Fariba Sadri
Ken Satoh (Eds.)

LNAI 5056

Computational Logic in Multi-Agent Systems

8th International Workshop, CLIMA VIII
Porto, Portugal, September 2007
Revised Selected and Invited Papers

Springer

Lecture Notes in Artificial Intelligence 5056

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

Subseries of Lecture Notes in Computer Science

Fariba Sadri Ken Satoh (Eds.)

Computational Logic in Multi-Agent Systems

8th International Workshop, CLIMA VIII
Porto, Portugal, September 10-11, 2007
Revised Selected and Invited Papers

Springer

Volume Editors

Fariba Sadri
Imperial College London
Department of Computing
London, UK
E-mail: fs@doc.ic.ac.uk

Ken Satoh
National Institute of Informatics, Sokendai
Tokyo, Japan
E-mail: ksatoh@nii.ac.jp

Library of Congress Control Number: 2008937800

CR Subject Classification (1998): I.2.11, I.2, C.2.4, F.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-88832-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-88832-1 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12537563 06/3180 5 4 3 2 1 0

Preface

Multi-agent systems are communities of problem-solving entities that can exhibit varying degrees of intelligence. They can perceive and react to their environment, they can have individual or joint goals, for which they can plan and execute actions. Work on such systems integrates many technologies and concepts in artificial intelligence and other areas of computing as well as other disciplines. The agent paradigm has become widely popular and widely used in recent years, due to its applicability to a large range of domains, from search engines to educational aids to electronic commerce and trade, e-procurement, recommendation systems, simulation and routing, and ambient intelligence, to cite only some.

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

The CLIMA workshop series was founded to provide a forum for discussing, presenting, and promoting computational logic-based approaches in the design, development, analysis, and application of multi-agent systems.

The first workshop in these series took place in 1999 in Las Cruces, New Mexico, USA, under the title Multi-Agent Systems in Logic Programming (MASLP 1999), and was affiliated with ICLP 1999. The name of the workshop changed after that to Computational Logic in Multi-Agent Systems (CLIMA), and it has since been held in the UK, Cyprus, Denmark, USA, Portugal, and Japan. Further information about the CLIMA series, including past and future events and publications, can be found at `http://centria.di.fct.unl.pt/~clima`.

The eighth edition of CLIMA (CLIMA VIII) was held during September 10–11, 2007 in Porto, Portugal. It was co-located with ICLP 2007 (International Conference on Logic Programming). It received 30 submissions for regular papers and 3 submissions for system description papers, with about 40 delegates registered for participation. More details about the event can be found at
`http://research.nii.ac.jp/climaVIII/`.

This volume of post-proceedings contains revised and improved versions of 14 regular papers and 1 system description paper presented at the workshop, as well as the workshop invited paper, by Witteveen, Steenhuisen, and Zhang on plan co-ordination. All the papers included in the post-proceedings have gone through a thorough revision process, with at least two, and in the majority of cases, three rounds of reviewing, with at least seven reviews, for each paper.

The accepted papers cover a broad range of topics. The invited paper addresses problems of interdependencies and co-ordination of task-based planning in multi-agent systems where multiple agents are required to find a joint plan. The paper by Hommersom and Lucas explores how interval temporal logic can be extended to provide an operator describing failure, useful for modeling agents' failures in performing tasks, for example, because of the dynamic nature of the environment. Jamroga and Bulling describe how game theoretic concepts can be used within an alternating-time temporal logic to provide a formalism for reasoning about rational agents. Pereira et al. propose an Emotional-BDI model of agency that allows the emotions of fear, anxiety, and self-confidence, contributing to agent behavior as well as the more conventional beliefs, desires, and intentions. The paper by Boersen and Brunel investigates properties concerning persistence into the future of obligations that have not been fulfilled yet.

Hakli and Negri give a proof theory for multi-agent epistemic logic with operators for distributed knowledge, whereby a proposition follows from the totality of what the members of a group know. In their regular paper, Slota and Leite give a transformational semantics for Evolving Logic Programs (EVOLP), prove its soundness and completeness, and discuss the complexity of the transformation. They use this transformation for an implementation of EVOLP which is described in their systems paper.

Dennis et al. propose, and provide semantics for, extensions of programming languages that are based on the BDI type of agent models. The extensions are aimed at allowing representation of organizational structures in multi-agent systems. Bryl et al. extend Tropos, a well-known agent-oriented early requirements engineering framework, and then show how the extended framework can be translated into a framework of abductive logic programming. Costantini et al. describe a heuristic algorithm for agent negotiation that exploits projections in convex regions of admissible values and discuss its implementation and complexity.

A collection of three papers are related to argumentation. Hezart et al. propose context-sensitive defeasible rules for argumentation-based defeasible reasoning. Stranders et al. describe a fuzzy argumentation framework for reasoning about trsut in agents. Toni proposes an approach to the selection and composition of services based on assumption-based argumentation.

Belardinelli and Lomuscio focus on the topic of first-order epistemic logic and give a sound and complete axiomatization for quantified interpreted systems. Gore and Nguyen give a tableau calculus for a class of modal logics and argue its usefulness in reasoning about agent beliefs.

We would like to thank all the authors for responding to the call for papers with their high-quality submissions, and for responding to and taking account of the reviewers comments thoroughly in revising their contributions for inclusion in this volume. We are also grateful to the members of the CLIMA VIII Programme Committee and other reviewers for their valuable work in reviewing and discussing the submitted articles over several rounds of reviews. We would also like to thank the ICLP and the local organizers in Portugal for all their help and support. We are grateful to the local organizers for handling all the registration details and providing a very enjoyable social programme.

April 2008 Fariba Sadri
Ken Satoh

Organization

Workshop Chairs

Fariba Sadri	Imperial College London, UK
Ken Satoh	National Institute of Informatics and Sokendai, Japan

Programme Committee

José Júlio Alferes	New University of Lisbon, Portugal
Rafael H. Bordini	University of Durham, UK
Gerhard Brewka	University of Leipzig, Germany
Stefania Costantini	University of L'Aquila, Italy
Jürgen Dix	Technical University of Clausthal, Germany
Klaus Fischer	DFKI, Germany
Michael Fisher	University of Liverpool, UK
James Harland	RMIT, Australia
Hisashi Hayashi	Toshiba, Japan
Wiebe van der Hoek	University of Liverpool, UK
Katsumi Inoue	National Institute of Informatics and Sokendai, Japan
João Leite	New University of Lisbon, Portugal
Fangzhen Lin	Hong Kong University of Science and Technology, Hong Kong
Paola Mello	University of Bologna, Italy
John-Jules Ch. Meyer	Utrecht University, The Netherlands
Leora Morgenstern	IBM T.J. Watson Research Center, USA
Naoyuki Nide	Nara Women's University, Japan
Maurice Pagnucco	University of New South Wales, Australia
Jeremy Pitt	Imperial College London, UK
Enrico Pontelli	New Mexico State University, USA
Chiaki Sakama	Wakayama University, Japan
Abdul Sattar	Griffith University, Australia
Renate Schmidt	University of Manchester, UK
Tran Cao Son	New Mexico State University, USA
Kostas Stathis	Royal Holloway, University of London, UK
Michael Thielscher	Dresden University of Technology, Germany
Satoshi Tojo	Japan Advanced Institute of Science and Technology, Japan
Francesca Toni	Imperial College London, UK
Paolo Torroni	University of Bologna, Italy
Marina De Vos	University of Bath, UK
Cees Witteveen	Delft University of Technology, The Netherlands

CLIMA Steering Committee

Jürgen Dix	Technical University of Clausthal, Germany
Michael Fisher	University of Liverpool, UK
João Leite	New University of Lisbon, Portugal
Fariba Sadri	Imperial College London, UK
Ken Satoh	National Institute of Informatics and Sokendai, Japan
Francesca Toni	Imperial College London, UK
Paolo Torroni	University of Bologna, Italy

Additional Reviewers

Iara Almeida
Trevor Bench-Capon
Massimo Benerecetti
Jamal Bentahar
David Billington
Jeff Blee
Federico Chesani
Naoki Fukuta
Guido Governatori
Wojtek Jamroga
Marco Montali
Peter Novak
Julian Padget
Dmitry Tishkovsky
Pinar Yolum
Hans de Nivelle

Table of Contents

Invited Paper

Regular Papers

System Description Paper

Plan-Coordination Mechanisms and the Price of Autonomy

J. Renze Steenhuisen, Cees Witteveen, and Yingqian Zhang

Delft University of Technology,
Faculty of Electrical Engineering, Mathematics and Computer Science,
Mekelweg 4, 2628CD Delft, The Netherlands
{J.R.Steenhuisen,C.Witteveen,Yingqian.Zhang}@tudelft.nl

Abstract. Task-based planning problems for multi-agent systems require multiple agents to find a joint plan for a constrained set of tasks. Typically, each agent receives a subset of tasks to complete. Due to task interdependencies, such task allocations induce interdependencies between agents as well. These interdependencies will prevent the agents from making a plan for their subset of tasks independently from each other, since the combination of such autonomously constructed plans will most probably result in conflicting plans. Therefore, a plan-coordination mechanism is needed to guarantee a conflict-free globally feasible plan.

In this paper, we first present a brief overview of the main results achieved on plan coordination for autonomous planning agents, distinguishing between problems associated with deciding whether a coordination mechanism is necessary, designing an arbitrary coordination mechanism, and designing an optimal (minimal) coordination mechanism. After finding out that designing an optimal coordination mechanism is difficult, we concentrate on an algorithm that is able to find a (non-trivial) coordination mechanism that is not always minimal. We then discuss some subclasses of plan-coordination instances where this algorithm performs very badly, but also some class of instances where a nearly optimal coordination mechanism is returned.

Hereafter, we discuss the price of autonomy as a measure to determine the loss of (global) performance of a system due to the use of a coordination mechanism, and we offer a case study on multi-modal transportation where a coordination mechanism can be designed that offers minimal restrictions and guarantee nearly optimal performance. We will also place the use of these coordination mechanisms in a more general perspective, claiming that they can be used to reuse existing (single) agent software in a complex multi-agent environment.

Finally, we briefly discuss some recent extensions of our coordination framework dealing with temporal planning aspects.

Keywords: Complex tasks, planning, coordination, autonomy, multi-agent systems.

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 1–21, 2008.

1 Introduction

Task planning is the problem of finding a suitable plan for carrying out a *complex task*. By calling a task *complex*, we mean that *(i)* it consists of a number of elementary tasks that *(ii)* are interdependent, and *(iii)* (usually) cannot be completed by a single agent. For example, building a house constitutes a complex task since it consists of laying a foundation, then building the walls and the roof, then assembling the cases and painting the walls, assembling doors, etc. Typically, each of these tasks requires a different agent. Other examples that come to mind are building a large ship on a wharf, preparing a manned spacecraft for launch, and planning inter-modal transportation jobs.

Usually, we specify such a complex task $\mathcal{T}$ by stating the set T of elementary tasks to be completed, the set of capabilities required to execute each elementary task, and a set of constraints between the tasks that have to be respected in order to ensure a correct execution of the complex task.

The specification of the capabilities is important in deciding which agent will execute which (elementary) task. In this paper, however, we will not pay attention to this important (task allocation) problem[1] and simply assume that task allocation has been completed and that each agent has to find a way to achieve the set of tasks allocated to it. Therefore, we will simply omit the specification of the capabilities required.

A complex task then is a tuple $\mathcal{T} = (T, \prec)$ where T is a finite set of (elementary) tasks and $\prec$ is a partial order. Each elementary task $t \in T$, or simply *task*, is a unit of work that can be executed by a single agent. These elementary tasks are interrelated by a partially-ordered *precedence relation* $\prec$: A task t_1 is said to precede a task t_2, denoted by $t_1 \prec t_2$, if the execution of t_2 may not start until t_1 has been completed.[2] For example, building a wall of a house may not start before the work on the foundations has been completed.

Suppose that such a complex task $\mathcal{T} = (T, \prec)$ is given to a set of autonomous planning agents $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$. We assume that the tasks in T are assigned to the agents in $\mathcal{A}$ by some task assignment $f : T \rightarrow \mathcal{A}$, thereby inducing a *partitioning* $\{T_i\}_{i=1}^{n}$ of T, where $T_i = \{t \in T \mid f(t) = A_i\}$ denotes the set of tasks allocated to agent A_i.

Example 1. There are two agents involved in a complex construction task. Agent A_1 has to deliver bricks (t_1) to agent A_2 who will use them to build a wall (t_2). Agent A_2 also has to ensure that garbage is collected (t_3) and to deliver it to agent A_1 (t_4). Agent A_1 has to pickup the garbage (t_5), and then to transport it from the construction place to a dumping ground (t_6).

[1] How to find a suitable assignment for a set of agents is an interesting problem on its own [1,2].

[2] Of course, this interpretation of the precedence relation might vary. In general, we might interpret $t \prec t'$ as task t should {start, be completed} before task t' is allowed to {start, be completed}.

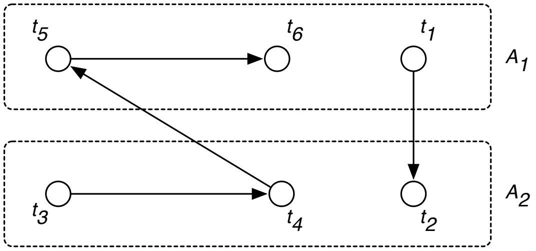


Fig. 1. A specification of a complex task. The tasks are allocated to two agents: A_1 and A_2. Task dependencies are denoted by arrows.

There are some natural precedence relations between the tasks that must be obeyed: $t_1 \prec t_2$, $t_3 \prec t_4$, $t_4 \prec t_5$, and $t_5 \prec t_6$. For an illustration of this complex task, see Figure 1.

As a result of this task assignment, each agent A_i also inherits the precedence constraints that apply to T_i, i.e., the set $\prec_i = \prec \cap (T_i \times T_i)$. These sets $\prec_i$ together constitute the set $\prec_{intra} = \bigcup_{i=1}^{n} \prec_i$ of *intra-agent* constraints, while the remaining set of constraints $\prec_{inter} = \prec \setminus \prec_{intra}$ constitutes the set of *inter-agent* constraints. So each agent A_i is now responsible for achieving the (complex) subtask $(T_i, \prec_i)$, while the agents depend on each other via the inter-agent constraints $\prec_{inter}$.

Example 2. Continuing Example 1, as the result of task allocation, the set T is partitioned into two sets $T_1 = \{t_1, t_5, t_6\}$ and $T_2 = \{t_2, t_3, t_4\}$. The set of inter-agent constraints is $\prec_{inter} = \{t_1 \prec t_2, t_4 \prec t_5\}$, while the intra-agent constraints are $\prec_1 = \{t_5 \prec t_6\}$ and $\prec_2 = \{t_3 \prec t_4\}$.

Due to time and resource constraints, an agent A_i will be forced to make a *plan* for its set of tasks T_i. Such a plan should be understood as the specification of some partially-ordered set of actions (plan steps) that satisfies the task constraints between the tasks given to the agents. It is not important to know exactly which planning tools are used by the agent and which set of primitive actions is used to construct the plan: What counts is that the plan respects all the task constraints $\prec_i$. As a consequence, we assume that, whatever plan/schedule representation the agents (internally) employ, the result of an internal plan P_i, developed by agent A_i for its set of tasks T_i, can always be specified by a structure $P_i = (T_i, \prec_i^*)$ that *refines* the structure $(T_i, \prec_i)$. We, therefore, require $\prec_i \subseteq \prec_i^*$ to hold, which means that an agent's plan must respect the original precedence relation $\prec_i$, but his plan may of course induce additional constraints.

Remark 1. The plan representation $P_i = (T_i, \prec_i^*)$ may be viewed as an *abstraction* of a concrete original plan P_i^c. Suppose that the concrete plan can be modelled as a partially-ordered set S of plan steps, i.e., $P_i^c = (S, <)$. Then we say that $P_i = (T_i, \prec_i^*)$ is an abstraction of P_i^c if there exists some function $steps : T_i \rightarrow 2^S$ mapping tasks to plan steps, such that *(i)* for every task $t \in T_i$, the concrete sub plan $(steps(t), <)$ of P_i^c realises task t, and *(ii)* for every $t, t' \in T_i : t \prec_i^* t'$

iff $\forall s \in steps(t)\ \forall s' \in steps(t') : s < s'$. Note that such a concrete plan might contain plan steps s that have no relationship with any task t.

Example 3. Continuing our previous example, our agent A_1 who has to deliver bricks (t_1) and to pickup the garbage (t_5), and then to carry it away (t_6) might construct a plan where he first drives with his truck to the construction place, will pickup the garbage, drives to the dumping place, takes some coffee, then loads some bricks and drives back to the construction place. This plan induces the following order on the tasks to be completed: $t_5 \prec t_6 \prec t_1$. Hence, his plan can be represented as $P_1 = (T_1, \prec_1^*)$ where $\prec_1^* = \{t_5 \prec_1^* t_6, t_6 \prec_1^* t_1\}$ is a refinement of $\prec_1$ and, therefore, is a valid plan for $(T_1, \prec_1)$.

Often, when more than one agent is involved in the task-planning process, we have to take into account some degree of *autonomy* that each of the participating agents might require when planning its part of the job. Here, autonomy has to be understood in the sense that an agent A_i is not able to predict exactly how the plan P_j of another agent A_j will look like for the set of tasks T_j given to that agent, and vice versa. However, as the result of allocating subsets of tasks to different agents, these agents might become interdependent as interdependent tasks might be allocated to different agents. The result of, on the one hand, *interdependency* and, on the other hand, *unpredictability* of individual planning outcomes, might easily lead to *conflicting* plans, in the sense that the structure $P = (T, (\bigcup_{i=1}^{n} \prec_i^*) \cup \prec_{inter})$, resulting from joining the individual plans, is no longer partially ordered. If this is the case, we call the resulting joint plan P *infeasible.*

Example 4. Continuing the example, suppose that agent A_2 develops a plan completely independent from agent A_1. This agent might decide to execute t_2 before t_3 and creates a plan $P_2 = (T_2, \prec_2^*)$ where $\prec_2^* = (t_2 \prec_2^* t_3, t_3 \prec_2^* t_4)$. The result of this plan together with the plan P_1 of agent A_1 (see Example 3) is depicted in Figure 2. As can be seen, the two plans together create a cycle and, therefore, constitute an infeasible joint plan.

Obviously, due to this combination of task dependencies and independent planning, some form of *coordination mechanism* [3] is needed to ensure that the results of the individual task-planning processes are jointly feasible.

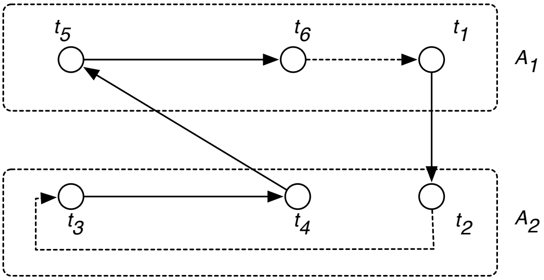


Fig. 2. Two plans of the agents, resulting in an infeasible joint plan

In this paper, we will address the following *plan-coordination problem* for self-interested planning agents: How to provide adequate coordination mechanisms for autonomous planning systems that want to plan independently and are not willing or able to revise their plans. In particular,

1. we present a framework for studying this plan-coordination problem,
2. we investigate the computational complexity of designing such plan-coordination mechanisms, and
3. we discuss the *price of autonomy*, investigating the additional costs incurred by independent planning.

Organisation. This paper is organised as follows. First, we present a brief overview of research on (plan) coordination to make clear what the relation is between our approach to plan coordination and other approaches. Second, we present our framework for plan coordination for self-interested agents and we give an overview of the complexity results obtained for designing adequate plan-coordination mechanisms. To prepare the reader for a case study of the application of coordination in logistic problems, we discuss a polynomial algorithm that can be used to find a suitable coordination mechanism. We prove that for some particular classes of complex tasks it delivers a nearly optimal mechanism. After introducing the price of autonomy, we apply this algorithm to a logistic problem, showing that in some particular cases plan-coordination mechanisms can be applied with almost negligible overhead, while ensuring autonomous planning.

2 Plan Coordination for Autonomous Planning Agents

2.1 Background

In general, a plan-coordination mechanism should guarantee, by possibly *re-designing* the original planning task, some minimal overall performance even if the agents are completely selfish [3].

Redesigning the original planning task usually imposes additional restrictions on the tasks to be completed in order to guarantee a minimal performance. In general, such additional restrictions will possibly affect both the planning freedom of the participating agents and the quality (cost, efficiency) of the plans they are able to develop. Therefore, we propose to define the quality of a plan-coordination mechanism both in terms of the *tightness* of the restrictions imposed, and the overall *plan quality* it ensures. Such a definition, however, would neglect an important factor that influences the choice of an adequate coordination mechanism: the *collaboration level* between the agents. Depending on this collaboration level we have to determine which coordination mechanism is suitable for the agents and this also determines the quality of the (optimal) coordination mechanism. For example, in approaches like [4,5] the authors propose to manage the coordinated planning and execution of the tasks by letting the agents keep each other informed about any changes (e.g., completed, new, or re-scheduled tasks). Here, a plan-coordination mechanism can be designed to facilitate inter-agent information exchange in order to improve the quality of the joint plan.

In other approaches, like the plan-merging approach [6], plans sometimes need to be revised when merging them into an overall plan, despite that agents are allowed to construct their plans independently. In this case, a plan-coordination mechanism could facilitate the exchange of mutual plan information in order to improve the overall plan quality by merging techniques.

It is clear that these approaches require the agents to be more or less *collaborative*, each agent willing to *inform* other agents about details of its individual plan and/or willing to *revise* its plan when necessary. However, such approaches are hardly usable if the agents are selfish (unwilling to communicate or revise their plans), or are not able to do so (e.g., in disaster-rescue operations, when communication is often impossible or difficult to establish).

Approaches to coordinating *self-interested, non-cooperative* agents, however, are mostly concentrating on task execution processes that do not require extensive forms of planning. Typical examples here are the study of (combinatorial) auctions for task allocation and coalition formation processes in multi-agent systems [1,7,8] and its relations to combinatorial optimisation problems (c.f. [9]). In these approaches, the tasks to be completed consist of a set of atomic tasks and each of the agents receives one single task of a subset of tasks. Even if the task description is more elaborate like in the Traderbots architecture [10], it is assumed that the set of subtasks does not require an elaborate planning process to execute. Therefore, the problem of identifying *planning constraints* and the problem of allocating them do not occur here.

2.2 The Plan Coordination Problem for Self-interested Agents

In contrast to the above mentioned approaches, some authors [11,12] studied the (computational) properties of coordination mechanisms that can be used for *selfish planning* agents. In such approaches, where the collaboration level is low or even absent, it is assumed that the agents *(i)* require autonomous planning of their part of the task, and *(ii)* are not willing to revise their own plan, thereby ruling out any form of collaboration either during planning or after planning. To meet such requirements, a plan-coordination mechanism should ensure that whatever plans are proposed by the individual agents, their combination always constitutes a feasible plan for the total set of tasks. The basic setup of such an approach has been discussed in the Introduction. Summarizing, the main ingredients of this framework are as follows:

1. We have a complex task $\mathcal{T} = (T, \prec)$, specifying a partially-ordered set of elementary tasks $t \in T$, and a set of self-interested agents $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ that require planning autonomy.
2. The tasks $t \in T$ are allocated to the agents A_i inducing a *task partitioning* $(\{T_i\}_{i=1}^n, \{\prec_i\}_{i=1}^n, \prec_{inter})$ where $(T_i, \prec_i)$ is a partially-ordered complex task allocated to agent A_i. Here, $\prec_i$ is the restriction of $\prec$ to $(T_i \times T_i)$, and $\prec_{inter}$ is the set of precedence relations between tasks allocated to different agents.

3. Each agent A_i is allowed to construct a plan P_i for its set of tasks T_i completely independent from the other agents. We assume that each such a plan P_i is representable as a partial order $P_i = (T_i, \prec_i^*)$ where $\prec_i \subseteq \prec_i^*$ (i.e., each plan P_i respects the *local* constraints $\prec_i$).

Now, the coordination problem we are facing can be stated as follows.

> *How to ensure that, whatever plans $P_i = (T_i, \prec_i^*)$ are developed by the individual agents A_i, each respecting the local constraints $\prec_i$, their combination, together with the set of inter-agent constraints, constitutes a feasible plan, that is, how to ensure that for every i and for every partially-ordered extension $\prec_i^*$ of $\prec_i$, the relation $(\bigcup_{i=1}^n \prec_i^*) \cup \prec_{inter}$ again is a partial order?*

As we have shown before [11,12], the only way to solve this problem is to design a coordination mechanism that adds, to each set of individual precedence constraints $\prec_i$, a set Δ_i of precedence constraints. The resulting set $\Delta = \bigcup_{i=1}^n \Delta_i$ is called a *coordination set* and the redesigned complex task is said to be *plan coordinated.*

Example 5. Let us consider the construction task from the previous examples. As we have shown in Example 4, there exists some combination of plans that turns out to be infeasible, creating a cycle. Therefore, this particular complex task instance is not plan coordinated. If, however, we change the task specification for agent A_1 adding the coordination set $\Delta = \Delta_1 = \{t_1 \prec_1 t_5\}$ to the complex task, no possible combination of plans developed by agent A_1 and A_2 will create a cycle (see Figure 3).

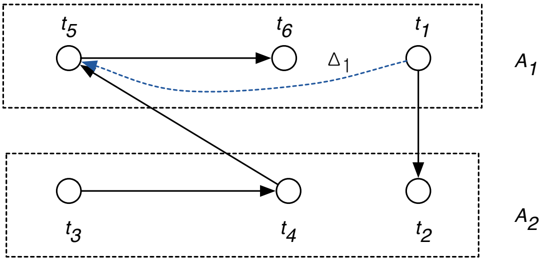


Fig. 3. Adding a precedence constraint prevents conflicting plans

2.3 Complexity of Designing Plan Coordination Mechanisms: Some Results

The results of this approach to plan coordination for self-interested agents can be summarised as follows:

1. **Verifying whether a plan-coordination mechanism is necessary.** It is intractable (coNP-complete) to decide whether or not a redesign of the

planning task is necessary to ensure feasibility of the joint plan (i.e., the problem to verify whether it is needed to design a plan-coordination mechanism is intractable)[11]. This holds already for instances with a few (at least 4) tasks per agent. If, however, the number of agents is fixed, the problem is polynomially decidable [12].

2. **Designing an arbitrary coordination mechanism.** It is always possible to find a (trivial) plan-coordination mechanism in polynomial time that guarantees the existence of a joint conflict-free plan. So the problem of finding an arbitrary plan-coordination problem is in P. The coordination mechanism simply specifies some additional constraints to the complex task to be solved in such a way that for every agent the set of tasks it has to complete is totally ordered [11].
3. **Designing a minimal coordination mechanism.** The construction of a minimal plan-coordination mechanism (minimally redesigning the original complex task specifications) is a highly complex task in itself. Even for instances where the agents are assigned 2 tasks, it is already an NP-complete problem, while in general, the problem of finding a minimal coordination mechanism is Σ_2^p-hard, even if the agents have a modest number of tasks (7 or more) to complete [12].

3 A Polynomial Algorithm to Achieve Plan Coordination

Since the problem of finding a minimal coordination set Δ is too complex to solve in reasonable time (unless P = NP), and the trivial solution to the coordination problem generates an inflexible solution, in this section, we investigate an algorithm to produce a coordination set that is more flexible than the trivial one, but not necessarily an optimal one. First, we show that the algorithm indeed may perform very badly on some instances of the coordination problem. Then we present a class of instances where the algorithm produces near-optimal results.

The algorithm, called the *Depth-Partitioning Algorithm*, is based on the simple construction described in Algorithm 1.

This algorithm is capable of making any complex task plan coordinated.

Example 6. Let us consider the application of the algorithm to the complex task $\mathcal{T}$ discussed in Example 1 and presented in Figure 1. There are six tasks $t_1, \ldots, t_6$ involved. In Figure 4, the depths of the tasks are indicated. Since the tasks belonging to agent A_1 are t_1, t_5, t_6, their different depths induce the set $\Delta_1 = \{t_1 \prec_1 t_5,\ t_5 \prec_1 t_6\}$ of constraints and as a result, the (new) precedence tuple $t_1 \prec_1 t_5$ is added to $\prec_1$ (indicated by the dashed arrow).

Likewise, since t_2, t_3, t_4 belong to agent A_2 and the depth of t_3 is smaller than the depth of t_4 and t_2, $\Delta_2 = \{t_3 \prec_2 t_2,\ t_3 \prec_2 t_4\}$ and the (new) tuple $t_3 \prec_2 t_2$ is added to $\prec_2$, again indicated by the dashed arrow. As can easily be seen, the instance is plan coordinated, since traversing the arrows, the depth of the tasks can never decrease and if a local plan constraint $t \prec_i t'$ is added, it can only be

Algorithm 1. Depth-Partitioning Algorithm.

1. Take the partially-ordered complex task $\mathcal{T} = (T, \prec)$ and consider the subsets $T^d = \{t \in T \mid \mathit{depth}(t) = d\}$ of tasks having the same depth in $\mathcal{T}$. Here, the depth $\mathit{depth}(t)$ of a task t is defined as follows: If t does not have predecessors in $\prec$ then $\mathit{depth}(t) = 0$, else $\mathit{depth}(t) = 1 + \max\{\mathit{depth}(t') \mid t' \prec t\}$. The depth $\mathit{depth}(T)$ of the set of tasks T is the maximum value of $\mathit{depth}(t)$ for a $t \in T$. Note that $\{T^d\}_{d=0}^{\mathit{depth}(T)}$ is a partitioning of T.
2. Consider the task partitioning $\{T_i\}_{i=1}^n$ of T induced by the task allocation to the agents $\{A_i\}_{i=1}^n$ and let T_i^d denote the set of tasks $t \in T_i$ such that $\mathit{depth}(t) = d$.
3. For every agent A_i, let $(T_i^{d_1}, T_i^{d_2}, \ldots, T_i^{d_k})$ be the sequence of all (non-empty) sets T_i^d sorted in increasing values of the depth value d_j.
4. For every agent A_i, let $\Delta_i = \{t \prec_i t' : (t, t') \in T_i^{d_j} \times T_i^{d_{j+1}},\ j = 1, 2, \ldots, k-1\}$ (i.e., all tasks of an agent A_i occurring at a lower depth are required to precede all tasks occurring at a higher depth).
5. Output $\Delta = \bigcup_{i=1}^n \Delta_i$.

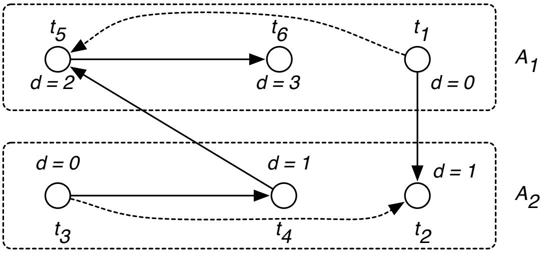


Fig. 4. Applying the Depth Partitioning Algorithm to the task discussed in Example 1. The dashed arrows indicate the precedence constraints added by the algorithm.

added if $\mathit{depth}(t) < \mathit{depth}(t')$. Hence, the task instance remains partially ordered whatever locally feasible plan constraints are added by the agents.

Proposition 1. *Let $\mathcal{T} = (T, \prec)$ be a complex task and Δ the set of additional precedence constraints induced by the depth-partitioning algorithm given above. Then the complex task $\mathcal{T}' = (T, \prec \cup \Delta)$ is plan coordinated.*

Proof. See Appendix A.

The quality of the coordination mechanism produced by an algorithm can be expressed as the number of constraints added by the algorithm versus the number of constraints added by a *minimal* coordination mechanism. We will call this ratio, a number greater than or equal to 1, the *performance ratio* of the algorithm. The closer this ratio is to 1, the better the algorithm is. It is not difficult to see that our simple algorithm can produce arbitrary bad coordination sets, which is illustrated by the following example.

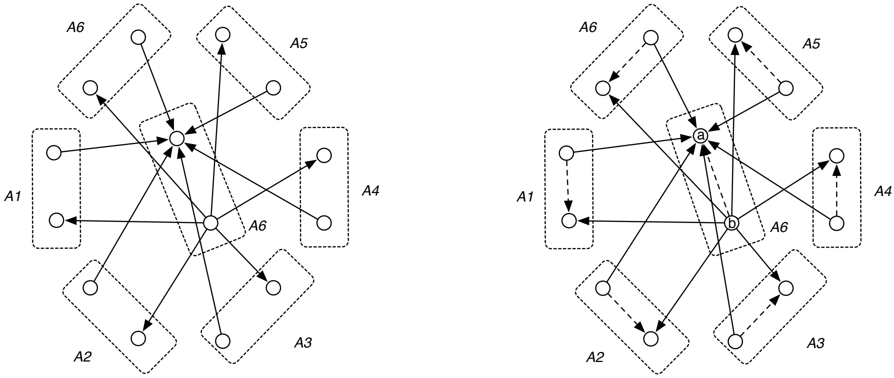


Fig. 5. A complex task (a) where the depth-partitioning algorithm performs badly, adding 7 constraints (indicated by the dashed arcs, see (b)) where only one (from task b to task a) is needed

Example 7. Consider the following complex task with $n+1$ agents, each having 2 tasks. Each agent A_i, for $i = 1, 2, \ldots, n$, has one task that precedes the task a belonging to agent A_{n+1}, while agent A_{n+1} also has a second task b that has to precede all (second) tasks of the other agents. In Figure 5, such a complex task is depicted for $n = 6$.

If we add one constraint between task b and task a for agent A_{n+1} then the instance is plan coordinated: Every inter-agent cycle is prevented by adding this constraint. On the other hand, every task t in this partially-ordered set of tasks has either depth 0 or depth 1. Moreover, every agent has exactly one task of depth 0 and one task of depth 1. Therefore, the algorithm will produce $n+1$ additional constraint arcs, for every agent one constraint. This shows that the performance ratio of this algorithm is $\frac{n+1}{1}$ and, hence, not bounded by a constant.

3.1 Task Chains

Although the algorithm produces coordination sets that are far from optimal in general, special cases exist where the algorithm produces (nearly) optimal results. One such a class is the class of complex tasks where the partial order defined on T generates a series of parallel chains each of depth d for some constant d. An example of such a set of chains and a partitioning of the tasks is given in Figure 6. Within this class we distinguish *left-right* and *right-left* chains and a partitioning such that an agent A_i, with $i = 0, 1, \ldots, d$, has tasks of depth i in left-right chains and tasks of depth $d - i$ of right-left chains (see Figure 6). Let us call a set of such chains of depth d a d-instance. If such a d-instance contains k left-right and m right-left chains, we will call such a set a (d, k, m)-instance. Note that the total number of tasks in such a set is $|T| = (d+1) \times (k+m)$.

It is not difficult to see that the minimum number of additional constraints to make a $(1, k, m)$ set coordinated is $k \times m$: In order to prevent cycles, we have to

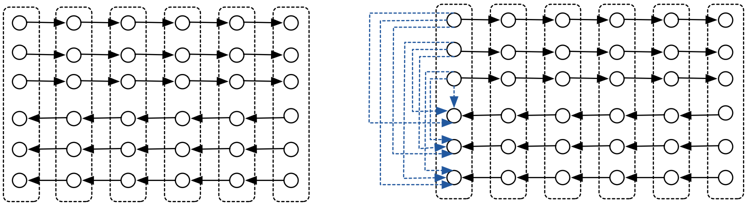


Fig. 6. A set of parallel chains (a $(5,3,3)$-set) as a complex task (a). In (b), the set Δ_1 of additional constraints for agent A_1 is shown.

add an arc between the beginning of a left-right and the end of every right-left chain or vice-versa. Proceeding inductively, suppose that we add 1 to the length of every chain in a given (d,k,m)-instance with $d+1$ agents and we create an additional agent to take care of the new tasks. Furthermore, suppose that the original (d,k,m)-instance was already coordinated. Then it is not difficult to see that again $k \times m$ coordination arcs have to be added in order to make this instance plan coordinated. This implies the following result.

Observation 1. *The minimum size of a coordination set to make a (d,k,m)-instance plan coordinated is $d \times k \times m$.*

We will show that the Depth-Partitioning Algorithm (Algorithm 1) produces minimal coordination sets for some (d,k,m)-sets. Note that the set of tasks of every agent (except the middle agent if d is even)[3] consists of exactly two subsets of tasks of *different* depth. If d is odd, the algorithm will, therefore, return a set of $(d+1) \times k \times m$ additional constraints. If d is even, there is exactly one agent whose tasks all have the same depth. As a result, the algorithm will return a set of $d \times k \times m$ additional arcs. Hence, we have the following result.

Proposition 2. *The depth-partitioning algorithm returns an optimal coordination set for every (d,k,m)-set, $d \geq 1$, where d is an even number. If $d \geq 1$ is odd, the performance ratio of the algorithm is $\frac{d+1}{d} \leq 2$.*

In particular, these performance measures do not change if, instead of one agent A_i for every set of tasks at level i in left-right chains and depth $d-i$ in right-left chains, we have more than one agent for such a sets of tasks. One such example is the *logistic planning* problem. Before we discuss this case, we will introduce the price of autonomy to qualify coordination mechanisms in a more detailed way.

4 The Price of Autonomy and an Application to Logistic Planning

A coordination mechanism like the one we discussed in the previous section guarantees that a set of agents can plan independently. It realises this guarantee

[3] Remember that d starts from 0. If d is even, the length of a chain and the number of agents is odd.

by imposing additional restrictions on the complex task to be completed. These additional restrictions can be seen as one aspect of the *price of autonomy*, since they restrict the planning freedom of the participating agents. One way to reduce the price of autonomy in this respect is to look for a *minimum* number of additional constraints. Taking into account the *loss of freedom*, however, is only one aspect of the price of autonomy.

Another, as important, behavioural aspect of autonomous planning is the *loss of performance* it might incur: Due to autonomous planning, the joint plan that is composed from the independently developed individual plans might have a significantly higher cost than an optimal plan for the complex task when assuming non-autonomous planners. So assuming an optimal (i.e., minimum) coordination mechanism that can be found efficiently, we could define the price of autonomy ρ w.r.t. the *performance* of the planning system as the ratio

$$\rho = \frac{\text{cost of joint plan composed of individually optimal plans}}{\text{cost of optimal joint plan}} \tag{1}$$

Here, the cost of a joint plan refers to the cost of the *concrete* joint plan of all agents together (see Remark 1). This implies that in order to establish the price of autonomy we have to investigate a concrete planning domain, where the set of plan steps used to compose plans is known and the cost of plans based on ordering these plan steps can be determined. Here we have chosen a simple *logistic planning* domain to investigate the price of autonomy.

4.1 Logistic Planning Problems

The logistic planning problem we have in mind consists of a triple (L, C, O) where L is a set of $m \times n$ locations $l_{i,j}$ and C is a set of n cities c_i. Each city c_i is a subset $c_i = \{l_{i,j} \mid j = 1, 2, \ldots, m\}$ of m locations in L. $O \subseteq (L \times L)$ is a set of orders $o = (l, l')$, indicating that a certain package has to be transported from (pickup) location l to (delivery) location l'. All locations belonging to a city c_i are interconnected via direct links. In each city c_i, we distinguish a special location $l_{i,1}$ as the *airport* of city c_i. All airports are assumed to be interconnected via direct flights. See Figure 7 for a simple illustration.

We assume a truck to be available in each city to carry packages from one location in the city to another. There is also a plane available carrying packages from one airport to another. Both trucks and planes can carry any amount of packages. We distinguish *intra-city* orders and *inter-city* orders. An intra-city order requires a *load action* at the pickup location, a *move action*, and an *unload action* at the destination location in the same city. So the cost of an intra-city order is minimally 3 actions. For an inter-city order, we distinguish a *pre-order* phase, a *plane-phase*, and a *post-order* phase. In the pre-order phase, an intra-city order is carried out by transporting the package to the airport of the pickup city. In the plane-phase, the package is transported to the destination airport. In the post-order phase, the package is transported from the airport to its final destination. So an inter-city transportation might require at least 6 load/unload

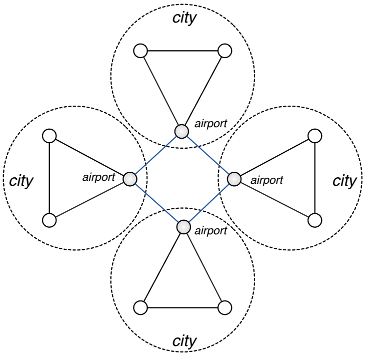


Fig. 7. A logistics example with 4 cities and 3 locations per city. No orders are specified.

actions and at least 3 move actions. We use a simple uniform cost model where every load, unload and move action has unit cost. Hence, the cost of a plan simply equals the number of actions it contains.

Given an instance (L, C, O) of this logistic planning problem we are looking for a plan P that carries out all orders in O. Such a plan is a sequence of load/unload and move actions completing all the orders in O. The cost of P, denoted by $c(P)$, is the sum of the cost of all (move, load and unload) actions occurring in P and of course, we would like to obtain an optimal plan P^*, i.e., a plan with minimum cost.[4]

4.2 Coordinating the Logistic Planning Problem

Note that an instance (L, C, O) of the logistic planning problem can be easily translated into a complex task $(T, \prec)$: Every inter-city order o_j consists of a linearly-ordered sequence of three elementary tasks $t_{j1} \prec t_{j2} \prec t_{j3}$, where t_{j1} is a truck task of transporting the package from its pickup location to the airport, t_{j2} is the plane task of transporting the package to the airport of the destination city, and finally t_{j3} is a truck task consisting in transporting the package to its destination location. Note that the task assignment is trivial here, since the pre- and post-order phase of an order $o \in O$ are assigned to the truck agent in the city where the pick-up and delivery location belong to, respectively, and the plane-phase orders are assigned to the plane agent. Note that the set O of orders induces a 2-instance (an instance containing chains of length 3) of size $|O|$ in the complex task $(T, \prec)$.

It is easy to see that a feasible joint solution is not guaranteed if the truck and plane agents plan independently from each other. Applying the Depth-partitioning algorithm on this 2-set, it is easy to see that the set of orders O_i of every truck agent T_i is partitioned into two sets O_i^0 and O_i^2: The first set is the set of all pre-order transportations tasks (of depth $d = 0$ to the airport of the city, and O_i^2 is the set of all post-order transportation tasks of depth $d = 2$.

[4] Note that we are not looking for a minimum time plan.

The Depth-partitioning algorithm forces the addition[5] of the constraints $t \prec t'$ for every $t \in O_i^0$ and $t' \in O_i^2$. Since the complex task induced by the logistic planning problem is a $(2, k, m)$-instance, from the discussion in Section 3.1, it can be easily seen that this set of additional constraints is a minimum set.

The Depth-partitioning algorithm now guarantees that every plan developed by any of the truck agents and every plan developed by the plane agent can be joined together to constitute a feasible plan for the complex task. Moreover, for this particular logistic problem, it guarantees that the coordination mechanism imposes a minimum amount of additional constraints.

We would like to determine the price of autonomy ρ, that is the cost incurred by the coordination mechanism. This cost measure is determined by the sum of the costs of the individually optimal plans developed by the agents versus the cost of the globally optimal transportation plan. Apparently, the cost of the optimal plan seems to be fixed: Every transportation order requires 6 load and unload actions and 3 move actions, so the cost of n transportation orders would be $\geq 9n$ using a uniform cost model. This line of reasoning, however, does not recognise that we can often build cheaper plans by carefully combining pickup and delivery orders. For example, if a truck has to bring m packages from city locations to the airport, a worst-case plan would be first to drive to the first pickup location then to bring each package to the airport separately, incurring a cost of 1 pickup + 1 move + 1 delivery = 3 actions per order + m move actions to go to the pickup locations. This will result in a plan cost of $3m + m = 4m$. A better plan, however, would be first to drive to all pickup locations, picking up all packages ($2m$ actions) then to drive to the airport (1 move action) and finally to unload the truck (m actions). This requires a plan with cost at most $3m+1$. Likewise, the cost of an optimal plane plan depends on finding a smallest number of move actions required to satisfy all transportation orders.

4.3 Visiting Sequences

To determine the price of autonomy, we have to determine the sum $c(P^*_{loc})$ of the costs of the independently constructed (locally) optimal plans and the cost $c(P^*)$ of the optimal plan. From the discussion above, we conclude that to find these optimal plans, we have to find the minimal number of move actions in each of these plans. This number of move actions required can be determined by introducing the notion of a *visiting sequence.*

Let $L = \{l_1, l_2, \ldots, l_n\}$ be a set of locations in a given city (or a set of airports) and let $O \subseteq (L \times L)$ be a set of pickup-delivery orders over L. We assume that all locations are directly connected and the cost to travel from c_i to c_j is the same (one move action) for all $i \neq j$.

A (move) plan for carrying out the set of orders O can be represented by a sequence S of locations over L such that all pickup-delivery orders can be carried out by visiting the locations in the order indicated by S. Such a *visiting sequence* S is a sequence over L with possible repetitions. An order $(l, l') \in O$

[5] The plane tasks, however, all belong to one partition.

is fulfilled by a sequence S if there exists sub-sequences α, β, γ over L such that $S = \alpha l \beta l' \gamma$ (i.e., S contains an occurrence of l before an occurrence of l'). A visiting sequence S over L is a *solution* to the instance (L, C, O) if all orders in O are fulfilled by S.

The problem to find an arbitrary visiting sequence solution S for a set of orders O is an easy problem: Let S' be an arbitrary sequence where every city in L occurs exactly once and consider the concatenation $S = S' \circ S'$. Since for every $(l, l') \in O$ an occurrence of l in the first subsequence and an occurrence of l' in the second subsequence can be selected, clearly, S is a solution. However, to find an *optimal* solution S^* (i.e., a visiting sequence of minimum length) is intractable. This *minimal visiting sequence* problem is an NP-hard problem since the NP-hard *Feedback Vertex Set* (FVS) [13] is polynomially reducible to it.

Without loss of generality, we may assume that for every visiting sequence solution S for O it holds that no location l mentioned in O occurs more than twice in S. Otherwise, just keep the first and the last occurrence of l in S and we still have a solution. Therefore, for every visiting sequence solution S, it should hold that $|S| \leq 2 \times |S^*|$, since every location l in O should appear at least once in S^*.

