ChooseRaleighs\} \\ WF_{Alex} &= \{wf\_Study, wf\_MoveToRestaurant, wf\_Wait, wf\_Eat\} \\ WF_{Clock} &= \{wf\_AsTimeGoesBy\} \\ B_{Clock} &= \{time = 0\} \\ F &= \{Alex.Loc = library, Bob.Loc = Home, Clock.time = 0\} \end{aligned}$$

4.2 Scheduler Rules

All the semantic rules used by the scheduler have the prefix 'Sch_'. The scheduler acts as a mediator between agents keeping them synchronized. It tells all the agents when the system has started, finished and when to move to the next part of the system cycle.

The scheduler is initiated first in any run of the system. It checks if all agents' current stage is either "active" or "finished". Since, in our example, it is the beginning of the system and the default setting for the agents' stage is finished then the system updates the stage of each agent to Set_Act. This will cause all the agents to start processing. This system action is expressed with the Sch_run rule.

RULE: Sch_run

$$\langle Ags, ag_i, B_\xi, F, T_\xi \rangle \xrightarrow[\forall ag_i \in Ags | (a_i^{stage} = fin \vee a_i^{stage} = idle), \neg(T_\xi = -1)]{\forall ag_i (ag_i^{'stage} = Set\_Act)} \langle Ags, ag_i', B_\xi, F, T_\xi \rangle$$

So, after this rule is executed Alex's state becomes:

$$\langle ag_{Alex}, \emptyset, \emptyset, Set\_Act, B_{Alex}, F, 0, TF_{Alex}, WF_{Alex} \rangle$$

While there is an agent in an active (not idle or finished) stage the scheduler waits. If all the agents are idle the system terminates:

RULE: Sch_Term

$$\langle Ags, ag_i, B_\xi, F, T_\xi \rangle \xrightarrow[\forall ag_i \in Ags | a_i^{stage} = idle]{} \langle Ags, ag_i, B_\xi, F, -1 \rangle$$

Agents and objects have their own internal clock but the scheduler manages the global clock which all agents/objects synchronize with. When agents or objects perform an activity they inform the scheduler of the time the activity will conclude. Once all agents are either idle or engaged in an activity (a set of stages marked Pop_CA*, Pop_MA* and Pop_CA* where "$*$" is a wild card) the scheduler then finds the smallest of all these times and updates the global clock to this time. This is achieved by the 'Sch_rcvd' rule:

RULE: Sch_rcvd

$$\langle Ags, ag_i, B_\xi, F, T_\xi \rangle \xrightarrow[\forall ag_i \in Ags | (ag_i^{stage} = Pop\_(PA*/MA*/CA*) \vee ag_i^{stage} = idle), \neg(T_\xi = -1)]{T_\xi' = MinTime(B_\xi)} \langle Ags, ag_i, B_\xi, F, T_\xi' \rangle$$

Where $MinTime()$ takes a belief base of the agents times and finds the minimum time with in it.

4.3 Agents and Objects Are Now Invoked

The agents and objects are invoked in order and each processes one rule in turn. In the Set_Act stage the agents run the Set_Act rule which, in this case, moves them all on to examining their thoughtframes for applicability. This is the 'Tf_*' stage which indicates that they will be looking at all the rules beginning with 'Tf_'. These are rules for processing thoughtframes. In our simple example there are currently no thoughtframes available for any objects or agents, so 'Tf_Exit' is selected, which moves Alex and the Clock on to checking for detectables.

In our example, no object or agent has a current workframe which checks detectables and so 'Det_Empty' is invoked which passes them on to checking workframes. This is the basic cycle of Brahms processing: thoughtframes, followed by detectables, followed by workframes.

Running workframes. $wf\_AsTimeGoesBy$ is the only workframe the Clock can process. This contains an activity of 3600 seconds duration and then the Clock increases its time by 1 hour. The guard on this workframe is that the current value of the Clock's time attribute is less than 20, which is currently true so the workframe is put forward for selection. Since there are no other workframes, the Clock selects this workframe as current and stores the list of instructions contained within the workframe in a stack.

Meanwhile Alex also selects a workframe for execution using Wf_Select.

RULE: Wf_Select

$$\langle ag_i, \emptyset, \emptyset, Wf\_*, B_i, F, T_i, TF_i, WF_i \rangle \xrightarrow[\exists W \in WF_i \mid B_i \models W^g]{\beta = MaxPri(W \in WF_i \mid B_i \models W^g)} \langle ag_i, \emptyset, \beta, Wf\_(true/false/once), B_i, F, T_i, TF_i, WF_i \rangle$$

This selects Alex's workframe (wf_Study) and restricts his rule choice to 'Wf_true', 'Wf_false' or 'Wf_once', depending on the workframe's repeat variable. The body of the workframe contains a single primitive activity, $Study$.

Alex's state is now

$$\langle\ ag_{Alex},\ \emptyset,\ \langle Study_d,\ [\texttt{Prim\_Act}^{3000}]\rangle,\ Wf\_(true/false/once),\ B_{Alex},\ F,\ 0,\ TF_{Alex},\ WF_{Alex}\rangle$$

The next semantic rule selected depends on the repeat variable of the current workframe. Here, wf_Study has repeat set to "true" (always repeat) so rule 'Wf_True' is applicable. This does nothing more than pass the agent to the next set of rules used for handling variables (denoted Var_*). There are two other repeat rules: 'Wf_False' (never repeat) means the workframe would be deleted from the set of workframes when finished; and 'Wf_Once' (repeat only one more time) sets the repeat variable in the workframe to false from this point onward.

Both the Clock and Alex now move to pop elements off the stack associated with the current workframe. These rules are denoted with 'Pop_*'.

Popping the stack. Both the Clock object and our Alex agent are now processing the elements on their current workframe's stack of activities, using the rules denoted by '**Pop_**'. wf_Study tells Alex to perform a primitive activity (essentially a wait) called $Study$ with duration 3000 seconds. Meanwhile the Clock has an activity to wait 3600 seconds before updating the time. For convenience we will simply show the current workframe's instruction stack in the following tuples, not the full workframe.

The current states of Alex and the Clock are:

$$\langle ag_{Alex}, \emptyset, \langle Study_d, [\texttt{Prim\_Act}^{3000}]\rangle, Pop\_*, B_{Alex}, F, 0, TF_{Alex}, WF_{Alex}\rangle$$

$$\langle\ ob_{Clock}, \emptyset, \langle AsTimeGoesBy_d, [\texttt{Prim\_Act}^{3600}]\rangle, Pop\_*, \\ B_{Clock}, F, 0, TF_{Clock}, WF_{Clock}\rangle$$

The Clock and Alex communicate the duration of their current activity to the scheduler by updating the system's beliefs about the time they are due to finish their next event. This is done using the rule '**Pop_PASend**'(Recall that individual agents can still act upon the main system state, e.g. T_ξ and B_ξ here).

RULE: **Pop_PASend**

$$\langle ag_i, \emptyset, \langle \beta_d, [\texttt{Prim\_Act}^t; \beta_{ins}]\rangle, Pop\_*, B_i, F, T_i, TF_i, WF_i\rangle$$
$$\xrightarrow[T_\xi = T_i]{B'_\xi = B_\xi \setminus \{T_i\} \cup \{T_i = T_i + t\}}$$
$$\langle ag_i, \emptyset, \langle \beta_d, [\texttt{Prim\_Act}^t; \beta_{ins}]\rangle, Pop\_PA*, B_i, F, T_i, TF_i, WF_i\rangle$$

$B'_\xi = B_\xi \setminus \{T_i\} \cup \{T_i + t\}$ shows that B'_ξ is a copy of B_ξ where T_i has been replaced by $T_i + t$.

Once all the agents have communicated their duration (excluding idle agents) the scheduler compares their durations to find the shortest activity and updates its internal clock using '**Sch_rcvd**'. In this case it updates the time from 0 to 3000 which is when Alex's activity will finish.

Both Alex's and the Clock's time remain at 0 but the global clock is now at 3000. The '**Pop_PA***' rules all have a time difference as a guard and act to decrease the remaining duration of the current activity and update the agent/object's internal time keeping.

In this situation the Clock object's primitive activity duration is decreased to 600. Alex's activity will have finished (since it is 3000) and a different rule, '**Pop_PA(t=0)**' is invoked:

RULE: **Pop_PA(t=0)**

$$\langle ag_i, \emptyset, \langle \beta_d, [\texttt{Prim\_Act}^t; \beta_{ins}]\rangle, Pop\_PA*, B_i, F, T_i, TF_i, WF_i\rangle$$
$$\xrightarrow[\neg(T_\xi = T_i), T_i + t - T_\xi = 0]{T'_i = T_\xi}$$
$$\langle ag_i, \emptyset, \langle \beta_d, \beta_{ins}\rangle, Pop\_concWf*, B, F, T'_i, TF_i, WF_i\rangle$$

Alex has now moved on to a stage where he will only perform 'conclude' actions in a workframe stack (denoted by stage '**Pop_concWf***'). There are no conclude actions on the stack so he is transferred to the workframe rules '**Wf_***' once more.

4.4 The Cycle Continues

The simulation continues to run and the Clock's time attribute is updated when its primitive activity finishes. This means Alex's belief about his perceived time no longer matches the Clock's time. Alex's thoughtframe, $tf\_FeelHungry$, becomes active. This places two "conclude" instructions on Alex's current thoughtframe stack. As mentioned above, thoughtframes act like workframes but only involve belief updates (conclude instructions) and so take no time. The rule 'Pop_concTf' updates an agent's belief using a thoughtframe.

RULE: Pop_concTf

$$\langle ag_i, \emptyset, \langle \beta_d, [conclude(b = v); \beta_{ins}] \rangle, Pop\_*, B_i, F, T_i, TF_i, WF_i \rangle$$
$$\xrightarrow[b=v' \in B_i]{B'_i = B_i \setminus \{b=v'\} \cup \{b=v\}}$$
$$\langle ag_i, \emptyset, \langle \beta_d, \beta_{ins} \rangle, Pop\_*, B'_i, F, T_i, TF_i, WF_i \rangle$$

After applying 'Pop_concTf' twice to conclude first *perceivedtime = Clock.time* and then *hungry = hungry + 3* Alex's beliefs are:

$$\left\{ \begin{array}{l} hungry = 18, Loc = Library, \\ Bob.Loc = Home, percievedtime = 1, \\ Clock.time = 1, \quad desiredRestaurant = \emptyset \end{array} \right\}$$

Alex continues to study. The cycle of Alex and the Clock counting time, studying and increasing hunger continues until the point where Alex's hunger level is 21 or above and he decides it is time to find some food.

The situation is as follows: the simulation time is 10800; Alex's hunger is 21; the Clock's time attribute is 2 and it has just completed a primitive activity causing Alex's $tf\_FeelHungry$ thoughtframe to execute; Alex has an active workframe with 1200 seconds remaining of study time. However, a thoughtframe has now been activated and has updated Alex's beliefs to conclude that his intended restaurant is Raleigh's. In our tuple representation the states of the Alex agent and the Clock object are:

$$\langle ag_{Alex}, \emptyset, \langle Study_d, [\texttt{Prim\_Act}^{1200}] \rangle, Pop\_*, B_{Alex}, F, 10800, TF_{Alex}, WF_{Alex} \rangle$$

$$\langle ob_{Clock}, \emptyset, \emptyset, fin, B_{Clock}, F, 10800, TF_{Clock}, WF_{Clock} \rangle$$

Alex is now hungry. Although Alex has a currently active workframe, the workframe $wf\_MoveToRestaurant$ is now applicable and has a higher priority. This requires that the current workframe is suspended. This is achieved by creating a new workframe which stores wf_Study's remaining instructions. This is achieved by 'Wf_Suspend'.

RULE: Wf_Suspend

$$\langle ag_i, \emptyset, \langle \beta_d, \beta_{ins} \rangle, Wf\_*, B_i, F, T_i, TF_i, WF_i \rangle$$
$$\xrightarrow[\exists W \in WF_i | B_i \models W^g \& W^{pri} > (\beta^{pri}+0.3)]{\beta' = \beta \setminus \{\beta^{pri}\} \cup \{\beta^{pri} = \beta^{pri}+0.2\}], WF'_i = WF_i \cup \beta'}$$
$$\langle ag_i, \emptyset, \emptyset, Wf\_*, B_i, F, T_i, TF_i, WF'_i \rangle$$

After applying this rule Alex's set of workframes becomes

$$WF_{Alex} = \left\{ \begin{array}{lll} \langle Study_d, [\texttt{Prim\_Act}^{1200}] \rangle, & wf\_MoveToRestaurant, \\ wf\_Study, & wf\_Wait, & wf\_Eat \end{array} \right\}$$

Alex calls Bob to meet. A simple communication activity is performed by Alex during a workframe which sends a message to Bob indicating that he wishes to meet for food. This communication works like a primitive activity followed by a simple belief update, the primitive activity would represent the duration of the communication. This communication will change the beliefs Bob has about Alex such that Bob will now believe (for simplicity) `meetAlex = true`, `Alex.desiredRestaurant = Raleigh`. Bob will then perform actions, similar to Alex's in the following, in order to get to Raleigh's restaurant.

Alex goes out for food. Alex has now selected a workframe to move to Raleigh's restaurant, $wf\_MoveToRestaurant$. This workframe is different to those we have seen previously because it has a *move* activity, which acts like a primitive activity followed by a conclude. The primitive activity has the duration dependant on journey time (calculated via pre-processing) and the conclude updates beliefs and facts of the our agent's location which, in this case, will be Raleigh's restaurant.

Waiting for Bob. Alex has arrived before Bob, so Alex initiates a workframe to wait for Bob. This workframe contains a detectable which detects when Bobs location is `Telegraph_Av_2405`. When Bob eventually arrives, an external belief (a fact) is updated which activates the detectable. The detectable on Alex's workframe which is of type 'Abort'. This abortion will cancel the current workframe and any activities Alex is working on but will also update his beliefs to match the fact (Bob's new location). The abort is is handled by 'Det_Abort':

RULE: Det_Abort

$$\langle ag_i, \emptyset, \langle \beta_d, \beta_{ins} \rangle, Det\_*, B_i, F, T_i, TF_i, WF_i \rangle$$
$$\xrightarrow[\exists d \in \beta^D | \exists F' \subseteq F \models d^g \& d^{type} = Abort]{B'_i = B_i \cup F'}$$
$$\langle ag_i, \emptyset, \emptyset, Det\_*, B'_i, T_i, F, TF_i, WF_i \rangle$$

where d is the detectable, β^D is the set of all detectables for workframe β, d^g is the *guard* condition of the detectable d, and d^{type} is the detectable type: *Impasse*; *Complete*; *Continue*; or *Abort*.

Scenario Conclusion. Alex's waiting has now been terminated and guards are satisfied for him to start the workframe to eat with Bob. The scenario finally terminates when the Clock object's time attribute has reached 20 hours, this is an additional condition in every single workframe/thoughtframe. After 20 hours each entity will no longer have any frames active so they all enter an idle state prompting the scheduler to use the 'Sch_Term' rule.

5 Concluding Remarks and Future Work

In this paper we have outlined the first formal semantics for the Brahms language. While the full semantics is given in the associated technical report [16], the worked scenario described above demonstrates much of the semantics of Brahms, including the most important aspects: *selection* of workframes and thoughtframes; *suspension* of workframes when a more important (higher priority) workframe becomes active; *detection* of facts; *performance* of 'concludes', primitive activities, 'move' activities and communication activities; and the use of the scheduler.

This formal semantics provides us with a route towards the formal verification of Brahms applications. Using these operational semantics we can devise model checking procedures and can either invoke standard model checkers, such as `Spin` [6] or agent model checkers such as `AJPF` [1]. Currently we are developing a translator for Brahms which, via the transition rules in our semantics, will then be able to generate input for such model checkers.

We are also currently identifying a suite of example Brahms scenarios (together with their required properties) for evaluating this tool. For the small Brahms example developed in Section 4, we might wish to verify that: "Alex will never starve"; or "Alex will eventually reach Raleigh's".

Brahms is an important language. It has been used to model very extensive applications in agent-human teamwork. While we have emphasized applications in space exploration, Brahms is equally at home in describing more mundane applications in home health-care or human-robot interaction. As such, the formal verification of this language would be very useful for assessing human safety; the operational semantics developed here are a necessary first step towards this.

References

1. Bordini, R.H., Dennis, L.A., Farwer, B., Fisher, M.: Automated Verification of Multi-Agent Programs. In: Proc. 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 69–78 (2008)
2. Bordini, R.H., Fisher, M., Sierhuis, M.: Formal Verification of Human-Robot Teamwork. In: Proc. 4th ACM/IEEE International Conference on Human Robot Interaction (HRI), pp. 267–268. ACM Press, New York (2009)
3. Clancey, W., Sierhuis, M., Kaskiris, C., van Hoof, R.: Advantages of Brahms for Specifying and Implementing a Multiagent Human-Robotic Exploration System. In: Proc. 16th Florida Artificial Intelligence Research Society (FLAIRS), pp. 7–11. AAAI Press, Menlo Park (2003)

4. Clancey, W.J., Sierhuis, M., Seah, C., Buckley, C., Reynolds, F., Hall, T., Scott, M.: Multi-agent Simulation to Implementation: A Practical Engineering Methodology for Designing Space Flight Operations. In: Artikis, A., O'Hare, G.M.P., Stathis, K., Vouros, G.A. (eds.) ESAW 2007. LNCS (LNAI), vol. 4995, pp. 108–123. Springer, Heidelberg (2008)
5. Dennis, L.A., Fisher, M., Lisitsa, A., Lincoln, N., Veres, S.M.: Satellite Control Using Rational Agent Programming. IEEE Intelligent Systems 25(3), 92–97 (2010)
6. Holzmann, G.J.: Software model checking with SPIN. Advances in Computers (2005)
7. Rao, A.S., Georgeff, M.: BDI Agents: From Theory to Practice. In: Proc. 1st International Conference on Multi-Agent Systems (ICMAS), San Francisco, USA, pp. 312–319 (1995)
8. Rao, A.S., Georgeff, M.P.: Modeling Agents within a BDI-Architecture. In: Proc. Conference on Knowledge Representation & Reasoning (KR). Morgan Kaufmann, San Francisco (1991)
9. Sierhuis, M.: Modeling and Simulating Work Practice. BRAHMS: a multiagent modeling and simulation language for work system analysis and design. PhD thesis, Social Science and Informatics (SWI), University of Amsterdam, The Netherlands (2001)
10. Sierhuis, M.: Multiagent Modeling and Simulation in Human-Robot Mission Operations (2006), http://ic.arc.nasa.gov/ic/publications
11. Sierhuis, M.: Brahms Language Specification, http://www.agentisolutions.com/documentation/language/LanguageSpecificationV3.0F.pdf
12. Sierhuis, M., Bradshaw, J.M., Acquisti, A., Hoof, R.V., Jeffers, R., Uszok, A.: Human-Agent Teamwork and Adjustable Autonomy in Practice. In: Proc. 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space, i-SAIRAS (2003)
13. Sierhuis, M., Clancey, W.J.: Modeling and Simulating Work Practice: A Human-Centered Method for Work Systems Design. IEEE Intelligent Systems 17(5) (2002)
14. Sierhuis, M., Clancey, W.J., van Hoof, R.J., Seah, C.H., Scott, M.S., Nado, R.A., Blumenberg, S.F., Shafto, M.G., Anderson, B.L., Bruins, A.C., Buckley, C.B., Diegelman, T.E., Hall, T.A., Hood, D., Reynolds, F.F., Toschlog, J.R., Tucker, T.: NASA's OCA Mirroring System: An application of multiagent systems in Mission Control (2009)
15. Plotkin, G.D.: A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, Computer Science Department. Aarhus University, Denmark (1981)
16. Stocker, R., Sierhuis, M., Dennis, L., Dixon, C., Fisher, M.: A Formal Semantics for the Brahms Language (2011), http://www.csc.liv.ac.uk/~rss/publications
17. van Hoof, R.: Brahms website (2000), http://www.agentisolutions.com
18. Wooldridge, M.: An Introduction to Multiagent Systems. John Wiley & Sons, Chichester (2002)

Making Golog Norm Compliant

Alfredo Gabaldon

Center for Artificial Intelligence (CENTRIA)
Universidade Nova de Lisboa
ag@di.fct.unl.pt

Abstract. In this work we consider how to enforce norms in the Situation Calculus based programming language Golog and its relatives. We define a notion of *norm compliant sequence of actions* with respect to norms prescribing some actions to be forbidden or obliged (ought-to-do norms), norms prescribing that a state-condition is forbidden (ought-to-be norms) and norms that are a form of deadline. We then show a procedure that allows incorporating the norms into the underlying action theory so that after this is done, the agent's behavior is guaranteed to be norm compliant.

1 Introduction

The use of *social laws* or *norms* as a behavior and coordination mechanism has attracted considerable interest among researchers in the area of autonomous agents and multi-agent systems. The work in this area includes [1,2,3,4,5,6] among many others.

Much of the current work on norms in autonomous agents involves developing agent programming languages that include facilities for expressing norms and mechanisms for enforcing them. Along these lines, in this work we look at an agent programming language and consider adding expressions for describing norms and then consider what it means for an agent programmed in this language to comply with the norms. The particular language we consider is Golog [7]. This high-level action programming language was developed for providing artificial agents with complex behaviors defined in terms of a set of primitive operations or actions. Golog consists of a set of programming constructs typical of imperative programming languages, e.g. sequence, conditional, iteration, and also some non-deterministic constructs such as choice between two subprograms. A distinguishing feature of Golog is that the primitive constructs are *actions* formalized in an underlying logic—the Situation Calculus [8].

Golog has grown into a family of languages which extend it with various features such as concurrency (ConGolog [9]), decision theory (dtGolog [10]), and incremental execution and sensing (IndiGolog [11]), among others. The underlying action language has also undergone substantial development with extensions including adding explicit time, an epistemic modality for knowledge, and stochastic actions. This makes the Golog family of languages an attractive choice for programming autonomous agents.

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 275–292, 2011.

2 The Golog Language

We briefly review the main components of a Basic Action Theory [12,13] and of the Golog language [7].

2.1 Basic Action Theories

A *basic action theory* is a classical logic formalization of the dynamic domain of an agent(s) in the *Situation Calculus* (SitCalc for short) [14]. The ontology of the SitCalc includes *actions*, *fluents*, which are the properties of the domain that change when actions occur, and *situations*, which are sequences of actions representing possible ways in which the domain may evolve.

Formally, the SitCalc is a dialect of First-Order logic with sorts *action, situation*, and *object*. Consequently, actions, situations and domain objects are treated as quantifiable, first-class citizens in the language. A special constant S_0 is used to denote the initial situation, and the function *do* of sort $(action \times situation) \mapsto situation$ is used to form sequences of actions. For instance, a sequence consisting of actions a_1, a_2, a_3 is represented by the term $do(a_3, do(a_2, do(a_1, S_0)))$.[1]

Fluents are represented by means of relations $F(\boldsymbol{x}, s)$ where $\boldsymbol{x}$ is a tuple of arguments of sorts *object* or *action* and the last argument s always of sort *situation*. For example, a fluent $owner(ag, file, s)$ could be used to represent that an agent ag is the owner of a $file$ in situation s.

Function symbols of sort $object \mapsto action$, $A(\boldsymbol{x})$, represent action types. For instance, a function $write(ag, file)$ could be used to represent the action of an agent ag writing to a $file$. We call them action *types* because a single function symbol can be used to create multiple *instances* of an action, e.g. the instances $write(Ag_1, File_1)$, $write(Ag_2, File_2)$, etc. We will use $a_1, a_2, \ldots$ to denote action variables and $\alpha_1, \alpha_2, \ldots$ to denote action terms. Similarly, we use $s_1, s_2, \ldots$ for situation variables and $\sigma_1, \sigma_2, \ldots$ for situation terms.

A Basic Action Theory $\mathcal{D}$ consists of the following sets of axioms (variables that appear free are implicitly universally quantified. $\boldsymbol{x}$ denotes a tuple of variables $x_1, \ldots, x_n$):

1. For each action type $A(\boldsymbol{x})$ there is exactly one **Action Precondition Axiom** (APA), of the form:

$$Poss(A(\boldsymbol{x}), s) \equiv \Pi_A(\boldsymbol{x}, s)$$

where variable s is the only term of sort situation in formula $\Pi_A(\boldsymbol{x}, s)$. The latter formula represents the conditions under which an action $A(\boldsymbol{x})$ is executable. The restriction that the only situation mentioned in this formula is s intuitively means that these preconditions depend only on the situation where the action would be executed.

[1] $do([a_1, a_2, \ldots, a_n], s)$ is an abbreviation of $do(a_n, do(a_{n-1}, \ldots, do(a_1, s)\ldots))$.

2. For each fluent $F(\boldsymbol{x}, s)$, there is exactly one **Successor State Axiom** (SSA), of the form:

$$F(\boldsymbol{x}, do(a, s)) \equiv \Phi_F(\boldsymbol{x}, a, s)$$

where s is the only term of sort situation in formula $\Phi_F(\boldsymbol{x}, a, s)$. This formula represents all and the only conditions under which executing an action a in a situation s results in a situation $do(a, s)$ where the fluent holds. These axioms embody Reiter's solution to the frame problem [12,13].
3. A set of sentences $\mathcal{D}_{S_0}$ describing the initial state of the world. This is a finite set of sentences whose only situation term may be the constant S_0 and describe the initial state of the domain. Any sentence is allowed as long as the only situation variable that appears in it is S_0, so one can write sentences such as $(\exists ag) owner(ag, File_1, S_0)$, reflecting incomplete information about the initial state.
4. The **Foundational Axioms** Σ which define situations in terms of the constant S_0 and the function do. Intuitively, these axioms define a tree-like structure for situations with S_0 as the root of the tree. They also define relation $\sqsubset$ on situations. Intuitively, $s \sqsubset s'$ means that the sequence of actions s is a prefix of sequence s'.
5. A set of unique names axioms (UNA) for actions. For example, $write(ag, f) \neq delete(ag, f)$, $write(ag, f) = write(ag', f') \supset (ag = ag' \wedge f = f')$, etc.

Given a basic action theory $\mathcal{D}$ we can define a few basic reasoning tasks. For instance, checking if a sequence of actions is executable, i.e. physically possible for the agent according to the axiomatization of its dynamic environment. This check is formally defined as follows: let $\alpha_1, \ldots, \alpha_k$ be action terms

$$\mathcal{D} \models executable(do([\alpha_1, \ldots, \alpha_k], S_0))$$

where $executable(\cdot)$ is defined as follows:

$$executable(s) \equiv (\forall a, s').do(a, s') \sqsubseteq s \supset Poss(a, s').$$

Another reasoning problem is *projection*: checking if some condition, denoted by a formula $\phi(s)$ with a free variable s, holds after a sequence of actions is executed:

$$\mathcal{D} \models \phi(do([\alpha_1, \ldots, \alpha_k], S_0)).$$

2.2 Golog

The situation calculus based programming language Golog [7] and variants such as ConGolog [9] and IndiGolog [11], provide Algol-like programming constructs for defining complex behaviors in terms of the primitive actions formalized in a basic action theory of the form described above. Among various applications, these languages have been employed for programming autonomous agents. For

the purpose of this work, which mainly deals with the logic underlying these languages, we need not go into the details. Here we will refer to Golog when generally referring to the family of Golog variants.

In addition to atomic actions, which are the primitive construct in Golog, the language includes constructs such as a test $\phi?$, test whether ϕ currently holds; sequence $\delta_1;\delta_2$, execute program δ_1 followed by δ_2; non-deterministic choice $\delta_1|\delta_2$, choose between executing δ_1 and executing δ_2; among others.

An important aspect of these languages is the fact that they allow one to write non-deterministic programs. Intuitively, the execution of a program $\delta_1|\delta_2$ can result in the execution of either one of δ_1 and δ_2, as long as their execution is successful. A program may fail to execute if one of the primitive actions in its execution trace turns out not to be executable. In the case of programs run concurrently, e.g. $\delta_1\|\delta_2$, the result of the execution is any of the possible interleavings of the primitive actions that result from δ_1 and those from δ_2. Again, some interleavings may fail because one of the actions is not executable at the given time.

The semantics for these languages is captured through a relation $Do(\delta, s, s')$, meaning that executing program δ in situation s results in situation s'. Given a background theory $\mathcal{D}$, including a definition of $Do(\delta, s, s')$, the execution of a program δ in the initial situation S_0 is defined in terms of logical entailment as the problem of finding a sequence of actions $\alpha_1, \ldots, \alpha_k$ such that

$$\mathcal{D} \models Do(\delta,\ S_0,\ do([\alpha_1, \ldots, \alpha_k], S_0)).$$

3 Norms

Social laws, norms or policies, are used as mechanisms for regulating the behavior of agents, as a mechanism for coordination, and for access control in systems security, among others. Many normative system frameworks use pairs of expressions (ϕ, a) to represent norms. The intuitive meaning of a pair (ϕ, a) would be that in states where ϕ holds, the action a is permitted/forbidden/obligatory.

We will represent norms in terms of formulae denoted by $\phi(\boldsymbol{x}, s)$ and $\psi(\boldsymbol{x}, s)$ with free-variables $\boldsymbol{x}, s$, with s the only situation term appearing in the formulae and $\boldsymbol{x}$ the remaining free variables. Norms are enforced in all situations so when writing them we will omit the universally quantified situation variable and write $\phi(\boldsymbol{x})$ and $\psi(\boldsymbol{x})$. We will use the notation $\mathbf{F}\ a$ to denote that an action a is forbidden, $\mathbf{O}\ a$ to denote that a is (immediately) obligatory, and $\mathbf{F}\ \psi$ to denote that a (state) condition ψ is forbidden.

We will assume that actions are permitted and not obligatory by default, that is, if there is no norm that in a given situation says that an action is forbidden (resp. obligatory), then the action is assumed permitted (resp. not obligatory). For this reason, it is unnecessary to write norms using negation in front of $\mathbf{F}\ a$ and $\mathbf{O}\ a$. Modifying the formalization to make the opposite assumptions, for example that actions are assumed forbidden unless a norm says otherwise, is straight forward.

In the norm expressions below, $\boldsymbol{t}, \boldsymbol{v}$ are tuples of terms, $\boldsymbol{t}$ a subset of the terms $\boldsymbol{v}$, and any variables appearing free, including omitted situation variables, are implicitly universally quantified.

3.1 Ought-to-Do Norms

1. Forbidden actions: $\phi(\boldsymbol{t}) \rightarrow \mathbf{F}\ A(\boldsymbol{v})$.
   Example: regular users are not allowed to write to files owned by others:

$$file(f) \wedge regUsr(r) \wedge \neg owner(r, f) \rightarrow \mathbf{F}\ write(r, f).$$

2. Obligatory actions: $\phi(\boldsymbol{t}) \rightarrow \mathbf{O}\ A(\boldsymbol{v})$.
   Example: if a licensed file has an expired license, the owner must delete it.

$$file(f) \wedge owner(r, f) \wedge expLic(f) \rightarrow \mathbf{O}\ del(r, f)$$

Given a set of norms, we can define a notion of *compliance* by a program with the norms. To that end, it will be useful to take a set of norms in the above forms and put them in a compact normal form by applying the following steps.

1. Take each norm $\phi(\boldsymbol{t}) \rightarrow \mathbf{F}\ A(\boldsymbol{v})$ and rewrite it so as to replace the parameters with variables::

$$\phi(\boldsymbol{x}') \wedge \boldsymbol{x}' = \boldsymbol{t} \wedge \boldsymbol{x}'' = \boldsymbol{u} \rightarrow \mathbf{F}\ A(\boldsymbol{x})$$

   where $\boldsymbol{u}$ are the terms in $\boldsymbol{v}$ not in $\boldsymbol{t}$ and $\boldsymbol{x}', \boldsymbol{x}''$ are among the variables $\boldsymbol{x}$. Let us denote the resulting formula by $\Phi(\boldsymbol{x}) \rightarrow \mathbf{F}\ A(\boldsymbol{x})$.
2. Next, for each action type $A(\boldsymbol{x})$ we take all the norms

$$\begin{aligned} &\Phi_1(\boldsymbol{x}) \rightarrow \mathbf{F}\ A(\boldsymbol{x}) \\ &\Phi_2(\boldsymbol{x}) \rightarrow \mathbf{F}\ A(\boldsymbol{x}) \\ &\ldots \\ &\Phi_k(\boldsymbol{x}) \rightarrow \mathbf{F}\ A(\boldsymbol{x}) \end{aligned}$$

   and rewrite them as the following single norm for $A(\boldsymbol{x})$:

$$\left[\bigvee_{i=1\ldots k} \Phi_i(\boldsymbol{x})\right] \rightarrow \mathbf{F}\ A(\boldsymbol{x})$$

   Let us denote the result by $\Phi_A(\boldsymbol{x}) \rightarrow \mathbf{F}\ A(\boldsymbol{x})$. This norm now completely characterizes the conditions that make any instance, i.e. for any $\boldsymbol{x}$, of the action type $A(\boldsymbol{x})$ forbidden.
3. Next we take each norm $\Phi_{A_i}(\boldsymbol{x}_i) \rightarrow \mathbf{F}\ A_i(\boldsymbol{x}_i)$ and rewrite it as follows using a fresh action variable a:

$$\Phi_{A_i}(\boldsymbol{x}_i) \wedge a = A_i(\boldsymbol{x}_i) \rightarrow \mathbf{F}\ a$$

4. Finally, we gather all the norms

$$\begin{array}{l} \Phi_{A_1}(\boldsymbol{x}_1) \wedge a = A_1(\boldsymbol{x}_1) \ \rightarrow \ \mathbf{F}\ a \\ \Phi_{A_2}(\boldsymbol{x}_2) \wedge a = A_2(\boldsymbol{x}_2) \ \rightarrow \ \mathbf{F}\ a \\ \ldots \\ \Phi_{A_n}(\boldsymbol{x}_n) \wedge a = A_n(\boldsymbol{x}_n) \ \rightarrow \ \mathbf{F}\ a \end{array}$$

and rewrite them as a single expression:

$$\left[\bigvee_{i=1\ldots n} \Phi_{A_i}(\boldsymbol{x}_i) \wedge a = A_i(\boldsymbol{x}_i) \right] \ \rightarrow \ \mathbf{F}\ a$$

Let us denote the result by $\Phi_F(a) \ \rightarrow \ \mathbf{F}\ a$.

Following the same steps with obligation norms, we can obtain the corresponding expression $\Phi_O(a) \ \rightarrow \ \mathbf{O}\ a$.

By restoring the situation argument on the left-hand-side formula to obtain $\Phi_F(a, s)$, resp. $\Phi_O(a, s)$, we now have a legit SitCalc formula that tells us, in any given situation, whether or not an action is forbidden, resp. obligatory, according to the system norms.

We can then define a notion of a sequence of actions s being compliant with a set of norms in the above forms, by means of the following equivalence:

$$\begin{array}{rl} \mathit{compliant}(s) \equiv & s = S_0 \vee \\ & (\exists a, s').s = do(a, s') \wedge \mathit{compliant}(s') \wedge \\ & \neg\Phi_F(a, s') \wedge (\forall a')[\Phi_O(a', s') \supset a' = a]. \end{array} \tag{1}$$

Intuitively, this says that a sequence of actions s is compliant with the given norms iff s is the empty sequence, S_0, or s is s' followed by a, where s' is compliant and a satisfies the norms in s'.

Given a set of norms and a corresponding definition of $\mathit{compliant}(s)$, we can define compliance with the norms by a Golog program δ. We will say that δ is compliant with the norms if all its execution traces comply with the norms.

Definition 1. *Let $\mathcal{D}$ be a basic action theory, δ a program, and N a set of norms with corresponding definition of $\mathit{compliant}_N(s)$. Program δ is* compliant *with norms N iff*

$$\mathcal{D} \models (\forall s).Do(\delta, S_0, s) \supset \mathit{compliant}_N(s).$$

The above definition assumes the program is executed in the initial situation S_0. A stronger version of compliance of a program can be defined by requiring the program to satisfy the norms when executed in any situation:

$$\mathcal{D} \models (\forall s', s).Do(\delta, s', s) \supset \mathit{compliant}_N(s).$$

On the other hand, a weak version of compliance can be simply defined by just requiring the program to have at least one compliant execution trace:

$$\mathcal{D} \models (\exists s).Do(\delta, S_0, s) \wedge compliant_N(s).$$

In terms of analyzing the norms themselves, a problem that is often of interest is that of checking whether two sets of norms are equivalent or if one set is subsumed by another. These problems can be defined in terms of classical entailment in our language. Consider two sets of norms N_1, N_2 with corresponding definitions of compliance denoted by $compliant_{N_1}(\cdot)$ and $compliant_{N_2}(\cdot)$ and defined by an equivalence of the form (1).

Definition 2. *We say that the sets of norms N_1, N_2 are equivalent (wrt $\mathcal{D}$) iff*

$$\mathcal{D} \models (\forall s).compliant_{N_1}(s) \equiv compliant_{N_2}(s).$$

Intuitively, the norms are said to be equivalent if the sequences that are compliant under one set of norms are exactly the same sequences that are compliant under the other set of norms.

Definition 3. *We say that the set of norms N_1 subsumes the set of norms N_2 (wrt $\mathcal{D}$) iff*

$$\mathcal{D} \models (\forall s).compliant_{N_1}(s) \supset compliant_{N_2}(s).$$

As expected, two sets of norms N_1, N_2 are equivalent if they coincide, in all situations, in designating the same actions as forbidden or obligatory. This is established formally as follows.

Proposition 1. *Let N_1 be the norms $\Phi_F^1(a, s) \rightarrow \mathbf{F}\ a$ and $\Phi_O^1(a, s) \rightarrow \mathbf{F}\ a$ and N_2 be the norms $\Phi_F^2(a, s) \rightarrow \mathbf{F}\ a$ and $\Phi_O^2(a, s) \rightarrow \mathbf{F}\ a$.*

Then N_1 and N_2 are equivalent (wrt $\mathcal{D}$) iff

$$\mathcal{D} \models (\forall a, s).[\Phi_F^1(a, s) \equiv \Phi_F^2(a, s)] \ \wedge \ [\Phi_O^1(a, s) \equiv \Phi_O^2(a, s)].$$

Another problem of interest is checking whether a set of norms is *consistent* in some sense. Perhaps the simplest notion of consistency would be to define it as existence of compliant sequences. That is, a set of norms N is *inconsistent* iff

$$\mathcal{D} \models (\forall s).S_0 \sqsubset s \supset \neg compliant_N(s). \tag{2}$$

But this is probably too strong to be very useful. Perhaps a more useful notion would be one requiring only those sequences that are actually physically possible for the agent, to be compliant. This is expressed as follows:

$$\mathcal{D} \models (\forall s).[S_0 \sqsubset s \wedge executable(s)] \supset \neg compliant_N(s)$$

where $executable(s)$ is defined in terms of $Poss$ as described in Section 2.

Another possibly useful notion is that of consistency with respect to an agent's goal, as denoted by a formula $Goal(s)$. In this case we might say that a set of norms N is *inconsistent with respect to goal* $Goal(s)$ iff

$$\mathcal{D} \models (\exists s)Goal(s) \ \wedge \ (\forall s)[Goal(s) \supset \neg compliant_N(s)].$$

Intuitively, this says that a set of norms is inconsistent with respect to a goal if the goal is achievable but it is not possible to achieve it and satisfy the norms at the same time.

Other properties can be defined in a similar fashion in terms of logical entailment from a background theory $\mathcal{D}$. For example, a notion of two sets of norms being *equivalent with respect to a goal*, etc.

3.2 Ought-to-Be Norms

Ought-to-be norms specify situations in which a state condition, instead of an action as in ought-to-do norms, is forbidden or obligatory. We write such laws in the following form: $\Phi(\boldsymbol{t}) \ \rightarrow \ \mathbf{F} \ \Psi(\boldsymbol{v})$.

For example: when a user is logged out, no processes owned by the user should be executing:

$$loggedOut(usr) \wedge owner(usr, proc) \ \rightarrow \ \mathbf{F} \ executing(proc)$$

Since negation can appear in front of $\Psi(\boldsymbol{t})$, we do not use the obligation symbol $\mathbf{O}$ in these norms. Also, we understand these laws as dynamic, not static laws. That is, the condition $\Phi(\boldsymbol{t})$ is intended to be evaluated in the "current" state and the condition $\Psi(\boldsymbol{v})$ evaluated in the next state.

These norms cannot be put together into a normal form as in the case of ought-to-do norms, so we simply put them together as a conjunction of implications relative to a situation and its predecessor situation.

Compliance of a sequence is then defined as follows:

$$\begin{aligned} compliant(s) \equiv \ & s = S_0 \ \vee \\ & (\exists a, s').s = do(a, s') \wedge compliant(s') \ \wedge \\ & \textstyle\bigwedge_{i=1,\ldots,n}(\forall)[\Phi_i(\boldsymbol{t}_i, s') \supset \neg\Psi_i(\boldsymbol{v}_i, s)]. \end{aligned}$$

Having defined $compliant(s)$ this way for ought-to-be norms, the formal definitions of program compliance and norm equivalence, subsumption and consistency are exactly as for ought-to-do norms.

3.3 Deadlines

Consider an abstract form of deadline specifying that some condition ψ is forbidden before condition φ, written as $\mathbf{F} \ \psi \prec \varphi$ (deadlines like this are discussed in [15]). Norms involving deadlines would then take the form (to simplify the presentation, we assume all formulae have the same terms $\boldsymbol{x}$ as arguments):

$$\phi(\boldsymbol{x}) \;\rightarrow\; \mathbf{F}\ \psi(\boldsymbol{x}) \prec \varphi(\boldsymbol{x}). \tag{3}$$

For example, first-year students are not allowed to registered before the session starts:

$$firstyear(st) \;\rightarrow\; \mathbf{F}\ registered(st) \prec sessionstart.$$

Not surprisingly, compliance with a deadline is slightly more involved since it imposes conditions over a full sequence of actions. For the sake of clarity we will define it only for a single deadline of the form (3). The definition basically says that a sequence complies with a deadline if after any state where ϕ holds, either ψ never holds afterwards or φ holds at some point and ψ does not hold before that.

$$compliant(s) \equiv$$
$$(\forall)(\forall s_1).[s_1 \sqsubset s \wedge \phi(\boldsymbol{x}, s_1)] \supset$$
$$\left\{ \begin{array}{l} (\forall s_2)[s_1 \sqsubset s_2 \sqsubseteq s \supset \neg\psi(\boldsymbol{x}, s_2)] \vee \\ (\exists s_2).s_1 \sqsubset s_2 \sqsubseteq s \wedge \varphi(\boldsymbol{x}, s_2) \wedge (\forall s_3).[s_1 \sqsubset s_3 \sqsubseteq s_2] \supset \neg\psi(\boldsymbol{x}, s_3) \end{array} \right\}$$

As before, the definitions of program compliance and norm equivalence, subsumption and consistency apply to deadlines using the above definition of $compliance(s)$.

4 Internalizing Norms

The straight forward way of enforcing a set of norms on an agent is to check that the norms are satisfied every time the agent chooses an action to execute next. That is, when an agent considers whether or not to execute an action α in situation σ, in addition to checking that α is physically possible, it would also check whether $compliant(do(\alpha, \sigma))$ holds.

An alternative way of enforcing the norms is to incorporate the norms into the agent's dynamic world description $\mathcal{D}$. Once the agent has "internalized" the norms into $\mathcal{D}$, it would behave in a norm compliant way.

One advantage of internalizing the norms is that, especially in the case of deadlines, compliance becomes a local check involving only the current state and the action being considered. There is no need to check conditions in the resulting state after executing an action nor on past states as required by the definition of compliance for deadlines. In implementations that update the belief base after each action is performed, internalizing the norms has the advantage that computing the tentative new state for each considered action, in order to check norm compliance, becomes unnecessary. And of course there is no need to store past states either.

4.1 Ought-to-Do Norms

Since we take norms as constraints on the behavior of an agent, their practical effect is to render some actions non-executable. The natural way then to

incorporate a set of norms into the agent's domain description is in the form a additional preconditions in the APAs.

Consider the APA of an action type $A(\boldsymbol{x})$:

$$Poss(A(\boldsymbol{x}), s) \equiv \Pi_A(x, s).$$

The simplest way to incorporate a set of ought-to-do norms is by adding conditions saying a) $A(\boldsymbol{x})$ is not one of the forbidden actions, and b) if there is any obligatory action at all, it is $A(\boldsymbol{x})$. Assuming the set of norms are in the forms $\Phi_F(a) \rightarrow \mathbf{F}\ a$ and $\Phi_O(a) \rightarrow \mathbf{O}\ a$, we obtain the following modified APA for $A(\boldsymbol{x})$:

$$\begin{aligned} Poss(A(\boldsymbol{x}), s) \equiv\ & \Pi_A(x, s) \wedge \\ & (\forall a)[\Phi_F(a, s) \supset a \neq A(\boldsymbol{x})] \wedge \\ & (\forall a)[\Phi_O(a, s) \supset a = A(\boldsymbol{x})]. \end{aligned}$$

The additional preconditions added to each APA are the same except for the $A(\boldsymbol{x})$ term appearing in the consequent of the implications.

A more "efficient" way of incorporating the norms into the APAs, however, follows from the observation that each APA describes the preconditions of one specific action type $A(\boldsymbol{x})$. So any laws that forbid other actions are in fact irrelevant with respect to $A(\boldsymbol{x})$.

Taking then only those norms that forbid $A(\boldsymbol{x})$, if any, put in the form $\Phi_A(\boldsymbol{x}) \rightarrow \mathbf{F}\ A(\boldsymbol{x})$, as derived in Section 3, we modify the APA for $A(\boldsymbol{x})$ as follows:

$$\begin{aligned} Poss(A(\boldsymbol{x}), s) \equiv\ & \Pi_A(x, s) \wedge \\ & \neg\Phi_A(\boldsymbol{x}, s) \wedge \\ & (\forall a)[\Phi_O(a, s) \supset a = A(\boldsymbol{x})]. \end{aligned}$$

In this case the additional precondition $\neg\Phi_A(\boldsymbol{x}, s)$ is specific to the action type $A(\boldsymbol{x})$ so it varies with each APA.

Once the norms have been "compiled" into the underlying action theory $\mathcal{D}$, all programs will be compliant with the norms. Formally, let $\mathcal{D}$ be the background theory of an agent, N be a set of norms and $\mathcal{D}_N$ be the theory that results from applying the above transformation to $\mathcal{D}$ with respect to N.

Proposition 2. *For every program* δ, $\mathcal{D}_N \models (\forall s).Do(\delta, S_0, s) \supset compliant_N(s)$.

In other words, a sequence of actions is now considered to be a "legal" execution trace of the program δ only if it satisfies the norms.

Note that if N is inconsistent, e.g. as in (2), then δ has no legal execution traces, i.e. $\neg(\exists s)Do(\delta, S_0, s)$, and the implication in Prop. 2 is vacuously satisfied. If one does not want to count such a program as compliant, one can simply modify Def. 1 by adding the condition for weak compliance as a conjunct and define δ as compliant if it has at least one legal execution trace.

4.2 Ought-to-Be Norms

In the case of an ought-to-be norm of the form $\Phi(\boldsymbol{t}) \rightarrow \mathbf{F}\ \Psi(\boldsymbol{v})$, when the agent is considering an action α in a situation σ, it needs to check $\Phi(\boldsymbol{t})$ with respect to σ and $\Psi(\boldsymbol{v})$ with respect to $do(\alpha, \sigma)$. However, for the reasons mentioned earlier, we want to incorporate norms in terms of conditions on the current situation and the action under consideration. Moreover, according to the definition of APAs, the only term of sort situation allowed to appear in the formula on the right-hand-side of an APA is the variable s.

Fortunately, there is a mechanism that will allow us to compute a precondition relative to situation s from a condition relative to situation $do(A(\boldsymbol{x}), s)$. This mechanism is the *regression operator* $\mathcal{R}$ from [16,13]. Roughly, this operator takes a formula $\Gamma(do([\alpha_1, \ldots, \alpha_n], S_0))$ relative to a sequence of actions $do([\alpha_1, \ldots, \alpha_n], S_0))$ and computes a formula $\Gamma'(S_0)$ relative to S_0 that is equivalent to the original formula with respect to the background theory $\mathcal{D}$. The computation is purely syntactic and works by iteratively replacing each occurrence of a fluent $F(\boldsymbol{t}, do(\alpha, \sigma))$ with the formula $\Phi_F(\boldsymbol{t}, \alpha, \sigma)$ given a corresponding SSA $F(\boldsymbol{x}, do(a, s)) \equiv \Phi_F(\boldsymbol{x}, a, s)$. For details on operator $\mathcal{R}$, please refer to [16,13].

Consider any action type $A(\boldsymbol{x})$ and its corresponding APA

$$Poss(A(\boldsymbol{x}), s) \equiv \Pi_A(\boldsymbol{x}, s)$$

Let us now describe a procedure for incorporating a norm $\Phi(\boldsymbol{v}) \rightarrow \mathbf{F}\ \Psi(\boldsymbol{v})$ as additional preconditions.

1. Restore s as the situation argument in the premise: $\Phi(\boldsymbol{t}, s)$.
2. Restore $do(A(\boldsymbol{x}), s)$ as the situation argument in $\Psi(\boldsymbol{v})$ to obtain: $\Psi(\boldsymbol{v}, do(A(\boldsymbol{x}), s))$
3. Apply one regression step to the formula $\Psi(\boldsymbol{v}, do(A(\boldsymbol{x}), s))$ to obtain a formula $\Psi_A(\boldsymbol{v}, \boldsymbol{x}, s)$ relative to s, that is, let

$$\Psi_A(\boldsymbol{v}, \boldsymbol{x}, s) = \mathcal{R}^1[\Psi(\boldsymbol{v}, do(A(\boldsymbol{x}), s))].$$

4. Take the formulae from steps 1 and 3 and put them together in an implication as follows:

$$\Phi(\boldsymbol{t}, s) \supset \neg\Psi_A(\boldsymbol{v}, \boldsymbol{x}, s)$$

5. Finally, include the universal closure of the implication as an additional precondition in the APA for action $A(\boldsymbol{x})$:

$$Poss(A(\boldsymbol{x}), s) \equiv \Pi_A(\boldsymbol{x}, s) \wedge\ (\forall)[\Phi(\boldsymbol{t}, s) \supset \neg\Psi_A(\boldsymbol{v}, \boldsymbol{x}, s)].$$

The right-hand-side of the modified APA mentions only one situation term, s, as required. Note also that the subformula $\Psi_A(\boldsymbol{v}, \boldsymbol{x}, s)$ obtained by regression is specific to the action type $A(\boldsymbol{x})$. This is important because in many cases the result is that the subformula can be substantially simplified, as illustrated in the examples below. In fact, when a norm is completely irrelevant to a particular

action type, the subformula frequently simplifies into a formula which is clearly valid and can be removed altogether.

As an example, consider a robot that lives in a university classroom building and has the norm

$$lecture(rm) \ \rightarrow \ \mathbf{F}\ at(rm).$$

saying that if there is a lecture occurring in a room, it should not be there. Suppose that it has actions $enter(rm)$ for entering a room and the action $wait$, as a representative of other actions that are not relevant to the location of the robot. The latter actions are interesting to consider since on the surface they appear irrelevant to the law. Let the corresponding APAs be:

$$Poss(enter(x), s) \equiv \neg at(x, s) \wedge nextto(door(x), s).$$

$$Poss(wait, s) \equiv True.$$

Let the SSA of fluent $at(x, s)$ be as follows:

$$at(x, do(a, s)) \equiv a = enter(x) \vee at(x, s) \wedge \neg(\exists y)a = enter(y).$$

Let us apply the above procedure to incorporate the law into the APAs. Starting with action $enter(x)$, we compute the regression of $at(rm, do(enter(x), s))$:

$$\mathcal{R}[at(rm, do(enter(x), s))] \ = \\ enter(x) = enter(rm) \vee at(rm, s) \wedge \neg(\exists y)enter(x) = enter(y).$$

Since $\neg(\exists y)enter(x) = enter(y)$ is unsatisfiable, the resulting formula can be simplified to $enter(x) = enter(rm)$. By UNA on actions, this can be further simplified to $x = rm$. Thus the procedure yields the implication $(\forall rm)[lecture(rm, s) \supset x \neq rm]$ which can be further simplified to $\neg lecture(x, s)$. Adding this to the preconditions of $enter(x)$ we obtain the following APA:

$$Poss(enter(x), s) \equiv \neg at(x, s) \wedge nextto(door(x), s) \ \wedge \neg lecture(x, s).$$

Intuitively, from the general norm we have obtain the additional precondition specific to the enter action saying that there should not be a lecture in progress in the room to be entered.

Consider now the $wait$ action. Following the procedure, we compute the regression of $at(rm, do(wait, s))$:

$$\mathcal{R}[at(rm, do(wait, s))] \ = \\ wait = enter(rm) \vee at(rm, s) \wedge \neg(\exists y)wait = enter(y).$$

By UNA on actions, $wait \neq enter(y)$ for any y, so the resulting formula can be simplified to $at(rm, s)$. So the procedure yields the implication $lecture(rm, s) \supset \neg at(rm, s)$, and thus we obtain the following APA for the action $wait$:

$$Poss(wait, s) \equiv lecture(rm, s) \supset \neg at(rm, s)$$

which intuitively says that the agent can *wait* as long as it is not in a room where there is a lecture in progress. Exactly the same result would be obtained for similar actions that are not relevant to the location of the agent, such as *paint*(*obj*), *pickup*(*book*), etc. In other words, if the agent happens to be in a room where a lecture has started, the norm imposes the obligation on the agent to take immediate action to change location, by rendering all other actions impossible.

As in the case of ought-to-do norms, the above procedure yields an action theory $\mathcal{D}_N$ such that executing a program will now always result in a norm compliant sequence of actions, as formally stated in Proposition 2.

4.3 Deadlines

Let us finally look at how we might incorporate deadlines into an agent's background theory. As for the other types of norm, the aim is to extract additional preconditions from deadlines and add them to the APAs. The complication in the case of deadlines is that they are not local but may refer to situations arbitrarily far from the current one. In order to access those situations while satisfying the requirement of basic action theories that axioms only refer to the current situation, we employ a technique that consists in adding a small number of auxiliary fluents to keep track of whether certain conditions have been satisfied in a previous situation. This approach has been employed before in other contexts such as DB integrity constraints, automated planning and evolving knowledge bases [17,18,19,20,21,22].

For a deadline of the form $\phi(\boldsymbol{z}) \rightarrow \mathbf{F}\ \psi(\boldsymbol{z}) \prec \varphi(\boldsymbol{z})$ it suffices to add one auxiliary fluent $F_\phi(\boldsymbol{z}, s)$ with the following corresponding SSA:

$$F_\phi(\boldsymbol{z}, do(a, s)) \equiv [\phi(\boldsymbol{z}, do(a, s)) \vee F_\phi(\boldsymbol{z}, s)] \ \wedge \neg\varphi(\boldsymbol{z}, do(a, s)).$$

Intuitively, $F_\phi(\boldsymbol{z}, s)$ holds in s if the deadline is active in s.

The above axiom is actually not yet in the required form of an SSA because of the subformulae on the right-hand-side that have argument $do(a, s)$. This needs to be fixed by applying one regression step using operator $\mathcal{R}$ on those subformulae.

Having applying this procedure to a deadline, we add the corresponding auxiliary fluent F_ϕ and its SSA to the background theory. Then we modify the APA of each action type $A(\boldsymbol{x})$ by adding an additional precondition as follows:

$$\begin{aligned} Poss(A(\boldsymbol{x}), s) \equiv\ & \Pi_A(\boldsymbol{x}, s) \wedge \\ & (\forall \boldsymbol{z}) F_\phi(\boldsymbol{z}, s) \supset [\neg\psi(\boldsymbol{z}, do(A(\boldsymbol{x}), s)) \vee \varphi(\boldsymbol{z}, do(A(\boldsymbol{x}), s))] \end{aligned}$$

Note that this again requires applying regression in order to obtain a legit APA. Moreover, since regression is applied for the specific action type $A(\boldsymbol{x})$, it is possible that the resulting formulae can be substantially simplified as was the case with the ought-to-be norms. The resulting theory $\mathcal{D}_N$ yields a result similar to that in Proposition 2.

5 Related Work

In addition to work already mentioned, we discuss here some other related work.

Governatori and Rotolo [23,24] present a logical language called *Process Compliance Language* (PCL) which has deontic operators of several kinds: *punctual obligation*, *maintenance obligation* and *achievement obligation*. In [23], an algorithm for checking compliance of a business process is given. The algorithm takes an execution trace of the process, described by a Petri Net, and checks if norms are satisfied. While the norms are formalized in PCL, the process is described as a Petri Net and the algorithm is extra-logical. This differs from our approach where the norms, the process (program) and compliance is all expressed in the same logical language. Another difference is that PCL is a much richer norm language. In [24], they show that in addition to expressing norms, PCL can be used to describe the control flow of business processes using the deontic features of PCL. Since PCL specifications can be executed by a rule engine, expressing business processes in terms of PCL allows for the execution of business processes under PCL norms on the same rule engine. Similarly, our work is based on expressing both processes (Golog programs) and norms in the same language (the Situation Calculus), which is an advantage over approaches using different formalisms for each task. Also related is earlier work by this group on business processes and business contracts [25,26].

In [27], Meneguzzi and Luck present an approach to adjusting the behavior of an agent to newly accepted norms. Similar to ours, the approach is based on modifying the agent's behavior by changing the implementation of such behavior. The main difference with our approach is that they apply the modifications to the programs in the plan library, e.g. by removing plans that include a forbidden action. These plans are then restored when the corresponding norm expires and the action is no longer forbidden. In our case, the modification is done in the underlying primitive action theory, not at the plan library level. This has several advantages: 1) If a program is non-deterministic, a norm may make some execution traces illegal while others remain legal. In our case, after modifying the underlying action theory, executing the program would result only in legal execution traces but the program remains the same. In their approach, the only choice seems to be to remove the program altogether. 2) Whether an action has a forbidden effect or not may depend on the state where the action is executed, and in turn the same applies for an agent program. In our approach, the agent would still be able to execute a program in a situation that does not lead to a forbidden state, even if the plan would violate a norm if executed in another context. Again, in their case the only choice is to remove the program altogether. On the other hand, they consider the interesting case of accepting new norms at run-time, which requires the ability to abandon programs already in execution or in the intention base.

The recent work of van Riemsdijk et al. [28] also deals with norms in multi-agent systems. Their main concern is to formalize norms present in the MOISE$^+$

organizational modeling language [29]. The work is complementary to ours since we consider how to ensure an agent complies with a set of accepted norms, without considering the source of the norms, which could very well be an organization such as those modeled in MOISE$^+$. The approach in [28] is to formalize norms in LTL, which has a close correspondence to the language used here. This means those norms would not be too difficult to integrate with our approach.

There is also a large amount of related work on verifying that an agent conforms to an interaction protocol. We discuss some of that work next.

In [30], Endriss et al. consider the problem of checking that communicating agents conform to a public dialog protocol. The approach is based on abductive logic programming and protocols are formalized as integrity constraints which "filter out" illegal dialog moves. This allows a simple way to enforce a protocol: add the protocol to the agent's knowledge base. The analogous way to enforce prohibitions in our framework would be to modify the definition of the relation $Do(\delta, s', s)$ to include $compliant(s)$ as a condition on s. This would be obviously correct. But contrast that with our proposed procedure for internalizing norms: in the former approach, *all* the norms have to be checked for *every* action the agent intends to execute next. In the latter approach, only those norms which are actually relevant to a particular action need to be checked, and in a simplified form. In a sense, the process of internalizing the norms computes what norms are relevant and simplifies them for each action.

Baldoni et al. [31,32] consider a-priori verification of conformance of an agent with a role in an interaction protocol. The main concern there is to guarantee interoperability: that the system will function correctly provided agents conform to their roles in the protocol. The approach is base on finite state automata.

Singh et al. [33,34,35] also look at the problem of conformance with an interaction protocol, but their approach is based on *commitments*. In [33], compliance with commitments is reduced to model checking CTL formulae against a model of the agents' interactions. Thus it roughly corresponds in our framework to checking $compliant(s)$ on an execution trace s. In [35], using a notion of *run subsumption*, an agent is said to conform to a protocol if the execution traces of its program are subsumed by the execution traces of the protocol. While we do not consider it here, run subsumption conformance is similar to norm subsumption (see Def. 3) so it seems it would not be too difficult to define a similar notion.

Finally, Chesani et al. [36] formalize a form of run-time conformance checking based on commitments using a reactive version of the Event Calculus. The approach allows "full" and "partial" violations where the latter allow the agent to fulfil a commitment after the deadline by paying a penalty.

Approaches to protocol conformance are mainly concerned with guaranteeing global interoperability and hence take an external view of agents. This is complementary to our work here where the problem is to ensure compliance with a set of norms by incorporating them into the agent.

6 Conclusions

In this work we have considered how to express several types of norms, namely ought-to-do, ought-to-be and a form of deadline, and how to incorporate them into the framework of Golog. We define a notion of a sequence of agent actions complying with a set of norms and a formal definition of an agent's program complying with the norms. We also describe notions of equivalence between norm systems with respect to an agent's background theory in the Situation Calculus, as well as notions of norm system subsumption and consistency. We have also shown procedures for incorporating a set of norms into the formalization of the primitive actions of an agent so that after the norms have been thus internalized, the agent is guaranteed to behave in a norm compliant manner.

This is a first approach at the problem of regulating the behavior of a Golog agent using a set of norms, so we make many strong simplifying assumptions. For example, we assume that the agent has accepted the norms and will not violate them. This should be extended to allow the agent to violate some of the norms if desirable, and perhaps provide also a penalty mechanism. We would also like to look at more complex forms of deadline involving explicit time, especially since there is already a temporal extension of Golog [37]. Further work is also necessary on the multi-agent aspect of this work.

References

1. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: Off-line design. Artificial Intelligence 73(1-2), 231–252 (1995)
2. Dastani, M., Grossi, D., Meyer, J.J.C., Tinnemeier, N.A.M.: Normative multi-agent programs and their logics. In: Meyer, J.-J.C., Broersen, J. (eds.) KRAMAS 2008. LNCS, vol. 5605, pp. 16–31. Springer, Heidelberg (2009)
3. Boella, G., van der Torre, L.W.N.: Regulative and constitutive norms in normative multiagent systems. In: Dubois, D., Welty, C.A., Williams, M.A. (eds.) Ninth International Conference on Principles of Knowledge Representation and Reasoning, pp. 255–266 (2004)
4. Sergot, M.: Norms, action and agency in multi-agent systems. In: Governatori, G., Sartor, G. (eds.) DEON 2010. LNCS, vol. 6181, pp. 2–2. Springer, Heidelberg (2010)
5. Fitoussi, D., Tennenholtz, M.: Choosing social laws for multi-agent systems: Minimality and simplicity. Artificial Intelligence 119(1-2), 61–101 (2000)
6. Craven, R., Sergot, M.J.: Agent strands in the action language nC+. Journal of Applied Logic 6(2), 172–191 (2008)
7. Levesque, H., Reiter, R., Lespérance, Y., Lin, F., Scherl, R.B.: Golog: A logic programming language for dynamic domains. Journal of Logic Programming 31(1-3), 59–83 (1997)
8. McCarthy, J.: Situations, actions and causal laws. Technical report, Stanford University (1963); Reprinted in Semantic Information Processing (M. Minsky ed.), pp. 410–417. MIT Press, Cambridge (1968)
9. De Giacomo, G., Lesperance, Y., Levesque, H.: ConGolog, a concurrent programming language based on the situation calculus. Artificial Intelligence 121, 109–169 (2000)

10. Boutilier, C., Reiter, R., Soutchanski, M., Thrun, S.: Decision-theoretic, high-level agent programming in the situation calculus. In: Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 2000), Austin, Texas, pp. 355–362 (2000)
11. De Giacomo, G., Levesque, H.J.: An incremental interpreter for high-level programs with sensing. In: Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter, pp. 86–102. Springer, Heidelberg (1999)
12. Reiter, R.: The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In: Lifschitz, V. (ed.) Artificial Intelligence and Mathematical Theory of Computation, pp. 359–380. Academic Press, London (1991)
13. Reiter, R.: Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems. MIT Press, Cambridge (2001)
14. McCarthy, J., Hayes, P.: Some philosophical problems from the standpoint of artificial intelligence. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence, vol. 4, pp. 463–502. Edinburgh University Press (1969); Also appears in Nilsson, N., Webber, B.(eds.) Readings in Artificial Intelligence. Morgan-Kaufmann, San Francisco
15. Dignum, F., Broersen, J., Dignum, V., Meyer, J.J.C.: Meeting the deadline: Why, when and how. In: Hinchey, M.G., Rash, J.L., Truszkowski, W.F., Rouff, C. (eds.) FAABS 2004. LNCS (LNAI), vol. 3228, pp. 30–40. Springer, Heidelberg (2004)
16. Pirri, F., Reiter, R.: Some contributions to the metatheory of the Situation Calculus. Journal of the ACM 46(3), 325–364 (1999)
17. Chomicki, J.: Efficient checking of temporal integrity constraints using bounded history encoding. ACM Transactions on Database Systems 20(2), 148–186 (1995)
18. Gabaldon, A.: Compiling control knowledge into preconditions for planning in the situation calculus. In: Gottlob, G., Walsh, T. (eds.) 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), pp. 1061–1066 (2003)
19. Gabaldon, A.: Precondition control and the progression algorithm. In: Dubois, D., Welty, C., Williams, M.A. (eds.) 9th International Conference on Principles of Knowledge Representation and Reasoning (KR 2004), pp. 634–643 (2004)
20. Bienvenu, M., Fritz, C., McIlraith, S.A.: Planning with qualitative temporal preferences. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Tenth International Conference on Principles of Knowledge Representation and Reasoning, pp. 134–144 (2006)
21. Alferes, J.J., Gabaldon, A., Leite, J.A.: Evolving logic programming based agents with temporal operators. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2008), pp. 238–244. IEEE, Los Alamitos (2008)
22. Alferes, J.J., Gabaldon, A., Leite, J.A.: Evolving logic programs with temporal operators. In: Balduccini, M., Son, T. (eds.) Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning. LNCS (LNAI), vol. 6565, pp. 193–212. Springer, Heidelberg (2011)
23. Governatori, G., Rotolo, A.: A conceptually rich model of business process compliance. In: Link, S., Ghose, A. (eds.) 7th Asia-Pacific Conference on Conceptual Modelling (APCCM 2010), vol. 110, pp. 3–12 (2010)
24. Governatori, G., Rotolo, A.: Norm compliance in business process modeling. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 194–209. Springer, Heidelberg (2010)

25. Padmanabhan, V., Governatori, G., Sadiq, S.W., Colomb, R., Rotolo, A.: Process modelling: the deontic way. In: Stumptner, M., Hartmann, S., Kiyoki, Y. (eds.) 3rd Asia-Pacific Conference on Conceptual Modelling (APCCM 2006), vol. 53, pp. 75–84 (2006)
26. Governatori, G., Milosevic, Z., Sadiq, S.W.: Compliance checking between business processes and business contracts. In: 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2006), pp. 221–232. IEEE Computer Society, Los Alamitos (2006)
27. Meneguzzi, F., Luck, M.: Norm-based behaviour modification in BDI agents. In: Sierra, C., Castelfranchi, C., Decker, K.S., Sichman, J.S. (eds.) 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), pp. 177–184 (2009)
28. van Riemsdijk, M.B., Hindriks, K.V., Jonker, C.M., Sierhuis, M.: Formalizing organizational constraints: a semantic approach. In: van der Hoek, W., Kaminka, G.A., Lespérance, Y., Luck, M., Sen, S. (eds.) 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), pp. 823–830 (2010)
29. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. International Journal of AOSE 1(3-4), 370–395 (2007)
30. Endriss, U., Maudet, N., Sadri, F., Toni, F.: Protocol conformance for logic-based agents. In: Gottlob, G., Walsh, T. (eds.) 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), pp. 679–684 (2003)
31. Baldoni, M., Baroglio, C., Martelli, A., Patti, V.: A priori conformance verification for guaranteeing interoperability in open environments. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 339–351. Springer, Heidelberg (2006)
32. Baldoni, M., Baroglio, C., Chopra, A.K., Desai, N., Patti, V., Singh, M.P.: Choice, interoperability, and conformance in interaction protocols and service choreographies. In: Sierra, C., Castelfranchi, C., Decker, K.S., Sichman, J.S. (eds.) 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), pp. 843–850 (2009)
33. Venkatraman, M., Singh, M.P.: Verifying compliance with commitment protocols. Autonomous Agents and Multi-Agent Systems 2(3), 217–236 (1999)
34. Desai, N., Chopra, A.K., Singh, M.P.: Representing and reasoning about commitments in business processes. In: Holte, R.C., Howe, A. (eds.) 22nd AAAI Conference on Artificial Intelligence (AAAI 2007), pp. 1328–1333. AAAI Press, Menlo Park (2007)
35. Chopra, A.K., Singh, M.P.: Producing compliant interactions: Conformance, coverage, and interoperability. In: Baldoni, M., Endriss, U. (eds.) DALT 2006. LNCS (LNAI), vol. 4327, pp. 1–15. Springer, Heidelberg (2006)
36. Chesani, F., Mello, P., Montali, M., Torroni, P.: Commitment tracking via the reactive event calculus. In: Boutilier, C. (ed.) 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 91–96 (2009)
37. Reiter, R.: Sequential, temporal GOLOG. In: Cohn, A., Schubert, L. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR 1998), pp. 547–556. Morgan Kaufmann, San Francisco (1998)

Probabilistic Action and Deontic Logic
(Invited Talk)

Jan Broersen

Department of Information and Computing Sciences, Utrecht University,
The Netherlands

Deontic logic aims at formally modeling the reasoning with norm-related modalities. Von Wright recognized that the obligation modality bears a resemblance to modal necessity and the permission modality to modal possibility, which resulted in his Standard Deontic Logic [9]. SDL is the modal logic KD. Since its conception, SDL has drawn a lot of criticism: e.g., Chisholm [2] argued that it was unfit to represent certain types of conditional obligation, and Makinson [5] and van der Torre [6] emphasized that deontic logic is much more naturally studied as a process of iterative detachment relative to explicitly represented normative systems; a view that does not fit well with the modal logic modeling proposed by Von Wright.

One way in which SDL, but also most other systems of deontic logic, falls short is the treatment of action with uncertain, probabilistic or attempted effects. Yet it is very natural, and sometimes even crucial for rational decision making, to reason, for instance, about being forbidden to take certain risks, being obliged to try something, or to avoid being liable for an attempted crime (see [10] for an excellent philosophical discussion of the role of attempt in criminal law). In our view, the modalities involved in these cases can all be viewed as applying to probabilistic action, that is, action with a certain chance of success. In particular, we suggest to model attempts as actions maximizing subjective probabilities of action success [1]. This modeling of attempt is quite different from other approaches in the literature. Placek [7] aims to define attempt entirely in terms of objective, non-mental modalities. Vanderveken [8] does take a subjective stance, but does not use probabilities or any other means to represent subjective epistemic attitudes. Finally, Herzig and Lorini [4] see attempt as an internal mental action preceding the objective external action. The first question we will discuss is how the obligation to attempt an action can be suitably modeled in a probabilistic *stit* framework extended with deontic modalities.

Recently Harel en Porat [3] put forward the issue of what we might rephrase as 'the probabilistic aggregation of blame'. They consider the question whether or not an agent who committed criminal action p with probability 0.9 and criminal action q also with probability 0.9, should be considered as having done wrong at least either p or q with a probability of 0.99. For instance, if the certainty threshold for convicting the agent of a crime is 0.95, then we could say that although we cannot convict the agent for either seeing to it that p or seeing to it that q, we can convict the agent for the non-deterministic action of seeing to

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 293–294, 2011.

it that $p \vee q$[1]. The laws in our legal systems do not allow for an accumulation of blame in this way. The second question we address is whether or not the semantics of deontic modalities like obligation and prohibition is vulnerable to similar effects in case they pertain to probabilistic action.

References

1. Broersen, J.: Modeling attempt and action failure in probabilistic stit logic. In: Proceedings of Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI 2011 (2011)
2. Chisholm, R.M.: Contrary-to-duty imperatives and deontic logic. Analysis 24, 33–36 (1963)
3. Harel, A., Porat, A.: Aggregating probabilities across cases: Responsibility for unspecified offences. Minnesota Law Review 94, 261–310 (2009)
4. Lorini, E., Herzig, A.: A logic of intention and attempt. Synthese 163(1), 45–77 (2008)
5. 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. 29–53. IOS Press, Amsterdam (1998)
6. Makinson, D., van der Torre, L.W.N.: Input-output logics. Journal of Philosphical Logic 29, 383–408 (2000)
7. Placek, T.: On attempting. In: Logica Yearbook 2009, pp. 155–161. College Publications, London (2010)
8. Vanderveken, D.: Attempt, success and action generation: A logical study of intentional action. In: Vanderveken, D. (ed.) Logic, Thought and Action, pp. 316–342. Springer, Heidelberg (2005)
9. von Wright, G.H.: Deontic logic. Mind 60, 1–15 (1951)
10. Yaffe, G.: Attempts. Oxford University Press, Oxford (2010)

[1] We rephrase Harel en Porat's problem here in terms of these *stit* sentences; Harel and Porat do not speak of non-deterministic actions but of 'unspecified offences'.

A Dynamic Logic of Institutional Actions

Andreas Herzig[1], Emiliano Lorini[1], and Nicolas Troquard[2]

[1] University of Toulouse, CNRS, IRIT, France
[2] Laboratory of Applied Ontology, ISTC-CNR, Italy

Abstract. We propose a logical framework to represent and reason about some important aspects of a theory of institutional action: (1) the distinctions between physical facts and actions and institutional facts and actions; (2) the distinction between causality and 'counts-as'; (3) the notion of institutional power. Technically, our contribution consists in extending a dynamic logic of propositional assignments with constructions allowing to express that an agent plays a given role; that a physical action causes another physical action; that a physical action performed by an agent playing a given role counts as an institutional action.

1 Introduction

We present a logical framework in which we can represent and reason about some important aspects of a theory of institutional action: (1) the distinctions between physical facts and actions and institutional facts and actions; (2) the distinction between causality and 'counts-as'; (3) the notion of institutional power. Our framework is that of a dynamic logic of propositional assignments in the sense of [32,33,6,2]. In preceding work [14] we have shown that this logic allows to reason about agent capabilities in the sense of coalition logic [26] and coalition logic of propositional control [16,15]. We here refine our account by distinguishing 'brute', physical facts from institutional facts. This leads us to the distinction between brute actions (changing brute facts) from institutional actions (changing institutional facts). We moreover add constructions allowing to express that an agent plays a given role; that a physical action causes another physical action (e.g. Jack's action of shooting Joe causes Jack's action of killing Joe); that a physical action performed by an agent playing a given role counts as an institutional action (e.g. an agent's act of performing certain gestures during the wedding ceremony while playing the role of priest counts as the act of marrying the couple). This provides a full-blown account of normative systems.

The paper is organized as follows. Section 2 establishes the conceptual basis of the logical analysis developed in the rest of the paper, providing a detailed discussion of the philosophical theory of institutional action developed by Goldman, Searle and other scholars, and it explains how the logic presented in the paper takes into account its different dimensions and aspects. Section 3 presents the syntax and the semantics of the logic, while Section 4 provides a complete axiomatization as well as a complexity result for the satisfiability problem. In Section 5 the logic is exploited to formalize the concept of institutional power. In Section 6 we discuss related works in the area of logic of normative systems. We finally conclude in Section 7 by discussing some perspectives for future work.

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 295–311, 2011.

2 Institutional Actions: Conceptual Analysis

Some background and clarifications of the notion of institutional action are needed in order to ground the logical analysis presented in the rest of the paper on a solid conceptual basis.

Physical facts and actions vs. institutional facts and actions. According to several authors working in legal theory and in the field of normative multiagent systems (MAS) (see e.g. [1,3]), normative systems are based both on regulative as well as constitutive (i.e., non-regulative) components. That is, normative systems are not only defined in terms of sets of permissions, obligations, and prohibitions (i.e. *norms of conduct*) but also in terms of rules which specify and create new forms of behavior and concepts. According to Searle for instance "[...] regulative rules regulate antecedently or independently existing forms of behavior [...]. But constitutive rules do not merely regulate, they create or define new forms of behavior" [29, p. 33]. In Searle's theory [29,30], constitutive rules are expressed by means of 'counts-as' assertions of the form "X counts as Y in context C" where the context C refers to the normative system in which the rule is specified. Constitutive rules relate "brute" physical facts and actions with institutional facts and actions. For example, in the context of the US federal reserve, receiving a piece of paper with a certain shape, color, etc. (a physical action) counts as receiving a certain amount of money (an institutional action); or in the context of Catholic Church the priest's action of performing certain gestures during the wedding ceremony (which is a physical action) counts as the act of marrying the couple (which is an institutional action). Although Searle's counts-as relation is between objects in general (such as a piece of paper counting as an amount of money), in this work we only consider the counts-as' relation between actions.

As pointed out by [17], the counts-as relation may also relate two institutional actions. For example, in the context of chess, the action of checkmating the opponent (an institutional action) counts as the action of winning the game (an institutional action).

Causality vs. counts-as. In his seminal work on the philosophical theory of action, Goldman studied a fundamental relation between actions and events of the form "action α is done *by* doing a different action β" [8]. The word "by" expresses a relation between actions which Goldman calls *generation*. This means that an action can be performed by way of one or more actions. According to Goldman's theory, there are actions which have a deep recursive structure. In fact, there could be an action α done by doing an action β which in turn is done by doing a further action γ and so on. Such a decomposition of an agent's action α stops at the level of basic actions. Basic actions therefore represent the agent's only direct intervention in the process of doing α. As Davidson puts it, "the rest is up to nature" [5].[1] By way of example, consider Jack's action of killing Joe. Jack kills Joe by shooting him and Jacks shoots Joe by pulling the trigger of the gun. Jack's bodily movement of pulling the trigger (which consists in Jack's moving his forefinger in a certain way) is a basic action, as it is the only part of the action of killing Joe which is directly controlled by Jack.

[1] See [19] for a formal analysis of basic actions in dynamic logic.

Goldman's theory opposes "causal generation" to "conventional generation". The latter can be identified with Searle's counts-as relation. According to Goldman, physical actions are causally generated, that is, they just consist in an agent bringing about (i.e. causing) a certain state of affairs to be true. On the other hand, institutional actions are conventionally generated, by which he meant that actions such as signaling before making a turn, and checkmating one's opponent, exist in virtue of rules or conventions relating physical actions with institutional effects. For example, in the sentence "a player wins a chess game against his opponent by checkmating him" the word "by" expresses a relation of conventional generation, that is, the action of checkmating the opponent *counts-as* the action of winning the chess game. On the contrary, in the sentence "Jack kills Joe by shooting him" the word "by" expresses a relation of causal generation, that is, Jack's action of shooting Joe *causes* the action of killing Joe (i.e. the action of making Joe dead). To carry the example further, the action of killing Joe might conventionally generate the action of murdering Joe (which is an institutional action).

We here explore Goldman's view. We assume that the causal relation and the counts-as' relation between actions are ontologically different for at least two reasons. While the former relates a physical action to another physical action, the latter relates a physical action to an institutional action, or an institutional action to another institutional action. Moreover, while the causal relation is merely a relation between physical actions performed by an agent, counts-as' is a relation between actions performed by an agent playing a certain role in a given institutional context. For example, in the institutional context of Catholic Church, an agent's act of performing certain gestures during the wedding ceremony while playing the role of priest counts as the act of marrying the couple. As the next paragraph highlights, this aspect of counts-as is fundamental to understand the notion of institutional power.

Institutional power. Some legal and social theorists [4,29,12] as well as some logicians [17,20,22] have emphasized the tight relationship between counts-as and the notion of institutional power. According to these authors, there exists a specific kind of norms called *norms of competence* whose function in a legal system is to assign institutional powers to the agents playing certain roles within a given institution.[2] Such power-conferring norms should not be reduced to norms of conduct such as obligations, prohibitions, commands and permissions. On the contrary, they are expressed by means of counts-as assertions relating physical or institutional actions to institutional actions. They have a fundamental function in normative and legal systems since they provide the criteria for institutional change, that is, they provide the criteria for the creation and modification of institutional facts (e.g. agent i and agent j are married, this house is i's property, etc.). In other words, according to these authors, saying that "an agent playing the role r has the institutional power to do the institutional action α *by* doing action β" just means that "an agent's performance of action β while playing the role r counts as the agent's performance of the institutional action α". For example, "an agent playing the role of priest has the institutional power to marry a couple *by* performing certain

[2] From this perspective, a given role can be identified with the set of norms of competence concerning it. This view is compatible with [27] in which a role is defined as a set (or cluster) of norms.

gestures during the wedding ceremony" just means that "an agent's act of performing certain gestures during the wedding ceremony while playing the role of priest counts as the act of marrying the couple".

The interesting aspect of this notion of institutional power is that it allows to properly understand how (human or software) agents, conceived as physical entities, can produce institutional effects by way of performing physical actions and by playing certain social roles. To summarize, the crucial point is the following. A given agent i has the ability to perform a certain institutional action α by way of performing another action β because: (1) the agent plays a certain social role r; (2) there is a norm of competence establishing that the performance of action α by an agent playing the role r counts as the performance of the institutional action β. For example, an agent i has the ability of marrying a couple by performing certain gestures during a wedding ceremony because: (1) i plays the role of priest; (2) there is a norm of competence establishing that an agent's physical act of performing certain gestures during the wedding ceremony while playing the role of *priest* counts as the institutional act of marrying the couple.

Remarks on the nature of roles and brute abilities. In the framework presented in Section 3, the world the agents populate will be a mere database listing the atomic facts that are true at this very moment. The world is dynamic because the agents can act upon their environment by changing the truth value of these atomic facts. Our proposal relies on the assumption that an agent's brute abilities can be identified with the set of propositional assignments that he can perform. As shown in [14], such a framework also supports the models of *propositional control* [16,7].

Roles are central in our study of institutional abilities. It is by occupying roles that an agent's brute action generates an institutional event. However, we do not need at this stage to ground our proposal on a rigorous ontology of roles. We refer to [23] for a foundational ontology of roles and a detailed interdisciplinary review of the literature.

The rest of the paper. We start by splitting the set of propositional variables into two disjoint sets: atomic physical facts and atomic institutional facts. In the logic presented in Section 3, a physical action just consists in setting to 'true' or to 'false' some atomic physical fact, while an institutional action consists in setting to 'true' or to 'false' some atomic institutional fact. The latter actions can only be performed indirectly, i.e. by performing a physical action. Moreover, we distinguish two different relations between actions: "counts-as" and "causes". There are conditionals of the form $\alpha_1 \Rightarrow \alpha_2$ expressing that i's performance of the physical action α_1 causes the performance of the physical action α_2, and there are conditionals of the form $\alpha_1 \overset{r}{\rightsquigarrow} \alpha_2$ expressing that i's performance of the physical or institutional action α_1 in the social role r counts as the performance of the institutional action α_2. Finally, in Section 5, we study the notion of institutional power by introducing special modal operators describing an agent's capability of producing a given institutional effect by way of performing a physical action while playing a given social role.

For the sake of simplicity we suppose that there is only one institution.

3 Logic

This section introduces the syntax and the semantics of the logic. It is basically an extension of our logic of [14] by a causality relation.

3.1 Language

We suppose that there is a finite set of *agents* $\mathbb{A}$, a finite set of *roles* $\mathbb{R}$, and a countable set of propositional variables.

Propositional variables are meant to capture the atomic facts about the world. They are partitioned into two kinds: *facts about the physical world (or brute facts)* and *facts about the institutional world (or institutional facts)* and are collected in two sets $\mathbb{P}_{\text{phys}}$ and $\mathbb{P}_{\text{inst}}$. These sets form a partition, and we thus assume that $\mathbb{P} = \mathbb{P}_{\text{phys}} \cup \mathbb{P}_{\text{inst}}$ and $\mathbb{P}_{\text{phys}} \cap \mathbb{P}_{\text{inst}} = \emptyset$. Our propositional variables are therefore typed.

These sets are again composed of variables of different sub-types: we suppose that they respectively contain a countable set $\mathbb{P}^0_{\text{phys}} \subseteq \mathbb{P}_{\text{phys}}$ of *basic physical facts*; and a countable set $\mathbb{P}^0_{\text{inst}} \subseteq \mathbb{P}_{\text{inst}}$ of *basic institutional facts*. Beyond these basic variables the set $\mathbb{P}_{\text{inst}}$ contains variables denoting that an agent plays a role and the set $\mathbb{P}_{\text{phys}}$ contains variables denoting that an agent is able to assign a variable to true or false. We suppose that the latter covers the case of ability variables being themselves assigned; we will therefore need a recursive definition.

In the sequel we are going to analyze in more detail what kinds of propositions actually populate $\mathbb{P}$.

Institutional facts. We have assumed the existence of a finite set of roles $\mathbb{R}$. These roles are occupied by agents. We write $R_i(r)$ to formalize the fact that agent $i \in \mathbb{A}$ occupies the role $r \in \mathbb{R}$. Holding a role is a societal construct, and an atomic institutional fact. It is also a contingent fact (or anti-rigid [23]) meaning that role occupations can change. Hence, we assume that expressions of the form $R_i(r)$ are propositional variables in $\mathbb{P}_{\text{inst}}$. These are the only 'special' atomic institutional facts in $\mathbb{P}_{\text{inst}}$ and we therefore have $\mathbb{P}_{\text{inst}} = \mathbb{P}^0_{\text{inst}} \cup \{R_i(r) \ : \ r \in \mathbb{R}, i \in \mathbb{A}\}$. We can write this in a BNF:

$$\mathbb{P}_{\text{inst}} : p_{\text{inst}} ::= p^0_{\text{inst}} \mid R_i(r)$$

where p^0_{inst} ranges over the set of basic institutional facts $\mathbb{P}^0_{\text{inst}}$, i ranges over the set of agents $\mathbb{A}$, and r ranges over the set of roles $\mathbb{R}$. Then the language $\mathcal{L}_{\text{inst}}$ of (complex) institutional facts is defined by the following grammar:

$$\mathcal{L}_{\text{inst}} : \varphi_{\text{inst}} ::= p_{\text{inst}} \mid \neg\varphi_{\text{inst}} \mid \varphi_{\text{inst}} \vee \varphi_{\text{inst}}$$

where p_{inst} ranges over the set of atomic institutional facts $\mathbb{P}_{\text{inst}}$.

Assignments. Assignments are expressions of the form $p{\leftarrow}\top$ or $p{\leftarrow}\bot$, where p is a propositional variable from $\mathbb{P}$. Assignments and formulas are different entities: the former are *events* modifying the truth values of propositional variables. The event $p{\leftarrow}\top$ sets p to true, and the event $p{\leftarrow}\bot$ sets p to false. We sometimes write $p{\leftarrow}\tau$ in order to

talk about $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ in an economic way; τ is therefore a placeholder for either $\top$ or $\bot$.

The sets

$$\mathcal{A}_{\mathsf{phys}} = \{p{\leftarrow}\tau \ : \ p \in \mathbb{P}_{\mathsf{phys}}, \tau \in \{\top, \bot\}\}$$
$$\mathcal{A}_{\mathsf{inst}} = \{p{\leftarrow}\tau \ : \ p \in \mathbb{P}_{\mathsf{inst}}, \tau \in \{\top, \bot\}\}$$

respectively collect the assignments of brute facts and the assignments of institutional facts. Observe that $\mathcal{A}_{\mathsf{phys}} \cap \mathcal{A}_{\mathsf{inst}} = \emptyset$ because $\mathbb{P}_{\mathsf{phys}} \cap \mathbb{P}_{\mathsf{inst}} = \emptyset$. The set of all assignments is $\mathcal{A} = \mathcal{A}_{\mathsf{phys}} \cup \mathcal{A}_{\mathsf{inst}}$. We write $\alpha_{\mathsf{phys}}, \alpha'_{\mathsf{phys}}, \ldots$ to denote assignments from $\mathcal{A}_{\mathsf{phys}}$ and $\alpha_{\mathsf{inst}}, \alpha'_{\mathsf{inst}}, \ldots$ to denote assignments from $\mathcal{A}_{\mathsf{inst}}$. We sometimes use $\alpha, \alpha', \ldots$ to denote generic assignments from $\mathcal{A}$.

Physical facts. We assume that an agent can act upon his environment by assigning values to some propositional variables. We also assume that these variables can only be the atomic physical facts in $\mathbb{P}_{\mathsf{phys}}$, while the values of the atomic institutional facts of $\mathbb{P}_{\mathsf{inst}}$ can only be modified indirectly.