4.4 Determining the Price of Autonomy

Note that the length of a visiting sequence S can be used to determine the cost of a plan P for a (truck or plane) planning problem: If S is the visiting sequence solution for the set of orders, the cost $c(P)$ of the corresponding transportation plan P equals $c(P) = 2 \cdot |O| + |S|$ (i.e., when taking into account one additional load and one unload action per action, each of unit cost).

We can now determine the price of autonomy for the logistic planning problem as follows: Let p be the number of orders, n the number of cities and $m + 1 \geq 2$ the number of locations per city.[6] Without loss of generality, we may assume that each location in each city is mentioned at least once in the set of orders. Therefore, $p \geq 0.5(n \times m)$, because every order specifies two different locations (not equal to the airport).

Let us first consider the cost $c(P^*_{loc})$ of the combination of locally optimal plans. Each order requires 6 load and unload actions, so we need $6p$ load and unload actions in total. Suppose that in city c_i we have m_{i1} pickup locations and m_{i2} delivery locations. Clearly, we should have $m \leq m_{i1} + m_{i2} \leq 2m$. To transport packages to the airport we need at most $m_{i1} + 1$ move actions per city and to transport them from the airport we need m_{i2} move actions per city. This means at least $m + 1 \leq m_{i1} + m_{i2} + 1 \leq 2m + 1$ move actions per city c_i. To transport the packages by plane, we need a minimal visiting sequence S^*_n for pickup and delivery at the n airports. Hence, the total cost of the optimal joint plan P^*_{loc} allowing independent planning equals

$$c(P^*_{loc}) = 6p + \sum_{i=1}^{n} (m_{i1} + m_{i2} + 1) + |S^*_n|. \tag{2}$$

[6] We assume that all orders create a transportation chain of length 3, which implies that every order requires transportation to and from an airport.

Considering the cost of a globally optimal plan P^*, we note that in every city transporting packages to the airport might be combined with the delivery of packages arrived by plane to their final destinations. Since every location in a city is mentioned at least once in some order (either as destination location or as source location) this number of (combined) move actions per city is $m+1$, so for all cities in total $(m+1) \times n$. The number of plane moves is $|S'^*_n|$, where the optimal plane plan has to satisfy additional constraints allowing for efficient pre- and post transportation. Thus, the total cost for the minimal plan $|P^*|$ is

$$c(P^*) = 6p + (m+1)n + |S'^*_n|, \tag{3}$$

where $|S'^*_n| \geq |S^*_n|$ since S^*_n is a globally optimal move plan for the plane. Hence,

$$\rho = \frac{c(P^*_{loc})}{c(P^*)} = \frac{6p + \sum_{i=1}^{n}(m_{i1} + m_{i2} + 1) + |S^*_n|}{6p + (m+1)n + |S'^*_n|}. \tag{4}$$

This ratio is maximal if $m_{i1} = m_{i2} = m$ for $i = 1, 2, \ldots, n$, implying that $p \geq m \times n$. Using the constraint $|S'^*_n| \geq |S^*_n| \geq n$, we derive

$$\rho = \frac{c(P^*_{loc})}{c(P^*)} \leq \frac{6mn + 2mn + 2n}{6mn + mn + 2n} = \frac{8m+2}{7m+2} \leq \frac{8}{7} \approx 1.14\ . \tag{5}$$

Note that this price of autonomy is based on the assumption that the plane plan to be developed by the plane agent is optimal (i.e., it costs an optimal number of $|S^*_n|$ move actions). As we have remarked above, determining this optimal visiting sequence is an intractable problem. If we don't require this plane plan to be optimal, the total number of move actions can be at most twice the number of move actions in an optimal plan. Then, we can determine the effective price of autonomy ρ_{eff} as

$$\rho_{\mathit{eff}} = \frac{c(P_{\mathit{eff}})}{c(P^*)} \leq \frac{6p + 2mn + n + 2 \times |S^*_n|}{6p + mn + n + |S^*_n|} \leq \frac{8mn + 3n}{7mn + 2n} \leq \frac{11}{9} \approx 1.22\ . \tag{6}$$

Remark 2. To place this result into perspective, we should mention that, unless there is some major breakthrough in complexity theory,[7] we cannot hope for an ϵ-approximation algorithm that solves the minimum visiting sequence problem with $\epsilon < 2$. This implies that the best performance ratio of an approximation algorithm for the logistic planning problem with $p = mn$ and $m = 1$ we can hope for is

$$\alpha = \frac{6mn + (m+1)n + 2 \times |S^*_n|}{6mn + (m+1)n + |S^*_n|} = \frac{7mn + 3n}{7mn + 2n} = \frac{7+3}{7+2} \geq \frac{10}{9} \approx 1.11 \tag{7}$$

If we compare this result with Equation 5 and $m = 1$ this directly implies that the best polynomial approximation algorithm would introduce almost the same overhead (worstcase) as locally optimal algorithms for solving the logistic problem with our coordination mechanism do.

[7] The breakthrough would be finding a constant-approximation algorithm for the Minimum Directed Feedback Vertex Set-problem.

5 Experimental Results

The results obtained in the previous sections are theoretical and worst-case results. We are guaranteed that the price of autonomy is never worse than about 1.22 implying that we do not increase the cost of our joint plan too much, compared to the cost of an optimal plan. Now, we will compare the performance of our coordination approach with planners that aim to solve the logistic problem centrally.

With this comparison we want to show two things: First, the average performance of our coordination approach is much better than what should be expected, if the worst-case performance ratio is taken as the norm. Second, the coordination approach can be used to enhance the planning power of existing planners significantly, thereby showing that it enables single-agent planning technology to be used in multi-agent problems.

In the Artificial Intelligence Planning and Scheduling (AIPS) competition of the year 2000, several general-purpose planning systems competed in a number of planning domains. The logistic planning problem as described in Section 4.1 was one of the domains featured. We have used the AIPS logistics dataset in our experiments because of its status as a benchmark problem set, and also because it allows us to compare our decomposition-by-coordination approach to a selection of centralised planning systems.

In Table 1, we compare plan cost (in terms of the number of moves in a plan) for four planners. In the second column, the costs of the optimal plans are given as calculated by encoding the complete instance as an ILP-problem and solving it exactly (of course not taking into account the time needed to find a solution). The third column represents the cost of the plans produced using the coordination approach. The fourth, fifth, and sixth columns represent a selection of the planning systems competing in the AIPS: The competition-winning TALplanner [14], and the above-average performers STAN [15] and HSP2 [16]. Each row in Table 1 represents an instance in the dataset, characterised by the number

Table 1. Results for 12 randomly chosen instances from the AIPS logistics dataset. For each instance the minimum number of moves is determined and for each planner the number of moves produced is given.

nr packages	min nr moves	Coordination	TALplanner	STAN	HSP2
20	107	113	111	110	145
25	143	150	152	149	206
30	175	182	183	177	250
35	177	181	186	182	264
40	228	239	239	232	337
45	269	284	285	276	–
50	286	299	306	293	–
55	319	327	338	326	–
60	369	391	398	376	–
65	371	387	397	382	–
70	405	426	437	416	–
75	438	458	471	448	–

of packages that have to be transported for that instance. Of the roughly 200 instances in the dataset, we have made a random selection of 12.

It will come as no surprise that the results produced using the coordination approach, especially since the local plans were in fact solved exactly, deviate little from the optimal plans (less than 5% on average). This is significantly better than the expected (25%) based on the worst-case performance ratio.

The plans produced by the coordination approach are comparable in quality with the plans produced by STAN (only 2% deviating from the optimum) and TALplanner (about 7%). To illustrate that for some solvers the problems in the logistics dataset are far from trivial, HSP2 does not manage to solve (within reasonable time and memory constraints) any instances where more than 40 packages have to be transported, and produces significantly worse plans (about 44% deviating from the optimum). The CPU-times needed to produce the plans by the coordination approach were a few seconds for each of the planning instances occurring in Table 1.

As we remarked before, the coordination approach cannot only be used to solve multi-agent planning problems using simpler single-agent planning tools. We can also apply it as a *pre-processing* step for a given planning system that has trouble solving such multi-agent planning problems. The idea is then to decompose the problem into smaller sub-problems that can be solved independently by the planning system. Thereafter, the solutions to the sub-problems can simply be combined into a solution to the whole problem instance.

Specifically, we propose to use the coordination approach to decompose a multi-agent logistics instance into a set of single-agent planning problems. Then, feed each of the single-agent planning sub-problems to the planning system, and combine the results (plans) according to the protocol into an overall plan, thereby solving the complete multi-agent instance. What we would like to see using this method is significant savings in computation time without significant loss in plan quality compared to the use of the planner solving the complete instance. For this experiment, we chose STAN (since it produced the best plans for the complete instance) and HSP2 (since it consumed a lot of CPU time).

To test these expectations, we randomly selected 51 problems from the logistics planning dataset and observed both the reduction in CPU-time and the reduction in plan cost, comparing a solution produced by using only the planner with a solution by using the planner in combination with the coordination approach. The results are given in Figure 8.

It can be o0wbserved that both STAN and HSP2 definitely benefit from preprocessing by the coordination approach: Both planning systems regularly achieve savings in computation time of over 80%. In addition, we can see that HSP2 produces plans that are on average 20% cheaper (i.e., requiring 20% less actions). Also note that the plan cost for STAN does not increase significantly when using the coordination approach. Finally, it can be observed that even after a decomposition into smaller sub-problems, for quite some instances HSP2 again was not able to produce a solution within reasonable time. This means that even for the local planning problems HSP2 still has considerable difficulty in solving them.

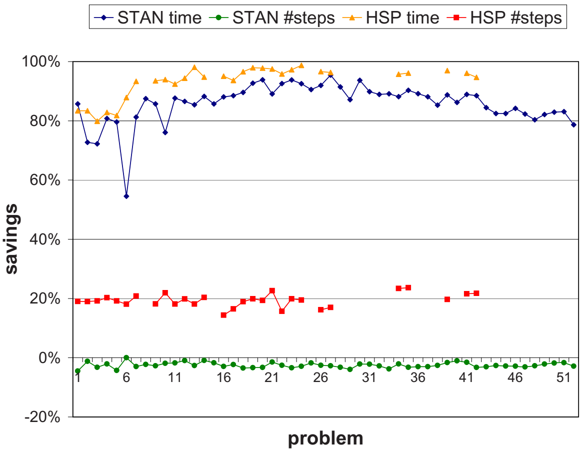


Fig. 8. Savings in CPU times and plan cost (#steps) when STAN and HSP2 make use of the coordination approach as a pre-processing step

6 Discussion and Future Work

We discussed the plan-coordination problem for selfish agents. We showed that, although optimal plan-coordination mechanisms are hard to obtain in general, in some special cases a polynomial algorithm can be used to find minimal coordination sets. We also discussed the price of autonomy as a means to determine the performance loss one can expect when allowing a planning problem to be solved by autonomous agents.

In more recent work [17], we have extended this approach to tightly-coupled tasks with dependency and synchronisation constraints. There, we show that plan-coordination mechanisms exist for such complex tasks, but that they require information exchange after planning to establish the exact time of scheduling synchronised tasks. This offers possibilities to extend the plan-coordination approach to the domain of temporal planning. We also showed that we can also provide coordination mechanisms for *durative tasks* with time constraints. Moreover, the techniques developed enable us to reduce the construction (and complexity) from previously-developed coordination mechanisms to the construction of coordination mechanisms for durative tasks with time constraints, implying that the latter are at least as hard to design as the former mechanisms.

A final extension we study are coordination mechanisms for temporal tasks where not only the feasibility of the total plan has to be guaranteed, but also completion time of the total task has to be minimised [18]. We show that, while the problem can be easily solved if the agents are able to execute an unbounded number of tasks concurrently, the problem to find suitable coordination algorithms in case the agents are autonomous and have bounded concurrency is difficult. In this latter context, we can determine the price of autonomy with respect to a coordination mechanism by taking the ratio of the completion time achieved by a coordination mechanism and the completion time achieved by a dictatorial optimal algorithm. Analogously, we can establish the price of coordination by taking into account the number of additional constraints enforced by the coordination mechanism. Taking both measures into account in determining the quality of coordination mechanisms enforces us to look at Pareto-efficient coordination mechanisms. In this way, we can choose the best alternative, given a tolerated performance loss and a desired level of autonomy.

References

1. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artificial Intelligence 101, 165–200 (1998)
2. Zlot, R.M., Stentz, A.: Market-based multirobot coordination for complex tasks. In: International Journal of Robotics Research, Special Issue on the 4th Int. Conf. on Field and Service Robotics, vol. 25, pp. 73–101 (2006)
3. Christodoulou, G., Koutsoupias, E., Nanavati, A.: Coordination mechanisms. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 345–357. Springer, Heidelberg (2004)
4. Smith, S.F., Gallagher, A., Zimmerman, T., Barbulescu, L., Rubinstein, Z.: Distributed management of flexible times schedules. In: Proc. of the 6th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 472–479 (2007)
5. Decker, K.S., Lesser, V.R.: Designing a family of coordination algorithms. In: Proc. of the 1st Int. Conf. on Multi-Agent Systems (ICMAS), San Francisco, CA, USA. AAAI Press, MIT Press (1995)
6. Cox, J.S., Durfee, E.H.: Efficient mechanisms for multiagent plan merging. In: Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS), Washington, DC, USA, aug 2004, vol. 3, pp. 1342–1343. IEEE Computer Society, Los Alamitos (2004)
7. Sandhlom, T.W., Lesser, V.R.: Coalitions among computationally bounded agents. Artificial Intelligence 94, 99–137 (1997)
8. Walsh, W.E., Wellman, M.P.: A market protocol for decentralized task allocation and scheduling with hierarchical dependencies. In: Proc. of the 3rd Int. Conf. on Multi-Agent Systems (ICMAS), pp. 325–332. IEEE Computer Society Press, Los Alamitos (1999)
9. Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. Journal of Robotics Research 23, 939–954 (2004)
10. Dias, M.B., Stentz, A.: Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination. Technical Report CMU-RI-TR-03-19, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA (2003)
11. Buzing, P.C., ter Mors, A.W., Valk, J.M., Witteveen, C.: Coordinating self-interested planning agents. Autonomous Agents and Multi-Agent Systems 12, 199–218 (2006)
12. Steenhuisen, J.R., Witteveen, C., ter Mors, A.W., Valk, J.M.: Framework and complexity results for coordinating non-cooperative planning agents. In: Fischer, K., Timm, I.J., André, E., Zhong, N. (eds.) MATES 2006. LNCS (LNAI), vol. 4196, pp. 98–109. Springer, Heidelberg (2006)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, San Fransisco (1979)

14. Kvarnström, J., Doherty, P.: TALplanner: A temporal logic based forward chaining planner. Annals of Mathematics and Artificial Intelligence 30, 119–169 (2000)
15. Long, D., Fox, M.: Efficient implementation of the plan graph in STAN. Journal of Artificial Intelligence Research 10, 87–115 (1999)
16. Bonet, B., Geffner, H.: Heuristic search planner 2.0. AI Magazine 22, 77–80 (2001)
17. Steenhuisen, J.R., Witteveen, C.: Plan coordination for durative tasks. In: Salido, M.A., Garrido, A., Bartak, R. (eds.) Proceedings of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS), pp. 73–80 (2007)
18. Yadati, C., Zhang, Y., Witteveen, C.: Performance of coordination mechanisms (published, 2008)

Appendix

A Proof of Proposition 1

Proof. Suppose, on the contrary, that the resulting complex task $\mathcal{T}'$ is not plan coordinated. Then there must exists some cycle $c = (t_{i_1}, t_{i_2}, \ldots, t_{i_m}, t_{i_1})$ where

1. c is not limited to tasks belonging to one agent, i.e., c contains at least two tasks t_{i_j} and t_{i_k} belonging to different agents.
2. c contains at least two tasks connected via a plan-refinement relation $\prec_i^*$. This means that c has a subsequence of tasks $(t_{i_j}, t_{i_{j+1}}, t_{i_{j+2}}, \ldots, t_{i_{j+k}}, t_{i_{j+k+1}})$, where $(t_{i_j}, t_{i_{j+1}}, t_{i_{j+2}}, \ldots, t_{i_{j+k}})$ is a $\prec^*$-sequence of tasks belonging to some agent A containing at least two different tasks, while the task $t_{i_{j+k+1}}$ belongs to another agent A'.

Since $t_{i_{j+k+1}}$ depends on $t_{i_{j+k}}$, it follows that

$$depth(t_{i_{j+k}}) < depth(t_{i_{j+k+1}}). \tag{8}$$

Due to the construction of the Δ_i-sets, for every subsequence $(t_{i_{j+h}}, t_{i_{j+h+1}})$ of an intra-agent path $(t_{i_j}, t_{i_{j+1}}, t_{i_{j+2}}, \ldots, t_{i_{j+k}})$ of agent A it must hold that

$$depth(t_{i_{j+h}}) \leq depth(t_{i_{j+h+1}}), \tag{9}$$

because $depth(t_{i_{j+h}}) > depth(t_{i_{j+h+1}})$ for two tasks belonging to the same agent would imply that $(t_{i_{j+h+1}}, t_{i_{j+h}}) \in \Delta_i$ and, therefore, $t_{i_{j+h}} \prec^* t_{i_{j+h+1}})$ cannot occur as part of A's plan, because it would create a cycle.

Equation 8 and 9 together imply that

1. traversing from one task t to another task t' via an inter-agent constraint in c strictly increases the depth: $depth(t) < depth(t')$, and
2. traversing from one task t to another task t' via an intra-agent constraint in c does not decrease the depth: $depth(t) \leq depth(t')$.

This implies that since there is at least one inter-agent constraint involved, the depth of the first task t_{i_1} occurring in c should be strictly less than the depth of the last task in c. But that implies $depth(t_{i_1}) < depth(t_{i_1})$: contradiction. Hence, such a cycle c cannot exist and the complex task $\mathcal{T}'$ is plan coordinated. □

Actions with Failures in Interval Temporal Logic

Arjen Hommersom and Peter Lucas

Institute for Computing and Information Sciences
Radboud University Nijmegen
{arjenh,peterl}@cs.ru.nl

Abstract. Failures are unavoidable in many circumstances. For example, an agent may fail at some point to perform a task in a dynamic environment. Robust systems typically have mechanisms to handle such failures. Temporal logic is a widely used representation language for reasoning about the behaviour of systems, although dealing with failures is not part of the language. In this paper, it is investigated how interval temporal logic can be extended with an operator describing failure. This logic has a close relationship to exception handling mechanisms in programming languages, which provides an elegant mechanism for modelling and handling failures. The approach is motivated from the context of specification of systems that have to operate in highly dynamic environments. A case study of the formal modelling and verification of the treatment of diabetes mellitus type 2 illustrates the practical usefulness of the approach.

1 Introduction

For agents that do not have a complete model of their environment or lack certain control over it, it is unavoidable that failures to perform tasks occur. Many systems require some type of robustness against these failures, e.g., robots need to make sure that their task will be accomplished, aviation systems need to make sure that the plane does not crash, etc. In agent literature, the semantics of failures have been investigated in a logical sense [11] and have been incorporated in agent programming languages [5]. Similarly, in software engineering, the use of exceptions as first-class citizens in programming languages is wide-spread.

Besides the internal aspects of a system, i.e., a program state or *mental* state of an agent, an important aspect of systems is *behaviour*, i.e., how it acts and reacts in a dynamic environment. To reason about this behaviour, mechanisms that go beyond the scope of classical predicate logic are employed. Since the late seventies, several temporal logics have been proposed to deal with specification and verification of hardware and software systems. In artificial intelligence, many of the logics dealing with actions usually contain some temporal component. In systems where failures heavily determine the final behaviour, modelling of this behaviour is more natural when failures are part of the modelling language. Moreover, we will argue that, since temporal logical formulas can be used to describe behaviour, the failure of a behaviour is best described using a sentential

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 22–41, 2008.

operator, i.e., as a property of a (temporal) logical sentence. This contrasts with other approaches, where failure is seen as a property of primitive events and corresponds to the major contribution of this paper.

In the next section, we will first consider some motivating examples for reasoning about failure and explain why a simple solution is often unsatisfactory. Then, in Section 3, this is related to exception handling mechanisms that are found in programming languages. In Section 4, preliminaries concerning Interval Temporal Logic (ITL) are introduced, which is subsequently extended with failures in Section 5. In Section 6, we apply this to to a medical guideline that deals with the treatment of diabetes mellitus type 2 and study the formalisation and its properties. In Section 7, related work of modelling failure in AI is discussed. Finally, in Section 8, we discuss the results and future work.

2 Motivating Examples

2.1 Robbing a Bank

A well-known logic to model agents is BDI logic as proposed by Rao & Georgeff [11]. It contains modal operators BEL, GOAL, and INTEND which should be interpreted as the believe, goal and intention of the agent. Moreover, amongst other operators, it contains an operator $failed$ to describe that an event has (just) failed and temporal operators such as $\Box$ (always) and $\Diamond$ (eventually). The introduction of a $failed$ operator is motivated by Rao & Georgeff by the fact that failure may force an agent to replan or revise her plans. They give the following example:

> (...) the consequence of a thief successfully robbing a bank is quite different from a thief failing in his attempt to rob the bank, which is again different from the thief not attempting to rob the bank.

For example, it is possible to model an agent that believes that if he fails to rob the bank, then he will go to jail:

$$\mathsf{BEL}\ \Box(failed(\texttt{rob\_bank}) \rightarrow \Diamond\texttt{locked\_up})$$

which allows the agent to revise its plan after the robbery accordingly. Robbing a bank, however, is not an easy task for any intelligent agent (over 50 percent of the bank robbers are arrested in the US[1]). For example, in case of an armed robbery, it involves threatening the people inside the bank at all times, demanding money and securing a getaway. Failure to accomplish any of the subtasks will result in failure to complete the overall task. Note that failure to demand money will in some sense result in failure to rob the bank; however, it is clear that this does not necessarily lead to the consequence of going to jail. There is a need to model the different 'types' of failure associated with a goal that the agent tries to

[1] http://www.fbi.gov/ucr/cius_02/html/web/specialreport/05-SRbankrobbery.html

accomplish. More importantly, the predicate *failed* describes failure of events; however, ‘robbing a bank’ is not a primitive event here, but rather a complex temporal description of several events to accomplish the overall task. Clearly, as *failed* is a predicate on events, it cannot be used in a temporal formula. The only possibility is to define the event `rob_bank` in terms of more primitive events. The downside of this approach is that there is no mechanism to infer that failure of some of the mandatory sub-tasks will result in failure of the overall task. While it might be possible to specify this, as `rob_bank` is a complex temporal description of events, a description of its failure is, most likely, complex as well. In the next subsection, this is illustrated with some temporal patterns that may occur in medical management.

2.2 Medical Management

Suppose we are modelling an agent, typically a physician, who treats a patient. As almost all drugs may result in side-effects, it is of great importance that the agent does not over-medicate the patient. Therefore, if the disease is not directly life-threatening, management of a disease should start with a non-invasive treatment where one expects as little side-effects as possible. It is not always possible to measure beforehand if the effects of the treatment will be desirable, as this could require a test that is considered to be too invasive or because it is not known which physiological variable should be measured. As a result, a failure to treat the patient may occur, which means that subsequent actions are required.

Medical treatments are performed in sequence or in parallel. Sequential actions are typically done in case an earlier treatment fails or when a certain physiological state should be reached before a subsequent state can be effective. In such a case, failure to perform a treatment will result in a failure of the whole protocol, as it will block the successful administering of subsequent treatments. Parallel treatments occur for example when multiple drugs are prescribed at the same time. If the effects of these drugs are combined, then the combination of drugs will fail if one of the individual actions fails. If failures are not handled appropriately, it may lead to medical mismanagement, e.g., in case drugs become ineffective due to failure of other treatment components, continuing to administer these drugs is considered bad medical practice. As a consequence, failure handling plays an important role in maintaining the quality of medical management.

This idea of an implicit mechanism that “propagates” the failures throughout the management of a disease leads to the idea that such failures could be seen as exceptions that need to be handled appropriately. This idea is pursued in the next section.

3 Exception Handling

The idea of handling failures while performing a task is well-known in the context of programming languages by means of exception handling mechanisms. An exception is a failure of an operation that cannot be resolved by the operation itself

[14]. Exception handling mechanisms provides a way for a program to deal with them. Many programming languages (C++, Ada, Java, etc) now incorporate such extensive exception mechanism in order to facilitate robust applications. Typically, such a mechanism consists of two parts. There is a mechanism to *throw* an exception, which sends a signal that an exception has occurred. Second, *catching* an exception transfers control to the exception handler that defines the response that the program takes when the exception occurs. Looking at it slightly differently, one could say that the program determines the plan that is being executed, while the exception handler is able to revise this plan in case an failure occurs.

For the purpose of this paper, it is useful to summarise the semantics of exception handling mechanisms. A formal semantic model of exceptions in Java based on denotational semantics [1] as well as operational semantics [10] exists. The complete mathematical description of these mechanisms is too extensive to be discussed here, as only a small part of the semantics deals with failures. Instead, we give a more general description of the operational semantics of the exception mechanism. A state, here denoted by σ, consists of the heap, values of the local variables, and optionally an exception. Evaluation rules describe how statements change the state, typically in the form $\sigma_0 \xrightarrow{s} \sigma_1$ which denotes that the execution of statement s starting in state σ_0 can terminate in state σ_1. For exception handling, the state is extended with an exception, i.e., we then deal with assertions $\sigma_0 \xrightarrow{s} \sigma_1^e$ which means that the execution of s in σ_0 can terminate in σ_1 throwing an exception denoted by the superscript e[2]. The operational semantics is then also extended with these assertions, e.g., for sequential composition this yields the following two rules depending on whether or not a failure has occurred in the first statement:

$$\frac{\Gamma \vdash \sigma_0 \xrightarrow{s_1} \sigma_1 \quad \Gamma \vdash \sigma_1 \xrightarrow{s_2} \sigma_2}{\Gamma \vdash \sigma_0 \xrightarrow{s_1;s_2} \sigma_2} \qquad \frac{\Gamma \vdash \sigma_0 \xrightarrow{s_1} \sigma_1^e}{\Gamma \vdash \sigma_0 \xrightarrow{s_1;s_2} \sigma_1^e}$$

where Γ defines the context of the rule. Logically speaking, what we see here is that failures are propagated through the semantics of each programming structure. We will show how to incorporate this idea in terms of temporal logic in the next two sections.

4 Interval Temporal Logic

In this section, we define the necessary preliminaries of interval temporal logic (ITL) [8], which acts as the basis for our approach and can be considered a rich framework for specifying many systems. As it is quite a rich system, it can be considered a rather heavy machinery for solving the problems that were discussed in the previous section. However, it shows that the incorporation of failures in the logic can be done for a wide range of logics, such as the more common linear temporal logic (LTL), a sub-logic of ITL.

[2] Abstracting from the different types of exceptions.

4.1 Syntax

For the purpose of this paper, we consider the propositional part of ITL. The main difference with standard temporal logic is that interval temporal logic deals with intervals rather than time points, which makes it suitable for logic-based modular reasoning involving periods of time. In this logic there are three primary temporal constructs:

- skip: the interval is a unit interval of length 1
- $\varphi; \psi$: chop the interval into two parts, such that φ holds in the first part and ψ holds in the second part
- φ^*: decompose the interval into a (possibly infinite) number of finite intervals in which φ holds.

Given a non-empty set of propositional variables $\mathcal{P}$, the full syntax can then be given in BNF notation as follows:

$$\varphi \longrightarrow p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{skip} \mid \varphi; \varphi \mid \varphi^*$$

with $p \in \mathcal{P}$. Let true be defined as $p \vee \neg p$ and false as $\neg\mathsf{true}$, for some $p \in \mathcal{P}$. Then, the following additional linear temporal operators are defined that are used in the remainder of this paper:

$\circ\varphi$	$\triangleq$	$\mathsf{skip}; \varphi$	in the next state φ
$\bullet\varphi$	$\triangleq$	$\neg \circ \neg\varphi$	if there is a next state, then in the next state φ
last	$\triangleq$	$\bullet\, \mathsf{false}$	this is the last state of the interval
finite	$\triangleq$	$\neg(\mathsf{true}; \mathsf{false})$	the interval is finite
$\Diamond\varphi$	$\triangleq$	$\mathbf{finite}; \varphi$	eventually φ
$\Box\varphi$	$\triangleq$	$\neg\Diamond\neg\varphi$	always φ

Other propositional connectives are defined as usual, i.e., $\varphi \vee \psi \triangleq \neg(\neg\varphi \wedge \neg\psi)$, $\varphi \rightarrow \psi \triangleq \neg\varphi \vee \psi$, and **if** α **then** φ **else** $\psi \triangleq (\alpha \wedge \varphi) \vee (\neg\alpha \wedge \psi)$.

4.2 Semantics

Models of this logic are (possibly infinite) sequences of states, denoted by σ, i.e., $\sigma = \sigma_0, \sigma_1, \ldots$. We write $|\sigma|$ to denote one less than the length of the sequence (as usual in ITL), which is either ∞ if there are infinite number of states and otherwise some natural number n. If $\sigma = \sigma_0, \ldots, \sigma_n, \ldots, \sigma_m, \ldots$, then $\sigma^{[n,m]}$ denotes the subsequence $\sigma_n, \ldots, \sigma_m$ of σ. Let each σ_i be a function of type $\mathcal{P} \rightarrow \{\bot, \top\}$, that denotes whether an atomic proposition is either true ($\top$) or false ($\bot$). The formal semantics is then as follows:

$$\begin{array}{lll}
\sigma \models p & \Leftrightarrow & \sigma_0(p) = \top \\
\sigma \models \neg\varphi & \Leftrightarrow & \sigma \not\models \varphi \\
\sigma \models \varphi \wedge \psi & \Leftrightarrow & \sigma \models \varphi \text{ and } \sigma \models \psi \\
\sigma \models \mathsf{skip} & \Leftrightarrow & |\sigma| = 1 \\
\sigma \models \varphi; \psi & \Leftrightarrow & |\sigma| = \infty \wedge \sigma \models \varphi \text{ or} \\
 & & \quad \text{there exists } n \leq |\sigma| \text{ with } \sigma^{[0,n]} \models \varphi \text{ and } \sigma^{[n,|\sigma|]} \models \psi
\end{array}$$

$$
\begin{aligned}
\sigma \models \varphi^* \Leftrightarrow\ & |\sigma| = 0 \\
& \text{or there exists } 0 = n_0 < n_1 < \ldots < n_m < |\sigma| \\
& \quad \text{with } \sigma^{[n_i, n_{i+1}]} \models \varphi \text{ for all } 0 \leq i < m \\
& \quad \text{and } \sigma^{[n_m, |\sigma|]} \models \varphi \\
& \text{or there exists infinite many } 0 = n_0 < n_1 < \ldots \\
& \quad \text{with } \sigma^{[n_i, n_{i+1}]} \models \varphi \text{ for all } 0 \leq i
\end{aligned}
$$

Sequences of, for example medical, actions can now be modelled quite easily, e.g., action a after b is modelled as $a; b$, provided that a and b may overlap. Repetition of this patterns could be modelled as, for example, $(a; \circ\, b)^*$. If, on the other hand, a must be applied for a longer period of time, this may be described as $\Box a; b$. What will happen when in this latter case $fail(b)$ holds at some point: does this constitute a failure of the complete sequence? Presumably not, as b does not necessarily have to hold by the given semantics. What if failure occurs at the last time that a holds? Then it seems to be the case that the whole sequence has failed. We will formalise this intuition in the next section.

5 Interval Temporal Action Logic with Failure

In this section, we extend the logic of ITL with actions and introduce an operator that denotes failure of the formula. We will refer to this extended logic as ITALF.

5.1 Syntax and Semantics

Let $\mathcal{A}$ be a set of actions, and $\mathcal{P}$ a set of atomic propositions. Models σ we will be working with consists of a (possible infinite) sequence of states $\sigma_0, \ldots$. Each σ_i is defined as $\langle \pi_i, \alpha_i \rangle$, where π_i is a function $\mathcal{P} \to \{\top, \bot\}$ and α_i a function $\mathcal{A} \to \{\mathsf{inactive}, \mathsf{active}, \mathsf{failed}\}$. When discussing a σ', we will write α'_i and π'_i such that $\sigma'_i = \langle \alpha'_i, \pi'_i \rangle$. Let the language be extended with actions and an operator **fail**. All semantics given by the language of ITL remains the same. Entailment of ITL will be denoted as $\models_{\mathrm{ITL}}$ from now on and $\models$ will be understood as entailment for ITALF.

Actions are interpreted as activations, hence, negations of actions are understood as actions that are not active (i.e., inactive or failed). This is formalised as follows:

$$\sigma \models a \Leftrightarrow \alpha_0(a) = \mathsf{active}$$

For the definition of failure, we need to consider models where we abstract from the difference between inactive and activation, but instead only consider the difference between failures and non-failures. In order to accomplish this, we use the following models, that we denote as $\mathsf{failmodel}(\sigma)$:

Definition 1. *For all σ, $\mathsf{failmodel}(\sigma) = \sigma'$ if:*

- $|\sigma| = |\sigma'|$
- *for all i such that $0 \leq i \leq |\sigma|$:*
  - *for all $p \in \mathcal{P}$: $\pi_i(p) = \pi'_i(p)$*

- *for all $a \in \mathcal{A}$: if $\alpha_i(a) = \textsf{\textit{failed}}$ then $\alpha'_i(a) = \textsf{\textit{failed}}$, otherwise $\alpha'_i(a) = \textsf{\textit{active}}$*

So $\mathsf{failmodel}(\sigma)$ describes σ where non-failures (in particular inactive actions) are interpreted as activations. We can then consider failure as a type of negation in the definition of **fail**, as follows:

$$\sigma \models \mathbf{fail}\,\varphi \Leftrightarrow \sigma \not\models \varphi \text{ and } \mathsf{failmodel}(\sigma) \not\models \varphi$$

To understand this definition, consider φ as a formula that implies that certain propositions and actions are true or false at certain moments in time, even though for some formulas, there is a choice to be made on which point in time. For atomic propositions, the definition is clear and is equivalent to the negation as $\mathsf{failmodel}(\sigma)$ does not evaluate propositions differently than σ. For actions, the situation is more complicated. First, if an action a is implied by φ at a certain moment in time, then φ fails if the action fails on that point in time, which is exactly given looking at $\neg a$ on $\mathsf{failmodel}(\sigma)$. This seems sufficient; however, consider the converse, i.e., that φ implies $\neg a$ at a certain point in time. Then, the formula fails if in fact the action is activated at that point, which corresponds to the first part of the definition. Note that by just looking at $\mathsf{failmodel}(\sigma)$, we can only derive that it must be active or inactive, however, a failure not to do an action does not correspond to this idea.

5.2 Logical Characterisation

As already mentioned in the previous section, with respect to atomic propositions p, it follows:

$$\mathbf{fail}\,p \equiv \neg p$$

i.e., failure to accomplish p simply means it is not true. So, the formalisation considers failure as a kind of negation. Typically, in the formalisation of medical management, we are interested in formulas such as:

$$\mathbf{fail}\,(p \rightarrow a)$$

i.e., in situation described by p, the action a must be activated. According to the semantics, this is equivalent to $p \wedge \mathbf{fail}\,a$, i.e., if the implication fails, then in the situation described by p, the action a indeed fails. As argued in the previous subsection, failure not to do an action a, i.e., $\mathbf{fail}\,\neg a$ means that a is

1. $\mathbf{fail}(\varphi) \rightarrow \mathbf{fail}(\varphi \wedge \psi)$
2. $\mathbf{fail}(\varphi \vee \psi) \rightarrow \mathbf{fail}(\varphi) \vee \mathbf{fail}(\psi)$
3. $\mathbf{fail}(\varphi) \rightarrow \mathbf{fail}(\varphi; \psi)$, where φ is objective
4. $\mathbf{fail}(\varphi) \rightarrow \mathbf{fail}(\varphi^*)$, where φ is objective

Fig. 1. Propagation of failures in ITALF, where objective formulas are formulas that do not contain any temporal operators

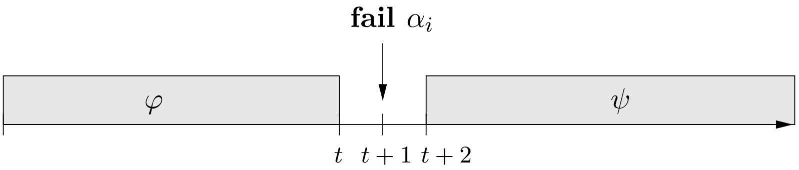


Fig. 2. Sketch of φ $\mathbf{orelse}_{\{\alpha_i\}}$ ψ in case of failure of α_i

in fact done. Conversely $\neg\mathbf{fail}\, a$ means that a is either active or inactive. Hence $\mathbf{fail}\,\neg\varphi \neq \neg\mathbf{fail}\,\varphi$.

In general, the definition of **fail** is such as to propagate to larger formulas. This is summarised in Fig. 1. What is interesting is that the calculus rules of the operational semantics of an imperative programming language described in Section 3 can now be understood in terms of failure inside the logic. For example, for sequential composition, the following calculus rule is sound with respect to the semantics:

$$\frac{\Gamma \vdash \mathbf{fail}(\varphi)}{\Gamma \vdash \mathbf{fail}(\varphi;\psi)}$$

where φ is objective, which follow directly from item (3) of Fig. 1 and modus ponens.

In order to describe acting on the basis of failure, we define an additional operator:

$$\varphi \ \mathbf{orelse}_A \ \psi \triangleq \varphi \wedge \neg\mathbf{last}; \circ\,(\mathbf{failstate}_A \wedge \circ\,\psi)$$

where

$$\mathbf{failstate}_A = \bigvee_{a_i \in A} \mathbf{fail}\ a_i \wedge \bigwedge_{a_i \in \mathcal{A}} \neg a_i$$

i.e., φ holds forever, or, an action fails at some point after which ψ holds. We assume that φ is true in at least a unit interval, which prevents failures to occur right away. In Fig. 2, a model where a failure occurs and is handled is sketched. The definition of the **failstate** ensures that during this time no action can be active, and thus no additional failures may occur. In some sense, this operator may be read as an exception handling mechanism where failures of 'type' A are caught in the execution of φ, such that ψ is executed when this occurs.

Finally, consider the robbing example of Subsection 2.1. The ITALF logic allows one to reason elegantly about, for example the temporal description $\mathbf{fail}(\Box\mathsf{threaten};\mathsf{flee})$. It is cumbersome to derive an equivalent formula in ITL, which has to describe that in case the fleeing fails or that threatening fails before fleeing, the robbing fails. It is not difficult to see however, that it is *possible* to write down such formula. In fact, this is true for arbitrary formulas of the ITALF language and is discussed next.

5.3 Reduction to ITL

In this subsection, we will show how ITALF can be translated to ITL. We thereby give means to exploit the proof techniques that were developed for ITL. To accomplish a reduction to ITL, additional propositional variables are required. We assume we have an infinite number of propositional variables such that we have a (unique) fresh proposition f_a, standing for failure, for each $a \in \mathcal{A}$.

Definition 2. *Given a formula φ, define $\Phi(\varphi)$ as φ where every occurrence of some action $a \in \mathcal{A}$ has been replaced with $\neg f_a$.*

Definition 3. *The reduction of a formula φ is defined on the structure of φ as follows:*

$$\begin{aligned}
\mathit{reduce}(p) &= p \\
\mathit{reduce}(a) &= a \\
\mathit{reduce}(\neg\varphi) &= \neg\mathit{reduce}(\varphi) \\
\mathit{reduce}(\varphi \wedge \psi) &= \mathit{reduce}(\varphi) \wedge \mathit{reduce}(\psi) \\
\mathit{reduce}(\mathit{skip}) &= \mathit{skip} \\
\mathit{reduce}(\varphi; \psi) &= \mathit{reduce}(\varphi); \mathit{reduce}(\psi) \\
\mathit{reduce}(\varphi^*) &= \mathit{reduce}(\varphi)^* \\
\mathit{reduce}(\boldsymbol{\mathit{fail}}\,\varphi) &= \neg\mathit{reduce}(\varphi) \wedge \neg\Phi(\mathit{reduce}(\varphi))
\end{aligned}$$

Below, we refer to $\mathsf{reduct}(\varphi)$ as the ITALF formula that is found by applying the definition exhaustively from left to right. The main result of this subsection which provides the connection between ITL and ITALF follows.

Definition 4. *Given a ITALF formula φ, intended meaning of the failure propositions is defined as follows:*

$$\mathcal{I}(\varphi) = \Box(a_0 \to \neg f_{a_0} \wedge \cdots \wedge a_n \to \neg f_{a_n})$$

where $\{a_0, \ldots, a_n\} \subseteq \mathit{actions}(\varphi)$, such that $\mathit{actions}(\varphi)$ is defined as the set of those $a \in \mathcal{A}$ that is a sub-formula of φ.

Note that $\mathcal{I}(\varphi)$ is finitely bounded by actions in a formula, which is important as the total number of actions in the language may be infinite.

Then, we have the following result:

Theorem 1. $\models \varphi$ *iff* $\mathcal{I}(\varphi) \models_{\mathrm{ITL}} \mathit{reduct}(\varphi)$

The proof of this theorem can be found in Appendix A.

5.4 Robbing the Bank Revisited

As an illustration of the theory introduced above, we revisit the example discussed in Section 2.1. Suppose we would model robbing the bank as follows:

$$\texttt{rob\_bank} \triangleq (\Box\texttt{threaten} \wedge \Diamond\texttt{collect\_money} \wedge \textbf{finite}); \texttt{get\_away}$$

i.e., personnel is threatened for a finite amount of time, money is collected, after which it is necessary to get away. Accepting the semantics of failures as presented in this paper, we can now directly talk about failure of the logical sentence above, possibly in combination with '$\Box\neg\textbf{fail}\,\texttt{collect\_money}$', as this action cannot fail

- Step 1: diet
- Step 2: if Quetelet Index (QI) $\leq$ 24, prescribe a sulfonylurea drug; otherwise, prescribe a biguanide drug
- Step 3: combine a sulfonylurea drug and biguanide (replace one of these by a α-glucosidase inhibitor if side-effects occur)
- Step 4: insulin

Fig. 3. Tiny fragment of a clinical guideline on the management of diabetes mellitus type 2. If one of the steps $k = 1, 2, 3$ is ineffective (fails), the management moves to step $k + 1$.

in the sense discussed in Section 2.1, i.e., failure to collect money does not lead to an arrest.

To illustrate what it then means to fail to rob the bank in the language of ITL, one can take the reduction of the sentence (we will omit the definition of failure propositions defined by $\mathcal{I}$ in the formulas below, i.e., we will only consider faithful models, see Appendix A) resulting in:

$$\cdots \wedge \neg((\Box\neg f_{\mathtt{threaten}} \wedge \Diamond\neg f_{\mathtt{collect\_money}} \wedge \mathbf{finite}); f_{\mathtt{get\_away}})$$

Now supposing that the money is collected, this can be further simplified using the semantics of the ITL operators, finally yielding models for which holds:

$$\forall n \leq |\sigma| \colon (\exists i \colon (0 \leq i \leq n \text{ and } \sigma_i \models f_{\mathtt{threaten}}) \text{ or } \sigma_n \models f_{\mathtt{get\_away}})$$

i.e., either in the first part the threatening fails or, otherwise, the robber fails to get away, which is arguably the intended meaning of failing to rob a bank in this example.

This, however, is not easily specified in future-time temporal logic. While we can obviously express this with the negated chop formula above, negated chop formulas are difficult to interpret. Using a more standard linear temporal operator, **until** (see e.g., [15]), the simplified formula can be rephrased as:

$$\neg((\neg f_{\mathtt{threaten}}) \ \mathbf{until}\ (\neg f_{\mathtt{get\_away}} \wedge \neg f_{\mathtt{threaten}}))$$

which is possibly slightly more understandable. Nevertheless, if the original temporal specification is more complicated than the rather simple example that we provide here, modelling failing behaviour quickly becomes a difficult task.

6 Application to a Medical Guideline

6.1 Introduction

As a more elaborate application of failures, we consider clinical guidelines, which are extensive documents advising clinicians appropriate management of disease.

The guideline shown in Fig. 3 is part of the guideline for general practitioners for the treatment of diabetes mellitus type 2 (DM2) [12], which aims at controlling the level of glucose in the blood. The knowledge in this fragment concerns information about order and time of treatment (e.g., sulfonylurea in step 2), about patients and their environment (e.g., Quetelet index lower than or equal to 27), and finally which drugs are to be administered to the patient (e.g., a sulfonylurea drug).

In order to reason about the quality of guidelines, we require additional medical knowledge, which is based on a methodology for checking quality medical guidelines proposed in [6]. While some of the below formalisation is also discussed in that paper, here, the guideline is formalised in temporal logic, rather than a specialised guideline representation language. In particular, we focus on the issue of failure of treatments.

6.2 Modelling of Medical Knowledge

In order to represent the medical knowledge, a specific language is defined in this section. We restrict ourselves to the knowledge which concerns itself with the primary aim of a guideline, which is to have a certain positive effect on a patient. To establish that this is indeed the case, knowledge concerning the physiology of a patient is required. This is here formalised as a causal model describing effects of the treatment.

We are interested in the prescription of drugs, taking into account their mode of action. Abstracting from the dynamics of their pharmacokinetics, this can be formalised in logic as follows:

$$(d \wedge r) \rightarrow \circ\,(m_1 \wedge \cdots \wedge m_n) \tag{1}$$

where d is the name of a drug, r is a (possibly negative or empty) *requirement* for the drug to take effect, and m_k is a mode of action, such as decrease of release of glucose from the liver, which holds at all future times.

Note that we assume that drugs are applied for an instant, here formalised as 'next'. This is reasonable if we think of the time instants as unspecified periods of time where certain propositions hold. Synergistic effects and interactions amongst drugs can also be formalised along those lines, as required by the guideline under consideration. This can be done either by combining their joint mode of action, by replacing d in the formula above by a conjunction of drugs, or by reasoning about modes of actions. As we do not require this feature for the clinical guideline considered in this chapter, we will not go into details.