For a physical assignment event $\alpha \in \mathcal{A}_{\mathsf{phys}}$ and an agent $i \in \mathbb{A}$, we formalize that i has the *physical ability* to perform α by writing $\mathrm{A}_i(\alpha)$. We take that the physical ability of an agent to perform an action is itself an atomic physical fact. Moreover, we assume that agent's physical abilities are contingent facts. Hence, we assume that expressions of the form $\mathrm{A}_i(\alpha)$ are propositional variables in $\mathbb{P}_{\mathsf{phys}}$.

We allow this to be recursive: for every ability variable $\mathrm{A}_i(\alpha)$ there is an ability variable $\mathrm{A}_j(\mathrm{A}_i(\alpha){\leftarrow}\top)$ that is also an atomic physical fact.

We suppose that the set of atomic physical facts $\mathbb{P}_{\mathsf{phys}}$ is made up of the basic physical facts of $\mathbb{P}^0_{\mathsf{phys}}$ plus all these ability variables, and nothing else. This set is therefore built according to the following grammar:

$$\mathbb{P}_{\mathsf{phys}} : p_{\mathsf{phys}} ::= p^0_{\mathsf{phys}} \mid \mathrm{A}_i(p_{\mathsf{phys}}{\leftarrow}\tau)$$

where p^0_{phys} ranges over the set of basic brute facts $\mathbb{P}^0_{\mathsf{phys}}$ and τ ranges over the set $\{\top, \bot\}$. Then the language $\mathcal{L}_{\mathsf{phys}}$ of (complex) physical facts is defined by the following grammar:

$$\mathcal{L}_{\mathsf{phys}} : \varphi_{\mathsf{phys}} ::= p_{\mathsf{phys}} \mid \neg\varphi_{\mathsf{phys}} \mid \varphi_{\mathsf{phys}} \vee \varphi_{\mathsf{phys}}$$

where p_{phys} ranges over the set of atomic brute facts $\mathbb{P}_{\mathsf{phys}}$.

'Causes'. We have a binary connective $\Rightarrow$ relating assignments of physical facts: if α_1 and α_2 are both assignments in $\mathcal{A}_{\mathsf{phys}}$ then $\alpha_1 \Rightarrow \alpha_2$ expresses that the performance of α_1 triggers α_2. For example, $p{\leftarrow}\top \Rightarrow q{\leftarrow}\bot$ says that the atomic physical fact q is made false by making the atomic physical fact p true.

'Counts-as'. We have a ternary connective $\rightsquigarrow$ whose arguments are a role, an atomic fact, and an atomic institutional fact, written $\alpha_1 \overset{r}{\rightsquigarrow} \alpha_2$. For instance, we write $p{\leftarrow}\top \overset{r}{\rightsquigarrow} q{\leftarrow}\bot$ to formalize the fact that setting the (physical or institutional) fact p to true while acting in role r counts as setting the institutional fact q to false.

Achieving by doing. Actions are physical events performed by an agent. The formula $\langle i{:}p_{\mathsf{phys}}{\leftarrow}\tau\rangle\varphi$ reads "i can achieve φ by performing $p_{\mathsf{phys}}{\leftarrow}\tau$". By convention, we adopt a strong reading of 'can' and assume that if i does not have the ability to perform $p_{\mathsf{phys}}{\leftarrow}\tau$, then i cannot achieve anything by performing $p_{\mathsf{phys}}{\leftarrow}\tau$.

Definition of the language. The language $\mathcal{L}$ of the logic is fully defined by the following grammar:

$$\begin{aligned}\mathcal{L} : \varphi ::= {} & p_{\mathsf{phys}} \mid p_{\mathsf{inst}} \mid \alpha_{\mathsf{phys}} \Rightarrow \alpha_{\mathsf{phys}} \mid \alpha_{\mathsf{phys}} \overset{r}{\rightsquigarrow} \alpha_{\mathsf{inst}} \mid \alpha_{\mathsf{inst}} \overset{r}{\rightsquigarrow} \alpha_{\mathsf{inst}} \mid \\ & \neg\varphi \mid \varphi \vee \varphi \mid \langle i{:}\alpha_{\mathsf{phys}}\rangle\varphi\end{aligned}$$

where p_{phys} and p_{inst} respectively range over the set of brute facts $\mathbb{P}_{\mathsf{phys}}$ and the set of institutional facts $\mathbb{P}_{\mathsf{inst}}$; α_{phys} and α_{inst} respectively range over the set of assignments of physical facts $\mathcal{A}_{\mathsf{phys}}$ and the set of assignments of institutional facts $\mathcal{A}_{\mathsf{inst}}$; r ranges over the set of roles $\mathbb{R}$; and i ranges over the set of agents $\mathbb{A}$.

The logical constants $\top$ and $\bot$ and the logical connectives $\wedge$, $\rightarrow$ and $\leftrightarrow$ have the usual meaning.

Remarks. A few remarks about our choices of language are in order. As our assignments only operate on $\mathbb{P}$, the truth values of $\alpha \Rightarrow \beta$ and $\alpha \overset{r}{\rightsquigarrow} \beta$ cannot be re-assigned. Assignments being the only means to change things in our logic, it follows that the 'causes' and the 'counts-as' relation do not evolve. We are aware that assuming that causality and counts-as relation are rigid facts of the world is a limitation of our formal theory. For instance, that flipping a switch causes the light to go on can be changed by an action of disconnecting the electric wires; and the counts-as relation in an institution can be changed by way of new agreements, laws, etc.

In contrast, we have modeled the ability relation and the role-playing relation by means of propositional variables $\mathrm{A}_i(\alpha)$ and $R_i(r)$, and these variables are elements of $\mathbb{P}$. Both are therefore contingent facts of the world, just like the physical fact that "the pen is on the table" [23], and their truth values can change due to the performance of assignments.

Let us have a closer look at the way these changes are brought about. First, agent i's ability to perform α being a (non-basic) physical fact of $\mathbb{P}_{\mathsf{phys}}$, that fact can be modified by an assignment event that is performed by some agent j; precisely, some agent j for which $\mathrm{A}_j(\mathrm{A}_i(\alpha){\leftarrow}\top)$ or $\mathrm{A}_j(\mathrm{A}_i(\alpha){\leftarrow}\bot)$ holds. The fact $\mathrm{A}_i(\alpha)$ can also be modified indirectly by the causality relation. For instance, my action of grabbing John's arm causes the loss of John's ability to raise his arm. Second, an agent playing a role being an institutional fact of $\mathbb{P}_{\mathsf{inst}}$, that fact cannot be modified directly by an agent's action: just as any institutional fact, it can only change via the application of the counts-as relation. This allows a very strict control over the institutional consequences of an event.

3.2 Models

A model is a tuple

$$M = (\mathbb{A}, \mathbb{R}, \mathbb{P}^0_{\mathsf{phys}}, \mathbb{P}^0_{\mathsf{inst}}, V_{\mathsf{phys}}, V_{\mathsf{inst}}, C_{\mathsf{phys}}, C_{\mathsf{inst}})$$

where the sets $\mathbb{A}$, $\mathbb{R}$, $\mathbb{P}^0_{\mathsf{phys}}$ and $\mathbb{P}^0_{\mathsf{inst}}$ are as detailed above; $V_{\mathsf{phys}} \subseteq \mathbb{P}^0_{\mathsf{phys}}$ and $V_{\mathsf{inst}} \subseteq \mathbb{P}^0_{\mathsf{inst}}$ are valuations describing which atomic fact are true; $C_{\mathsf{phys}} \subseteq \mathcal{A}_{\mathsf{phys}} \times \mathcal{A}_{\mathsf{phys}}$ is a relation between assignments of physical facts; and $C_{\mathsf{inst}} : \mathbb{R} \longrightarrow 2^{\mathcal{A}\times\mathcal{A}_{\mathsf{inst}}}$ is a function mapping roles to relations between assignments and institutional assignments. When $(\alpha_1, \alpha_2) \in C_{\mathsf{phys}}$ then the occurrence of action α_1 causes the occurrence of action α_2. When $(\alpha_1, \alpha_2) \in C_{\mathsf{inst}}(r)$ then the occurrence of action α_1 performed by an agent playing role r counts as the occurrence of action α_2. The relations C_{phys} and $C_{\mathsf{inst}}(r)$ are nothing but the relations of "causal generation" and "conventional generation" in Goldman's sense as described in Section 2.

3.3 Constraints on Models

Models have to satisfy the following two constraints:

($Refl_{\mathsf{phys}}$) C_{phys} is reflexive:
for every $\alpha \in \mathcal{A}_{\mathsf{phys}}$, $(\alpha, \alpha) \in C_{\mathsf{phys}}$.

(Coh_{phys}) C_{phys} is coherent:
for every $\alpha \in \mathcal{A}_{\mathsf{phys}}$ and $q \in \mathbb{P}$, if $(\alpha, q{\leftarrow}\top) \in C_{\mathsf{phys}}$ then $(\alpha, q{\leftarrow}\bot) \notin C_{\mathsf{phys}}$.

($Trans_{\mathsf{phys}}$) C_{phys} is transitive:
$C_{\mathsf{phys}} \circ C_{\mathsf{phys}} \subseteq C_{\mathsf{phys}}$

(Coh_{inst}) C_{inst} is coherent:
for every $\alpha \in \mathcal{A}$, $q \in \mathbb{P}$, and $r_1, r_2 \in \mathbb{R}$,
if $(\alpha, q{\leftarrow}\top) \in C_{\mathsf{inst}}(r_1)$ then $(\alpha, q{\leftarrow}\bot) \notin C_{\mathsf{inst}}(r_2)$.

($Trans_{\mathsf{phys,inst}}$) C_{phys} and C_{inst} satisfy a mixed transitivity property:
for every $r \in \mathbb{R}$, $C_{\mathsf{phys}} \circ C_{\mathsf{inst}}(r) \subseteq C_{\mathsf{inst}}(r)$.

In the rest of the section we briefly discuss these properties in the light of our exposition in Section 2. (See also [28] for a review of properties of causality relations in the framework of artificial intelligence.)

The constraint ($Refl_{\mathsf{phys}}$) means that we consider that causality is reflexive. We are aware that this can be criticized because causes temporally precede their effects. It however simplifies the technicalities when updating models; the reader may wish to think of it as the reflexive closure of the causality relation. (Coh_{phys}) says that a physical action cannot have inconsistent causal ramifications. (Coh_{inst}) is a similar principle for the counts-as relation: for every assignment α there cannot be two roles leading to inconsistent consequences via the counts-as relations. Constraint ($Trans_{\mathsf{phys,inst}}$) is the institutional counterpart of the transitivity of causality as expressed by ($Trans_{\mathsf{phys}}$).

Reflexivity of event generation relations is rejected by Goldman [8, p. 5] on the simple ground that it is not intuitive to say that John turns on the light by turning on the light. In our proposal, the counts-as relation is not necessarily reflexive; we however allow that $\alpha \overset{r}{\rightsquigarrow} \alpha$ if α is an institutional action. Moreover, our modeling of the causality

relation assumes reflexivity. This is essentially motivated by the fact that this way, our definitions are less cluttered, but since Goldman's argument against reflexivity is merely linguistic, we believe it is not a major conceptual transgression.

Goldman insists that an event generation relation should be antisymmetric. We neither preclude the symmetry of the causality relation nor of the counts-as relation since we could have for instance that a subset of events form an equivalence class in which all events causally generate all events.

It is worth noting that there is some disagreement in the literature whether the counts-as relation should satisfy transitivity. For a discussion on this matter see [17,11,21]. In our logic this is not necessarily the case.

Finally, some author argues that the counts-as should satisfy contraposition [11], while other authors have a different opinion on this matter [17]. Again, we remain uncommitted w.r.t. this point, and it may be the case that $(p{\leftarrow}\top, q{\leftarrow}\top) \in C_{\mathsf{inst}}(r)$ while $(q{\leftarrow}\bot, p{\leftarrow}\bot) \notin C_{\mathsf{inst}}(r)$.

3.4 Updating a Model by an Action

An agent's capability can be represented semantically by the valuations V' his actions can bring about. This is achieved by interpreting the agents' actions as model updates.

Definition 1. *Let*

$$M = (\mathbb{A}, \mathbb{R}, \mathbb{P}^0_{\mathsf{phys}}, \mathbb{P}^0_{\mathsf{inst}}, V_{\mathsf{phys}}, V_{\mathsf{inst}}, C_{\mathsf{phys}}, C_{\mathsf{inst}})$$

be a model and let $\alpha \in \mathcal{A}_{\mathsf{phys}}$. The update of M by the action $i{:}\alpha$ is defined as

$$M^{i:\alpha} = (\mathbb{A}, \mathbb{R}, \mathbb{P}^0_{\mathsf{phys}}, \mathbb{P}^0_{\mathsf{inst}}, V^{\alpha}_{\mathsf{phys}}, V^{i:\alpha}_{\mathsf{inst}}, C_{\mathsf{phys}}, C_{\mathsf{inst}})$$

where the updates $V^{i:\alpha}_{\mathsf{phys}}$ and $V^{i:\alpha}_{\mathsf{inst}}$ of the valuations V_{phys} and V_{inst} by $i{:}\alpha$ are defined as follows:

$$\begin{aligned} V^{\alpha}_{\mathsf{phys}} &= \Big(V_{\mathsf{phys}} \setminus \{q \;:\; (\alpha, q{\leftarrow}\bot) \in C_{\mathsf{phys}}\} \Big) \cup \{q \;:\; (\alpha, q{\leftarrow}\top) \in C_{\mathsf{phys}}\} \\ V^{i:\alpha}_{\mathsf{inst}} &= \big(V_{\mathsf{inst}} \setminus \{q \;:\; \exists r \in \mathbb{R} : R_i(r) \in V_{\mathsf{inst}}, (\alpha, q{\leftarrow}\bot) \in C_{\mathsf{inst}}(r)\} \big) \\ &\qquad \cup \{q \;:\; \exists r \in \mathbb{R} : R_i(r) \in V_{\mathsf{inst}}, (\alpha, q{\leftarrow}\top) \in C_{\mathsf{inst}}(r)\} \end{aligned}$$

Actions therefore (1) directly affect the physical world (via the causality relation), and (2) affect the institutional world via the counts-as relation.[3] Suppose that $\alpha = p{\leftarrow}\top$. Then, due to reflexivity of the causality relation C_{phys}, the valuation $V^{p\leftarrow\top}_{\mathsf{phys}}$ contains p and $V^{p\leftarrow\bot}_{\mathsf{phys}}$ does not contain p. Note that the physical valuation is actually updated by the event α, not by the action $i{:}\alpha$.

Our constraints on models are clearly preserved under updates because neither the causal relation nor the counts-as relation can be modified.

[3] Note that the order of the set theoretic operations in the definition seems to privilege positive facts; however, due to our two constraints (Coh_{phys}) and (Coh_{inst}) —and also because $\mathbb{P}_{\mathsf{phys}}$ and $\mathbb{P}_{\mathsf{inst}}$ have empty intersection— the ramifications of an assignment of a physical fact will never conflict.

Let us illustrate our definition by a couple of examples.

Example 1. Suppose V_{inst} contains $R_i(r_1)$ and $p{\leftarrow}\top \overset{r_1}{\rightsquigarrow} q{\leftarrow}\top$, i.e. agent i plays role r_1, and in role r_1 making p true counts as making q true. Then $V_{\mathsf{phys}}^{p\leftarrow\top}$ contains p, and $V_{\mathsf{inst}}^{i:p\leftarrow\top}$ contains q. Hence, under the hypothesis that V_{phys} contains $\mathrm{A}_i(p{\leftarrow}\top)$ (that is that agent i is indeed able to make p true), agent i can achieve about $p \wedge q$ by doing $p{\leftarrow}\top$.

Example 2. Suppose V_{inst} contains $R_i(r_1)$ and $R_i(r_2)$, and $(p{\leftarrow}\top, q_1{\leftarrow}\top) \in C_{\mathsf{inst}}(r_1)$ and $(p{\leftarrow}\top, q_2{\leftarrow}\bot) \in C_{\mathsf{inst}}(r_2)$, i.e. agent i plays two roles r_1 and r_2, and in role r_1 this counts as making q_1 true, while in role r_2 this counts as making q_2 false. Then in $M^{i:p\leftarrow\top}$, the valuation $V_{\mathsf{phys}}^{p\leftarrow\top}$ contains p and $V_{\mathsf{inst}}^{i:p\leftarrow\top}$ contains q_1 and does not contain q_2. Hence, assuming that V_{phys} contains $\mathrm{A}_i(p{\leftarrow}\top)$ (that is agent i is indeed able to make p true), agent i can achieve $p \wedge q_1 \wedge \neg q_2$ by doing $p{\leftarrow}\top$.

3.5 Truth Conditions

Let

$$M = (\mathbb{A}, \mathbb{R}, \mathbb{P}^0_{\mathsf{phys}}, \mathbb{P}^0_{\mathsf{inst}}, V_{\mathsf{phys}}, V_{\mathsf{inst}}, C_{\mathsf{phys}}, C_{\mathsf{inst}})$$

be a model. The truth conditions are as usual for the Boolean operators, and we only state those clauses that are not standard.

$$\begin{array}{ll}
M \models p_{\mathsf{phys}} & \text{iff } p_{\mathsf{phys}} \in V_{\mathsf{phys}} \\
M \models p_{\mathsf{inst}} & \text{iff } p_{\mathsf{inst}} \in V_{\mathsf{inst}} \\
M \models \alpha_1 \Rightarrow \alpha_2 & \text{iff } (\alpha_1, \alpha_2) \in C_{\mathsf{phys}} \\
M \models \alpha_1 \overset{r}{\rightsquigarrow} \alpha_2 & \text{iff } (\alpha_1, \alpha_2) \in C_{\mathsf{inst}}(r) \\
M \models \langle i{:}\alpha_{\mathsf{phys}}\rangle\varphi & \text{iff } \mathrm{A}_i(\alpha_{\mathsf{phys}}) \in V_{\mathsf{phys}} \text{ and } M^{i:\alpha_{\mathsf{phys}}} \models \varphi
\end{array}$$

The operator $\langle i{:}\alpha_{\mathsf{phys}}\rangle\varphi$ captures the notion of "achieving by doing" that has been sketched in Section 3.1 and in the two examples of the last sub-section.

4 Axiomatization and Complexity

The logic is axiomatized as an extension of classical propositional logic with (1) a theory describing the constraints imposed on the counts-as and causality relations, (2) the reduction axioms of the dynamic operator, and (3) an inference rule of replacement of equivalents in the scope of a dynamic operator.

Theory of counts-as and causality.

$$\begin{array}{l}
\alpha \Rightarrow \alpha \\
(\alpha \Rightarrow p{\leftarrow}\bot) \rightarrow \neg(\alpha \Rightarrow p{\leftarrow}\top) \\
(\alpha \overset{r_1}{\rightsquigarrow} p{\leftarrow}\top) \rightarrow \neg(\alpha \overset{r_2}{\rightsquigarrow} p{\leftarrow}\bot) \\
((\alpha_1 \Rightarrow \alpha_2) \wedge (\alpha_2 \Rightarrow \alpha_3)) \rightarrow (\alpha_1 \Rightarrow \alpha_3) \\
((\alpha_1 \Rightarrow \alpha_2) \wedge (\alpha_2 \overset{r_1}{\rightsquigarrow} \alpha_3)) \rightarrow (\alpha_1 \overset{r_1}{\rightsquigarrow} \alpha_3)
\end{array}$$

Reduction axioms for the dynamic operator.

$$
\begin{array}{ll}
\langle i{:}\alpha\rangle\top & \leftrightarrow \mathrm{A}_i(\alpha) \\
\langle i{:}\alpha\rangle\bot & \leftrightarrow \bot \\
\langle i{:}\alpha\rangle(\alpha_1 \Rightarrow \alpha_2) & \leftrightarrow \mathrm{A}_i(\alpha) \wedge (\alpha_1 \Rightarrow \alpha_2) \\
\langle i{:}\alpha\rangle(\alpha_1 \overset{r}{\rightsquigarrow} \alpha_2) & \leftrightarrow \mathrm{A}_i(\alpha) \wedge (\alpha_1 \overset{r}{\rightsquigarrow} \alpha_2) \\
\langle i{:}\alpha\rangle p_{\mathsf{phys}} & \leftrightarrow \mathrm{A}_i(\alpha) \wedge ((\alpha \Rightarrow p_{\mathsf{phys}} \leftarrow \top) \vee (p_{\mathsf{phys}} \wedge \neg(\alpha \Rightarrow p_{\mathsf{phys}} \leftarrow \bot))) \\
\langle i{:}\alpha\rangle p_{\mathsf{inst}} & \leftrightarrow \mathrm{A}_i(\alpha) \wedge (\bigvee_{r\in\mathbb{R}} (R_i(r) \wedge (\alpha \overset{r}{\rightsquigarrow} p_{\mathsf{inst}} \leftarrow \top)) \\
 & \qquad\qquad \vee (p_{\mathsf{inst}} \wedge \neg \bigvee_{r\in\mathbb{R}} (R_i(r) \wedge (\alpha \overset{r}{\rightsquigarrow} p_{\mathsf{inst}} \leftarrow \bot)))) \\
\langle i{:}\alpha\rangle\neg\varphi & \leftrightarrow \mathrm{A}_i(\alpha) \wedge \neg\langle i{:}\alpha\rangle\varphi \\
\langle i{:}\alpha\rangle(\varphi \vee \psi) & \leftrightarrow \mathrm{A}_i(\alpha) \wedge (\langle i{:}\alpha\rangle\varphi \vee \langle i{:}\alpha\rangle\psi)
\end{array}
$$

Inference rule.

$$\text{From } \varphi \leftrightarrow \psi \text{ infer } \langle i{:}\alpha\rangle\varphi \leftrightarrow \langle i{:}\alpha\rangle\psi$$

Proofs are defined in the standard way. For example, the rule of replacement of equivalents can be proved from our axiomatization (due to the inference rule).

Given a formula φ let $red(\varphi)$ be the formula obtained by iterating the application of the reduction axioms from the left to the right. Thanks to the rule of replacement of equivalents it is clear that $red(\varphi) \leftrightarrow \varphi$ is valid.

Proposition 1. *For every formula φ, $red(\varphi) \leftrightarrow \varphi$ is valid, and the length of $red(\varphi)$ is linear in the length of φ.*

Proposition 2. *Let φ be a formula without dynamic operators $\langle . \rangle$. φ is valid if and only if $\mathcal{T}_\varphi \rightarrow \varphi$ is valid in classical propositional logic, where $\mathcal{T}_\varphi$ is the conjunction of the axiom schemas of the theory of counts-as and causality instantiated by those assignments occurring in φ.*

As the length of $\mathcal{T}_\varphi$ is cubic in the length of φ we obtain a complexity result for our logic.

Corollary 1. *The problem of checking satisfiability of a formula is NP-complete.*

Our logic has therefore the same complexity as classical propositional logic. We however believe that it allows to express things in a more natural way. In the rest of the paper we give some arguments for this.

5 Institutional Power and Compact Characterization

Let S be a finite set of assignments. We identify the concept "agent i has the capability to achieve outcome φ by possibly performing one of the actions in the set S" with the formula

$$\Diamond_{i:S}\varphi \overset{\mathtt{def}}{=} \varphi \vee \bigvee_{\alpha\in S} \langle i{:}\alpha\rangle\varphi$$

If φ belongs to the language of institutional facts $\mathcal{L}_{\mathsf{inst}}$ then i's capability to achieve φ can be rightly called i's *institutional power* to achieve φ by doing actions from S.

We now illustrate our logic by adapting an example of water management from [14]. In that paper, beyond abilities $\mathrm{A}_i(\alpha)$ there are also atomic facts $\mathrm{P}_i(\alpha)$ whose intended meaning is that agent i is permitted to perform α. Clearly, it seems natural to assume that such atomic facts are institutional facts from $\mathbb{P}_{\mathsf{inst}}$. Note that any combination of abilities and permissions is consistent: an agent might be able to perform α but not permitted to do so, etc.

Example 3. There are two farmers i_1 and i_2 working in a certain area close to a town called Thirstytown who need water in order to irrigate their fields. In this area there are three different exploitable water basins 1, 2 and 3. Only water basins 1 and 2 can be safely used by the farmers; basin 3 provides drinkable water to the population of Thirstytown, and if it is exploited for irrigation then Thirstytown will fall short of water. There are two other actors in this scenario: agent i_3 plays the role of chief of the Water Authority which has the jurisdiction over the area, and agent i_4 is a local policeman working in Thirstytown. Let wAuth denote the role of head of water authority, and let pol denote the role of policeman. We consider that $R_{i_3}(\mathsf{wAuth})$ and $R_{i_4}(\mathsf{pol})$ are both true.

The propositional variables $\{p_1, p_2, p_3\}$ indicate whether the level of water in a given basin is high or low: p_1 means that "the level of water in the basin 1 is high", $\neg p_1$ means that "the level of water in the basin 1 is low", etc. Furthermore, for every farmer $i_k \in \{i_1, i_2\}$ and for every propositional variable p_h with $h \in \{1, 2, 3\}$, $\mathrm{A}_{i_k}(p_h \leftarrow \bot)$ expresses that basin h is physically connected to the field of farmer i_k so that i_k is able to exploit the water of basin h and $\mathrm{P}_{i_k}(p_h \leftarrow \bot)$ expresses that i_k is authorized to exploit the water of basin h.

Let $\mathsf{prohibSign}_h$ mean that there are prohibition-to-pump signs at basin h. We suppose that the the counts-as relation is such that $\mathsf{prohibSign}_h \leftarrow \top \overset{\mathsf{pol}}{\rightsquigarrow} \mathrm{P}_{i_k}(p_h \leftarrow \bot) \leftarrow \bot$ for every k and h: to make $\mathsf{prohibSign}_h$ true action while performing the policeman role counts as disallowing to anybody to pump water from that basin.

Our causality relation allows us to model indirect effects of actions. For example, basin 1 being close to basin 2, farmer pumping from 1 also lowers the water level in basin 2. This can be expressed by stating $p_1 \leftarrow \bot \Rightarrow p_2 \leftarrow \bot$.

Our definition of capability is only about performance of a single action from the set S. It can be generalized by allowing for arbitrary combinations of actions from S. Let us introduce a modal operator of *iterative capability* $\Diamond^*_{i:S}$ whose truth condition is:

$$M \models \Diamond^*_{i:S}\varphi \text{ iff } \begin{array}{l}\text{there is a sequence } (\alpha_1, \ldots, \alpha_n) \text{ of assignments from } S \text{ such that} \\ M \models \langle i{:}\alpha_1\rangle \ldots \langle i{:}\alpha_n\rangle\varphi\end{array}$$

It is useful to first introduce the dynamic logic program operators `skip` and $\cup$. Their semantics requires to move from the functional interpretation of actions to a relational interpretation: now for every action $i{:}\alpha$, $R_{i:\alpha}$ relates models M to their updates $M^{i:\alpha}$. The recursive definition of R_π is as follows, where π is a program:

$$\begin{array}{ll} R_{i:\alpha} & = \{(M, M') \ : \ M' = M^{i:\alpha}\} \\ R_{\mathtt{skip}} & = \{(M, M') \ : \ M' = M\} \\ R_{\pi_1 \cup \pi_2} & = R_{\pi_1} \cup R_{\pi_2} \end{array}$$

We therefore have $\langle \mathtt{skip} \rangle \varphi \leftrightarrow \varphi$ and $\langle i{:}\alpha \cup j{:}\beta \rangle \varphi \leftrightarrow \langle i{:}\alpha \rangle \varphi \vee \langle j{:}\beta \rangle \varphi$. The truth condition becomes:

$$M \models \langle \pi \rangle \varphi \text{ iff } M' \models \varphi \text{ for every } M' \text{ such that } MR_\pi M'$$

As a simple illustration, observe that the above definition of 'one shot capability' $\Diamond_{i:S}$ can now be written in an alternative and more elegant way. Let $S = \{\alpha_1, \ldots, \alpha_n\}$ be a set of assignments. We have:

$$\Diamond_{i:S} \varphi \leftrightarrow \langle \mathtt{skip} \cup i{:}\alpha_1 \cup \ldots \cup i{:}\alpha_n \rangle \varphi$$

The next result states that in order to check whether $\Diamond^*_{i:S} \varphi$ it suffices to check whether φ can be obtained by performing some of the assignments in S once, in any order, provided that abilities are not used before being acquired, and are not abandoned before used.

Proposition 3. *Let $S = \{\alpha_1, \ldots, \alpha_{\mathsf{card}(S)}\}$ be a set of assignments. The formula*

$$\Diamond^*_{i:S} \varphi \leftrightarrow \langle \mathtt{skip} \cup i{:}\alpha_1 \rangle \cdots \langle \mathtt{skip} \cup i{:}\alpha_{\mathsf{card}(S)} \rangle \varphi$$

is valid, where $(\alpha_1, \ldots, \alpha_{\mathsf{card}(S)})$ is any ordering of the elements of S such that for all $\alpha_l \in S$, whenever $\alpha_k = \mathrm{A}_i(\alpha_l) \leftarrow \top$ then $k < l$, and whenever $\alpha_k = \mathrm{A}_i(\alpha_l) \leftarrow \bot$ then $l < k$.

Proof (sketch). *Suppose $M \models \Diamond^*_{i:S} \varphi$. Hence there is a sequence $(\alpha_1, \ldots, \alpha_n)$ of assignments from S such that $M \models \langle i{:}\alpha_1 \rangle \ldots \langle i{:}\alpha_n \rangle \varphi$. We can permute assignments and put them in the appropriate order by applying the following valid equivalences.*

$$\langle i{:}p \leftarrow \tau \rangle \langle i{:}q \leftarrow \tau' \rangle \varphi \leftrightarrow \begin{cases} \langle i{:}q \leftarrow \tau' \rangle \varphi & \text{when } q = p \\ \langle i{:}q \leftarrow \tau' \rangle \langle i{:}p \leftarrow \tau \rangle \varphi & \text{when } q \neq p \end{cases}$$

$$\langle i{:}\alpha \rangle \langle i{:}\mathrm{A}_j(\beta) \leftarrow \top \rangle \varphi \leftrightarrow \begin{cases} \langle i{:}\alpha \rangle \varphi & \text{when } \beta = \alpha \text{ and } j = i \\ \langle i{:}\mathrm{A}_j(\beta) \leftarrow \top \rangle \langle i{:}\alpha \rangle \varphi & \text{when } \beta \neq \alpha \text{ or } j \neq i \end{cases}$$

$$\langle i{:}\mathrm{A}_j(\beta) \leftarrow \bot \rangle \langle i{:}\alpha \rangle \varphi \leftrightarrow \begin{cases} \bot & \text{when } \beta = \alpha \text{ and } j = i \\ \langle i{:}\alpha \rangle \langle i{:}\mathrm{A}_j(\beta) \leftarrow \bot \rangle \varphi & \text{when } \beta \neq \alpha \text{ or } j \neq i \end{cases}$$

The first equivalence allows to eliminate multiple occurrences of the same assignment from S. The second equivalence allows to move ability gain assignments to the left, while the third equivalence allows to move ability loss assignments to the right. Together, these three equivalences allow to replace $\langle i{:}\alpha_1 \rangle \ldots \langle i{:}\alpha_n \rangle \varphi$ by the equivalent $\langle i{:}\beta_1 \rangle \ldots \langle i{:}\beta_m \rangle \varphi$ such that for all $\beta_l \in S$, whenever $\beta_k = \mathrm{A}_i(\beta_l) \leftarrow \top$ then $k < l$, and whenever $\beta_k = \mathrm{A}_i(\beta_l) \leftarrow \bot$ then $l < k$. It finally follows from the valid implication $\psi \rightarrow \langle \mathtt{skip} \cup i{:}\beta \rangle \psi$ that those elements of S that are not in our sequence yet can be inserted, and that $M \models \langle \mathtt{skip} \cup i{:}\beta_1 \rangle \ldots \langle \mathtt{skip} \cup i{:}\beta_{\mathsf{card}(S)} \rangle \varphi$ where $(\beta_1, \ldots, \beta_{\mathsf{card}(S)})$ is an ordering of the elements of S satisfying the condition of the proposition.

The other direction of the proof is straightforward. ∎

Note that unfolding the right-hand side of the equivalence in Proposition 3 yields a formula in $\mathcal{L}$ that is exponentially larger. In fact, extending the language $\mathcal{L}$ with the

program construct $\cup$ increases the complexity of the logic from NP-complete to PSPACE-complete.[4] This seems to indicate that reasoning about the notion of iterative capability with the operator $\Diamond^*_{i:S}\varphi$ is computationally more expensive.

6 Related Works

In the last decade several logicians have focused on a number of aspects of counts-as such as institutional power [17], defeasibility [3,9], contextual and classificatory aspects [11], mental aspects [21,20], the distinction between brute facts and institutional facts [10].

In their seminal paper [17], Jones and Sergot gave the status of an implication-like logical connective to the counts-as relation. The latter links two propositions φ_1 and φ_2 within a normative system (or institution) s. This is formally written $\varphi_1 \rightsquigarrow_s \varphi_2$ and reads "φ_1 counts as φ_2 in s". Jones and Sergot gave a possible worlds semantics for the counts-as connective together with an axiomatic characterization of the valid formulas of that logic. In order to capture the notion of institutional power they extended their logic with an action component: the 'bringing it about that' modal operator $E_i\varphi$ which has to be read "agent i brings it about that φ". $E_i\varphi_1 \rightsquigarrow_s E_i\varphi_2$ then expresses that "in s, i's action of bringing about φ_1 counts as i's action of bringing about φ_2".

In a more recent paper [11], Grossi and colleagues paved the way towards a substantial simplification of Jones and Sergot's logic. Contrarily to the latter they did not consider the counts-as relation as primitive: the basic logical operators of Grossi et col.'s logic are normal modal operators of the form $[s]\varphi$, reading "in normative system s, it is the case that φ". These operators can be combined with the standard Boolean operators. For example, $[s](\varphi \rightarrow \psi)$ is a formula of the language of Grossi et col., reading "in s, if φ then ψ". Based on the $[s]$ connectives, Grossi et col. then define the counts-as connective. First of all they argue that the formula $[s](\varphi \rightarrow \psi)$ is already an approximation of $\varphi \rightsquigarrow_s \psi$. Nevertheless, this approximation validates formulas such as $\varphi \rightsquigarrow_s \top$: in s, any φ counts as a tautology. This is felt to be counter-intuitive. Therefore, in order to better capture Jones and Sergot's $\rightsquigarrow_s$ connective, Grossi et col. introduce a so-called universal modality $[\forall]$, where $[\forall]\varphi$ reads "φ universally holds". The latter is used in order to strengthen the link between φ and ψ: in addition to $[s](\varphi \rightarrow \psi)$, Grossi et col. moreover require that for φ to count as ψ it should not be universally true that φ implies ψ. In formulas, they define a so-called *proper classificatory rule* $\varphi \rightsquigarrow^{cl+}_s \psi$ by stipulating:

$$\varphi \rightsquigarrow^{cl+}_s \psi \stackrel{\mathtt{def}}{=} [s](\varphi \rightarrow \psi) \wedge \neg[\forall](\varphi \rightarrow \psi)$$

In this way they guarantee that no φ counts as a tautology.

There are several novel aspects in our logical analysis of counts-as. First of all, differently from other approaches, our framework allows to explicitly represent physical actions and institutional actions, as well as the links between the two kinds of actions. By distinguishing in the object language a counts-as relation from a causal relation between events, our logic clearly opposes Goldman's notion of causal generation to that of conventional generation. We have shown that these two relations are ontologically

[4] There is an easy reduction from QSAT, see e.g. [14].

different for at least two reasons. While the former relates a physical action to another physical action, the latter relates a physical action to an institutional action. Moreover, while the causal relation is merely a relation between physical actions performed by an agent, counts-as is a relation between actions performed by an agent playing a certain role.

Furthermore, while previous logical accounts of counts-as were mainly conceptual and did not consider decidability issues, our work also focuses on the computational aspects of a logic of institutional action: we have provided in Section 4 a complete axiomatization of our logic based on reduction axioms and have characterized the complexity of the satisfiability problem.

We note that technically, our reduction axioms in terms of a causality relation are close to causality-based solutions to the ramification problem in reasoning about actions [18, 24, 25, 31, 28].

7 Conclusion

In the framework presented in this paper counts-as and causal relations are static, that is, there is no way to update models in order to modify these relations. An interesting direction of future research is to integrate into the framework a dynamic dimension of counts-as and causality in order to be able to model interesting phenomena such as: (1) the modification of causal connections between physical events (e.g. by disconnecting the electric wires, I can remove the causal relation *"flipping the switch" causes "turning on the lights"*); (2) norm promulgation (creating a new counts-as relation between events); (3) norm cancellation (removing a pre-existent counts-as relation between events).

Another interesting topic of future research is the creation of institutional facts. We intend to extend our logic in order to model scenarios such as the following one. Before 2000, it was not possible to assign a truth value to the sentence "he has a note of 50 Euro in his pocket", as the concept "Euro" was not an element of our vocabulary of institutional facts and objects. After its introduction the Euro became an element of our vocabulary of institutional facts and objects. This might be integrated into our framework by adapting approaches to awareness such as [13].

Finally, at the current stage our logic allows to clearly distinguish physical actions with physical effects from institutional actions with institutional effects. Nevertheless, it does not support reasoning about physical actions that an agent may decide to perform on the basis of his preferences. A further interesting direction of future research is to relate our framework with game and decision theory by introducing a notion of preference. This extended framework will allow to reason about situations in which agents desire that certain physical and/or institutional facts obtain, and choose strategically a given physical action in order to ensure these facts.

Acknowledgements

We are grateful to the reviewers of CLIMA XII. Nicolas Troquard is supported by a Marie Curie Trentino Fellowship.

References

1. Alchourrón, C., Bulygin, E.: Normative systems. Springer, New York (1971)
2. van Benthem, J., van Eijck, J., Kooi, B.: Logics of communication and change. Information and Computation 204(204), 1620–1662 (2006)
3. Boella, G., van der Torre, L.: Regulative and constitutive norms in normative multiagent systems. In: KR 2004, pp. 255–266. AAAI Press, Menlo Park (2004)
4. Bulygin, E.: On norms of competence. Law and Philosophy 11(3), 201–216 (1992)
5. Davidson, D.: Agency. In: Essays on Actions and Events. Oxford University Press, New York (1980)
6. van Ditmarsch, H.P., van der Hoek, W., Kooi, B.: Dynamic epistemic logic with assignment. In: AAMAS 2005, pp. 141–148. ACM Press, New York (2005)
7. Gerbrandy, J.: Logics of propositional control. In: Proc. AAMAS 2006, pp. 193–200 (2006)
8. Goldman, A.: A Theory of Human Action. Prentice-Hall, Englewood Cliffs (1970)
9. Governatori, G., Gelati, J., Rotolo, A., Sartor, G.: Actions, institutions, powers: preliminary notes. In: Lindemann, G., Moldt, D., Paolucci, M. (eds.) RASTA 2002. LNCS (LNAI), vol. 2934, pp. 131–147. Springer, Heidelberg (2004)
10. Grossi, D.: A note on brute vs. institutional facts: modal logic of equivalence up to a signature. In: Boella, G., Pigozzi, G., Noriega, P., Verhagen, H. (eds.) Proceedings of Dagstuhl Seminar on Normative Multi-agent Systems (2009)
11. Grossi, D., Meyer, J.-J.C., Dignum, F.: Classificatory aspects of counts-as: An analysis in modal logic. Journal of Logic and Computation 16(5), 613–643 (2006)
12. Hart, H.L.A.: The concept of law, new edn. Clarendon Press, Oxford (1992)
13. Heifetz, A., Meier, M., Schipper, B.: Interactive unawareness. Journal of Economic Theory 130(1), 78–94 (2006)
14. Herzig, A., Lorini, E., Moisan, F., Troquard, N.: A dynamic logic of normative systems (regular paper). In: Walsh, T. (ed.) Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011). Morgan Kaufmann Publisher, Barcelona (2011)
15. van der Hoek, W., Walther, D., Wooldridge, M.: Reasoning about the transfer of control. JAIR 37, 437–477 (2010)
16. van der Hoek, W., Wooldridge, M.: On the logic of cooperation and propositional control. Artificial Intelligence 164(1-2), 81–119 (2005)
17. Jones, A., Sergot, M.J.: A formal characterization institutionalised power. J. of the IGPL 4, 429–445 (1996)
18. Lin, F.: Embracing causality in specifying the indirect effects of actions. In: Proc. 14th Int. Joint Conf. on Artificial Intelligence (IJCAI 1995), pp. 1985–1991 (1995)
19. Lorini, E., Herzig, A.: A logic of intention and attempt. Synthese 163(1), 45–77 (2008)
20. Lorini, E., Longin, D.: A logical account of institutions: from acceptances to norms via legislators. In: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), pp. 38–48. AAAI Press, Menlo Park (2008)
21. Lorini, E., Longin, D., Gaudou, B., Herzig, A.: The logic of acceptance: grounding institutions on agents' attitudes. J. of Logic and Computation 19(6), 901–940 (2009)
22. Makinson, D.: On the formal representation of rights relations: remarks on the work of Stig Kanger and Lars Lindahl. The Journal of Philosophical Logic 15, 403–425 (1986)
23. Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gangemi, A., Guarino, N.: Social roles and their descriptions. In: Dubois, D., Welty, C.A., Williams, W.-A. (eds.) KR, pp. 267–277. AAAI Press, Menlo Park (2004)
24. McCain, N., Turner, H.: A causal theory of ramifications and qualifications. In: Proc. 14th Int. Joint Conf. on Artificial Intelligence (IJCAI 1995), pp. 1978–1984 (1995)

25. McCain, N., Turner, H.: Causal theories of action and change. In: Proc. Nat. Conf. on Artificial Intelligence (AAAI 1997), pp. 460–465 (1997)
26. Pauly, M.: A modal logic for coalitional power in games. Journal of Logic and Computation 12(1), 149–166 (2002)
27. Pörn, I.: Action Theory and Social Science: Some Formal Models. Synthese Library, vol. 120. D. Reidel, Dordrecht (1977)
28. Schwind, C.: Causality in action theories. ETAI Electronic Articles in Computer and Information Science 3(A), (1999), www.ida.liu.se/ext/etai
29. Searle, J.R.: Speech acts: An essay in the philosophy of language. Cambridge University Press, New York (1969)
30. Searle, J.R.: The Construction of Social Reality. The Free Press, New York (1995)
31. Thielscher, M.: Ramification and causality. Artificial Intelligence 89, 317–364 (1997)
32. Tiomkin, M.L., Makowsky, J.A.: Propositional dynamic logic with local assignments. Theor. Comput. Sci. 36, 71–87 (1985)
33. van Eijck, J.: Making things happen. Studia Logica 66(2), 41–58 (2000)

A Paraconsistent Multi-agent Framework for Dealing with Normative Conflicts*

Mathieu Beirlaen and Christian Straßer

Centre of Logic and Philosophy of Science, Ghent University
{Mathieu.Beirlaen,Christian.Strasser}@UGent.be

Abstract. In a multi-agent deontic setting, normative conflicts can take a variety of different logical forms. In this paper, we present a very general characterization of such conflicts, including both intra- and inter-agent normative conflicts, conflicts between groups of agents, conflicts between obligations and permissions, and conflicts between contradictory norms. In order to account for the consistent possibility of this wide variety of conflict-types, we present a paraconsistent deontic logic, i.e. a logic that invalidates the classical principle of non-contradiction. Next, we strengthen this logic within the adaptive logics framework for defeasible reasoning. The resulting inconsistency-adaptive deontic logic interprets a given set of norms 'as consistently as possible'.

1 Introduction

The development of systems capable of tolerating conflicting norms is considered an important challenge in the fields of deontic logic [15] and normative multi-agent systems [7]. In this paper, we try to meet this challenge by consistently allowing for various types of normative conflicts within a non-classical multi-agent framework, i.e. a multi-agent framework that invalidates some rules and theorems of Standard Deontic Logic (**SDL**).

For reasons of presentation we will first introduce a classical variant of the framework (Section 2), and illustrate how the resulting logic **MDC** treats individual and collective obligations. Next, we present a subdivision of various types of normative conflicts (Section 3), and show that **MDC** cannot consistently allow for the possibility of such conflicts.

In order to prevent instances of normative conflicts from giving rise to deontic explosion, we introduce a paraconsistent variant of the logic **MDC**: the logic **MDP** (Section 4). As opposed to **MDC**, **MDP** can consistently deal with any type of normative conflict. However, the conflict-tolerance of **MDP** comes at a price. Since **MDP** gives up some of the rules validated by **SDL** (and hence by **MDC**), it loses much of the latter system's inferential power. This drawback is common to any monotonic paraconsistent deontic logic presented so far (Section 5).

* Research for this paper was supported by subventions from the Research Foundation – Flanders (FWO Vlaanderen). The authors are indebted to the anonymous referees for valuable comments and suggestions.

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 312–329, 2011.

The solution to this problem presented here consists of extending **MDP** within the adaptive logics framework [4]. In the resulting logic (called $\mathbf{MDP^m}$), some **MDC**-inferences are made conditional upon the behavior of the premises: $\mathbf{MDP^m}$ verifies only those inferences which rely on premises that can safely be assumed to behave 'normally'. The technically precise sense in which $\mathbf{MDP^m}$ does so is spelled out in Section 6. $\mathbf{MDP^m}$ has the nice property that for premise sets all members of which behave 'normally' in this sense, $\mathbf{MDP^m}$ delivers the same consequences as **MDC**.

2 A Simple Classical Multi-agent Framework

2.1 Language

We use a denumerable set $\mathcal{W}^a$ of propositional constants (atoms) $p, q, r, \ldots$. The $\langle \neg, \wedge, \vee, \supset, \equiv \rangle$-closure of $\mathcal{W}^a$ is denoted by $\mathcal{W}$. We call formulas in $\mathcal{W}$ (purely) propositional formulas.

Next to propositional formulas, we use a finite set $I = \{i_1, \ldots, i_n\}$ of agents. We will in the remainder often refer to groups of agents J in I, i.e. non-empty subsets of I. The following notation is useful for this: $J \subseteq_\emptyset I$ iff $J \neq \emptyset$ and $J \subseteq I$. The set $\mathcal{W}_I = \{\langle A, J\rangle \mid A \in \mathcal{W}, J \subseteq_\emptyset I\}$ denotes the set of agent-proposition pairs. Throughout the paper, we will use "A_J" as a shortcut for "$\langle A, J\rangle$." Where $i \in I$, we will in the remainder of the paper abbreviate $A_{\{i\}}$ by A_i. A formula $A_J \in \mathcal{W}_I$ is translated as "group J brings about A by a joint effort". We will discuss and distinguish this notion from another, weaker reading of group obligations in Section 2.3. $\mathcal{W}_I^c$ is the set of all formulas in the $\langle \neg, \wedge, \vee, \supset, \equiv \rangle$-closure of $\mathcal{W}_I$. Where $\mathcal{W}^l$ denotes the set of literals (i.e. the set of atoms in $\mathcal{W}^a$ and their negations), we also define the set $\mathcal{W}_I^l = \{A_J \mid A \in \mathcal{W}^l, J \subseteq_\emptyset I\}$ of agent-literal complexes. Finally, the set $\mathcal{W}^c$ of well-defined formulas for the classical multi-agent framework is defined recursively as follows:

$$\mathcal{W}^c := \langle \mathcal{W} \cup \mathcal{W}_I \rangle \mid \mathsf{O}\langle \mathcal{W}_I^c \rangle \mid \mathsf{P}\langle \mathcal{W}_I^c \rangle \mid \neg\langle \mathcal{W}^c \rangle \mid \langle \mathcal{W}^c \rangle \wedge \langle \mathcal{W}^c \rangle \mid \langle \mathcal{W}^c \rangle \vee \langle \mathcal{W}^c \rangle \mid \langle \mathcal{W}^c \rangle \supset \langle \mathcal{W}^c \rangle \mid \langle \mathcal{W}^c \rangle \equiv \langle \mathcal{W}^c \rangle$$

Where $A \in \mathcal{W}_I^c$, a formula $\mathsf{O}A$ [$\mathsf{P}A$] is interpreted as "it ought to be [is permitted] that A". Hence, $\mathsf{O}A_i$ is read as "It ought to be that agent i brings about A". Similarly, $\mathsf{O}A_J$ is read as "The group of agents J ought to bring about A by a joint effort". We do not allow for formulas $\mathsf{O}A$ and $\mathsf{P}A$ where $A \in \mathcal{W}^c \setminus \mathcal{W}_I^c$ such as $\mathsf{O}B$ where B is a propositional atom. This is because we are only interested in obligations that are addressed directly to (groups of) agents. Note that we do allow for formulas such as $\mathsf{O}(A_i \vee B_j)$ and $\mathsf{O}A_i \vee \mathsf{O}B_j$. While the former expresses that it ought to be the case that either i brings about A or that j brings about B, the latter expresses that either i ought to bring about A or j ought to bring about B. This difference corresponds to the distinction in **SDL** between the formulas $\mathsf{O}(A \vee B)$ and $\mathsf{O}A \vee \mathsf{O}B$.

There is another subtlety worth pointing out, namely the difference between $\mathsf{O}\neg(A_i)$ and $\mathsf{O}(\neg A)_i$. While the latter indicates i's obligation to bring about $\neg A$, the former is literally read as "It ought to be that it is not the case that i brings about A". This can be understood as i's obligation to refrain from bringing about A.

2.2 The Logic MDC

In this section we present a classical system for modeling normative reasoning. We presuppose that (i) norms dealt with by this system arise from the same source, and (ii) agents have epistemic access to *all* norms issued by this source.

Let us demonstrate how to adjust the Kripke-frames that are usually used in order to semantically characterize **SDL** to the multi-agent setting of **MDC**. We shortly outline some of the basic ideas. An **SDL**-model is a tuple $M = \langle W, R, v, w^0\rangle$. W is a set of worlds where each world is associated with a set of atoms by the assignment function $v : \mathcal{W}^a \rightarrow \wp(W)$. A propositional atom A is said to hold in a world w iff it is assigned to the world by v, i.e. $w \in v(A)$. The validity of complex formulas is then recursively defined as usual. $R \subseteq W \times W$ is a serial accessibility relation. A formula A is obliged in a world w iff it is valid in all the accessible worlds of w. Moreover, $w^0 \in W$ is the so-called actual world.

Let us now step-by-step generalize these frames for the multi-agent setting. First we need to introduce agents. We represent them by a finite non-empty set $I = \{i_1, \ldots, i_n\}$. An **MDC**-model is a tuple $M = \langle W, I, R, v, v_I, w^0\rangle$ where as before W is a set of worlds, $R \subseteq W \times W$ is a serial accessibility relation, $v : \mathcal{W}^a \rightarrow \wp(W)$ is an assignment function, and $w^0 \in W$ is the actual world. Just as before, the idea is that the propositional atom A is the case in w iff $w \in v(A)$.

We are not only interested in what is the case in our worlds, but also in causation, more precisely the question which agents cause certain events. In order to express this, our worlds are not just points, such as in the case of the **SDL**-semantics, but they are structured. Every world $w \in W$ is associated with tuples $\langle w, J\rangle$, for all $J \subseteq_\emptyset I$. We use w_J as a shortcut for $\langle w, J\rangle$.

While in **SDL** the assignment v associates a world $w \in W$ with atoms in order to express what atoms hold in w, we add now an additional assignment v_I that associates each w_J with literals in order to express what literals are brought about by the group of agents J. The idea is that a literal A is brought about in w by a group of agents J iff $w_J \in v(A)$. Hence, $v_I : \mathcal{W}^l \rightarrow \wp(\{w_J \mid w \in W, J \subseteq_\emptyset I\})$.

v associates only atoms (and not literals) with worlds because this provides enough information to uniquely define whether a complex propositional formula representing factual information holds in a world. We for instance do not need to assign worlds to negated propositional atoms such as $\neg A$, since by means of a semantic clause such as the following it can be determined whether $\neg A$ holds in a world w: (†) "$\neg A$ holds in w in a model M iff A does not hold in w in M". Note that, in order to determine whether J brings about $\neg A$ in w, we cannot rely on the fact that J does not bring about A. After all, from the fact that J refrains from bringing about A we cannot infer that J brings about $\neg A$. The

fact that A or $\neg A$ holds in a world may be independent of actions by J. Hence, we need to specify for each literal by what group of agents it is brought about (if any).

In the **SDL**-semantics the clause (†) ensures that the worlds are *consistent* in the sense that it is not the case that for an atom A, A holds in a world w and at the same time $\neg A$ holds in the world w. Since v_I associates worlds with both atoms and their negations we need to ensure the consistency by a frame-condition:

F-Con. For all $A \in \mathcal{W}^a$, for all $w \in W$, and for all $J, K \subseteq_\emptyset I$: (i) if $w_J \in v_I(A)$ then $w_K \notin v_I(\neg A)$ and (ii) if $w_J \in v_I(\neg A)$ then $w_K \notin v_I(A)$.

Moreover, we want to ensure that whenever an agent or group brings about A, then A is also the case (factually). This is guaranteed by adding the following frame condition:

F-Fac. For all $A \in \mathcal{W}^a$ and all $w \in W$, (i) if $w_J \in v_I(A)$ then $w \in v(A)$ and (ii) if $w_J \in v_I(\neg A)$ then $w \notin v(A)$.

The valuation $v_M : \mathcal{W}^c \to W$ associated with the model M is defined by:

C_I^l	where $A_J \in \mathcal{W}_I^l : M, w \models A_J$ iff $w_J \in v_I(A)$
$\mathrm{C}_I\wedge$	where $A, B \in \mathcal{W} : M, w \models (A \wedge B)_J$ iff $M, w \models A_J$ and $M, w \models B_J$
$\mathrm{C}_I\vee$	where $A, B \in \mathcal{W} : M, w \models (A \vee B)_J$ iff $M, w \models A_J$ or $M, w \models B_J$
$\mathrm{C}_I\supset$	where $A, B \in \mathcal{W} : M, w \models (A \supset B)_J$ iff $M, w \not\models A_J$ or $M, w \models B_J$
$\mathrm{C}_I\equiv$	where $A, B \in \mathcal{W} : M, w \models (A \equiv B)_J$ iff $(M, w \models A_J$ iff $M, w \models B_J)$
$\mathrm{C}_I\neg\neg$	where $A \in \mathcal{W} : M, w \models (\neg\neg A)_J$ iff $M, w \models A_J$
$\mathrm{C}_I\neg\vee$	where $A, B \in \mathcal{W} : M, w \models (\neg(A \vee B))_J$ iff $M, w \models (\neg A \wedge \neg B)_J$
$\mathrm{C}_I\neg\wedge$	where $A, B \in \mathcal{W} : M, w \models (\neg(A \wedge B))_J$ iff $M, w \models (\neg A \vee \neg B)_J$
$\mathrm{C}_I\neg\supset$	where $A, B \in \mathcal{W} : M, w \models (\neg(A \supset B))_J$ iff $M, w \models (A \wedge \neg B)_J$
$\mathrm{C}_I\neg\equiv$	where $A, B \in \mathcal{W} : M, w \models (\neg(A \equiv B))_J$ iff $M, w \models ((A \vee B) \wedge (\neg A \vee \neg B))_J$
C^a	where $A \in \mathcal{W}^a : M, w \models A$ iff $w \in v(A)$
$\mathrm{C}\neg$	where $A \in \mathcal{W}^c : M, w \models \neg A$ iff $M, w \not\models A$
$\mathrm{C}\wedge$	where $A, B \in \mathcal{W}^c : M, w \models A \wedge B$ iff $M, w \models A$ and $M, w \models B$
$\mathrm{C}\vee$	where $A, B \in \mathcal{W}^c : M, w \models A \vee B$ iff $M, w \models A$ or $M, w \models B$
$\mathrm{C}\supset$	where $A, B \in \mathcal{W}^c : M, w \models A \supset B$ iff $M, w \not\models A$ or $M, w \models B$
$\mathrm{C}\equiv$	where $A, B \in \mathcal{W}^c : M, w \models A \equiv B$ iff $(M, w \models A$ iff $M, w \models B)$
$\mathrm{C}\mathsf{O}$	where $A \in \mathcal{W}_I^c : M, w \models \mathsf{O}A$ iff $M, w' \models A$ for all w' such that Rww'
$\mathrm{C}\mathsf{P}$	where $A \in \mathcal{W}_I^c : M, w \models \mathsf{P}A$ iff $M, w' \models A$ for some w' such that Rww'

An **MDC**-model M verifies $A \in \mathcal{W}^c$ ($M \Vdash_{\mathbf{MDC}} A$) iff $M, w^0 \models A$. Where $\Gamma \subseteq \mathcal{W}^c$, M is an **MDC**-model of Γ iff M is an **MDC**-model and $M \Vdash_{\mathbf{MDC}} A$ for all $A \in \Gamma$. Moreover, $\models_{\mathbf{MDC}} A$ iff all **MDC**-models verify A, and $\Gamma \models_{\mathbf{MDC}} A$ iff all **MDC**-models of Γ verify A.

All of the following inferences are valid in **MDC** (where $A, B \in \mathcal{W}_I^c$):

$$\mathsf{O}A, \mathsf{O}B \models_{\mathbf{MDC}} \mathsf{O}(A \wedge B)$$
$$\mathsf{O}A \models_{\mathbf{MDC}} \neg\mathsf{O}\neg A$$
$$\mathsf{O}(A \vee B), \mathsf{O}\neg A \models_{\mathbf{MDC}} \mathsf{O}B$$

2.3 More on Group Obligations

Where $i, j \in I$, the formula $\mathsf{O}A_{\{i,j\}}$ abbreviates a collective obligation for group $\{i, j\}$ to bring about A. Note that none of $\mathsf{O}A_i$, $\mathsf{O}A_j$, $\mathsf{O}A_i \vee \mathsf{O}A_j$, and $\mathsf{O}(A_i \vee A_j)$ is **MDC**-derivable from $\mathsf{O}A_{\{i,j\}}$. This is due to the fact that $\mathsf{O}A_{\{i,j\}}$ expresses that i and j should bring about A by a joint effort. Collective obligations of this kind are called *strict collective obligations* by Dignum & Royakkers [12]. A strict collective obligation to bring about A is satisfied only if *all* agents in the collective bring about A *together*.

Not all collective obligations are strict collective obligations. Suppose, for instance, that a mother of three children orders her offspring to do the dishes. In order to satisfy this obligation, it might not matter if only one or two of the children actually do the dishes. All that matters is that, in the end, the dishes are clean. The obligation issued by this agent is hence not a strict collective obligation. It is what Dignum & Royakkers call a *weak collective obligation*. A weak collective obligation to bring about A is satisfied as soon as any subset of the collective brings about A.

Although the formula $\mathsf{O}A_J$ is in **MDC** interpreted as a strict collective obligation, we can also define an obligation operator O^{w} in order to express weak collective obligations:

$\mathsf{O}^{\mathsf{w}}A_J =_{\mathrm{df}} \mathsf{O}(\bigvee_{K \subseteq_\emptyset J} A_K)$

The weak collective obligation operator O^{w} captures the intended meaning that if a group of agents ought to bring about a certain state of affairs, then this state of affairs ought to be brought about by some subset of the group.[1] It follows immediately by the definition and $\mathrm{C}_I\vee$ that $\models_{\mathbf{MP}} \mathsf{O}A_J \supset \mathsf{O}^{\mathsf{w}}A_J$. Obviously, if the group J faces the strict collective obligation to bring about A, then some subgroup of J –namely J itself– has to bring about A. Note that $\mathsf{O}^{\mathsf{w}}A_i = \mathsf{O}A_i$.

The disambiguation of the notion of collective obligation by means of the distinction between strict and weak collective obligations allows us to further illustrate some subtle differences in **MDC**. Suppose that some agent i ought to bring about $\neg A$, whereas agents i and j ought to bring about $A \vee B$. If the latter obligation is interpreted as a strict collective obligation, then it is **MDC**-derivable that i and j share the strict collective obligation to bring about B:

(1) $\mathsf{O}(\neg A)_i, \mathsf{O}(A \vee B)_{\{i,j\}} \models_{\mathbf{MDC}} \mathsf{O}B_{\{i,j\}}$

[1] The O^{w}-operator as defined here is slightly different from the one defined by Dignum & Royakkers in [12]. We write the latter operator as O_{w}. Then $\mathsf{O}_{\mathsf{w}}A_J =_{\mathrm{df}} \bigvee_{K \subseteq_\emptyset J} \mathsf{O}A_K$. Note that $\mathsf{O}^{\mathsf{w}}A_{\{a,b\}} = \mathsf{O}(A_a \vee A_b \vee A_{\{a,b\}})$, while $\mathsf{O}_{\mathsf{w}}A_{\{a,b\}} = \mathsf{O}A_a \vee \mathsf{O}A_b \vee \mathsf{O}A_{\{a,b\}}$. We prefer to define weak obligation in terms of O^{w} instead of O_{w} because we take a weak (collective) obligation to be a single norm rather than a disjunction of norms.

In general, if some group faces a strict collective obligation, then it should try to satisfy this obligation in a way that conflicting obligations are avoided whenever possible. This is exactly what happens in the above example.

If we interpret i and j's obligation to bring about $A \vee B$ as a weak collective obligation, then $\mathsf{O}B_{\{i,j\}}$ is no longer **MDC**-derivable, but the weaker obligation $\mathsf{O}^{\mathsf{w}}B_{\{i,j\}}$ is:

(2) $\mathsf{O}(\neg A)_i, \mathsf{O}^{\mathsf{w}}(A \vee B)_{\{i,j\}} \models_{\mathbf{MDC}} \mathsf{O}^{\mathsf{w}}B_{\{i,j\}}$

Again, conflicting obligations are neatly avoided: i and j's weak obligation to bring about $A \vee B$ is satisfied in a consistent way whenever i, j, or i and j together bring about B.

If instead of supposing that i has the obligation to bring about $\neg A$, we suppose that i merely has the obligation to refrain from bringing about A, the above reasoning no longer applies:

(3) $\mathsf{O}\neg(A_i), \mathsf{O}(A \vee B)_{\{i,j\}} \not\models_{\mathbf{MDC}} \mathsf{O}B_{\{i,j\}}$

That i ought to refrain from bringing about A, does not entail that the group $\{i, j\}$ ought to do so.[2] Hence there is no strict obligation for $\{i, j\}$ to bring about B. In the variant for weak collective obligation, a similar reasoning applies:

(4) $\mathsf{O}\neg(A_i), \mathsf{O}^{\mathsf{w}}(A \vee B)_{\{i,j\}} \not\models_{\mathbf{MDC}} \mathsf{O}^{\mathsf{w}}B_{\{i,j\}}$

That i should refrain from bringing about A does not allow us to derive a weak collective obligation for i and j to bring about B, because $\mathsf{O}^{\mathsf{w}}(A \vee B)_{\{i,j\}}$ is also satisfied if, for instance, j brings about A or if i and j together (in the strict sense) bring about A.

3 Normative Conflicts

In single-agent settings, normative conflicts (moral conflicts, deontic conflicts) are usually conceived as situations in which an agent has two (or more) conflicting obligations. In the language of **MDC**, such *intra-agent* conflicts between obligations can have two logical forms. Where the agent in question is represented by the subscript i, we say that i faces an obligation-obligation conflict (in short, an OO-*conflict*) if, for some A, either $\mathsf{O}A_i \wedge \mathsf{O}(\neg A)_i$ or $\mathsf{O}A_i \wedge \mathsf{O}\neg(A_i)$. In the first case, i has both an obligation to bring about A and an obligation to bring about $\neg A$. In the second case, i has both an obligation to bring about A and an obligation to refrain from bringing about A. Similarly, a group of agents J faces an OO-conflict if $\mathsf{O}A_J \wedge \mathsf{O}(\neg A)_J$ or if $\mathsf{O}A_J \wedge \mathsf{O}\neg(A_J)$.

In a multi-agent setting, we have to allow for the possibility of *inter-agent* conflicts next to intra-agent conflicts. Conflicts of obligations between *different* (groups of) agents can arise only in case one of the agents or groups, say J, has to bring about a state of the world inconsistent with a state of the world that should be brought about by another agent or group, say K, i.e. if $\mathsf{O}A_J \wedge \mathsf{O}(\neg A)_K$.

[2] Suppose, for example, that i has to refrain from lifting a heavy cupboard (because, for instance, i has chronic back pain). From this it does not follow that i should still refrain from doing so if she is assisted by j.

Note that if $J \neq K$, a formula $\mathsf{O}A_J \wedge \mathsf{O}\neg(A_K)$ no longer guarantees a conflict of obligations in the multi-agent setting: it is perfectly possible that agent or group J brings about A, while another agent or group K refrains from bringing about A. Altogether, in a multi-agent framework an OO-conflict has one of the following two logical forms: $\mathsf{O}A_J \wedge \mathsf{O}\neg(A_J)$ or $\mathsf{O}A_J \wedge \mathsf{O}(\neg A)_K$ (where possibly $J = K$).

Logicians often limit their study of normative conflicts to conflicts between two or more obligations, e.g. [14,16,18,19,25]. However, other types of normative conflicts can occur. It might, for instance, be the case that an agent or group J ought to bring about A, while J is also permitted to refrain from doing so, i.e. $\mathsf{O}A_J \wedge \mathsf{P}\neg(A_J)$. Moreover, J might have the obligation to bring about A while a possibly different group or agent K is permitted to bring about $\neg A$, i.e. $\mathsf{O}A_J \wedge \mathsf{P}(\neg A)_K$. In what follows such conflicts will be called obligation-permission conflicts or OP-*conflicts*. For some examples of OP-conflicts in a single-agent setting, see [6,30]. The possibility of OP-conflicts was also defended in [1,2,8,35].

In [6,24] examples were given of *contradicting norms*. Suppose, for instance, that in some country the constitution contains the following clauses concerning parliamentary elections: (i) it is not the case that women are permitted to vote, and (ii) property holders are permitted to vote. Suppose further that (possibly due to a recent revision of the property law) women are allowed to hold property. Then the law is inconsistent: any female property holder i is both permitted and not permitted to vote: $\mathsf{P}V_i \wedge \neg\mathsf{P}V_i$ (example from [24, pp. 184-185]).

The same reasoning holds, of course, for formulas of the form $\mathsf{O}A \wedge \neg\mathsf{O}A$ (where $A \in \mathcal{W}_I^c$). As hinted at above, normative conflicts of the type $\mathsf{P}A \wedge \neg\mathsf{P}A$ or $\mathsf{O}A \wedge \neg\mathsf{O}A$ are called contradicting norms.

Next to contradicting norms, i.e. different norms that contradict *each other*, one might also face a *contradictory norm*, i.e. a norm that contradicts *itself*. A contradictory norm is of the form $\mathsf{O}(A_J \wedge \neg(A_J))$, $\mathsf{P}(A_J \wedge \neg(A_J))$, $\mathsf{O}(A_J \wedge (\neg A)_K)$, $\mathsf{P}(A_J \wedge (\neg A)_K)$, $\mathsf{O}(A \wedge \neg A)_J$, or $\mathsf{P}(A \wedge \neg A)_J$. For a defense of contradictory norms, we refer to [24].

Unfortunately, none of the types of normative conflicts presented above can be dealt with consistently by the logic **MDC**. **MDC** trivializes all instances of all types of normative conflicts. This gives rise to what is usually called *deontic explosion*: the fact that from a deontic conflict any obligation follows. See [14,30] for a more detailed discussion of this phenomenon in deontic logic. An oversight of the various types of normative conflicts and their accompanying principles of deontic explosion is provided in the table below. Where $A \in \mathcal{W}$ and $B, C \in \mathcal{W}_I^c$:

$$
\begin{array}{rrr}
\text{OO-conflicts:} & \mathsf{O}A_J \wedge \mathsf{O}\neg(A_J) \models \mathsf{O}C, & \mathsf{O}A_J \wedge \mathsf{O}(\neg A)_K \models \mathsf{O}C \\
\text{OP-conflicts:} & \mathsf{O}A_J \wedge \mathsf{P}\neg(A_J) \models \mathsf{O}C, & \mathsf{O}A_J \wedge \mathsf{P}(\neg A)_K \models \mathsf{O}C \\
\text{Contradicting norms:} & \mathsf{O}B \wedge \neg\mathsf{O}B \models \mathsf{O}C, & \mathsf{P}B \wedge \neg\mathsf{P}B \models \mathsf{O}C \\
\text{Contradictory norms:} & \mathsf{O}(A_J \wedge \neg(A_J)) \models \mathsf{O}C, & \mathsf{P}(A_J \wedge \neg(A_J)) \models \mathsf{O}C, \\
 & \mathsf{O}(A_J \wedge (\neg A)_K) \models \mathsf{O}C, & \mathsf{P}(A_J \wedge (\neg A)_K) \models \mathsf{O}C, \\
 & \mathsf{O}(A \wedge \neg A)_J \models \mathsf{O}C, & \mathsf{P}(A \wedge \neg A)_J \models \mathsf{O}C
\end{array}
$$

4 Avoiding Deontic Explosion: The Logic MDP

Since **MDC** causes explosion when faced with a normative conflict, and since we want to allow for the consistent possibility of normative conflicts, we need a logic that is weaker than **MDC**.[3] The situation is analogous in non-agentive settings. There too, **SDL** gives rise to explosion in view of formulas of the form $\mathsf{O}A \wedge \mathsf{O}\neg A$, $\mathsf{O}A \wedge \mathsf{P}\neg A$, etc. And there too, authors have suggested weakening the logic in order to tolerate normative conflicts; for some examples, see [13,22,25,28,31,32]. A good oversight can be found in [14].

The solution presented here is to replace the classical negation operator by a weaker negation operator that renders invalid the *Ex Contradictione Quodlibet* principle (ECQ), i.e. $A \wedge \neg A \models B$. One of the main reasons for invalidating ECQ in deontic logic is that it is the only possible solution for consistently allowing for contradicting norms.

Logics that invalidate ECQ are usually called *paraconsistent* logics. In a single-agent deontic setting, paraconsistent deontic logics have been presented in [6,11,24]. To the best of our knowledge, this solution was never before used in a multi-agent deontic setting.

The logic obtained by replacing the classical negation of **MDC** by a weaker, paraconsistent negation is called **MDP**.[4]

Since we want **MDP** to invalidate all explosion principles from the table in Section 3, frame condition **F-Con** must be given up. In **MDC**, **F-Con** excludes accessible worlds which validate both A_J and $(\neg A)_K$ for some J and K. Hence this condition immediately trivializes e.g. normative conflicts of the form $\mathsf{O}A_J \wedge \mathsf{O}(\neg A)_K$ or $\mathsf{O}A_J \wedge \mathsf{P}(\neg A)_K$. Thus if we want to consistently allow for all types of normative conflicts, **F-Con** must be rejected.

Giving up **F-Con** takes us one step closer towards a conflict-tolerant deontic logic. However, even if **F-Con** is rejected, triviality still ensues in view of e.g. conflicts of the form $\mathsf{O}A_J \wedge \mathsf{O}\neg(A_J)$ or $\mathsf{O}A_J \wedge \mathsf{P}\neg(A_J)$. Hence more work is needed in order to make the new logic fully conflict-tolerant, i.e. in order to invalidate all explosion principles stated for **MDC** in the table in Section 3.

Analogous to **MDC**-models, **MDP**-models are tuples $\langle W, I, R, v, v_I, w^0 \rangle$. The only difference is that the factual assignment v is now defined more broadly, i.e. $v : \mathcal{W}^l \cup \{\neg(A_J) \mid A \in \mathcal{W}^l\} \rightarrow \wp(W)$. Moreover we remove the **MDC**-frame condition **F-Con**, and replace **F-Fac** with **F-Fac′**:

[3] Some authors circumnavigate the problems posed by normative conflicts by making their formal system more expressive rather than by weakening its axioms or rules. For instance, in [19] Kooi & Tamminga add super- and subscripts to the deontic operators in order to express the source and the interest group in view of which a norm holds. However, in their system explosion still ensues when faced with conflicting norms that hold for the same source and interest group. Such 'hardcore' normative conflicts are sometimes called *symmetrical* conflicts [21,27]. In order to consistently allow for the possibility of these conflicts in deontic logic, we need a non-standard formalism, i.e. a formalism that invalidates one or more of the theorems and rules of **SDL**.

[4] The negation of **MDP** is that of the paraconsistent logic **CLuNs** as found in e.g. [3,5].

F-Fac′. For all $A \in \mathcal{W}^a$, all $w \in W$ and all $J \subseteq_\emptyset I$, (i) if $w_J \in v_I(A)$ then $w \in v(A)$ and (ii) if $w_J \in v_I(\neg A)$ then $w \notin v(A)$ or $w \in v(\neg A)$.

The valuation $v_M : \mathcal{W}^c \to W$ is defined by C^l_I, $\mathrm{C}_I\wedge$, $\mathrm{C}_I\vee$, $\mathrm{C}_I\supset$, $\mathrm{C}_I\equiv$, $\mathrm{C}_I\neg\neg$, $\mathrm{C}_I\neg\vee$, $\mathrm{C}_I\neg\wedge$, $\mathrm{C}_I\neg\supset$, $\mathrm{C}_I\neg\equiv$, C^a, $\mathrm{C}\wedge$, $\mathrm{C}\vee$, $\mathrm{C}\supset$, $\mathrm{C}\equiv$, $\mathrm{C}\bot$, CO, CP, and the following:

$C\neg'$	where $A \in \mathcal{W}^l \cup \mathcal{W}^l_I : M, w \models \neg A$ iff $M, w \not\models A$ or $w \in v(\neg A)$
$C\neg\neg$	where $A \in \mathcal{W}^c : M, w \models \neg\neg A$ iff $M, w \models A$
$C\neg\vee$	where $A, B \in \mathcal{W}^c : M, w \models \neg(A \vee B)$ iff $M, w \models \neg A \wedge \neg B$
$C\neg\wedge$	where $A, B \in \mathcal{W}^c : M, w \models \neg(A \wedge B)$ iff $M, w \models \neg A \vee \neg B$
$C\neg\supset$	where $A, B \in \mathcal{W}^c : M, w \models \neg(A \supset B)$ iff $M, w \models A \wedge \neg B$
$C\neg\equiv$	where $A, B \in \mathcal{W}^c : M, w \models \neg(A \equiv B)$ iff $M, w \models (A \vee B) \wedge (\neg A \vee \neg B)$

As before, an **MDP**-model M verifies A ($M \Vdash_{\mathbf{MDP}} A$) iff $M, w^0 \models A$. Where $\Gamma \subseteq \mathcal{W}^c$, M is an **MDP**-model of Γ iff M is an **MDP**-model and $M \Vdash_{\mathbf{MDP}} A$ for all $A \in \Gamma$. Moreover, $\models_{\mathbf{MDP}} A$ iff all **MDP**-models verify A, and $\Gamma \models_{\mathbf{MDP}} A$ iff all **MDP**-models of Γ verify A.

$C\neg\neg, C\neg\wedge, C\neg\vee, C\neg\supset$, and $C\neg\equiv$ ensure that de Morgan laws and the double-negation rule are valid for complex formulas in $\mathcal{W}^c$. Due to the weakened negation of **MDP** this does not follow anymore from the other semantic clauses.

MDP allows for both A_J and $\neg(A_J)$ to be true at one and the same accessible world. Consequently, this logic can consistently model situations in which for an agent or group J it ought to be that J brings about A and that J refrains from bringing about A. In general, for any $A \in \mathcal{W}^c$, **MDP** allows for both A and $\neg A$ to be true at one and the same accessible world. This is exactly what we need if we also want to consistently allow for the possibility of contradicting norms.

Altogether, the paraconsistent multi-agent deontic logic **MDP** is fully conflict-tolerant (where $A \in \mathcal{W}$, and $B, C \in \mathcal{W}^c_I$):

OO-conflicts:	$\mathsf{O}A_J \wedge \mathsf{O}\neg(A_J) \not\models_{\mathbf{MDP}} \mathsf{O}C$,	$\mathsf{O}A_J \wedge \mathsf{O}(\neg A)_K \not\models_{\mathbf{MDP}} \mathsf{O}C$
OP-conflicts:	$\mathsf{O}A_J \wedge \mathsf{P}\neg(A_J) \not\models_{\mathbf{MDP}} \mathsf{O}C$,	$\mathsf{O}A_J \wedge \mathsf{P}(\neg A)_K \not\models_{\mathbf{MDP}} \mathsf{O}C$
Contradicting norms:	$\mathsf{O}B \wedge \neg\mathsf{O}B \not\models_{\mathbf{MDP}} \mathsf{O}C$,	$\mathsf{P}B \wedge \neg\mathsf{P}B \not\models_{\mathbf{MDP}} \mathsf{O}C$
Contradictory norms:	$\mathsf{O}(A_J \wedge \neg(A_J)) \not\models_{\mathbf{MDP}} \mathsf{O}C$,	$\mathsf{P}(A_J \wedge \neg(A_J)) \not\models_{\mathbf{MDP}} \mathsf{O}C$,
	$\mathsf{O}(A_J \wedge (\neg A)_K) \not\models_{\mathbf{MDP}} \mathsf{O}C$,	$\mathsf{P}(A_J \wedge (\neg A)_K) \not\models_{\mathbf{MDP}} \mathsf{O}C$,
	$\mathsf{O}(A \wedge \neg A)_J \not\models_{\mathbf{MDP}} \mathsf{O}C$,	$\mathsf{P}(A \wedge \neg A)_J \not\models_{\mathbf{MDP}} \mathsf{O}C$

5 Drawbacks of MDP

In an **MDP**-model, accessible worlds can consistently verify contradictions. This is what causes **MDP** to avoid deontic explosion when faced with a normative conflict. However, this property comes at a cost. We illustrate this by means of an example. Suppose that Frank has baked cookies and that it's hot in his kitchen. In order to let some fresh air in, Frank ought to open the door or open

the window ($\mathsf{O}(D \vee W)_f$). However, if someone opens the door, the neighbour's dog might smell Frank's cookies and try to steal them from the table. Hence Frank should take care that the door remains closed ($\mathsf{O}(\neg D)_f$). In this situation Frank can consistently satisfy his obligations by simply opening the window.

Yet $\mathsf{O}W_f$ is not **MDP**-derivable from $\mathsf{O}(D \vee W)_f$ and $\mathsf{O}(\neg D)_f$. Note that there are **MDP**-models of the premise set $\Gamma_1 = \{\mathsf{O}(D \vee W)_f, \mathsf{O}(\neg D)_f\}$ in which inconsistent worlds are accessible from the actual world, i.e. worlds in which both D_f and $(\neg D)_f$ are true (and, consequently, in which both D and $\neg D$ are true). In these worlds, W_f may be false while the premises are true. In contrast, all the **MDC**-models of our premise set Γ_1 are such that all the accessible worlds are consistent and verify $(D \vee W)_f$, $(\neg D)_f$ and hence W_f. This is the reason why $\Gamma_1 \models_{\mathbf{MDC}} \mathsf{O}W_f$ while $\Gamma_1 \not\models_{\mathbf{MDP}} \mathsf{O}W_f$. Obviously our premise set is not conflicting. In such cases we would ideally expect from any deontic logic that its models do not verify normative conflicts. Hence, in our case we are interested in **MDP**-models that –just like the **MDC**-models– do not validate $D_f \wedge (\neg D)_f$ in any of the accessible worlds, i.e. models M for which $M \Vdash_{\mathbf{MDP}} P(D_f \wedge (\neg D)_f)$. It is easy to see that all these models validate W_f in all the accessible worlds, just like the **MDC**-models. In other words, since $\Gamma_1 \models_{\mathbf{MDP}} \mathsf{O}W_f \vee P(D_f \wedge (\neg D)_f)$ we get $\mathsf{O}W_f$ by selecting models that do not validate any normative conflicts.

The solution offered above is obviously not working as soon as we have to deal with conflicting premise sets. Suppose Frank invited his aunt Maggie for a cup of coffee and cookies in the afternoon. However, his other aunt Beth is an awfully jealous person: she would be deeply insulted if she's not also invited. Hence Frank has the obligation to also invite Beth ($\mathsf{O}B_f$). On the other hand, Maggie cannot stand Beth (she's a rather difficult person) and whenever they are together all hell breaks loose. Thus, Frank should make sure that Beth is not invited ($\mathsf{O}(\neg B)_f$). Let $\Gamma_2 = \Gamma_1 \cup \{\mathsf{O}B_f, \mathsf{O}(\neg B)_f\}$. While **MDC** trivializes Γ_2, **MDP** does not trivialize Γ_2 but is again too weak. For the same reason as above, $\Gamma_2 \not\models_{\mathbf{MDP}} \mathsf{O}W_f$. However, in contrast to above we cannot now simply select models whose worlds are consistent since there are no such models. Indeed, all models of Γ_2 are such that in all accessible worlds B_f and $(\neg B)_f$ are valid. In other words, all models validate $\mathsf{O}(B_f \wedge (\neg B)_f)$. But, similar to above, the idea is to not take into consideration models that validate $\mathsf{P}(D_f \wedge (\neg D)_f)$.

In a nutshell the idea is to strengthen **MDP** by selecting models whose accessible worlds are "as non-conflicting as possible". This idea will be realized by means of the adaptive logic $\mathbf{MDP^m}$.

Before we introduce this logic in Section 6, let us focus on some other weaknesses of **MDP**. For instance, all of the following inferences are invalid in **MDP**, for the same reason why $\mathsf{O}W_f$ is not **MDP**-derivable from $\mathsf{O}(D \vee W)_f$ and $\mathsf{O}(\neg D)_f$: because of the possibility of contradictions being true in accessible worlds in **MDP**-models.[5] Where $A, B \in \mathcal{W}$, and $C, D \in \mathcal{W}^c$:

(1) $\mathsf{O}(A \vee B)_J, \mathsf{O}(\neg A)_J \not\models_{\mathbf{MDP}} \mathsf{O}B_J$

[5] These problems are common to monotonic logics with a paraconsistent negation. In [6], it was argued that the paraconsistent deontic logics presented in [11,24,25] are too weak to account for deontic reasoning.

(2) $\mathsf{O}(A \vee B)_J, \mathsf{O}\neg(A_J) \not\models_{\mathbf{MDP}} \mathsf{O}B_J$
(3) $\mathsf{O}A_J \not\models_{\mathbf{MDP}} \neg\mathsf{P}\neg(A_J)$
(4) $\neg\mathsf{O}A_J \not\models_{\mathbf{MDP}} \mathsf{P}\neg(A_J)$
(5) $\mathsf{P}A_J \not\models_{\mathbf{MDP}} \neg\mathsf{O}\neg(A_J)$
(6) $\neg\mathsf{P}A_J \not\models_{\mathbf{MDP}} \mathsf{O}\neg(A_J)$
(7) $C \vee D, \neg C \not\models_{\mathbf{MDP}} D$
(8) $C \supset D \not\models_{\mathbf{MDP}} \neg D \supset \neg C$

Items (1) and (2) represent deontic variants of Disjunctive Syllogism, (3)–(6) represent variants of the interdefinability between obligations and permissions, (7) is the propositional version of Disjunctive Syllogism, and (8) is Contraposition. In contrast, the following inferences *are* valid in **MDP**.

(1') $\mathsf{O}(A \vee B)_J, \mathsf{O}(\neg A)_J \models_{\mathbf{MDP}} \mathsf{O}B_J \vee \mathsf{P}(A_J \wedge (\neg A)_J)$
(2') $\mathsf{O}(A \vee B)_J, \mathsf{O}\neg(A_J) \models_{\mathbf{MDP}} \mathsf{O}B_J \vee \mathsf{P}(A_J \wedge \neg(A_J))$
(3') $\mathsf{O}A_J \models_{\mathbf{MDP}} \neg\mathsf{P}\neg(A_J) \vee \mathsf{P}(A_J \wedge \neg(A_J))$
(4') $\neg\mathsf{O}A_J \models_{\mathbf{MDP}} \mathsf{P}\neg(A_J) \vee (\mathsf{O}A_J \wedge \neg\mathsf{O}A_J)$
(5') $\mathsf{P}A_J \models_{\mathbf{MDP}} \neg\mathsf{O}\neg(A_J) \vee \mathsf{P}(A_J \wedge \neg(A_J))$
(6') $\neg\mathsf{P}A_J \models_{\mathbf{MDP}} \mathsf{O}\neg(A_J) \vee (\mathsf{P}A_J \wedge \neg\mathsf{P}A_J)$
(7') $C \vee D, \neg C \models_{\mathbf{MDP}} D \vee (C \wedge \neg C)$
(8') $C \supset D \models_{\mathbf{MDP}} (\neg D \supset \neg C) \vee (D \wedge \neg D)$

In items (1')–(8'), all formulas on the right-hand side of the "$\vee$"-sign represent normative conflicts. As in our example above, interpreting premise sets as non-conflicting as possible will validate the deontic and propositional versions of Disjunctive Syllogism, the interdefinability between obligations and permissions, and Contraposition as much as possible. Indeed, given that the normative conflicts on the right-hand side of "$\vee$" are false in (1')–(8'), the left-hand disjuncts must be true.

6 The Adaptive Logic MDP$^{\mathbf{m}}$

Adaptive logics are characterized by means of a triple consisting of a lower limit logic (henceforth LLL), a set of abnormalities Ω, and an adaptive strategy.[6] The LLL constitutes the stable part of an adaptive logic: everything that is LLL-derivable from a given set of premises, is still derivable by means of the adaptive logic. Formulating adaptive logics in the standard format has the advantage that a rich meta-theory is immediately available for this format [4]. Although adaptive logics come with an attractive dynamic proof theory we will for the sake of conciseness focus in this paper exclusively on the semantics.

Typically, an adaptive logic enables one to derive, for most premise sets, some extra consequences on top of those that are LLL-derivable. These supplementary

[6] For an introduction to adaptive logics, see [4]. However, familiarity with this framework for non-monotonic reasoning is not necessary for understanding the workings of the logic **MDP$^{\mathbf{m}}$**.

consequences are obtained by interpreting a premise set "as normally as possible". The exact interpretation of this idea depends on the adaptive strategy which defines which models of the LLL are selected.[7] For our present purposes, we shall use the Minimal Abnormality strategy. The logic $\mathbf{MDP^m}$ is characterized by:

(1) LLL: **MDP**

(2) Set of abnormalities: $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$, where
$\Omega_1 = \{A \wedge \neg A \mid A \in \mathcal{W}^c\}$
$\Omega_2 = \{\mathsf{P}(A_J \wedge \neg(A_J)) \mid A \in \mathcal{W}, J \subseteq_\emptyset I\}$
$\Omega_3 = \{\mathsf{P}(A_J \wedge (\neg A)_K) \mid A \in \mathcal{W}; J, K \subseteq_\emptyset I\}$

(3) Adaptive strategy: Minimal Abnormality

By (1) we make sure that we select **MDP**-models. This ensures that $\mathbf{MDP^m}$ inherits the conflict-tolerance from **MDP**.

As mentioned, adaptive logics interpret premise sets in a way that as few abnormalities as possible are verified. The attentive reader will have noticed that not all conflict-types that were listed in the table in Section 3 occur in Ω. This is justified due to the fact that all other conflict-types give rise to abnormalities in Ω, as the following table shows (where $A \in \mathcal{W}$, and $B, C \in \mathcal{W}_I^c$)[8]:

$$\begin{array}{rr}
\mathsf{O}A_J, \mathsf{O}\neg(A_J) \models_{\mathbf{MDP}} \mathsf{P}(A_J \wedge \neg(A_J)), & \mathsf{O}A_J, \mathsf{O}(\neg A)_K \models_{\mathbf{MDP}} \mathsf{P}(A_J \wedge (\neg A)_K) \\
\mathsf{O}A_J, \mathsf{P}\neg(A_J) \models_{\mathbf{MDP}} \mathsf{P}(A_J \wedge \neg(A_J)), & \mathsf{O}A_J, \mathsf{P}(\neg A)_K \models_{\mathbf{MDP}} \mathsf{P}(A_J \wedge (\neg A)_K) \\
\mathsf{O}(A_J \wedge \neg(A_J)) \models_{\mathbf{MDP}} \mathsf{P}(A_J \wedge \neg(A_J)), & \mathsf{O}(A_J \wedge (\neg A)_K) \models_{\mathbf{MDP}} \mathsf{P}(A_J \wedge (\neg A)_K) \\
\mathsf{O}(A \wedge \neg A)_J \models_{\mathbf{MDP}} \mathsf{P}(A_J \wedge (\neg A)_J), & \mathsf{P}(A \wedge \neg A)_J \models_{\mathbf{MDP}} \mathsf{P}(A_J \wedge (\neg A)_J)
\end{array}$$

For our semantic selection we will make use of the notion of the *abnormal part* of an **MDP**-model, i.e. the set of all abnormalities verified by it: $Ab(M) = \{A \in \Omega \mid M \Vdash_{\mathbf{MDP}} A\}$. The Minimal Abnormality strategy selects all **MDP**-models of a premise set Γ which have a *minimal* abnormal part (w.r.t. set-inclusion).