The modes of action m_k can be combined, together with an *intention* n (achieving normoglycaemia, i.e., normal blood glucose levels, for example), a particular patient *condition* c, and *requirements* r_j for the modes of action to be effective:

$$(\circ\, m_{i_1} \wedge \cdots \wedge \circ\, m_{i_m} \wedge r_1 \wedge \cdots \wedge r_p \wedge c) \rightarrow \circ\, n \tag{2}$$

For example, if the mode describes that there is a stimulus to secrete more insulin and the requirement that sufficient capacity to provide this insulin is fulfilled, then the amount of glucose in the blood will decrease.

(1) $\mathit{insulin} \rightarrow$
$\circ\, (\mathit{uptake}(\mathit{liver}, \mathit{glucose}) = \mathit{up} \wedge \mathit{uptake}(\mathit{peripheral\text{-}tissues}, \mathit{glucose}) = \mathit{up})$
(2) $\mathit{uptake}(\mathit{liver}, \mathit{glucose}) = \mathit{up} \rightarrow \mathit{release}(\mathit{liver}, \mathit{glucose}) = \mathit{down}$
(3) $\mathrm{SU} \wedge \neg \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{exhausted}) \rightarrow \circ\, \mathit{secretion}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{up}$
(4) $\mathrm{BG} \rightarrow \circ\, \mathit{release}(\mathit{liver}, \mathit{glucose}) = \mathit{down}$
(5) $\mathit{diet} \wedge \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{normal} \rightarrow \circ\, \mathrm{Condition}(\mathit{normoglycaemia})$
(6) $(\circ\, \mathit{secretion}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{up} \wedge \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{subnormal} \wedge$
$\mathrm{QI} \leq 27 \wedge \mathrm{Condition}(\mathit{hyperglycaemia})) \rightarrow \circ\, \mathrm{Condition}(\mathit{normoglycaemia})$
(7) $(\circ\, \mathit{release}(\mathit{liver}, \mathit{glucose}) = \mathit{down} \wedge \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{subnormal} \wedge$
$\mathrm{QI} > 27 \wedge \mathrm{Condition}(\mathit{hyperglycaemia})) \rightarrow \circ\, \mathrm{Condition}(\mathit{normoglycaemia})$
(8) $((\circ\, \mathit{release}(\mathit{liver}, \mathit{glucose}) = \mathit{down} \vee \circ\, \mathit{uptake}(\mathit{peripheral\text{-}tissues}, \mathit{glucose}) = \mathit{up}) \wedge$
$\mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{nearly\text{-}exhausted} \wedge \circ\, \mathit{secretion}(\mathit{b\text{-}cells}, \mathit{insulin}) =$
$\mathit{up} \wedge$
$\mathrm{Condition}(\mathit{hyperglycaemia})) \rightarrow \circ\, \mathrm{Condition}(\mathit{normoglycaemia})$
(9) $(\circ\, \mathit{uptake}(\mathit{liver}, \mathit{glucose}) = \mathit{up} \wedge \circ\, \mathit{uptake}(\mathit{peripheral\text{-}tissues}, \mathit{glucose}) = \mathit{up} \wedge$
$\mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{exhausted} \wedge \mathrm{Condition}(\mathit{hyperglycaemia})) \rightarrow$
$\circ\, (\mathrm{Condition}(\mathit{normoglycaemia}) \vee \mathrm{Condition}(\mathit{hypoglycaemia}))$
(10) $(\mathrm{Condition}(\mathit{normoglycaemia}) \oplus \mathrm{Condition}(\mathit{hypoglycaemia}) \oplus$
$\mathrm{Condition}(\mathit{hyperglycaemia})) \wedge \neg\, (\mathrm{Condition}(\mathit{normoglycaemia}) \wedge$
$\mathrm{Condition}(\mathit{hypoglycaemia}) \wedge \mathrm{Condition}(\mathit{hyperglycaemia}))$

Fig. 4. Background knowledge $\mathcal{B}_{\mathrm{DM2}}$ of diabetes mellitus type 2. An action α holds iff drug x is being administered at that moment in time. The $\oplus$ operator denotes the exclusive OR operator.

The fragment of DM2 is relatively simple, however, diabetes is in fact a complicated disease: various metabolic control mechanisms are deranged and many different organ systems may be affected by the disorder. Pathophysiologically, there are two main phenomena, namely, insufficient secretion of the hormone insulin due to a decreased production of insulin by *B cells* in the Langerhans islets of the *pancreas*, and insulin resistance in liver, muscle and fat tissue. Parts of these mechanisms are described in more detail in [6]. These physiological mechanisms were modelled in temporal logic, which is described in Fig. 4.

6.3 Modelling of the Guideline

In this paper, we mainly focus on the modelling of the guideline fragment of Fig. 3. The possible actions that can be performed is the set $\mathcal{A}$ consisting of $\{\mathit{diet}, \mathit{SU}, \mathit{BG}, \mathit{insulin}\}$. Each treatment A is a subset of $\mathcal{A}$. Treatment changes if a treatment has failed, which can be conveniently be formalised in ITALF. The main structure of the guideline, denoted by $\mathcal{M}$, is then:

$$\begin{array}{l}
\Box\ \mathrm{treatment} = \{\mathrm{diet}\} \\
\mathbf{orelse}_{\{\mathrm{diet}\}}\ (\mathbf{if}\ \mathrm{QI} < 27\ \ \mathbf{then}\ (\Box \mathrm{treatment} = \{\mathrm{SU}\}) \\
\qquad\qquad\qquad\qquad\qquad\quad \mathbf{else}\ (\Box \mathrm{treatment} = \{\mathrm{BG}\}) \\
\qquad\qquad\quad \mathbf{orelse}_{\{\mathrm{SU},\mathrm{BG}\}}\ (\Box \mathrm{treatment} = \{\mathrm{SU}, \mathrm{BG}\} \\
\qquad\qquad\qquad\qquad\qquad\quad\ \ \mathbf{orelse}_{\{\mathrm{SU},\mathrm{BG}\}}\ \Box \mathrm{treatment} = \{\mathrm{insulin}\}))
\end{array}$$

where each term treatment $= A$ is an abbreviation for:

$$\bigwedge(\{\alpha \mid \alpha \in A\} \cup \{\neg\alpha, \neg\mathbf{fail}\ \alpha \mid \alpha \in (\mathcal{A} \setminus A)\})$$

i.e., the actions in A are activated, and all other actions are inactive (i.e., false and have not failed). This formalisation includes the handling of the failures in some sense, however, we also need to define in which cases these failures occur. One can think of this as 'throwing' the exceptions during the management of the disease. Define an abbreviation for this as follows:

$$\mathbf{fails}\,\varphi \triangleq \circ\,\mathbf{fail}\,\varphi$$

The guideline does not specify what amount of time is allowed to pass before it can be concluded that the treatment is not effective. Clearly, if a failure occurs immediately, then patients will all receive insulin treatment. Here, we assume the property of the background knowledge that relevant effects with respect to the condition of the patient are known in the next state. Hence, decisions whether the treatment fails can be taken after one step in the execution. These failure axioms are denoted as $\mathcal{F}$ and formalised as follows:

$$\Box\ (\alpha_i \rightarrow \circ\ ((\alpha_i \wedge \text{Condition}(\mathit{hyperglycaemia})) \leftrightarrow \mathbf{fails}\,\alpha_i))$$

for all $\alpha \in \mathcal{A}$.

6.4 Verification

Several tools for ITL have been developed, such as the interpreter Tempura [9] and support for ITL in the theorem prover PVS [3]. For our experiments, we have used the KIV system, an interactive theorem prover, designed for program verification and capable of reasoning about algebraic specifications using classical, dynamic and (interval) temporal logic. The main proof strategy for temporal logic is symbolic execution with induction. Symbolic execution unwinds formulas, e.g.,

$$\Box\ \varphi \Leftrightarrow \varphi \wedge \circ\ \Box\ \varphi$$

and induction is used to proof reason about recurring temporal states. Its theoretical background is described extensively in [2]. Below, we will write sequents $\Gamma \vdash \Delta$ to denote $\mathcal{I}(\Gamma \cup \Delta) \vdash_{\text{KIV}} \mathsf{reduce}(\bigwedge \Gamma \rightarrow \bigvee \Delta)$, where $\vdash_{\text{KIV}}$ denotes the deductibility relation defined by the sound (propositional and temporal) inference rules implemented in KIV.

In the specification of properties presented, we made use of algebraic specification to specify the variables in the background knowledge, though it could be translated to propositional logic if necessary. Furthermore, we made use of some additional variables to represent each treatment (e.g., '*treatmentdiet*' defined as 'treatment $= \{$diet$\}$'), and both failure-states. In practice, this makes the proofs more manageable. The relationship between the actions and these additional variables are defined appropriately in the system, i.e., all the additional propositional variables could be replaced by actions and failure of actions.

Example 1: Diet may be applied indefinitely. The first example is the following property. Let $\mathcal{B}_{DM2}$ be the background knowledge, $\mathcal{M}$ be the guideline given in Section 6.3, and $\mathcal{F}$ failure axioms defined in Section 6.3, then:

$$\begin{array}{l}\mathcal{B}_{DM2},\, \mathcal{M}, \mathcal{F},\, \Box\ \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{normal} \\ \qquad\qquad \vdash \Box \bullet \mathrm{Condition}(\mathit{normoglycaemia})\end{array}$$

i.e., in case the patient has B cells with sufficient capacity to produce insulin, then diet is sufficient for lowering the level of glucose in the blood. As only the failure of diet is relevant in the proof, $\mathcal{M}$ can be weakened to:

$$(\Box \mathit{treatmentdiet}) \wedge \neg\, \mathbf{last};\, f_{\mathrm{diet}}$$

Symbolic execution, in the context of the background knowledge, leads to the situation where:

$$(\Box \mathit{treatmentdiet};\, f_{\mathrm{diet}}) \wedge \mathrm{Condition}(\mathit{normoglycaemia})$$

Since we have Condition(*normoglycaemia*), it can be derived that *diet* does not fail, thus in the next step it can be derived that the condition is still normoglycaemia, which is exactly the same situation as we had before. By induction, we can then reason that this will *always* be the case. A more detailed proof can be found in Appendix B.

Example 2: Reasoning about the patient in case of failure. Guidelines are not applied blindly by physicians, as the physician has to make a decision for an individual patient on the basis of all known information. As a consequence, a physician might be interested in reasons of failure. Suppose we have an arbitrary patient, then we can prove the following:

$$\mathcal{B}_{DM2},\, \mathcal{M}, \mathcal{F}\ \vdash \mathbf{fail}(\Box\ \mathit{diet}) \rightarrow \Diamond \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) \neq \mathit{normal}$$

i.e., if *always* applying diet fails, then apparently the patient has non-normal capacity of its B cells at a certain moment in time. $\mathcal{M}$ is needed here to derive that in case diet stops, a failure has occurred rather than a non-failing termination of diet. Proving this in KIV is similar as the previous example.

Example 3: Level of sugar in the blood will decrease. As a third example, we use one of the quality criteria for the diabetes guideline from [6]. This property says that the guideline reaches its intention, namely, the level of sugar in the blood will be lowered for any patient group. This property is formalised as follows:

$$\begin{array}{l}\mathcal{B}_{DM2},\, \mathcal{M},\, \mathcal{F}, \Box\ (\mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin})'') \wedge \\ \qquad\qquad \Box\ \mathrm{QI} = \mathrm{QI}'' \vdash \Diamond \neg\, \mathrm{Condition}(\mathit{hyperglycaemia})\end{array}$$

where V'' denotes the value of the variable V in the next step. Our proof strategy consisted of splitting the patient group into groups which are cured by the same treatment, e.g., similar to the previous example, when the capacity is normal, then diet is sufficient.

Consider the example where the capacity of insulin in the B cells is nearly-exhausted. KIV derives from the failure axioms that:

$$\square\ (\alpha_i \rightarrow \circ\ (\alpha_i \leftrightarrow \circ\ (\neg\alpha_i \wedge f_{\alpha_i})))$$

as we may assume that $\square\ \neg$ Condition(*hyperglycaemia*), because the negation of this formula immediately proves the property. Furthermore, reasoning with the background knowledge, we can derive that proving $\Diamond\ (SU \wedge BG)$ is sufficient to prove this property, because for this patient group a treatment consisting of SU and BG is sufficient to conclude Condition(*normoglycaemia*). It is then easy to see how to complete this proof as the failure axioms specify that all the treatments will fail (after two steps), hence symbolic execution shows that eventually the third step will be activated.

7 Related Work

Failure has received little attention in formal theories of action. Of course, reasoning of actions had always taken into account the notion of failure, as illustrated by the logic of Rao & Georgeff, but it is assumed that failure can be added in a relatively straightforward manner. One notable example of where the notion of failure is part of both the syntax and semantics is the approach of Giunchiglia et al. [4]. Its primitive syntactic structure is:

$$\textbf{iffail}\ \alpha\ \textbf{then}\ \beta\ \textbf{else}\ \gamma$$

And from this, abbreviations are defined such that it allows one to reason conveniently about failures. The semantics is defined in terms of *behaviours* where it said that some behaviours have *failed*, while others are *successful*. Behaviours are defined technically in terms of linear models.

What this language lacks is the notion of time, as behaviours are simply considered a sequence of actions which either fail or do not fail. For medical management, this poses a problem, as failure may occur after a longer period of time. This means that the notion of failure needs a richer structure, so that it is possible to interact between time and failure.

Another important shortcoming for using this language in the context of medical management is that failures are considered properties of a *behaviour*. As said before, in medical management, actions are often performed in parallel, for example, the administering of a combination of drugs. In such cases, some drugs may fail to reach the required effects, while others may be successful. Hence, in the language decisions need to be made on, not only *if* a failure has occurred, but also *what* action has failed. We believe we have clearly demonstrated this in the previous section.

8 Discussion and Conclusions

In this paper, we have introduced semantics of failures in interval temporal logic inspired by the exception mechanism that can be found in many programming languages. The practical usefulness of our approach has been validated using medical guidelines by showing the verification of a fragment of diabetes mellitus type 2 which was formalised elegantly using this logic. However, we think that the results could be used in a much wider context. First, the reasoning about failures can have its applications in agent-based systems. Failures to perform tasks are an important aspect for decision making by agents, so having a reasonably rich language for modelling these failures seems justified. Second, in the context of program refinement, the process of (high-level) specifications to implementations of systems, exceptions are introduced at some point to model failure of components. The results of this paper makes it possible to abstract of concrete programming construct to describe how control of flow should change in case exceptions occur.

The logic that is proposed here can be seen as a three-valued logic, i.e., formulas are true, false, or failed. Some work has been done to link three-valued logics idea to temporal reasoning [7], which is based on Kleen's three-valued calculus that deals with 'unknown' values. This results in different logical properties compared to ITALF, e.g., unknown values propagate over a disjunctions, while failures do not.

Compared to [6], the verification of the investigated properties required significantly less effort. This is mainly due to the fact that in [6] the guideline was formalised in the guideline representation language Asbru [13], which yields overhead in complexity due to a complicated semantics. On the other hand, many of the steps that are required in ITALF were done manually, as it is not obvious to predict the correct next step in the proof. For example, it is important during verification to 'weaken' the irrelevant parts of the guideline, making the symbolic execution more efficient. Moreover, failure propositions on the sequent introduce additional complexity, as the human needs to remember the semantics of these propositions in order to apply the relevant axioms. These facts combined makes it interesting to consider more automatic techniques, such as automated theorem proving or model checking. This will a subject of further research.

References

1. Alves-Foss, J., Lam, F.S.: Dynamic denotational semantics of Java. In: Alves-Foss, J. (ed.) Formal Syntax and Semantics of Java. LNCS, vol. 1523. Springer, Heidelberg (1999)
2. Balser, M.: Verifying Concurrent Systems with Symbolic Execution – Temporal Reasoning is Symbolic Execution with a Little Induction. PhD thesis, University of Augsburg, Augsburg, Germany (2005)
3. Cau, A., Moszkowski, B.: Using PVS for interval temporal logic proofs. part 1: The syntactic and semantic encoding. Technical report, SERCentre, De Montfort University, Leicester (1996)
4. Giunchiglia, F., Spalazzi, L., Traverso, P.: Planning with failure. Artificial Intelligence Planning Systems, 74–79 (1994)
5. Hindriks, K., de Boer, F., van der Hoek, W., Meyer, J.-J.C.: Failure, monitoring and recovery in the agent language 3APL. In: De Giacomo, G. (ed.) AAAI 1998 Fall Symposium on Cognitive Robotics, pp. 68–75 (1998)

6. Hommersom, A.J., Groot, P.C., Lucas, P.J.F., Balser, M., Schmitt, J.: Verification of medical guidelines using background knowledge in task networks. IEEE Transactions on Knowledge and Data Engineering 19(6), 832–846 (2007)
7. Konikowska, B.: A three-valued linear temporal logic for reasoning about concurrency. Technical report, ICS PAS, Warsaw (1998)
8. Moszkowski, B.: A temporal logic for multilevel reasoning about hardware. IEEE Computer 18(2), 10–19 (1985)
9. Moszkowski, B.: The programming language Tempura. Journal of Symbolic Computation 22(5/6), 730–733 (1996)
10. von Oheimb, D., Nipkow, T.: Machine-checking the Java specification: Proving type-safety. In: Alves-Foss, J. (ed.) Formal Syntax and Semantics of Java. LNCS, vol. 1523. Springer, Heidelberg (1999)
11. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In: Allen, J., Fikes, R., Sandewall, E. (eds.) Proceedings of KR 1991, pp. 473–484. Morgan Kaufmann, San Francisco (1991)
12. Rutten, G.E.H.M., Verhoeven, S., Heine, R.J., de Grauw, W.J.C., Cromme, P.V.M., Reenders, K.: NHG-standaard diabetes mellitus type 2 (eerste herziening). Huisarts Wet 42, 67–84 (1999)
13. Shahar, Y., Miksch, S., Johnson, P.: The Asgaard project: A task-specific framework for the application and critiquing of time-orientied clinical guidelines. Artificial Intelligence in Medicine 14, 29–51 (1998)
14. Tucker, A.B., Noonan, R.E.: Programming Languages – Principles and Paradigms, 2nd edn. McGraw-Hill, New York (2007)
15. Turner, R.: Logics for Artificial Intelligence. Ellis Horwood, Chichester (1985)

A Proof of Theorem 1

A helpful semantic notion is faithfulness, which means that the failure propositions correspond exactly to the failure of the the action it has been introduced for.

Definition 5. *σ is called* faithful *iff for all $a \in \mathcal{A}$ and all i s.t. $0 \leq i \leq |\sigma|$ holds $\alpha_i(a) = \mathsf{failed}$ iff $\pi_i(f_a) = \top$.*

In the following two lemmas, it is proven that the reduction is found with respect to those faithful models. In the first lemma, we show that Φ acts as failmodel on the syntactic level, which is then used to prove equivalence of formulas with its reduction.

Lemma 1. *For all faithful σ and φ:*

$$\mathit{failmodel}(\sigma) \models \varphi \textit{ iff } \sigma \models \Phi(\varphi)$$

Proof. By induction on the structure of φ. First suppose $\varphi = a$: ($\Rightarrow$) suppose $\mathsf{failmodel}(\sigma) \models a$ then $\alpha_0(a) \neq \mathsf{failed}$. By faithfulness $\pi_i(f_a) = \bot$, thus $\sigma \models \neg f_a$. All steps can be reversed. The rest of the cases follow almost immediately, taking into account that if the model is faithful, so is every interval within this model, and vice versa.

Lemma 2. *For all faithful models σ it holds that $\sigma \models \varphi \leftrightarrow \mathit{reduce}(\varphi)$.*

Proof. By induction on the structure of φ. In this case, the only interested case is for $\varphi = \mathbf{fail}(\psi)$: ($\Rightarrow$) $\sigma \models \mathbf{fail}(\psi)$ iff $\sigma \not\models \psi$ and $\mathsf{failmodel}(\sigma) \not\models \psi$. By I.H. on the first part, it follows that $\sigma \not\models \mathsf{reduce}(\varphi)$. As σ is faithful, it follows that $\mathsf{failmodel}(\sigma)$ is faithful. Therefore $\mathsf{failmodel}(\sigma) \not\models \mathsf{reduce}(\varphi)$. Using Lemma 1, we get $\sigma \not\models \Phi(\mathsf{reduce}(\varphi))$. Therefore $\sigma \models \neg\mathsf{reduce}(\varphi) \wedge \neg\Phi(\mathsf{reduce}(\varphi))$. By definition, $\sigma \models \mathsf{reduce}(\mathbf{fail}(\varphi))$. All steps are valid in the other direction as well.

These results do not hold for any model, e.g., it is not for all models the case that $f_a \rightarrow \neg a$. A weak form of faithfulness can be encoded as an ITL formula, bounded by the number of actions in some formula. The fact it is bounded by actions in a formula is relevant, because we may have an infinite number of actions in the language, while each formula has a finite length in standard temporal logic.

Using Definition 4, we can then proof the main lemma, which characterises the relation between a formula and its reduction for any model.

Lemma 3. $\models \varphi$ *iff* $\models \mathcal{I}(\varphi) \rightarrow \mathit{reduce}(\varphi)$

Proof. Without loss of generality, this property can be reformulated as

$$\models \neg\varphi \text{ iff } \mathcal{I}(\varphi) \models \mathsf{reduce}(\neg\varphi)$$

as every formula can be stated as a negation and $\mathcal{I}(\neg\varphi) = \mathcal{I}(\varphi)$. Using the definition of reduce, and taking negation on both sides, rewrite this to:

$$\exists_\sigma \sigma \models \varphi \text{ iff } \exists_\sigma \sigma \models \mathcal{I}(\varphi) \wedge \mathsf{reduce}(\varphi)$$

($\Rightarrow$) Suppose there is some σ such that $\sigma \models \varphi$. Construct a σ' such that $\pi'_i(f_a) = \top$ iff $\alpha_i(a) = \mathsf{failed}$, for all $0 \leq i \leq |\sigma|$, actions a, and all fresh variables f_a introduced in the reduction. Let σ' be the same as σ in every other respect. As φ does not contain any variables f_a, it is clear that then $\sigma' \models \varphi$. As σ' is faithful (by construction), it then follows by Lemma 2 that $\sigma' \models \mathsf{reduce}(\varphi)$. Moreover, by construction, it follows that $\sigma' \models \mathcal{I}(\varphi)$.
($\Leftarrow$) Suppose for some σ, $\sigma \models \mathcal{I}(\varphi) \wedge \mathsf{reduce}(\varphi)$. Construct σ' such that for all i such that $0 \leq i \leq |\sigma|$, and all actions a:

- if $\pi_i(f_a) = \top$ then $\alpha'_i(a) = \mathsf{failed}$
- if $\pi_i(f_a) = \bot$ and $\alpha_i(a) = \mathsf{active}$ then $\alpha'_i(a) = \mathsf{active}$
- if $\pi_i(f_a) = \bot$ and $\alpha_i(a) \neq \mathsf{active}$ then $\alpha'_i(a) = \mathsf{inactive}$

In all other respects (length, valuation of atomic propositions), σ and σ' are the same. We then prove for all i and $a \in \mathit{actions}(\varphi)$:

$$\alpha_i(a) = \mathsf{active} \Leftrightarrow \alpha'_i(a) = \mathsf{active}$$

($\Rightarrow$) $\alpha_i(a) = \mathsf{active}$. Then, by the fact $\sigma \models \mathcal{I}(\varphi)$, we know that $\pi_i(f_a) = \bot$. Thus, by definition $\alpha'_i(a) = \mathsf{active}$. ($\Leftarrow$) Suppose $\alpha_i(a) \neq \mathsf{active}$. Then either

$\alpha'_i(a) = \mathsf{failed}$ (if $\pi_i(f_a) = \top$) or $\alpha'_i(a) = \mathsf{inactive}$ (if $\pi_i(f_a) = \bot$). In any case, we conclude: $\alpha'_i(a) \neq \mathsf{active}$.
As $\mathsf{reduce}(\varphi)$ does not contain a **fail** operator, it cannot distinguish if an action is inactive or failed. Hence, it follows that $\sigma' \models \mathsf{reduce}(\varphi)$. It is easy to see that σ' is faithful, so by Lemma 2 it follows that $\sigma' \models \varphi$.

Now, Theorem 1 is proved in the following way. By Lemma 3, we know $\models \varphi$ iff $\models \mathcal{I}(\varphi) \to \mathsf{reduce}(\varphi)$. Observe that the right side does not contain the **fail** operator, hence it cannot distinguish between failures and inactivations. Therefore, $\models \mathcal{I}(\varphi) \to \mathsf{reduce}(\varphi)$ if all actions are interpreted as propositions. By doing this, $\mathcal{I}(\varphi) \to \mathsf{reduce}(\varphi)$ is also an ITL formula. Finally, note that the semantics of ITL and ITALF coincide for the language of ITL.

B Proof of Example 1

This appendix provides an outline of the proof performed in KIV. The first steps of the proof consists of simple manipulation of the formulas in order to put them in a comfortable form for presenting the proof. Note that we implicitly use axiom (10) of the background knowledge for making sure that normo-, hyper-, and hypoglycaemia are mutually exclusive. First, recall that the translated failure axiom for diet is:

$$\Box\ (diet \to \circ\ ((diet \land Condition(hyperglycaemia) \leftrightarrow \circ\ \mathbf{fail}\ \ diet))$$

Reduction of this to an ITL formula yields:

$$\Box\ (diet \to \circ\ ((diet \land Condition(hyperglycaemia) \leftrightarrow \circ\ (\neg diet \land f_{diet})))$$

which, by the use of Γ, can be written as:

$$\Box\ (diet \to (\circ\ (diet \land Condition(hyperglycaemia)) \leftrightarrow \circ\ \circ\ f_{diet})) \tag{3}$$

Second, from the background knowledge, we know that:

$$\Box\ (diet \land capacity(b\text{-}cells, insulin) = normal \to \circ\ \mathrm{Condition}(normoglycaemia))$$

which, together with the fact that $\Box\ capacity(b\text{-}cells, insulin) = normal$, it can be automatically derived that:

$$\Box\ (diet \to \circ\ \mathrm{Condition}(normoglycaemia)) \tag{4}$$

Finally, note that the proof obligation can be presented as

$$\bullet\ \Box\ \mathrm{Condition}(normoglycaemia) \tag{5}$$

By weakening all the uninteresting parts for proving the property, we finally end up with the main proof obligation:

$$\begin{array}{lr}
\Box\ (diet \to (\circ\ (diet \land \mathrm{Condition}(hyperglycaemia)) \leftrightarrow \circ\ \circ\ f_{diet})), & Eq.(3)\\
\Box\ (diet \to \circ\ \mathrm{Condition}(normoglycaemia)), & Eq.(4)\\
(\Box\ treatmentdiet \land \neg\ \mathbf{last});\ \circ\ f_{diet}, & \mathcal{M}\\
\Box\ (treatmentdiet \to diet), & \\
\vdash\ \bullet\ \Box\ \mathrm{Condition}(normoglycaemia) & Eq.(5)
\end{array}$$

Symbolically executing this sequent requires only one possible situation that needs to be proven:

$$\begin{array}{l} \Box\ (\textit{diet} \to (\circ\ (\textit{diet} \wedge \text{Condition}(\textit{hyperglycaemia})) \leftrightarrow \circ\ \circ\ f_{diet})), \\ \Box\ (\textit{diet} \to \circ\ \text{Condition}(\textit{normoglycaemia})), \\ (\Box\ \textit{treatmentdiet});\ \circ\ f_{diet}, \\ \Box\ (\textit{treatmentdiet} \to \textit{diet}), \\ \text{Condition}(\textit{normoglycaemia}), \neg\ \circ\ f_{diet} \\ \vdash \Box\ \text{Condition}(\textit{normoglycaemia}) \end{array}$$

This sequent represents the situation where diet has been applied in the first step. From this it was derived that then the condition is normoglycaemia. Using this fact, the failure axiom is used to derive that $\neg\ \circ\ f_{diet}$, i.e., diet will not fail in the next step. The rest of the proof consists of the claim that this temporal situation will remain as it is. So we reason by induction that $\Box$ Condition $(\textit{normoglycaemia})$. Abbreviate the sequent above as $\Gamma \vdash \Delta$: then the sequent is rewritten to:

$$\begin{array}{l} \Box\ (\textit{diet} \to (\circ\ (\textit{diet} \wedge \text{Condition}(\textit{hyperglycaemia})) \leftrightarrow \circ\ \circ\ f_{diet})), \\ \Box\ (\textit{diet} \to \circ\ \text{Condition}(\textit{normoglycaemia})), \\ (\Box\ \textit{treatmentdiet});\ \circ\ f_{diet}, \\ \Box\ (\textit{treatmentdiet} \to \textit{diet}), \\ \text{Condition}(\textit{normoglycaemia}), \neg\ \circ\ f_{diet}, \\ t = N, N = N'' + 1\ \textbf{until}\ \neg\ \text{Condition}(\textit{normoglycaemia}), \textbf{IND-HYP} \vdash \end{array}$$

where **IND-HYP** $\triangleq t < N \to (\bigwedge \Gamma \to \bigvee \Delta)$, N a fresh dynamic variable and t a static variable. The remaining steps consists of symbolically executing this sequent, which ends up in the same sequent with $t = N - 1$. Then, the induction hypothesis can be applied, which finishes the proof.

A Logic for Reasoning about Rational Agents

Wojciech Jamroga and Nils Bulling

Department of Informatics, Clausthal University of Technology, Germany
{wjamroga,bulling}@in.tu-clausthal.de

Abstract. We have recently proposed an extension of alternating-time temporal logic for reasoning about behavior and abilities of agents under various rationality assumptions. The logic, called **ATLP** ("alternating-time temporal logic with plausibility") used abstract, unstructured terms for addressing rationality assumptions. Here, we propose a more complex language of terms that allows to specify sets of rational strategy profiles in the object language, building upon existing work on logical characterizations of game-theoretic solution concepts. In particular, we recall how the notions of Nash equilibrium, subgame-perfect Nash equilibrium, and Pareto optimality can be characterized with logical formulae and we show how these can be used within **ATLP** for reasoning about what rational agents should achieve. We also prove complexity results for model checking of **ATLP** formulae.

1 Introduction

Alternating-time temporal logic (**ATL**) [2] is a temporal logic that incorporates some basic game theoretical notions. In [15], we extended **ATL** with a notion of *plausibility*, which can be used to model and reason about what agents can plausibly achieve. Our intuition was to use game-theoretical solution concepts (like Nash equilibrium, Pareto optimality, dominant strategies etc.) to define what it means to play rationally, and then to assume it plausible that agents behave in a rational way. Technically, some strategies (or rather *strategy profiles*) were assumed plausible in a given model, and one could reason about what can happen if only the plausible profiles are used.

The formulation of *alternating-time temporal logic with plausibility* (**ATLP**) from [15] was rather abstract, with unstructured terms used to address various rationality assumptions, and their denotation "hard-wired" in the model. In this paper, we propose to refine the language of terms so that it allows us to specify sets of rational strategy profiles in the object language. The idea is to build the terms on formulae of **ATLI** (*ATL with intentions*, [21]), as these can be used to describe sets of strategies and strategy profiles. We build upon existing work on modal logic characterizations of solution concepts [13,12,3,31,32,21]. In particular, we recall how the notions of Nash equilibrium, subgame-perfect Nash equilibrium, and Pareto optimality can be characterized with **ATLI** formulae. Then, we show how these characterizations can be used within **ATLP** for reasoning about abilities and behavior of rational agents.

The idea to define some strategies as plausible (or rational) is very much in the spirit of game theory. There, it is usually assumed that some solution concept is given and that agents are rational if they behave in accordance with it. Thus, assuming rationality

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 42–61, 2008.

of agents ultimately restricts the strategies that can be played by the agents. There are two possible points of focus in this context: *characterization* of rationality (and defining most appropriate solution concepts) vs. *using* the solution concepts in order to predict the outcome in a given game. In our previous paper [15], we proposed a logic for the latter task, i.e. for "plugging in" a given rationality criterion, and reasoning about what can happen according to it. In this paper, we try to bridge the gap, and propose how logic-based characterizations of rationality (formulated in **ATLI**) can be used to reason about the outcome of rational play within the framework of **ATLP**. The work is still in preliminary stages, and it certainly has its limitations, but we believe it to be a step in the right direction.

We begin our presentation with an overview of the logics **ATL** and **ATLP** (Section 2) which can be used to reason about abilities of arbitrary and rational agents, respectively. In Section 3, we recall how some rationality criteria can be captured with formulae of an **ATL**-like logic. Section 4 contains the novel contribution of this paper. First, we propose how those **ATL**-like characterizations of rationality can be used in **ATLP** in order to "plug in" rationality assumptions in a flexible way. Then, we discuss the computational complexity of model checking properties of rational play, both for "prewired" and flexible rationality definitions.

2 Preliminaries

In this section, we summarize two modal logics for reasoning about agents in game-like scenarios: first, the basic logic of **ATL** [2]; then, its extension **ATLP** [15].

2.1 Alternating-Time Temporal Logic

Alternating-time temporal logic (**ATL**) [2] enables reasoning about temporal properties and strategic abilities of agents. Formally, the language of **ATL** is given as follows.

Definition 1 **($\mathcal{L}_{ATL}$ [2]).** *Let* $\mathbb{A}\text{gt} = \{1, \dots, k\}$ *be a nonempty finite set of all agents, and* Π *be a set of propositions (with typical element* p*). We will use symbol* a *to denote a typical agent, and* A *to denote a typical group of agents from* $\mathbb{A}\text{gt}$. *The logic* $\mathcal{L}_{ATL}(\mathbb{A}\text{gt}, \Pi)$ *is defined by the following grammar:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A \rangle\!\rangle \bigcirc \varphi \mid \langle\!\langle A \rangle\!\rangle \Box \varphi \mid \langle\!\langle A \rangle\!\rangle \varphi \,\mathcal{U}\, \varphi.$$

Informally, $\langle\!\langle A \rangle\!\rangle \varphi$ says that agents A have a collective strategy to enforce φ. **ATL** formulae include the usual temporal operators: $\bigcirc$ ("in the next state"), $\Box$ ("always from now on") and $\mathcal{U}$ (strict "until"). Additionally, $\Diamond$ ("now or sometime in the future") can be defined as $\Diamond\varphi \equiv \top \,\mathcal{U}\, \varphi$. It should be noted that the path quantifiers A, E of computation tree logic **CTL** [8] can be expressed in **ATL** with $\langle\!\langle \emptyset \rangle\!\rangle, \langle\!\langle \mathbb{A}\text{gt} \rangle\!\rangle$, respectively. The semantics of **ATL** is defined in so-called *concurrent game structures*.

Definition 2 **(CGS [2]).** *A* concurrent game structure (**CGS**) *is a tuple:* $M = \langle \mathbb{A}\text{gt}, Q, \Pi, \pi, Act, d, o \rangle$, *consisting of: a set* $\mathbb{A}\text{gt} = \{1, \dots, k\}$ *of* agents*; set* Q *of* states*; set* Π *of* atomic propositions*;* valuation of propositions $\pi : Q \to \mathcal{P}(\Pi)$*; set* Act *of* actions. *Function* $d : \mathbb{A}\text{gt} \times Q \to \mathcal{P}(Act)$ *indicates the actions available to agent*

$a \in \mathbb{A}\mathrm{gt}$ *in state* $q \in Q$*. We will often write* $d_a(q)$ *instead of* $d(a, q)$*, and use* $d(q)$ *to denote the set* $d_1(q) \times \cdots \times d_k(q)$ *of* action profiles *in state* q*. Finally,* o *is a* transition function *which maps each state* $q \in Q$ *and action profile* $\overrightarrow{\boldsymbol{\alpha}} = \langle \alpha_1, \ldots, \alpha_k \rangle \in d(q)$ *to another state* $q' = o(q, \overrightarrow{\boldsymbol{\alpha}})$*.*

A *computation* or *path* $\lambda = q_0 q_1 \cdots \in Q^+$ is an infinite sequence of states such that there is a transition between each q_i, q_{i+1}.We define $\lambda[i] = q_i$ to denote the i-th state of λ. Λ_M denotes all paths in M. The set of all paths starting in q is given by $\Lambda_M(q)$.

Definition 3 (Strategy, outcome [2]). *A (memoryless)* strategy *of agent* a *is a function* $s_a : Q \to Act$ *such that* $s_a(q) \in d_a(q)$[1]*. We denote the set of such functions by* Σ_a*. A* collective strategy s_A *for team* $A \subseteq \mathbb{A}\mathrm{gt}$ *specifies an individual strategy for each agent* $a \in A$*; the set of* A*'s collective strategies is given by* $\Sigma_A = \prod_{a \in A} \Sigma_a$*. The set of all* strategy profiles *is given by* $\Sigma = \Sigma_{\mathbb{A}\mathrm{gt}}$*.*

The outcome *of strategy* s_A *in state* q *is defined as the set of all paths that may result from executing* s_A*:* $out(q, s_A) = \{\lambda \in \Lambda_M(q) \mid \forall i \in \mathbb{N}_0\ \exists \overrightarrow{\boldsymbol{\alpha}} = \langle \alpha_1, \ldots, \alpha_k \rangle \in d(\lambda[i])\ \forall a \in A\ (\alpha_a = s_A^a(\lambda[i]) \wedge o(\lambda[i], \overrightarrow{\boldsymbol{\alpha}}) = \lambda[i+1])\}$*, where* s_A^a *denotes agent* a*'s part of the collective strategy* s_A*.*

The semantics of **ATL** is given by the following clauses:

$M, q \models p$ iff $p \in \pi(q)$

$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$

$M, q \models \varphi \wedge \psi$ iff $M, q \models \varphi$ and $M, q \models \psi$

$M, q \models \langle\langle A \rangle\rangle \bigcirc \varphi$ iff there is $s_A \in \Sigma_A$ such that $M, \lambda[1] \models \varphi$ for all $\lambda \in out(q, s_A)$

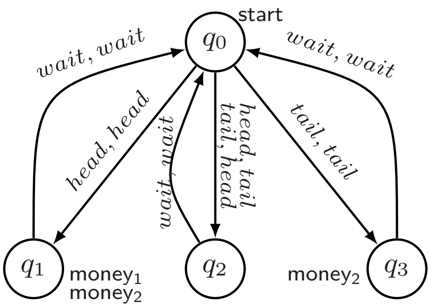
$M, q \models \langle\langle A \rangle\rangle \Box \varphi$ iff there is $s_A \in \Sigma_A$ such that $M, \lambda[i] \models \varphi$ for all $\lambda \in out(q, s_A)$ and $i \in \mathbb{N}_0$

$M, q \models \langle\langle A \rangle\rangle \varphi \mathcal{U} \psi$ iff there is $s_A \in \Sigma_A$ such that, for all $\lambda \in out(q, s_A)$, there is $i \in \mathbb{N}_0$ with $M, \lambda[i] \models \psi$, and $M, \lambda[j] \models \varphi$ for all $0 \leq j < i$.

Example 1 (Matching pennies). Consider a variant of the "matching pennies" game, presented in Figure 1A. If both players show the heads in q_0, both win a prize in the next step; if they both show tails, only player 2 wins. If they show different sides, nobody wins. Note that, e.g., $M_1, q_0 \models \langle\langle 2 \rangle\rangle \Box \neg \mathsf{money}_1$, because agent 2 can play $tail$ all the time, preventing 1 from winning the prize. On the other hand, $M_1, q_0 \models \neg\langle\langle 2 \rangle\rangle \Diamond \mathsf{money}_2$: agent 2 has no strategy to guarantee that he will win himself.

Such an analysis of the game is of course correct, yet it appears to be quite coarse. It seems natural to assume that players prefer winning money over losing it. If we additionally assume that the players are rational thinkers, it seems plausible that player 1 should always play $head$, as it keeps the possibility of a win open (while playing $tail$ guarantees loss). Under this assumption, player 2 has complete control over the outcome of the game: he can play $head$ too, granting himself and the other agent with the prize, or respond with $tail$, in which case both players lose. Note that this kind of analysis

[1] This is a deviation from the original semantics of **ATL** [2], where strategies assign agents' choices to *sequences* of states. While the choice between the two types of strategies affects the semantics of most **ATL** extensions, both yield equivalent semantics for "pure" **ATL** .



$1\backslash2$	$head$	$tail$
$head$	$1,1$	$0,0$
$tail$	$0,0$	$0,1$

Fig. 1. "Matching pennies": (A) Concurrent game structure M_1; both agents can only execute action $wait$ in states q_1, q_2, q_3 (we omit the labels from the picture to make it easier to read). (B) The corresponding normal form game under the assumption that the winners always get the same amount of money.

corresponds to the game-theoretical notion of *dominant strategy*: for agent 1, playing $head$ is strongly dominant in the corresponding normal form game in Figure 1B, while both strategies of player 2 are undominated, so they can be in principle considered for playing.

It is still possible to refine our analysis of the game: note that 2, knowing that 1 ought to play $head$ and preferring to win money too, should decide to play $head$ himself. This kind of reasoning corresponds to the notion of *iterated undominated strategies*. If we assume that both players do reason this way, then $\langle head, head\rangle$ is the only rational strategy profile, and the game should end with both agents winning the prize.

2.2 ATL with Plausibility: Reasoning about Rational Agents

Agents have limited ability to predict the future. However, some lines of action seem often more sensible or realistic than others. Having defined a rationality criterion, we obtain means to determine the most plausible plays, and compute their outcome. In [15], we proposed an extension of **ATL** for reasoning about rational agents, which had in turn been inspired by the work by Van Otterloo and colleagues [34,36,37]. We called the logic **ATLP**, i.e., "**ATL** with plausibility"[2].

Definition 4 **($\mathcal{L}_{ATLP}$ [15]).** *Let* $\mathbb{A}\mathrm{gt}, \Pi$ *be as before, and* Ω *be a set of plausibility terms (with typical element* ω*). The language* $\mathcal{L}_{ATLP}(\mathbb{A}\mathrm{gt}, \Pi, \Omega)$ *is defined recursively as:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi\wedge\varphi \mid \langle\langle A\rangle\rangle \bigcirc\varphi \mid \langle\langle A\rangle\rangle\Box\varphi \mid \langle\langle A\rangle\rangle\varphi\,\mathcal{U}\,\varphi \mid \mathbf{Pl}\,\varphi \mid \mathbf{Ph}\,\varphi \mid (\textbf{\textit{set-pl}}\ \omega)\varphi.$$

Pl restricts the considered strategy profiles to ones that are *plausible* in the given model. **Ph** disregards plausibility assumptions, and refers to all *physically* available strategies.

[2] We observe that our framework is semantically similar to the approach of *social laws* [29,25,33]. However, we refer to *strategy profiles* as rational or not, while social laws define constraints on agents' *individual actions*. Also, our motivation is different: in our framework, agents are expected to behave in a specified way because it is rational in some sense; social laws prescribe behavior sanctioned by social norms and legal regulations.

$(\textbf{set-pl}\ \omega)$ allows to define (or redefine) the set of plausible strategy profiles to the ones described by plausibility term ω (in this sense, it implements *revision* of plausibility). With **ATLP**, we can for example say that $\textbf{Pl}\,\langle\!\langle\emptyset\rangle\!\rangle\Box\,(\mathsf{closed} \wedge \textbf{Ph}\,\langle\!\langle guard\rangle\!\rangle\bigcirc\neg\mathsf{closed})$: "it is plausible that the emergency door will always remain closed, but the guard retains the physical ability to open them"; or$(\textbf{set-pl}\ \omega_{NE})\textbf{Pl}\,\langle\!\langle a\rangle\!\rangle\Diamond\neg\mathsf{jail_a}$: "suppose that only playing Nash equilibria is rational; then, agents a can plausibly reach a state where he is out of prison. To define the semantics of **ATLP**, we extend **CGS** to *concurrent game structures with plausibility* (**CGSP**). Apart from an actual plausibility set Υ, a **CGSP** specifies a *plausibility mapping* $[\![\cdot]\!] : Q \rightarrow (\Omega \rightarrow \mathcal{P}(\Sigma))$ that maps each term $\omega \in \Omega$ to a set of strategy profiles, dependent on the current state.

Definition 5 (CGSP [15]). *A* concurrent game structure with plausibility (**CGSP**) *is given by a tuple* $M = \langle \mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o, \Upsilon, \Omega, [\![\cdot]\!]\rangle$ *where* $\langle \mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o\rangle$ *is a* ***CGS****,* $\Upsilon \subseteq \Sigma$ *is a set of plausible strategy profiles;* Ω *is a set of of plausibility terms, and* $[\![\cdot]\!]$ *is a plausibility mapping.*

When talking about the outcome of rational/plausible play (e.g., with formula $\textbf{Pl}\,\langle\!\langle A\rangle\!\rangle\gamma$), the strategy profiles that can be used *by all the agents* are restricted to the ones from Υ. Thus, coalition A can only choose strategies that are *substrategies* of plausible strategy profiles. Moreover, the agents in $\mathbb{A}\mathrm{gt}\backslash A$ can only respond in a way that yields a plausible strategy profile.