Definition 1. *An* **MDP***-model M of Γ is* minimally abnormal *iff there is no* **MDP***-model M' of Γ such that $Ab(M') \subset Ab(M)$.*

The semantic consequence relation of the logic $\mathbf{MDP^m}$ is defined by selecting the minimally abnormal **MDP**-models:

Definition 2. *$\Gamma \models_{\mathbf{MDP^m}} A$ iff A is verified by all minimally abnormal* **MDP***-models of Γ.*

The fact that the set of $\mathbf{MDP^m}$-models of Γ is a subset of the set of **MDP**-models of Γ immediately ensures that $\mathbf{MDP^m}$ strengthens **MDP**.

Theorem 1. *If $\Gamma \models_{\mathbf{MDP}} A$, then $\Gamma \models_{\mathbf{MDP^m}} A$.*

[7] Besides adaptive logics many other formal frameworks make use of semantic selections, e.g. [20,26].

[8] conflicts of the form $\mathsf{O}B \wedge \neg\mathsf{O}B$, $\mathsf{P}B \wedge \neg\mathsf{P}B$, $\mathsf{P}(A_J \wedge \neg(A_J))$, or $\mathsf{P}(A_J \wedge (\neg A)_K)$ are not listed in the table, since these conflicts already have the form of an abnormality.

For an illustration of the logic, let's return to the example presented in Section 5. Remember that $\Gamma_1 \not\models_{\mathbf{MDP}} \mathsf{O}W_f$. However, $\Gamma_1 \models_{\mathbf{MDP}} \mathsf{O}W_f \vee \mathsf{P}(D_f \wedge (\neg D)_f)$. By C$\vee$, we know that (†) for all **MDP**-models M of Γ_1: if $M \not\Vdash_{\mathbf{MDP}} \mathsf{P}(D_f \wedge (\neg D)_f)$ then $M \Vdash_{\mathbf{MDP}} \mathsf{O}W_f$.

No abnormality $A \in \Omega$ is an **MDP**-consequence of Γ_1, hence there are **MDP**-models M of Γ_1 such that $Ab(M) = \emptyset$. By Definition 1, these and only these are the minimal abnormal models of Γ_1. It follows that, for all minimal abnormal models M of Γ_1, $M \not\Vdash_{\mathbf{MDP}} \mathsf{P}(D_f \wedge (\neg D)_f)$. By (†), it follows that $M \Vdash_{\mathbf{MDP}} \mathsf{O}W_f$ for all minimal abnormal models M of Γ_1. Hence, by Definition 2, $\Gamma_1 \models_{\mathbf{MDP^m}} \mathsf{O}W_f$.

By the same reasoning as applied in the example above, we can show that all of (1")-(8") below are $\mathbf{MDP^m}$-valid in view of the **MDP**-validity of (1')-(8') as displayed in Section 5:

$$
\begin{array}{lr}
(1") & \mathsf{O}(A \vee B)_J, \mathsf{O}(\neg A)_J \models_{\mathbf{MDP^m}} \mathsf{O}B_J \\
(2") & \mathsf{O}(A \vee B)_J, \mathsf{O}\neg(A_J) \models_{\mathbf{MDP^m}} \mathsf{O}B_J \\
(3") & \mathsf{O}A_J \models_{\mathbf{MDP^m}} \neg\mathsf{P}\neg(A_J) \\
(4") & \neg\mathsf{O}A_J \models_{\mathbf{MDP^m}} \mathsf{P}\neg(A_J) \\
(5") & \mathsf{P}A_J \models_{\mathbf{MDP^m}} \neg\mathsf{O}\neg(A_J) \\
(6") & \neg\mathsf{P}A_J \models_{\mathbf{MDP^m}} \mathsf{O}\neg(A_J) \\
(7") & C \vee D, \neg C \models_{\mathbf{MDP^m}} D \\
(8") & C \supset D \models_{\mathbf{MDP^m}} \neg D \supset \neg C
\end{array}
$$

In a similar fashion, we can show that other intuitive **MDC**-inferences are also $\mathbf{MDP^m}$-valid in the absence of normative conflicts. Remember from Section 2.3 that $\mathsf{O}(\neg A)_i, \mathsf{O}(A \vee B)_{\{i,j\}} \models_{\mathbf{MDC}} \mathsf{O}B_{\{i,j\}}$ and $\mathsf{O}(\neg A)_i, \mathsf{O}^{\mathsf{w}}(A \vee B)_{\{i,j\}} \models_{\mathbf{MDC}} \mathsf{O}^{\mathsf{w}}B_{\{i,j\}}$. Both of these inferences are invalidated by **MDP**. However, $\mathsf{O}(\neg A)_i$, $\mathsf{O}(A \vee B)_{\{i,j\}} \models_{\mathbf{MDP}} \mathsf{O}B_{\{i,j\}} \vee \mathsf{P}(A_{\{i,j\}} \wedge (\neg A)_i)$, and $\mathsf{O}(\neg A)_i, \mathsf{O}^{\mathsf{w}}(A \vee B)_{\{i,j\}} \models_{\mathbf{MDP}} \mathsf{O}^{\mathsf{w}}B_{\{i,j\}} \vee \mathsf{P}(A_i \wedge (\neg A)_i) \vee \mathsf{P}(A_j \wedge (\neg A)_i) \vee \mathsf{P}(A_{\{i,j\}} \wedge (\neg A)_i)$. Note that none of the minimal abnormal **MDP**-models of $\{\mathsf{O}(\neg A)_i, \mathsf{O}(A \vee B)_{\{i,j\}}\}$ and $\{\mathsf{O}(\neg A)_i, \mathsf{O}^{\mathsf{w}}(A \vee B)_{\{i,j\}}\}$ validate one of the abnormalities $\mathsf{P}(A_{\{i,j\}} \wedge (\neg A)_i), \mathsf{P}(A_i \wedge (\neg A)_i)$, or $\mathsf{P}(A_j \wedge (\neg A)_i)$. Hence $\mathsf{O}(\neg A)_i, \mathsf{O}(A \vee B)_{\{i,j\}} \models_{\mathbf{MDP^m}} \mathsf{O}B_{\{i,j\}}$ and $\mathsf{O}(\neg A)_i, \mathsf{O}^{\mathsf{w}}(A \vee B)_{\{i,j\}} \models_{\mathbf{MDP^m}} \mathsf{O}^{\mathsf{w}}B_{\{i,j\}}$.

The following theorem shows that for any **MDC**-consistent premise set the $\mathbf{MDP^m}$-consequences are identical to the **MDC**-consequences:

Theorem 2. *For all* **MDC***-consistent* Γ, $\Gamma \models_{\mathbf{MDP^m}} A$ *iff* $\Gamma \models_{\mathbf{MDC}} A$.

A proof of Theorem 2 is contained in the Appendix. Note that (1")-(8") immediately follow as a corollary to Theorem 2.

If all **MDP**-models of given a premise set verify at least one abnormality, then $\mathbf{MDP^m}$ is still considerably stronger than **MDP**. Consider the premise set Γ_2 from Section 5, where we enriched Γ_1 with the conflicting obligations concerning the invitation of aunt Beth, $\mathsf{O}B_f$ and $\mathsf{O}(\neg B)_f$. Here too, Frank's obligation to open the window is an $\mathbf{MDP^m}$-consequence: $\Gamma_2 \models_{\mathbf{MDP^m}} \mathsf{O}W_f$. Although there are no models of Γ_2 that have an empty abnormal part since all models validate

the abnormality $\mathsf{P}(B_f \wedge (\neg B)_f)$, the minimal abnormal models do not validate $\mathsf{P}(D_f \wedge (\neg D)_f)$ (as the reader can easily verify).

Imagine now that we add to Γ_2 the premise $\mathsf{O}(\neg W)_f$, which abbreviates Frank's obligation to take care that the window remains closed (e.g. because it was painted recently and the paint is not dry yet). Let us call this extended premise set Γ_3. Then $\Gamma_3 \models_{\mathbf{MDP}} \mathsf{P}(D_f \wedge (\neg D)_f) \vee \mathsf{P}(W_f \wedge (\neg W)_f)$. Consequently, all minimally abnormal **MDP**-models M of Γ_3 verify at least one of $\mathsf{P}(D_f \wedge (\neg D)_f)$ and $\mathsf{P}(W_f \wedge (\neg W)_f)$. Γ_3 has minimally abnormal **MDP**-models which verify $\mathsf{P}(D_f \wedge (\neg D)_f)$. Since it is no longer the case that, for all minimally abnormal **MDP**-models M of Γ_3, $M \Vdash_{\mathbf{MDP}} \mathsf{P}(D_f \wedge (\neg D)_f)$, for these models it no longer follows that $M \Vdash_{\mathbf{MDP}} \mathsf{O}W_f$. Hence $\Gamma_3 \not\models_{\mathbf{MDP^m}} \mathsf{O}W_f$. Since $\Gamma_1 \subset \Gamma_3$ and $\Gamma_1 \models_{\mathbf{MDP^m}} \mathsf{O}W_f$, this shows that the logic $\mathbf{MDP^m}$ is non-monotonic.

The following theorems state some further meta-theoretical properties of $\mathbf{MDP^m}$. Let $\mathcal{M}_\Gamma^{\mathbf{MDP}}$ [$\mathcal{M}_\Gamma^{\mathbf{MDP^m}}$] abbreviate the set of **MDP**- [$\mathbf{MDP^m}$-] models of Γ.

Theorem 3. *If $M \in \mathcal{M}_\Gamma^{\mathbf{MDP}} - \mathcal{M}_\Gamma^{\mathbf{MDP^m}}$, then there is a $M' \in \mathcal{M}_\Gamma^{\mathbf{MDP^m}}$ such that $Ab(M') \subset Ab(M)$ (Strong reassurance).*

For the proof of Theorem 3, we refer to [4].[9]

Theorem 4. *If $\Gamma \models_{\mathbf{MDP^m}} A$ for all $A \in \Gamma'$, then $\mathcal{M}_\Gamma^{\mathbf{MDP^m}} = \mathcal{M}_{\Gamma\cup\Gamma'}^{\mathbf{MDP^m}}$.*

Proof. Suppose (†) $\Gamma \models_{\mathbf{MDP^m}} A$ for all $A \in \Gamma'$. Consider a $M \in \mathcal{M}_{\Gamma\cup\Gamma'}^{\mathbf{MDP^m}}$. Then $M \in \mathcal{M}_{\Gamma\cup\Gamma'}^{\mathbf{MDP}}$ and whence $M \in \mathcal{M}_\Gamma^{\mathbf{MDP}}$. Assume $M \notin \mathcal{M}_\Gamma^{\mathbf{MDP^m}}$. By the strong reassurance there is a $M' \in \mathcal{M}_\Gamma^{\mathbf{MDP^m}}$ such that $Ab(M') \subset Ab(M)$. In view of (†), $M' \Vdash_{\mathbf{MDP}} A$ for every $A \in \Gamma'$. Hence, $M' \in \mathcal{M}_{\Gamma\cup\Gamma'}^{\mathbf{MDP}}$. But then $M \notin \mathcal{M}_{\Gamma\cup\Gamma'}^{\mathbf{MDP^m}}$,— a contradiction.

Consider a $M \in \mathcal{M}_\Gamma^{\mathbf{MDP^m}}$. By (†), $M \Vdash_{\mathbf{MDP}} A$ for every $A \in \Gamma'$. By definition also $M \in \mathcal{M}_\Gamma^{\mathbf{MDP}}$. Hence $M \in \mathcal{M}_{\Gamma\cup\Gamma'}^{\mathbf{MDP}}$. Assume $M \notin \mathcal{M}_{\Gamma\cup\Gamma'}^{\mathbf{MDP^m}}$. Hence, there is a $M' \in \mathcal{M}_{\Gamma\cup\Gamma'}^{\mathbf{MDP}}$ for which $Ab(M') \subset Ab(M)$. By definition, $M \in \mathcal{M}_\Gamma^{\mathbf{MDP}}$. But then $M \notin \mathcal{M}_\Gamma^{\mathbf{MDP^m}}$,— a contradiction.

Corollary 1. *If $\Gamma \models_{\mathbf{MDP^m}} A$ for all $A \in \Gamma'$, then*

(i) if $\Gamma \models_{\mathbf{MDP^m}} A$ then $\Gamma \cup \Gamma' \models_{\mathbf{MDP^m}} A$ (Cautious Monotonicity);
(ii) if $\Gamma \cup \Gamma' \models_{\mathbf{MDP^m}} A$ then $\Gamma \models_{\mathbf{MDP^m}} A$ (Cautious Cut).

Theorem 5. *$\mathcal{M}_\Gamma^{\mathbf{MDP^m}} = \mathcal{M}_{\{B \mid \Gamma \models_{\mathbf{MDP^m}} B\}}^{\mathbf{MDP^m}}$, and whence $\Gamma \models_{\mathbf{MDP^m}} A$ iff $\{B \mid \Gamma \models_{\mathbf{MDP^m}} B\} \models_{\mathbf{MDP^m}} A$ (Fixed point).*

Proof. Since obviously $\{B \mid \Gamma \models_{\mathbf{MDP^m}} B\} \models_{\mathbf{MDP^m}} A$ for all $A \in \Gamma$, this is an immediate consequence of Theorem 4 (where $\Gamma' = \{B \mid \Gamma \models_{\mathbf{MDP^m}} B\}$).

[9] In [4], the strong reassurance property is proven for logics that fit the so-called standard format for adaptive logics. In order for the proof for strong reassurance from [4] to work, $\mathbf{MDP^m}$ needs to contain all classical connectives. $\mathbf{MDP^m}$ can easily be adjusted to do so by adding the constant false symbol "$\bot$" to its language, and by defining a classical negation connective "$\sim$" as $\sim A =_{\mathrm{df}} A \supset \bot$.

7 Outlook

The central problem tackled in this paper is the modeling of normative conflicts in multi-agent deontic logic. Because of this focus and reasons of conciseness, we have presented the logics **MDC**, **MDP** and $\mathbf{MDP^m}$ in a very basic form. In this section we will briefly demonstrate that they can be enhanced in various ways.

Some may wish to increase the expressiveness of our logics by alethic modalities. One way to technically realize this is to add another accessibility relation R' to the **MDC**-models so that the models are tuples $\langle W, I, R, R', v, w^0\rangle$.[10] Validity for the $\Box$-operator is characterized as usual: $M, w \models \Box A$ iff for all $w' \in W$, if $R'ww'$ then $M, w' \models A$ (and the dual version for $\Diamond$). By requiring $R \subseteq R'$ it could be ensured that the Kantian "ought implies can" holds: $\mathsf{O}A \supset \Diamond A$.

Another extension could indicate the authority that issues a norm. For instance, $\mathsf{O}^a A_J$ reads "authority a issues the norm that J brings about A". Technically, introducing authorities is straightforward. First, we enhance our models by a set A of authorities. This set may intersect with or even be identical to the set of agents I. Second, instead of one accessibility relation we introduce an accessibility relation R^a for each authority $a \in A$. The semantic clauses are adjusted as expected: $M, w \models \mathsf{O}^a A$ iff for all $w' \in W$, if $R^a ww'$ then $M, w' \models A$ (and dually for P^a).

In a way technically analogous to the representation of different authorities via superscripts to the deontic operators, we could add subscripts for distinguishing between various interest groups in view of which the norms hold (cfr. [19]). Moreover, the adaptive framework could be enhanced so as to allow for varying degrees of priority amongst norms and/or conditional norms [29].

The framework used in this paper is elementary not only in its limited expressive power, but also in its treatment of the notions of action and agency. At the moment, this paper is lacking a comparison with other frameworks for representing agency in deontic logic. Further research includes (i) the relation of the agentive setting applied here with other such settings, e.g. dynamic logic [9,23], stit theory [17,19], and their historical predecessors [10,33,34]; and (ii) the application of the inconsistency-adaptive approach for accommodating normative conflicts within these other frameworks for accounting for action in deontic logic.

References

1. Alchourrón, C.E.: Logic of norms and logic of normative propositions. Logique & Analyse 47, 242–268 (1969)
2. Alchourrón, C.E., Bulygin, E.: The expressive conception of norms. In: Hilpinen, R. (ed.) New Studies in Deontic Logic, pp. 95–124. D. Reidel Publishing Company, Dordrecht (1981)

[10] We will exemplify all enhancements by means of **MDC**. The arguments are analogous for **MDP** and $\mathbf{MDP^m}$.

3. Batens, D.: A survey of inconsistency-adaptive logics. In: Batens, D., Priest, G., van Bendegem, J.-P. (eds.) Frontiers of Paraconsistent Logic, pp. 49–73. Research Studies Press, Kings College Publication, Baldock (2000)
4. Batens, D.: A universal logic approach to adaptive logics. Logica Universalis 1, 221–242 (2007)
5. Batens, D., Meheus, J.: Recent results by the inconsistency-adaptive labourers. In: Béziau, J.-Y., Carnielli, W., Gabbay, D. (eds.) Handbook of Paraconsistency, pp. 81–99. College Publications, London (2007)
6. Beirlaen, M., Meheus, J., Straßer, C.: An inconsistency-adaptive deontic logic for normative conflicts. Under review
7. Boella, G., Van Der Torre, L., Verhagen, H.: Introduction to the special issue on normative multiagent systems. Autonomous Agents and Multi-Agent Systems 17, 1–10 (2008)
8. Boella, G., van der Torre, L.: Permissions and obligations in hierarchical normative systems. In: Proceedings of the 9th International Conference On Artificial Intelligence And Law, ICAIL 2003, pp. 109–118. ACM, New York (2003)
9. Broersen, J.: Action negation and alternative reductions for dynamic deontic logics. Journal of Applied Logic 2, 153–168 (2004)
10. Castañeda, H.-N.: The paradoxes of deontic logic: the simplest solution to all of them in one fell swoop. In: Hilpinen, R. (ed.) New Studies in Deontic Logic, pp. 37–85. D. Reidel Publishing Company, Dordrecht (1981)
11. Da Costa, N., Carnielli, W.: On paraconsistent deontic logic. Philosophia 16, 293–305 (1986)
12. Dignum, F., Royakkers, L.: Collective commitment and obligation. In: Ciampi, C., Marinai, E. (eds.) Proceedings of 5th Int. Conference on Law in the Information Society, Firenze, Italy, pp. 1008–1021 (1998)
13. Goble, L.: Multiplex semantics for deontic logic. Nordic Journal of Philosophical Logic 5(2), 113–134 (2000)
14. Goble, L.: A logic for deontic dilemmas. Journal of Applied Logic 3, 461–483 (2005)
15. Hansen, J., Pigozzi, G., van der Torre, L.: Ten philosophical problems in deontic logic. In: Boella, G., van der Torre, L., Verhagen, H. (eds.) Normative Multi-agent Systems, Germany. Dagstuhl Seminar Proceedings, vol. 07122. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Scholos Dagstuhl (2007)
16. Horty, J.F.: Moral dilemmas and nonmonotonic logic. Journal of Philosophical Logic 23(1), 35–66 (1994)
17. Horty, J.F.: Agency and Deontic Logic. Oxford University Press, Oxford (2001)
18. Horty, J.F.: Reasoning with moral conflicts. Noûs 37, 557–605 (2003)
19. Kooi, B., Tamminga, A.: Moral conflicts between groups of agents. Journal of Philosophical Logic 37, 1–21 (2008)
20. Kraus, S., Lehmann, D.J., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44, 167–207 (1990)
21. McConnell, T.: Moral dilemmas. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy (2010) (Summer 2010 edn.)
22. Meheus, J., Beirlaen, M., Van De Putte, F.: Avoiding deontic explosion by contextually restricting aggregation. In: Governatori, G., Sartor, G. (eds.) DEON 2010. LNCS(LNAI), vol. 6181, pp. 148–165. Springer, Heidelberg (2010)
23. Meyer, J.-J.: A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic. Notre Dame Journal of Formal Logic 29, 109–136 (1988)
24. Priest, G.: In Contradiction: A Study of the Transconsistent, 2nd edn. Oxford University Press, Oxford (2006)

25. Routley, R., Plumwood, V.: Moral dilemmas and the logic of deontic notions. In: Priest, G., Routley, R., Norman, J. (eds.) Paraconsistent Logic. Essays on the Inconsistent, pp. 653–702. Philosophia Verlag, München (1989)
26. Shoham, Y.: A semantical approach to nonmonotonic logics. In: Ginsberg, M.L. (ed.) Readings in Nonmonotonic Reasoning, pp. 227–250. Morgan Kaufmann Publishers, San Francisco (1987)
27. Sinnott-Armstrong, W.: Moral Dilemmas. Basil Blackwell, Oxford (1988)
28. Straßer, C.: An adaptive logic framework for conditional obligations and deontic dilemmas. Logic and Logical Philosophy (2010) (forthcoming)
29. Straßer, C.: A deontic logic framework allowing for factual detachment. Journal of Applied Logic 9(1), 61–80 (2011)
30. Straßer, C., Beirlaen, M.: Towards more conflict-tolerant deontic logics by relaxing the interdefinability between obligations and permissions. Under review
31. Straßer, C., Meheus, J., Beirlaen, M.: Tolerating deontic conflicts by adaptively restricting inheritance. Under review
32. van der Torre, L., Tan, Y.H.: Two-phase deontic logic. Logique et Analyse (171-172), 411–456 (2000)
33. von Wright, G.H.: Norm and Action. A Logical Enquiry. Routledge and Kegan Paul, London (1963)
34. von Wright, G.H.: On the logic of norms and actions. In: Hilpinen, R. (ed.) New Studies in Deontic Logic, pp. 3–35. D. Reidel Publishing Company, Dordrecht (1981)
35. von Wright, G.H.: Deontic logic: a personal view. Ratio Juris 12(1), 26–38 (1999)

A Appendix: Proof of Theorem 2

For every adaptive logic, there is a so-called *upper limit logic*. The upper limit logic **UMDP** of $\mathbf{MDP^m}$ is defined as follows: given a premise set Γ we select all **MDP**-models M of Γ such that $Ab(M) = \emptyset$. **UMDP** is a monotonic logic that trivializes premise sets that give rise to abnormalities.

Lemma 1. For each **UMDP**-model M of Γ, **F-Con** holds.

Proof. Let $M = \langle W, I, R, v, v_I, w^0\rangle$. Suppose for some $w \in W$, some $A \in \mathcal{W}^a$, and some $J, K \subseteq_\emptyset I$, $w_J \in v_I(A)$ and $w_K \in v_I(\neg A)$. By **F-Fac′**, $w \in v(A)$ and $w \in v(\neg A)$. If $w = w^0$, by C^a, C$\neg'$ and C$\wedge$, $M, w^0 \models A \wedge \neg A$ and hence $A \wedge \neg A \in Ab(M)$,—a contradiction. If $w \neq w^0$, then by C^a, C$\neg'$, C$\wedge$ and CP, $M, w^0 \models \mathsf{P}(A_J \wedge (\neg A)_K)$ and hence $\mathsf{P}(A_J \wedge (\neg A)_K) \in Ab(M)$,—a contradiction. Hence, **F-Con** holds. □

Let an **MDP**-model $\langle W, I, R, v, v_I, w^0\rangle$ be **MDC**-*like* iff, (a) for all $A \in \mathcal{W}^a$, $w \in v(\neg A)$ iff $w \notin v(A)$; (b) for all $A_J \in \mathcal{W}^l_I$, $w \in v(\neg(A_J))$ iff $w_J \notin v_I(A)$; and (c) **F-Fac** holds. We say that two models *are equivalent* iff they validate the same formulas.

Lemma 2. For each **UMDP**-model $M = \langle W, I, R, v, v_I, w_0\rangle$ there is an equivalent **MDC**-like **UMDP**-model $M' = \langle W, I, R, v', v_I, w_0\rangle$.

Proof. Define v' as follows: (1) where $A \in \mathcal{W}^l \cup \{\neg(B_J) \mid B \in \mathcal{W}^l\}$, $w^0 \in v'(A)$ iff $w^0 \in v(A)$; (2) where $w \in W \setminus \{w^0\}$ and $A \in \mathcal{W}^a$, $w \in v'(A)$ iff there is a $J \subseteq_\emptyset I$ for which $w_J \in v_I(A)$; (3) where $w \in W \setminus \{w^0\}$ and $\neg A \in \mathcal{W}^l$, $w \in v'(\neg A)$ iff $w \notin v'(A)$; and (4) where $A \in \mathcal{W}^l$, $w \in v'(\neg(A_J))$ iff $w_J \notin v_I(A)$.

F-Con only depends on v_I and hence holds for M' due to Lemma 1. **F-Fac′**(i) holds by (2) and (ii) by (3) and due to **F-Con**. Hence, M' is a **MDP**-model.

F-Fac (i) holds due to **F-Fac′** (i). Let $w_J \in v_I(\neg A)$. Suppose first that $w = w^0$. By **F-Fac′**, $w^0 \in v'(\neg A)$ or $w^0 \notin v'(A)$ and whence $w^0 \in v(\neg A)$ or $w^0 \notin v(A)$. Assume that $w^0 \in v(\neg A) \cap v(A)$. But then $A \wedge \neg A \in Ab(M)$,—a contradiction. Hence $w^0 \notin v(A)$ and whence $w^0 \notin v'(A)$. Let now $w \in W \setminus \{w^0\}$. By **F-Con**, there is no $K \subseteq_\emptyset I$ for which $w_K \in v_I(A)$. Hence, by (2), $w \notin v'(A)$. Thus, **F-Fac** holds for M'.

Note that (a) holds for v' due to (3), and (b) holds due to (4). Hence, M' is **MDC**-like.

$M \Vdash_{\mathbf{MDP}} A$ iff $M' \Vdash_{\mathbf{MDP}} A$ is shown by an induction over the length of the formula A. The induction base is easily established. Where $A \in \mathcal{W}^a$ the equivalence holds by C^a and (1). Where $A \in \mathcal{W}^l_I$ the equivalence holds due to C^l_I. For the induction step let first $A = \neg A'$. Suppose $A' \in \mathcal{W}^a \cup \mathcal{W}^l_I$. Note that by the induction hypothesis, $\mathrm{C}\neg'$ and (1) we have the same valuation for A. Let now $A' \in \mathcal{W}_I \setminus \mathcal{W}^l_I$. Since both models have the same assignment v_I the valuation is analogous due to C^l_I, $\mathrm{C}_I\wedge$, $\mathrm{C}_I\vee$, $\mathrm{C}_I\supset$, $\mathrm{C}_I\equiv$, $\mathrm{C}_I\neg\neg$, $\mathrm{C}_I\neg\vee$, $\mathrm{C}_I\neg\wedge$, $\mathrm{C}_I\neg\supset$, and $\mathrm{C}_I\neg\equiv$. The similar cases for $A' = B\pi C$ where $B, C \in \mathcal{W}^c$ and $\pi \in \{\vee, \wedge, \supset, \equiv\}$ resp. for $A' = \neg B$ where $B \in \mathcal{W}^c$ are left to the reader. The induction proceeds in a similar way if $A \in \mathcal{W}_I \setminus \mathcal{W}^l_I$, $A = \mathsf{O}A'$ or $A = \mathsf{P}A'$ where $A' \in \mathcal{W}^c_I$, or $A = B\pi C$ where $B, C \in \mathcal{W}^c$ and $\pi \in \{\vee, \wedge, \supset, \equiv\}$. Since M and M' are equivalent, $Ab(M') = Ab(M) = \emptyset$ and whence M' is an **UMDP**-model. □

Corollary 1. $\Gamma \models_{\mathbf{UMDP}} A$ iff each **MDC**-like **UMDP**-model M of Γ validates A.

Where $M = \langle W, I, R, v, v_I, w^0\rangle$ is an **MDC**-like **UMDP**-model, let $M_c = \langle W, I, R, v_c, v_I, w^0\rangle$ be an **MDC**-model where $v_c : \mathcal{W}^a \to \wp(W), A \mapsto v(A)$. Note that M_c is indeed an **MDC**-model since M satisfies **F-Con** and **F-Fac** and thus by the definition also M_c.

Lemma 3. M and M_c are equivalent.

The Lemma is proved by a similar induction over the length of A as in the proof of Lemma 2. Due to space restrictions this is left to the reader.

Lemma 4. Where M is an **MDC**-model of Γ, $Ab(M) = \emptyset$.

Proof. Suppose $A \in Ab(M)$. Let $A = B \wedge \neg B \in \Omega_1$. By C¬, $M \models B$ and $M \not\models B$,—a contradiction. The other cases are similar and left to the reader. □

Where $M = \langle W, I, R, v, v_I, w^0\rangle$ is an **MDC**-model, let $M_p = \langle W, I, R, v_p, v_I, w^0\rangle$ be an **MDC**-like **MDP**-model where $v_p : \mathcal{W}^l \cup \{\neg(A_J) \mid A \in \mathcal{W}^l, J \subseteq_\emptyset I\} \to \wp(W)$ is defined by: $w \in v_p(A)$ iff $M, w \models A$. The reader can easily verify that M_p is **MDC**-like.

Lemma 5. M and M_p are equivalent and M_p is a **UMDP**-model.

Again, the proof of the equivalence proceeds by a similar induction over the length of A as in the proof of Lemma 2. By Lemma 4, $Ab(M) = \emptyset$ and hence $Ab(M_p) = \emptyset$. Hence, M_p is a **UMDP**-model.

Theorem 2 is an immediate consequence of Corollary 1, Lemma 3 and Lemma 5.

Normative Systems Represented as Hybrid Knowledge Bases

Marco Alberti, Ana Sofia Gomes, Ricardo Gonçalves,
João Leite, and Martin Slota

CENTRIA & Departamento de Informática, Universidade Nova de Lisboa, Portugal

Abstract. Normative systems have been advocated as an effective tool to regulate interaction in multi-agent systems.

Logic programming rules intuitively correspond to conditional norms, and their semantics is based on the closed world assumption, which allows default negation, often used in norms. However, there are cases where the closed world assumption is clearly not adequate, and others that require reasoning about unknown individuals, which is not possible in logic programming.

On the other hand, description logics are based on the open world assumption and support reasoning about unknown individuals, but do not support default negation.

In this paper, we demonstrate the need for the aforementioned features (closed and open world assumptions, and reasoning about unknown individuals) in order to model human laws, with examples from the Portuguese Penal Code. We advocate the use of hybrid knowledge bases combining rules and ontologies, which provide the joint expressivity of logic programming and description logics.

We define a normative scenario as the pair of a set of facts and a set of norms, and give it a formal semantics by translation into an MKNF knowledge base.

We describe the implementation of the language, which computes the relevant consequences of given facts and norms, and use it to establish the resulting sentence in a penal scenario.

1 Introduction

In this paper we argue for the need to jointly use the Closed World Assumption based features of Logic Programming Rules, and the Open World Assumption based features of Description Logic based Ontologies, to represent and reason about Norms. We present a solution grounded on Hybrid MKNF Knowledge Bases [18], illustrate its use with an excerpt of the Portuguese Penal Code, and describe an efficient implementation.

Normative systems have long been advocated as an effective tool to regulate interaction in multi-agent systems [24], and the theory and practice of normative multi-agent systems constitutes a young, but very active, research area [6].

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 330–346, 2011.

Essentially, norms encode desirable behaviours for the population of natural or artificial societies. For example, a (conditional) norm might specify that drivers are expected to stop if so signaled by an authority. In general, they are commonly understood as a specification of what is expected to follow (obligations, goals, contingency plans, advices, actions, ...) from a specific state of affairs.

In practical multi-agent systems, norms are often implemented through electronic institutions which take a formal representation of the normative system and, through automated reasoning, check observable agents' behaviours in order, for instance, to detect norm violation and to apply sanctions.

One key problem to implement such practical normative systems involves the representation of, and reasoning with norms. If, on the one hand, we need a representation language that is expressive enough to represent the norms we wish to encode, on the other hand it must be such that we can reason with it efficiently. In this paper, we will take a closer look at the problem of norm representation and reasoning, trying to combine expressivity and efficiency, which has proved a difficult task in automated reasoning.

Despite the specificities of multi-agent systems, many of their aspects are inspired by human societies, and an intimate parallel between laws in real world legal systems and norms in multi-agent systems can often be drawn.

Ever since the formalisation of the British Nationality Act using Logic Programming by Sergot et al. [23], *non-monotonic* formalisms have been used to deal with many aspects of legal rules and regulations. Important work on this topic also includes the early use of argument-based extended logic programming with defeasible priorities by Prakken and Sartor [20] and the use of defeasible logic by Governatori et al. [12].

The non-monotonic features common to the languages used in these approaches, which implement the Closed-World Assumption, have been shown necessary in the context of reasoning with laws and regulations, for example to represent exceptions.

In this paper, instead of tailoring an artificial multi-agent based scenario to illustrate our points, we will use the Portuguese Penal Code which is filled with examples rich in intrinsic subtleties.

Example 1. The Portuguese Penal Code[1] defines the penalty for murder as follows:

> **Article 131. Murder**
> *Who kills another person shall be punished with imprisonment from eight to sixteen years.*

However, exceptional circumstances for murder increase the duration of the conviction:

[1] Translation by the authors. For the original text, in Portuguese, consult, for example, http://www.portolegal.com/CPENAL.htm

Article 132. Aggravated murder

1. *If death is produced in circumstances which present a special reprehensibility or perversity, the agent is punished with imprisonment of twelve to twenty-five years.*
2. *Is likely to reveal the special perversity or reprehensibility referred to in the preceding paragraph,* among others*, the fact that the agent:*
   - *(...)*
   - d) *employs torture or cruel act to increase the suffering of the victim;*
   - *(...)*
   - h) *performs the fact with at least two other people, or uses particularly dangerous means, or [means] which would result in the crime of common danger;*
   - *(...)*

Accordingly, killing someone is punished with imprisonment from eight to sixteen years, *except* if some additional facts are established, in which case the penalty is aggravated. In other words, unless one of these aggravating facts is proved, *by default* this crime is punished with imprisonment from eight to sixteen years. The relevant part can easily be captured by Logic Programming rules using non-monotonic default negation as follows:

$$\mathsf{AggravatedMurder}(X,Y) \leftarrow \mathsf{KillingBy}(X,Y), \mathsf{Censurable}(X).$$
$$\mathsf{Murder}(X,Y) \leftarrow \mathsf{KillingBy}(X,Y), {\sim}\mathsf{AggravatedMurder}(X,Y).$$

together with the definition of $\mathsf{Censurable}(X)$, which is the predicate representing the "special perversity or reprensehibility" referred to in the law.

The use of non-monotonic rule based languages founded on Logic Programming brings added value, as they also have well-studied computational properties and mature implementations.

However, in legal reasoning, we sometimes need to represent concepts that cannot be handled by the Logic Programming approach. There are cases where the Open World Assumption is needed and, more importantly, others where we need to deal with existential knowledge and unknown individuals.

Example 2. Going back to the previous example, encoding item h) as a condition to establish special perversity or reprehensibility requires that we refer to (at least) two possibly unknown individuals (a witness or a security camera recording could be sufficient to establish that the culprit acted together with two more people, but not their identity). The relevant part could be encoded in Description Logics as follows:

$$\mathsf{Censurable} \sqsupseteq (\geq 3\ \mathsf{PerformedBy.Person})$$

encoding that special censurability of the fact is established if it was committed by at least three people. Such a condition cannot be expressed in the body of a

logic programming rule since it does not permit encoding unknown individuals. Just as it would not be possible to assert that some fact was performed by e.g. five people, but whose identities, besides that of the accused, are unknown.

Description Logics [2] based ontology languages are based on the Open World Assumption and allow for reasoning with unknown individuals. Furthermore, they are quite appropriate for taxonomic representations of facts and concepts, and have been extensively used in legal reasoning [10]. However, they are monotonic and therefore lack the aforementioned important ability to model defeasible knowledge.

For all of these reasons, the representation and reasoning about normative systems, and in particular those inspired by human legal systems, seems to demand an approach that combines the best of the two families of formalisms, rules and ontologies, and exhibits, at least, the following characteristics:

- have a formal rigorous semantics so that agents and institutions can both reason about the norms to determine their actions and sanctions;
- support both the Open and Closed World Assumptions, and the ability to represent and reason with unknown individuals;
- be equipped with efficient operational semantics to be usable in practical multi-agent systems.

With these requirements in mind, in this paper we propose a language where facts are represented as a description logic ABox, and norms as a combination of description logic TBox and logic programming rules. We will then root our language on Hybrid MKNF [18], a language that tightly integrates rules and ontologies, and focus on its well founded version [1,14] for which we can exploit an efficient implementation [11].

The remainder of the paper is structured as follows. In Sect. 2, we present the language and the running example from the Portuguese penal code. In Sect. 3, we give the language a formal semantics by mapping it to Hybrid MKNF. In Sect. 4, we introduce the implementation of the reasoner for our language, and show how it correctly handles the running example. Conclusions and comments on future research follow.

2 Framework

In this section, we introduce the model of electronic institution that we envisage by analogy with the Portuguese judicial system; the institution's modus operandi motivates the choice of a language for a normative system.

In Portugal, the first phase of a penal trial is a discussion aimed at establishing the facts, where each of the parties (prosecution and defense) provides relevant evidence.

At the end of the debate, a list is made of all the facts that have been established in the debate. For example, it may have been established that (*i*) Bob attempted murder on Mary by means of a knife, and (*ii*) Alice defended Mary's life by killing Bob with a handgun.

Then, the judge applies the relevant laws to the facts and pronounces a sentence.

This procedure implies that the sentence depend on (*i*) facts established in court; and (*ii*) relevant laws.

A similar model is in fact applied to normative multi-agent systems, where an electronic institution automatically determines possible sanctions depending on the agents' behaviour (possibly detected by a runtime monitoring system) and the norms.

In order to enable the electronic institution to perform this task by means of automated reasoning, the language used to express the facts and the norms must have a formal semantics.

2.1 Language

We record both the facts about a particular legal case as well as the relevant norms for applying the law in a *judicial knowledge base*. The formalisation of this notion must build upon a formalism that, on the one hand, is capable of reasoning with unknown individuals, and, on the other hand, allows for expressing exceptions in a natural way. At the same time, the formalism should be amenable to an efficient implementation. The natural candidates for such formalisms are the integrative formalisms for Description Logics and non-monotonic rules. We proceed by briefly introducing the syntax of both these formalisms.

Throughout the paper we use a function-free first-order signature. By a *first-order atom* we mean a formula $P(t_1, t_2, \ldots, t_n)$ where P is a predicate symbol of arity n and $t_1, t_2, \ldots, t_n$ are first-order terms.

Description Logics. Description Logics (DLs) [2] are (usually) decidable fragments of first-order logic that are frequently used for knowledge representation and reasoning in applications. Throughout the paper we assume that some Description Logic is used to describe an ontology, i.e. it is used to specify a shared conceptualisation of a domain of interest. Basic building blocks of such a specification are *constants*, representing objects (or individuals), *concepts*, representing groups of objects, and *roles*, representing binary relations between objects and properties of objects. Typically, an ontology is composed of two distinguishable parts: a TBox specifying the required terminology, i.e. concept and role definitions, and an ABox with assertions about constants.

Most Description Logics can be equivalently translated into function-free first-order logic, with constants represented by constant symbols, atomic concepts represented by unary predicates and atomic roles represented by binary predicates. We assume that for any DL axiom ϕ, $\zeta(\phi)$ denotes such a translation of ϕ.

Logic Programs. We consider ground logic programs for specifying nonmonotonic domain knowledge. The basic syntactic blocks of such programs are ground atoms. A *default literal* is a ground atom preceded by $\sim$. A *literal* is either a ground atom or a default literal. A *rule* r is an expression of the form

$$p \leftarrow p_1, p_2, \ldots, p_m, \sim q_1, \sim q_2, \ldots, \sim q_n.$$

where m, n are natural numbers and $p, p_1, p_2, \ldots, p_m, q_1, q_2, \ldots, q_n$ are first-order atoms. The atom p is called the *head of* r and is denoted by $H(r)$; the set $\{ p_1, p_2, \ldots, p_m, \sim q_1, \sim q_2, \ldots, \sim q_n \}$ is called the *body of* r and is denoted by $B(r)$. A rule r is a *fact* if its body is empty. A *(normal) logic program* $\mathcal{P}$ is a finite set of rules.

Judicial Knowledge Base. As illustrated in Examples 1 and 2, a combination of TBox axioms and rules is sufficient to faithfully represent significant parts of real-world penal code. Also, ABox assertions are a natural candidate for describing the facts established in court. We henceforth define a judicial knowledge base as follows:

Definition 1. *A* judicial knowledge base *is a pair* $\langle \mathcal{F}, \mathcal{N} \rangle$ *where* $\mathcal{F}$ *is a set of ABox axioms and* $\mathcal{N}$ *is a set of TBox axioms and rules.*

2.2 Example

In this section, we demonstrate the expressivity of our language by encoding one example from real-world law, which would be problematic to express in either logic programming or description logic formalisms.

Together with the definitions of murder and aggravated murder reported in the introduction, consider the circumstances that exclude the illegality of an act listed in the Portuguese Penal Code.

Article 31. Precluding wrongfulness

1. *The act is not punishable when its wrongfulness is precluded by law considered in its entirety.*
2. *In particular, it (the act) is not unlawful if committed:*
   a) *In legitimate defense;*
   b) *In the exercise of a right;*
   c) *In fulfilling a duty imposed by law or lawful order of the authority, or*
   d) *With the consent of the holder of the harmed legal interest.*

Among such circumstances, legitimate defense is defined as follows.

Article 32. Legitimate defense
Legitimate defense is an act executed as necessary means to repel an actual and illicit aggression to legally protected interest of the executor or of a third party.

Suppose that the following was established in court by debate:

- John committed murder together with three more people;
- Mary killed someone who was pointing a gun at her;
- an unidentified person committed murder and torture.

A formalisation using a judicial knowledge base $\mathcal{K} = \langle \mathcal{F}, \mathcal{N} \rangle$ is in Figs. 1 and 2.

$$
\begin{aligned}
\mathsf{Censurable} &\sqsupseteq (\geq 3\ \mathsf{PerformedBy.Person}) \\
\mathsf{Censurable} &\sqsupseteq \mathsf{Torture} \\
\mathsf{AggravatedMurder}(X,Y) &\leftarrow \mathsf{KillingBy}(X,Y), \mathsf{Censurable}(X). \\
\mathsf{Murder}(X,Y) &\leftarrow \mathsf{KillingBy}(X,Y), {\sim}\mathsf{AggravatedMurder}(X,Y). \\
\mathsf{KillingBy}(X,Y) &\leftarrow \mathsf{Killing}(X), \mathsf{CrimeBy}(X,Y). \\
\mathsf{CrimeBy}(X,Y) &\leftarrow \mathsf{Illegal}(X), \mathsf{Guilt}(X,Y). \\
\mathsf{Illegal}(X) &\leftarrow \mathsf{Unlawful}(X), {\sim}\mathsf{ExceptionIllegal}(X). \\
\mathsf{Unlawful}(X) &\leftarrow \mathsf{Killing}(X). \\
\mathsf{ExceptionIllegal}(X) &\leftarrow \mathsf{JudicialOrder}(X). \\
\mathsf{ExceptionIllegal}(X) &\leftarrow \mathsf{ExerciseOfARight}(X). \\
\mathsf{ExceptionIllegal}(X) &\leftarrow \mathsf{WithConsentOfVictim}(X). \\
\mathsf{ExceptionIllegal}(X) &\leftarrow \mathsf{LegitimateDefense}(X). \\
\mathsf{LegitimateDefense}(X) &\leftarrow \mathsf{RelevantAggression}(Z), \mathsf{EffectiveReaction}(X,Z), \\
&\qquad {\sim}\mathsf{RequiredDifferentBehaviour}(X,Z). \\
\mathsf{Guilt}(X,Y) &\leftarrow \mathsf{Illegal}(X), \mathsf{PerformedBy}(X,Y), {\sim}\mathsf{ExceptionGuilt}(X,Y). \\
\mathsf{ExceptionGuilt}(X,Y) &\leftarrow \mathsf{RelevantThreat}(Z), \mathsf{EffectiveReaction}(X,Z), \\
&\qquad {\sim}\mathsf{RequiredDifferentBehaviour}(Y,X,Z). \\
\mathsf{JailSentence}(Y,8,16) &\leftarrow \mathsf{Murder}(X,Y), {\sim}\mathsf{AggravatedMurder}(X,Y). \\
\mathsf{JailSentence}(Y,12,25) &\leftarrow \mathsf{AggravatedMurder}(X,Y).
\end{aligned}
$$

Fig. 1. Set of relevant norms $\mathcal{N}$

$a : \mathsf{Act}$	$a : \mathsf{Killing}$	$a : (\geq 4\ \mathsf{PerformedBy.Person})$
$john : \mathsf{Person}$	$\langle a, john\rangle : \mathsf{PerformedBy}$	
$b : \mathsf{Act}$	$b : \mathsf{Killing}$	
$mary : \mathsf{Person}$	$\langle b, mary\rangle : \mathsf{PerformedBy}$	
$gun : \mathsf{RelevantThreat}$	$\langle b, gun\rangle : \mathsf{EffectiveReactionTo}$	
$c : \mathsf{Act}$	$c : \mathsf{Killing}$	$c : \mathsf{Torture}$

Fig. 2. Set of facts established in court $\mathcal{F}$

3 Formal Semantics

In order for judicial knowledge bases to be used in determining sanctions by means of automated reasoning in a rigorous and verifiable fashion, they need a

formal semantics. This semantics is established here by translating them into the logic of Minimal Knowledge and Negation as Failure (MKNF) [16,18]. In the following, we first briefly introduce the syntax and semantics of MKNF, adopting the well-founded MKNF model semantics introduced in [15] for which top-down querying procedures have been introduced and an implementation with support for the Description Logic $\mathcal{ALCQ}$ is available [11]. In Sect. 4, we use this implementation to represent and query the judicial knowledge base in Figs. 1 and 2.

3.1 Hybrid MKNF Knowledge Bases

Syntactically, MKNF is an extension of function-free first-order logic with two modal operators: **K** and **not**. A *first-order atom* $P(t_1, t_2, \ldots, t_n)$ is an MKNF formula; if ϕ is an MKNF formula, then $\neg\phi$, $\exists x : \phi$, $\mathbf{K}\,\phi$, and $\mathbf{not}\,\phi$ are MKNF formulas, and so are $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\phi_1 \supset \phi_2$, and $\phi_1 \equiv \phi_2$, if ϕ_1, ϕ_2 are MKNF formulas. An MKNF formula ϕ is a *sentence* if it has no free variables. By $\phi[t/x]$ we denote the formula obtained by simultaneously replacing in ϕ all free occurrences of variable x by term t.

Hybrid MKNF knowledge bases, as presented in [15], are sets of MKNF formulas restricted to a certain form. They consist of two components: a description logic knowledge base and a logic program.[2] Formally, a *hybrid knowledge base* is a pair $\langle \mathcal{O}, \mathcal{P} \rangle$ where $\mathcal{O}$ is a DL ontology and $\mathcal{P}$ is a logic program.

For interpreting hybrid MKNF knowledge bases in terms of the logic of MKNF, a transformation π is defined that translates both ontology axioms and rules into MKNF sentences. For any ground atom p, set of literals B, rule r with the vector of free variables $\boldsymbol{x}$, and hybrid knowledge base $\mathcal{K} = \langle \mathcal{O}, \mathcal{P} \rangle$, we define:

- $\pi(p) = \mathbf{K}\,p$,
- $\pi(\sim p) = \mathbf{not}\,p$,
- $\pi(B) = \{\, \pi(L) \mid L \in B \,\}$,
- $\pi(r) = (\forall \boldsymbol{x} : \bigwedge \pi(B(r)) \supset \pi(H(r)))$,
- $\pi(\mathcal{O}) = \bigwedge \{\, \zeta(\phi) \mid \phi \in \mathcal{O} \,\}$,
- $\pi(\mathcal{P}) = \bigwedge \{\, \pi(r) \mid r \in \mathcal{P} \,\}$,
- $\pi(\mathcal{K}) = \mathbf{K}\,\pi(\mathcal{O}) \wedge \pi(\mathcal{P})$.

Hybrid MKNF knowledge bases are in general undecidable, unless they are restricted in some way. The reason for that is that rules can be applied to all the objects in an infinite domain. The basic idea to make reasoning with hybrid MKNF knowledge bases decidable is to apply rules only to the individuals that appear in the knowledge base. This restriction is achieved by DL-safety [19]. Given a hybrid knowledge base $\langle \mathcal{O}, \mathcal{P} \rangle$, a first-order atom $P(t_1, t_2, \ldots, t_n)$ such that P occurs in $\mathcal{O}$ is called a *DL-atom*; all other atoms are called *non-DL-atoms*. A rule r is *DL-safe* if every variable in r occurs in at least one non-DL-atom p occurring positively in the body of r.

[2] In difference to [15], in this paper we consider only non-disjunctive rules.

3.2 Well-Founded MKNF Model

Following [18,15], we only consider Herbrand interpretations in our semantics and adopt the *standard names assumption*, so apart from the constants used in formulae, we assume our signature to contain a countably infinite supply of constants. The Herbrand Universe of such a signature is denoted by Δ. A *three-valued MKNF structure* $\langle I, \mathcal{M}, \mathcal{N} \rangle$ consists of a first-order interpretation I and two pairs $\mathcal{M} = \langle M, M_1 \rangle$ and $\mathcal{N} = \langle N, N_1 \rangle$ of sets of first-order interpretations where $M_1 \subseteq M$ and $N_1 \subseteq N$. Each three-valued MKNF structure assigns one of the truth values $\mathbf{t}$, $\mathbf{u}$ and $\mathbf{f}$ to all MKNF sentences. We use an ordering $\mathbf{f} < \mathbf{u} < \mathbf{t}$ of these truth values and operators max and min that, respectively, choose the greatest and least element with respect to this ordering. The truth value of a ground atom $P(t_1, t_2, \ldots, t_n)$ and of an MKNF sentence ϕ in a three-valued MKNF structure $\langle I, \mathcal{M}, \mathcal{N} \rangle$ is defined as follows:

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(P(t_1, t_2, \ldots, t_n)) = \begin{cases} \mathbf{t} & \text{iff } \langle t_1^I, t_2^I, \ldots, t_n^I \rangle \in P^I \\ \mathbf{f} & \text{iff } \langle t_1^I, t_2^I, \ldots, t_n^I \rangle \notin P^I \end{cases}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\neg\phi) = \begin{cases} \mathbf{t} & \text{iff } (I, \mathcal{M}, \mathcal{N})(\phi) = \mathbf{f} \\ \mathbf{u} & \text{iff } (I, \mathcal{M}, \mathcal{N})(\phi) = \mathbf{u} \\ \mathbf{f} & \text{iff } (I, \mathcal{M}, \mathcal{N})(\phi) = \mathbf{t} \end{cases}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_1 \wedge \phi_2) = \min \{ \langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_1), \langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_2) \}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_1 \supset \phi_2) = \begin{cases} \mathbf{t} & \text{iff } \langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_2) \geq \langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_1) \\ \mathbf{f} & \text{otherwise} \end{cases}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\exists x : \phi) = \max \{ \langle I, \mathcal{M}, \mathcal{N} \rangle(\phi[\alpha/x]) \mid \alpha \in \Delta \}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\mathbf{K}\,\phi) = \begin{cases} \mathbf{t} & \text{iff } \langle J, \langle M, M_1 \rangle, \mathcal{N} \rangle(\phi) = \mathbf{t} \text{ for all } J \in M \\ \mathbf{f} & \text{iff } \langle J, \langle M, M_1 \rangle, \mathcal{N} \rangle(\phi) = \mathbf{f} \text{ for some } J \in M_1 \\ \mathbf{u} & \text{otherwise} \end{cases}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\mathbf{not}\,\phi) = \begin{cases} \mathbf{t} & \text{iff } \langle J, \mathcal{M}, \langle N, N_1 \rangle \rangle(\phi) = \mathbf{f} \text{ for some } J \in N_1 \\ \mathbf{f} & \text{iff } \langle J, \mathcal{M}, \langle N, N_1 \rangle \rangle(\phi) = \mathbf{t} \text{ for all } J \in N \\ \mathbf{u} & \text{otherwise} \end{cases}$

An *MKNF interpretation* is a non-empty set of first-order interpretations. An *MKNF interpretation pair* $\langle M, N \rangle$ consists of two MKNF interpretations M, N with $\emptyset \subsetneq N \subseteq M$. An MKNF interpretation pair *satisfies* an MKNF sentence ϕ, written $\langle M, N \rangle \models \phi$, if and only if $\langle I, \langle M, N \rangle, \langle M, N \rangle \rangle(\phi) = \mathbf{t}$ for every $I \in M$. If there exists an MKNF interpretation pair satisfying ϕ, then ϕ is *consistent*. An MKNF interpretation pair $\langle M, N \rangle$ is a *three-valued MKNF model* for a given MKNF sentence ϕ if

1. $\langle M, N\rangle$ satisfies ϕ and
2. for each MKNF interpretation pair $\langle M', N'\rangle$ with $M \subseteq M'$ and $N \subseteq N'$, where at least one of the inclusions is proper and $M' = N'$ if $M = N$, there is some $I' \in M'$ such that $\langle I', \langle M', N'\rangle, \langle M, N\rangle\rangle(\phi) \neq \mathbf{t}$.

For any MKNF interpretation pairs $\langle M_1, N_1\rangle$ and $\langle M_2, N_2\rangle$ we define the knowledge ordering as: $\langle M_1, N_1\rangle \succeq_k \langle M_2, N_2\rangle$ iff $M_1 \subseteq M_2$ and $N_1 \supseteq N_2$. Such an order is of particular interest for comparing models. In logic programming the least model w.r.t. derivable knowledge among all three-valued models for a given program is the well-founded model. Here, we introduce a similar notion referring to the minimal three-valued MKNF models, i.e. the ones among all three-valued MKNF models that leave as much as possible undefined.

Definition 2 (Well-Founded MKNF Model). *Let ϕ be an MKNF sentence and $\langle M, N\rangle$ a three-valued MKNF model of ϕ such that $\langle M_1, N_1\rangle \succeq_k \langle M, N\rangle$ for all three-valued MKNF models $\langle M_1, N_1\rangle$ of ϕ. Then $\langle M, N\rangle$ is a* well-founded MKNF model *of ϕ.*

Given a hybrid knowledge base $\mathcal{K}$, the well-founded model of $\mathcal{K}$ *is the well-founded model of $\pi(\mathcal{K})$.*

The following Theorem pinpoints the fact that every consistent DL-safe hybrid MKNF knowledge base has a unique well-founded model.

Theorem 1 (Theorem 1 in [15]). *If $\mathcal{K}$ is a consistent DL-safe hybrid MKNF knowledge base, then a well-founded MKNF model of $\mathcal{K}$ exists, and it is unique.*

As shown in [15], the well-founded MKNF semantics also generalises the well-founded semantics for logic programs [9] – for every logic program $\mathcal{P}$, the well-founded model of $\mathcal{P}$ directly corresponds to well-founded MKNF model of $\pi(\mathcal{P})$.

Finally, the well-founded model of a judicial knowledge base is determined by the hybrid knowledge base to which it directly corresponds.

Definition 3 (Well-Founded Model of a Judicial Knowledge Base). *Let $\mathcal{K} = \langle \mathcal{F}, \mathcal{N}\rangle$ be a judicial knowledge base, $\mathcal{T}$ a TBox and $\mathcal{P}$ a logic program such that $\mathcal{N} = \mathcal{T} \cup \mathcal{P}$. The* hybrid knowledge base associated to $\mathcal{K}$ *is $\mathcal{K}' = \langle \mathcal{F} \cup \mathcal{T}, \mathcal{P}\rangle$.*

The well-founded model of $\mathcal{K}$ *is the well-founded model of $\mathcal{K}'$.*

4 Implementation

As defined in Section 3, the MKNF Semantics is parametric on a decidable description logic in which the ontology is written. As shown in [14], the choice of this description logic determines the usability of the system, as the complexity of reasoning in the well-founded MKNF semantics is in the same class as the decidable description logic; a complexity result that is extended to a query-driven approach in [1].

CDF-Rules[3] [11] is an implementation of the well-founded MKNF semantics, on XSB system, that fixes the description logic component to CDF [**?**] ontologies which, in its Type-1 version, fully supports the $\mathcal{ALCQ}$ description logic. Since reasoning in $\mathcal{ALCQ}$ is performed in `EXPtime`, *CDF-Rules* also achieves `EXPtime` complexity.

One particular advantage of *CDF-Rules* is that it is a query-driven system. In fact, the definition of the well-founded MKNF semantics presented in Section 3 constructs the complete well-founded model for a given hybrid knowledge base, based on a bottom-up computation. However, in practical cases, this is not what is intended. Particularly, in the context of judicial systems, what one usually wants is to know whether or not a particular law is applicable to a particular individual, or what is the applicable penalty. Deriving all the consequences of the knowledge base to answer a query about an individual, or about a particular crime, would be in most cases impractical.

CDF-Rules provides a practical framework for query evaluation in well-founded MKNF knowledge bases, combining rules and ontologies, which is sound and complete w.r.t. the well-founded semantics. Moreover, since the example presented in Section 2.2 can be expressed using $\mathcal{ALCQ}$ theories, *CDF-Rules* is suitable for our intents.

4.1 *CDF-Rules*

To evaluate queries, *CDF-Rules* makes use of XSB's SLG Resolution [7], together with tableaux mechanisms supported by CDF theorem prover to check entailment in the ontology. Accordingly, to decide the truth value of a formula defined in the rules component, *CDF-Rules* employs SLG Resolution which, in a nutshell, is a tabling method of resolution that is able to answer queries under the well-founded semantics with polynomial-time complexity.

However, to decide the entailment of a formula ϕ w.r.t. an ontology $\mathcal{O}$ a tableau algorithm tries to construct a common *model* for $\neg\phi$ and $\mathcal{O}$, sometimes called a *completion graph* (cf. e.g. [22]). If such a model can not be constructed then $\mathcal{O}$ entails ϕ; otherwise $\mathcal{O}$ does not entail ϕ. Similar to other description logic provers, the CDF theorem prover attempts to traverse as little of an ontology as possible when proving ϕ. As a result, when the prover is invoked on an atom p, the prover attempts to build a model for the underlying individual(s) to which p refers, and explores additional individuals only as necessary.

Given the interdependence between the rules and the ontology in MKNF knowledge bases, the prover must consider the knowledge inferred by the rules in the program for the entailment proof, as a DL-atom can be derived by rules, which in turn may rely on other DL-atoms proven by the ontology. Thus, in order do compute a query Q, *CDF-Rules* constructs a fixed point that iteratively computes a (sub-)model for the objects that appear in Q, deriving at each iteration new information about their roles and classes, along with information about

[3] The implementation is available from the XSB CVS repository as part of the CDF package in the subdirectory `packages/altCDF/mknf`.

other individuals related to them, either in the ontology (via CDF's tableau algorithm) or in the rules (via SLG procedures). Since this (sub-)model is only constructed for objects that are *relevant* to the query, *CDF-Rules* significantly reduces the amount of computation required when compared to the Definition of well-founded MKNF model presented in Section 3.2.

As stated in Section 3, the definition of MKNF imposes the restriction of DL-safe knowledge bases to obtain decidability. The idea of this concept is basically to constrain the use of rules to individuals actually appearing in the knowledge base under consideration. To implement such concept, *CDF-Rules* has two special predicates *definedClass/2* and *definedRole/3* that define the *domain* of the rules, i.e. the set of individuals that are applicable to a given rule. These predicates can be defined explicitly by the compiler or the programmer, but they can also be inferred if all rules are DL-safe.

Next, we provide details on how to implement the example from Section 2.2 in *CDF-Rules*. For more information on the implementation of *CDF-Rules* the interested reader is referred to [11].

4.2 Implementing Judicial Knowledge Bases

Intuitively, to implement a judicial knowledge base (and particularly, the example presented in Section 2.2), one needs to define two different components in *CDF-Rules*: one component defining the logic-programming rules; and one component defining the intensional (TBox) and extensional (ABox) portions of the ontology.

Rules in *CDF-Rules* are stated in a Prolog-like style but with the inclusion of two additional special predicates *known/1* and *dlnot/1* that account for the modalities **K** and **not** of MKNF[4]. For instance, the rules for defining murder and aggravatedMurder presented in Section 2.2 can be represented as illustrated in Figure 3.

```
aggrMurder(X,Y) :- known(killingBy(X, Y)), known(cens(X)).
murder(X,Y)     :- known(killingBy(X, Y)),
                                          dlnot(aggrMurder(X,Y)).
```

Fig. 3. *CDF-Rules* implementation of murder and aggravatedMurder properties

On the other hand, the ontology component in *CDF-Rules* is defined under the standard CDF syntax. Instances of classes, roles and objects in CDF syntax are defined by using the constructs *cid/2*, *rid/2*, *oid/2*, respectively. These constructs are then used in reserved predicates that are interpreted as ontology definitions.

[4] As defined in [11], we can relax the assumption of DL-Safe rules if we assume that the predicates *definedClass/2* and *definedRole/3* are defined for each rule.

Furthermore, to express more complex statements to define classes as negation or cardinality, CDF-syntax supports a special kind of identifier: *a virtual identifier*, denoted by the functor *vid/1*. This identifier is then used within the special fact *necessCond/2* that has the goal to state necessary conditions.

As illustration, in order to state that the class Torture is included in the class Censurable (i.e., Censurable $\sqsupseteq$ Torture) we use the predicate *isa/2* that defines state inclusion (cf. Figure 4).

```
isa_ext(cid(torture,mknf),cid(cens,mknf)).
```

Fig. 4. CDF syntax for Censurable $\sqsupseteq$ Torture

Moreover, to state the expression Censurable $\sqsupseteq$ ($\geq$ 3 PerformedBy.Person) we need a little tweak. Particularly, since CDF syntax only allows one to state inclusion (via *isa/2*) between identifiers of classes or objects, to express such statement we need to transform the expression ($\geq$ 3 PerformedBy.Person) also into a class identifier. With this goal, we first need to define an auxiliary class Aux such that Aux $\doteq$ ($\geq$ 3 PerformedBy.Person). Since we are defining cardinality of a class, to express such condition we need the special predicate *necessCond/2*. Afterwards, all one needs to do is to state that the auxiliary class definition Aux is a subclass of Censurable. This is materialised in Figure 5.

```
necessCond_ext(cid(aux,mknf),
     vid(atLeast(3,rid(performedBy,mknf),cid(person,mknf)))).
isa_ext(cid(aux,mknf),cid(cens,mknf)).
```

Fig. 5. CDF syntax for Censurable $\sqsupseteq$ ($\geq$ 3 PerformedBy.Person)

Furthermore, to define simple minimal and maximal cardinality of objects in CDF syntax, one can use the predicates *minAttr/4* and *maxAttr/4*, respectively. For instance, the predicate *minAttr/4* can be used to encode the ABox a : ($\geq$ 4 PerformedBy.Person). This is illustrated in Figure 6.

As expected, such implementation is able to answer queries about the individuals *john* and *mary* and about the facts a, b and c. Particularly, one is able to ask questions to the system, in a Prolog-like style, and conclude that *john* should be sentenced to 12–25 years in jail, and that *mary* is not guilty of crime b.[5]

[5] The full encoding of the example presented in Figs. 1 and 2 in *CDF-Rules* is available in http://centria.di.fct.unl.pt/~jleite/climaXIIexample.zip

```
minAttr_ext(oid(f,mknf),rid(performedBy,mknf),
                                              cid(person,mknf),4).
```

Fig. 6. CDF syntax for $a : (\geq 4\ \mathsf{PerformedBy.Person})$

5 Conclusions and Future Work

In this paper, we considered human-inspired normative systems and, in particular, those to be employed to regulate interaction in multi-agent systems. We advocated the need for an integration of rules and ontologies (already recognized in other areas of knowledge representation and reasoning) by showing how both are required to express an excerpt of a real-world Penal Code. We presented a language to express established facts (as a description logic ABox) and laws (as a hybrid of logic programming rules and description logic TBox). We provided the language with a formal semantics by means of a mapping to Hybrid MKNF, and exploited an efficient implementation to draw correct conclusions in the motivating example.

In our working example, we have focused on the outcome of applying the relevant laws to a given set of facts established in court. Therefore, the trial discussion phase, where each of the parties (prosecution and defense) provides relevant evidence to establish the facts, is outside of the scope of our framework. A neat approach to deal with the discussion phase is the argumentation framework of Sartor and Prakken [20].

Several other important aspects of rule-based approaches to normative systems and, in particular, to legal reasoning are outside the scope of the present work. Many of them have been acknowledged in the field of artificial intelligence and law, where there is now much agreement about the structure and properties of rules representing norms. These aspects include, for example, updates of norms, jurisdiction, authority, temporal properties, conflicting rules, exclusionary rules, legal interpretation and contrary-to-duty obligations. Notably, the works of Sartor and Prakken [20,21] using an Argumentation framework, Governatori et al [12,4] using a Defeasible Logic framework, Makinson and Van der Torre et al [17,5] using Input-Output Logic, are examples of works that fruitfully tackled some of these problems.

Boella et al. [3], among others, formally support the distinction between *constitutive norms*, which define concepts, and *regulative norms*, which describe desirable behaviour. While this distinction is conceptually clear and most probably useful for designing artificial normative systems, in this paper we did not follow this approach, because we preferred to remain faithful to the context of our motivating example, the Portuguese Penal Code, where such a formal distinction is not present. However, splitting the normative part of the judicial knowledge base into constitutive and regulatory portions would present to technical problems, and would allow to apply MKNF hybrid knowledge base evolution to support updates of constitutive norms (as done in [3]), regulative norms, or both.

In [8] we can find an application of first-order logic to the problem of specifying and reasoning about information flow and privacy polices. The logic has an expressive language, which allows the support for some features relevant in the area of information flow and privacy polices, and has been used to formalise large fragments of real world privacy laws, such as the Health Insurance Portability and Accountability Act (HIPAA) and the Gramm-Leach-Bliley Act (GLBA). The main difference w.r.t. our proposal is that they use classical implication to model policy rules, whereas we use logic programming implication to model norms. Moreover, their approach does not have a feature which is fundamental in the area of normative systems which is the ability to represent defeasible knowledge, allowed in our proposal by means of the use of default negation in rules.

With respect to future work, one of our main goals is to investigate the important problem of dealing with the evolution of normative systems, in particular those with hybrid representations such as the one described in this paper. The problems associated with knowledge evolution have been extensively studied, over the years, in different research communities, namely in the context of Classical Logic, and in the context of Logic Programming. They proved to be extremely difficult to solve, and existing solutions, even within each community, are still subject of active debate as they do not seem adequate in all kinds of situations in which their application is desirable. Then, when one looks at the solutions developed by the two communities, one immediately realises that they are based on entirely different intuitions and principles which makes them apparently irreconcilable. Two recent proposals [25,26] towards the development of update operators for Hybrid knowledge Bases seem promising and can perhaps serve as starting points to tackle the problem of dealing with the evolution of normative systems represented as Hybrid Theories.

We also intend to investigate the integration of our reasoner as a normative engine in existing multi-agent platforms, as a runtime compliance verification tool. In particular, we envisage the hybrid TBox and rules part of the judicial knowledge base to be defined by the system designer, and the ABox part to be provided by a monitoring system; the reasoner's task would be to draw conclusions about, for instance, applicable sanctions. Another long-term target of our research is a-priori compliance verification, given a system specification and a normative system. A possible approach would be similar to those proposed in the business process domain by Governatori and Rotolo [13]: compute logical consequences of the system formal model and the norms to which the system will be subject.

Acknowledgments

The authors would like to thank Ana Leite for all her advice on the Portuguese Penal Code and Matthias Knorr for discussions on the well-founded semantics for Hybrid MKNF. Ana Sofia Gomes was partially supported by FCT Grant SFRH/BD/64038/2009. Ricardo Gonçalves was partially supported by FCT

Grant SFRH/BPD/47245/2008. João Leite was partially supported by FCT Project ASPEN PTDC/EIA-CCO/110921/2009. Martin Slota was partially supported by FCT Grant SFRH/BPD/47245/2008.

References

1. Alferes, J.J., Knorr, M., Swift, T.: Queries to hybrid MKNF knowledge bases through oracular tabling. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 1–16. Springer, Heidelberg (2009)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
3. Boella, G., Governatori, G., Rotolo, A., van der Torre, L.: lex minus dixit quam voluit, lex magis dixit quam voluit: A formal study on legal compliance and interpretation. In: Casanovas, P., Pagallo, U., Sartor, G., Ajani, G. (eds.) AICOL-II/JURIX 2009. LNCS, vol. 6237, pp. 162–183. Springer, Heidelberg (2010)
4. Boella, G., Governatori, G., Rotolo, A., van der Torre, L.: A logical understanding of legal interpretation. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) KR. AAAI Press, Menlo Park (2010)
5. Boella, G., Pigozzi, G., van der Torre, L.: Normative framework for normative system change. In: Sierra, C., Castelfranchi, C., Decker, K.S., Sichman, J.S. (eds.) AAMAS. IFAAMAS , vol. 1, pp. 169–176 (2009)
6. Boella, G., van der Torre, L., Verhagen, H.: Introduction to normative multiagent systems. Computational and Mathematical Organization Theory 12, 71–79 (2006), http://dx.doi.org/10.1007/s10588-006-9537-7, 10.1007, doi:10.1007/s10588-006-9537-7
7. Chen, W., Warren, D.S.: Tabled Evaluation with Delaying for General Logic Programs. J. ACM 43(1), 20–74 (1996)
8. DeYoung, H., Garg, D., Jia, L., Kaynar, D.K., Datta, A.: Experiences in the logical specification of the HIPAA and GLBA privacy laws. In: Al-Shaer, E., Frikken, K.B. (eds.) WPES, pp. 73–82. ACM, New York (2010)
9. Gelder, A.V., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of the ACM 38(3), 620–650 (1991)
10. Sartor, G., Pompeu Casanovas, M.A.B., Fernandez-Barrera, M. (eds.): Approaches to Legal Ontologies: Theories, Domains, Methodologies. Law, Governance and Technology series. Springer, Heidelberg (2011)
11. Gomes, A.S., Alferes, J.J., Swift, T.: Implementing query answering for hybrid MKNF knowledge bases. In: Carro, M., Peña, R. (eds.) PADL 2010. LNCS, vol. 5937, pp. 25–39. Springer, Heidelberg (2010)
12. Governatori, G., Rotolo, A.: Bio logical agents: Norms, beliefs, intentions in defeasible logic. Autonomous Agents and Multi-Agent Systems 17(1), 36–69 (2008)
13. Governatori, G., Rotolo, A.: Norm compliance in business process modeling. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 194–209. Springer, Heidelberg (2010)
14. Knorr, M., Alferes, J.J., Hitzler, P.: A coherent well-founded model for hybrid mknf knowledge bases. In: ECAI, pp. 99–103 (2008)
15. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artificial Intelligence (2011) (accepted for publication)

16. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI 1991), pp. 381–386 (1991)
17. Makinson, D., van der Torre, L.W.N.: What is input/output logic? input/output logic, constraints, permissions. In: Boella, G., van der Torre, L.W.N., Verhagen, H. (eds.) Normative Multi-agent Systems. Dagstuhl Seminar Proceedings, vol. 07122. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl (2007)
18. Motik, B., Rosati, R.: Reconciling description logics and rules. Journal of the ACM 57(5) (2010)
19. Motik, B., Sattler, U., Studer, R.: Query answering for owl-dl with rules. J. Web Sem. 3(1), 41–60 (2005)
20. Prakken, H., Sartor, G.: Argument-based extended logic programming with defeasible priorities. Journal of Applied Non-Classical Logics 7(1) (1997)
21. Prakken, H., Sartor, G.: The role of logic in computational models of legal argument: A critical survey. In: Kakas, A.C., Sadri, F. (eds.) Computational Logic: Logic Programming and Beyond. LNCS (LNAI), vol. 2408, pp. 342–381. Springer, Heidelberg (2002)
22. Schmidt-Strauss, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence 48, 1–26 (1990)
23. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T.: The british nationality act as a logic program. Commun. ACM 29, 370–386 (1986), http://doi.acm.org/10.1145/5689.5920
24. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: off-line design. Artif. Intell. 73, 231–252 (1995), http://dx.doi.org/10.1016/0004-37029400007-N
25. Slota, M., Leite, J.: Towards closed world reasoning in dynamic open worlds. TPLP 10(4-6), 547–563 (2010)
26. Slota, M., Leite, J., Swift, T.: Splitting and updating hybrid knowledge bases. TPLP (to appear, 2011)
27. Swift, T., Warren, D.S.: Cold Dead Fish: A System for Managing Ontologies (2003), http://xsb.sourceforge.net

Acting on Norm Constrained Plans

Nir Oren[1], Wamberto Vasconcelos[1], Felipe Meneguzzi[2], and Michael Luck[3]

[1] Department of Computing Science, University of Aberdeen
Aberdeen AB24 3UE, UK
n.oren@abdn.ac.uk and wvasconcelos@acm.org
[2] Robotics Institute, Carnegie Mellon University
Pittsburgh PA 15213, USA
meneguzz@cs.cmu.edu
[3] Department of Informatics , King's College London
London WC2R 2LS, UK
michael.luck@kcl.ac.uk

Abstract. The behaviour of deliberative agents is often guided by a plan library designed to achieve goals given certain environmental conditions. Plans in these plan libraries are designed to achieve individual goals, and cannot possibly account for all possible variations of restrictions in the societies within which these agents must operate. These restrictions, captured through *norms*, include obligations, prohibitions, and permissions. Unlike traditional planning restrictions, norms can often be contradictory and impossible to achieve simultaneously, necessitating some form of compromise. In this paper we describe a technique for taking norms into consideration when deciding how to execute a plan. Our norms are constraint based, allowing for fine-grained control over actions. Our technique allows for reasoning about the interactions between norms, and resolves conflict by selecting actions where the cost of violating one set of norms is outweighed by the reward obtained in complying with another.

Keywords: Norms, Plans, Constraints.

1 Introduction

Most agent architectures (e.g. BDI based approaches such as AgentSpeak(L) [11]) make use of *offline* planning, where a plan library is created before execution in the environment begins. An agent utilising an offline plan library selects a plan for execution based on the state of the environment. A problem when using pre-generated plan libraries involves how to select plans appropriate to the current situation. A pre-generated plan is often conditional, identifying the environmental *context* in which it is applicable. However, it is difficult for the plan designer to envisage all the situations in which a plan could be considered for execution at design time. In particular, the society in which an agent operates may impose a given set of *norms*, which restrict the acceptable behaviour of the agent in that society. Norms do not have to be fixed, and those which apply to the agent may vary over time, and even emerge from multi-agent interaction, so

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 347–363, 2011.

it is often infeasible for the designer to take account of them in pre-generated plans. For example, consider a plan to build a refugee camp following some disaster. Such a plan may take the terrain in which the refugee camp is to be located into account, but may not have considered some other logistical, social or operational restrictions, such as fuel availability. If the original plan assumed that fuel is freely available, then in the context of fuel limits, the original plan may be unusable. However, by limiting the amount of fuel use, for example by introducing a prohibition on driving large distances once the camp is built (assuming that long drives are necessary for setting up the camp), the plan can still be used.

An advantage in taking norms into account when selecting a plan for execution involves the possibility of norm violation. In some situations, an agent may ignore a norm in order to achieve a critical goal. By basing plan selection not only on some context and invocation condition, but also on the norms that would be complied with and violated during plan execution, more flexible (and robust) behaviour can be achieved.

Clearly, the ability to adapt an existing plan library to cater for norms, and thus function in a variety of different situations, greatly promotes plan reuse. Given this, the main contribution of this paper lies in specifying how an agent should execute a plan, while deciding which norms to adhere to, in such a way so as to maximise its utility.

[14] created a constraint and predicate based norm representation, and we extend this representation to actions found within plans. Constraints allow for fine-grained control of the value of variables, increasing the expressiveness of our notation, and the sophistication of the mechanisms to manipulate them. In order to act in the presence of norms, we adopt a utility based approach. Informally, given a plan represented as an AND/OR tree, with actions specified as constrained predicates, we recursively compute the effects of executing the next action from the plan, identifying what constraints would appear given that the agent decides to comply, or violate a set of norms. Such norm compliance or violation affects the ultimate utility of the plan. Executing an action with specific bindings can trigger rules creating, or deleting additional norms, further constraining future action. Therefore, different sets of constraints may lead to plans with different utilities, and we must consider all possible sets of constraints. Our goal is thus to identify the set of constraints on action that result in maximal utility. These constraints are then used to guide plan execution.

The plans we consider for our approach are similar to a Hierarchical Task Network (HTN) [5]. However, the exploration of these plans by our approach differs from HTN planning with preferences [13] in that active norms, unlike preferences, change dynamically during planning. The defeasible nature of norms further complicates the determination of an optimal plan.

The remainder of this paper is structured as follows. Section 2 describes our approach's underlying data structures. Sections 3 and 4 then detail how these components are combined to reason about plans in the context of normative restrictions. We evaluate our approach in Section 5, and place our approach

in the context of existing work in Section 6, before concluding and identifying future work in Section 7.

2 Plans and Norms

We begin this section by describing the basic building blocks of our system. We then explain how plans, built up of actions, are represented. Section 2.3 then provides details regarding the specification of norms, following which we describe *normative rules* that identify when norms begin, and when they cease, to affect the scope of an action. Finally, we describe *enactment states*, which denote the norms affecting execution at a single point in time. In the remainder of this paper, we denote first-order terms generically as τ; variables are represented as $X, Y, \ldots$ and constants as a, b etc.

2.1 Constraints, Substitution and Unification

Our system makes extensive use of constraints to limit agent action. A constraint γ is an atomic formula of the form $\tau \bowtie \tau'$, where $\bowtie \in \{=, \neq, >, \geq, <, \leq\}$ and τ, τ' are first order terms. We write Γ to denote a generic, possibly empty, set of constraints. We employ the predicate $satisfy(\Gamma)$ for a set of constraints Γ, which holds if and only if the constraints allow at least one solution, i.e. if they are satisfiable. Note that a set of constraints $\{\gamma_1, \ldots, \gamma_n\}$ can be represented as a conjunction of constraints $\bigwedge_{i=1}^{n} \gamma_i$, and we typically employ this latter representation within the paper.

We also make use of unification and substitution relationships, usually applied to a first order formula and/or a constraint. The application of a substitution σ, which consists of a set of pairs X/τ to some structure β is written $\beta \cdot \sigma$, and consists of replacing all instances of X in β by τ. Finally, two structures β, β' unify according to substitution σ (abbreviated $unify(\beta, \beta', \sigma)$) if and only if $\beta \cdot \sigma = \beta' \cdot \sigma$.

2.2 Actions and Plans

To affect its environment, an agent executes actions, which we represent as ground atomic first order formulae. Plans identify groups of actions that must be taken, and are applicable in different situations. Plans are thus represented using partially ground actions. Applying a plan to a specific situation occurs via the grounding of variables (and therefore, only fully ground actions can be executed). We call partially ground actions *abstract actions*.

Definition 1. ***(Abstract Action, Actions and Entailment)*** *We define an abstract action α as $\psi \circ \Gamma$, where ψ is a first order atomic formula, and Γ is a set of constraints over a subset of the variables in ψ. Act is the set of all abstract actions.*

Given an abstract action $\alpha = \psi \circ \Gamma$ and a substitution σ, we say that ψ' is an action iff $\psi' = \psi \cdot \sigma$ such that ψ' contains no free variables. An abstract action

$\alpha = \psi \circ \Gamma$ entails an abstract action $\beta = \psi' \circ \Gamma'$ if and only if for all σ such that $unify(\psi, \psi', \sigma)$, whenever $satisfy(\Gamma \cdot \sigma)$, $satisfy(\Gamma' \cdot \sigma)$. If α entails β, we write $\alpha \supset \beta$.

Where obvious, we abbreviate abstract actions such as $a(X, Y) \circ X = \tau_1 \wedge Y = \tau_2$, with τ_1, τ_2 terms, as $a(\tau_1, \tau_2)$. Similarly, we write $\psi \circ \emptyset$ simply as ψ.

Inspired by work on HTN planning [5], we represent plans as AND/OR trees, with nodes in the tree generically specifying the actions that must be taken in order to execute the plan. Leaf nodes are associated with *primitive actions* (that is, those actions that an agent executes in order to affect the environment), while other nodes represent *compound actions*, made up of all of the node's children in the case of the node being an *AND* node, or of any one of the node's children in the case of an *OR* node.

Definition 2. ***(Plan)*** *A Plan P is one of*

1. *α, where α is an abstract action.*
2. **andN**$(P_1, \ldots, P_n)$ *where $P_1 \ldots P_n$ are plans.*
3. **orN**$(P_1, \ldots, P_n)$ *where $P_1 \ldots P_n$ are plans.*

The α node represents a primitive action within the plan. A node of the form **andN**$(P_1, \ldots, P_n)$ represents an *AND* node in the plan. This node is a compound action, requires all of the actions specified within $P_1, \ldots, P_n$ to be executed. Plan nodes of the form **orN**$(P_1, \ldots, P_n)$ represent *OR* nodes in the plan tree; for this compound action to be executed, one of $P_1, \ldots, P_n$ must have been executed[1].

As an example of such a plan, consider the requirement to establish a refugee camp at position *(X,Y)*. In order to do so, intelligence must first be collected (via a *intel(X,Y)* action). The area must then be cleared, either by the agent executing the plan (using a *selfClear(X,Y)* action, or through some other organisation (via the *outsourceClear(X,Y)* action). Finally, the camp itself must be built by executing the *b_camp* and *b_roads* primitive actions. This plan contains both *AND*, and *OR* nodes, and any constraints on the actions themselves (e.g. stating that $X < 5$) act as hard constraints on variable values. Such a plan (with no hard constraints) can be written as follows

andN(*intel*(*X*, *Y*),
orN(*selfClear*(*X*, *Y*),*outsourceClear*(*X*, *Y*)),
b_camp(*X*, *Y*), *b_roads*(*X*, *Y*))

Note that our representation considers plans independently of their context with regards to the environment; we do not associate any preconditions with plans or actions, and do not identify how the effects of a plan alter the environment. This independence of context also means that we ignore any factors that might make one OR node selected within a plan as compared to another.

[1] Compound actions can be associated with an abstract action, but this yields no additional representative power.

2.3 Norms

Within our system, norms are obligations, prohibitions and permissions on the possible values of specific variables, in the context of specific actions. An obligation can thus, for example, specify exactly where the refugee camp must be placed in the previous example, by restricting X and Y to specific values for the $build(X, Y)$ action. In order to create this restriction, norms make use of constraints.

Definition 3. ***(Norms and Constraints)*** *A norm is an obligation, permission or prohibition, written respectively as* $\mathsf{O}\alpha$, $\mathsf{P}\alpha$ *and* $\mathsf{F}\alpha$, *where* $\alpha = \psi \circ \Gamma$ *is an abstract action. A norm is interpreted as obliging, permitting, or prohibiting the execution of* ψ *according to constraints* Γ.

A norm $\omega_1 = \mathsf{X}\psi \circ \Gamma$ *entails another norm* $\omega_2 = \mathsf{X}\psi' \circ \Gamma'$, *where* X *is some modality, if and only if for all* σ *such that* $unify(\psi, \psi', \sigma)$, *whenever* $satisfy(\Gamma \cdot \sigma)$, *then* $satisfy(\Gamma' \cdot \sigma)$. *If* ω_1 *entails* ω_2, *we write* $\omega_1 \supset \omega_2$

A generic norm is represented by the symbol ω.

Norms are intended to constrain the values assigned to some variables within an abstract action. Critically, and unlike most work on norms (e.g. [1],[6]), an obligation $\mathsf{O}\psi \circ \Gamma$ thus does not specify that ψ *should be* executed, but instead states that *if* action ψ is executed, it should be done in a way that is consistent with constraints Γ.

Definition 4. ***(Auxilary Definitions for Norms)***

If $unify(\psi, \phi, \sigma)$, $\mathsf{X} \in \{\mathsf{O}, \mathsf{P}\}$, *and* $satisfy(\Gamma' \cdot \sigma)$, *then we say that* α complies with ω. *Alternatively, if* $\mathsf{X} = \mathsf{F}$, *then the norm is complied with if, for all* σ *such that* $unify(\psi, \phi, \sigma)$, $\neg satisfy(\Gamma' \cdot \sigma)$.

A norm $\omega = \mathsf{X}\psi \circ \Gamma'$ *is said to be applicable for an abstract action* $\alpha = \phi \circ \Gamma$, *written* $applicableNorm(\omega, \alpha)$ *if and only if* $unify(\phi, \psi, \sigma)$ *for some* σ. *Given a set of norms* Ω *and abstract action* α, *we define the function* $applicableNorms(\Omega, \alpha) = \{\omega | \omega \in \Omega$ *and* $applicableNorm(\omega, \alpha)\}$

Consider a norm $\omega_c = \mathsf{O}selfClear(X, Y) \circ X \leq 8 \wedge Y = 2$. ω_c is applicable for an abstract action $selfClear(A, B) \circ \Gamma$ for any Γ. Similarly, $selfClear(5, 2)$ complies with ω_c.

2.4 Permissions and Conflicts

We treat permissions as exceptions to obligations and prohibitions, so they do not have meaning in isolation. Thus, for example, the norm $\mathsf{O}selfClear(X, Y) \circ \{X < 30, Y = 20\}$ imposes an obligation, when executing the *selfClear* action, that X is bound to a value less than 30, and Y is equal to 20. The permission $\mathsf{P}selfClear(X, Y) \circ \{X < 40\}$ allows X to be less than 40 *if* the obligation is present, while still complying with the obligation.

Violations apply in specific cases where an obligation or prohibition is not complied with, and no permission exists that permits this non-compliance to occur (we refer to such a permission as a *mitigating permission*). Clearly, given this model of norms, violation must be considered with regards to a set of permissions.

Definition 5. ***(Mitigating Permissions and Violations)*** *Given a norm* $\omega = \mathsf{X}\psi \circ \Gamma$ *where* $\mathsf{X} \in \{\mathsf{O}, \mathsf{F}\}$*, and a permission* $\omega' = \mathsf{P}\psi \circ \Gamma'$*, we refer to* ω' *as a* mitigating permission*.*

An abstract action β violates *an obligation or prohibition* ω *if* ω *is applicable for* β*,* β *does not comply with* ω *and there is no mitigating permission* $\mathsf{P}\phi \circ \Gamma'$ *such that satisfy*$(\Gamma' \cdot \sigma)$*.*

Finally, multiple obligations or prohibitions may conflict, requiring contradictory behaviour. Informally, a set of norms over an action is in conflict if no substitution of variables can be made that is consistent with the obligations and prohibitions found within the norm set, and no mitigating permissions exist allowing this substitution.

Definition 6. ***(Normative Conflict)*** *Given a set of norms* Ω*, let* $\Omega^O, \Omega^P, \Omega^F$ *represent the subsets of obligations, permissions and prohibitions within* Ω*. Similarly,* $\Gamma^O/\Gamma^P/\Gamma^F$ *represents the set of constraints found in* $\Omega^O/\Omega^P/\Omega^F$*. Then, given a set of norms* Ω *of the form* $\omega_i = \mathsf{X}\psi \circ \Gamma_i$ *(i.e. referring to the same action* ψ*), the set* Ω *is* in conflict*, if and only if there is no substitution* σ *such that the following holds*

$$\bigwedge_{\gamma^O \in \Gamma^O} \gamma^O \cdot \sigma \bigwedge_{\gamma^F \in \Gamma^F} \neg\gamma^F \cdot \sigma \bigvee_{\gamma^P \in \Gamma^P} \gamma^P \cdot \sigma \tag{1}$$

When conflicting norm sets occur, a reasoner must choose which subset of norms to comply with, and which to ignore.

2.5 Normative Rules

The norms imposed on agents are situation dependent, and we make use of a simple rule language to specify *normative rules* that identify the cases in which a norm starts, and ceases, to exist. Normative rules are written in the form $LHS \Rightarrow RHS$, where LHS contains conjunctions of actions and norms, and RHS identifies which obligations, permissions and prohibitions should be added to, or removed from, the set currently affecting the agent. Informally, if an action in the LHS of such a rule has been executed, then the set of norms must be modified according to the RHS of the rule. Similarly, if a norm ω exists in a rule's LHS, then the set of norms is modified as per the rule's RHS. The LHS is formed of a maximum of one abstract action, together with a conjunctive combination of zero or more norms. The RHS of the rule then identifies the norm to be added or removed. The BNF for normative rules is shown in Figure 1.

$$
\begin{aligned}
R &::= LHS \Rightarrow RHS \\
LHS &::= \alpha | NLHS \\
NLHS &::= \omega | NLHS \wedge NLHS \\
RHS &::= RHS \wedge RHS | \oplus \omega | \ominus \omega
\end{aligned}
$$

Fig. 1. BNF for normative rules

$\oplus\omega$ denotes the addition of ω to the set of currently active norms, while $\ominus\omega$ denotes the removal of ω from this set. *Rules* represents the set of all normative rules in the system.