Definition 6 (Substrategy, outcome [15]). *Let* $A \subseteq B \subseteq \mathbb{A}\mathrm{gt}$*, and let* s_B *be a collective strategy for* B*. We use* $s_B[A]$ *to denote the* substrategy *of* s_B *for agents A, i.e., strategy* t_A *such that* $t_A^a = s_B^a$ *for every* $a \in A$*. Additionally, for a set of strategy profiles* P*,* $P(s_A)$ *denotes all strategy profiles from* P *that contain* s_A *as substrategy (i.e.,* $P(s_A) = \{s' \in P \mid s'[A] = s_A\}$*).*

Let M *be a* ***CGSP****,* $A \subseteq \mathbb{A}\mathrm{gt}$ *be a set of agents,* $q \in Q$ *be a state,* $s_A \in \Sigma_A$ *be a collective strategy of A, and* $P \subseteq \Sigma$ *be a set of strategy profiles. The set* $out(q, s_A, P)$ *contains all paths which may result from agents* A *executing* s_A*, when only strategy profiles from* P *can be played. Formally:* $out(q, s_A, P) = \{\lambda \in \Lambda_M(q) \mid \exists z \in P(s_A)\forall i\big(\lambda[i+1] = o(\lambda[i], z(\lambda[i]))\big)\}$*. Furthermore,* $\Sigma_A(P)$ *denotes all profiles of* A *consistent with P, i.e.,* $\Sigma_A(P) = \{s_A \in \Sigma_A \mid \exists t \in P\ \ s_A = t[A]\}$*.*

Let $P \subseteq \Sigma_{\mathbb{A}\mathrm{gt}}$ be a set of strategy profiles. The semantics of **ATLP** is given by the satisfaction relation $\models_P$ defined as follows:

$M, q \models_P p$ iff $p \in \pi(q)$
$M, q \models_P \neg\varphi$ iff $M, q \not\models_P \varphi$
$M, q \models_P \varphi \wedge \psi$ iff $M, q \models_P \varphi$ and $M, q \models_P \psi$
$M, q \models_P \langle\!\langle A\rangle\!\rangle\bigcirc\varphi$ iff there is $s_A \in \Sigma_A(P)$ with $M, \lambda[1] \models_P \varphi$ for all $\lambda \in out(q, s_A, P)$
$M, q \models_P \langle\!\langle A\rangle\!\rangle\Box\varphi$ iff there is $s_A \in \Sigma_A(P)$ such that $M, \lambda[i] \models_P \varphi$ for all $\lambda \in out(q, s_A, P)$ and all $i \in \mathbb{N}_0$
$M, q \models_P \langle\!\langle A\rangle\!\rangle\varphi\,\mathcal{U}\,\psi$ iff there is $s_A \in \Sigma_A(P)$ such that, for all $\lambda \in out(q, s_A, P)$, there is $i \in \mathbb{N}_0$ with $M, \lambda[i] \models_P \psi$, and $M, \lambda[j] \models_P \varphi$ for all $0 \le j < i$
$M, q \models_P \textbf{Pl}\,\varphi$ iff $M, q \models_\Upsilon \varphi$
$M, q \models_P \textbf{Ph}\,\varphi$ iff $(M, q) \models \varphi$

$M, q \models_P$ (**set-pl** $\omega)\varphi$ iff $M^\omega, q \models_P \varphi$ where the new model M^ω is equal to M but the new set Υ^ω of plausible strategy profiles is set to $[\![\omega]\!]_q$.

The "absolute" satisfaction relation $\models$ is given by $\models_\Sigma$. Note that an ordinary concurrent game structure (without plausibility) can be interpreted as a **CGSP** with all strategy profiles assumed plausible, i.e., with $\Upsilon = \Sigma$. In this way satisfaction of **ATLP** formulae can be extended to ordinary **CGS**.

Example 2 (Matching pennies ctd.). Suppose that it is plausible to expect that both agents are rational in the sense that they only play undominated strategies. Let $head_i$ denote the strategy of player i to play $head$ in q_0 and $wait$ elsewhere (analogously for $tail_i$). Then, $\Upsilon = \{(head_1, head_2), (head_1, tail_2)\}$. Under this assumption, agent 2 is free to grant himself with the prize or to refuse it: **Pl** $(\langle\langle 2\rangle\rangle \Diamond\, \mathsf{money}_2 \wedge \langle\langle 2\rangle\rangle \Box\, \neg\mathsf{money}_2)$. Still, he cannot choose to win without making the other player win too: **Pl** $\neg\langle\langle 2\rangle\rangle \Diamond(\mathsf{money}_2 \wedge \neg\mathsf{money}_1)$. Likewise, if rationality is defined via iterated undominated strategies, then we have $\Upsilon = \{(head_1, head_2)\}$, and therefore the outcome of the game is completely determined: **Pl** $\langle\langle\emptyset\rangle\rangle \Box\, (\neg\mathsf{start} \rightarrow \mathsf{money}_1 \wedge \mathsf{money}_2)$.

Note that, in order to include *both* notions of rationality in the model, we can encode them as denotations of two different plausibility terms – say, ω_{undom} and ω_{iter}, with $[\![\omega_{undom}]\!]_{q_0} = \{(head_1, head_2), (head_1, tail_2)\}$, and $[\![\omega_{iter}]\!]_{q_0} = \{(head_1, head_2)\}$. Let M_1' be model M_1 with plausibility terms and their denotation defined as above. Then, we have that $M_1', q_0 \models$ (**set-pl** ω_{undom})**Pl** $(\langle\langle 2\rangle\rangle \Diamond \mathsf{money}_2 \wedge \langle\langle 2\rangle\rangle \Box\, \neg\mathsf{money}_2) \wedge$ (**set-pl** ω_{iter})**Pl** $\langle\langle\emptyset\rangle\rangle \Box\, (\neg\mathsf{start} \rightarrow \mathsf{money}_1 \wedge \mathsf{money}_2)$.

3 How to Capture Rationality

It is easy to see that concurrent game *frames* (i.e., **CGS** without propositions and their valuation) generalize extensive game frames of perfect information (i.e., game trees without utilities)[3]. Thus, it is enough to "emulate" utilities (with e.g. special propositions) to obtain an embedding of extensive games in **CGS**. In Section 3.1, we present the construction proposed to this end in [3,21]. Having game trees represented as **ATL** models, we can use strategic formulae to characterize various rationality criteria. However, such formulae need to refer to strategies explicitly (which is not possible in **ATL**). **ATLI** ("**ATL** with intentions") is an extension of **ATL** that allows to assume that a particular strategy is intended for execution by a particular agent [21]. We briefly present the logic in Section 3.2; **ATLI** characterizations of several solution concepts (mostly after [32,21]) are presented in Section 3.3.

3.1 CGS and Extensive Games

In this paper we use game-theoretical concepts to describe the behavior of rational agents. For this purpose it is necessary to establish a precise correspondence between game trees and **CGS**. We only consider game trees in which the set of payoffs is finite.

[3] The class of concurrent game frames is strictly more general, as they can include cycles and simultaneous moves of players, which are absent in game trees.

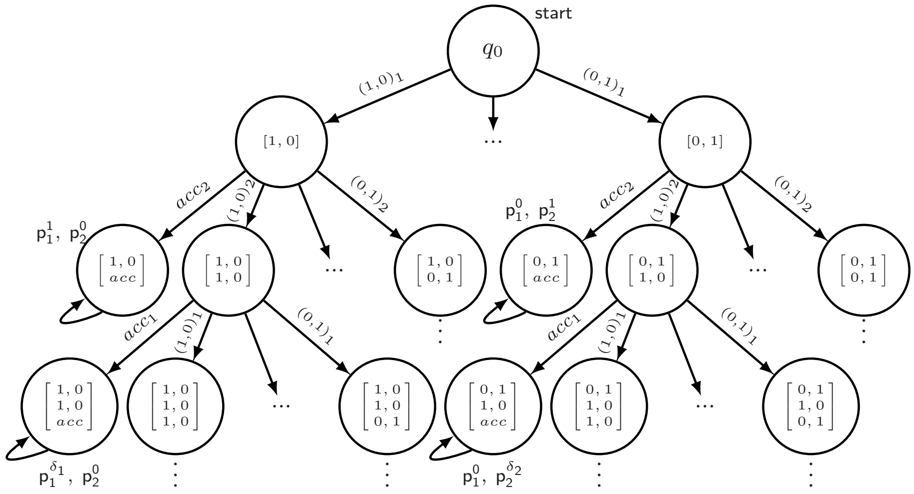

start
q_0
$(1,0)_1$
$(0,1)_1$
[1, 0]
...
[0, 1]
acc_2
$(1,0)_2$
$(0,1)_2$
$p_1^1,\ p_2^0$
$p_1^0,\ p_2^1$
$\begin{bmatrix}1,0\\acc\end{bmatrix}$
$\begin{bmatrix}1,0\\1,0\end{bmatrix}$
$\begin{bmatrix}1,0\\0,1\end{bmatrix}$
$\begin{bmatrix}0,1\\acc\end{bmatrix}$
$\begin{bmatrix}0,1\\1,0\end{bmatrix}$
$\begin{bmatrix}0,1\\0,1\end{bmatrix}$
acc_1
$(1,0)_1$
$(0,1)_1$
$\begin{bmatrix}1,0\\1,0\\acc\end{bmatrix}$
$\begin{bmatrix}1,0\\1,0\\1,0\end{bmatrix}$
$\begin{bmatrix}1,0\\1,0\\0,1\end{bmatrix}$
$\begin{bmatrix}0,1\\1,0\\acc\end{bmatrix}$
$\begin{bmatrix}0,1\\1,0\\1,0\end{bmatrix}$
$\begin{bmatrix}0,1\\1,0\\0,1\end{bmatrix}$
$p_1^{\delta_1},\ p_2^0$
$p_1^0,\ p_2^{\delta_2}$

Fig. 2. CGS M_2 for the bargaining game

We recall the construction proposed in [21] (and inspired by [3,32]). Let U denote the set of all possible utility values in a game; U will be finite and fixed for any given game. For each value $v \in U$ and agent $a \in \mathbb{A}\mathrm{gt}$, we introduce a proposition p_a^v into our set Π, and fix $\mathsf{p}_a^v \in \pi(q)$ iff a gets payoff of at least v in q. States in the model represent finite histories in the game. The correspondence between a traditional game tree Γ and a CGS $M(\Gamma)$ can be captured as follows. Let $\Gamma = \langle \mathcal{P}, \mathcal{A}, H, ow, u \rangle$, where $\mathcal{P}$ is a finite set of players, $\mathcal{A}$ a set of actions (moves), H a set of finite action sequences (game histories), and $ow(h)$ defines which player "owns" the next move after history h. Moves available at h are: $\mathcal{A}(h) = \{m \mid h \cdot m \in H\}$, and terminal positions are $Term = \{h \mid \mathcal{A}(h) = \emptyset\}$. Function $u : \Sigma \times Term \to U$ assigns agents' utilities to every terminal position of the game [26]. We say that $M(\Gamma) = \langle \mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o \rangle$ corresponds to Γ iff: (1) $\mathbb{A}\mathrm{gt} = \mathcal{P}$, (2) $Q = H$, (3) Π, π include propositions p_a^v to emulate utilities for terminal states in the way described above, (4) $Act = \mathcal{A} \cup \{nop\}$, (5) $d_a(q) = \mathcal{A}(q)$ if $a = ow(q)$ and $\{nop\}$ otherwise, (6) $o(q, nop, ..., m, ..., nop) = qm$, and (7) $o(q, nop, nop, ..., nop) = q$ for $q \in Term$. Note that, for every extensive form game Γ, there is a corresponding CGS, but the reverse is not true [21].

Example 3 (Bargaining). Consider bargaining with discount [26,27]. Two agents, a_1 and a_2, bargain about how to split goods worth initially $w_0 = 1$ EUR, with finite precision represented by a rounding function $r : \mathbb{R} \to \mathbb{R}$. After each round without agreement, the subjective worth of the goods reduces by *discount rates* δ_1 (for player a_1) and δ_2 (for player a_2). So, after t rounds, the goods are worth $\langle r(\delta_1^t), r(\delta_2^t) \rangle$, respectively. Subsequently, a_1 (if t is even) or a_2 (if t is odd) makes an offer to split the goods in proportions $\langle x, 1-x \rangle$, and the other player accepts or rejects it. If the offer is accepted, then a_1 takes $r(x\delta_1^t)$, and a_2 gets $r((1-x)\delta_2^t)$; otherwise the game continues. A **CGS** modeling this game is presented in Figure 2. Nodes represent various states of the negotiation process, and arcs show how agents' moves change the state of the game. Note that the **CGS** is the same as the original game tree from [26], save for the presence of propositions (instead of utility values) and loops added at the terminal nodes.

3.2 ATL with Intentions

ATLI [21] extends **ATL** with formulae $(\mathbf{str}_a\sigma_a)\varphi$ with the intuitive reading: "suppose that player a intends to play according to strategy σ_a, then φ holds". Thus, it allows to refer to agents' strategies explicitly via strategic terms $\sigma_a \in \mathfrak{Str}_a$. We assume that all $\mathfrak{Str}_a$ are pairwise disjoint. The set of all strategic terms is denoted by $\mathfrak{Str} = \bigcup_{a\in\mathbb{A}\mathrm{gt}} \mathfrak{Str}_a$.

Definition 7 ($\mathcal{L}_{ATLI}$ [21]). *The language $\mathcal{L}_{ATLI}(\mathbb{A}\mathrm{gt}, \Pi, \mathfrak{Str})$ is defined as follows:*

$$\theta ::= p \mid \neg\theta \mid \theta \wedge \theta \mid \langle\langle A\rangle\rangle \bigcirc \theta \mid \langle\langle A\rangle\rangle \Box\, \theta \mid \langle\langle A\rangle\rangle \theta\, \mathcal{U}\, \theta \mid (\mathbf{str}_a\sigma_a)\theta.$$

Models of **ATLI** $M = \langle \mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o, \mathcal{I}, \mathfrak{Str}, \|\cdot\|\rangle$ extend concurrent game structures with intention relations $\mathcal{I} \subseteq Q \times \mathbb{A}\mathrm{gt} \times Act$ (with $q\mathcal{I}_a\alpha$ meaning that a possibly intends to do action α when in q). Moreover, strategic terms are interpreted as strategies according to function $\|\cdot\| : \mathfrak{Str} \to \bigcup_{a\in\mathbb{A}\mathrm{gt}} \Sigma_a$ such that $\|\sigma_a\| \in \Sigma_a$ for $\sigma_a \in \mathfrak{Str}_a$. The set of paths consistent with all agents' intentions is defined as $\Lambda^{\mathcal{I}} = \{\lambda \in \Lambda_M \mid \forall i\, \exists\alpha \in d(\lambda[i])\, (o(\lambda[i], \alpha) = \lambda[i{+}1] \wedge \forall a \in \mathbb{A}\mathrm{gt}\; \lambda[i]\mathcal{I}_a\alpha_a)\}$. We say that strategy s_A is *consistent with A's intentions* if $q\mathcal{I}_a s_A[a](q)$ for all $q \in Q, a \in A$. The *intention-consistent outcome set* is defined as: $out^{\mathcal{I}}(q, s_A) = out(q, s_A) \cap \Lambda^{\mathcal{I}}$. The semantics of strategic operators in **ATLI** is given as follows:

$M, q \models \langle\langle A\rangle\rangle \bigcirc \theta$ iff there is a collective strategy s_A consistent with A's intentions, such that for every $\lambda \in out^{\mathcal{I}}(q, s_A)$, we have that $M, \lambda[1] \models \theta$;
$M, q \models \langle\langle A\rangle\rangle \Box\, \theta$ and $M, q \models \langle\langle A\rangle\rangle \theta\, \mathcal{U}\, \theta'$: analogous;
$M, q \models (\mathbf{str}_a\sigma)\theta$ iff $revise(M, a, \|\sigma\|), q \models \theta$;

Function $revise(M, a, s_a)$ updates model M by setting a's intention relation to $\mathcal{I}'_a = \{\langle q, s_a(q)\rangle \mid q \in Q\}$, so that s_a and $\mathcal{I}_a$ represent the same mapping in the resulting model. Note that a "pure" **CGS** M can be seen as a **CGS** with the "full" intention relation $\mathcal{I}^0 = \{\langle q, a, \alpha\rangle \mid q \in Q, a \in \mathbb{A}\mathrm{gt}, \alpha \in d_a(q)\}$. Additionally, for $A = \{a_1, ..., a_r\}$ and $\sigma_A = \langle\sigma_1, ..., \sigma_r\rangle$, we define: $(\mathbf{str}_A\sigma_A)\varphi \equiv (\mathbf{str}_{a_1}\sigma_1)...(\mathbf{str}_{a_r}\sigma_r)\varphi$.

3.3 Temporalized Solution Concepts

Players usually have a big repository of actions they can choose from. Analyzing games with no additional constraints is often infeasible or at least not very helpful since too many outcomes are possible. Our main focus is on "reasonable" outcomes that can only be obtained by "sensible" strategy profiles. Thus, we need a notion of rationality to reduce the space of possible moves, and "solve" a game. To this end, game theory proposes several solution concepts, e.g., Nash equilibrium and Pareto optimality.

Example 4 (Bargaining ctd.). Consider the bargaining game from Example 3. The game has an immense number of possible outcomes. Worse still, every strategy profile

$$s^x : \begin{cases} a_1 \text{ always offers } \langle x, 1-x\rangle\text{, and agrees to } \langle y, 1-y\rangle \text{ for } y \geq x \\ a_2 \text{ always offers } \langle x, 1-x\rangle\text{, and agrees to } \langle y, 1-y\rangle \text{ iff } 1-y \geq 1-x \end{cases}$$

is a Nash equilibrium; an agreement is reached in the first round. Thus, every split $\langle x, 1-x\rangle$ can be achieved through a Nash equilibrium; it seems that a stronger solution concept is needed. Indeed, the game has a unique *subgame perfect Nash equilibrium*. Because of the finite precision, there is a minimal round T with $r(\delta_i^{T+1}) = 0$ for $i = 1$ or $i = 2$. For simplicity, assume that $i = 2$ and agent a_1 is the offerer in T (i.e., T is even). Then, the only subgame perfect NE is given by the strategy profile s^κ with $\kappa = (1-\delta_2)\frac{1-(\delta_1\delta_2)^{\frac{T}{2}}}{1-\delta_1\delta_2} + (\delta_1\delta_2)^{\frac{T}{2}}$. The goods are split $\langle\kappa, 1-\kappa\rangle$; the agreement is reached in the first round[4].

With temporal logic, it is natural to define outcomes of strategies via properties of resulting paths rather than single states. Let $\sigma = \langle\sigma_1,\ldots,\sigma_k\rangle$. The notion of *temporal* T*-Nash equilibrium*, parametrized with a unary operator $T = \bigcirc, \Box, \Diamond, \_\,\mathcal{U}\,\psi, \psi\,\mathcal{U}\,\_$, was proposed in [21], with the following **ATLI** specification:

$$BR_a^T(\sigma) \equiv (\mathbf{str}_{\mathbb{A}\mathrm{gt}\setminus\{a\}}\sigma[\mathbb{A}\mathrm{gt}\setminus\{a\}]) \bigwedge_{v\in U} ((\langle\langle a\rangle\rangle T \mathsf{p}_\mathsf{a}^\mathsf{v}) \rightarrow (\mathbf{str}_a\sigma[a])\langle\langle\emptyset\rangle\rangle T\mathsf{p}_\mathsf{a}^\mathsf{v})$$

$$NE^T(\sigma) \equiv \bigwedge_{a\in\mathbb{A}\mathrm{gt}} BR_a^T(\sigma).$$

Thus, we have a family of equilibria now: $\bigcirc$-Nash equilibrium, $\Box$-Nash equilibrium etc., each corresponding to a different temporal pattern of utilities. For example, we may assume that agents "get v" if utility of at least v is guaranteed for every time moment ($\Box\,\mathsf{p}_\mathsf{a}^\mathsf{v}$), achieved eventually ($\Diamond\mathsf{p}_\mathsf{a}^\mathsf{v}$), and so on. The correspondence between Nash equilibria and temporal Nash equilibria for extensive games is captured by the following proposition.

Proposition 1 ([21]). *Let Γ be a game with a finite set of utilities. Then $M(\Gamma), \emptyset \models NE^\Diamond(\sigma)$ iff σ denotes a Nash equilibrium in Γ (i.e., $\|\sigma\|_{M(\Gamma)}$ is a NE in Γ)*[5].

It is easy to extend the above characterization to *subgame-perfect Nash equilibria*:

$$SPN^T(\sigma) \equiv \langle\langle\emptyset\rangle\rangle\Box\, NE^T(\sigma).$$

Proposition 2. *Let Γ be a game with a finite set of utilities. Then $M(\Gamma), \emptyset \models SPN^\Diamond(\sigma)$ iff σ denotes a subgame perfect Nash equilibrium in Γ.*

Proof. $M(\Gamma), \emptyset \models SPN^\Diamond(\sigma)$ iff $M(\Gamma), q \models NE^\Diamond(\sigma)$ for every q reachable from the root $\emptyset$ (*). However, Γ is a tree, so every node is reachable from $\emptyset$ in $M(\Gamma)$. So, by Proposition 1, (*) iff σ denotes a Nash equilibrium in every subtree of Γ.

We can use the above ATLI formulae to express game-theoretical properties of strategies in a straightforward way.

Example 5 (Bargaining ctd.). For the CGS in Figure 2, we have $M_2, q_0 \models NE^\Diamond(\sigma)$, with σ interpreted in M_2 as s^x (for any $x \in [0,1]$). Still, $M_2, q_0 \models SPN^\Diamond(\sigma)$ if, and only if, $\|\sigma\|_{M_2} = s^\kappa$.

[4] For the standard version of bargaining with discount (with the continuous set of payoffs $[0,1]$), cf. [26,27]. Restricting the payoffs to a finite set requires to alter the solution slightly [30,24].

[5] The empty history $\emptyset$ denotes the root of the game tree.

We now propose a tentative **ATLI** characterization of *Pareto optimality* (based on the characterization from [32] for strategic game forms):

$$PO^T(\sigma) \equiv \bigwedge_{v_1} \cdots \bigwedge_{v_k} \big((\langle\langle \mathbb{A}\mathrm{gt} \rangle\rangle T \bigwedge_{i} \mathsf{p}_\mathsf{i}^{\mathsf{v_i}}) \to (\mathbf{str}_{\mathbb{A}\mathrm{gt}}\sigma)\langle\langle \emptyset \rangle\rangle T(\bigwedge_{i} \mathsf{p}_\mathsf{i}^{\mathsf{v_i}} \vee \bigvee_{i} \bigvee_{\substack{v' \text{ st.} \\ v' > v_i}} \langle\langle \emptyset \rangle\rangle T \mathsf{p}_\mathsf{i}^{\mathsf{v}'}))$$

That is, the strategy profile denoted by σ is Pareto optimal iff, for every achievable pattern of payoff profiles, either it can be achieved by σ, or σ obtains a strictly better payoff pattern for at least one player. Note that the above formula has exponential length with respect to the number of payoffs in U. Moreover, it is not obvious that this characterization is the right one, as it refers in fact to the evolution of "payoff profiles" (i.e., combinations of payoffs achieved by agents at the same time), and not temporal patterns of payoff evolution for each agent separately. So, for example, $PO^\Diamond(\sigma)$ may hold even if there is a strategy profile σ' that makes each agent achieve eventually a better payoff, as long as not all of them will achieve these better payoffs at the same moment. Still, the following holds.

Proposition 3. *Let Γ be a game with a finite set of utilities. Then $M(\Gamma), \emptyset \models PO^\Diamond(\sigma)$ iff $\|\sigma\|_{M(\Gamma)}$ is Pareto optimal in Γ.*

Proof (sketch). Let $M(\Gamma), \emptyset \models PO^\Diamond(\sigma)$. Then, for every payoff profile $\langle v_1, \ldots, v_k\rangle$ reachable in Γ, we have that either $\|\sigma\|$ obtains at least as good a profile[6], or it obtains an incomparable payoff profile. Thus, $\|\sigma\|$ is Pareto optimal. The proof for the other direction is analogous.

Example 6 (Matching pennies ctd.). Let M_1'' be our "matching pennies" model M_1 with additional propositions $\mathsf{p}_\mathsf{i}^1 \equiv \mathsf{money_i}$. Then, we have $M_1'', q_0 \models PO^\Diamond(\sigma)$ iff σ denotes the strategy profile $\langle head_1, head_2\rangle$.

4 Flexible Plausibility Specifications

4.1 How to Impose Rationality

So far, we used abstract terms ω to describe plausibility sets in the framework of **ATLP**. However, such a solution does not realize the idea of "plugging in" arbitrary rationality criteria very well. In a way, it only shifts the problem to another level. True, we can reason about both arbitrary and rational behavior of agents (which *is* an advantage!), but the actual notion(s) of rationality must be "hard-wired" in the model through the denotation of plausibility terms $[\![\cdot]\!]$. Modeling an actual multi-agent system amounts therefore to crafting a concurrent game structure and a fixed number of "trimming filters" that trim off irrational options. Ultimately, this is not much different from hand-crafting a collection of separate game models: one for the unrestricted (arbitrary) play of agents, plus one for each possible limitation of agents' play (i.e., one model per plausibility term). Of course, not all of these models can be easily represented as **CGS** (since in **CGS** availability of actions for different agents can be defined only separately; the same holds for

[6] We recall that $\bigwedge_i \mathsf{p}_\mathsf{i}^{\mathsf{v_i}}$ means that each player i gets *at least* v_i.

defining availability of actions at different states). Also, **ATLP** allows for "switching" between these pre-wired model versions, which can be valuable in itself. Still, such a framework does not seem truly elegant and satisfying.

Ideally, one would like to have a *flexible language of terms* that would allow to specify any sensible rationality assumption, and then impose it on the system. Section 3.3 gives a hint how it can be done. Our idea is to use **ATLI** formulae θ to specify sets of plausible strategy profiles, with the presumed meaning that Υ collects exactly the profiles for which θ holds. Then, we can embed such **ATLI**-based plausibility specifications in formulae of **ATLP** in order to reason about rational agents. We call the resulting language **ATLP**$^{\textbf{[ATLI]}}$, and define it formally in Section 4.2.

4.2 ATLI-Based Plausibility Terms

Definition 8 **($\mathcal{L}_{\mathbf{ATLP}^{[ATLI]}}$).** *Let $\Omega^* = \{(\sigma.\theta) \mid \theta \in \mathcal{L}_{ATLI}(\mathbb{A}\mathrm{gt}, \Pi, \{\sigma[1], \ldots, \sigma[k]\})\}$. That is, Ω^* collects terms of the form $(\sigma.\theta)$, where θ is an **ATLI** formula including only references to individual agents' parts of the strategy profile σ. The language of **ATLP**$^{[ATLI]}$ is defined as $\mathcal{L}_{ATLP}(\mathbb{A}\mathrm{gt}, \Pi, \Omega^*)$.*

The idea behind terms of this form is simple. We have an **ATLI** formula θ, parameterized with a variable σ that ranges over the set of strategy profiles Σ. Now, we want $(\sigma.\theta)$ to denote exactly the set of profiles from Σ, for which formula θ holds. However – as σ denotes a strategy profile, and **ATLI** allows only to refer to strategies of individual agents – we need a way of addressing substrategies of σ in θ. This can be done by using **ATLI** terms $\sigma[i]$, which will be interpreted as the i's substrategy in σ. Below, we define the concept formally.

Definition 9 **(CGSP for $\mathcal{L}_{\mathbf{ATLP}^{[ATLI]}}$).** *Let $\langle \mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o\rangle$ be a **CGS**, and let $\Upsilon \subseteq \Sigma$ be a set of plausible strategy profiles. $M = \langle \mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o, \Upsilon, \Omega^*, [\![\cdot]\!]\rangle$ is a **CGS** with plausibility iff the denotation $[\![\cdot]\!]$ of terms from Ω^* is defined as follows.*

*First, we define a family of **ATLI** models $M^s{=}\langle \mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o, \mathcal{I}^0, \mathfrak{Str}, \|\cdot\|\rangle$, one for each strategy profile $s \in \Sigma$, with $\mathfrak{Str}_a = \{\sigma[a]\}$, and $\|\sigma[a]\| = s[a]$. Then, we define the plausibility mapping as:*

$$[\![\sigma.\theta]\!]_q = \{s \in \Sigma \mid M^s, q \models \theta\}.$$

For example, we may assume that a rational agent does not grant the other agents with too much control over his life: $(\sigma\ .\ \bigwedge_{a\in\mathbb{A}\mathrm{gt}}(\mathbf{str}_a\sigma[a])\neg\langle\!\langle \mathbb{A}\mathrm{gt} \setminus \{a\}\rangle\!\rangle\Diamond\mathsf{dead}_a)$. Note that games defined by **CGS** are, in general, not determined, so the above specification does not guarantee that each rational agent can efficiently protect his life. It only requires that he should behave cautiously so that his opponents do not have complete power to kill him.

It is now possible to plug in arbitrary **ATLI** specifications of rationality, and reason about their consequences.

Example 7 (Matching pennies ctd.). It seems that explicit quantification over the opponents' responses (lacking in **ATLI**) is essential to express undominatedness of strategies (cf. [32]). Still, we can at least assume that a rational player should avoid playing strategies that *guarantee* failure if a potentially successful strategy is available. Under this assumption, player 1 should never play $tail$, and in consequence player 2 controls the outcome of the game:

$$M_1, q_0 \models (\textbf{set-pl}\ \sigma. \textstyle\bigwedge_{a\in\mathbb{A}\mathrm{gt}} \langle\!\langle \mathbb{A}\mathrm{gt}\rangle\!\rangle \Diamond \mathsf{money_a} \to (\mathbf{str}_a\sigma[a])\langle\!\langle\mathbb{A}\mathrm{gt}\rangle\!\rangle\Diamond\mathsf{money_a}) \\ \mathbf{Pl}\,\big(\langle\!\langle 2\rangle\!\rangle\Diamond(\mathsf{money_1}\wedge\mathsf{money_2}) \wedge \langle\!\langle 2\rangle\!\rangle\Box\neg(\mathsf{money_1}\wedge\mathsf{money_2})\big).$$

Moreover, if only Pareto optimal strategy profiles can be played, then both players are bound to keep winning money (we recall that M_1'' is model M_1 extended with propositions $\mathsf{p}_\mathsf{i}^1 \equiv \mathsf{money_i}$):

$$M_1'', q_0 \models (\textbf{set-pl}\ \sigma.PO^{\Diamond}(\sigma))\ \mathbf{Pl}\,\langle\!\langle\emptyset\rangle\!\rangle\Box\,(\neg\mathsf{start} \to \mathsf{money_1}\wedge\mathsf{money_2}).$$

Finally, restricting plausible strategy profiles to Nash equlibria guarantees that player 2 should plausibly win, but the outcome of player 1 is not determined:

$$M_1'', q_0 \models (\textbf{set-pl}\ \sigma.NE^{\Diamond}(\sigma))\ \mathbf{Pl}\,\big(\langle\!\langle\emptyset\rangle\!\rangle\Box\,(\neg\mathsf{start} \to \mathsf{money_2}) \\ \wedge\neg\langle\!\langle\emptyset\rangle\!\rangle\Diamond\mathsf{money_1}\wedge\neg\langle\!\langle\emptyset\rangle\!\rangle\Box\neg\mathsf{money_1}\big).$$

Example 8 (Bargaining ctd.). For the bargaining agents and $\kappa = (1-\delta_2)\frac{1-(\delta_1\delta_2)^{\frac{T}{2}}}{1-\delta_1\delta_2} + (\delta_1\delta_2)^{\frac{T}{2}}$, we have accordingly:

1. $M_2, q_0 \models (\textbf{set-pl}\ \sigma.NE^{\Diamond}(\sigma))\mathbf{Pl}\,\langle\!\langle\emptyset\rangle\!\rangle\bigcirc(\mathsf{p}_1^{\mathsf{x}}\wedge\mathsf{p}_2^{1-\mathsf{x}})$ for every x;
2. $M_2, q_0 \models (\textbf{set-pl}\ \sigma.SPN^{\Diamond}(\sigma))\mathbf{Pl}\,\langle\!\langle\emptyset\rangle\!\rangle\bigcirc(\mathsf{p}_1^{\kappa}\wedge\mathsf{p}_2^{1-\kappa})$;
3. $M_2, q_0 \models (\textbf{set-pl}\ \sigma.SPN^{\Diamond}(\sigma))\mathbf{Pl}\,\langle\!\langle\emptyset\rangle\!\rangle\Box\,(\neg\mathsf{p}_1^{\mathsf{x}_1}\wedge\neg\mathsf{p}_2^{\mathsf{x}_2})$ for every $x_1 \neq \kappa$ and $x_2 \neq 1-\kappa$.

Thus, we can encode a game as a **CGS** M, specify rationality assumptions with an **ATLI** formula θ, and ask if a desired property φ of the system holds under these assumptions by model checking $(\textbf{set-pl}\ \sigma.\theta)\varphi$ in M. We report our results on the complexity of model checking **ATLP** in Section 5.

5 Model Checking Rational Play

In this section we show that model checking **ATLP** is $\mathbf{\Delta_3^P}$-complete, which seems in line with existing results on the complexity of solving games. It is well known that determining the existence of a solution concept instance with certain natural properties (e.g., a Nash equilibrium with expected utility of at least k, or a Pareto-optimal Nash equilibrium) is **NP**-hard even for normal form (i.e., one-step) games in the set of mixed strategies [10,7]. Similar results are known for extensive turn-based games with imperfect information and recall [9,22,5]. Formally, mixed strategies and imperfect information are absent in **ATLP**. However, the framework turns out to be quite powerful in terms of expressiveness. In particular, imperfect information strategies (sometimes called *uniform* strategies) can be characterized in **ATLP** for a relevant subclass of models, and checking strategic properties of systems in which *all* agents must play uniform strategies is $\mathbf{\Delta_3^P}$-complete – which renders **ATLP** model checking also $\mathbf{\Delta_3^P}$-complete.

This coincides with another result from game theory: if both players in a 2-player imperfect information game have imperfect recall, and chance moves are allowed, then the problem of finding a max-min pure strategy is $\mathbf{\Sigma_2^P}$-complete [22][7].

We mainly consider checking formulae of **ATLP** against "pure" concurrent game structures (i.e., we assume that plausibility assumptions will be specified explicitly in the formula), although we briefly show, too, that the results carry over to model checking against **CGS** with plausibility. The size of the input is measured with the number of transitions in the model (m) and the length of the formula (l). Note that the problem of checking **ATLP** with respect to the size of the *whole* **CGSP** (including the plausibility set Υ), is trivially linear in the size of the model – but the model size is exponential with respect to the number of states and transitions.

5.1 Model Checking ATLP Is in $\mathbf{\Delta_3^P}$

Model Checking ATLP with Plausibility Terms Expressed in ATLI. A detailed algorithm for model checking $\mathbf{ATLP^{[ATLI]}}$ formulae against concurrent game structures is presented in Figure 3. Apart from the model, the state, and the formula to be checked, the input includes *two* plausibility specifications (each represented by an **ATLI** formula and a state at which it should be evaluated). The first specification describes the current set of plausible strategy profiles Υ. The latter is the argument of the most recent ($\textbf{set-pl}\ \cdot$) operation, not necessarily incorporated into the definition of Υ yet – unless the **Pl** operator has been used since. As both **CTL** and **ATLI** model checking is linear in the number of transitions in the model and the length of the formula [6,21], we get the following.

Proposition 4. *$M, q \models \varphi$ iff $mcheck(M, q, \varphi, \top, q, \top, q)$. The algorithm runs in time $\mathbf{\Delta_3^P}$ with respect to the number of transitions in the model and the length of the formula.*

Model Checking ATLP with Arbitrary Plausibility Terms. The algorithm in Figure 3 uses the **ATLI**-based plausibility terms presented in Section 4.2. In the general case, we can think of any arbitrary implementation of terms in Ω. As long as $plausiblestrat$ (s, M, q, θ) can be computed in polynomial time, it does not affect the overall complexity of $mcheck$. In fact, it is enough to require that $plausiblestrat(s, M, q, \theta)$ can be computed in *nondeterministic* polynomial time, as the witness for $plausiblestrat$ can be guessed together with the strategy profile s in function $solve$, and with the strategy profile t in function $beatable$, respectively.

Proposition 5. *If the verification of plausibility ($plausiblestrat$) is in $\mathbf{NP}$, then the model checking algorithm ($mcheck$) is in $\mathbf{\Delta_3^P}$ with respect to m, l.*

Note that, if a list (or several alternative lists) of plausible strategy profiles is given explicitly in the model (via the plausibility set Υ and/or the denotations of abstract plausibility terms ω from Section 2.2), then the problem of guessing an appropriate

[7] Note that strategic operators can be nested in an **ATLP** formula, thus specifying a sequence of games, with the outcome of each game depending on the previous ones – and solving such games requires adaptive calls to a $\mathbf{\Sigma_2^P}$ oracle.

```
**function** $mcheck(M, q, \varphi, \theta_1, q_1, \theta_2, q_2)$;
Returns "true" iff $\varphi$ holds in $M, q$. The current plausibility assumptions are specified by the truth of the **ATLI** formula $\theta_1$ at state $q_1$. The most recent plausibility specification (not necessarily incorporated into the definition of the current plausibility set $\Upsilon$ yet) corresponds to the truth of $\theta_2$ at $q_2$.

**cases** $\varphi \equiv p, \varphi \equiv \neg\psi, \varphi \equiv \psi_1 \wedge \psi_2$ : proceed as usual;
**case** $\varphi \equiv (\textbf{set-pl } \sigma.\theta')\psi$ : return( $mcheck(M, q, \psi, \theta_1, q_1, \theta', q)$);
**case** $\varphi \equiv \mathbf{Pl}\,\psi$ : return( $mcheck(M, q, \psi, \theta_2, q_2, \theta_2, q_2)$);
**case** $\varphi \equiv \mathbf{Ph}\,\psi$ : return( $mcheck(M, q, \psi, \top, q_1, \theta_2, q_2)$);
**case** $\varphi \equiv \langle\langle A\rangle\rangle \bigcirc \psi$, **where** $\psi$ **includes some** $\langle\langle B\rangle\rangle$ : Label all $q' \in Q$, in which
    $mcheck(M, q, \psi, \theta_1, q_1, \theta_2, q_2)$ returns "true", with a new proposition yes. Return
    $mcheck(M, q, \langle\langle A\rangle\rangle \bigcirc \mathsf{yes}, \theta_1, q_1, \theta_2, q_2)$;
**case** $\varphi \equiv \langle\langle A\rangle\rangle \bigcirc \psi$, **where** $\psi$ **includes no** $\langle\langle B\rangle\rangle$ : Remove all operators $\mathbf{Pl}$, $\mathbf{Ph}$, (**set-pl** $\cdot$)
    from $\psi$ (they are irrelevant, as no cooperation modality comes further), yielding $\psi'$. Return
    $solve(M, q, \langle\langle A\rangle\rangle \bigcirc \psi', \theta_1, q_1)$;
**cases** $\langle\langle A\rangle\rangle \Box \psi$ **and** $\langle\langle A\rangle\rangle \psi_1 \,\mathcal{U}\, \psi_2$ : analogously ;
**end case**
```

```
**function** $solve(M, q, \varphi, \theta, q')$;
Returns "true" iff $\varphi$ holds in $M, q$ under plausibility assumptions specified by the truth of $\theta$ at $q'$. We assume that $\varphi \equiv \langle\langle A\rangle\rangle \Box \psi$, where $\psi$ is a propositional formula, i.e., it includes no $\langle\langle B\rangle\rangle$, **Pl**, **Ph**, (**set-pl** $\cdot$).

■ Label all $q' \in Q$, in which $\psi$ holds, with a new proposition yes;
■ Guess a strategy profile $s$;
■ **if** $plausiblestrat(s, M, q', \theta)$ **then** return( not $beatable(s[A], M, q, \langle\langle A\rangle\rangle \Box \mathsf{yes})$);
  **else** return( false);
```

```
**function** $beatable(s_A, M, q, \langle\langle A\rangle\rangle\gamma, q', \theta)$;
Returns "true" iff the opponents can beat $s_A$ so that it does not enforce $\gamma$ in $M, q$ under plausibility assumptions specified by the **ATLI** formula $\theta$ at $q'$. The path formula $\gamma$ is of the form $\bigcirc\psi, \Box\psi, \psi \,\mathcal{U}\, \psi'$ with propositional $\psi, \psi'$.

■ Guess a strategy profile $t$;
■ **if** $plausiblestrat(t, M, q', \theta)$ and $t[A] = s_A$ **then**
    – $M'$ := "trim" $M$, removing all transitions that cannot occur when $t$ is executed;
    – return( $mcheck_{CTL}(M', q, \mathsf{A}\gamma)$);
  **else** return( false);
```

```
**function** $plausiblestrat(s, M, q, \theta)$;
Checks if strategy profile $s$ satisfies formula $\theta$ in $M, q$.

■ return( $mcheck_{ATLI}(M^s, q, \theta)$);     // For $M^s$, cf. Definition 9
```

Fig. 3. Model checking **ATLP**

strategy from such a list is in **NP** (memoryless strategies have polynomial size with respect to m). As a consequence, we have the following:

Corollary 1. *Model checking **ATLP** (with both abstract and **ATLI**-based plausibility terms) against **CGSP** is in $\mathbf{\Delta_3^P}$ with respect to m, l.*

5.2 Model Checking ATLP Is $\mathbf{\Delta_3^P}$-Hard

We prove the $\mathbf{\Delta_3^P}$-hardness through a reduction of **SNSAT**$_2$, a typical $\mathbf{\Delta_3^P}$-complete variant of the Boolean satisfiability problem. The reduction follows in two steps. First, we define a modification of **ATL**$_{ir}$ [28], in which *all* agents are required to play only uniform strategies. We call it "uniform **ATL**$_{ir}$" (**ATL**$_{ir}^u$ in short), and show a polynomial reduction of **SNSAT**$_2$ to **ATL**$_{ir}^u$ model checking. Then, we point out how each formula and model of **ATL**$_{ir}^u$ can be equivalently translated (in polynomial time) to a **CGS** and a formula of **ATLP**$^{[\mathbf{ATLI}]}$, thus yielding a polynomial reduction of **SNSAT**$_2$ to **ATLP**$^{[\mathbf{ATLI}]}$. Again, we consider two cases: **ATLP** with arbitrary plausibility terms, and **ATLP** with terms defined through formulae of **ATLI**. The first part of the reduction (from **SNSAT**$_2$ to model checking **ATL**$_{ir}^u$) is the same in both cases, but the second part (from model checking **ATL**$_{ir}^u$ to **ATLP**) proceeds differently, and we discuss both variants accordingly.

Interested readers are referred to the technical report [16], where the construction is described in more detail.

Uniform ATL$_{ir}$. The semantics of **ATL**$_{ir}^u$ can be defined as follows. First, we define models as *concurrent epistemic game structures* (**CEGS**), i.e. **CGS** with epistemic relations $\sim_a \subseteq Q \times Q$, one per agent. (The intended meaning of $q \sim_a q'$ is that agent a cannot distinguish between between states q and q'.) Additionally, we require that agents have the same options in indistinguishable states, i.e., that $q \sim_a q'$ implies $d_a(q) = d_a(q')$. A (memoryless) strategy s_A is *uniform* if $q \sim_a q'$ implies $s_A^a(q) = s_A^a(q')$ for all $q, q' \in Q, a \in A$.

$M, q \models \langle\langle A\rangle\rangle_{ir}^u \bigcirc \varphi$ iff there is a uniform strategy s_A such that, for every uniform strategy $t_{\mathbb{A}\mathrm{gt}\setminus A}$, every $a \in A$, q' such that $q \sim_a q'$, and $\lambda \in out(\langle s_A, t_{\mathbb{A}\mathrm{gt}\setminus A}\rangle, q')$, we have $M, \lambda[1] \models \varphi$;

$\langle\langle A\rangle\rangle_{ir}^u \Box \varphi$, $\langle\langle A\rangle\rangle_{ir}^u \varphi \,\mathcal{U}\, \psi$: analogously.

Reduction of SNSAT$_2$ to Model Checking of ATL$_{ir}^u$. We recall the definition of **SNSAT**$_2$ after [23].

Definition 10 (SNSAT$_2$)
Input: *p sets of propositional variables $X_r = \{x_{1,r}, ..., x_{k,r}\}$, p sets of propositional variables $Y_r = \{y_{1,r}, ..., y_{k,r}\}$, p propositional variables z_r, and p Boolean formulae φ_r in positive normal form (i.e., negation is allowed only on the level of literals). Each φ_r involves only variables in $X_r \cup Y_r \cup \{z_1, ..., z_{r-1}\}$, with the following requirement: $z_r \equiv \exists X_r \forall Y_r . \varphi_r(z_1, ..., z_{r-1}, X_r, Y_r)$.*
Output: *The value of z_p.*

Note that every non-literal formula φ_r can be written as $\chi_1 \; op \; \chi_2$ with $op \in \{\wedge, \vee\}$. Recursively, χ_i can be written as $\chi_{i1} \; op_i \; \chi_{i2}$ and χ_{ij} as $\chi_{ij1} \; op_{ij} \; \chi_{ij2}$ etc.

Our reduction of **SNSAT**$_2$ is an extension of the reduction of **SNSAT** presented in [17]. That is, we construct the **CEGS** M_r corresponding to z_r with two players: *verifier* **v** and *refuter* **r**. The **CEGS** is turn-based, that is, every state is "governed" by a single player who determines the next transition. Each subformula $\chi_{i_1 \dots i_l}$ of φ_r has

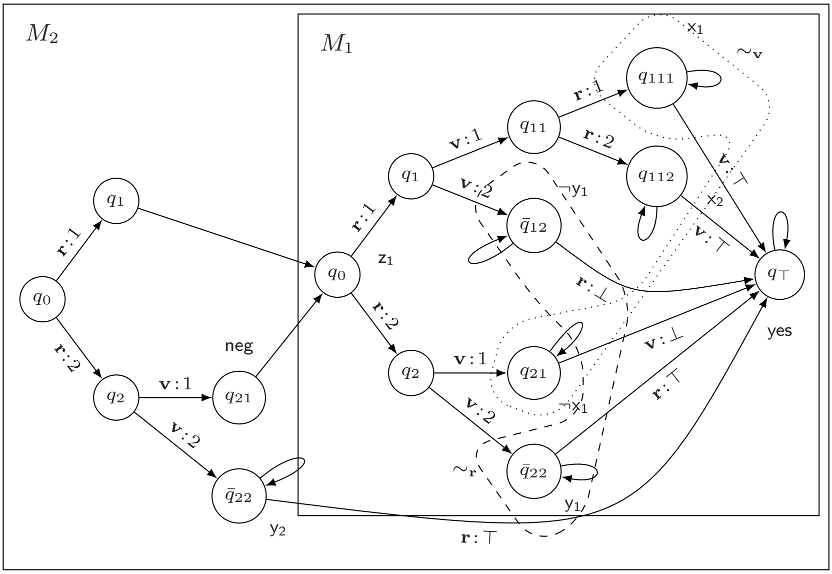


Fig. 4. CEGS M_2 for $\varphi_1 \equiv ((x_1 \wedge x_2) \vee \neg y_1) \wedge (\neg x_1 \vee y_1)$, $\varphi_2 \equiv z_1 \wedge (\neg z_1 \vee y_2)$

a corresponding state $q_{i_1 \dots i_l}$ in M_r. If the outermost logical connective of φ_r is $\wedge$, the refuter decides at q_0 which subformula χ_i of φ_r is to be satisfied, by proceeding to the "subformula" state q_i corresponding to χ_i. If the outermost connective is $\vee$, the verifier decides which subformula χ_i of φ_r will be attempted at q_0. This procedure is repeated until all subformulae are single literals. The states corresponding to literals are called "proposition" states.