Thus, the rule $intel(20, 5) \Rightarrow \oplus\omega_c$ states that if action $intel(20, 5)$ is executed, norm ω_c will come into force. Normative rules containing a norm in the *LHS* can represent norms that come into force due to the presence of other norms, allowing contrary-to-duty obligations to be modelled. For example, an obligation to build a camp, represented by ω_{bc}, might create an obligation to build a road, represented as ω_{br}. The normative rule $\omega_{bc} \Rightarrow \oplus\omega_{br}$ captures this situation.

Before discussing the semantics of norms, we describe the structure used to track the normative status of an executing system. This structure, referred to as an *enactment state*, identifies the norms that exist at any point in time due to the application of normative rules in a previous time point.

2.6 Enactment States

By executing actions, an agent changes the state of the world around it. Under the influence of normative rules, such changes cause new norms to be instantiated, or existing ones to be lifted, affecting future actions. Similarly, past actions can limit future action, by binding values to some of the future action's variables. Following [4], we represent the environment affecting the agent as a transition system between individual *enactment states*, each of which represent the system at a single time point. By executing an action, the system is transitioned to a new enactment state. Such a transition system, starting at an initial state, and transitioning to new enactment states until all agent actions have been executed, represents an entire run of the system.

To capture the portion of the environment affecting the agent, enactment states must track the obligations, permissions and prohibitions that are in force, and the constraints on variable values that have already been committed to.

Definition 7. ***(Enactment State)*** *$\Delta = (\Omega_\Delta, \Gamma_\Delta)$ is an enactment state, with Ω_Δ,a set of norms, and a constraint Γ_Δ.*

We have now described the basic data structures used by an agent in determining how to act in the presence of norms and normative rules. Next, we describe the rules that govern transitions between enactment states in more detail. These rules are then extended to provide an algorithm for acting in the presence of normative rules.

3 Transitioning between Enactment States

The previous section described the data structures used in our framework, and we now assign an operational semantics to these structures, using them to describe legal transitions between enactment states. Our approach modifies that taken by [4] in two ways. First, we allow only norms and actions in the *LHS* of a rule, simplifying our semantics. Second, [4] was concerned with identifying a new enactment state following the execution of an action. We are concerned with determining the *possible* enactment states following the execution of some action specified by an abstract action. Thus, multiple enactment states are possible. For example, consider the rules

$$intel(X, Y) \Rightarrow \oplus\omega_1 \quad intel(5, 6) \Rightarrow \oplus\omega_2 \quad intel(7, 8) \Rightarrow \oplus\omega_3$$

Executing action $intel(2, 2)$ results in norm ω_1 added to the resulting enactment state. Executing $intel(5, 6)$ leads to both ω_1 and ω_2 appearing in the new enactment state. If it is known that the *intel* action is executed, but its parameters are unknown (e.g. due to the action appearing in a partially ground plan), three possible enactment states can be transitioned to, namely one in which ω_1 exists, one where ω_1, ω_2 exist, and one in which ω_1, ω_3 exists. Since our approach considers the possible enactment states resulting from the execution of the abstract actions, our semantics, unlike those of [4], must identify a set of possible enactment states rather than a single enactment state.

Rules are applied when they are consistent with the action being executed, and the norms found in the system. Since an abstract action can encapsulate a large range of actions, we must identify when a rule is *potentially applicable*. This situation occurs when the abstract action found in the rule entails the abstract action being entailed, and all norms found in the rule are entailed by norms found within the current enactment state. Formally,

Definition 8. ***(Potentially Applicable Rule)*** *A rule $R \equiv \beta \wedge \omega_1, \ldots, \omega_n \Rightarrow RHS$ is potentially applicable with respect to a set of norms Ω and an abstract action α if and only if $\alpha \supset \beta \wedge \forall\omega_i \exists\omega \in \Omega$ such that $\omega \supset \omega_i$. The predicate $potApp(R, \Omega, \alpha)$ holds if R is potentially applicable with respect to Ω and α.*

Given an enactment state, an abstract action, and a set of normative rules, Algorithm 1 returns the possible enactment states that can result from executing all possible actions represented by the abstract action. Within this algorithm, the *actionConstraints* function returns the constraint associated with the abstract action found within the rule (or true if no abstract action exists).

The algorithm operates by first identifying all combinations of potentially applicable rules, and then evaluating each of these combinations individually (Line 5). Line 7 computes the constraints imposed due to the abstract action and the norms under consideration, together with those constraints imposed to ensure that the remaining norms are not applied. If these constraints can be satisfied, the set of rules under consideration will result in a new enactment state, and Lines 9–14 create this new enactment state, based on the old one, by adding the appropriate norms, and the constraints imposed by the applied rules.

Algorithm 1. Computing all possible enactment states.

1: **function** PosEnactStates($\Delta, \alpha, Rules$)
2: $\Delta = (\Omega_\Delta, \Gamma_\Delta)$
3: $\alpha = \psi \circ \Gamma_\alpha$
4: $\Delta_N = \{\}$
5: $P = 2^R$ s.t. $R = \{r | r \in Rules$ and $potApp(r, \Omega_\Delta, \alpha)\}$
6: **for all** $p \in P$ **do**
7: $\Gamma = \Gamma_\Delta \wedge \Gamma_\alpha \bigwedge_{r \in p} actionConstraints(r) \bigwedge_{s \in R \setminus p} \neg actionConstraints(s)$
8: **if** satisfy(Γ) **then**
9: $\Omega = \Omega_\Delta$
10: **for all** $RHS \Rightarrow \oplus\omega \in p$ **do**
11: $\Omega = \Omega \cup \{\omega\}$
12: **for all** $RHS \Rightarrow \ominus\omega \in p$ **do**
13: $\Omega = \Omega \setminus \omega$
14: $\Delta_N = \Delta_N \cup \{(\Omega, \Gamma)\}$
15: **for all** $(\Omega, \Gamma) \in \Delta_N$ **do**
16: **if** $\exists(\Omega', \Gamma') \in \Delta_N$ s.t. $\Omega \subset \Omega'$ and $\Gamma' \rightarrow \Gamma$ **then**
17: $\Delta_N = \Delta_N \setminus (\Omega, \Gamma)$
18: **return** Δ_N

Finally, starting at Line 15, we remove all enactment states obtained due to the application of non-maximally consistent sets of potentially applicable rules.

Algorithm 1 provides us with a semantics for normative rules, identifying the normative effects of an action on the current enactment state[2]. In the next section, we investigate what abstract action should be executed given that the current enactment state contains some set of norms.

4 From Plans to Norm Constrained Actions

When executing an abstract action, we must reason about the constraints that should be imposed on it. These constraints are obtained from the norms found within the current enactment state. For example, when executing action $a(X)$, given the norm $\mathsf{O}a(X) \circ X < 5$, the agent could constrain the value of X to less than 5 if it decides to comply with the norm. Now, consider a sequence of abstract actions, such as $a(X), b(Y), c(X, Z)$. Constraints on the value of X, selected due to $a(X)$, could affect norm compliance when executing $c(X, Z)$. Thus, compliance with norms at one time point can affect later norm compliance choices.

We adopt a utility based model of norm compliance. More specifically, we assume that the execution of a plan results in some base utility, and that different types of norms are associated with different utility measures. Obligations and

[2] Note that our algorithms do not explicitly manipulate substitutions, as these are applied to variables during the computations.

Algorithm 2. Finding the optimal plan.

Require: A plan *Plan*, utility function *cost* and set of rules *Rules*

for all *Action* ∈ first possible actions of *Plan* **do**
  Let $\Upsilon = \{\langle(\{\}, \top), [Action], 0\rangle\}$
$Best = \emptyset$
while $\Upsilon! = \emptyset$ **do**
  $\langle(\Omega, \Gamma), Actions, U\rangle$=removed highest utility element of Υ
  Let α = The last element of *Actions*
  if $\alpha = \emptyset$ & $U >$ utility of *Best* **then**
    $Best = \langle(\Omega, \Gamma), Actions, U\rangle$
    break
  $AN = applicableNorms(\Omega, \alpha)$
  for all $\Omega_{AN} \in 2^{AN}$ **do**
    $\Gamma_N = \Gamma \wedge constraint(\alpha)$
    for all $\mathsf{X}\psi \cdot \Gamma_\psi \in \Omega_{AN}$ **do**
      if $\mathsf{X} = \mathsf{O}$ **then** $\Gamma_N = \Gamma_N \wedge \Gamma_\psi$
      if $\mathsf{X} = \mathsf{F}$ **then** $\Gamma_N = \Gamma_N \wedge \neg\Gamma_\psi$
      if $\mathsf{X} = \mathsf{P}$ **then** $\Gamma_N = \Gamma_N \vee \Gamma_\psi$
    if $satisfy(\Gamma_N)$ **then**
      $U_n = U + cost(\alpha, \Omega_{AN}, AN \backslash \Omega_{AN})$
      $\Delta_N = posEnactStates((\Omega, \Gamma_N), \alpha, Rules)$
      for all $\delta \in \Delta_N$ **do**
        for all β=next possible action of *Plan* **do**
          insert $\langle\delta, [Actions, \beta], U_n\rangle$ into Υ

prohibitions are associated with a a utility gain for compliance, and a utility loss for violation. Permissions are associated with a utility loss for utilisation (for example, obtaining permission to construct the refugee camp further away than is normally allowed might incur a loss of trust within the society, reflected by loss of utility). Actions also have a utility cost. Formally, we represent this via a utility function $cost : Act \times 2^{Norms} \times 2^{Norms} \rightarrow \mathbb{R}$. The *cost* function is a partial function, and its first parameter represents obligations and prohibitions that are complied with, while its second parameter is those obligations and prohibitions that are not complied with, together with the permissions that have been utilised. Thus, if a norm appears in the set passed in as one parameter, it may not appear within the other parameter. Under this model, the problem we are addressing reduces to selecting a path through the plan, and a set of norms (created by the rules as actions are executed) with which to comply, that is conflict free, and which lead to maximal utility.

Our approach undertakes a best-first incremental search in the enactment state space created by selectively expanding plans and selecting a subset of norms for compliance. We define a data structure $\langle\Delta, Actions, Utility\rangle$ to track the execution of a plan. Here, Δ is an enactment state, *Actions* is a sequence of abstract actions, and *Utility* is the current utility of the plan. Using this data structure Algorithm 2 describes the process of identifying an optimal plan.

The algorithm first creates a initial structure Υ containing a single element representing the plan with no actions having yet been executed. It also initialises the currently found solution (represented by *Best*) to the empty set. The heart of the algorithm starts at Line 4. We begin by selecting the current best action sequence (Line 5) and checking if it satisfies the plan in a manner better than the current best solution. If so, this action sequence replaces the current best solution. Otherwise, all applicable norms are identified. For each possible combination of applicable norms, the constraint of those norms that are applied are added to any existing constraints (Lines 11–16). Γ_N is analogous to (1) from Section 2.4, and if it is satisfiable, then this combination of norms is not in conflict, and can thus executed. The utility for complying with this subset of applicable norms is thus computed (Line 18), and all possible enactment states resulting from this action are then created (Line 19). Finally, all possible next abstract actions are obtained from the plan, and the updated action sequences, utilities, and enactment states are added back to Υ (Lines 20–22) allowing the process to continue.

We do not describe how to extract the next actions from an AND/OR tree, as standard algorithms exist to do so [5]. It should be noted that our algorithm can easily be extended to reason over multiple plans by associating each plan with its own base utility, and storing the plan in Υ. Also, note that a fully norm-compliant reasoner (that is, one that will only act if it can comply with all its norms) can be obtained from Algorithm 2 by modifying Line 11 to consider the set of applicable norms rather than its powerset.

While our algorithm is guaranteed to terminate, and is sound and complete if left to run to termination, its worst case complexity is clearly exponential. However, this complexity is mitigated by two factors. First, our algorithm is anytime, storing incrementally better solutions in *Best* (if they exist) as time progresses. Second, it is possible to use heuristics to improve the algorithm's performance. Before discussing such heuristics, we evaluate our algorithm in a simple domain.

5 Evaluation

We implemented our system in SWI-Prolog[3], using the *CLPFD* constraint library, and evaluated it on a simple bomb clearing scenario, as shown in Figure 2. The domain consists of a tile world with a single agent. Each tile could be empty, or contain a bomb that is either moderately (grey), or very (black), dangerous. The bomb clearing agent has only one plan available to it, abstractly represented as follows:

$$\mathbf{andN}(scanC, moveC, \mathbf{orN}(nothing, pickup, explodeC))$$

All except the *nothing* and *pickup* actions are compound actions, made up of a **orN** of primitive actions. For example, the *moveC* compound action identifies

[3] http://www.swi-prolog.org

the choice of moves available to the action at any point in time. It is defined as follows:

$$\begin{aligned} moveC \equiv \mathbf{orN}(&move(X,Y,A,B) \cdot A = X \wedge B = Y,\\ &move(X,Y,A,B) \cdot A = X+1 \wedge B = Y,\\ &move(X,Y,A,B) \cdot A = X-1 \wedge B = Y,\\ &move(X,Y,A,B) \cdot A = X \wedge B = Y+1,\\ &move(X,Y,A,B) \cdot A = X \wedge B = Y-1) \end{aligned}$$

This action thus allows the agent to move to a neighbouring tile, or stay in its current position (X and Y are replaced by the current position in our implementation). Similarly, the *scanC* action scans all four compass points around it to a range of 2, identifying the contents of the tile and its associated scan threat level. The *explodeC* action consists of an OR node composed of 8 primitive actions, allowing the agent to trigger an explosion up to 2 tiles away from it. The *pickup* action, used to clear a bomb from the square occupied by the agent, is defined as

$$pickup(C,D), C = A \wedge D = B$$

Note that this action takes two parameters identifying the agent's current position, and makes use of A and B, whose values are bound during the instantiation of the *moveC* action. Like the explode action, picking up a bomb removes it from the tile world.

We defined 10 normative rules for the system. Due to space constraints, we describe most of these only informally. The first normative rule is designed to prevent an agent from wandering out of the area in which bombs may exist. Additional normative rules are designed to prohibit an agent from moving onto a dangerous bomb, oblige the agent to explode such dangerous bombs, and also to oblige the agent to pick up low threat level bombs. Another rule prohibits explosions within 1 tile of the agent, and is defined as follows:

$$\begin{aligned} &move(R4XO, R4YO, R4X, R4Y) \cdot \top \Rightarrow\\ &\quad \oplus \mathsf{F} explode(R4A, R4B) \cdot (R4A = R4X \wedge R4B = R4Y) \vee\\ &\qquad\qquad (R4A = R4X - 1 \wedge R4B = R4Y) \vee\\ &\qquad\qquad (R4A = R4X + 1 \wedge R4B = R4Y) \vee\\ &\qquad\qquad (R4A = R4X \wedge R4B = R4Y - 1) \vee\\ &\qquad\qquad (R4A = R4X \wedge R4B = R4Y + 1))) \end{aligned}$$

A similar rule removing this obligation was also created. The order of rule evaluation (removal and then addition) allows these rules to operate correctly. Finally, 4 rules were defined to remove any obligations associated with bombs that have been removed from the environment. Finally, we associated a utility gain of 10 with exploding a bomb, and 5 with picking it up. We associated a utility cost of 100 with exploding a bomb too close to the agent, and a cost of 1 for moving into a square containing a dangerous bomb.

In this environment, an agent using the algorithms described in this paper will perform differently to a fully norm compliant agent. Consider the situation

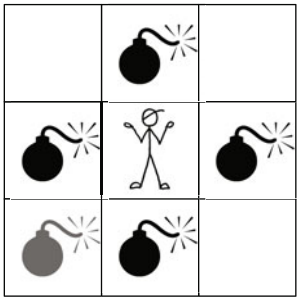


Fig. 2. A bomb clearing scenario in which norm aware and norm compliant agents will behave differently. Black bombs are very dangerous, grey bombs moderately dangerous.

illustrated in Figure 2, where the agent is surrounded by dangerous bombs. A fully norm compliant agent will not move from its starting position as doing so would violate one of the norms imposed by its normative rules. An agent capable of violating norms will move into one of the dangerous squares (violating a less important norm) and explode the bomb opposite it from 2 squares away (complying with a more important norm), thereby freeing it to continue moving around the environment. It should be noted that this is the only situation (outside similar cases when the agent is in a corner or edge of the world) where norm awareness allows an agent to select a different action to one that would be chosen by a norm compliant agent. Given this, we saw only a small improvement in the performance of the former agent over the latter when evaluated over randomly generated worlds.

Now norm-aware and norm-compliant agents should be contrasted with classic (norm unaware) BDI type agents. The latter type of agent, when operating in the sample domain, would execute some version of the plan at random, often moving into dangerous squares, randomly triggering explosions in tiles near them, and so on, and ultimately perform poorly. The difference between this type of agent, and the ones described previously does not lie in their plans, but rather in their norms. The ability to assign and modify norms in this way thus changes the behaviour of an agent *without requiring any modification to its plan library.*

The improved performance in bomb clearing comes at the cost of additional time; the norm aware and norm compliant agents both took approximately 13 seconds to select an action on a 2.4 GHz computer. This occurs as all possible executions of the plan, with regards to all possible combinations of constraints and norm violations, are evaluated by the system. In the next section, we discuss a number of possible techniques for improving the performance of our algorithms.

6 Discussion

Algorithms such as A* have shown that the addition of a heuristic to estimate the remaining utility gained by executing the rest of a plan can improve the

speed at which good solutions can be found. Making use of such a heuristic within our framework is simple, requiring only a change to Line 18 of Algorithm 2. However, identifying an appropriate heuristic is more difficult. [13] suggests several heuristics usable in HTN planning with preferences, and inspired by these, possible heuristics include assuming that no more norm violations will occur; that all norms will be complied with; or that some norms will be ignored. More complex heuristics include Monte-Carlo sampling of a plan.

Pruning low utility elements from Υ can also improve algorithm runtime. This is achieved by modifying Line 22 of Algorithm 2 to not run if the candidate addition's utility is much worse than the current best solution's. However, this speedup comes at the cost of completeness unless the cost function is monotonic.

The focus of this paper is on the role of norms within plans. While our work can be viewed as a form of HTN planning, the presence of norms provides a differentiator for our work. Norms provide guarantees to open large-scale distributed systems, establishing limits to the autonomy of components/agents [1]. There have been attempts at connecting the computational behaviours of individual components/agents and norms, whether to detect norm violation [4], or with a view to verify if a set of norms can ever be fulfilled by a society of autonomous software agents [14], or to inform agents about changes in their behaviour so as to make the behaviours norm-compliant [10]. However, our problem is distinct, and our approach novel: autonomous agents, with access to a library of plans to choose from, but subject to norms, can make use of our mechanism to choose a plan that will achieve individual and global goals while attempting to abide by these norms. Our approach was inspired by [14], which presents a mechanism to detect potential normative conflicts before they arise. However, that approach is overcautious, detecting conflicts that may never arise in actual system execution. In contrast, the work in this paper adopts a more accurate representation of agent behaviour, represented as a plan (with non-determinism in the choices of values for variables and choices for *OR* branches). Finally, Dignum et al. [2] propose the idea of an action having potential deontic effects. When reasoning about action execution, the norms resulting from the action are computed, and the norms resulting from those norms (e.g. contrary to duty obligations) are recursively identified. If normative conflict is detected, the action would not be executed. Dignum et al. focus on the deontic effects of a single action in the context of contrary to duty obligations, while we concentrate on the effects of norms on an entire plan.

Additionally, there is some similarity with work pursued by Governatori and others (e.g. [7,8]), in that both use an initial specification of possible behaviours, and check the norm-compliance of these behaviours. [7] presented an early form of Governatori's model, which concentrated on manually constructed plans, while [8] appears to be the most fully developed version of their approach. Both our approach and theirs contain conditional norms, which are represented as rules. However, there are also many differences. For example, while they utilise

business process descriptions and informally define a mix of predicate and first order logic for their underlying representation, we use a more abstract, and simpler (but fully formalised) plan description. Furthermore, [8] addresses a specific class of norms, namely, reparational obligations, in which violated norms can be repaired or compensated via other norms, but does not address, for instance, the violation of prohibitions, as we have done. Furthermore, the propositional nature of their work makes handling deadlines difficult, and their approach does not support norm removal.

[9] also considered norm compliance. However, this work was at a more abstract level, and while the interaction between an agent's goals and norms was discussed, no computational mechanism for deciding whether to comply with a norm was proposed. Like us, [3] attempts to maximise utility based on the consequence of complying with or violating a norm, with future world states represented as MDPs. However, while norms in our framework act as constraints on the values of parameters, Fagundes consideres norms as affecting the ability to perform an action in a given space. Thus, the space of actions to be considered, and the effects of norm compliance and violation, are very different.

Our work can be compared with the work of Sohrabi *et al.* [13] on the **HTNPlan-P** planner. This planner uses an extended version of PDDL that allows preferences on the decompositions and actions employed in HTN planning in a similar way to which we use norms to restrict action execution. Their extended preferences include temporally extended preferences regarding when a particular action (or goal) should or should not be executable, conditional on a subset of linear temporal logic (LTL). Preferences in LTL do not interact in the same way as permissions affect obligations and prohibitions, limiting the applicability of their techniques to our domain.

Finally, we believe our work can be applied to agent programming in a manner similar to the one followed by Sardina and Padgham [12]. Like them, our work expands a plan and finds applicable paths, and can thus be integrated into a BDI agent in order to allow it to select an action for execution.

7 Conclusions and Future Work

When utilising offline planning, a plan is selected for execution from a pregenerated plan library, with the selection being based on the goal to be met, and on the current state of the environment in which the plan is to be executed. However, such a plan cannot easily be adapted to operate under normative constraints which were not originally anticipated by the plan designer. By making use of a utility based approach, we have shown how a reasoner can act in an optimal manner, violating, and complying with norms as needed.

We can identify a number of avenues of future work. Our current focus involves investigating the heuristics discussed in Section 6. Additionally, as mentioned previously, our obligations, prohibitions and permissions can be viewed as a

specific type of conditional norm, imposing constraints on the manner in which an action should, should not, or may be executed, but only in the case that the action is executed. We intend to extend our framework to cope with more general norms, for example, obliging an action to be executed subject to some specific constraints. Such an extension would allow us to apply our work to areas outside the practical reasoning domain, such as electronic contracts, where the contracting parties can analyse the contract, and their plans, in order to determine whether they can achieve their own goals in a satisfactory manner while following the contract. We also intend to enrich our representation language in order to allow for constraints over finite sets and inference over rules. Finally, we intend to investigate how our approach can be adapted to domains containing uncertainty.

References

1. Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., van der Torre, L.: The BOID architecture. In: Proceedings of the Fifth International Conference on Autonomous Agents, pp. 9–16 (2001)
2. Dignum, F., Morley, D., Sonenberg, E.A., Cavedon, L.: Towards socially sophisticated BDI agents. In: Proceedings of the Fourth International Conference on Multi-Agent Systems, pp. 111–118 (2000)
3. Fagundes, M.S., Billhardt, H., Ossowski, S.: Reasoning about norm compliance with rational agents. In: Proceedings of the 2010 European Conference on Artificial Intelligence, pp. 1027–1028 (2010)
4. García-Camino, A., Rodríguez-Aguilar, J.A., Sierra, C., Vasconcelos, W.: Constraint rule-based programming of norms for electronic institutions. Journal of Autonomous Agents and Multiagent Systems 18(1), 186–217 (2009)
5. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann, San Francisco (2004)
6. Governatori, G., Hulstijn, J., Riveret, R., Rotolo, A.: Characterising deadlines in temporal modal defeasible logic. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 486–496. Springer, Heidelberg (2007)
7. Governatori, G., Milosevic, Z., Sadiq, S.: Compliance checking between business processes and business contracts. In: 10th International Enterprise Distributed Object Computing Conference, pp. 221–232 (2006)
8. Governatori, G., Rotolo, A.: How do agents comply with norms? In: Proceedings of the 2009 Workshop on Normative Multi-Agent Systems, vol. 09121 (2009)
9. López y López, F., Luck, M., d'Inverno, M.: A normative framework for agent-based systems. In: Boella, G., van der Torre, L.W.N., Verhagen, H. (eds.) Proceedings of the First International Symposium on Normative Multi-Agent Systems, vol. 07122 (2007)
10. Meneguzzi, F., Luck, M.: Norm-based behaviour modification in BDI agents. In: Proceedings of the 2009 International Conference on Autonomous Agents and Multiagent Systems, pp. 177–184 (2009)

11. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, pp. 42–55 (1996)
12. Sardiña, S., Padgham, L.: A bdi agent programming language with failure handling, declarative goals, and planning. Autonomous Agents and Multi-Agent Systems 23(1), 18–70 (2011)
13. Sohrabi, S., Baier, J.A., McIlraith, S.A.: HTN planning with preferences. In: Proceedings of the 2009 International Joint Conference on Artificial Intelligence, pp. 1790–1797 (2009)
14. Vasconcelos, W., Kollingbaum, M., Norman, T.J.: Normative conflict resolution in multi-agent systems. Journal of Autonomous Agents and Multi-Agent Systems 19(2), 124–152 (2009)

Justice Delayed Is Justice Denied: Logics for a Temporal Account of Reparations and Legal Compliance

Guido Governatori[1,⋆] and Antonino Rotolo[2]

[1] NICTA, Queensland Research Lab, Australia
guido.governatori@nicta.com.au
[2] CIRSFID, University of Bologna, Italy
antonino.rotolo@unibo.it

Abstract. In this paper we extend the logic of violation proposed by [14] with time, more precisely, we temporalise that logic. The resulting system allows us to capture many subtleties of the concept of legal compliance. In particular, the formal characterisation of compliance can handle different types of legal obligation and different temporal constraints over them. The logic is also able to represent, and reason about, chains of reparative obligations, since in many cases the fulfillment of these types of obligation still amount to legally acceptable situations.

1 Introduction

Developments in open MAS have pointed out that normative concepts can play a crucial role in modeling agents' interaction [24, 8]. Like in human societies, desirable properties of MASs can be ensured if the interaction of artificial agents adopts institutional models whose goal is to regiment agents' behaviour through normative systems in supporting coordination, cooperation and decision-making. However, to keep agents autonomous it is often suggested that norms should not simply work as hard constraints, but rather as soft constraints [4]. In this sense, norms should not limit in advance agents' behaviour, but would instead provide standards which can be violated, even though any violations should result in sanctions or other normative effects applying to non-compliant agents. The detection of violations and the design of agents' compliance can amount to a relatively affordable operation when we have to check whether agents are compliant with respect to simple normative systems. But things are tremendously harder when we deal with realistic, large and articulated systems of norms such as the law. To the best of our knowledge, no systematic investigation has been so far proposed in this regard in the MAS field.

Among other things, the complexities behind the concept of legal compliance are due to the following reasons:

Reparative Obligations. Legal norms often specify obligatory actions to be taken in case of their violation. Obligations in force after some other obligations have been violated correspond in general to contrary-to-duty obligations (CTDs) (see [7] for an

⋆ NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

J. Leite et al. (Eds.): CLIMA XII 2011, LNAI 6814, pp. 364–382, 2011.

overview). A peculiar subclass of CTDs is particularly relevant for the law: the so-called reparative obligations. For instance, in contract and in tort law reparative obligations protect individual legitimate interests by imposing actions that compensate any damages following from non-compliance [12]. These constructions affect the formal characterisation of legal compliance since they identify situations that are not ideal, but still legally acceptable. Consider the following example (where norms have as usual a conditional structure: if the antecedents are jointly the case, then the consequent is obligatory):

$$\begin{aligned} &\mathit{Invoice} \Rightarrow \mathrm{OBL}\mathit{PayBy7days} \\ &\quad \mathrm{OBL}\mathit{PayBy7days}, \neg \mathit{PayBy7days} \Rightarrow \mathrm{OBL}\mathit{Pay5\%Interest} \\ &\quad\quad \mathrm{OBL}\mathit{Pay5\%Interest}, \neg \mathit{Pay5\%Interest} \Rightarrow \mathrm{OBL}\mathit{Pay10\%Interest} \end{aligned}$$

What about if a customer violates both the obligation to pay by 7 days after having received the invoice for her purchase, and the obligation to pay the 5% of interest of the due amount, but she pays the total amount plus the 10% of interest? In the legal perspective (which aims at protecting the rights of the vendor), the customer is compliant.

If so, these constructions can give rise to very complex rule dependencies, because we can have that the violation of a single rule can activate other (reparative) rules, which, in case of their violation, refer to other rules, and so forth [15]. Clearly, if we take the above legal norms in isolation, the depicted situation is non-compliant, since two applicable legal norms are violated. However, if we compensate the violations, then we are still in a "legal" situation.

Obligation and Time. The law makes use of different types of obligations (see Section 2) also depending on how legal effects are temporally qualified. A first basic distinction is between those legal obligations which persist over time unless some other and subsequent events terminate them (e.g., "If one causes damage, one has to provide compensation"), and those that hold at a specific time on the condition that the norm preconditions hold and with a specific temporal relationship between such preconditions and the obligation (e.g., "If one is in a public building, one is forbidden to smoke").

In regard to the concept of compliance, it is worth noting that we may have obligations requiring (1) to be always fulfilled during a certain time interval, (2) that a certain condition must occur at least once before a certain deadline and such that the obligations may, or may not, persist after this deadline if they are not complied with, (3) that something is done immediately [13].

Things are definitely harder when these types of obligations occur in chains of reparative obligations. For example, if the primary obligation is persistent and states to pay before tomorrow, and the secondary (reparative) obligation is to pay a fine in three days after the violation of the primary obligation, we are compliant not only when we pay by tomorrow, but also when we do not meet this deadline and pay both the due amount and the fine on the day after tomorrow.

Formal Requirements for Legal Compliance. From a logical point of view, a formal characterisation of the concept of legal compliance requires to address the following related research tasks: (a) We need a logic able to handle different types of legal obligation and different temporal constraints over them; (b) This logic should be able to

represent, and reason about, chains of reparative obligations. In particular, we need a procedure for making hidden conditions and reparative chains explicit; without this, we do not know whether a certain situation is legally acceptable; (c) We have to embed into the logic aspects of time, such as persistence and deadlines.

In the following section we informally discuss the types of obligation we will handle in the proposed framework.

2 The Many Faces of Obligations

We can distinguish *achievement* from *maintenance obligations* [13]. For an *achievement obligation*, a certain condition must occur at least once before a deadline:

Example 1. Customers must pay within 7 days, after receiving the invoice.

The deadline refers to an obligation triggered by receipt of the invoice. After that the customer is obliged to pay. The fulfilment of the obligation by its deadline terminates the persistence of the obligation.

For *maintenance obligations*, a certain condition must obtain during all instants before the deadline:

Example 2. After opening a bank account, customers must keep a positive balance for 30 days.

In Example 2 the deadline only signals that the obligation is terminated: a violation occurs when the obliged state does not obtain at some time before the deadline.

Finally, *punctual obligations* only apply to single instants:

Example 3. When banks proceed with any wire transfer, they must transmit a message, via SWIFT, to the receiving bank requesting that the payment is made according to the instructions given.

Punctual obligations apply to single instants; they can be thought as maintenance obligations in force in time intervals where the endpoints are equal. Typically punctual obligations must occur at the same time of their triggering conditions.

Norms can be associated with an explicit sanction. For example,

Example 4. Customers must pay within 7 days, after receiving the invoice. Otherwise, 10% of interest must be paid within 10 days.

Example 5. After opening a bank account, customers must keep a positive balance for 30 days. Otherwise, their account must be immediately blocked.

A sanction is often implemented through a separate obligation, which is triggered by a detected violation. Thus, different types of obligations can be combined in chains of reparative obligations: in Example 4, the violation of the primary achievement obligation is supposed to be repaired by another achievement obligation; in Example 5, the violation of a primary maintenance obligation is compensated by a punctual obligation.

We introduced in [15, 14] the non-boolean connective $\otimes$: a formula like $a \otimes b$ means that a is obligatory, but if the obligation a is not fulfilled, then the obligation b is activated and becomes in force until it is satisfied or violated. However, the violation

condition of an obligation varies depending on the types of obligations used. In the remainder, we will extend the approach of [15, 14] by adding temporal qualifications to cover these cases.

3 Temporalised Violation Logic

To start with, we consider a logic whose language is defined as follows:

Definition 1 (Language). *Let $\mathscr{T} = (t_1, t_2, \dots)$ be a discrete linear order of instants of time, $Atm = \{a, b, \dots\}$ be a set of atomic propositions, and O be a deontic operator.*

- *A literal is either an atomic proposition or the negation of an atomic proposition, that is: $Lit = Atm \cup \{\neg l : l \in Atm\}$.*
- *If $l \in Lit$ and $t \in \mathscr{T}$, then l^t is a* temporal literal*; $\top$ and $\bot$ are* temporal literals*. $TLit$ denotes the set of temporal literals.*
- *If l^t is a temporal literal, then Ol^t and $\neg Ol^t$ are* deontic literals*. The set of deontic literals is denoted by $DLit$.*
- *If a^{t_a} and b^{t_b} are temporal literals, $t \in \mathscr{T}$, and $t_a \leq t$, then $a^{t_a} \otimes_t^x b^{t_b}$ (for $x \in \{p, m, a\}$) is an $\otimes$-chain.*
- *If α is an $\otimes$-chain, a^{t_a} is a temporal literal and $t \in \mathscr{T}$, then $\alpha \otimes_t^x a^{t_a}$ (for $x \in \{p, m, a\}$) is an $\otimes$-chain.*
- *Let α be either a temporal literal, or an $\otimes$-chain, $t \in \mathscr{T}$, then $\bot$, $\alpha \otimes \bot$ and $\alpha \otimes_t \bot$ are deontic expressions. Nothing else is a deontic expression. The set of deontic expressions is denoted by $DExp$.*

Let us explain the intuitive meaning of the various elements of the language. The meaning of a temporal literal a^t is that proposition a holds at time t. The deontic literal Oa^t means that we have the obligation that a holds at time t. The meaning of $\top$ and $\bot$ is that $\top$ is a proposition that is always complied with (or in other terms it is impossible to violate) and $\bot$, on the other hand, is a proposition that is always violated (or it is impossible to comply with). According to the intended meaning it is useless in the present context to temporalise them. $\otimes$ is a binary operator to express complex normative positions. More specifically, the meaning of a deontic expression like $\alpha \otimes_{t_\alpha}^x a^{t_a} \otimes_{t'_a}^y b^{t_b}$ is that the violation of a triggers a normative position whose content is b^{t_b}. What counts as a violation of a^{t_a} depends on the parameter x, encoding the type of obligation whose content is a, and the two temporal parameters t_a and t'_a. The nature of the normative position whose content is b^{t_b} depends on $\otimes^y$. The type of obligation whose content is a^{t_a} is determined by x. If $x = p$, then we have a punctual obligation (in this case we require that $t_a = t'_a$) and this means that to comply with this prescription have must hold at time t_a. If $x = a$, then we have an achievement obligation; in this case a is obligatory from t_a to t'_a, and the obligation is fulfilled if a holds for at least one instant of time in the interval $[t_a, t'_a]$. Finally, if $x = m$, similarly to the previous case, a is obligatory in the interval $[t_a, t'_a]$, but in this case, to comply with the prescription, a must hold for all the instants in the interval. As we have said, the $\otimes$ operator introduces normative positions in response to a violation of the formula on the left of the operator, thus this is a contrary-to-duty operator. An important application of contrary-to-duties is that a contrary-to-duty can be used to encode a sanction or compensation or reparation for

a violation. The focus of this paper is mostly on this type of contrary-to-duties. What about *DExp*? The meaning of a *DExp*, in particular of $\bot$ at the end of them, is that we have reached a situation that cannot be compensated for, This means that the penultimate element of a deontic expression identifies the 'last chance' to be compliant. After that the deontic expression results in a situation that cannot be complied with anymore.

Definition 2 (Rules/norms[1]). *A* rule $r : \Gamma \hookrightarrow \alpha$ *is an expression where r is a unique rule label,* $\Gamma \subseteq TLit \cup DLit$, $\hookrightarrow \in \{\Rightarrow^x, \rightsquigarrow\}$, $\alpha \in DExp$. *If* $\hookrightarrow$ *is* $\Rightarrow^x$, *the rule is a* defeasible rule*; If* $\hookrightarrow$ *is* $\rightsquigarrow$, *the rule is a* defeater. *For defeasible rules* $x \in \{a, m, p\}$, *and: If* $x = a$ *the rule is an* achievement rule*; If* $x = m$ *the rule is a* maintenance rule*; If* $x = p$ *the rule is a* punctual rule. *For defeaters* $\alpha \in TLit$.

A rule is a relationships between a set of premises and a conclusion, thus we use several types of rules to describe different types of relationships. We use the distinction of the types of the rules (defeasible and defeater) for the strength of the relationship between the premises and the conclusion. The superscript x indicates the mode of a rule. The mode of a rule tells us what kind of conclusion we can obtain from the rule. In the context the mode identifies the type of obligation we can derive. The idea is that from a rule of mode a, an achievement rule, we derive an achievement obligation.

A defeasible rule is a rule where when the body holds then typically the conclusion holds too unless there are other rules/norms overriding it. For example, when you receive an invoice, you have the obligation to pay for it:

$$r_1 : invoice^t \Rightarrow^a pay^t \tag{1}$$

The meaning of the above rule is that if you received an invoice at time t, then you have the obligation to pay for it, starting from time t.[2]

Defeaters are the weakest rules. They cannot be used to derive obligations, but they can be used to prevent the derivation of an obligation. Hence, they can be used to describe exceptions to obligations, and in this perspective they can be used to terminate existing obligations. For this reason, the arrow $\rightsquigarrow$ is not labeled by either a, m, nor p. Continuing the previous example, paying for the invoice terminates the obligation to pay for it:

$$r_2 : paid^t \rightsquigarrow pay^t \tag{2}$$

Rule r_2 says that if you pay at time t then, from time t on, there is no longer the obligation to pay. Notice that the defeater does not introduce the prohibition to pay again.

Definition 3 (Defeasible Theory). *A* Defeasible Theory *is a structure* $(F, R, \succ)$, *where F, the set of facts, is a set of temporal literals; R is a set of rules; and* $\succ$, *the superiority relation, is a binary relation over R.*

A theory corresponds to a normative system, i.e., a set of norms, where every norm is modelled by rules. The superiority relation is used for conflicting rules, i.e., rules

[1] In the reminder, we will interchangeably use both the terms 'norm' and 'rule', but we will prefer 'norm' whenever the usage of the term 'rule' may be confused with 'inference rule'.

[2] We assume the usual inter-definability between obligations and prohibition, thus $O\neg \equiv F$, and $F\neg \equiv O$.

whose conclusions are complementary literals, in case both rules fire. Notice that we do not impose any restriction on the superiority relation, which is a binary relation that just determines the relative strength of two rules. For example, if we consider the two rules in (1) and (2), given an invoice, and that the invoice has been paid the two rules alone cannot allow us to conclude anything due to the sceptical nature of Defeasible Logic. But if we further establish that $r_2 \succ r_1$, then the second rule prevails, and we will conclude that we are permitted not to pay.

Definition 4. *Given an $\otimes$-chain α, the* length *of α is the number of elements in it. Given an $\otimes$-chain $\alpha \otimes_t^x b^{t_b}$, the* index *of b^{t_b} is n iff the length of $\alpha \otimes_t^x b^{t_b}$ is n. We also say that b appears at index n in $\alpha \otimes_t^x b^{t_b}$.*

Definition 5 (Notation). *Given a rule $r : \Gamma \hookrightarrow \alpha$, we use $A(r) = \Gamma$ to indicate the antecedent or body of the rule, and $C(r) = \alpha$ for the consequent or conclusion or head of r. Given a set or rules R: $R_{\Rightarrow}$ is the set of defeasible rules in R; $R_{\rightsquigarrow}$ is the set of defeaters in R; R^a is the set of achievement rules in R; R^m is the set of maintenance rules in R; R^p is the set of punctual rules in R; $R[a^t]$ is the set of rules whose head contains a^t. $R[a^t, k]$ is the set of rules where a^t is at index k in the head of the rules.*

To simplify and uniform the notation we can combine the above notations, and we use subscripts and superscripts before the indication relative of the head. Thus, for example, $R_{\rightsquigarrow}[p^{10}]$ is the set of defeaters whose head is the temporal literal p^{10}, and the rule

$$r : a_1^{t_1}, \ldots, a_n^{t_n} \Rightarrow^p a^{10} \otimes_{10}^m b^{20}$$

is in $R_{\Rightarrow}^m[b^{20}]$, as well as in $R^p[a^{10}]$ and $R_{\Rightarrow}[b^{20}, 2]$.

Finally, notice that we will sometimes abuse the notation and omit (a) the timestamp t_l in the temporal literal l^{t_l} whenever it is irrelevant to refer to it in the specific context, (b) the mode x in the rule arrow $\Rightarrow^x$ when x can be instantiated with any of a, m or p, (c) x and y in $\otimes_y^x$ when x and y can be instantiated, respectively, with any of a, m, p and with any time instants.

Properties of the $\otimes$ operator When we have a deontic expression $\alpha = a_1 \otimes \cdots \otimes a_n$ we do not have information about the type of obligation for the first element. This information is provided when we use the expression in a rule. In this section we are going to investigate properties of $\otimes$, in particular when two (sub-)sequences of deontic expression are equivalent and thus we can replace them preserving the meaning of the whole expression (or rule). To simplify the notation, we introduce the following conventions.

Definition 6. *Let $r : \Gamma \Rightarrow^x \alpha$ be a rule, then ${}^x\alpha$ is an $\otimes$-sequence. The empty sequence is an $\otimes$-sequence. If $\alpha \otimes_{t_\alpha}^x a^{t_a} \otimes_{t'_a}^y \beta \otimes_{t_\beta}^z \gamma$ is an $\otimes$-sequence, where α, β, γ are $\otimes$-sequences, then ${}^x a^{t_a} \otimes_{t'_a}^y \beta$ is an $\otimes$-sequence.*

Given a rule $r : \Gamma \hookrightarrow^x \alpha \otimes_t^y \beta$, α can be the empty $\otimes$-sequence, and if so, then the rule reduces to $r : \Gamma \Rightarrow^y \beta$.

From now on, we will refer to $\otimes$-sequences simply as sequences and we will provide properties for sequences to be used in rules.

The first property we want to list is the commutativity of the $\otimes$ operator.

$$\alpha \otimes_t^x (\beta \otimes_{t'}^y \gamma) \equiv (\alpha \otimes_t^x \beta) \otimes_{t'}^y \gamma \tag{3}$$

We extend the language with $\top$ and $\bot$. Given their meaning, those two propositions can be defined in terms of the following sequence and equivalence[3]

$${}^p a^0 \otimes_0^p \neg a^0 \equiv \top \qquad \bot \equiv \neg\top. \tag{4}$$

The two new propositions are useful to define reduction rules for deontic expressions. Let us start with equivalences for $\top$.

$$\top \otimes \alpha \equiv \top. \tag{5}$$

This equivalence says that a violation of $\top$ can be compensated by α; however, $\top$ is a proposition that cannot be violated. Thus, the whole expression cannot be violated. What about when $\top$ appears as the last element of $\otimes$?

$$\alpha \otimes \top \equiv \top. \tag{6}$$

The meaning of $\alpha \otimes \top$ is that $\top$ is the compensation of α, thus the violation of α is sanctioned by $\top$. This means that the violation of α is always compensated for, thus we have a norm whose violation does not result in any effective sanction, thus violating α does not produce any effect. Hence, we have two possibilities: to reject (6) if we are interested to keep trace of violations, or to accept it if we want to investigate the effects of violations. In this paper we take the first option and we reject the equivalence of $\alpha \otimes \top$ and $\top$. Notice that reducing $\alpha \otimes \top$ to α would change the meaning, since this would mean that the violation of α cannot be repaired. To see this we move to the properties involving $\bot$.

$${}^p a^{t_a} \otimes_{t_a}^x \bot \equiv a^{t_a} \tag{7}$$

The above equivalence specifies that if $\bot$ is the compensation of a punctual obligation a at time t, then there is no compensation, since the compensation cannot be complied with. The effect of the rules is that we can eliminate $\bot$ from the deontic expression and we maintain the same meaning. Notice, however, that the same is not true for other types of obligations. For example, for $x \in \{a, m\}$, we cannot eliminate $\bot$ from rules like

$$\Gamma \Rightarrow^x a^t \otimes_{t'}^m \bot$$

since the resulting expression would be $\Gamma \Rightarrow^x a^t$ and we would miss the information about the deadline to comply with a. Nevertheless, the following equivalence states that $\bot$ can be safely eliminated if it is not the last element of a deontic expression, or when it is the 'compensation' of a maintenance obligation without deadline.

$$\alpha \otimes_{t_\alpha}^x \bot \otimes_t^y \beta \equiv \alpha \otimes_{t_\alpha}^y \beta \qquad {}^m a^{t_a} \otimes \bot \equiv {}^m a^{t_a} \tag{8}$$

[3] In case one wants the temporalised version, $\top^t \equiv {}^p a^t \otimes_t^p \neg a^t$, and $\bot^t \equiv \neg\top^t$.