The difference from the construction from [17] is that formulae are in positive normal form (rather than CNF) and that we have two kinds of "proposition" states now: $q_{i_1 \dots i_l}$ refers to a literal consisting of some $x \in X_r$ and is governed by $\mathbf{v}$; $\bar{q}_{i_1 \dots i_l}$ refers to some $y \in Y_r$ and will be governed by $\mathbf{r}$. Now, the values of the underlying propositional variables x, y are declared at the "propositional" states, and the outcome is computed. That is, if $\mathbf{v}$ executes $\top$ for a positive literal, i.e. $\chi_{i_1 \dots i_l} = x$, (or $\bot$ for $\chi_{i_1 \dots i_l} = \neg x$) at $q_{i_1 \dots i_l}$, then the system proceeds to the "winning" state $q_\top$; otherwise, the system goes to the "sink" state $q_\bot$. For states $\bar{q}_{i_1 \dots i_l}$ the procedure is analogous. Models corresponding to subsequent z_r are nested like in Figure 4[8]. "Proposition" states referring to the same variable x are indistinguishable for $\mathbf{v}$ (so that he has to declare the same value of x in all of them), and the states referring to the same y are indistinguishable for $\mathbf{r}$. A sole $\mathbf{ATL}^u_{ir}$ proposition yes holds only in the "winning" state $q_\top$. As in [17], we have the following result which concludes the reduction.

Proposition 6. *The above construction depicts a polynomial reduction of* ***SNSAT***$_2$ *to model checking* ***ATL***$^u_{ir}$ *in the following sense. Let*

[8] All states in the model for z_r are additionally indexed by r.

$$\Phi_1 \equiv \langle\!\langle \mathbf{v} \rangle\!\rangle^u_{ir} (\neg \mathsf{neg})\, \mathcal{U}\, \mathsf{yes}, \quad \textit{and}$$
$$\Phi_r \equiv \langle\!\langle \mathbf{v} \rangle\!\rangle^u_{ir} (\neg \mathsf{neg})\, \mathcal{U}\, (\mathsf{yes} \vee (\mathsf{neg} \wedge \langle\!\langle \varnothing \rangle\!\rangle^u_{ir} \bigcirc \neg \Phi_{r-1})) \quad \textit{for } r = 2, \ldots, p.$$

Then, we have z_p iff $M_p, q_0^p \models_{\mathbf{ATL}^u_{ir}} \Phi_p$.

From ATL$^u_{ir}$ to ATLP with Arbitrary Plausibility Terms. Now we show how **ATL**$^u_{ir}$ model checking can be reduced to model checking of **ATLP**. We are given a **CEGS** M, a state q in M, and an **ATL** $^u_{ir}$ formula φ. First, we sketch the reduction to model checking arbitrary **ATLP** formulae against **CGSP** (i.e., **CGS** with plausibility sets given explicitly in the model). Let Σ^u be the set of all uniform strategy profiles in M. We take **CGSP** M' as M (sans epistemic relations) extended with plausibility set $\Upsilon = \Sigma^u$. Then:

$$M, q \models_{\mathbf{ATL}^u_{ir}} \langle\!\langle A \rangle\!\rangle^u_{ir} \varphi \quad \text{iff} \quad M', q \models_{\mathbf{ATLP}} \mathbf{Pl}\, \langle\!\langle A \rangle\!\rangle \varphi,$$

which completes the reduction[9].

For model checking **ATLP** formulae with abstract terms ω against "pure" concurrent game structures, the reduction is similar. We take **CGS** M' as M minus epistemic relations, and plus a plausibility mapping $[\![\cdot]\!]$ such that $[\![\omega]\!]_q = \Sigma^u$. Then, again,

$$M, q \models_{\mathbf{ATL}^u_{ir}} \langle\!\langle A \rangle\!\rangle^u_{ir} \varphi \quad \text{iff} \quad M', q \models_{\mathbf{ATLP}} (\textbf{set-pl}\ \omega)\mathbf{Pl}\, \langle\!\langle A \rangle\!\rangle \varphi.$$ [10]

From ATL$^u_{ir}$ to ATLP with ATLI-Based Plausibility Terms. The reduction of **ATL**$^u_{ir}$ model checking to model checking of **ATLP**$^{[\mathbf{ATLI}]}$ against "pure" **CGS** is more sophisticated. We do not present a reduction for full model checking of **ATL**$^u_{ir}$; it is enough to show the reduction for the kind of models that we get in Section 5.2. We begin the reduction by reconstructing M_p to M'_p in which the last action profile is "remembered" in the final states. That is, the construction yields states of the form $\langle q, \alpha_1, \ldots, \alpha_k \rangle$, where $q \in \{q_\top, q_\bot\}$ is a final state of the original model M_p, and $\langle \alpha_1, \ldots, \alpha_k \rangle$ is the action profile executed just before the system proceeded to q. Each copy has the same valuation of propositions as the original state q, i.e., $\pi'(\langle q, \alpha_1, \ldots, \alpha_k \rangle) = \pi(q)$. Then, for each action $\alpha \in Act$ and agent $i \in \mathbb{A}\mathrm{gt}$, we add a new proposition $\mathsf{i} : \alpha$. Moreover, we fix the valuation of $\mathsf{i} : \alpha$ in M'_p so that it holds exactly in the final states that can be achieved by an action profile in which i executes α (i.e., states $\langle q, \alpha_1, ..., \alpha_i, ..., \alpha_k \rangle$). Note that the number of both states and transitions in M'_p is linear in the transitions of

[9] We note in passing that, technically, the size of the resulting model M' is not entirely polynomial. M' includes the plausibility set Υ, which is exponential in the number of states in M (since it is equal to the the set of all uniform strategy profiles in M). This is of course the case when we want to store Υ explicitly. However, checking if a strategy profile is uniform can be done in time linear wrt the number of states in M, so an *implicit* representation of Υ (e.g., the checking procedure itself) requires only linear space.

We do not discuss this issue in more depth, as we focus on the other variant of **ATLP** (with **ATLI**-based terms) in this paper.

[10] Cf. footnote 9.

M_p. The transformation produces model M'_p which is equivalent to M_p in the following sense. Let φ be a formula of $\mathbf{ATL}^u_{ir}$ that does not involve special propositions $\mathsf{i} : \alpha$. Then, for all $q \in Q$: $M_p, q \models_{\mathbf{ATL}^u_{ir}} \varphi$ iff $M'_p, q \models_{\mathbf{ATL}^u_{ir}} \varphi$.

In the next step, we will show that uniformity of a strategy can be characterized in **ATLI** *extended with epistemic operators* K_a. $K_a\varphi$ reads as "agent a knows that φ". The semantics of **ATLI+K** extends that of **ATLI** by adding the standard semantic clause from epistemic logic: $M, q \models K_a\varphi$ iff $M, q' \models \varphi$ for every q' such that $q \sim_a q'$. Let us now consider the following formula of **ATLI+Knowledge**:

$$\mathit{uniform}(\sigma) \equiv (\mathbf{str}\sigma)\langle\langle\emptyset\rangle\rangle\Box \bigwedge_{i\in\mathbb{A}\mathrm{gt}} \bigvee_{\alpha\in d(i,q)} K_i\langle\langle\emptyset\rangle\rangle\bigcirc \mathsf{i} : \alpha.$$

The reading of $\mathit{uniform}(\sigma)$ is: suppose that profile σ is played $(\mathbf{str}\sigma)$; then, for all reachable states $(\langle\langle\emptyset\rangle\rangle\Box)$, every agent has a single action $(\bigwedge_{i\in\mathbb{A}\mathrm{gt}} \bigvee_{\alpha\in d(i,q)})$ that is determined for execution $(\langle\langle\emptyset\rangle\rangle\bigcirc \mathsf{i} : \alpha)$ in every state indistinguishable from the current state (K_i). Thus, formula $\mathit{uniform}(\sigma)$ characterizes the *uniformity* of strategy profile σ. Formally, for every concurrent epistemic game structure M, we have that $M, q \models_{\mathbf{ATLI+K}} \mathit{uniform}(\sigma)$ iff $\|\sigma[a]\|$ is uniform for each agent $a \in \mathbb{A}\mathrm{gt}$ (for all states reachable from q). Of course, only reachable states matter when we look for strategies that should enforce a temporal goal.

To get rid of the epistemic operators from formula $\mathit{uniform}(\sigma)$ and epistemic relations from model M'_p, we use the construction from [14]. The construction yields a concurrent game structure $tr(M'_p)$ and an **ATLI** formula $tr(\mathit{uniform}(\sigma))$ with the following characteristics. For every **CEGS** M and $\mathbf{ATL}^u_{ir}$ formula φ, we have:
(1) $M, q \models_{\mathbf{ATL}^u_{ir}} \varphi$ iff $tr(M), q \models_{\mathbf{ATL}^u_{ir}} tr(\varphi)$;
(2) $M'_p, q \models_{\mathbf{ATLI+K}} \mathit{uniform}(\sigma)$ iff $tr(M'_p), q \models_{\mathbf{ATLI+K}} tr(\mathit{uniform}(\sigma))$.

This way, we obtain a reduction of $\mathbf{SNSAT}_2$ to model checking of $\mathbf{ATLP}^{[\mathbf{ATLI}]}$.

Proposition 7
z_p *iff* $tr(M'_p), q_0^p \models_{\boldsymbol{ATLP}^{[\boldsymbol{ATLI}]}} (\boldsymbol{set\text{-}pl}\ \sigma.tr(\mathit{uniform}(\sigma)))\mathbf{Pl}\, tr(\Phi_p)$.

Proof. We have z_p iff $M'_p, q_0^p \models_{\mathbf{ATL}^u_{ir}} \Phi_p$ iff $tr(M'_p), q_0^p \models_{\mathbf{ATL}^u_{ir}} tr(\Phi_p)$ iff $tr(M'_p), q_0^p \models_{\mathbf{ATLP}^{[\mathbf{ATLI}]}} (\mathbf{set\text{-}pl}\ \sigma.tr(\mathit{uniform}(\sigma)))\mathbf{Pl}\, tr(\Phi_p)$. ■

Theorem 1. *Model checking* ***ATLP*** *is* $\mathbf{\Delta_3^P}$*-complete with respect to the number of transitions in the model and the length of the formula.*

On the way, we have also proved that checking strategic abilities when *all* players are required to play uniformly is $\mathbf{\Delta_3^P}$-complete (that is, harder than ability against the worst line of events captured by $\mathbf{ATL}_{ir}$ formulae, which is "only" $\mathbf{\Delta_2^P}$-complete). We believe it is an interesting result with respect to verification of various kinds of agents' ability under incomplete information.

6 Conclusions

We propose a logic in which one can study the outcome of rational play in a logical framework, under various rationality criteria. To our knowledge, there has been very little work on this issue (although "solving" game-like scenarios with help of various solution concepts is arguably the main application of game theory). Note that we are *not* discussing the merits of this or that rationality criterion here, nor the pragmatics of using particular criteria to predict the actual behavior of agents. Our aim, most of all, is to propose a conceptual tool in which the consequences of accepting one or another criterion can be studied.

We believe that the logic we propose provides much flexibility and modeling power. The results presented in Section 5 also suggest that expressive power of the language is quite high. In terms of technical results, we prove that model checking **ATLP** is $\mathbf{\Delta_3^P}$-complete, and establish the model checking complexity of another interesting problem on the way.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time Temporal Logic. In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS), pp. 100–109. IEEE Computer Society Press, Los Alamitos (1997)
2. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time Temporal Logic. Journal of the ACM 49, 672–713 (2002)
3. Baltag, A.: A logic for suspicious players. Bulletin of Economic Research 54(1), 1–46 (2002)
4. Bulling, N., Jamroga, W.: Agents, beliefs and plausible behaviour in a temporal setting. In: Proceedings of AAMAS 2007, pp. 570–577 (2007)
5. Chu, F., Halpern, J.: On the NP-completeness of finding an optimal strategy in games with common payoffs. International Journal of Game Theory (2001)
6. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems 8(2), 244–263 (1986)
7. Conitzer, V., Sandholm, T.: Complexity results about Nash equilibria. Technical Report CMU-CS-02-135, School of Computer Science, Carnegie-Mellon University (2002)
8. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, vol. B, pp. 995–1072. Elsevier Science Publishers, Amsterdam (1990)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W. H. Freeman, San Francisco (1979)
10. Gilboa, I., Zemel, E.: Nash and correlated equilibria: Some complexity considerations. Games and Economic Behavior (1989)
11. Goranko, V., Jamroga, W.: Comparing semantics of logics for multi-agent systems. Synthese 139(2), 241–280 (2004)
12. Harrenstein, B.P., van der Hoek, W., Meyer, J.-J., Witteveen, C.: A modal characterization of Nash equilibrium. Fundamenta Informaticae 57(2–4), 281–321 (2003)
13. Harrenstein, P., van der Hoek, W., Meijer, J.-J., Witteveen, C.: Subgame-perfect Nash equilibria in dynamic logic. In: Pauly, M., Baltag, A. (eds.) Proceedings of the ILLC Workshop on Logic and Games. University of Amsterdam, Tech. Report PP-1999-25, pp. 29–30 (2002)
14. Jamroga, W.: Reducing knowledge operators in the context of model checking. Technical Report IfI-07-09, Clausthal University of Technology (2007)
15. Jamroga, W., Bulling, N.: A framework for reasoning about rational agents. In: Proceedings of AAMAS 2007, pp. 592–594 (2007)
16. Jamroga, W., Bulling, N.: Model checking rational play. Technical Report IfI-07-05, Clausthal University of Technology (2007)

17. Jamroga, W., Dix, J.: Model checking ATL$_{ir}$ is indeed Δ_2^P-complete. In: Proceedings of EUMAS 2006 (2006)
18. Jamroga, W., Dix, J.: Model checking abilities of agents: A closer look. Theory of Computing Systems (to appear, 2007)
19. Jamroga, W., Åtnes, T.: What agents can achieve under incomplete information. In: Proceedings of AAMAS 2006, pp. 232–234. ACM Press, New York (2006)
20. Jamroga, W., van der Hoek, W.: Agents that know how to play. Fundamenta Informaticae 63(2–3), 185–219 (2004)
21. Jamroga, W., van der Hoek, W., Wooldridge, M.: Intentions and strategies in game-like scenarios. In: Bento, C., Cardoso, A., Dias, G. (eds.) EPIA 2005. LNCS (LNAI), vol. 3808, pp. 512–523. Springer, Heidelberg (2005)
22. Koller, D., Megiddo, N.: The complexity of twoperson zero-sum games in extensive form. Games and Economic Behavior 4, 528–552 (1992)
23. Laroussinie, F., Markey, N., Schnoebelen, P.: Model checking CTL+ and FCTL is hard. In: Honsell, F., Miculan, M. (eds.) FOSSACS 2001. LNCS, vol. 2030, pp. 318–331. Springer, Heidelberg (2001)
24. Mas-Colell, A., Whinston, M.D., Green, J.R.: Microeconomic Theory. Oxford (1995)
25. Moses, Y., Tennenholz, M.: Artificial social systems. Computers and AI 14(6), 533–562 (1995)
26. Osborne, M., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
27. Sandholm, T.W.: Distributed rational decision making. In: Weiss, G. (ed.) Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, pp. 201–258. MIT Press, Cambridge (1999)
28. Schobbens, P.Y.: Alternating-time logic with imperfect recall. Electronic Notes in Theoretical Computer Science 85(2) (2004)
29. Shoham, Y., Tennenholz, M.: On the synthesis of useful social laws for artificial agent societies. In: Proceedings of AAAI 1992 (1992)
30. Ståhl, I.: Bargaining Theory. Stockholm School of Economics, Stockholm (1972)
31. van Benthem, J.: Rational dynamics and epistemic logic in games. In: Vannucci, S. (ed.) Logic, Game Theory and Social Choice III, pp. 19–23 (2003)
32. van der Hoek, W., Jamroga, W., Wooldridge, M.: A logic for strategic reasoning. In: Proceedings of AAMAS 2005, pp. 157–164 (2005)
33. van der Hoek, W., Roberts, M., Wooldridge, M.: Social laws in alternating time: Effectiveness, feasibility and synthesis. Synthese (2005)
34. van der Hoek, W., van Otterloo, S., Wooldridge, M.: Preferences in game logics. In: Proceedings of AAMAS 2004 (2004)
35. van der Hoek, W., Wooldridge, M.: Tractable multiagent planning for epistemic goals. In: Castelfranchi, C., Johnson, W.L. (eds.) Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2002), pp. 1167–1174. ACM Press, New York (2002)
36. van Otterloo, S., Jonker, G.: On Epistemic Temporal Strategic Logic. Electronic Notes in Theoretical Computer Science, XX, 35–45 (2004); Proceedings of LCMAS 2004
37. van Otterloo, S., Roy, O.: Verification of voting protocols. Working paper, University of Amsterdam (2005)

Formal Modelling of Emotions in BDI Agents

David Pereira[1], Eugénio Oliveira[2], and Nelma Moreira[1]

[1] DCC-FC & LIACC – University of Porto
Rua Campo Alegre, 1021/1055
4169-007 Porto, Portugal
{dpereira,nam}@ncc.up.pt
[2] DEEC-FE & LIACC – University of Porto
Rua Dr. Roberto Frias, s/n Lab. I 1212 - NIAD&R
4200-465 Porto, Portugal
eco@fe.up.pt

Abstract. Emotional-BDI agents are BDI agents whose behaviour is guided not only by beliefs, desires and intentions, but also by the role of emotions in reasoning and decision-making. The $\mathcal{E}_{\mathsf{BDI}}$ logic is a formal system for expressing the concepts of the Emotional-BDI model of agency. In this paper we present an improved version of the $\mathcal{E}_{\mathsf{BDI}}$ logic and show how it can be used to model the role of three emotions in Emotional-BDI agents: *fear*, *anxiety* and *self-confidence*. We also focus in the computational properties of $\mathcal{E}_{\mathsf{BDI}}$ which can lead to its use in automated proof systems.

1 Introduction

Emotional-BDI agents are BDI agents whose behaviour is guided not only by beliefs, desires and intentions, but also by the role of emotions in reasoning and decision-making. This conceptual model was developed by Pereira *et al.* [1] and a first version of the $\mathcal{E}_{\mathsf{BDI}}$ logic was presented in [2], where a first formalisation of *fear* was given. In this paper we present an improved version of the $\mathcal{E}_{\mathsf{BDI}}$ logic in order to model the role of three emotions in Emotional-BDI agents: *fear*, *anxiety* and *self-confidence*. The aim of this paper is to show how $\mathcal{E}_{\mathsf{BDI}}$ logic has enough expressivity to model some of the properties of these emotions, following Oliveira & Sarmento's model of emotional agent [3,4,5].

The main motivation for the current work was to provide a formal system in which the concepts of the Emotional-BDI model of agency could be logically expressed. Using these concepts we can specify distinct behaviours which are expected from agents under the influence of emotions. The existing formal systems for rational agency such as Rao & Georgeff's BDI logics [6,7] and Meyer's *et al.* **KARO** framework [8,9,10,11] do not allow a straightforward representation of emotions. However, both have properties which we can combine in order to properly model Emotional-BDI agents.

The $\mathcal{E}_{\mathsf{BDI}}$ logic is an extension of the $\mathsf{BDI}_{\mathsf{CTL}}$ logic, equipped with explicit reference to actions, capabilities and resources. The choice of $\mathsf{BDI}_{\mathsf{CTL}}$, and not the more

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 62–81, 2008.

powerful $\mathsf{BDI}_{\mathsf{CTL}^*}$, was motivated by our interest in automated proof methods that will allow the development of executable specification languages of rational agency or of formal verification systems for the Emotional-BDI model of agency.

This paper is organised as follows. In Section 2 we define the $\mathcal{E}_{\mathsf{BDI}}$ logic. This logic is based in $\mathsf{BDI}_{\mathsf{CTL}}$ logic and we begin by presenting the new operators that were added. Besides the syntax and semantics of $\mathcal{E}_{\mathsf{BDI}}$, we present the axiom systems for the new modal operators. We also establish the decidability of $\mathcal{E}_{\mathsf{BDI}}$-formulae, by transforming $\mathcal{E}_{\mathsf{BDI}}$-formulae into equivalent $\mathsf{BDI}_{\mathsf{CTL}}$ ones. In Section 3 we use the $\mathcal{E}_{\mathsf{BDI}}$-logic to define a set of conditions which are pre-requisites for defining how emotions are activated in Emotional-BDI agents and also special purpose actions which are executed by the agent when it ”feels” these emotions. In Section 4 we model the activation and effects of each of the emotions in Emotional-BDI agents using the previous conditions. In Section 5 we give some information about the ongoing implementation work on the decision procedures of $\mathcal{E}_{\mathsf{BDI}}$. In Section 6 some related work is considered. Finally, in Section 7 we present some conclusions about this work and point some topics for ongoing and future research in the $\mathcal{E}_{\mathsf{BDI}}$ logic.

2 The $\mathcal{E}_{\mathsf{BDI}}$ Logic

The $\mathcal{E}_{\mathsf{BDI}}$ is an extension of Rao & Georgeff's $\mathsf{BDI}_{\mathsf{CTL}}$. This extension adds new modal operators for representing the concepts of fundamental desires, capabilities, action execution and resources. The semantics of $\mathcal{E}_{\mathsf{BDI}}$ is therefore given by the satisfiability of $\mathcal{E}_{\mathsf{BDI}}$-formulae on extended $\mathsf{BDI}_{\mathsf{CTL}}$-models, considering accessibility-relations and functions for modelling the new operators.

2.1 Informal Description

The $\mathsf{BDI}_{\mathsf{CTL}}$ logic is a multi-modal logic which combines Emerson's *et al.* branching-time logic CTL [12] and modal operators for representing the mental states of belief (Bel), desire (Des) and intention (Int) as defined by Bratman *et al.* in [13]. The underlying model of $\mathsf{BDI}_{\mathsf{CTL}}$ has a two dimensional structure. One dimension is a set of possible worlds that correspond to the different perspectives of the agent representing his mental states. The other is a set of temporal states which describe the temporal evolution of the agent. A pair $\langle world, temporal\_state \rangle$ is called a *situation*. In the $\mathcal{E}_{\mathsf{BDI}}$ logic we added the following modal operators:

Fundamental desire: a fundamental desire is a desire which represents vital conditions to the agent, like its life or alike propositions. We model this concept using the modal operator Fund.

Actions: in $\mathcal{E}_{\mathsf{BDI}}$ we consider regular actions as defined in Propositional Dynamic Logic PDL [14]. In this way we can refer to the actions that the agent performs, in particular when he is under the influence of emotions. Given a finite set of atomic actions, regular actions are derived through the usual test operator ? and regular action operations (sequence, disjunction and Kleene closure).

Capabilities: a capability represents the operational structure of the execution of an action. This concept is similar to **KARO**'s *ability*. This is represented by the modal operator Cap.

Resources: resources are the means (physical or virtual) for engaging the execution of actions. For the modelling of resources we consider the operators:

- $\mathsf{Needs}(a, r)$: the atomic action a needs a unit of the resource r to be executed.
- $\mathsf{Available}^q(r)$: the agent has q units of the resource r, with $0 \leq q \leq MAX$, $MAX > 0$.
- $\mathsf{Saved}^q(r)$: the agent has q units of resource r saved for future usage.

We also consider the operator Res for representing the availability or not of all the resources needed to execute a regular action. We consider both $\mathsf{Available}$ and Saved operators for the following reasons: the $\mathsf{Available}$ operator provides only information about the available resources at the agent's current state of execution. No extra-information is given about the amount of resources available in the future. However, this last information is important for agent decision-making. If an agent considers that it would be inevitable to execute some action in the future, it may also consider necessary to protect the needed resources from being badly used and therefore not available when really needed. This "protection" of resources is given by the Saved operator. If a unit of a resource r is saved it cannot be used in other actions. It must first be freed and made available by the agent.

In terms of actions we consider three families of special purpose atomic actions, for the management of resource availability:

- $\texttt{get}(r)$: the agent gets one more unit of the resource r and this unit becomes available for being used.
- $\texttt{save}(r)$: the agent saves one unit of the resource r which was previously made available to the agent.
- $\texttt{free}(r)$: the agent frees one unit of the resource r which has been previously saved by the agent and makes it available to be used.

2.2 Syntax

As in $\mathsf{BDI}_{\mathsf{CTL}}$ we distinguish between *state formulae*, which are evaluated in a single situation, and *path formulae* which are evaluated along a path.

Definition 1. *Considering a non-empty set of propositional variables P, a finite set of atomic actions A_{At} that include the set of resource availability management actions, a finite set of resource symbols R and a set of resource quantities $\{0, \ldots, MAX\}$, with $MAX > 0$, the language of $\mathcal{E}_{\mathsf{BDI}}$-formulae is given by the following BNF-grammar:*

- *state-formulae:*
  $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\alpha\rangle\varphi \mid \mathsf{E}\psi \mid \mathsf{A}\psi \mid \mathsf{Bel}(\varphi) \mid \mathsf{Des}(\varphi) \mid \mathsf{Int}(\varphi) \mid$
  $\mathsf{Fund}(\varphi) \mid \mathsf{Needs}(a, r) \mid \mathsf{Available}^q(r) \mid \mathsf{Saved}^q(r) \mid \mathsf{Cap}(\alpha) \mid \mathsf{Res}(\alpha)$
  where $p \in P$, $a \in A_{\mathsf{At}}$, $r \in R$ and$0 \leq q \leq MAX$.

- *path-formulae:*
  $\psi ::= \mathsf{X}\varphi \,|\, (\varphi \mathsf{U} \varphi)$
- *regular actions (A_{Ra}):*
  $\alpha ::= id \,|\, a \in A_{\mathsf{At}} \,|\, \varphi? \,|\, \alpha;\alpha \,|\, \alpha + \alpha \,|\, \alpha^*$

In addition, we introduce the following abbreviations: $\top$, $\bot$, $\varphi \vee \psi$ and $\varphi \rightarrow \psi$ are abbreviations of $\neg(p \wedge \neg p)$ (with p being a fixed element of P), $\neg\top$, $\neg(\neg\varphi \wedge \neg\psi)$ and $\neg(\varphi \wedge \neg\psi)$, respectively; $\mathsf{AF}\varphi$, $\mathsf{EF}\varphi$, $\mathsf{AG}\varphi$ and $\mathsf{EG}\varphi$ are abbreviations of $\mathsf{A}(\top \mathsf{U} \varphi)$, $\mathsf{E}(\top \mathsf{U} \varphi)$, $\neg\mathsf{EF}\neg\varphi$ and $\neg\mathsf{AF}\neg\varphi$, respectively. The formula $[\alpha]\varphi$ stands for $\neg\langle\alpha\rangle\neg\varphi$. Iterated action α^n, with $n \geq 0$, are inductively defined by $\alpha^0 = id$ and $\alpha^{(n+1)} = \alpha;\alpha^n$. Informally, X means *next temporal state*, U *true until*, F *in a future temporal state*, G *globally true*. The path quantification modal operators E and A mean, respectively, *in one path* and *in all paths*. The regular action modal operator $\langle\alpha\rangle$ means *possibly true after a execution of α*.

2.3 Semantics

$\mathcal{E}_{\mathsf{BDI}}$-formulae are interpreted in extended $\mathsf{BDI}_{\mathsf{CTL}}$ models, called $\mathcal{E}_{\mathsf{BDI}}$-models. We follow Schild's approach to $\mathsf{BDI}_{\mathsf{CTL}}$ [15], by considering a *situation* as a pair $\delta = \langle w, s\rangle$, where s is a temporal state of the non-empty set T and w refers to a world (mental state perspective) from the non-empty set W.

Definition 2. *Given a non-empty set of situations Δ, a non-empty set of propositional variables P, a finite set of atomic actions A_{At}, a set of resource symbols R and a positive constant MAX, we define an $\mathcal{E}_{\mathsf{BDI}}$-model as a tuple:*

$$M = \langle \Delta, \mathcal{R}_T, \{\mathcal{R}_a : a \in A_{\mathsf{At}}\}, \mathcal{B}, \mathcal{D}, \mathcal{I}, \mathcal{F}, V, C, avl, svd, needs \rangle$$

such that:

- $\mathcal{R}_T \subseteq \Delta \times \Delta$ *is a temporal accessibility-relation, such that:*
  - *it is serial,* i.e., $\forall\delta \in \Delta$, $\exists\delta' \in \Delta$ *such that* $(\delta, \delta') \in \mathcal{R}_T$*;*
  - *if* $(\langle w_i, s_j\rangle, \langle w_k, s_l\rangle) \in \mathcal{R}_T$*, then* $w_i = w_k$*.*
- $\mathcal{R}_a \subseteq \mathcal{R}_T$ *is an atomic action accessibility-relation, with* $a \in A_{\mathsf{At}}$*;*
- $\mathcal{B}, \mathcal{D}, \mathcal{I}, \mathcal{F} \subseteq \Delta \times \Delta$ *are accessibility-relations for the mental state operators. These relations have the following property (considering* $\mathcal{O} \in \{\mathcal{B}, \mathcal{D}, \mathcal{I}, \mathcal{F}\}$*):*

$$\textit{if}\ (\langle w_i, s_j\rangle, \langle w_k, s_l\rangle) \in \mathcal{O}\ \textit{then}\ s_j = s_l;$$

- $V : P \rightarrow \wp(\Delta)$ *is a propositional variable labelling function;*
- $C : A_{\mathsf{At}} \rightarrow \wp(\Delta)$ *is a capability labelling function;*
- $needs : A_{\mathsf{At}} \rightarrow \wp(R)$ *is a function that defines which resource symbols in* R *are needed to execute each action of* A_{At}*;*
- $avl : \Delta \times R \rightarrow \{0, \ldots, MAX\}$ *is a function that for each situation defines which quantity of each resource is available;*
- $svd : \Delta \times R \rightarrow \{0, \ldots, MAX\}$ *is a function that for each situation defines which quantity of each resource is saved.*

As in $\mathsf{BDI}_{\mathsf{CTL}}$ path-formulae are evaluated along a path $\pi_\delta = (\delta_0, \delta_1, \delta_2, \ldots)$, such that $\delta = \delta_0$ and $\forall i \geq 0$, $(\delta_i, \delta_{i+1}) \in \mathcal{R}_T$. The k^{th} element of a path π_δ is denoted by $\pi_\delta[k]$.

The accessibility-relation and the capability labelling function for atomic actions are extended to regular actions α, as usual in PDL and **KARO**. We denote them, respectively, by R^A_α and c^A_α.

For the modelling of resources the functions *avl* and *svd* verify the following properties:

- the total amount of resources which the agent can deal with cannot be greater than MAX:
  $\forall \delta \in \Delta, \forall r \in R,\ 0 \leq avl(\delta, r) + svd(\delta, r) \leq MAX$.
- the execution of an atomic action consumes one unit of each resource needed for the execution of that action:
  $\forall r \in needs(a), \forall (\delta, \delta') \in \mathcal{R}_a,\ avl(\delta', r) = avl(\delta, r) - 1$.

Also, we assume that for the resource management atomic actions we have:

$$needs(\mathtt{get}(r)) = needs(\mathtt{save}(r)) = needs(\mathtt{free}(r)) = \emptyset,\ \forall r \in R$$

The availability of resources for executing regular actions is given by:

$$\begin{aligned}
res: &\quad A_{\mathsf{Ra}} \to \wp(\Delta) \\
res_a &= \begin{cases} \{\delta \mid \text{if } r \in needs(a) \text{ then } avl(r, \delta) \geq 1\}, \text{if} needs(a) \neq \emptyset \\ \\ \Delta, \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \\
res_{\varphi?} &= \Delta \\
res_{\alpha;\beta} &= \{\delta \mid \delta \in res_\alpha \wedge \exists (\delta, \delta') \in R^A_\alpha,\ \delta' \in res_\beta\} \\
res_{\alpha+\beta} &= res_\alpha \cup res_\beta \\
res_{\alpha^*} &= \cup_{n \geq 0}(res_{\alpha^n})
\end{aligned}$$

The intuition behind the value of the resource availability function *res* for α^* is that the iterated execution of α is bounded to the existence of a finite amount of resources. We are now in conditions to define the satisfiability for an $\mathcal{E}_{\mathsf{BDI}}$-formula.

Definition 3. *Let M be an $\mathcal{E}_{\mathsf{BDI}}$-model and δ a situation. The* satisfiability *of an $\mathcal{E}_{\mathsf{BDI}}$-formula is defined inductively as follows:*

- *state formulae satisfaction rules:*
  - $M, \delta \models p$ *iff* $\delta \in V(p)$
  - $M, \delta \models \neg\varphi$ *iff* $M, \delta \not\models \varphi$
  - $M, \delta \models \varphi \wedge \psi$ *iff* $M, \delta \models \varphi$ *e* $M, \delta \models \psi$
  - $M, \delta \models \mathsf{E}\psi$ *iff exists a path* π_δ *such that* $M, \pi_\delta \models \psi$
  - $M, \delta \models \mathsf{A}\psi$ *iff for all paths* π_δ, $M, \pi_\delta \models \psi$
  - $M, \delta \models \langle\alpha\rangle\varphi$ *iff exists* $(\delta, \delta') \in R^A_\alpha$ *such that* $M, \delta' \models \varphi$
  - $M, \delta \models \mathsf{Bel}(\varphi)$ *iff for all* $(\delta, \delta') \in \mathcal{B}$, $M, \delta' \models \varphi$
  - $M, \delta \models \mathsf{Des}(\varphi)$ *iff for all* $(\delta, \delta') \in \mathcal{D}$, $M, \delta' \models \varphi$

  - $M, \delta \models \mathsf{Int}(\varphi)$ *iff for all* $(\delta, \delta') \in \mathcal{I}$, $M, \delta' \models \varphi$
  - $M, \delta \models \mathsf{Fund}(\varphi)$ *iff for all* $(\delta, \delta') \in \mathcal{F}$, $M, \delta' \models \varphi$
  - $M, \delta \models \mathsf{Cap}(\alpha)$ *iff* $\delta \in c^A_\alpha$
  - $M, \delta \models \mathsf{Needs}(a, r)$ *iff* $r \in needs(a)$
  - $M, \delta \models \mathsf{Available}^q(r)$ *iff* $avl(\delta, r) = q$
  - $M, \delta \models \mathsf{Saved}^q(r)$ *iff* $svd(\delta, r) = q$
  - $M, \delta \models \mathsf{Res}(\alpha)$ *iff* $\delta \in res_\alpha$
- *path formulae satisfaction rules:*
  - $M, \pi_\delta \models \mathsf{X}\varphi$ *iff* $M, \pi_\delta[1] \models \varphi$
  - $M, \pi_\delta \models \varphi_1 \mathsf{U} \varphi_2$ *iff* $\exists\, k \geq 0$ *such that* $M, \pi_\delta[k] \models \varphi_2$ *and* $\forall j,\ 0 \leq j < k\,\big(M, \pi_\delta[j] \models \varphi_1\big)$

2.4 Properties of $\mathcal{E}_{\mathsf{BDI}}$

The axiomatic characterisation of $\mathcal{E}_{\mathsf{BDI}}$'s modal operators of time and BDI mental states are the same as in $\mathsf{BDI_{CTL}}$-logic. The modal operator Fund, for fundamental desires, follows the axiom set of Des and Int operators, which is the KD system [16], *i.e.*, $\mathcal{F}$ is a serial accessibility-relation. The Bel operator verifies the $KD45$ axioms, *i.e.*, $\mathcal{B}$ is an equivalence relation. The temporal operators follow the axioms of CTL and the action execution operators verify the axioms of PDL. Since both branching-time and regular action execution structures coexist, we have the following properties:

Theorem 1. *Let M be an $\mathcal{E}_{\mathsf{BDI}}$-model, a an atomic action and α a regular action. We have:*

1. *if* $M, \delta \models \langle a \rangle \varphi$ *then* $M, \delta \models \mathsf{EX}\varphi$, $a \neq id$.
2. *if* $M, \delta \models \langle \alpha \rangle \varphi$ *then* $M, \delta \models \mathsf{EF}\varphi$.
3. *if* $M, \delta \models \langle \alpha^* \rangle \varphi$ *then* $M, \delta \models \mathsf{E}(\langle \alpha \rangle \top \mathsf{U} \varphi)$.

Proof (sketch). In the first case, let $M, \delta \models \langle a \rangle \varphi$. Then it exists $(\delta, \delta') \in \mathcal{R}_a$ such that $M, \delta' \models \varphi$. By definition, $(\delta, \delta') \in \mathcal{R}_T$ and it exists $\pi_\delta = (\delta, \delta', \ldots)$ such that $M, \pi_\delta[1] \models \varphi$. Again, by definition, we have $M, \delta \models \mathsf{EX}\varphi$.

In the second case, we proceed by induction in the structure of α. The base case proves along the lines of the previous proof. For the induction step, we present here only the case of $\alpha = \beta + \gamma$ we have $M, \delta \models \langle \beta \rangle \varphi$ or $M, \delta \models \langle \gamma \rangle \varphi$. By the induction hypothesis we $M, \delta \models \mathsf{EF}\varphi$ or $M, \delta \models \mathsf{EF}\varphi$, which is equivalent $M, \delta \models \mathsf{EF}\varphi$. The other cases are proved analogously.

For the last case proceed again by induction on α. We only present the base case. Let $\alpha = a$. By definition it exists $n \geq 0$ such that $M, \delta \models \langle a^n \rangle \varphi$. Therefore it exists δ' such that $(\delta, \delta') \in R^A_{a^n}$ and $M, \delta' \models \varphi$. Considering now the path $\pi_\delta = (\pi_\delta[0], \pi_\delta[1], \ldots, \pi_\delta[n-1]).\pi'$ such that $\pi_\delta[n] = \pi'[0] = \delta'$. Since $\forall(\pi_\delta[i], \pi_\delta[i+1]) \in R^A_a$, for $0 \leq i \leq n-1$ by definition we have $M, \pi_\delta[i] \models \langle a \rangle \top$ for the same i and $M, \pi_\delta[n] \models \varphi$. By definition and considering the path π_δ we have $M, \delta \models \mathsf{E}(\langle a \rangle \top \mathsf{U} \varphi)$.

Capabilities are characterised similarly to *abilities* in the **KARO** framework. The axioms for the Cap modal operator are:

- $\mathsf{Cap}(\varphi?) \leftrightarrow \top$
- $\mathsf{Cap}(\alpha;\beta) \leftrightarrow \mathsf{Cap}(\alpha) \wedge \langle\alpha\rangle\mathsf{Cap}(\beta)$
- $\mathsf{Cap}(\alpha+\beta) \leftrightarrow \mathsf{Cap}(\alpha) \vee \mathsf{Cap}(\beta)$
- $\mathsf{Cap}(\alpha^*) \leftrightarrow \mathsf{Cap}(\alpha) \wedge \langle\alpha\rangle\mathsf{Cap}(\alpha^*)$
- $\mathsf{Cap}(\alpha) \wedge \langle\alpha^*\rangle(\mathsf{Cap}(\alpha) \rightarrow \langle\alpha\rangle\mathsf{Cap}(\alpha)) \rightarrow \mathsf{Cap}(\alpha^*)$

Resource availability for regular actions follows almost the same axioms that characterise the Cap operator. However, the unbounded composition operator * behaves differently, bounding the execution of an action α^* to a finite number of compositions of α. This composition stops when there are no resources to execute α once more. The Res operator verifies the following axioms:

- $\mathsf{Res}(\texttt{get}(r)) \leftrightarrow \top$
- $\mathsf{Res}(\texttt{save}(r)) \leftrightarrow \top$
- $\mathsf{Res}(\texttt{free}(r)) \leftrightarrow \top$
- $\mathsf{Res}(\varphi?) \leftrightarrow \top$
- $\mathsf{Res}(a) \leftrightarrow \bigwedge_{r\in R}\left(\mathsf{Needs}(a,r) \rightarrow \bigvee_{n=1}^{MAX} \mathsf{Available}^n(r)\right)$
- $\mathsf{Res}(\alpha;\beta) \leftrightarrow \mathsf{Res}(\alpha) \wedge \langle\alpha\rangle\mathsf{Res}(\beta)$
- $\mathsf{Res}(\alpha+\beta) \leftrightarrow \mathsf{Res}(\alpha) \vee \mathsf{Res}(\beta)$
- $\mathsf{Res}(\alpha^*) \leftrightarrow \mathsf{Res}(\alpha) \wedge \langle\alpha\rangle\mathsf{Res}(\alpha^*)$
- $\mathsf{Res}(\alpha^*) \wedge \langle\alpha^*\rangle(\mathsf{Res}(\alpha) \rightarrow \langle\alpha\rangle\mathsf{Res}(\alpha)) \rightarrow \mathsf{Res}(\alpha^*)$

Resources are also characterised by axioms which deal with the modal operators Available, Needs and Saved. First we define some abbreviations that represent, respectively, the maximum quantity of available and saved resources, in a situation:

- $\mathsf{MaxAvl}^q(r) =_{def} \mathsf{Available}^q(r) \wedge \neg\mathsf{Available}^{(q+1)}(r)$
- $\mathsf{MaxSvd}^q(r) =_{def} \mathsf{Saved}^q(r) \wedge \neg\mathsf{Saved}^{(q+1)}(r)$

The following axioms characterise the interaction between action execution and resource availability, considering $a \notin \{\texttt{get}(r), \texttt{save}(r), \texttt{free}(r) \,|\, r \in R\}$:

- $\mathsf{MaxAvl}^q(r) \wedge \mathsf{Needs}(a,r) \rightarrow [a]\mathsf{MaxAvl}^{(q-1)}(r)$, $0 < q \leq MAX$
- $\mathsf{MaxAvl}^q(r) \wedge \neg\mathsf{Needs}(a,r) \rightarrow [a]\mathsf{MaxAvl}^q(r)$, $0 \leq q \leq MAX$

The following axioms characterise the dynamics of the availability of resources, considering both resource availability limits and the execution of the special actions to manage them. We have:

- resource availability limits:
  - $\mathsf{Available}^0(r)$, $\forall r \in R$
  - $\mathsf{Saved}^0(r)$, $\forall r \in R$
  - $\mathsf{Available}^q(r) \rightarrow \mathsf{Available}^{(q-1)}(r)$, $1 < q \leq MAX$
  - $\mathsf{Saved}^q(r) \rightarrow \mathsf{Saved}^{(q-1)}(r)$, $1 < q \leq MAX$

- resource availability and resource management actions:
  - $\mathsf{Needs}(\mathtt{get}(r), r') \rightarrow \bot$, $\forall r, r' \in R$
  - $\mathsf{Needs}(\mathtt{save}(r), r') \rightarrow \bot$, $\forall r, r' \in R$
  - $\mathsf{Needs}(\mathtt{free}(r), r') \rightarrow \bot$, $\forall r, r' \in R$
  - $\mathsf{MaxAvl}^q(r) \rightarrow [\mathtt{get}(r)]\mathsf{MaxAvl}^{(q+1)}(r)$, for $0 \leq q < MAX$
  - $\mathsf{MaxAvl}^q(r) \wedge \mathsf{MaxSvd}^{q'}(r) \rightarrow [\mathtt{save}(r)](\mathsf{MaxAvl}^{(q-1)}(r) \wedge \mathsf{MaxSvd}^{(q'+1)}(r))$, with $0 \leq q + q' \leq MAX$
  - $\mathsf{MaxAvl}^q(r) \wedge \mathsf{MaxSvd}^{q'}(r) \rightarrow [\mathtt{free}(r)](\mathsf{MaxAvl}^{(q+1)}(r) \wedge \mathsf{MaxSvd}^{(q'-1)}(r))$, with $0 \leq q + q' \leq MAX$

2.5 Decidability

The decidability of $\mathcal{E}_{\mathsf{BDI}}$ is obtained by transforming an original $\mathcal{E}_{\mathsf{BDI}}$-formula φ into a new formula φ' which is evaluated in a modified $\mathcal{E}_{\mathsf{BDI}}$-model. This modified model is a $\mathsf{BDI}_{\mathsf{CTL}}$-model which considers the accessibility relation $\mathcal{F}$ and special propositional variables which represent the execution of atomic actions, capabilities and resource availability. Let $\mathcal{L}$ be an $\mathcal{E}_{\mathsf{BDI}}$ language and P the set of propositional variables. We define a new language $\mathcal{L}'$ equal to $\mathcal{L}$ except that it has a new set of propositions P' that is the union of the following disjunct sets:

- the set of propositional variables P,
- the set of propositional variables which represent the atomic actions: $\{done\_a \,|\, a \in A_{\mathsf{At}}\}$,
- the set of propositional variables which represent the capabilities for atomic actions: $\{cap\_a \,|\, a \in A_{\mathsf{At}}\}$,
- the set of propositional variables which represent the resources for atomic actions: $\{res\_a \,|\, a \in A_{\mathsf{At}}\}$,
- a set of propositional variables for representing the various quantities of resources available: $\{avl\_q\_r, svd\_q\_r \,|\, q \in \{0, \ldots, MAX\}, r \in R\}$,
- a set of propositional variables for representing the resources needed for the execution of each atomic action: $\{needs\_a\_r \,|\, a \in A_{\mathsf{At}}, r \in R\}$.

Considering an $\mathcal{E}_{\mathsf{BDI}}$-model M, the modified model M' is defined as follows, extending the propositional labelling function of M.

Definition 4. *Let M be an $\mathcal{E}_{\mathsf{BDI}}$-model such that:*

$$M = \langle \Delta, \mathcal{R}_T, \{\mathcal{R}_a : a \in A_{\mathsf{At}}\}, \mathcal{B}, \mathcal{D}, \mathcal{I}, \mathcal{F}, V, C, avl, svd, needs \rangle,$$

a model M' is a tuple:

$$M' = \langle \Delta, \mathcal{R}_T, \mathcal{B}, \mathcal{D}, \mathcal{I}, \mathcal{F}, V' \rangle,$$

such that $V' : P' \to \wp(\Delta)$ is defined as follows, where $a \in A_{\mathsf{at}}$, $p \in P$ and $r \in R$:

- $V'(p) = V(p)$,
- $V'(done\_a) = \{\delta' \mid (\delta, \delta') \in R_a^A\}$,
- $V'(cap\_a) = C(a)$,
- $V'(res\_a) = res_a$,
- $V'(avl\_q\_r) = \{\delta \mid M, \delta \models \mathsf{Available}^q(r)\}$,
- $V'(svd\_q\_r) = \{\delta \mid M, \delta \models \mathsf{Saved}^q(r)\}$,
- $V'(needs\_a\_r) = \{\delta \mid M, \delta \models \mathsf{Needs}(a, r)\}$.

Note that in V' only atomic actions are considered. Therefore, any $\mathcal{E}_{\mathsf{BDI}}$-formula must be normalised into an equivalent one where only atomic actions can occur.

Definition 5. *For all $\mathcal{E}_{\mathsf{BDI}}$-formula φ there exists a normalised formula $\varphi' \equiv \xi(\varphi)$, such that the normalisation ξ is inductively defined as follows:*

- *normalisation of regular action formulas:*

$$\begin{array}{ll}
\xi(\langle a \rangle \varphi) & = \langle a \rangle \xi(\varphi), \\
\xi(\langle \psi ? \rangle \varphi) & = \xi(\psi \wedge \varphi), \\
\xi(\langle \alpha \rangle (\varphi \vee \psi)) & = \xi(\langle \alpha \rangle \varphi) \vee \xi(\langle \alpha \rangle \psi), \\
\xi(\langle \alpha ; \beta \rangle \varphi) & = \xi(\langle \alpha \rangle \langle \beta \rangle \varphi), \\
\xi(\langle \alpha + \beta \rangle \varphi) & = \xi(\langle \alpha \rangle \varphi) \vee \xi(\langle \beta \rangle \varphi), \\
\xi(\langle id \rangle \varphi) & = \xi(\varphi), \\
\xi(\langle \alpha^{(n+1)} \rangle \varphi) & = \xi(\langle \alpha \rangle \langle \alpha^n \rangle \varphi), \\
\xi(\langle \alpha^* \rangle \varphi) & = \xi(\mathsf{E}(\langle \alpha \rangle \top \mathsf{U} \varphi)).
\end{array}$$

- *normalisation of capability formulas:*

$$\begin{array}{ll}
\xi(\mathsf{Cap}(a)) & = \mathsf{Cap}(a), \\
\xi(\mathsf{Cap}(\varphi ?)) & = \top, \\
\xi(\mathsf{Cap}(\alpha ; \beta)) & = \xi(\mathsf{Cap}(\alpha) \wedge \langle \alpha \rangle \mathsf{Cap}(\beta)), \\
\xi(\mathsf{Cap}(\alpha + \beta)) & = \xi(\mathsf{Cap}(\alpha)) \vee \xi(\mathsf{Cap}(\beta)), \\
\xi(\mathsf{Cap}(\alpha^*)) & = \xi(\mathsf{E}(\mathsf{Cap}(\alpha) \wedge \langle \alpha \rangle \mathsf{Cap}(\alpha)) \mathsf{U} \top)).
\end{array}$$

- *normalisation of resource formulas:*

$$\begin{array}{ll}
\xi(\mathsf{Needs}(a, r)) & = \mathsf{Needs}(a, r), \\
\xi(\mathsf{Available}^q(r)) & = \mathsf{Available}^q(r), \\
\xi(\mathsf{Saved}^q(r)) & = \mathsf{Saved}^q(r), \\
\xi(\mathsf{Res}(a)) & = \mathsf{Res}(a), \\
\xi(\mathsf{Res}(\varphi ?)) & = \top, \\
\xi(\mathsf{Res}(\alpha ; \beta)) & = \xi(\mathsf{Res}(\alpha) \wedge \langle \alpha \rangle \mathsf{Res}(\beta)), \\
\xi(\mathsf{Res}(\alpha + \beta)) & = \xi(\mathsf{Res}(\alpha)) \vee \xi(\mathsf{Res}(\beta)), \\
\xi(\mathsf{Res}(\alpha^*)) & = \xi(\mathsf{E}(\mathsf{Res}(\alpha) \wedge \langle \alpha \rangle \top)) \mathsf{U} \neg \mathsf{Res}(\alpha))).
\end{array}$$

- *normalisation of other formulas:*
  $\xi(\top) = \top,$
  $\xi(p) = p,$
  $\xi(\neg\varphi) = \neg(\xi(\varphi)),$
  $\xi(\varphi \wedge \psi) = \xi(\varphi) \wedge \xi(\psi),$
  $\xi(\mathsf{A}\psi) = \mathsf{A}(\xi(\psi)),$
  $\xi(\mathsf{E}\psi) = \mathsf{E}(\xi(\psi)),$
  $\xi(\mathsf{X}\varphi) = \mathsf{X}(\xi(\varphi)),$
  $\xi(\varphi_1 \mathsf{U} \varphi_2) = (\xi(\varphi_1)\mathsf{U}\xi(\varphi_2)),$
  $\xi(\mathsf{Bel}(\varphi)) = \mathsf{Bel}(\xi(\varphi)),$
  $\xi(\mathsf{Des}(\varphi)) = \mathsf{Des}(\xi(\varphi)),$
  $\xi(\mathsf{Int}(\varphi)) = \mathsf{Int}(\xi(\varphi)),$
  $\xi(\mathsf{Fund}(\varphi)) = \mathsf{Fund}(\xi(\varphi)),$

After normalisation, we apply the transformation defined below, so that the resulting formula can be evaluated in a model M'.

Definition 6. *Let φ be an normalised $\mathcal{E}_{\mathsf{BDI}}$-formula. The transformation of φ to φ' is given by τ, inductively defined as follows:*

- *propositional-formulae:*
  $\tau(\top) = \top,$
  $\tau(p) = p,$
  $\tau(\neg\varphi) = \neg(\tau(\varphi)),$
  $\tau(\varphi \wedge \psi) = \tau(\varphi) \wedge \tau(\psi).$
- *temporal-formulae:*
  $\tau(\mathsf{A}\psi) = \mathsf{A}(\tau(\varphi)),$
  $\tau(\mathsf{E}\psi) = \mathsf{E}(\tau(\varphi)),$
  $\tau(\mathsf{X}\varphi) = \mathsf{X}(\tau(\varphi)),$
  $\tau(\varphi_1 \mathsf{U} \varphi_2) = (\tau(\varphi_1)\mathsf{U}\tau(\varphi_2)).$
- *action execution formulae:*
  $\tau(\langle a \rangle \varphi) = \mathsf{EX}(done\_a \wedge \tau(\varphi)),$
  $\tau([a]\varphi) = \mathsf{AX}(done\_a \rightarrow \tau(\varphi)).$
- *mental-state formulae:*
  $\tau(\mathsf{Bel}(\varphi)) = \mathsf{Bel}(\tau(\varphi)),$
  $\tau(\mathsf{Des}(\varphi)) = \mathsf{Des}(\tau(\varphi)),$
  $\tau(\mathsf{Int}(\varphi)) = \mathsf{Int}(\tau(\varphi)),$
  $\tau(\mathsf{Fund}(\varphi)) = \mathsf{Fund}(\tau(\varphi)),$
- *capabilities and resources formulae:*
  $\tau(\mathsf{Cap}(a)) = cap\_a,$
  $\tau(\mathsf{Res}(a)) = res\_a,$
  $\tau(\mathsf{Needs}(a, r)) = needs\_a\_r,$
  $\tau(\mathsf{Available}^q(r)) = \bigwedge_{0 \le s \le q}(avl\_s\_r),$
  $\tau(\mathsf{Saved}^q(r)) = \bigwedge_{0 \le s \le q}(svd\_s\_r).$

Now we can present the following theorem.

Theorem 2. *Let M be an $\mathcal{E}_{\mathsf{BDI}}$-model, δ a situation and φ a normalised $\mathcal{E}_{\mathsf{BDI}}$-formula. If $M, \delta \models \varphi$ then $M', \delta \models \tau(\varphi)$.*

Proof. (Sketch). Proof is done by induction of the structure of φ. We present here only the case $\varphi = \mathsf{Cap}(a)$. Assume that $M, \delta \models \mathsf{Cap}(a)$. By definition, $\delta \in C(a)$. By definition of M' we get $\delta \in V'(cap\_a)$ which is equivalent to $M', \delta \models cap\_a$. Since $\tau(\mathsf{Cap}(a)) = cap\_a$ we get $M', \delta \models \tau(\mathsf{Cap}(a))$.

Using this theorem, we obtain the decidability of a $\mathcal{E}_{\mathsf{BDI}}$-formula φ by transforming it into $\tau(\xi(\varphi))$ and applying to the latter the tableau construction for $\mathsf{BDI}_{\mathsf{CTL}}$, with a rule for expanding formulas containing the Fund modal operator. The algorithm for building such tableau is based on the decision procedures for $\mathsf{BDI}_{\mathsf{CTL}}$, developed by Rao & Georgeff in [6]. In particular we use the notions of *fully extended propositional tableau* and of a *fragment $DAG[w, \varphi]$* as defined in the cited work of Rao & Georgeff.

Definition 7. *Let ψ be a $\mathcal{E}_{\mathsf{BDI}}$-formula and $\tau(\xi(\psi)) = \varphi$. The tableau construction for $\mathcal{E}_{\mathsf{BDI}}$ is defined as follows:*

1. *build a tree with just one node w_0, called root, such that $L(w_0) = \{\varphi\}$.*
2. *repeat $(a) - (d)$ until none apply:*
   - *(a)* build a propositional tableau: *if w is a leaf, $L(w)$ is not inconsistent, $L(w)$ is not a propositional tableau and ψ is the smaller witness of this fact, then:*
     - *i. if ψ is $\neg\neg\gamma$, create a node w', son of w, such that $L(w') = L(w) \cup \{\gamma\}$,*
     - *ii. if ψ is $\gamma \wedge \theta$, create a node w', son of w, such that $L(w') = L(w) \cup \{\gamma, \theta\}$,*
     - *iii. if ψ is $\neg(\gamma \wedge \theta)$, create two nodes w' and w'', sons of w, such that $L(w') = L(w) \cup \{\neg\gamma\}$ e $L(w'') = L(w) \cup \{\neg\theta\}$.*
   - *(b)* build a fully extended propositional tableau: *if w is a leaf, $L(w)$ is not inconsistent, $L(w)$ is not a fully extended propositional tableau and ψ is a witness of this fact, then create two nodes w' e w'', sons of w, such that $L(w') = L(w) \cup \{\psi\}$ e $L(w'') = L(w) \cup \{\neg\psi\}$,*
   - *(c)* extend CTL-formulas: *if w is a leaf, $L(w)$ is not inconsistent, $L(w)$ is a fully extended propositional tableau and contains the formulas $\mathsf{AX}\varphi_1, \ldots, \mathsf{AX}\varphi_n, \mathsf{EX}\psi_1, \ldots, \mathsf{EX}\psi_m$, then create m successors i, each containing the set $\{\varphi_1, \ldots, \varphi_n, \psi_i\}$,*
   - *(d)* create mental states operator successors: *if w is a leaf, $L(w)$ is not inconsistent and $L(w)$ is a fully extended propositional tableau, then:*
     - *i. if $L(w)$ contains $\neg\mathsf{Bel}(\varphi_1), \ldots, \neg\mathsf{Bel}(\varphi_n), \mathsf{Bel}(\psi_1), \ldots, \mathsf{Bel}(\psi_m)$, then create n $\mathcal{B}$-successors w_i, each containing $\{\neg\varphi_i, \psi_1, \ldots, \psi_m\}$;*
     - *ii. if $L(w)$ contains $\neg\mathsf{Des}(\varphi_1), \ldots, \neg\mathsf{Des}(\varphi_n), \mathsf{Des}(\psi_1), \ldots, \mathsf{Des}(\psi_m)$, then create n $\mathcal{D}$-successors w_i, each containing $\{\neg\varphi_i, \psi_1, \ldots, \psi_m\}$;*
     - *iii. if $L(w)$ contains $\neg\mathsf{Int}(\varphi_1), \ldots, \neg\mathsf{Int}(\varphi_n), \mathsf{Int}(\psi_1), \ldots, \mathsf{Int}(\psi_m)$, then create n $\mathcal{I}$-successors w_i, each containing $\{\neg\varphi_i, \psi_1, \ldots, \psi_m\}$;*
     - *iv. if $L(w)$ contains $\neg\mathsf{Fund}(\varphi_1), \ldots, \neg\mathsf{Fund}(\varphi_n), \mathsf{Fund}(\psi_1), \ldots, \mathsf{Fund}(\psi_m)$, then create n $\mathcal{F}$-successors w_i, each containing $\{\neg\varphi_i, \psi_1, \ldots, \psi_m\}$;*

(e) mark nodes as "satisfiable": *if w is not marked as "satisfiable", then mark it as so if:*
   i. *$L(w)$ is not a fully extended* CTL *tableau and exists a successor w' of w which is marked as "satisfiable";*
   ii. *$L(w)$ is a fully extended* CTL *tableau and all formulas* $\mathsf{AX}\varphi$ *and* $\mathsf{EX}\varphi$ *are satisfied (through the existence of a fragment $DAG[w, \varphi]$) and all the $\mathcal{B}$,$\mathcal{D}$,$\mathcal{I}$,$\mathcal{F}$-successors are marked as "satisfiable",*
   iii. *$L(w)$ is a fully extended* CTL *tableau and don't exist formulas of the type* $\mathsf{AX}\varphi$*, nor the type* $\mathsf{EX}\varphi$*, nor the type* $\neg\mathsf{Bel}(\varphi)$*, nor the type* $\neg\mathsf{Des}(\varphi)$*, nor the type* $\neg\mathsf{Int}(\varphi)$ *nor the type* $\neg\mathsf{Fund}(\varphi)$*, and $L(W)$ is not inconsistent.*

3. if the root of the tableau is marked as "satisfiable" then return "φ is satisfiable". Otherwise return "φ is not satisfiable".

Extending the work of Rao & Georgeff [6], we have the decidability of $\mathcal{E}_{\mathsf{BDI}}$:

Theorem 3. *The $\mathcal{E}_{\mathsf{BDI}}$ logic is decidable.*

Proof. The extension of the syntax and semantics of $\mathsf{BDI}_{\mathsf{CTL}}$ to support the Fund operator is similar to the proof of the decidability of the modal operators of Des and Int in [6].

3 Preliminaries for Modelling Emotions in $\mathcal{E}_{\mathsf{BDI}}$

In this section we present a series of concepts which will be useful for modelling emotions in $\mathcal{E}_{\mathsf{BDI}}$. These concepts refer to conditions that are the basis for modelling the activation of emotions and the consequences that these emotions have in the behaviour of the agent.

3.1 Resource Management Actions

We begin by defining special regular actions for dealing with resource management. For that we consider the following abbreviations for regular actions:

- $\mathtt{If}(\varphi, \alpha) =_{def} (\varphi?; \alpha)$
- $\mathtt{IfE}(\varphi, \alpha, \beta) =_{def} \mathtt{If}(\varphi, \alpha) + \mathtt{If}(\neg\varphi, \beta)$
- $\mathtt{WhileDo}(\varphi, \alpha) =_{def} ((\varphi?; \alpha)^*); \neg\varphi?$

We also consider a special function which, given a finite set of regular actions S, returns the composition of all the actions in S, in some order (in this function we consider that regular actions commute). This function, which we denominate by *eval_set*, is inductively defined as:

$$\begin{aligned} &eval\_set &&: \wp(A_{\mathsf{Ra}}) \rightarrow A_{\mathsf{Ra}} \\ &eval\_set(\emptyset) &&= id \\ &eval\_set(\{\alpha\} \cup S) &&= \alpha; eval\_set(S),\ \alpha \notin S \end{aligned}$$

Based on the atomic actions for the of resource management, we define the following set of resource management regular actions:

GET: the agent gets all the resources needed to execute some atomic action. Considering:
$Cond_1(a, r) = \mathsf{Needs}(a, r) \wedge \mathsf{MaxAvl}^0(r)$
we have:
$\mathtt{GET}(a) = eval\_set(\{\mathtt{If}(Cond_1(a, r), \mathtt{get}(r)) \,|\, r \in R\})$

SAVE: the agent saves a unit of each resource needed to execute an atomic action. Considering:
$Cond_2(a, r) = \mathsf{Needs}(a, r) \wedge \mathsf{MaxSvd}^0(r)$
we have:
$\mathtt{SAVE}(a) = eval\_set(\{\mathtt{If}(Cond_2(a, r), \mathtt{IfE}(\mathsf{Avl}(r), \mathtt{save}(r), \mathtt{get}(r); \mathtt{save}(r))) \,|\, r \in R\})$

FREE: the agent frees the resources previously saved for executing an atomic action. Considering:
$Cond_3(a, r) =_{def} \mathsf{Needs}(a, r) \wedge \mathsf{Saved}^1(r)$
we have:
$\mathtt{FREE}(a) = eval\_set(\{\mathtt{If}(Cond_3(a, r), \mathtt{free}(r)) \,|\, r \in R\})$

All these definition scale for regular actions $\alpha \in A_{\mathsf{Ra}}$ and we can work with for instance $\mathtt{FREE}(\alpha)$ instead of $\mathtt{FREE}(a)$.

3.2 Proposition Achievement

For the agent to succeed in the execution of an action it must have both the capability and resources for that action. We denote the existence of both of them as *effective capability*. Formally we have:

- $\mathsf{EffCap}(\alpha) =_{def} \mathsf{Cap}(\alpha) \wedge \mathsf{Res}(\alpha)$

The agent also considers if it *can* or *cannot* execute some action to achieve the truth of some proposition. Formally we have:

- $\mathsf{Can}(\alpha, \varphi) =_{def} \mathsf{Bel}(\langle\alpha\rangle\varphi \wedge \mathsf{EffCap}(\alpha))$
- $\mathsf{Cannot}(\alpha, \varphi) =_{def} \mathsf{Bel}(\neg\langle\alpha\rangle\varphi \vee \neg\mathsf{EffCap}(\alpha))$

3.3 Risk and Favourable Conditions

The activation of emotions is based on conditions of the environment that show to be positive or negative to the desires and fundamental desires of the agent. First we define the following conditions:

Risk condition: a proposition φ is said to be *at risk* if there is a next situation in which $\neg\varphi$ is true:
$\mathsf{AtRisk}(\varphi) =_{def} \mathsf{EX}(\neg\varphi)$

Possibly at risk: a proposition φ is said to be *possibly at risk* if there exists a future situation where $\neg\varphi$ is true. Formally this is defined as:
$\mathsf{PossAtRisk}(\varphi) =_{def} \mathsf{EF}(\neg\varphi)$

Safe: a proposition φ is said to be *safe* if it will always be true in the future. Formally we have:
$\mathsf{Safe}(\varphi) =_{def} \mathsf{AF}(\varphi)$

On believing on the above, and the propositions being either fundamental desires or only desires, the agent distinguishes between three types of conditions for activating emotions:

1. *Threats:* a *threat* is a condition of the environment in which a fundamental desire is in imminent risk of failure. We consider the following kinds of threats:
   - a fundamental desire φ is said to be *threatened* if the agent believes that φ is at risk:
     $\mathsf{Threatened}(\varphi) =_{def} \mathsf{Bel}(\mathsf{AtRisk}(\varphi)) \wedge \mathsf{Fund}(\varphi)$
   - a fundamental desire φ is said to be *threatened by a proposition* ψ if the agent believes that the truth of ψ implies φ being at risk:
     $\mathsf{ThreatProp}(\psi, \varphi) =_{def} \mathsf{Bel}(\psi \rightarrow \mathsf{AtRisk}(\varphi)) \wedge \mathsf{Fund}(\varphi)$
   - a fundamental desire φ is said to be *threatened by the execution of an action* a if the agent believes that the successful execution of a will put φ at risk:
     $\mathsf{ThreatAct}(a, \varphi) =_{def} \mathsf{Bel}(\langle a \rangle \mathsf{AtRisk}(\varphi)) \wedge \mathsf{Fund}(\varphi)$ $\mathsf{ThreatsEffC}(a, \varphi) =_{def} \mathsf{Bel}(\neg \mathsf{EffCap}(a) \rightarrow \mathsf{AtRisk}(\langle a \rangle \varphi)) \wedge \mathsf{Fund}(\varphi)$

2. *Not favourable:* a condition is *not favourable* if it reveals a possible failure of one of the agent's desires, in the future. As in the case of the threats, we consider the following kinds of not favourable conditions:
   - $\mathsf{NotFavourable}(\varphi) =_{def} \mathsf{Bel}(\mathsf{PossAtRisk}(\varphi)) \wedge \mathsf{Des}(\varphi)$
   - $\mathsf{NotFavourableProp}(\psi, \varphi) =_{def} \mathsf{Bel}(\psi \rightarrow \mathsf{PossAtRisk}(\varphi)) \wedge \mathsf{Des}(\varphi)$
   - $\mathsf{NotFavourableAct}(\alpha, \varphi) =_{def} \mathsf{Bel}(\langle \alpha \rangle \mathsf{PossAtRisk}(\varphi)) \wedge \mathsf{Des}(\varphi)$

   Note that here we consider regular actions instead of atomic ones since the risk condition is not bounded to verify in a next situation.

3. *Favourable:* a condition is said to be *favourable* if it refers to a current situation of the environment in which a desire of the agent has the possibility to be achieved. We define the following kinds of favourable conditions:
   - $\mathsf{Favourable}(\varphi) =_{def} \mathsf{Bel}(\mathsf{Safe}(\varphi)) \wedge \mathsf{Des}(\varphi)$
   - $\mathsf{FavourableProp}(\varphi, \psi) =_{def} \mathsf{Bel}(\psi \rightarrow \mathsf{Safe}(\varphi)) \wedge \mathsf{Des}(\varphi)$
   - $\mathsf{FavorableAct}(\alpha, \varphi) =_{def} \mathsf{Bel}(\langle \alpha \rangle \mathsf{Safe}(\varphi)) \wedge \mathsf{Des}(\varphi)$

4 Modelling Emotions in $\mathcal{E}_{\mathsf{BDI}}$

In this section we present the modelling of three emotions within $\mathcal{E}_{\mathsf{BDI}}$ logic: *Fear*, *Anxiety* and *Self-Confidence*. For each of these emotions we model both its activation conditions and the effects that their presence have in the future behaviour of an Emotional-BDI agent. This modelling is based in the work of Oliveira & Sarmento in [4].

The activation condition of each of the emotions corresponds precisely to a condition defined in the previous section. We opted by this approach to avoid the logical omniscience problem [17]. The use of a notation $Emotion(F(\varphi))$ allows a more intuitive meaning and can help in the future development of a formal calculus for (emotional) $\mathcal{E}_{\mathsf{BDI}}$-formulae.

4.1 Fear

The activation of fear occurs when a fundamental desire of the agent is put at risk of failure. Using other words, fear is activated when the agent detects a threat. Therefore we have the following kinds of fear:

- $\mathsf{Fear}(\neg\varphi) \equiv \mathsf{Threatened}(\varphi)$
- $\mathsf{Fear}(\psi \rightarrow \neg\varphi) \equiv \mathsf{ThreatsProp}(\psi, \varphi)$
- $\mathsf{Fear}(\langle a\rangle\neg\varphi) \equiv \mathsf{ThreatsAct}(a, \varphi)$

The main effect of fear is bringing the agent into a cautious perspective towards the environment and, in particular, to the threat he detected. Depending on the kind of threat, the agent will aim at avoiding that threat. We consider the following behaviours under the effect of fear:

- if the agent can avoid a threat through the execution of an action a the he intends to execute it:
  $\mathsf{Fear}(\neg\varphi) \wedge \mathsf{Can}(a, \varphi) \rightarrow \mathsf{Int}(\langle a\rangle\varphi)$
- if the agent cannot avoid the threat through an action a then he does not intend to execute it:
  $\mathsf{Fear}(\neg\varphi) \wedge \mathsf{Cannot}(a, \varphi) \rightarrow \neg\mathsf{Int}(\langle a\rangle\varphi)$
- if the agent can avoid a proposition which is a threat, or can make the proposition and the fundamental desire coexist – both through the execution of an action – then the agent intends to execute that action:
  $\mathsf{Fear}(\psi \rightarrow \neg\varphi) \wedge \mathsf{Can}(a, \neg\psi) \rightarrow \mathsf{Int}(\langle a\rangle\neg\psi)$
  $\mathsf{Fear}(\psi \rightarrow \neg\varphi) \wedge \mathsf{Can}(a, \psi \wedge \varphi) \rightarrow \mathsf{Int}(\langle a\rangle(\psi \wedge \varphi))$
- if the execution of an action is a threat to the agent then the agent will not intend to execute it (possibly for achieving some proposition ψ) until it causes no fear:
  $\mathsf{Fear}(\langle a\rangle\neg\varphi) \rightarrow \mathsf{A}(\neg\mathsf{Int}(\langle a\rangle\top)\mathsf{U}\neg\mathsf{Fear}(\langle a\rangle\varphi))$
- if the agent believes that an action a for which it does not have resources can eliminate the threat, then one of the following conditions apply:
  1. the agent can eliminate the fear by freeing previously saved resources to execute other action:
     $\mathsf{Fear}(\neg\varphi) \wedge \mathsf{Cannot}(a, \varphi) \wedge \mathsf{Bel}([\mathtt{FREE}(\alpha)]\mathsf{Can}(a, \varphi)) \rightarrow \mathsf{Int}(\langle\mathtt{FREE}(\alpha); a\rangle\varphi)$
  2. the agent believes it can get the resources for a before compromising its fundamental desire:
     $\mathsf{Fear}(\neg\varphi) \wedge \mathsf{Cannot}(a, \varphi) \wedge \mathsf{Bel}([\mathtt{GET}(\alpha)]\mathsf{Can}(a, \varphi)) \rightarrow \mathsf{Int}(\langle\mathtt{GET}(\alpha); a\rangle\varphi)$

4.2 Anxiety

The activation of anxiety occurs when the desires of the agent can be at risk in the future. Therefore, anxiety works as preventive alert system towards future situations which may compromise the overall performance of the agent. We consider the following kinds of anxiety activation:

- $\mathsf{Anx}(\mathsf{EF}\neg\varphi) \equiv \mathsf{NotFavourable}(\varphi)$
- $\mathsf{Anx}(\psi \rightarrow \mathsf{EF}\neg\varphi) \equiv \mathsf{NotFavourableProp}(\psi, \varphi)$
- $\mathsf{Anx}(\langle\alpha\rangle\mathsf{EF}\neg\varphi) \equiv \mathsf{NotFavourableAct}(\alpha, \varphi)$

The effects of anxiety are mainly preparing the agent to face future risk conditions, or to avoid them before they occur. We consider the following cases:

- if an action α guarantees that the desire will not be at risk, the agent intends to execute α. If he does not have enough resources, he will save them:
  $\mathsf{Anx}(\mathsf{EF}\neg\varphi) \wedge \mathsf{Can}(\alpha, \mathsf{AF}\varphi) \rightarrow \mathsf{Int}(\langle\alpha\rangle\mathsf{AF}\varphi)$
  $\mathsf{Anx}(\mathsf{EF}\neg\varphi) \wedge \mathsf{Int}(\langle\alpha\rangle\mathsf{AF}\varphi) \wedge \neg\mathsf{Res}(\alpha) \rightarrow \langle\mathtt{SAVE}(\alpha)\rangle\mathsf{Int}(\langle\alpha\rangle\mathsf{AF}\varphi)$
- if a proposition causes anxiety and the agent has a way to either negate that proposition or make that proposition coexist with the desire possibly at risk, then the agent will execute that action:
  $\mathsf{Anx}(\psi \rightarrow \mathsf{EF}\neg\varphi) \wedge \mathsf{Can}(\alpha, \mathsf{AF}(\neg\psi \vee (\psi \wedge \varphi)) \rightarrow \mathsf{Int}(\langle\alpha\rangle\mathsf{AF}(\neg\psi \vee (\psi \wedge \varphi)))$
- if the execution of an action is causing anxiety and the execution of that action is an intention of the agent, the agent will not intend it until it becomes harmful:
  $\mathsf{Anx}(\langle\alpha\rangle\mathsf{EF}\neg\varphi) \wedge \mathsf{Int}(\langle\alpha\rangle\varphi) \rightarrow \mathsf{AX}(\mathsf{A}(\neg\mathsf{Int}(\langle\alpha\rangle\varphi)\mathsf{U}\mathsf{Bel}(\mathsf{AF}\varphi)))$

4.3 Self-confidence

Self-confidence represents the well-being of the agent relatively to the future achievement of one of its desires. Using other words, if a desire is in a favourable condition to be achieved, the agent feels self-confidence about its achievement. We consider the following kinds of self-confidence:

- $\mathsf{SConf}(\varphi) \equiv \mathsf{Favourable}(\varphi)$
- $\mathsf{Sconf}(\psi \rightarrow \varphi) \equiv \mathsf{FavourableProp}(\psi, \varphi)$
- $\mathsf{SConf}(\langle\alpha\rangle\varphi) \equiv \mathsf{FavourableAct}(\alpha, \varphi)$

Self-confidence deals mostly with the maintainance of intentions. Since the desires are considered to be achievable, the agent only cares about maintaining them in the set of intentions until he believes he achieved them. We consider the following kinds of behaviour:

- if the agent already intends a desire to which he is self-confident about, the agent will continue to intend it until he believes it is achieved:
  $\mathsf{SConf}(\varphi) \wedge \mathsf{Int}(\langle\alpha\rangle\varphi) \rightarrow \mathsf{A}(\mathsf{Int}(\langle\alpha\rangle\varphi)\mathsf{U}\mathsf{Bel}(\varphi))$
- if the agent still does not intend the desire, he will begin to intend it from the next situation on:
  $\mathsf{SConf}(\varphi) \wedge \mathsf{Can}(\alpha, \varphi) \wedge \neg\mathsf{Int}(\langle\alpha\rangle\varphi) \rightarrow \mathsf{AXInt}(\langle\alpha\rangle\varphi)$

- if a proposition causes self-confidence about a desire, then the agent will start intending that proposition and also intend both the proposition and the desire itself:
  $\mathsf{SConf}(\psi \to \varphi) \wedge \mathsf{Can}(\alpha, \psi) \wedge \neg\mathsf{Int}(\langle\alpha\rangle\psi) \to \mathsf{AXInt}(\langle\alpha\rangle\varphi)$
  $\mathsf{SConf}(\psi \to \varphi) \to \mathsf{Int}(\psi \wedge \varphi)$
- if the agent has the resources needed to execute an action which will guarantee the achievement of a desire to which it is self-confident about, then the agent will free those resources and intend to get them right before executing the action:
  $\mathsf{SConf}(\langle\alpha\rangle\varphi) \wedge \mathsf{Int}(\langle\alpha\rangle\varphi) \wedge \mathsf{Saved}(\alpha) \to \langle\mathtt{FREE}(\alpha)\rangle\mathsf{Int}(\langle\mathtt{GET}(\alpha);\alpha\rangle\varphi)$

4.4 Usability of $\mathcal{E}_{\mathsf{BDI}}$

The main goal behind the development of $\mathcal{E}_{\mathsf{BDI}}$ was to provide a language expressive enough to specify conditions where emotions are triggered and the effect that the presence of such emotions have in the behaviour of the agent. The formulas we presented in the Sections 4.1, 4.2 and 4.3 are not supposed to be all present but rather combined to fit the special needs of the environment and role of the agent. This combination should define the *emotional state* of the agent. This mental state works on top of the properties already present in the BDI logic.

Lets consider a scenario where a fire-fighter agent is fighting a nearby fire. It is acceptable that the agent may fear of being burned, although believing that he can extinguish the fire. The emotional state would contain:

1. $\mathsf{Fear}(\neg\mathtt{healthy})$
2. $\mathsf{SConf}(\mathtt{extinguished\_fire})$

The result of this emotional state could be getting back to protect from very close fire but still continuing to throw it water, which is formalised as:

1. $\mathsf{Fear}(\neg\mathtt{healthy}) \wedge \mathsf{Can}(\mathit{get\_back}, \mathtt{healthy}) \to \mathsf{Int}(\langle \mathit{get\_back}\rangle)\mathtt{healthy}$
2. $\mathsf{A}(\mathsf{Int}(\langle \mathit{through\_water}\rangle\mathtt{extinguished})\mathsf{U}\mathsf{Bel}(\mathtt{extinguished}))$

We could select different conditions to model for instance another fire-fighter agent fighting the fire, a police agent, etc.

5 Implementation

We have implemented the tableau algorithm presented in Section 2.5 for determining the satisfiability of $\mathcal{E}_{\mathsf{BDI}}$-formulas. Our implementation was done in the Prolog language. Currently we are implementing $\mathcal{E}_{\mathsf{BDI}}$ syntax and semantics as a module of the Coq interactive theorem prover system [18]. Our aim is to provide a computational mean of doing model-checking for $\mathcal{E}_{\mathsf{BDI}}$-formulae. We base our approach in the work of de Wind [19] in implementing normal modal logic in Coq plus some implementations of formal concepts present in the $\mathcal{E}_{\mathsf{BDI}}$ logic and already implemented in Coq, as CTL logic.

6 Related Work

The work which more relates to the one we present in this paper is the one of Meyer in [20], where he proposes the formal modelling of *happiness*, *sadness*, *anger* and *fear* in the **KARO** logical framework.

Meyer suggests the introduction of a modal operator $Goal_m(\varphi)$ which represents a so called *maintainance goal*. This modal operator is used to model fear with a similar intuition as the one behind our $\mathsf{Fund}(\varphi)$ modal operator, *i.e.*, to define a more important kind of desire. In terms of modelling the evolution of the agent, Meyer uses computational sequences of atomic actions to refer to future states of an agent, while we use the standard CTL's temporal operators.

In a more recent work, Meyer and Dastani introduce the modelling of emotions previously done in an agent oriented programming language [21]. In this work the authors present transition rules for the generation of each of the emotions modelled in [20]. This generated emotions are then feed into the programming language's deliberation process which determine the effects that these emotions have in the mental states of an agent.

Comparing both approaches we conclude that:

1. Our approach provides a more expressive language to model emotions in BDI agents. The combination of time and action execution and the detailed definition of resources and resource-management notions fits in the needs of emotional agent architecture [3,1,2].
2. The new operators which we introduced were conveniently defined syntactically and semantically. The work of Meyer introduces similar concepts but just in the language of its logic. Our work also has a strong focus on the logical foundations of $\mathcal{E}_{\mathsf{BDI}}$ whereas Meyer's work focus only in expressing emotional states of rational agents.

Despite the differences, both logical frameworks try to model rational agents with emotions in the same context: they are not interested about the internal structure of the emotions, but only in specifying at which conditions they are activated and how their presence influence the behaviour of the agent.

7 Conclusions and Future Work

In this paper we have presented an improved version of the $\mathcal{E}_{\mathsf{BDI}}$ logic to model the activation and effects of emotions in the behaviour exhibited by a Emotional-BDI agent. The emotions analysed were fear, anxiety and self-confidence. This formalisation was based in the $\mathsf{BDI}_{\mathsf{CTL}}$ logic, which was extended with the notions of fundamental desire, explicit reference to actions, capabilities and resources.

We have shown that the satisfiability of $\mathcal{E}_{\mathsf{BDI}}$-formulae can be reduced to the satisfiability of $\mathsf{BDI}_{\mathsf{CTL}}$-formulae. We have implemented an extended version of the $\mathsf{BDI}_{\mathsf{CTL}}$'s tableau decision procedure for $\mathcal{E}_{\mathsf{BDI}}$-formulae.

Currently we are developing a library for the Coq interactive theorem prover system [18] with the purposes of reasoning and model-checking specifications of Emotional-BDI agents within the $\mathcal{E}_{\mathsf{BDI}}$ framework.

As a future work, it would be interesting to add a notion of graded importance to the Fund modal operator in order to provide a more accurate notion of the importance of a desire to the decision-making process of an agent, in the line of the work done by Godo *et al.* in [22].

Acknowledgements

We kindly thanks the reviewers for their constructive comments which helped improving this work.

This work was partially founded by Fundao para a Cincia e Tecnologia (FCT) and program POSI.

References

1. Pereira, D., Oliveira, E., Moreira, N., Sarmento, L.: Towards an architecture for emotional BDI agents. In: Carlos Bento, A.C., Dias, G. (eds.) EPIA 2005 12th Portuguese Conference on Artificial Intelligence, Universidade da Beira Interior, December 2005, pp. 40–46. IEEE, Los Alamitos (2005); ISBN 0-7803-9365-1
2. Pereira, D., Oliveira, E., Moreira, N.: Modelling emotional BDI agents. In: Workshop on Formal Approaches to Multi-Agent Systems (FAMAS 2006), Riva Del Garda, Italy (August 2006)
3. Oliveira, E., Sarmento, L.: Emotional valence-based mechanisms and agent personality. In: Bittencourt, G., Ramalho, G. (eds.) SBIA 2002. LNCS (LNAI), vol. 2507, pp. 152–162. Springer, Heidelberg (2002)
4. Oliveira, E., Sarmento, L.: Emotional advantage for adaptability and autonomy. In: AAMAS, pp. 305–312 (2003)
5. Sarmento, L., Moura, D., Oliveira, E.: Fighting fire with fear. In: Proceedings of 2nd European Workshop on Multi-Agent Systems (EUMAS 2004) (December 2004)
6. Rao, A.S., Georgeff, M.P.: Decision procedures for BDI logics. J. Log. Comput. 8(3), 293–342 (1998)
7. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In: Allen, J., Fikes, R., Sandewall, E. (eds.) Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR 1991), pp. 473–484. Morgan Kaufmann publishers Inc., San Francisco (1991)
8. van der Hoek, W., van Linder, B., Meyer, J.J.C.: A logic of capabilities. In: Nerode, A., Matiyasevich, Y. (eds.) LFCS. LNCS, vol. 813, pp. 366–378. Springer, Heidelberg (1994)
9. Schmidt, R.A., Tishkovsky, D., Hustadt, U.: Interactions between knowledge, action and commitment within agent dynamic logic. Studia Logica 78(3), 381–415 (2004)
10. van Linder, B., van der Hoek, W., Meyer, J.J.C.: Formalising abilities and opportunities of agents. Fundamenta Informaticae 34(1-2), 53–101 (1998)
11. van der Hoek, W., van Linder, B., Meyer, J.J.C.: On agents that have the ability to choose. Studia Logica 66(1), 79–119 (2000)
12. Emerson, E.A.: Temporal and modal logic. In: Handbook of Theoretical Computer Science. Formal Models and Sematics (B), vol. B, pp. 995–1072 (1990)
13. Bratman, M.E., Israel, D., Pollack, M.E.: Plans and resource-bounded practical reasoning. Computational Intelligence 4, 349–355 (1988)

14. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
15. Schild, K.: On the relationship between bdi logics and standard logics of concurrency. Autonomous Agents and Multi-Agent Systems 3(3), 259–283 (2000)
16. Halpern, J.Y., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artif. Intell. 54(3), 319–379 (1992)
17. Whitsey, M.: Logical omniscience: A survey (2003)
18. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions. In: Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science. Springer, Heidelberg (2004)
19. de Wind, P.: Modal logic in coq. Master's thesis, Vrije Universiteit in Amsterdam (2001)
20. Meyer, J.J.C.: Reasoning about emotional agents. In: de Mántaras, R.L., Saitta, L. (eds.) ECAI, pp. 129–133. IOS Press, Amsterdam (2004)
21. Dastani, M., Meyer, J.J.C.: Programming agents with emotions. In: Brewka, G., Coradeschi, S., Perini, A., Traverso, P. (eds.) ECAI, pp. 215–219. IOS Press, Amsterdam (2006)
22. Casali, A., Godo, L., Sierra, C.: Graded bdi models for agent architectures. In Leite, J.A., Torroni, P. (eds.) CLIMA V. In: Leite, J.A., Torroni, P. (eds.) CLIMA 2004. LNCS (LNAI), vol. 3487, pp. 126–143. Springer, Heidelberg (2005)

‘What I Fail to Do Today, I Have to Do Tomorrow’: A Logical Study of the Propagation of Obligations

Jan Broersen[1] and Julien Brunel[2]

[1] Institute of Information and Computing Sciences
Utrecht University, The Netherlands
[2] Institut de Recherche en Informatique de Toulouse
University of Toulouse, France
broersen@cs.uu.nl, brunel@irit.fr

Abstract. We study a logical property that concerns the preservation of future directed obligations that have not been fulfilled yet. We call this property ’propagation property’. The goal is to define a combination of temporal and deontic logics which satisfies this property. Our starting point is the product of temporal and deontic logics. We investigate some modifications of the semantics of the product in order to satisfy the propagation property, without losing too much of the basic properties of the product. We arrive at a semantics in which we only consider ideal histories that share the same past as the current one, and that enables an interesting characterization of the states in which obligations propagate: these are the states where there are no violations of present directed obligations.

1 Introduction

A strong intuition concerning the interaction of deontic and temporal modalities is that an obligation to achieve something in the future should propagate to future moments if it is not met presently. This is particularly true for deadline obligations; if I have to finish my paper before the end of the week, and I do not finish it today, tomorrow I still have to finish it before the end of the week. But, the propagation property also pertains to future directed obligations without a deadline: if today I need to give a party someday, and I do not give the party today, then tomorrow I still have to give the party someday.

We want to emphasize that such properties are only valid if we assume that the ‘deontic realm’ is not changed by an explicit update of the norms that construct it. What we call the ‘deontic realm’ is relative to the actual situation and to an external body of norms that determines what the obligations of the agent in the given situation are (e.g. a lawbook, or some other set of rules the agents have to comply to). If we allowed the body of norms to vary, the propagation property would not hold. For instance, to come back to the above mentioned examples, there may be an unexpected extension of the deadline for the paper,

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 82–99, 2008.

or my friends have waited in vain for too long and no longer demand that I give the party. This means that I no longer have to finish the paper before the end of the week, and, that I no longer have to organize the party. In such cases, the preservation of the original obligations is not prolonged, due to a change of the norms, and accordingly, a change of the deontic realm. In this paper we will not be concerned with these explicit updates of the norms; we only consider logical properties for the case where the norms are settled, and where it makes sense to reason about the preservation and propagation of the resulting obligations on the basis of what actually happens. So, the only changes of the deontic realm we consider are the ones due to changes of the *situation*.

The problem of propagation of obligations is an instance of the more general problem of the interaction of ’what is obligatory’ with ’what is actually the case’. In deontic logic such interactions are only considered sporadically. For instance, in SDL [18], we do not have the interaction property $\boldsymbol{O}(\varphi \vee \psi) \wedge \neg\varphi \Rightarrow \boldsymbol{O}(\psi)$, although it might be considered quite reasonable: if I have to be smart or strong, and in fact I am not strong, I have to be smart. A possible ground for not wanting this property is that in combination with ’weakening’, that is $\boldsymbol{O}(\varphi) \Rightarrow \boldsymbol{O}(\varphi \vee \psi)$, which is valid in SDL, we get that $\boldsymbol{O}(\varphi) \wedge \neg\varphi \Rightarrow \boldsymbol{O}(\psi)$. Thus by the combination of weakening and the proposed interaction of obligations with conditions being settled, we get that when there is a violation, everything is obligatory. Then, two reactions are possible: (1) indeed this property is bad, and we have to see what we can do to avoid it (while keeping the interaction with facts, in which we are interested), or (2) maybe this property is not as bad as it seems. Below we elaborate on both reactions.

If we want to avoid $\boldsymbol{O}(\varphi) \wedge \neg\varphi \Rightarrow \boldsymbol{O}(\psi)$ the obvious choice would be to attack weakening. Ross’ famous paradox [17] indeed questions this property. Ross’ paradox just says that $\boldsymbol{O}(\varphi) \Rightarrow \boldsymbol{O}(\varphi \vee \psi)$ is not intuitive under some readings. For instance, being obliged to post a letter does not imply being obliged to post or burn it. But many deontic logicians (see e.g., [11]) have argued that the paradox is due to a naive interpretation of the formula. If we read $\boldsymbol{O}(\varphi)$ properly as ’φ is a necessary condition of any state that is optimal according to ones obligations’, then the property poses no problems. Another way to say this is that $\boldsymbol{O}(\varphi)$ expresses an ‘at least’ reading of what is obligatory: it is obligatory to at least satisfy φ, but maybe other properties also have to be obligatory at the same time.

Another way to avoid $\boldsymbol{O}(\varphi) \wedge \neg\varphi \Rightarrow \boldsymbol{O}(\psi)$ is to refine the interaction between ’what is obligatory’ with ’what is actually the case’, i.e., to specialise the interaction property $\boldsymbol{O}(\varphi \vee \psi) \wedge \neg\varphi \Rightarrow \boldsymbol{O}(\psi)$. Our point is to consider that what happens today can only have an effect on what will be obligatory tomorrow, and not on what is obligatory today. We follow the idea that one time step is needed so that the deontic realm takes into account what happens. According to this reading, if I have to be smart or strong today, then the fact I am in one of the four possible situations (strong and smart, strong and not smart, etc.) does not change my obligation to be either smart or strong. In one of these situations (neither strong nor smart) the obligation is violated, while it is fulfilled in the other ones. On the other hand, things change in a temporal context. Indeed, if I have the obligation to be in Paris today or in Amsterdam tomorrow, then the obligation I will have tomorrow will depend on what I do today: if I am not in Paris today, then tomorrow I will have the obligation to be in Amsterdam, otherwise I will not. (Whether I am in Paris or not today, today's obligation does not change, only tomorrow's obligation does.) It is closely related to the fact that an obligation only concerns the present or the future: an obligation that yesterday I was in Paris does not make sense. Thus, in case I am not in Paris today, tomorrow's obligation will not be 'to be in Amsterdam or to have been in Paris yesterday' but simply 'to be in Amsterdam'. So in this paper the interaction property we will consider is not $\boldsymbol{O}(\varphi \vee \psi) \wedge \neg\varphi \Rightarrow \boldsymbol{O}(\psi)$ but $\boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\varphi \Rightarrow X\boldsymbol{O}(\psi)$, and we will call it propagation property.