To complete the description for the properties for $\bot$, we need to specify when we can generate a new rule introducing $\bot$ from two other rules.

$$\frac{\Gamma \Rightarrow^x \alpha \otimes^y_{t_\alpha} a^t \otimes_{t_a} \bot \quad \Delta \hookrightarrow \neg a^{t'} \otimes_{t''} \bot}{\Gamma, \Delta \Rightarrow^x \alpha \otimes^y_{t_\alpha} a^t \otimes_{t'-1} \bot} t < t' \text{ and } y \in \{a, m\} \tag{9}$$

The meaning of the above inference rule is that if we have a norm determining the termination of an obligation, then we can encode the obligation, the time when the obligation comes to force and the time when the norm terminates its normative effect. The idea behind a norm like $a^t \Rightarrow^x b^{t'}$ is the obligation b enters into force from time t'. Here we assume the intuition developed in [16] that a 'new' rule takes precedence over a conclusion obtained in the past and carrying over to the current moment by persistence. Thus if we have a rule $c^{t_c} \Rightarrow \neg b^{t''}$ with $t'' > t'$ the rule for $\neg c^{t''}$ effectively terminates the force of the obligation b. Consider the following instance of the rule

$$\frac{r_1 : a^5 \Rightarrow^m b^{10} \otimes_{15} \bot \quad c^{12} \Rightarrow^a \neg b^{12} \otimes_{20} \bot}{a^5, c^{12} \Rightarrow^m b^{10} \otimes_{11} \bot}$$

In this case r_1 puts the obligation of b in force in the interval from 10 to 15, and r_2 enforces $\neg b$ from 12 to 20, thus when both conditions to apply, the effective time when the obligation of b is in force is from 10 to 11 (after that the obligation $\neg b$ enters into force).

The $\otimes$ operator, introduced in [14], is a substructural operator corresponding to the comma on the right hand side of a sequent in sequent system. In a classical sequent system both the left hand side and right hand side of a sequent are set of formulas, thus the order of the formulas does not matter, and properties like contraction and duplication hold. In [14] we established the equivalence between $\alpha \otimes a \otimes \beta \otimes a \otimes \gamma$ and $\alpha \otimes a \otimes \beta \otimes \gamma$. This states that if a literal occurs multiple times, we can remove all but the first occurrence. We turn our attention to study conditions under which we have contraction for the various (combination of) $\otimes$ operators we have.

Tables 1 and 2 give the conditions to remove duplicates of the same atom. Consider for example, the instance ${}^p a^{10} \otimes^m a^0 \otimes_{20} \bot$ of the reduction Punctual-Maintenance in

Table 1. Reductions to $\bot$

Punctual-Punctual	${}^p a^t \otimes^x_t \beta \otimes^p_{t_\beta} a^{t'} \otimes^y_{t'} \gamma \equiv^p a^t \otimes^x_t \beta \otimes^p_{t_\beta} \bot \otimes^y \gamma$	$t = t'$
Punctual-Achievement	${}^p a^t \otimes^x_t \beta \otimes^a_{t_\beta} a^{t_s} \otimes^y_{t_e} \gamma \equiv^p a^t \otimes^x_t \beta \otimes^a_{t_\beta} \bot \otimes^y \gamma$	$t = t_s = t_e$
Punctual-Maintenance	${}^p a^t \otimes^x_t \beta \otimes^m_{t_\beta} a^{t_s} \otimes^y_{t_e} \gamma \equiv^p a^t \otimes^x_t \beta \otimes^m_{t_\beta} \bot \otimes^y \gamma$	$t \in [t_s, t_e]$
Achievement-Punctual	${}^a a^{t_s} \otimes^x_{t_e} \beta \otimes^p_{t_\beta} a^{t'} \otimes^y_{t'} \gamma \equiv^a a^{t_s} \otimes^x_{t_e} \beta \otimes^p_{t_\beta} \bot \otimes^y \gamma$	$t' = t_s = t_e$
Achievement-Achievement	${}^a a^{t_s} \otimes^x_{t_e} \beta \otimes^a_{t_\beta} a^{t'_s} \otimes^y_{t'_e} \gamma \equiv^a a^{t_s} \otimes^x_{t_e} \beta \otimes^a_{t_\beta} \bot \otimes^y \gamma$	$[t'_s, t'_e] \subseteq [t_s, t_e]$
Achievement-Maintenance	${}^a a^{t_s} \otimes^x_{t_e} \beta \otimes^m_{t_\beta} a^{t'_s} \otimes^y_{t'_e} \gamma \equiv^a a^{t_s} \otimes^x_{t_e} \beta \otimes^m_{t_\beta} \bot \otimes^y \gamma$	$[t_s, t_e] \cap [t'_s, t'_e] \neq \emptyset$
Maintenance-Punctual	${}^m a^{t_s} \otimes^x_{t_e} \beta \otimes^p_{t_\beta} a^{t'} \otimes^y_{t'} \gamma \equiv^m a^{t_s} \otimes^x_{t_e} \beta \otimes^p_{t_\beta} \bot \otimes^y \gamma$	$t' = t_s = t_e$
Maintenance-Achievement	${}^m a^{t_s} \otimes^x_{t_e} \beta \otimes^a_{t_\beta} a^{t'_s} \otimes^y_{t'_e} \gamma \equiv^m a^{t_s} \otimes^x_{t_e} \beta \otimes^a_{t_\beta} \bot \otimes^y \gamma$	$t_s = t_e = t'_s = t'_e$
Maintenance-Maintenance	${}^m a^{t_s} \otimes^x_{t_e} \beta \otimes^m_{t_\beta} a^{t'_s} \otimes^y_{t'_e} \gamma \equiv^m a^{t_s} \otimes^x_{t_e} \beta \otimes^m_{t_\beta} \bot \otimes^y \gamma$	$[t_s, t_e] \subseteq [t'_s, t_e]$

Table 2. Reductions to $\top$

Punctual-Punctual	${}^{p}a^{t} \otimes^{x}_{t} \beta \otimes^{p}_{t_\beta} \sim a^{t'} \otimes^{y}_{t'} \gamma \equiv^{p} a^{t} \otimes^{x}_{t} \beta \otimes^{p}_{t_\beta} \top \otimes^{y} \gamma$	$t = t'$
Punctual-Achievement	${}^{p}a^{t} \otimes^{x}_{t} \beta \otimes^{a}_{t_\beta} \sim a^{t_s} \otimes^{y}_{t_e} \gamma \equiv^{p} a^{t} \otimes^{x}_{t} \beta \otimes^{a}_{t_\beta} \top \otimes^{y} \gamma$	$t \in [t_s, t_e]$
Punctual-Maintenance	${}^{p}a^{t} \otimes^{x}_{t} \beta \otimes^{m}_{t_\beta} \sim a^{t_s} \otimes^{y}_{t_e} \gamma \equiv^{p} a^{t} \otimes^{x}_{t} \beta \otimes^{m}_{t_\beta} \top \otimes^{y} \gamma$	$t = t_s = t_e$
Achievement-Punctual	${}^{a}a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{p}_{t_\beta} \sim a^{t'} \otimes^{y}_{t'} \gamma \equiv^{a} a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{p}_{t_\beta} \top \otimes^{y} \gamma$	$t' = t_s = t_e$
Achievement-Achievement	${}^{a}a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{a}_{t_\beta} \sim a^{t'_s} \otimes^{y}_{t'_e} \gamma \equiv^{a} a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{a}_{t_\beta} \top \otimes^{y} \gamma$	$[t_s, t_e] \subseteq [t'_s, t'_e]$
Achievement-Maintenance	${}^{a}a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{m}_{t_\beta} \sim a^{t'_s} \otimes^{y}_{t'_e} \gamma \equiv^{a} a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{m}_{t_\beta} \top \otimes^{y} \gamma$	$[t'_s, t'_e] \subseteq [t_s, t_e]$
Maintenance-Punctual	${}^{m}a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{p}_{t_\beta} \sim a^{t'} \otimes^{y}_{t'} \gamma \equiv^{m} a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{p}_{t_\beta} \top \otimes^{y} \gamma$	$t' = t_s = t_e$
Maintenance-Achievement	${}^{m}a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{a}_{t_\beta} \sim a^{t'_s} \otimes^{y}_{t'_e} \gamma \equiv^{m} a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{a}_{t_\beta} \top \otimes^{y} \gamma$	$[t_s, t_e] \subseteq [t'_s, t'_e]$
Maintenance-Maintenance	${}^{m}a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{m}_{t_\beta} \sim a^{t'_s} \otimes^{y}_{t'_e} \gamma \equiv^{m} a^{t_s} \otimes^{x}_{t_e} \beta \otimes^{m}_{t_\beta} \top \otimes^{y} \gamma$	$t_s = t_e = t'_s = t_e$

Table 1, where the primary obligation is to have a at time 10, and whose compensation is to maintain a from 0 to 20. To trigger the secondary obligation we should have the violation of the primary obligation. This means that $\sim a$ holds at 10, but this implies that it is not possible to maintain a from 0 to 20, thus it is not possible to compensate the violation of the primary obligation. Notice that in several cases the reductions are possible only when the intervals are just single instants.

Introduction Rules. Besides the properties given so far the full meaning of the $\otimes$ operator is given by the rules to introduce (and modify) the operator. The general idea of the introduction rules is to determine the conditions under which a norms is violated. If these conditions imply a particular obligation then, then this obligation can be seen as a compensation of the norm the conditions violate.

$$\frac{\Gamma \Rightarrow^{x} \alpha \otimes^{p}_{t_\alpha} b^{t_b} \otimes^{Y}_{t_b} \gamma \quad \Delta, \neg b^{t_b} \hookrightarrow^{z} \delta}{\Gamma, \Delta \Rightarrow^{x} \alpha \otimes^{p}_{t_\alpha} b^{t_b} \otimes^{z}_{t_b} \delta} \otimes I_p$$

The punctual obligation $O^{p} b^{t_b}$ (implied by the first sequent) holds only at time t_b thus the only instant when the obligation can be violated is exactly t_b.

Rule $\otimes I_p$ is the standard rule to introduce a (novel) compensation or CTD (see [14] for further discussion about it).

$$\frac{\Gamma \Rightarrow^{x} \alpha \otimes^{m}_{t_\alpha} b^{t_s} \otimes^{y}_{t_e} \beta \quad \Delta, \Theta \Rightarrow^{z} \delta}{\Gamma, \Delta \Rightarrow^{x} \alpha \otimes^{m}_{t_\alpha} b^{t'_s} \otimes^{z}_{t'_e} \delta} \otimes I_m \text{ where } \Theta = \{\sim b^{t'} : t_s < t'_s \leq t' \leq t'_e \leq t_e\}.$$

The introduction rule for $\otimes^{m}$ defines a slice of the interval where a specific compensation of the violation holds. This conditions requires a rule whose antecedent contains the complement of a maintenance obligation in the head of the other rule, such that the literal is temporalised with the last n consecutive instants. For example given the rules

$$a^{10} \Rightarrow^{m} b^{10} \otimes_{20} \bot \qquad c^{15}, \neg b^{17}, \neg b^{18}, \neg b^{19} \Rightarrow^{p} d^{20} \otimes_{20} \bot$$

we can derive the new rule

$$a^{10}, c^{15} \Rightarrow^{m} b^{17} \otimes^{p}_{19} d^{20} \otimes_{20} \bot$$

The conditions to derive a new compensation rule for an achievement obligation are more complicated. As we have seen from the previous two cases, the structure of the introduction rules is that the negation of a consequent of a norm is a member of the antecedent of another norm (with the appropriate time). This ensures that the antecedent of the norm is a breach of the other one. The idea is the same for achievement obligations, but now detecting a violation is more complex.

$$\frac{\Gamma \Rightarrow^{x} \alpha \otimes^{a}_{t_\alpha} a^{t^s_a} \otimes^{x}_{t^e_a} \beta \quad \Delta, O a^{t'_a}, \sim a^{t'_a} \Rightarrow^{z} \delta \quad \{\Delta, \sim a^{t''_a} \Rightarrow^{z} \delta\}_{\forall t''_a : t^s_a < t'_a \leq t''_a \leq t^e_a}}{\Gamma, \Delta \Rightarrow^{x} \alpha \otimes^{a}_{t_\alpha} a^{t^s_a} \otimes^{z}_{t'_a} \delta} \otimes I_a$$

The idea behind the introduction of a compensation for achievement obligation is that we have to determine that the obligation has not been fulfilled at a time before the deadline and for all instant greater or equal to it the complement is required. Essentially, the $\otimes I_a$ amounts to shortening the deadline for an achievement obligation.

$$\frac{a^1 \Rightarrow^{a} b^5 \otimes_{10} \bot \quad Ob^8, \neg b^8 \Rightarrow^{p} c^{15} \otimes_{15} \bot \quad \neg b^9 \Rightarrow^{p} c^{15} \otimes_{15} \bot \quad \neg b^{10} \Rightarrow^{p} c^{15} \otimes_{15} \bot}{a^1 \Rightarrow^{a} b^5 \otimes^{p}_{8} c^{15} \otimes_{15} \bot}$$

The first norm initially sets the deadline by when b at to be achieved to 10. The last n norms, in this case $n = 2$, have as premises the opposite of an obligation of the first norm covering the last n instant of the force period of the obligation and the same conclusion. This means that refraining to fulfill the obligation in the last n instants results in the same consequence. The last part is to assess that we have a violation. This is achieved by the second norm; here, we have the obligation in the antecedent (an achievement obligation is no longer in force in two cases: we are after the deadline or the content of the obligation has been achieved), thus the condition Ob^8 and $\neg b^8$ is to ensure that the obligation is still in force at the time, and the combination of the norms ensures that from now on not fulfilling the obligation results in the same compensation.

Subsumption. The inference rules combine premises in such a way as the deontic content of at least one of them is included by the conclusion. Consequently, some original rules are no longer needed. To deal with this issue we introduce the notion of subsumption. A norm subsumes a second when the behaviour of the second norm (its compliance condition) is implied by the first one. Here below is an example illustrating this idea.

Example 6. Consider the following norms:

$$r: \ Invoice^{t} \Rightarrow^{a} Pay^{t} \otimes^{p}_{t+6} PayInterest^{t+7} \otimes_{t+7} \bot$$
$$r': Invoice^{t}, OPay^{t+6}, \neg Pay^{t+6} \Rightarrow^{a} PayInterest^{t+7} \otimes_{t+8} \bot$$

The first norm says that after the seller sends the invoice, the buyer has the achievement obligation to pay within 7 days, otherwise immediately after the violation the buyer has to pay the principal plus the interest (punctual obligation to pay at $t + 7$). According to the second norm, given the same set of circumstances *Invoice* at time t, if we have still the obligation on the seventh day after the invoice receipt date and the payment is not made yet, we have the achievement obligation to pay the interest by the eighth day. However, (a) the primary obligation of r' obtains when we have a violation of the

primary obligation of r; (b) after the primary obligation of r is violated, complying with its secondary obligation entails complying with the primary obligation of r' (but not vice versa); (c) hence, r is more general than r', and so the latter can be discarded.

In what follows, Definition 10 characterizes the concept of subsumption that we have informally illustrated in Example 6. Since we need to check whether the compliance of a norm guarantees the compliance of another norm (the subsumed one), we provide below the following auxiliary definitions to establish (a) Definition 7: the modes with which the compliance conditions for one obligation covers the compliance conditions of another one; (b) Definition 8: when the compliance conditions of an $\otimes$-chain cover the compliance conditions of another $\otimes$-chain; (c) Definition 9: the conditions under which a literal belonging to an $\otimes$-chains is violated (indeed, subsumption allows to remove the norms whose applicability conditions require to violate another norm, while these conditions are encoded in the $\otimes$-chain of the subsuming norm).

Definition 7. *Let $X, Y \in \{a, m, p\}$. Then, $Y \sqsubseteq X$ iff 1) if $Y = a$, then $X \in \{a, m, p\}$; 2) if $Y = m$, then $X = m$; 3) if $Y = p$, then $X \in \{p, m\}$.*

Definition 8. *Let*

$$\gamma = {}^{x_1}c_1^{t_{c_1}} \otimes^{x_2}_{t'_{c_1}} c_2^{t_{c_2}} \otimes^{x_3}_{t'_{c_2}} \cdots \otimes^{x_j}_{t'_{c_{j-1}}} c_j^{t_{c_j}} \qquad \beta = {}^{y_1}b_1^{t_{b_1}} \otimes^{y_2}_{t'_{b_1}} b_2^{t_{b_2}} \otimes^{y_3}_{t'_{b_2}} \cdots \otimes^{y_k}_{t'_{b_{k-1}}} b_k^{t_{b_k}}$$

be $\otimes$-chains. The $\otimes$-chain γ d-includes *the $\otimes$-chain β iff*

1. $j = k$,
2. $c_i = b_i$,
3. $y_i \sqsubseteq x_i$;
4. *(a) if $y_i = a$, then $t'_{c_i} \geq t_{b_i}$ when $x_i = m$, otherwise $t_{c_i} = t_{b_i}$ and $t'_{c_i} \leq t'_{b_i}$;*
   (b) if either $y_i = m$ or $y_i = p$, then $t_{c_i} \leq t_{b_i}$ and $t'_{c_i} \geq t'_{b_i}$

where $1 \leq i \leq j, k$.

Definition 9. *Let ${}^{x_1}c_1^{t_{c_1}} \otimes^{x_2}_{t'_{c_1}} c_2^{t_{c_2}} \otimes^{x_3}_{t'_{c_2}} \cdots \otimes^{x_j}_{t'_{c_{j-1}}} c_j^{t_{c_j}}$ be any $\otimes$-chain. For any c_i, where $1 \leq i \leq j$, a set X* violates *c_i iff*

1. *if $x_i = a$, then $X = \{Oc_i^{t'_{c_i}}, {\sim}c_i^{t'_{c_i}}\}$;*
2. *if $x_i = m$ or $x_i = p$, then $X \subseteq \{{\sim}c_i^t | t_{c_i} \leq t \leq t'_{c_i}\}$.*

Definition 10. *Let $r_1 : \Gamma \Rightarrow \alpha \otimes \beta \otimes \gamma$ and $r_2 : \Delta \Rightarrow \delta$ be two rules, where α, β, γ, and δ are $\otimes$-chains such that $\gamma = {}^{z_1}c_1^{t_{c_1}} \otimes^{z_2}_{t'_{c_1}} c_2^{t_{c_2}} \otimes^{z_3}_{t'_{c_2}} \cdots \otimes^{z_l}_{t'_{c_{l-1}}} c_l^{t_{c_l}}$.*

Then r_1 subsumes *r_2 iff*

1. *$\Gamma = \Delta$ and α d-includes δ; or*
2. *$\Gamma \cup X = \Delta$, where X violates all elements in α, and β d-includes δ; or*
3. *$\Gamma \cup Y = \Delta$, where Y violates all elements in β, and $\alpha \otimes {}^{z_1}c_1^{t_{c_1}} \otimes^{z_2}_{t'_{c_1}} c_2^{t_{c_2}} \otimes^{z_3}_{t'_{c_2}} \cdots \otimes^{z_n}_{t'_{c_{n-1}}} c_n^{t_{c_n}}$ d-includes δ, where $n \leq l$.*

4 Proof Conditions

We introduce the conditions that allow us to determine whether an obligation is in force at time t (and the type of obligation as well). The problem reduces to determine whether a (temporalised) literal follows from a theory, in other terms whether we can derive the (temporalised) literal. In addition the conditions allow us to establish whether a theory has been complied with. In Definition 1 we stated that a deontic expression extends an $\otimes$-chain with $\perp$ at the end. Thus effectively the penultimate element of a deontic expression identifies the 'last chance' to be compliant. After that the deontic expression results in a situation that cannot be complied with anymore. Hence, checking whether a theory is not compliant amounts to deriving $\perp$.

Definition 11. *A tagged literal is an expression* $\#l$, *where* $\# \in \{+\partial, -\partial, +\partial^p, -\partial^p, +\partial^a, -\partial^a, +\partial^m, -\partial^m\}$.

Definition 12. *A proof P is a sequence $P(1)\ldots P(n)$ of tagged literals satisfying the proof conditions given in Definitions 15, 16, 17 and 18. Each $P(i)$, $1 \leq i \leq n$ is called a line of the proof. Given a proof P, $P(1..n)$ denotes the first n lines of the proof.*

Definition 13. *A rule r is applicable at index i in a proof P at line $P(n+1)$ iff*[4]

1. $\forall a \in A(r)$:
   (a) *if* $a \in TLit$, *then* $a \in F$, *and*
   (b) i. *if* $a = Ol^t$, *then* $+\partial l^t \in P(1..n)$,
   ii. *if* $a = \neg Ol^t$, *then* $-\partial l^t \in P(1..n)$; *and*
2. $\forall c_j \in C(r), 1 \leq j \leq i$:
   (a) *if* $mode(c_j) = p$, *then* $c_j \notin F$ *or* $\sim c_j \in F$,
   (b) *if* $mode(c_j) = a$, *then* $\forall t$, $start(c_j) \leq t \leq end(c_j)$, $c_j^t \notin F$ *or* $\sim c_j^t \in F$,
   (c) *if* $mode(c_j) = m$, *then* $\exists t$, $start(c_j) \leq t \leq end(c_j)$, $c_j^t \notin F$ *or* $\sim c_j^t \in F$.

Definition 14. *A rule r is discarded at index i in a proof P at line $P(n+1)$ iff*

1. $\exists a \in A(r)$:
   (a) *if* $a \in TLit$, *then* $a \in F$; *or*
   i. *if* $a = Ol^t$, *then* $-\partial l^t \in P(1..n)$,
   ii. *if* $a = \neg Ol^t$, *then* $+\partial l^t \in P(1..n)$; *or*
2. $\forall c_j \in C(r), 1 \leq j \leq i$
   (a) *if* $mode(c_j) = p$, *then* $c_j \in F$,
   (b) *if* $mode(c_j) = a$, *then* $\forall t$, $start(c_j) \leq t \leq end(c_j)$, $c_j^t \in F$,
   (c) *if* $mode(c_j) = m$, *then* $\exists t$, $start(c_j) \leq t \leq end(c_j)$, $c_j^t \in F$.

In the proof conditions below we will simply use applicable/discarded at index i, instead of applicable/discarded at index i in the proof P at line $P(n+1)$.

All proof tags presented in the paper will be defined according the principle of strong negation [12]. According to it, the pair of tags $+\#$ and $-\#$ are the strong negation of each

[4] In the following, if ${}^{x_1}c_1^{t_{c_1}} \otimes_{t'_{c_1}}^{x_2} \cdots \otimes_{t'_{c_{j-1}}}^{x_j} c_j^{t_{c_j}} \otimes_{t'_j}^{x_{j+1}} \cdots \otimes_{t'_n} \perp$ is an $\otimes$-chain of length $n+1$, $mode(c_j) = x_j$, $start(c_j) = t_{c_j}$, and $end(c_j) = t'_{c_j}$.

other, where the strong negation is a function replacing/exchanging: $\forall$ and $\exists$, conjunctions and disjunctions, and 'applicable' and 'discarded'. For space reasons, we provide the definition of both the positive and negative proof tags for punctual obligation (i.e., $+\partial^p$ and $-\partial^p$), and only the positive definition of the proof tags for achievement and maintenance obligations; the corresponding negative proof tags can be derived using the above mentioned principle.

Definition 15. (Proof Conditions for $\pm\partial^p$)

If $P(n+1) = +\partial^p p^t$ *then*
(1) $\exists r \in R^p_{\Rightarrow}[p^t, i]$ *r is applicable at index i and*
(2) $\forall s \in R[\sim p^t, j]$*, either*
  (2.1) s is discarded at index j or
  (2.2) $\exists w \in R[p^t, k]$ *such that w is applicable at k and* $w \succ s$*.*

If $P(n+1) = -\partial^p p^t$ *then*
(1) $\forall r \in R^p_{\Rightarrow}[p^t, i]$ *either r is discarded at i, or*
(2) $\exists s \in R[\sim p^t, j]$ *such that*
  (2.1) r is applicable at index j and
  (2.2) $\forall w \in R[p^t, k]$ *either w is discarded at k or* $s \not\succ w$*.*

The proof conditions above are essentially a simple combination of the condition for $\otimes$ given in [12] and those for punctual obligation of [16]. To prove $+\partial^p a^t$, there must be a rule for a^t such that all the antecedents have to be provable, and for all elements preceding a^t in the head, we have to ensure that a violation occurred. This means that we have to examine the mode of the conclusions at indexes lower that the index of a^t, and then for a punctual obligation we have to see that the content of the obligation did not happen at t. We have two cases: the first is that we do not have a^t in the set of facts, and second we have the opposite, i.e., we have $\sim a^t$. For an achievement obligation we have to check that for all instants in the interval the same condition as that for a punctual obligation is satisfied, while for a maintenance obligation, a violation occurs when the condition holds for at least one instant of time in the interval. Condition (2.1) and (2.2) are the usual conditions of Defeasible Logic, that is: we have to verify that rules for the opposite either do not fire (2.1), they are not applicable, or (2.2) they are defeated by applicable rules for the conclusion we want to prove.

Definition 16. Proof Conditions for $\pm\partial^a$)

If $P(n+1) = +\partial^a p^t$ *then*
(1) $\exists r \in R^a_{\Rightarrow}[p^t, i]$ *r is applicable at index i and*
(2) $\forall s \in R[\sim p^t, j]$*, either*
  (2.1) s is discarded at index j or
  (2.2) $\exists w \in R[p^t, k]$ *such that w is applicable at k and* $w \succ s$*; or*
(3) $\exists x \in R^a_{\Rightarrow}[p^{t'}, i]$*,* $t' < t$*,* $end(p^{t'}) \geq t$ *and*
  (3.1) x is applicable at index i, and
  (3.2) $\forall y \in R[\sim p^{t''}, j]$*,* $t' \leq t'' < t$ *either*
    (3.2.1) y is discarded at j or
    (3.2.3) $\exists z \in R[p^{t''}, k]$*, z is applicable at k and* $z \succ y$*; and*
  (3.3) $\forall t''', t'' < t''' \leq t$*,* $p^{t'''} \notin F$*.*

The conditions for $+\partial^a p^t$ are similar to those for punctual obligations. The differences are that we have to consider persistence, clause (3). This means that we could have derived the obligation in the past, let us say at time t', and the obligation has not been terminated since them. We have two ways to terminate it: there is a rule for the opposite that is applicable between t and t' (3.2) see [16], or the obligation has been already fulfilled (3.3).

Definition 17. (Proof Conditions for $\pm\partial^m$)

If $P(n+1) = +\partial^m p^t$ *then*
(1) $\exists r \in R^m_{\Rightarrow}[p^t, i]$ *r is applicable at index i and*
(2) $\forall s \in R[\sim p^t, j]$, *either*
  (2.1) s is discarded at index j or
  (2.2) $\exists w \in R[p^t, k]$ *such that w is applicable at k and* $w \succ s$; *or*
(3) $\exists x \in R^m_{\Rightarrow}[p^{t'}, i]$, $t' < t$, $end(p^{t'}) \geq t$ *and*
  (3.1) x is applicable at index i, and
  (3.2) $\forall y \in R[\sim p^{t''}, j]$, $t' \leq t'' < t$ *either*
    (3.2.1) y is discarded at j or
    (3.2.3) $\exists z \in R[p^{t''}, k]$, *z is applicable at k and* $z \succ y$.

The conditions for maintenance obligations are the same as those for achievement obligation with the difference that fulfilling the obligation does not terminate it.

Definition 18 (Proof Condition for $\pm\partial$). *If* $P(n+1) = +\partial p^t$, *then either* $+\partial^p p^t \in P(1..n)$, *or* $+\partial^a p^t \in P(1..n)$, *or* $+\partial^{a^p r} p^t \in P(1..n)$, *or* $+\partial^m p^t \in P(1..n)$.

If $P(n+1) = -\partial p^t$, *then* $-\partial^p p^t \in P(1..n)$, *and* $-\partial^a p^t \in P(1..n)$, *and* $+\partial^{a^p r} p^t \in P(1..n)$, *and* $+\partial^m p^t \in P(1..n)$.

Definition 19. *Given a theory D, the* universe *of D* (U^D) *is the set of all the atoms occurring in D. The* extension E^D *of D is a structure* (∂^+, ∂^-), *where, for* $X \in \{p, a, m\}$, $\partial^+_D = \{l^t : D \vdash +\partial^X l^t\}$ *and* $\partial^-_D = \{l^t : D \vdash -\partial^X l^t\}$.

Example 7. Consider the following theory:

$$
\begin{aligned}
F &= \{Invoice^t, \neg Pay^t, \neg Pay^{t+1}, PayInterest^{t+2}, Defective^t\} \\
R &= \{r_1 : Invoice^t \Rightarrow^a Pay^t \otimes_{t+1} \bot \\
&\qquad r_2 : Invoice^t, OPay^{t+1}, \neg Pay^{t+1} \Rightarrow^a PayInterest^{t+2} \otimes_{t+3} \bot, \\
&\qquad r_3 : Defective^t \rightsquigarrow \neg Pay^t\} \\
\succ &= \{r_1 \succ r_3\}
\end{aligned}
$$

The first two norms basically describe the same situation of Example 6: the only difference is that here we have not yet applied any introduction rule for $\otimes$. r_3 states that, if the delivered good is defective, the customer is allowed not to pay. The facts trigger r_1, thus we derive the obligation to pay by $t+1$ (starting from t): also r_3 is triggered but is weaker than r_1. The obligation to pay is however not fulfilled by F. Since $\neg Pay^t \in F$, we obtain $OPay^{t+1}$ from r_1, which contributes to triggers r_2, thus obtaining the obligation to pay the interest by $t+3$ (starting from $t+2$). Since the obligation to pay by $t+1$ is not fulfilled, the extension of the theory D contains $\bot$: r_1 was not complied with.

5 Checking Compliance

If we work on the idea that a set of facts may fulfill a set of norms even when some of these norms are violated (but such violations are always compensated), then the following definition of compliance does not suffice:

Definition 20 (Theory compliance). *A Defeasible Theory D is compliant iff $\bot \notin \partial^+_D$.*

Definition 20 is very simple and exploits the basic properties of any temporalized obligations: since all $\otimes$-chains have $\bot$ as their last element, they have an ultimate deadline beyond which we derive $\bot$: this amounts to saying that after that deadline we state that it is impossible to compensate. Since the proof conditions for our logic establish that an obligation in an $\otimes$-chain is derived only if the previous obligations in that chain are violated, if we have $\bot$ in the positive extension of a theory, this means that there is at least one obligation whose violation cannot be compensated. For instance, if we consider Example 7, according to Definition 20 the theory D is not compliant because the theory extension contains $\bot$. However, such a theory should be considered compliant, since norm r_2, which provides a compensation for the violation of r_1, is indeed fulfilled.

Normalisation Process. The inference rules $(\otimes I_p)$, $(\otimes I_m)$, and $(\otimes I_a)$ provide a method for representing the norms in a format that can be used to check the compliance of a theory. In fact, they allow for making explicit the hidden reparative relation between obligations. Once applied, the redundant rules can be removed. For instance, in Example 7 above, we could apply $(\otimes I_a)$ to r_1 and r_2 and obtain the new rule

$$r_3 : \mathit{Invoice}^t \Rightarrow^a \mathit{Pay}^t \otimes^a_{t+1} \mathit{PayInterest}^{t+2} \otimes_{t+3} \bot$$

Once r_3 is obtained, since r_2 is subsumed by r_3, then r_2 is deontically redundant and can be removed from the theory.

Formally, this process is called normalisation of a theory. Before presenting the process, some auxiliary notions are needed: (a) Definition 21 identifies all the instances of inference rules we can obtain from a theory; (b) since such instances allow to introduce new norms, we should establish when these norms can inherit the same strength qualifications (via $\succ$) of previous norms; we should also remove redundant norms and norm priorities (Definitions 22 and 23); (c) Definition 24 introduces the deductive closure of a theory under the inference conditions for $\otimes$.

Definition 21. *Let $D = (F, R \succ)$ be any defeasible theory. Any instance I of the inference rules $(\otimes I_p)$, $(\otimes I_m)$, and $(\otimes I_a)$ is* based on *D if each of the premises r_i and r_j of I is either (a) in R (in which case, the instance is* rooted*), or (b) is the conclusion of another instance of the inference rules $(\otimes I_p)$, $(\otimes I_m)$, and $(\otimes I_a)$ based on D.*

The instances of the inference rules $(\otimes I_p)$, $(\otimes I_m)$, and $(\otimes I_a)$ based on D are also called D-$\otimes$-instances.

Definition 22. *Let $D = (F, R \succ)$ be any defeasible theory. The superiority relation $\succ^\infty = \cup^\infty_{i=1} \succ_i$ is recursively defined as follows:*

- $\succ_0 = \succ \cup \{(j,k) |$ *j (or k) is the conclusion of a rooted D-$\otimes$-instance such that $k \in R$ (or $j \in R$) and, for any $i \in R$, $(i,k) \in \succ$ (or $(j,i) \in \succ$)* $\}$;

– $\succ_{i+1}=\succ_i \cup\{(j,k)|\ j$ *(or* k*) is the conclusion of a* D-$\otimes$*-instance such that* $(i,k)\in\succ_i$ *(or* $(j,i)\in\succ_i$*)* $\}$.

The relation $\succ^\infty$ *is called the* D-saturation of $\succ$.

Definition 23. *Let* $D=(F,R\succ)$ *be any defeasible theory. Let* $\mathscr{S}$ *be an operation over* D *defined as follows: if* $\Pi=\{r|r\in R,\exists r'\in R: r'$ *subsumes* $r\}$*, then*

$$\mathscr{S}(D)=\begin{cases} D' & \text{where } D'=(F,R',\succ') \text{ such that} \\ & R'=R-\Pi \text{ and} \\ & \succ'=\succ^\infty -\{(x,y)\in\succ \mid \text{either } x\in\Pi \text{ or } y\in\Pi\} \\ D & \text{otherwise} \end{cases} \tag{10}$$

Definition 24. *If* $D=(F,R\succ)$ *is any defeasible theory, let* $\vdash_\otimes$ *be the consequence relation defined by the inference rules* $(\otimes I_p)$*,* $(\otimes I_m)$*, and* $(\otimes I_a)$*. The* closure $(D,\vdash_\otimes)$ *of* D *under* $\vdash_i$ *is a theory* $D'=(F,R',\succ')$ *where (a)* R' *is the smallest set containing all elements of* R *and the conclusions of all* D-$\otimes$*-instances; (b)* $\succ'$ *is the* D-saturation of $\succ$.

Definition 25 (Theory normalisation). *The* normalisation D^∞ *of a theory* D *is a theory recursively obtained as follows: (a)* $D_0=D$*, (b)* $D_{i+1}=\mathscr{S}(D_i,\vdash_\otimes)$.

The inference rules and the rule removal via subsumption must be done several times in the appropriate order. The normalised theory is the fixed-point of the above constructions. At each step of the the procedure we have to first apply the inference rules for $\otimes$ and then the subsumption: suppose we have a theory containing the following three norms

$$r_1: f^{t_f}\Rightarrow^p a^{t_a}\otimes^p_{t_a} g^{t_g}\otimes_{t_g}\bot \qquad r_2: e^{t_e}\Rightarrow^p a^{t_a}\otimes^p_{t_a} b^{t_b}\otimes^p_{t_b} c^{t_c}\otimes^p_{t_c} d^{t_d}\otimes_{t_d}\bot$$
$$r_3: e^{t_e},\neg a^{t_a},\neg b^{t_b}\Rightarrow^p c^{t_c}\otimes_{t_c}\bot$$

The normalisation process would consist here in a single cycle leading to apply (i) $(\otimes Ip)$ to r_1 and r_3, thus producing $r_4: e^{t_e}, f^{t_f},\neg b^{t_b}\Rightarrow^p a^{t_a}\otimes^p_{t_a} c^{t_c}\otimes_{t_c}\bot$; (ii) subsumption and remove r_3. Notice that also r_2 subsumes r_3. However, if we apply subsumption first on this basis we have to delete r_3 and r_4 would be no longer derivable from r_1 and r_3 alone.

After a theory is normalised, Definition 20 can be safely applied, as all redundant rules are removed and all hidden reparative connections between obligations are made explicit.

Finally, notice that (i) the structure of the inference rules $(\otimes I_p)$, $(\otimes I_m)$, and $(\otimes I_a)$ states that one premise in all instances is subsumed by the conclusion and so is removed at the end of each step of the process; (ii) any defeasible theory contains only finitely many rules and each rule has finitely many elements; also the operation on which the construction is defined is monotonic [14].

If a superiority relation $\succ$ is consistent iff $(x,y),(y,x)\notin\succ$, then reason (i) above supports the following result:

Proposition 1. *For any defeasible theory* D*, the normalisation* $D^\infty=(F,R,\succ^\infty)$ *is such that* $\succ^\infty$ *is consistent.*

Also, so by standard set theory results, reason (ii) above supports the following:

Proposition 2. *The normalisation* D^∞ *of any defeasible theory* D *exists and is unique.*

6 Summary and Related Work

This paper extends the logic of violation proposed by [14] with time. This extension introduces a temporal dimension to the language saying when a norm produces its normative effects, or in other terms when the obligation (or, in general the normative position) corresponding to the normative effect of the norm is in force. An immediate consequence of the extended language is that it is possible to investigate the 'lifecycle' of obligations, and more precisely if there are deadlines to comply with an obligation. The extension is done to properly deal with the concept of legal compliance. To do this we argue that we have to handle different types of temporalised legal obligations and devise a normalisation procedure for making hidden conditions and reparative chains explicit. One open research issue is to investigate the complexity of this procedure, which requires, several times and in the appropriate order, to apply the inference rules for $\otimes$ and to remove redundant norms.

The literature on norm compliance is MAS is large (see, e.g., [5, 9, 20, 10, 1, 11, 17, 3, 19]). However, to the best of our knowledge no work in the field has so far attempted to model *legal* compliance pertaining to realistic systems where complex norm-enforcement mechanisms such as reparative chains are combined with a rich ontology of obligations as the one described here. In the literature on deontic logic, besides a few exceptions like [6], the research has mostly devoted extensive, but separate, efforts to the role of time for dealing with CTDs (since the seminal [25]) and on logical systems for modeling the concept deontic preference and CTDs (for an overview, [23]). This paper combines the two perspectives: in this sense, it also inherits from [14] the advantage of avoiding the most well-known CTD paradoxes. In this sense, [6] shares with our paper the same general view, but time is captured there at the semantic level and the language does not explicitly handle timestamps.

Combination of time and norms are not novel, as many combinations of temporal (or tense) logic and deontic logic have been investigated. However, temporal logic cannot handle specific times (or timestamps). Typically these logics can express the temporal relationships between events (represented by propositions), or the relationships between states. A possible solution to obviate this is to consider hybrid logic using nominals to capture nominals [22]. A nominal represents a proposition true only in one possible worlds. A temporal nominal represents a particular instant of time. In most temporal logic it is possible to model branching of time, and the meaning of nominals is not clear in this kind of situations (is the world corresponding to a nominal the same in all the branches, or we have different copies of the same instant of time?). On the other hand timestamps (and events) have been used in the Event Calculus. Event Calculus has been used to model the interaction between norms and time (see, e.g., [21]). However, Event Calculus is a dialect of first-order logic and Herrestad [18] has shown that these types of logic are not suitable to model normative reasoning in presence of violations.

References

1. Alberti, M., Gavanelli, M., Lamma, E., Chesani, F., Mello, P., Torroni, P.: Compliance verification of agent interaction: a logic-based software tool. Applied Artificial Intelligence 20(2-4), 133–157 (2006)

2. Antoniou, G., Billington, D., Governatori, G., Maher, M.: A flexible framework for defeasible logics. In: Proc. AAAI-2000. AAAI Press, Menlo Park (2000)
3. Boella, G., Broersen, J., van der Torre, L.: Reasoning about constitutive norms, counts-as conditionals, institutions, deadlines and violations. In: Bui, T.D., Ho, T.V., Ha, Q.T. (eds.) PRIMA 2008. LNCS (LNAI), vol. 5357, pp. 86–97. Springer, Heidelberg (2008)
4. Boella, G., van der Torre, L.: Fulfilling or violating obligations in multiagent systems. In: Procs. IAT 2004 (2004)
5. Bou, E., López-Sánchez, M., Rodríguez-Aguilar, J.A.: Adaptation of autonomic 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 (LNAI), vol. 4457, pp. 300–319. Springer, Heidelberg (2007)
6. Broersen, J., van der Torre, L.: Conditional norms and dyadic obligations in time. In: Proc. ECAI 2008. IOS Press, Amsterdam (2008)
7. Carmo, J., Jones, A.J.I.: Deontic logic and contrary to duties. In: Gabbay, D., Guenther, F. (eds.) Handbook of Philosophical Logic, 2nd edn. Kluwer, Dordrecht (2002)
8. Dastani, M., Grossi, D., Meyer, J.-J.C., Tinnemeier, N.: Normative multi-agent programs and their logics. In: Bordini, R., Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) Programming Multi-Agent Systems, Dagstuhl, Germany. Dagstuhl Seminar Proceedings, vol. 08361. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2008)
9. Esteva, M., Rosell, B., Rodríguez-Aguilar, J.A., Arcos, J.L.: Ameli: An agent-based middleware for electronic institutions. In: Proc. of AAMAS 2004, vol. 3394. ACM, New York (2005)
10. Pasquier, P., Flores, R., Chaib-draa, B.: Modelling flexible social commitments and their enforcement. In: Gleizes, M.-P., Omicini, A., Zambonelli, F. (eds.) ESAW 2004. LNCS (LNAI), vol. 3451, pp. 139–151. Springer, Heidelberg (2005)
11. Gaertner, D., Garcia-Camino, A., Noriega, P., Rodriguez-Aguilar, J.-A., Vasconcelos, W.: Distributed norm management in regulated multiagent systems. In: Proc. AAMAS 2007. ACM, New York (2007)
12. Governatori, G.: Representing business contracts in RuleML. International Journal of Cooperative Information Systems 14(2-3), 181–216 (2005)
13. Governatori, G., Hulstijn, J., Riveret, R., Rotolo, A.: Characterising deadlines in temporal modal defeasible logic. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 486–496. Springer, Heidelberg (2007)
14. Governatori, G., Rotolo, A.: Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations. Australasian Journal of Logic 44, 193–215 (2006)
15. Governatori, G., Rotolo, A.: An algorithm for business process compliance. In: Sartor, G. (ed.) Jurix 2008, pp. 186–191. IOS Press, Amsterdam (2008)
16. Governatori, G., Rotolo, A., Sartor, G.: Temporalised normative positions in defeasible logic. In: 10th International Conference on Artificial Intelligence and Law (ICAIL 2005), pp. 25–34 (2005)
17. Grossi, D., Aldewereld, H., Dignum, F.: Ubi lex, ibi poena: Designing norm enforcement in e-institutions. In: In Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems II. Springer, Heidelberg (2006)
18. Herrestad, H.: Norms and formalization. In: ICAIL, pp. 175–184 (1991)
19. Hübner, J.F., Boissier, O., Bordini, R.: From organisation specification to normative programming in multi-agent organisations. In: Dix, J., Leite, J., Governatori, G., Jamroga, W. (eds.) CLIMA XI. LNCS, vol. 6245, pp. 117–134. Springer, Heidelberg (2010)

20. López y López F., Luck, M., d'Inverno, M.: Constraining autonomy through norms. In: Proc. AAMAS 2002. ACM, New York (2002)
21. Marín, R.H., Sartor, G.: Time and norms: a formalisation in the event-calculus. In: ICAIL, pp. 90–99 (1999)
22. Smith, C., Rotolo, A., Sartor, G.: Temporal reasoning and mas. In: SNAMAS 2010 (2010)
23. van Benthem, J., Grossi, D., Liu, F.: Deontics = betterness + priority. In: Governatori, G., Sartor, G. (eds.) DEON 2010. LNCS, vol. 6181, pp. 50–65. Springer, Heidelberg (2010)
24. van der Torre, L., Boella, G., Verhagen, H. (eds.): Normative Multi-agent Systems, Special Issue of JAAMAS, vol. 17(1) (2008)
25. van Eck, J.: A system of temporally relative modal and deontic predicate logic and its philosophical applications. Logique et Analyse 25, 339–381 (1982)

Author Index