As a further motivation for this work we will first point to some related problems. Preservation properties have been studied for intentions. However, intentions are preserved for a different reason. As Bratman [3,4] explains, intentions serve to stabilize an agent's deliberations. An agent cannot continuously reconsider his decisions, simply because usually there is no time to do that. It is usually more rational to stick to reached decisions (thereby turning them into intentions), and to only let achievement of that what is intended be a cause for discharging the obligation. In AI (that is, AI as studied in computer science), the best-known formalizations of rationality postulates for the phenomenon of intention preservation are Rao and Georgeff's 'commitment strategies' [15]. For obligations, the reason to preserve them is different: they are preserved simply because an agent has to meet up to his obligations at some point, unless, of course, he is explicitly relieved from his obligations. But, as said, we do not consider that issue in this paper. Another motivating example is the preservation of goals in the mechanisms underlying agent programming languages like AgentSpeak [16] and 3APL [12,8]. Such agent programming languages usually comprise programming rules that work on data structures for beliefs, plans and goals. The goals in the 'goal bases' of agent programs are often sets of propositional formulas denoting properties the agent aims at making true. The operational semantics of the programming languages typically treats the goals in the goal bases as information that has to be preserved unless a state is reached where the agent believes the goal is achieved. Of course, goals are not identical to obligations. But it is clear that they at least share the phenomenon of propagation. One of the motivations is thus to provide a logical basis for verification of agent programs against logically formulated rationality postulates about the propagation of goals and obligations.

The present paper is organised as follows. Section 2 presents the product of Linear Temporal Logic (LTL) [14] and Standard Deontic Logic (SDL) [18], which is an appropriate starting point for our investigation. Section 3 shows that the propagation property is not compatible with the genuine product and presents modifications of the product semantics which guarantee the propagation of obligations. Section 5 concludes the paper.

2 Product of Temporal and Deontic Logic

We present here standard deontic logic SDL [18], linear temporal logic LTL [14], and their product logic $LTL \times SDL$. We choose the product as the starting point because the commutativity properties, which are specific for products, correspond to a setting without norm updates. As we explained in the introduction we are not interested in explicit updates here.

2.1 Deontic and Temporal Logics

Deontic logic is the modal logic of obligation, permission, and prohibition.

Definition 1 (Deontic language). *The deontic language $\mathcal{DL}$ is defined as*

$$\mathcal{DL} ::= P \mid \bot \mid \mathcal{DL} \Rightarrow \mathcal{DL} \mid \mathbf{O}(\mathcal{DL})$$

The necessity operator $\mathbf{O}$ *is read* it is obligatory that*. The possibility operator* $\mathbf{P} \overset{def}{=} \neg\mathbf{O}\neg$ *is read* it is permitted that*. We can read* $\mathbf{O}(\neg\varphi)$ *as* it is prohibited that φ*.*

The boolean operators are defined as usual:

$$\neg\varphi \overset{def}{=} \varphi \Rightarrow \bot \quad \top \overset{def}{=} \neg\bot \quad \varphi_1 \vee \varphi_2 \overset{def}{=} \neg\varphi_1 \Rightarrow \varphi_2 \quad \varphi_1 \wedge \varphi_2 \overset{def}{=} \neg(\varphi_1 \Rightarrow \neg\varphi_2)$$

The truth-relation $\models$ between a world of a Kripke model and a formula is given through the usual possible world semantics, with which we consider the reader is familiar. Standard Deontic Logic SDL can be defined as the logic over the language $\mathcal{DL}$ determined by the Kripke frames (W, R) such that R is serial, i.e. as the set of the $\mathcal{DL}$-formulas that are valid in every frame (W, R) such that R is serial. In a deontic frame, W is the set of the worlds, and the intuitive reading of R is that it associates each world with a set of ideal worlds, in which every obligation is fulfilled.

Definition 2 (Temporal language). *We consider a monadic operator X called* next*, and a dyadic operator U called* until*. Given a set P of atomic propositions, the temporal language $\mathcal{TL}$ is defined as*

$$\mathcal{TL} ::= P \mid \bot \mid \mathcal{TL} \Rightarrow \mathcal{TL} \mid X\ \mathcal{TL} \mid \mathcal{TL}\ U\ \mathcal{TL}$$

The informal meaning of the temporal operators X and U are as follows:

$X\varphi$: "at the next moment, φ will hold."

$\varphi_1\ U\ \varphi_2$: "φ_2 will eventually hold at some moment m, while φ_1 holds from now until the moment before m."

The temporal operators F (finally, or eventually), G (globally, or always) and $F_{\leqslant k}$ (before k time units) are defined as the following abbreviations (the boolean operators are defined as for $\mathcal{DL}$):

$$F\ \varphi \overset{def}{=} \top U \varphi \qquad G\ \varphi \overset{def}{=} \neg F\ \neg\varphi \qquad F_{\leqslant k}\varphi \overset{def}{=} \begin{cases} \varphi \text{ if } k = 0 \\ \varphi \vee XF_{\leqslant k-1}\varphi \text{ else} \end{cases}$$

Definition 3 (Linear temporal structure and model). *We consider here a linear, infinite, and discrete time. The unique temporal frame is* $(\mathbb{N}, <)$ *where*

- $\mathbb{N}$ *is the set of the natural numbers.*
- $<$ *is the usual strict order on the natural numbers.*

Given a set of atomic propositions P*, a temporal valuation* V *is a function* $V : \mathbb{N} \to 2^P$ *which associates each state with a set of atomic propositions.*

Let us define the satisfaction relation between a state of a model and a temporal formula.

Definition 4 (Satisfaction). *Given a set of atomic propositions* P*, a temporal valuation* V*, a moment* $i \in \mathbb{N}$*, and a formula* φ *of* $\mathcal{TL}$*, the satisfaction relation* $\models$ *is defined by induction on* φ *as follows:*

$$\begin{array}{lll}
i \models p & \textit{iff} & p \in V(i) \quad \textit{where } p \in P \\
i \not\models \bot & & \\
i \models \varphi_1 \Rightarrow \varphi_2 & \textit{iff} & \textit{if} \quad i \models \varphi_1 \quad \textit{then} \quad i \models \varphi_2 \\
i \models X\varphi & \textit{iff} & i+1 \models \varphi \\
i \models \varphi_1 \; U \; \varphi_2 & \textit{iff} & \exists i' \geqslant i \quad \textit{such that} \quad i' \models \varphi_2 \quad \textit{and} \\
 & & \forall\, i'' \in \mathbb{N} \quad \textit{if} \quad i \leqslant i'' < i' \quad \textit{then} \quad i'' \models \varphi_1
\end{array}$$

The logic formulated in the language $\mathcal{TL}$ which is determined by the unique frame $(\mathbb{N}, <)$ is called *LTL* (*Linear Temporal Logic*)[14].

2.2 Temporal and Deontic Product

We define here the product of temporal and deontic logics. The product frames correspond to the usual product definition [10] (see figure 1 for an illustration).

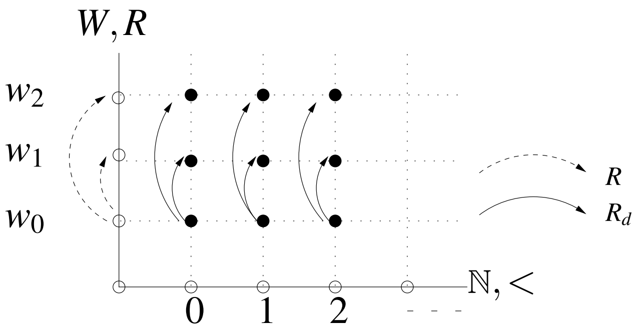


Fig. 1. Illustration of the product $(\mathbb{N}, <) \times (W, R)$

Definition 5 (Product frame, product model). *Let* $T = (\mathbb{N}, <)$ *and* $D = (W, R)$ *be respectively a temporal frame and a deontic frame. Then the product frame* $T \times D$ *is a triple* $(S, <_t, R_d)$ *where*

- *$S = \mathbb{N} \times W$ (the set of the states) is the Cartesian product of the set $\mathbb{N}$ of the natural numbers, viewed as a set of moments, and the set W of the worlds,*
- *$<_t \subseteq S \times S$ is the temporal relation on states such that $(i, w) <_t (i', w')$ if and only if $i < i'$ and $w = w'$,*
- *$R_d \subseteq S \times S$ is the deontic relation on the states such that $(i, w)R_d(i', w')$ if and only if wRw' and $i = i'$. We then say that w' is an ideal world of w at moment i, or that (i, w') is an ideal state of (i, w).*

Given a set P of atomic propositions, a valuation V for $T \times D$ is a function $V : S \to 2^P$ that associates each state with a set of atomic propositions. The pair $(T \times D, V)$ is then called a product model based on $T \times D$.

The language of the product logic $LTL \times SDL$ combines LTL operators and SDL operator.

Definition 6 (Syntax of $LTL \times SDL$). *Given a countable set P of atomic propositions, the temporal deontic language $\mathcal{TDL}$ of $LTL \times SDL$ is defined by:*

$$\mathcal{TDL} ::= P \;\mid\; \bot \;\mid\; \mathcal{TDL} \Rightarrow \mathcal{TDL} \;\mid\; X(\mathcal{TDL}) \;\mid\; \mathcal{TDL}\ U\ \mathcal{TDL} \;\mid\; \mathbf{O}(\mathcal{TDL})$$

Usual boolean and temporal operators defined as abbreviations in the definition of temporal and deontic languages (definitions 1 and 2 respectively) are also available.

We can now define the satisfaction relation for the deontic and temporal product logic.

Definition 7 (Satisfaction). *A formula φ of $\mathcal{TDL}$ is interpreted on a state of a product model. Given a product model $((S, <_t, R_d), V)$, a state $s = (i, w) \in S$, and a formula φ, we can define the satisfaction relation $\models$ by induction on φ:*

$$\begin{array}{llllll}
s \models X\varphi & \textit{iff} & (i+1, w) \models \varphi & \textit{where} & s = (i, w) & \\
s \models \varphi_1\ U\ \varphi_2 & \textit{iff} & \exists s' \geqslant_t s & \textit{such that} & s' \models \varphi_2 & \textit{and} \\
 & & \forall s\text{”} \in S & \textit{if} & s \leqslant_t s'' <_t s' & \textit{then} \quad s'' \models \varphi_1
\end{array}$$

where “$\leqslant_t$” is defined by $\quad s \leqslant_t s' \quad$ *iff* $\quad s <_t s' \quad$ *or* $\quad s = s'$

$$s \models \mathbf{O}\varphi \qquad \textit{iff} \quad \forall s' \in S \quad \textit{if} \quad sR_ds' \quad \textit{then} \quad s' \models \varphi$$

A product model $((W, <_t, R_d), V)$ satisfies a formula φ if every state of the product satisfies it.

A product frame $F = (W, <_t, R_d)$ validates a formula φ if every model based on F satisfies it.

A formula φ is valid if every product frame validates it.

Let us elaborate on the interaction of the temporal and deontic dimensions. For instance, there is no difference between “it is permitted that φ holds tomorrow”, and “tomorrow, φ will be permitted”. This corresponds to the validity of $\boldsymbol{P}(X\varphi) \Leftrightarrow X\boldsymbol{P}\varphi$. Indeed, let $s = (i, w) \in W$ be a state. Suppose that

$s \models \boldsymbol{P}(X\varphi)$. Then there is a state $s' = (i', w')$ such that $sR_d s'$ and $s' \models X\varphi$. So $(i+1, w') \models \varphi$. And thus $(i+1, w) \models \boldsymbol{P}\varphi$. So we can deduce $s \models X\boldsymbol{P}\varphi$. In the same way, we can show that $\models X\boldsymbol{P}\varphi \;\Rightarrow\; \boldsymbol{P}X\varphi$. Then,

$$\models \boldsymbol{P}X\varphi \;\Leftrightarrow\; X\boldsymbol{P}\varphi$$

This property can also be formulated as follows

$$\models \boldsymbol{O}X\varphi \;\Leftrightarrow\; X\boldsymbol{O}\varphi \tag{1}$$

The above commutativity properties are typical for product logics. The properties reflect the fact the deontic realm is not updated, as we said in the introduction. So, if it is obligatory to go to Paris tomorrow, then tomorrow it will be obligatory to go to Paris immediately, and vice versa. Now the question of the next section is whether or not we can add propagation properties to the temporal deontic product while leaving the product intact: is there a non-empty (and interesting) subset of the product frames which validate the propagation property? Intuitively, this should not be the case: propagation means that obligations are 'created' for future moments. The *trigger* for this creation is the circumstance that the obligations are not met presently.

3 Adding a Propagation Property

We want to consider a propagation property as general as possible. For instance we want to capture the obligation with deadline, or the obligation to meet something eventually (without deadline). The obligation to satisfy φ now, or ψ next seems to be the most general kind of obligation for which we want to study the propagation. Indeed, the obligation with deadline $\boldsymbol{O}(F_{\leqslant k}(\varphi))$ can be re-written $\boldsymbol{O}(\varphi \vee XF_{\leqslant k-1}(\varphi))$, and the obligation to satisfy φ eventually $\boldsymbol{O}(F\varphi)$ can be re-written $\boldsymbol{O}(\varphi \vee XF(\varphi))$.

As a first attempt for formalizing a propagation property to be added to the product logic, we consider:

$$\boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\varphi \;\Rightarrow\; X\boldsymbol{O}(\psi) \tag{2}$$

If it is obligatory to meet φ now, or ψ next, and φ is not satisfied now, then it will be obligatory next to meet ψ.

As argued in the introduction, we would not want that from the propagation property and the properties of the temporal deontic logic it follows that $\boldsymbol{O}\varphi \wedge \neg\varphi \;\Rightarrow\; X\boldsymbol{O}(\psi)$. Yet this property does follow from 2 in combination with (a temporal variant of) weakening of obligations: $\boldsymbol{O}\varphi \Rightarrow \boldsymbol{O}(\varphi \vee X\psi)$. To solve this problem, we re-formalize the propagation property, in order to prevent that in combination with temporal weakening it can be used to derive this unwanted property. To achieve this, in the propagation property, we exclude that $\boldsymbol{O}(\varphi \vee X\psi)$ holds *only* because $\boldsymbol{O}(\varphi)$ holds[1], and we thus arrive at the following property instead of (2):

$$\boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\boldsymbol{O}\varphi \wedge \neg\varphi \;\Rightarrow\; X\boldsymbol{O}(\psi) \tag{3}$$

[1] Another strategy might be to attack the temporal weakening property directly.

Similarly, we may explicitly exclude that $\boldsymbol{O}(\varphi \vee X\psi)$ holds *only* because $\boldsymbol{O}(X\psi)$ holds. So we may formulate the propagation formula as follows:

$$\boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\boldsymbol{O}\varphi \wedge \neg\boldsymbol{O}X\psi \wedge \neg\varphi \ \Rightarrow \ X\,\boldsymbol{O}(\psi) \tag{4}$$

However, with the product property $X\,\boldsymbol{O}\varphi \leftrightarrow \boldsymbol{O}X\varphi$, property 4 is equivalent with $\boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\boldsymbol{O}\varphi \wedge \neg\boldsymbol{O}X\psi \wedge \neg\varphi \ \Rightarrow \ \boldsymbol{O}X\psi$. This, in turn, is logically equivalent with $\boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\boldsymbol{O}\varphi \wedge \neg\varphi \ \Rightarrow \ \boldsymbol{O}X\psi$, which is exactly the same property as 3. So, in the product setting, properties 3 and 4 are equivalent.

But, now we have to conclude that the propagation property is not compatible with a genuine product: we can consistently add property 3 to the product logic, but we will never have a case where $X\,\boldsymbol{O}(\psi)$ is really a *consequence* of $\boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\boldsymbol{O}\varphi \wedge \neg\varphi$ being true. In fact, a product model satisfies property 4 only if it does not satisfy the hypothesis $\boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\boldsymbol{O}(\varphi) \wedge \neg\boldsymbol{O}(X\psi) \wedge \neg\varphi$. (Indeed, if a product model satisfied the hypothesis $\boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\boldsymbol{O}(\varphi) \wedge \neg\boldsymbol{O}(X\psi) \wedge \neg\varphi$, in some state s, for some φ and ψ, then we could deduce $\neg X\,\boldsymbol{O}(\psi)$ in s.) This corresponds to a product model where all the ideal states of a given state have the same valuation, which is clearly not interesting to work with.

The only way to preserve the propagation property is then to drop the 'no learning' property $X\,\boldsymbol{O}\varphi \Rightarrow \boldsymbol{O}X\varphi$. So, we will no longer have a genuine product. But this is in accordance with intuitions. Obligations may now be transferred to future states. The above discussion shows that this is incompatible with a product; we have to allow some dynamics in the deontic dimensions because obligations may be inherited from earlier states. We do however preserve the 'perfect recall' property $\boldsymbol{O}X\varphi \Rightarrow X\,\boldsymbol{O}\varphi$ that expresses that no obligations are 'forgotten' over time. Note that this last property is sufficient to ensure that properties 3 and 4 are equivalent. So, in the rest of the paper, we will study property 3, which is shorter.

3.1 Restricting the Ideal States

Our goal in this section will be to define a semantics that satisfies the propagation property and the perfect recall property. To account for propagation, in the semantics we have to introduce a stronger interaction between what happens and what is obligatory, i.e., between what is true in the current world and what is true in the (next) ideal worlds. If we want to satisfy the perfect recall property, the set of ideal worlds in the next state is a *subset* of the set of the ideal worlds in the current state. The principle of propagation then should point us to what subset to take. Our idea is that for ideal worlds at a next moment we should only take into account the worlds that share the same past as the current world until the present moment. This reflects the idea that what is deontically ideal at the next moment depends on what actually occurs presently.

Below, we first define the predicate $SamePast(s, s')$ which says that the states s and s' of a temporal deontic model share the same past:

$$SamePast((i,w),(i',w')) \quad \overset{def}{=} \quad i = i' \ \wedge \ \forall j < i \ \ V(j,w) = V(j,w')$$

When interpreting an obligation in a state s, we only consider the states s' which satisfy sR_ds' and $SamePast(s, s')$.

Definition 8 (Semantics of the obligation (2)). *Given a product model $(S, <_t, R_d)$, a state s, and a formula φ, we now consider the following semantics for obligation:*

$$s \models \mathbf{O}\varphi \quad \textit{iff} \quad \forall s' \in S \quad \textit{if} \quad (SamePast(s', s) \quad \textit{and} \quad sR_ds') \quad \textit{then} \quad s' \models \varphi$$

With this new semantics, the deontic realm is described by fewer and fewer worlds (which means that more and more formulas are obligatory) when time passes. This is conform the fact that we keep $\boldsymbol{O}(X\varphi) \Rightarrow X\boldsymbol{O}(\varphi)$, and avoid $X\boldsymbol{O}(\varphi) \Rightarrow \boldsymbol{O}(X\varphi)$; no obligations are forgotten, but some obligations may appear (in particular when they are propagated from a more general obligation in the previous state).

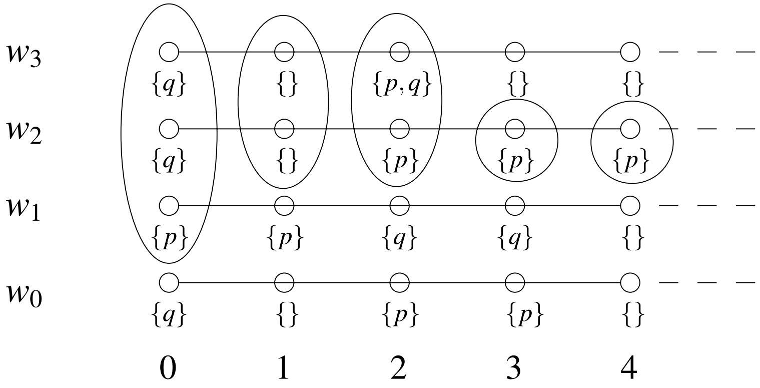


Fig. 2. Semantics of obligation

Let us illustrate, by the way of an example, how an obligation may propagate. Consider the product model illustrated in Figure 2, where, in state $(0, w_0)$, histories w_1, w_2, and w_3, are ideal. Then, we have for instance $0, w_0 \models \boldsymbol{O}(p \vee XXp) \wedge \neg p$. Since w_0 does not satisfy p at instant 0, the history w_1 which satisfies p at instant 0 is not ideal anymore at the next instant. So, only w_2 and w_3 (which satisfy XXp at instant 0) remain ideal at instant 1. Thus, the propagation applies, and we have $0, w_0 \models X\boldsymbol{O}(Xp)$. Let us now state the propagation property and propose a proof in the general case where φ is a propositional formula and ψ can be any formula.

Property 1 (Propagation property). Let M be a temporal deontic product model. Then it satisfies the propagation property for the obligation operator of definition 8:

$$M \models \boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\boldsymbol{O}(\varphi) \wedge \neg\varphi \ \Rightarrow \ X\boldsymbol{O}(\psi)$$

for φ propositional formula, and ψ any formula.

Proof. Let $M = ((S, <_t, R_d), V)$ be a temporal deontic model, and $s = (i, w) \in S$ a state such that $s \models \boldsymbol{O}(\varphi \vee X\psi) \wedge \neg \boldsymbol{O}(\varphi) \wedge \neg\varphi$. Every s' such that $SamePast(s, s')$ and $sR_d s'$ satisfies $\varphi \vee X\psi$, and it is not the case that every such s' satisfies φ. If some of these states $s' = (i, w')$ have the same valuation as s, then they satisfy $\neg\varphi$ (since $s \models \neg\varphi$), and $\varphi \vee X\psi$. So, they satisfy $X\psi$. Thus, for every state $(i+1, w')$ which is ideal from $(i+1, w)$ and has the same past as $(i+1, w)$ satisfies ψ, i.e., $(i+1, w) \models \boldsymbol{O}\psi$. Otherwise (if none of the states s' have the same valuation as s), there is no ideal state having the same past as $(i+1, w)$. So, every formula is obligatory in $(i+1, w)$. In particular, $(i+1, w) \models \boldsymbol{O}\psi$. □

The fact that φ is a propositional formula in accordance with intuition. Indeed, the propagation property expresses that what will be obligatory at the next step may depend on what happens now (φ not being true). So it is natural to consider that φ is a formula which only concerns the present moment, i.e. a propositional formula. Otherwise, if φ contained future operators, what will be obligatory at the next step would depend on something which has not happened yet.

So we have that some of the obligations that may appear at a next state are due to the propagation property. In fact, the following property claims that the propagation property completely characterizes the new obligations that appear.

Property 2 (Characterization of new obligations). For any formula ψ, if in a state s both the formulas $X\boldsymbol{O}(\psi)$ and $\neg\boldsymbol{O}(X\psi)$ hold, then there exists a propositional formula φ such that

$$s \models \boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\varphi$$

So, if there will be next an obligation to satisfy ψ and if this obligation is new (i.e., now, there is no obligation to meet ψ next), then it is due to a current obligation to satisfy $\varphi \vee X\psi$ where φ is propositional and not fulfilled.

Proof. Let ψ a formula and s a state such that $s \models X\boldsymbol{O}(\psi) \wedge \neg\boldsymbol{O}(X\psi)$. Let E the set of the ideal states of s which do not satisfy $X\psi$:

$$E \overset{def}{=} \{s' \in S \ / \ sRs' \text{ and } s' \models \neg X\psi\}$$

We now define the set $V(E)$ of all the valuations of states in E. This set is finite (even if E is infinite) because it belongs to 2^{2^P}. $V(E) \overset{def}{=} \{V(s) \ / \ s \in E\}$. Then we define the propositional formula

$$\varphi \overset{def}{=} \bigvee_{v \in V(E)} \Big(\bigwedge_{p \in v} p \ \wedge \ \bigwedge_{p \notin v} \neg p\Big)$$

Then every ideal state of s either satisfies $X\psi$ or is in E and satisfies φ. So $s \models \boldsymbol{O}(\varphi \vee X\psi)$.

Moreover, since $s \models X\boldsymbol{O}(\psi)$, the states in E - which do not satisfy $X\psi$ - become not ideal at the next step. So they do not share the same atomic propositions with s. Thus $s \models \neg\varphi$. □

Unfortunately, not everything is fine. In particular, the deontically ideal worlds may shrink to the empty set when time passes, as we saw in the proof of property 1. This conflicts with our desire to stay in accordance with SDL where obligations are always consistent: $\neg\boldsymbol{O}\bot$. Another formulation is the D axiom: $\boldsymbol{O}\varphi \;\Rightarrow\; \boldsymbol{P}\varphi$. Then, if from a state s, there is no ideal state with the same past, these properties cannot be satisfied, and every formula is obligatory in s, including $s \models \boldsymbol{O}\bot$. In particular this occurs if there is a violation of a proposition p in a state $s = (i, w)$, i.e., if $s \models p \wedge \boldsymbol{O}(\neg p)$. In this case no ideal state is associated with $(i + 1, w)$.

Property 3. With the new semantics of the obligation, the D axiom is not valid:

$$\not\models \neg\boldsymbol{O}(\bot) \qquad \text{and} \qquad \not\models \boldsymbol{O}\varphi \;\Rightarrow\; \boldsymbol{P}\varphi \quad \text{for any formula } \varphi$$

Proof. The problem is due to the fact that there may be states without any ideal states with the same past. Indeed, let $((S, <_d, R_d), V)$ be a product model, and $s \in S$ a state such that no other state has the same past as s. (It is easy to build such a model.) Then, according to definition 8, $s \models \bot$, which invalidates the D axiom. □

As a solution to this problem, we might consider to add a constraint on the models expressing that from every state there exists an ideal state with the same past.

Definition 9 (Ideal existence constraint on models). *Let $M = ((S, <_d, R_d), V)$ be a temporal deontic product model. We say that M satisfies the ideal existence constraint if*

$$\forall s \in S \quad \exists s' \in S \quad \textit{such that} \quad sR_ds' \textit{ and } SamePast(s, s')$$

This constraint now guarantees validity of the D-axiom.

Property 4 (D axiom). Let M be a temporal deontic product model that satisfies the ideal existence constraint. Then

$$M \models \neg\boldsymbol{O}\bot \quad \text{ or equivalently } \quad M \models \boldsymbol{O}\varphi \Rightarrow \boldsymbol{P}\varphi$$

for any formula φ.

Proof. Let $M = ((S, <_d, R_d), V)$ be a temporal deontic product model that satisfies the ideal existence constraint, and $s \in S$ a state. From definition 9 we have that there exists an ideal state s' with the same past as s. So, from the definition of obligation (definition 8), $s \models \neg\boldsymbol{O}(\bot)$. □

However, again we have to face a problem: the ideal existence constraint interacts with the identical past criterion in an undesirable way. In particular, if there is a simple obligation $\boldsymbol{O}p$ that is violated in the current world, the identical past criterion demands that all ideal next worlds satisfy $\neg p$ in their previous state. However, of these there can be none, since otherwise we would not have had $\boldsymbol{O}p$.

But then, under the identical past criterion, there can be no ideal next worlds as soon as there is a simple obligation $\boldsymbol{O}p$ that is violated presently. But then this directly conflicts with the ideal existence constraint. So, if in our logic, we impose both properties, we actually get that obligations can never be violated.

Property 5 (No violation). Let M be a model satisfying the ideal existence constraint and φ a formula. Then $M \models \neg(\varphi \wedge \boldsymbol{O}(\neg\varphi))$.

The conclusion has to be that we still have to refine the semantics: the violation of obligations should be possible, without losing the interaction between what happens and the deontic realm.

3.2 Levels of Deontic Ideality

Another way to view the problem of the previous section is to say that the semantics should be able to deal with 'contrary to duty' (CTD) situations. In states where there is a violation, something happens that is contrary to what is obligatory for that state. It should not be the case that such situations cause the deontic realm to collapse. So when there is a violation, it should still be possible to point out what is obligatory and what not, despite of the violation in the present state.

We look for a solution to the problem by switching to *levels* of ideality. Rather than an accessibility relation which gives the ideal states, we consider a preference relation $\leqslant_d$, where $s \leqslant_d s'$ means that the state s' is "better" than the state s. This allows us to have several "levels of ideality". The ideal states will be the best states among those which share the same past as the current state. The idea is now that if a state (i, w) violates an obligation of a propositional formula then the ideal states of $(i + 1, w)$ are states which were not ideal for (i, w): the deontic realm thus switches to a *lower level* of ideality. This contrasts with the setting of the previous section, where in this case there would be no ideal states left.

Definition 10 (Temporal deontic frame and model). *A temporal deontic frame* $(S, <_t, \leqslant_d)$ *is defined as the product* $(\mathbb{N}, <)\times(W, \leqslant_{pref})$ *of a temporal frame* $(\mathbb{N}, <)$ *and a deontic frame* $(W, \leqslant_{pref})$*, where* $\leqslant_{pref}$*, considered as a preference relation, is a total quasi-order (total and transitive relation) on* W*.*

A temporal deontic model is defined as a product model based on a temporal deontic frame.

For the temporal and boolean operators the satisfaction relation is defined as above. For the obligation operator it is defined as follows.

Definition 11 (Semantics of the obligation (3)). *Given a temporal deontic model* $((S, <_t, \leqslant_d), V)$*, and a state* $s \in S$*,* φ *is obligatory if there is a state with the same past as s such that every "better" state with the same past satisfies* φ*.*

$$
\begin{aligned}
s \models \mathbf{O}\varphi \quad \textit{iff} \quad & \exists s' \in S \quad \textit{such that} \quad \textit{SamePast}(s, s') \quad \textit{and} \\
& \textit{and} \quad \forall s'' \in S \quad \textit{if} \quad (\textit{SamePast}(s, s'') \wedge s' \leqslant_d s'') \quad \textit{then} \quad s'' \models \varphi
\end{aligned}
$$

Remark 1. If every set of states has at least one maximum element for the quasi-order $\leqslant_d$ (i.e., the relation $\geqslant_d$, defined by $s \geqslant_d s'$ iff $s' \leqslant_d s$, is a well-quasi-order), then we can define the set of the best states among those having the same past:

$$\begin{aligned} BestSamePast(s) \overset{def}{=} \{s' \in S \ / \ SamePast(s, s') \quad \text{and} \\ \forall s'' \in S \quad \text{if} \quad SamePast(s, s'') \ \text{ then } \ s'' \leqslant_d s'\} \end{aligned}$$

And the semantic definition of $\boldsymbol{O}(\varphi)$ becomes more simple:

$$s \models \boldsymbol{O}\varphi \quad \text{iff} \quad \forall s' \in BestSamePast(s) \quad s' \models \varphi$$

For the newly defined models (definition 10) with levels of ideality, there is no need for a constraint to guarantee the validity of the D axiom. Indeed, the existence of a state with the same past is guaranteed by the current state itself. Thus the existence of ideal states is also guaranteed. (Recall that the ideal states are the best among those which share the same past as the current state.) So the D axiom is valid and violations can be satisfied.

However, there still is a phenomenon that has to be considered more closely. When an obligation of a proposition p is violated in a state (i, w), then the ideal states at the step $i+1$ are completely disjoint from the ideal states at the step i. This is easy to see: if $(i, w) \models \neg p \wedge \boldsymbol{O}p$, then all the ideal states of (i, w) satisfy p. On the other hand, the ideal states of $(i+1, w)$ have the same past as $(i+1, w)$, and thus they are states $(i+1, w')$ such that (i, w') does *not* satisfy p. So none of the ideal worlds of (i, w) are ideal for $(i+1, w)$ and vice versa. The problem is now that in such states, the propagation property is not guaranteed anymore because of the change to a completely different set of lower level ideal worlds.

Actually, the condition that makes the set of ideal worlds change between (i, w) and $(i+1, w)$ is a little more general than suggested by the example with the violation of an atomic proposition. More in general, the condition concerns the violation of an obligation for any propositional formula which can be seen as an immediate obligation, that is, any propositional formula concerning the present moment. So, if such an obligation is violated, the current ideal worlds will not be considered as ideal in the future. The current norms become obsolete, and we switch to the norms of a lower level. If not, we have a strong link between what is obligatory now and next, and the propagation property holds.

To characterize these two kinds of states, we define the condition $IdealSameProp(s)$ on a state s which expresses that for every state with the same past as s, there is better state which still has the same past at the next step. This condition ensures that some of the current ideal worlds are still ideal at the next step.

Given a temporal deontic model $((S, <_t, \leqslant_d), V)$ and a state $s \in S$,

$$\begin{aligned} IdealSameProp(s) \overset{def}{=} \quad \forall s' \in S \quad \text{if} \quad SamePast(s, s') \quad \text{then} \\ \exists s'' \in S \quad \text{such that} \quad SamePast(s, s'') \ \text{ and } \ V(s) = V(s'') \quad \text{and} \quad s' \leqslant_d s'' \end{aligned}$$

Remark 2. If $(W, \geqslant_{pref})$ is a well-quasi-order, then $IdealSameProp(s)$ is defined in a more simple way:

$$IdealSameProp(s) \quad \overset{def}{=} \quad \exists s' \in BestSamePast(s) \quad \text{such that} \quad V(s) = V(s')$$

Property 6. Given a temporal deontic model $((S, <_t, \leqslant_d), V)$ and a state $s \in S$, the condition $IdealSameProp(s)$ holds iff there is no violation of a propositional formula in s, that is, iff for any propositional formula φ, $s \models \neg(\boldsymbol{O}(\varphi) \wedge \neg\varphi)$.

Proof. We first prove that if $IdealSameProp(s)$ does not hold, then there is some propositional formula φ such that $s \models \boldsymbol{O}(\varphi) \wedge \neg\varphi$. We then prove the other direction.

'$\Leftarrow$' : Suppose that $IdealSameProp(s)$ does not hold, i.e.,

$$\begin{array}{lll} \exists s' \in S \quad \text{such that} & SamePast(s, s') \quad \text{and} & \\ & \forall s'' \geqslant_d s' \quad \text{if} \quad SamePast(s, s'') \quad \text{then} \quad V(s) \neq V(s'') \end{array}$$

Then, we consider such a state s' and define the set $VAL(s')$ of all the valuations of the states which are at least as good as s' and share the same past.

$$VAL(s') \overset{def}{=} \{V(s") \ / \ s' \leqslant_d s" \quad \text{and} \quad \text{SamePast}(s', s")\}$$

$VAL(s')$ is finite since it is included in the set 2^{2^P}. Let us consider the propositional formula φ defined as follows:

$$\varphi \overset{def}{=} \bigvee_{v \in VAL(s')} \left(\bigwedge_{p \in v} p \ \wedge \ \bigwedge_{p \notin v} \neg p\right)$$

Since every such state s'' has a valuation which is distinct from the valuation of s, then $s \models \neg\varphi$. Besides, from the definition of obligation we have that $s \models \boldsymbol{O}(\varphi)$. Thus, $s \models \boldsymbol{O}(\varphi) \wedge \neg\varphi$.

'$\Rightarrow$' : Let us suppose now that there exists some propositional formula φ such that $s \models \boldsymbol{O}(\varphi) \wedge \neg\varphi$. Then,

$$\begin{array}{lll} \exists s' \in S \quad \text{such that} & SamePast(s, s') \quad \text{and} & \\ & \forall s'' \geqslant_d s' \quad \text{if} \quad SamePast(s, s'') \quad \text{then} \quad s'' \models \varphi \end{array}$$

Every such s'' has a valuation which differs from the valuation of s, i.e., $V(s'') \neq V(s)$, since $s \models \neg\varphi$ and $s'' \models \varphi$. Therefore, $IdealSameProp(s)$ does not hold. □

In a state s that satisfies $IdealSameProp(s)$, the deontic realm that will be considered next is a subset of the current deontic realm. So we still have, as in section 3.1, that no obligations are forgotten, but some may appear. If $IdealSameProp(s)$, then $s \models \boldsymbol{O}(X\varphi) \Rightarrow X\boldsymbol{O}(\varphi)$, but $X\boldsymbol{O}(\varphi) \Rightarrow \boldsymbol{O}(X\varphi)$ does not hold necessarily.

Property 7 (Propagation). A state which does not satisfy any violation of a propositional formula satisfies the propagation property.

If $IdealSameProp(s)$ then

$$s \models \boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\boldsymbol{O}\varphi \wedge \neg\varphi \ \Rightarrow \ X\boldsymbol{O}\psi$$

for φ propositional formula, and ψ any formula.

Proof. The proof is similar to the proof of property 1 in section 3.1, except that we have not the case where every formula is obligatory in the temporal successor of s. □

We still have, as in section 3.1, property 2, a more precise characterization.

Property 8 (Characterization of new obligations). For any formula ψ, if in a state s, which satisfies $\mathit{IdealSameProp}(s)$, both the formulas $X\boldsymbol{O}(\psi)$ and $\neg\boldsymbol{O}(X\psi)$ hold, then there exists a propositional formula φ such that

$$s \models \boldsymbol{O}(\varphi \vee X\psi) \wedge \neg\boldsymbol{O}(\varphi) \wedge \neg\varphi$$

When an obligation appears, it is necessary due to the propagation of some more general obligation in the previous state. So the propagation property completely characterizes the new obligations that appear.

Proof. The proof follows the same idea as the proof of property 2 in section 3.1. □

As said in the introduction of section 3, as a consequence of the general propagation property, if a state s satisfies $\mathit{IdealSameProp}(s)$, then it satisfies the following property of propagation for an obligation with deadline, since $F_{\leqslant k}\varphi \Leftrightarrow \varphi \vee XF_{\leqslant k-1}\varphi$, for $k > 0$:

$$s \models \boldsymbol{O}(F_{\leqslant k}\varphi) \wedge \neg\boldsymbol{O}(\varphi) \wedge \neg\varphi \;\Rightarrow\; X\boldsymbol{O}(F_{\leqslant k-1}\varphi)$$

for any deadline $k > 0$, and φ propositional formula. This property expresses that if it is obligatory to satisfy φ before a deadline k (and it is not obligatory to satisfy it now) then, if φ is not true now, the obligation is propagated.

In a state which does not satisfy $\mathit{IdealSameProp}$, that is, a state which violates an obligation of some propositional formula, the deontic realm of the next state switches to a lower level. We consider that when a state violates the present rules, then they become obsolete. In such a state, $\boldsymbol{O}(X\varphi) \Rightarrow X\boldsymbol{O}(\varphi)$ is not guaranteed, and neither is any link between what is satisfied in the current state, and what is obligatory next.

4 Branching Time Structures

Many proposals to combine temporal and deontic concepts use a branching time structure, where the ideal alternatives are subsets of the possible future worlds [13,2,1,9]. This has several advantages. In particular, the principle “must implies can” is guaranteed. And our “identical past” criterion is automatically satisfied by branching time structures. However, some of the intuitive properties we have discussed in this paper do not necessarily hold in a branching time setting, and may be hard to implement. For instance, we consider that an atomic proposition p can be both true in the current state and false in some ideal state. In pure branching time approaches [1,2] this is usually not possible. Branching time approaches thus have problems modelling immediate obligations. For example, $\boldsymbol{O}(p) \wedge \neg p$ is not satisfiable in such logics.

An exception has to be made for deontic STIT formalisms [13], where moments are partitioned into choices, and where a proposition can be obligatory but not true at the same moment. However, it is quite unclear how to implement a principle of propagation in the conceptually rich setting of STIT models. Actually, we can represent our model in a tree like view as in STIT theory by putting together the histories sharing the same past. In a STIT framework, the moment/history pairs (m, h) would naturally correspond to our states (i, w), and the histories h to our deontic worlds w. To have a branching time view, the states (i, w) that share the same past would be represented by a unique moment m. So two deontic worlds w and w' that share the same past until i would naturally be represented by histories that go through the moment m, where m represents both states (i, w) and (i, w'). But in the STIT framework, there is a valuation of atomic propositions for every moment/history pair. So two histories that go trough the same moment m, also go trough the same moments w' for every $w' < w$, while the valuation they have may differ. Our model can thus be easily translated to a STIT setting under the restriction that two histories sharing the same past moments also share the same valuation on these moments. More formally, for a moment m, and two histories h_1 and h_2,

$$(m \in h_1 \text{ and } m \in h_2) \text{ if and only if } (\forall m' < m \, \forall p \in P \;\; m', h_1 \models p \text{ iff } m', h_2 \models p)$$

The strict operator $<$ allows us to have different valuations for the same moment m depending on the history we consider, if the histories split after this moment. Our framework is close to a STIT framework with this restriction. To characterize a state (called moment in a STIT model) that satisfies $IdealSameProp$ corresponds in such a STIT model to a moment/history pair (m, h) which has the same valuation as some pair (m, h'), where h' is ideal. Such pairs, which would need a formal characterization, would satisfy the propagation properties we have studied.

5 Conclusion

In this paper, we have studied properties concerning the propagation of obligations for the future that have not been fulfilled yet. Because we do not want to consider explicit updates of the deontic realm, we started with a product of temporal and deontic logics, and concluded that to account for propagation properties we had to drop the property of 'no learning'. Going through some "hybrid" semantics, we settled on a semantics with levels of ideality exposing interesting similarities with branching time STIT settings. We characterized the states where obligations are propagated: these are the states in which no immediate obligations are violated. In such states, obligations are not forgotten, and the new obligations are due to the propagation of some past obligations which are not fulfilled yet. On the other hand, in the states which violate immediate obligations, the ideal states switch to a lower level of ideality, and the obligations of the next state do not depend on what is true now.

The propagation of obligations has been studied in the restricted case of dedicated operators for obligations with a deadline, for instance in [5,6,7,9]. But, to our knowledge, the more general propagation property we have focused on is new.

We only considered a linear time setting with one agent. We cannot express "must implies can", nor "it is obligatory to make something possible", nor "it is obligatory for an agent to do something". Therefore it would be interesting to further develop the link with STIT models mentioned in section 4. We plan to formalize it and thus study the propagation of obligations in a framework which allows branching time and multi-agents reasoning.

Another issue is the decidability of our logic. A clue is that the genuine product $LTL \times SDL$ is decidable (see [10] for the decidability of $LTL \times K$), but the decision problem is non elementary.

References

1. Åqvist, L.: Combinations of tense and deontic logic. Journal of Applied Logic 3, 421–460 (2005)
2. Bailhache, P.: Canonical models for temporal deontic logic. Logique et Analyse, 3–21 (1995)
3. Bratman, M.: Intention, plans, and practical reason. Harvard University Press, Cambridge (1987)
4. Bratman, M.: Planning and the stability of intentions. Minds and Machines 2, 1–16 (1992)
5. Broersen, J.: Strategic deontic temporal logic as a reduction to ATL, with an application to Chisholm's scenario. In: Goble, L., Meyer, J.-J.C. (eds.) DEON 2006. LNCS (LNAI), vol. 4048, pp. 53–68. Springer, Heidelberg (2006)
6. Brunel, J., Bodeveix, J.-P., Filali, M.: A state/event temporal deontic logic. In: Goble, L., Meyer, J.-J.C. (eds.) DEON 2006. LNCS (LNAI), vol. 4048, pp. 85–100. Springer, Heidelberg (2006)
7. Cuppens, F., Cuppens-Boulahia, N., Sans, T.: Nomad: a security model with non atomic actions and deadlines. In: Proceedings of the 18th IEEE Computer Security Foundations Workshop (June 2005)
8. Dastani, M., van Riemsdijk, B., Dignum, F., Meyer, J.: A programming language for cognitive agents: Goal directed 3APL. In: PROMAS 2003. ACM Press, New York (2003)
9. Demolombe, R., Bretier, P., Louis, V.: Formalisation de l'obligation de faire avec délais. In: Proc. Journées Francophones sur la Modélisation Formelle de l'Interaction, Caen (2005)
10. Gabbay, D., Kurucz, A., Wolter, F., Zakharyachev, M.: Many-Dimensional Modal Logics: Theory and Applications. Elsevier, Amsterdam (2003)
11. Hilpinen, R.: New studies in deontic logic. Reidel (1981)
12. Hindriks, K.V., de Boer, F.S., van der Hoek, W., Meyer, J.-J.C.: Agent programming in 3apl. International Journal of Autonomous Agents and Multi-Agent Systems 2(4), 357–401 (1999)
13. Horty, J.: Agency and Deontic Logic. Oxford University Press, Oxford (2001)
14. Pnueli, A.: The temporal semantics of concurrent programs. Theoretical Computer Science 13, 45–60 (1981)

15. Rao, A., Georgeff, M.: Modeling rational agents within a BDI-architecture. In: Allen, J., Fikes, R., Sandewall, E. (eds.) Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR 1991), pp. 473–484. Morgan Kaufmann Publishers, San Francisco (1991)
16. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: van Hoe, R. (ed.) Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands (1996)
17. Ross, A.: Imperatives and logic. Theoria 7, 53–71 (1941)
18. Wright, G.v.: Deontic logic. Mind 60, 1–15 (1951)

Proof Theory for Distributed Knowledge

Raul Hakli and Sara Negri

Department of Philosophy
P.O. Box 9 (Siltavuorenpenger 20 A)
FIN-00014 University of Helsinki, Finland
{Raul.Hakli,Sara.Negri}@Helsinki.Fi

Abstract. The proof theory of multi-agent epistemic logic extended with operators for distributed knowledge is studied. A proposition A is distributed knowledge within a group G if A follows from the totality of what the individual members of G know. There are known axiomatizations for epistemic logics with the distributed knowledge operator, but apparently no cut-free proof system for such logics has yet been presented. A Gentzen-style contraction-free sequent calculus system for propositional epistemic logic with operators for distributed knowledge is given, and a cut-elimination theorem for the system is proved. Examples of reasoning about distributed knowledge that use the calculus are given.

1 Introduction

Distributed knowledge is usually characterized by saying that A is distributed knowledge within a group G if A follows from the totality of what the individual members of G know. For instance, A is distributed knowledge in group G (denoted $\mathcal{D}_G A$) consisting of three agents of which the first one knows B, the second one knows $B \supset C$, and the third one knows $B \,\&\, C \supset A$. Reasoning about the combined information possessed by different agents is an important task in multi-agent systems in which all information is not available in one central source but distributed among several agents.

In such situations, epistemic logic [1] is typically used for representing and reasoning about knowledge. In the literature concerning multi-agent epistemic logics, e.g. [2,3], operators for distributed knowledge are often included. However, these treatments usually concentrate on the model theory of the logics, whereas the proof-theoretical part is limited to providing Hilbert-style axiomatizations. Since theorem-proving is difficult in Hilbert-style systems, we shall here study Gentzen-style sequent calculi as a step towards mechanization of proof search.

One proof-theoretical approach to reasoning about distributed knowledge is given in [4], but the approach is different because of the use of natural deduction instead of sequent calculus and context-based logic instead of epistemic logic. The development of a proof system for logic of distributed knowledge has been recently posed by S. Artemov as an open problem for the system of *evidence-based knowledge* (see [5]). This paper presents a solution for ordinary multi-agent epistemic logic by the methods developed in [6,7].

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 100–116, 2008.

Formal systems for drawing inferences in distributed knowledge may be useful in several application areas that attempt to combine knowledge of agents, such as cooperative problem solving, knowledge base merging, and judgement aggregation. In cooperative problem solving it is usually assumed that the agents are willing to provide any information they have and that all information is certain. In such situations, it is possible to combine the separate knowledge bases into one and then derive theorems from the large knowledge base. However, typically in knowledge base merging the data can contain errors, and is thus not strictly speaking knowledge. The combination of information from multiple sources may then lead to an inconsistent knowledge base, and special methods have to be used for dealing with contradictory information (see, e.g. [8,9,10]). These methods often involve discarding some information in order to maintain the integrity of the database.

In cases with heterogeneous information sources, the knowledge modalities should not be understood as knowledge proper but rather as beliefs. In open information systems, and in situations involving strategic considerations, such as in judgement aggregation or voting, agents can even provide false information on purpose, so it is not possible to infer their real beliefs from what they report, but the information they provide must be treated as claims, acceptances or just as messages with propositional content.

The introduction of the knowledge modalities and the modality for distributed knowledge into the logical language can be beneficial, because the management of the meta-information concerning the sources of knowledge and their various combinations becomes easier. When the source of information is stored in addition to the content, also contradictory information can be dealt with: If agent 1 claims that A is the case and agent 2 claims that not-A is the case, the receiving agent should decide which piece of information to accept and which one to reject. However, when such a situation arises there may not be enough information available for resolving the conflict. If our language is rich enough to allow also knowledge propositions and the agents are able to reason about distributed knowledge, incoming information need not be discarded nor is it necessary to immediately judge some agents unreliable. Instead, we can store the knowledge claims without violating the integrity constraints, and we can use the stored information to find out which agents we can trust, possibly later when we have gathered more information.

Thus, the addition of the knowledge operators to the language makes it possible for the agents to perform reasoning about the distributed information possessed by various agents and groups of agents and to detect inconsistencies between claims made by agents. Also, the possibility to iterate knowledge operators allows for more complex reasoning tasks than reasoning from an integrated knowledge base without iterated modalities. Reasoning of this type may be used in cooperative information systems to find out which agents have useful information with respect to the task at hand.

In Section 2, we introduce the logical system and show that it can be used to derive the axioms given in complete axiomatizations for the logic of distributed knowledge. In Section 3, we show that the system has the required structural properties such as admissibility of the structural rules, and discuss the relevance of these results for proof search. In Section 4, we present examples of derivations in our calculus and discuss possible application areas of our methods. In particular, we show in a concrete case that if a formula is in the deductive closure of the totality of what the agents individually know, then it is derivable as distributed knowledge in our system; we present the same-birthday example from [11] within our formalism; we show how proof search can be used for discarding unreliable information sources and for performing co-operative problem solving. Finally, after summarizing our results, we discuss in Section 5 the possibility of extending our system with the principle of full communication mentioned in [12].

2 Logic of Distributed Knowledge

Our starting point is the modal sequent calculus system **G3K** [7]. (For a general introduction to Gentzen-style sequent calculus, see [6].) We replace the modal operator $\Box$ with the knowledge operators $\mathcal{K}_a$ for individual agents $a \in G$. We extend the logic with the operator $\mathcal{D}_G$ with the intended meaning for $\mathcal{D}_G A$ that A is distributed knowledge within the group G (sometimes, for ease of readability, the subscript G will be omitted when clear from the context).

In [7] the rules for $\Box$ are determined by the forcing relation of Kripke semantics

$$x \Vdash \Box A \text{ iff } \forall y (xRy \rightarrow y \Vdash A)$$

where x, y range in the set of possible worlds and R is the accessibility relation. In multi-agent epistemic logic there is an accessibility relation R_a for each agent a, and validity of $\mathcal{K}_a A$ is defined by

$$x \Vdash \mathcal{K}_a A \text{ iff } \forall y (xR_a y \rightarrow y \Vdash A).$$

The right to left direction of the equivalence gives the right rule of $\mathcal{K}_a$, the opposite, the left rule. We shall use colon ':' to stand for the forcing relation (so $x : A$ can be read as saying that A holds at world x). In general, sequents of the form $\Gamma \Rightarrow \Delta$ can be understood as saying that the disjunction of the formulas in the multiset Δ can be derived from the conjunction of formulas in the multiset Γ representing the open assumptions. The rules are

$$\frac{xR_a y, \Gamma \Rightarrow \Delta, y : A}{\Gamma \Rightarrow \Delta, x : \mathcal{K}_a A}\, R\mathcal{K}_a \qquad\qquad \frac{y : A, x : \mathcal{K}_a A, xR_a y, \Gamma \Rightarrow \Delta}{x : \mathcal{K}_a A, xR_a y, \Gamma \Rightarrow \Delta}\, L\mathcal{K}_a$$

Rule $R\mathcal{K}_a$ has the variable condition that y must not appear in the conclusion.

In possible world semantics, distributed knowledge of A in G is (usually but not always, see [12,13]) taken to hold if and only if A holds in every world that every agent in G considers possible. Thus, distributed knowledge is defined as follows (see e.g. [2]) w.r.t. a Kripke structure M and a world s

$$(M, s) \models \mathcal{D}_G A \text{ iff } (M, t) \models A \text{ for all } t \text{ such that } (s, t) \in \bigcap_{a \in G} R_a.$$

The rules for distributed knowledge are found accordingly

$$\frac{\{xR_a y\}_{a\in G}, \Gamma \Rightarrow \Delta, y : A}{\Gamma \Rightarrow \Delta, x : \mathcal{D}_G A}\ {\scriptstyle R\mathcal{D}_G} \qquad \frac{y : A, x : \mathcal{D}_G A, \{xR_a y\}_{a\in G}, \Gamma \Rightarrow \Delta}{x : \mathcal{D}_G A, \{xR_a y\}_{a\in G}, \Gamma \Rightarrow \Delta}\ {\scriptstyle L\mathcal{D}_G}$$

Similarly to rule $R\mathcal{K}_a$, also rule $R\mathcal{D}_G$ has the restriction that y must not appear in the conclusion. The intended meaning of the notation $\{xR_a y\}_{a\in G}$ is that what is inside the curly brackets should be repeated for each agent $a \in G$; for instance, with a group of two agents 1 and 2, the right rule becomes

$$\frac{xR_1 y, xR_2 y, \Gamma \Rightarrow \Delta, y : A}{\Gamma \Rightarrow \Delta, x : \mathcal{D}_{\{1,2\}} A}\ {\scriptstyle R\mathcal{D}_{\{1,2\}}}$$

The rules for the calculus are given in Table 1. Observe that initial sequents are restricted to atomic formulas P. This feature, common to all **G3** systems of sequent calculus, is needed in order to ensure invertibility of the rules and other structural properties. Note also that no rules for negation nor equivalence are needed because we take $\sim A$ to be a shorthand for $A \supset \bot$ and $A \supset\subset B$ as a shorthand for $(A \supset B) \,\&\, (B \supset A)$.

In addition to these rules, the properties of the agents' accessibility relations can be chosen by adding to the system suitable rules corresponding to desired properties, as explained in [7]. The common choices in the case of epistemic logic are reflexivity (which guarantees that the actual world is always taken to be epistemically possible so that nothing false can be known) and transitivity

Table 1. System $\mathbf{G3KE}_D$

Initial sequents:

$$x : P, \Gamma \Rightarrow \Delta, x : P$$

Propositional rules:

$$\frac{x : A, x : B, \Gamma \Rightarrow \Delta}{x : A \,\&\, B, \Gamma \Rightarrow \Delta}\ {\scriptstyle L\&} \qquad \frac{\Gamma \Rightarrow \Delta, x : A \quad \Gamma \Rightarrow \Delta, x : B}{\Gamma \Rightarrow \Delta, x : A \,\&\, B}\ {\scriptstyle R\&}$$

$$\frac{x : A, \Gamma \Rightarrow \Delta \quad x : B, \Gamma \Rightarrow \Delta}{x : A \vee B, \Gamma \Rightarrow \Delta}\ {\scriptstyle L\vee} \qquad \frac{\Gamma \Rightarrow \Delta, x : A, x : B}{\Gamma \Rightarrow \Delta, x : A \vee B}\ {\scriptstyle R\vee}$$

$$\frac{\Gamma \Rightarrow \Delta, x : A \quad x : B, \Gamma \Rightarrow \Delta}{x : A \supset B, \Gamma \Rightarrow \Delta}\ {\scriptstyle L\supset} \qquad \frac{x : A, \Gamma \Rightarrow \Delta, x : B}{\Gamma \Rightarrow \Delta, x : A \supset B}\ {\scriptstyle R\supset}$$

$$\frac{}{x : \bot, \Gamma \Rightarrow \Delta}\ {\scriptstyle L\bot}$$

Modal rules:

$$\frac{y : A, x : \mathcal{K}_a A, xR_a y, \Gamma \Rightarrow \Delta}{x : \mathcal{K}_a A, xR_a y, \Gamma \Rightarrow \Delta}\ {\scriptstyle L\mathcal{K}_a} \qquad \frac{xR_a y, \Gamma \Rightarrow \Delta, y : A}{\Gamma \Rightarrow \Delta, x : \mathcal{K}_a A}\ {\scriptstyle R\mathcal{K}_a}$$

$$\frac{y : A, x : \mathcal{D}_G A, \{xR_a y\}_{a\in G}, \Gamma \Rightarrow \Delta}{x : \mathcal{D}_G A, \{xR_a y\}_{a\in G}, \Gamma \Rightarrow \Delta}\ {\scriptstyle L\mathcal{D}_G} \qquad \frac{\{xR_a y\}_{a\in G}, \Gamma \Rightarrow \Delta, y : A}{\Gamma \Rightarrow \Delta, x : \mathcal{D}_G A}\ {\scriptstyle R\mathcal{D}_G}$$

(which gives the property of positive introspection: if an agent knows something then she knows that she knows). These together yield an **S4**-based epistemic system. In the case of doxastic logic, that is, the logic of belief, reflexivity is abandoned to allow the possibility of false beliefs. Sometimes also symmetry (which together with transitivity gives negative introspection: If an agent does not know something, she knows that she does not know it) is added, in which case the accessibility relations are equivalence relations and the resulting system is then based on what is known as **S5**. The rules corresponding to reflexivity, transitivity and symmetry for agent a are, respectively

$$\frac{xR_a x, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\, {\scriptstyle Ref_a} \quad \frac{xR_a z, xR_a y, yR_a z, \Gamma \Rightarrow \Delta}{xR_a y, yR_a z, \Gamma \Rightarrow \Delta}\, {\scriptstyle Trans_a} \quad \frac{yR_a x, xR_a y, \Gamma \Rightarrow \Delta}{xR_a y, \Gamma \Rightarrow \Delta}\, {\scriptstyle Sym_a}$$

Observe that the rules have active and principal formulas in the antecedents of sequents, so they correspond to implication from the atoms in the conclusion to those in premisses.

Our system is modular in the sense that one need not be committed to a particular set of properties for the accessibility relations but the results given in this paper hold for accessibility relations with any combinations of these properties. Also other properties can be used as explained in [7]. It is also possible to have several modalities in one system without losing the good structural properties of the system. For example, knowledge and belief can be treated simultaneously by adding suitable rules for the belief operators and the doxastic accessibility relation for each agent. The relationship between modalities may require new rules, like in this case a rule for ensuring that the doxastic accessibility relation is included in the epistemic accessibility relation corresponding to the idea that knowledge entails belief. Temporal modalities can be added in a similar fashion. In the examples presented in this paper, we shall not combine different information attitudes so we can just use one type of modal operator $\mathcal{K}_a$ (specific to each agent a) to stand for whichever modality is appropriate in the situation. Similarly, the operator $\mathcal{D}_G$ is taken to mean a distributed version of the $\mathcal{K}$-modality, be it knowledge, belief, or something else.

According to [12], the intuitive characterization of distributed knowledge as everything that follows from the combined knowledge of the individual agents does not always coincide with the semantic characterization based on the intersection of the agents' accessibility relations. In particular, if the number of possible worlds is infinite or if there are worlds that cannot be distinguished from each other using the logical language, what they call the principle of full communication may fail. This means that the agents are not able to express their full epistemic state to the other agents and thus the combination of the sentences they know may not convey all the information that would result from taking the intersection of the agents' accessibility relations (see [12], [13]). As explained in Section 5, our system can be extended so as to include the principle of full communication.

As shown in [7], the standard axiomatic sequents $x : A, \Gamma \Rightarrow \Delta, x : A$, for arbitrary, not just atomic, A, and the characteristic axioms of the standard modal logics are derivable in the respective sequent calculus systems, and the necessitation rule is admissible. These results extend to multi-agent epistemic logic with knowledge generalization rules for each agent $a \in G$

$$\frac{\Rightarrow x : A}{\Rightarrow x : K_a A}$$

The addition of distributed knowledge operator requires new axioms. A sound and complete axiomatization for the epistemic logic based on **S5** with distributed knowledge is provided in [14], [15], and in [3]. The axioms to be added to standard axiomatizations of epistemic logics would be the following:

$$\mathcal{K}_a A \supset \mathcal{D}_G A, \text{ for each agent } a \in G \tag{1}$$

and

$$(\mathcal{D}_G A \,\&\, \mathcal{D}_G(A \supset B)) \supset \mathcal{D}_G B. \tag{2}$$

In order to demonstrate the completeness of our calculus, we first show that these axioms are derivable.

Proposition 1. *$\mathcal{K}_a A \supset \mathcal{D}_G A$ is derivable for each agent $a \in G$ in **G3KE**$_D$.*

Proof. For each agent $a \in G$, the derivation goes as follows:

$$\cfrac{\cfrac{\cfrac{y : A, \{xR_a y\}_{a\in G}, x : \mathcal{K}_a A \Rightarrow y : A}{\{xR_a y\}_{a\in G}, x : \mathcal{K}_a A \Rightarrow y : A}\,{\scriptstyle L\mathcal{K}_a}}{x : \mathcal{K}_a A \Rightarrow x : \mathcal{D}_G A}\,{\scriptstyle R\mathcal{D}_G}}{\Rightarrow x : \mathcal{K}_a A \supset \mathcal{D}_G A}\,{\scriptstyle R\supset}$$

where the uppermost sequent is derivable.

Proposition 2. *$(\mathcal{D}_G A \,\&\, \mathcal{D}_G(A \supset B)) \supset \mathcal{D}_G B$ is derivable in **G3KE**$_D$.*

Proof. The derivation is:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{y : A, \ldots \Rightarrow y : B, y : A \qquad y : B, \ldots \Rightarrow y : B}{y : A \supset B, y : A, \{xR_a y\}_{a\in G}, x : \mathcal{D}_G A, x : \mathcal{D}_G(A \supset B) \Rightarrow y : B}\,{\scriptstyle L\supset}}{y : A, \{xR_a y\}_{a\in G}, x : \mathcal{D}_G A, x : \mathcal{D}_G(A \supset B) \Rightarrow y : B}\,{\scriptstyle L\mathcal{D}_G}}{\{xR_a y\}_{a\in G}, x : \mathcal{D}_G A, x : \mathcal{D}_G(A \supset B) \Rightarrow y : B}\,{\scriptstyle L\mathcal{D}_G}}{x : \mathcal{D}_G A, x : \mathcal{D}_G(A \supset B) \Rightarrow x : \mathcal{D}_G B}\,{\scriptstyle R\mathcal{D}_G}}{x : \mathcal{D}_G A \,\&\, \mathcal{D}_G(A \supset B) \Rightarrow x : \mathcal{D}_G B}\,{\scriptstyle L\&}}{\Rightarrow x : (\mathcal{D}_G A \,\&\, \mathcal{D}_G(A \supset B)) \supset \mathcal{D}_G B}\,{\scriptstyle R\supset}$$

where the uppermost sequents are derivable.

Completeness with respect to the mentioned Hilbert-style system further requires closure under modus ponens and under the necessitation rules for epistemic operators. These properties are shown in the following section.

In some applications it is useful to be able to reason about shared knowledge, that is, something that all the agents know. It is straightforward to add to the calculus an operator $\mathcal{E}$ for shared knowledge: Since $\mathcal{E}_G A$ means that everyone in group G knows that A, it can be used as a short-hand expression for the conjunction $\mathcal{K}_{a_1} A \,\&\, \ldots \,\&\, \mathcal{K}_{a_n} A$ where $G = \{a_1, \ldots, a_n\}$. The right and left rules for shared knowledge are thus not required for the calculus but can be derived from the K_i rules. These are as follows

$$\frac{\{xR_a y_a, \Gamma \Rightarrow \Delta, y_a : A\}_{a \in G}}{\Gamma \Rightarrow \Delta, x : \mathcal{E}_G A}\ {\scriptstyle R\mathcal{E}_G} \qquad \frac{y_a : A, x : \mathcal{E}_G A, xR_a y_a, \Gamma \Rightarrow \Delta}{x : \mathcal{E}_G A, xR_a y_a, \Gamma \Rightarrow \Delta}\ {\scriptstyle L\mathcal{E}_G \quad (a \in G)}$$

with the variable condition in $R\mathcal{E}_G$ that no variable y_a appears in the conclusion.

3 Structural Properties

We shall now proceed with the structural properties of our system. The use of variables referring to possible worlds requires that we define substitution and prove a substitution lemma as in [7]. Substitution is defined as follows:

$$\begin{aligned}
&xR_a y(z/w) \equiv xR_a y \text{ if } w \neq x \text{ and } w \neq y,\\
&xR_a y(z/x) \equiv zR_a y \text{ if } x \neq y,\\
&xR_a y(z/y) \equiv xR_a z \text{ if } x \neq y,\\
&xR_a x(z/x) \equiv zR_a z,\\
&x : A(z/y) \equiv x : A \text{ if } x \neq y,\\
&x : A(z/x) \equiv z : A
\end{aligned}$$

for all $a \in G$. Substitution in multisets is defined componentwise.

Lemma 1 (Substitution lemma). *If $\Gamma \Rightarrow \Delta$ is derivable in $\boldsymbol{G3KE}_D$, then also $\Gamma(y/x) \Rightarrow \Delta(y/x)$ is derivable, with the same derivation height.*

Proof. The proof is by induction on the height n of the derivation of $\Gamma \Rightarrow \Delta$ as in [7]. If $n = 0$ and the substitution y/x is not vacuous, the sequent $\Gamma \Rightarrow \Delta$ is either an initial sequent or conclusion of $L\bot$. In either case $\Gamma(y/x) \Rightarrow \Delta(y/x)$ is also an initial sequent of the same form or conclusion of $L\bot$. Suppose then that the claim holds for derivations of height n and consider the last rule applied in the derivation. If the last rule is a propositional rule or a modal rule without variable conditions, apply the inductive hypothesis to the premisses and then apply the rule. If the last rule is a rule with a variable condition ($R\mathcal{K}_a$ or $R\mathcal{D}_G$), we must be careful with the the cases in which either x or y is the eigenvariable of the rule, because a straightforward substitution may result in a violation of the restriction. In those cases we must apply the inductive hypothesis to the premiss and replace the eigenvariable with a fresh variable that does not appear in the derivation. The details are omitted here but similar cases are considered in [7, Lemma 4.3].

Theorem 1 (Height-preserving weakening). *The rules of weakening*

$$\frac{\Gamma \Rightarrow \Delta}{x : A, \Gamma \Rightarrow \Delta}\ {\scriptstyle LW} \qquad \frac{\Gamma \Rightarrow \Delta}{xR_a y, \Gamma \Rightarrow \Delta}\ {\scriptstyle LW_{R_a}} \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, x : A}\ {\scriptstyle RW}$$

are height-preserving admissible in $\boldsymbol{G3KE}_D$.

Proof. The proof is by induction on the height of the derivation of the premiss. The cases with propositional rules and the modal and nonlogical rules without variable conditions are straightforward. As in [7], if the last step is a rule with a variable condition ($R\mathcal{K}_a$ or $R\mathcal{D}$), we need to apply the substitution lemma to the premisses of the rule in order to avoid a clash with the variables in $x : A$ or xR_ay. The conclusion is then obtained by applying the inductive hypothesis and the rule in question.

Theorem 2. *The necessitation rules*

$$\frac{\Rightarrow x : A}{\Rightarrow x : \mathcal{K}_a A} \qquad \frac{\Rightarrow x : A}{\Rightarrow x : \mathcal{D}_G A}$$

are admissible in $\boldsymbol{G3KE}_D$.

Proof. Suppose we have a derivation of $\Rightarrow x : A$. By the substitution lemma we obtain a derivation of $\Rightarrow y : A$ and, by admissibility of weakening, of $xR_ay \Rightarrow y : A$, and $\{xR_ay\}_{a\in G} \Rightarrow y : A$. By $R\mathcal{K}_a$ and $R\mathcal{D}$, respectively, we have $\Rightarrow x : \mathcal{K}_a A$ and $\Rightarrow x : \mathcal{D}_G A$.

Theorem 3. *The rules of* $\boldsymbol{G3KE}_D$ *are height-preserving invertible.*

Proof. For the propositional rules, the proof is exactly as the proof of height-preserving invertibility of the rules of **G3c** in [6, Theorem 3.1.1]. For the $\mathcal{K}$-rules and rules for the accessibility relations, the proof is similar to [7, Proposition 4.11]. Invertibility of LD_G is immediate because the premiss can be obtained from the conclusion by (height-preserving) weakening.

Invertibility of $R\mathcal{D}_G$ is proved by induction on the height n of the derivation of the conclusion $\Gamma \Rightarrow \Delta, x : \mathcal{D}_G A$. If $n = 0$, it is an axiom or conclusion of $L\bot$ and so is the premiss $\{xR_ay\}_{a\in G}, \Gamma \Rightarrow \Delta, y : A$. If $n > 0$ and $\Gamma \Rightarrow \Delta, x : \mathcal{D}_G A$ is concluded by a rule other than $R\mathcal{K}_a$ or $R\mathcal{D}_G$ (which have a variable condition), we apply the inductive hypothesis to the premiss(es) and the rule. If the rule is $R\mathcal{K}_a$, we have a derivation ending with

$$\frac{xR_aw, \Gamma \Rightarrow \Delta, x : \mathcal{D}_G A, w : A}{\Gamma \Rightarrow \Delta, x : \mathcal{D}_G A, x : \mathcal{K}_a A}\ {\scriptstyle R\mathcal{K}_a}$$

We can assume that the eigenvariable w is different from y, otherwise we can apply the substitution lemma. Now the inductive hypothesis applied to the premiss gives a derivation of the same height ending with

$$\frac{\{xR_ay\}_{a\in G}, xR_aw, \Gamma \Rightarrow \Delta, w : A, y : A}{\{xR_ay\}_{a\in G}, \Gamma \Rightarrow \Delta, x : \mathcal{K}_a A, y : A}\ {\scriptstyle R\mathcal{K}_a}$$

The case in which the conclusion was derived using $R\mathcal{D}_G$ and the principal formula is in Δ is similar. In case the principal formula was $x : \mathcal{D}_G A$ itself, the premiss is already the sequent we wanted to prove derivable, except for the possibly different eigenvariable, which can be changed by height-preserving substitution.

Invertibility of the rules is useful for theoretical purposes because it simplifies some other proofs but it is crucial for the practical reason that root-first proof search requires no backtracking mechanism if all the rules are invertible: If we find out in our current branch of a proof tree that a sequent is not derivable we can immediately infer that the proof search has failed and can be terminated because, by invertibility, the conclusion cannot be derivable either.

Theorem 4. *The rules of contraction*

$$\frac{x:A, x:A, \Gamma \Rightarrow \Delta}{x:A, \Gamma \Rightarrow \Delta}\,{\scriptstyle LCtr} \qquad \frac{xR_ay, xR_ay, \Gamma \Rightarrow \Delta}{xR_ay, \Gamma \Rightarrow \Delta}\,{\scriptstyle LCtr_{R_a}} \qquad \frac{\Gamma \Rightarrow \Delta, x:A, x:A}{\Gamma \Rightarrow \Delta, x:A}\,{\scriptstyle RCtr}$$

are height-preserving admissible in $\boldsymbol{G3KE}_D$*.*

Proof. By simultaneous induction on the height of derivation for left and right contractions. In the base case, observe that an initial sequent stays initial if two occurrences of a formula are contracted into one. For the inductive step, three cases are distinguished: The case with none of the contraction formulas principal in the last rule, the case with one principal, and (only for $RCtr_{R_a}$) the case with both principal. In the former, apply inductive hypothesis to the premiss of the rule, then the rule. In the latter, apply the matching height-preserving inversion to the premiss(es) of the rule, the inductive hypothesis, and the rule. In the third, the *closure condition*, as explained in detail in [7].

Also admissibility of contraction is useful for the practical reason that it guarantees that we need not multiply formulas in sequents during the proof search. If a sequent can be derived using contraction, it can be derived without using it. In addition, height-preserving admissibility of contraction permits the restriction of the search space also with respect to other rules: Whenever application of a rule, root-first, produces a duplication, by height-preserving admissibility of contraction the conclusion of the rule can be obtained in one step less. The possible applicable rule can thus be discarded if we reasonably assume that the derivation we are looking for is a minimal one, i.e. one that does not admit any local shortening through the elimination of contraction steps.

Theorem 5. *The cut rule*

$$\frac{\Gamma \Rightarrow \Delta, C \quad C, \Gamma' \Rightarrow \Delta'}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\,{\scriptstyle Cut}$$

is admissible in $\boldsymbol{G3KE}_D$*.*

Proof. The proof proceeds by induction on the structure of the cut formula C with subinduction on the *cut-height*, that is, the sum of the heights of the derivations of the premisses. The proof is to a large extent similar to the cut-elimination proofs in [6] (e.g. Theorem 3.2.3) so we shall consider in detail only the case in which the cut formula is $\mathcal{D}_G A$ and is principal in both premisses:

$$\frac{\dfrac{\{xR_ay\}_{a\in G}, \Gamma \Rightarrow \Delta, y:A}{\Gamma \Rightarrow \Delta, x:\mathcal{D}_G A}\,{\scriptstyle R\mathcal{D}} \qquad \dfrac{z:A, x:\mathcal{D}_G A, \{xR_az\}_{a\in G}, \Gamma' \Rightarrow \Delta'}{x:\mathcal{D}_G A, \{xR_az\}_{a\in G}, \Gamma' \Rightarrow \Delta'}\,{\scriptstyle L\mathcal{D}}}{\Gamma, \{xR_az\}_{a\in G}, \Gamma' \Rightarrow \Delta, \Delta'}\,{\scriptstyle Cut}$$

Let n be the height of the derivation of the left premiss and m the height of the second. Then the cut-height is $n+1+m+1$. This derivation can be transformed into the following:

$$\dfrac{\dfrac{\dfrac{\{xR_ay\}_{a\in G},\Gamma\Rightarrow\Delta,y:A}{\{xR_az\}_{a\in G},\Gamma\Rightarrow\Delta,z:A}{\scriptstyle Subst}\qquad\dfrac{\dfrac{\{xR_ay\}_{a\in G},\Gamma\Rightarrow\Delta,y:A}{\Gamma\Rightarrow\Delta,x:\mathcal{D}_GA}{\scriptstyle R\mathcal{D}}\quad z:A,x:\mathcal{D}_GA,\{xR_az\}_{a\in G},\Gamma'\Rightarrow\Delta'}{z:A,\Gamma,\{xR_az\}_{a\in G},\Gamma'\Rightarrow\Delta,\Delta'}{\scriptstyle Cut}}{\{xR_az\}_{a\in G},\Gamma,\Gamma,\{xR_az\}_{a\in G},\Gamma'\Rightarrow\Delta,\Delta,\Delta'}{\scriptstyle Cut}}{\Gamma,\{xR_az\}_{a\in G},\Gamma'\Rightarrow\Delta,\Delta'}{\scriptstyle Ctr^*}$$

Note that the height-preserving substitution in the derivation of the left premiss of the second cut has no effect on Γ or Δ because, by the variable restriction of rule $R\mathcal{D}$ used in the original derivation, y does not appear free in Γ or Δ. The derivation has two cuts, the first of which has lower height and the second smaller size of the cut formula.

As a consequence of admissibility of cut, it follows that our system is closed under modus ponens, and therefore it is complete with respect to the known Hilbert-type systems for the logic of distributed knowledge.

In [3], an alternative Hilbert-type system is presented; the system is obtained by adding to the standard axiomatizations of **T**, **S4**, or **S5**, the rule

$$\frac{A_1\&\dots\&A_m\supset B}{\mathcal{K}_{a_1}A_1\&\dots\&\mathcal{K}_{a_m}A_m\supset\mathcal{D}_GB}\tag{3}$$

where $a_1,\dots,a_m$ are the agents in G. The labelled version of the rule is shown admissible in our system as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Rightarrow x:A_1\&\dots\&A_m\supset B}{x:A_1,\dots,x:A_m\Rightarrow x:B}{\scriptstyle L\&\text{-}Inv,R\supset\text{-}Inv}}{y:A_1,\dots,y:A_m\Rightarrow y:B}{\scriptstyle Subst}}{\{xR_{a_i}y\}_{a_i\in G},y:A_1,\dots,y:A_m,x:\mathcal{K}_{a_1}A_1,\dots,x:\mathcal{K}_{a_m}A_m\Rightarrow y:B}{\scriptstyle LW^*}}{\{xR_{a_i}y\}_{a_i\in G},x:\mathcal{K}_{a_1}A_1,\dots,x:\mathcal{K}_{a_m}A_m\Rightarrow y:B}{\scriptstyle L\mathcal{K}_{a_1},\dots,L\mathcal{K}_{a_m}}}{x:\mathcal{K}_{a_1}A_1,\dots,x:\mathcal{K}_{a_m}A_m\Rightarrow x:\mathcal{D}_GB}{\scriptstyle R\mathcal{D}}}{x:\mathcal{K}_{a_1}A_1\&\dots\&\mathcal{K}_{a_m}A_m\Rightarrow x:\mathcal{D}_GB}{\scriptstyle L\&^*}}{x:\mathcal{K}_{a_1}A_1\&\dots\&\mathcal{K}_{a_m}A_m\supset\mathcal{D}_GB}{\scriptstyle R\supset}$$

where *L&-Inv*, *R⊃-Inv* denote the (admissible) invertibilities of $L\&$ and $R\supset$, respectively, *Subst* the admissible rule of substitution, and the asterisk indicates possibly repeated applications of a rule.

Admissibility of cut is crucial for delimiting the space of proof search, because it guarantees that no arbitrary new formulas need to be constructed during the search. However, our system does not enjoy a full subformula property because some rules remove atoms, but a *weak form of subformula property*, that is, all formulas in a derivation are either subformulas of (formulas in) the endsequent or atomic formulas of the form xRy. By considering minimal derivations, that is, derivations in which shortenings are not possible, the weak subformula property can be strengthened by restricting the labels that can appear in the relational atoms to those in the conclusion or to eigenvariables (*subterm property*). This property, together with height-preserving admissibility of contraction, ensures the consequences of the full subformula property and has been used for establishing decidability through terminating proof search for the system **G3K** and several extensions in [7]. The proofs are involved so we shall not consider the issue for $\mathbf{G3KE}_D$ here but leave it to future work. However, we do not expect problems from the addition of the rules for distributed knowledge.

4 Examples

As a simple example of reasoning about distributed knowledge, consider the case of three agents mentioned in the beginning of the article. Suppose we have encoded the initial situation to a knowledge base KB so that it consists of the formulas $x : \mathcal{K}_1 B, x : \mathcal{K}_2(B \supset C), x : \mathcal{K}_3(B \,\&\, C \supset A)$. We can now ask whether $x : \mathcal{D}_{\{1,2,3\}} A$ can be derived from the knowledge base. (We shall use for clarity $\mathcal{D}$ instead of $\mathcal{D}_{\{1,2,3\}}$). Proceeding in a root-first fashion we get the following derivation in which the uppermost sequents have the same formula on both sides of the sequent arrow and are thus derivable:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\ldots, y : B, \ldots \Rightarrow y : A, y : B \qquad \dfrac{\dfrac{y : C, y : B \ldots \Rightarrow \ldots, y : B \quad y : C \ldots \Rightarrow \ldots, y : C}{y : C, y : B \ldots \Rightarrow y : A, y : B \,\&\, C}{\scriptstyle R\&} \qquad y : A, y : C, \ldots \Rightarrow y : A}{y : C, y : B \,\&\, C \supset A, y : B, \ldots \Rightarrow y : A}{\scriptstyle L\supset}}{y : B \,\&\, C \supset A, y : B \supset C, y : B, \ldots \Rightarrow y : A}{\scriptstyle L\supset}}{y : B \supset C, y : B, xR_1y, xR_2y, xR_3y, x : K_1B, x : \mathcal{K}_2(B \supset C), x : \mathcal{K}_3(B \,\&\, C \supset A) \Rightarrow y : A}{\scriptstyle L\mathcal{K}_3}}{y : B, xR_1y, xR_2y, xR_3y, x : K_1B, x : \mathcal{K}_2(B \supset C), x : \mathcal{K}_3(B \,\&\, C \supset A) \Rightarrow y : A}{\scriptstyle L\mathcal{K}_2}}{xR_1y, xR_2y, xR_3y, x : K_1B, x : \mathcal{K}_2(B \supset C), x : \mathcal{K}_3(B \,\&\, C \supset A) \Rightarrow y : A}{\scriptstyle L\mathcal{K}_1}}{x : K_1B, x : \mathcal{K}_2(B \supset C), x : \mathcal{K}_3(B \,\&\, C \supset A) \Rightarrow x : \mathcal{D}A}{\scriptstyle R\mathcal{D}}$$

Next we shall consider the derivation of the birthday case mentioned e.g. in [11]: In any group it is distributed knowledge whether two agents have the same birthday. The assumption is, of course, that everyone knows one's own birthday. If these pieces of individual knowledge were combined, it would be easy to verify whether the birthdays of any two agents are identical. We shall here consider only a group of two people but the extension would be straightforward.

Take $P(i, t)$ to be the proposition that agent i's birthday is t and consider the proposition that it is distributed knowledge in group G whether two agents in G have the same birthday. This could be expressed as follows:

$$\mathcal{D}_G \exists i, j, t\ (P(i, t) \,\&\, P(j, t) \,\&\, i \neq j) \vee \mathcal{D}_G \sim\!\exists i, j, t\ (P(i, t) \,\&\, P(j, t) \,\&\, i \neq j).$$

Note that although we have been concerned with propositional epistemic logic, we shall here use first-order notation for ease of exposition. The example can be cast in propositional logic as well, for example, by using standard propositionalization techniques, see e.g. [16, pages 274–275]. To avoid having to use a large number of propositional symbols, we shall use a notation that looks like first-order notation, but we shall assume that everything is encoded in propositional logic. We can thus suppose that our knowledge base KB consists of sentences such as the following: $x : P(i,t) \supset \mathcal{K}_i P(i,t)$ for all agents i and all dates t. In propositional logic, this would look something like:

$$Birthday\_Agent1\_January01 \supset \mathcal{K}_1(Birthday\_Agent1\_January01) \text{ etc.}$$

KB now encodes the assumptions that each agent, here 1 and 2, knows one's own birthday.

Instead of putting the assumptions in KB to the left-hand side of the sequent arrow as in the previous example, we shall here employ the method of converting axioms to rules as in [17,6]. Thus, each axiom in KB will be replaced by a corresponding rule of the form:

$$\dfrac{x : \mathcal{K}_i P(i,t), x : P(i,t), \Gamma \Rightarrow \Delta}{x : P(i,t), \Gamma \Rightarrow \Delta}\;{\scriptstyle KB_{i,t}}$$

Observe that the addition of a rule of this form maintains all the structural properties of the system: Admissibility of contraction is guaranteed by the repetition of the principal formula $x : P(i,t)$ in the premiss. Further, there is no interference with the process of cut elimination because the principal formula $x : P(i,t)$ is atomic. Also invertibility of all the rules is for the same reason unaffected by the addition.

To make the example manageable, we shall prove only the part saying that if two agents have the same birthday, it will be distributed knowledge that they have the same birthday. (Proving the other part saying that if their birthdays differ, they will know that they differ, would require stating in addition that each agent can have only one birthday.) By making a further simplification and treating t as a constant here, we can express the claim as follows:

$$P(1,t) \;\&\; P(2,t) \supset \mathcal{D}_{\{1,2\}}(P(1,t) \;\&\; P(2,t)).$$

Without this simplification we would have to use the more complete expression, that is, a conjunction of all implications of the above kind for all possible values of t.

We get root-first the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y : P(1,t), x : \mathcal{K}_1 P(1,t), xR_1y, xR_2y, x : P(1,t), x : P(2,t) \Rightarrow y : P(1,t)}\;{\scriptstyle Ax}}{x : \mathcal{K}_1 P(1,t), xR_1y, xR_2y, x : P(1,t), x : P(2,t) \Rightarrow y : P(1,t)}\;{\scriptstyle L\mathcal{K}_1}}{xR_1y, xR_2y, x : P(1,t), x : P(2,t) \Rightarrow y : P(1,t)}\;{\scriptstyle KB_1}}{xR_1y, xR_2y, x : P(1,t) \;\&\; P(2,t) \Rightarrow y : P(1,t)}\;{\scriptstyle L\&} \qquad \vdots\;\text{(similarly)}}{xR_1y, xR_2y, x : P(1,t) \;\&\; P(2,t) \Rightarrow y : P(1,t) \;\&\; P(2,t)}\;{\scriptstyle R\&}}{x : P(1,t) \;\&\; P(2,t) \Rightarrow x : \mathcal{D}_{\{1,2\}}(P(1,t) \;\&\; P(2,t))}\;{\scriptstyle R\mathcal{D}}}{\Rightarrow x : P(1,t) \;\&\; P(2,t) \supset \mathcal{D}_{\{1,2\}}(P(1,t) \;\&\; P(2,t))}\;{\scriptstyle R\supset}$$

Thus, it is proved that if two agents have the same birthday, this fact is distributed knowledge.

Consider next an example in which agents possess information that put together leads to a contradiction. Suppose that agent 1 knows that $A \supset B$ and

$B \supset C$. She then receives a message from agent 2 claiming that A and a message from agent 3 claiming that $\sim B \mathbin{\&} (A \vee C)$. Agent 1 stores the information so her knowledge base KB contains the following formulas: $x : \mathcal{K}_1(A \supset B)$, $x : \mathcal{K}_1(B \supset C)$, $x : \mathcal{K}_2 A, x : \mathcal{K}_3(\sim B \mathbin{\&} (A \vee C))$. To find out whether all incoming information can be safely believed, agent 1 should check that the claimed contents do not lead to a contradiction as happens here:

$$
\begin{prooftree}
\AxiomC{$y : A, \dots \Rightarrow y : \bot, y : A$}
\AxiomC{$y : B, \dots \Rightarrow y : \bot, y : B$}
\AxiomC{}
\RightLabel{$\scriptstyle L\bot$}
\UnaryInfC{$y : \bot, \dots \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle L\supset$}
\BinaryInfC{$y : B \supset \bot, y : A \vee C, y : B, y : A, xR_1y, xR_2y, xR_3y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle L\&$}
\UnaryInfC{$y : (B \supset \bot) \mathbin{\&} (A \vee C), y : B, y : A, xR_1y, xR_2y, xR_3y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle L\mathcal{K}_3$}
\UnaryInfC{$y : B, y : A, xR_1y, xR_2y, xR_3y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle L\supset$}
\BinaryInfC{$y : A \supset B, y : A, xR_1y, xR_2y, xR_3y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle L\mathcal{K}_1$}
\UnaryInfC{$y : A, xR_1y, xR_2y, xR_3y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle L\mathcal{K}_2$}
\UnaryInfC{$xR_1y, xR_2y, xR_3y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle R\mathcal{D}_{\{1,2,3\}}$}
\UnaryInfC{$KB \Rightarrow x : \mathcal{D}_{\{1,2,3\}}\bot$}
\end{prooftree}
$$

Since combining all the information leads to a contradiction, agent 1 must find a subset of agents such that contradiction cannot be inferred. In this particular case she may decide that either 2 or 3 is less reliable than the others, or she may even decide that her own previous beliefs should be revised in light of the new information provided by 2 and 3. Supposing that she decides to drop the information provided by agent 3, she should then check that contradiction cannot be derived from the combined knowledge of 1 and 2 as follows:

$$
\begin{prooftree}
\AxiomC{$y : A, \dots \Rightarrow y : \bot, y : A$}
\AxiomC{$y : B, \dots \Rightarrow y : \bot, y : B$}
\AxiomC{$y : C, y : B, y : A, xR_1y, xR_2y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle L\supset$}
\BinaryInfC{$y : B \supset C, y : B, y : A, xR_1y, xR_2y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle L\mathcal{K}_1$}
\UnaryInfC{$y : B, y : A, xR_1y, xR_2y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle L\supset$}
\BinaryInfC{$y : A \supset B, y : A, xR_1y, xR_2y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle L\mathcal{K}_1$}
\UnaryInfC{$y : A, xR_1y, xR_2y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle L\mathcal{K}_2$}
\UnaryInfC{$xR_1y, xR_2y, KB \Rightarrow y : \bot$}
\RightLabel{$\scriptstyle R\mathcal{D}_{\{1,2\}}$}
\UnaryInfC{$KB \Rightarrow x : \mathcal{D}_{\{1,2\}}\bot$}
\end{prooftree}
$$

The uppermost premiss on the right hand side is not derivable: It is possible to continue the derivation by re-applying the left knowledge rules applied to formulas in KB or using the reflexivity and symmetry (and later transitivity) rules for the accessibility relation. After that left knowledge rules can also be applied with the expression xR_1x (or xR_2x) to yield $x : A$, $x : B$ and $x : C$ on the left hand side. Eventually only duplicates of existing formulas will be produced and the search can be terminated.

Agent 1 can now conclude that it is safe to reason about distributed knowledge among herself and agent 2 (because not everything can be inferred). Then she can find out, for instance, that together they can conclude that C holds:

$$
\begin{prooftree}
\AxiomC{$y : A, \dots \Rightarrow y : C, y : A$}
\AxiomC{$y : B, \dots \Rightarrow y : C, y : B$}
\AxiomC{$y : C, y : B, y : A, xR_1y, xR_2y, KB \Rightarrow y : C$}
\RightLabel{$\scriptstyle L\supset$}
\BinaryInfC{$y : B \supset C, y : B, y : A, xR_1y, xR_2y, KB \Rightarrow y : C$}
\RightLabel{$\scriptstyle L\mathcal{K}_1$}
\UnaryInfC{$y : B, y : A, xR_1y, xR_2y, KB \Rightarrow y : C$}
\RightLabel{$\scriptstyle L\supset$}
\BinaryInfC{$y : A \supset B, y : A, xR_1y, xR_2y, KB \Rightarrow y : C$}
\RightLabel{$\scriptstyle L\mathcal{K}_1$}
\UnaryInfC{$y : A, xR_1y, xR_2y, KB \Rightarrow y : C$}
\RightLabel{$\scriptstyle L\mathcal{K}_2$}
\UnaryInfC{$xR_1y, xR_2y, KB \Rightarrow y : C$}
\RightLabel{$\scriptstyle R\mathcal{D}_{\{1,2\}}$}
\UnaryInfC{$KB \Rightarrow x : \mathcal{D}_{\{1,2\}}C$}
\end{prooftree}
$$

This is actually the same derivation as the previous one just with $\bot$ replaced by C, but now all the premisses are derivable.

Instead of having decided to trust agent 2, agent 1 could have decided that agent 3 is more reliable. Then she would have had to check that $\mathcal{D}_{\{1,3\}}\bot$ cannot be derived and then to use the distributed knowledge between 1 and 3 as the basis of her reasoning. In general, reasoning and decision-making of an agent a can be based on the distributed knowledge $\mathcal{D}_{T_a}$ where $T_a \subseteq G$ is the set of agents currently held reliable by agent a. The choice of which agents to trust can later be retracted: If it turns out that some of the agents provide information that is clearly false, these agents can then be dropped from the subset T_a. The main point is that storing the source of information as well as the information content in the databases combined with the use of proof methods for reasoning about distributed knowledge provides a flexible way to deal with possibly erroneous multi-source information in a controlled fashion.

Note, however, that reasoning about distributed knowledge should not be seen as an alternative to existing information merging methods but rather as a tool for recognizing inconsistencies and making inferences from combined knowledge bases. This is because distributed knowledge is defined as whatever follows from the totality of what a collection of agents know. Thus, the approach does not directly support taking just one part of an agent's knowledge and rejecting another part that causes contradictions, as is often done in belief base merging. Reasoning about distributed knowledge requires either including everything an agent knows or excluding the agent altogether. Certainly, the methods can be modified by using a more fine-grained conception of agency: Instead of labelling everything agent a has claimed under $\mathcal{K}_a$ we can use, for instance, occasion-based labels or topic-based labels, as in $\mathcal{K}_{a\_on\_Thursday}$ or $\mathcal{K}_{Politician\_about\_taxes\_before\_elections}$. Then only certain parts of an agent's total knowledge can be taken into consideration.

Similarly, in the present system it is not possible to infer from contradictory reports that their disjunction must hold, contrary to e.g. [8]. If one witness claims that a car seen was black and another says it was red, it is often inferred that it must have been either black or red, but not white, for instance. This inference is not directly supported in our approach but must be implemented as a meta-level principle: If it is distributed knowledge within one consistent subset of agents that the car is black and within another that the car is red, we may want to conclude that it is either black or red. In a similar fashion meta-level principles are required for implementing other conflict-resolution methods like accepting a view supported by a majority of agents.

Another application area is cooperative problem solving, where it is assumed that all information is correct and the agents work together to solve theoretical or practical reasoning problems. Then distributed knowledge can be used to identify the collection of agents needed to provide a solution to a problem. Suppose, for instance, that the agents are asked to find out whether B holds. Suppose that we have in our use the following sentences obtained from agents 1 and 2:

$\mathcal{K}_1(A \supset\subset B)$, $\mathcal{K}_2(\mathcal{K}_3 A \vee \mathcal{K}_3 \sim A)$. We are interested in the truth of B, and the first agent knows that another proposition, A, is equivalent to B, and the second agent knows that a third agent knows whether this proposition A holds. Now agents 1,2 and 3, distributively know whether B holds but, in fact, after getting from agent 2 information concerning agent 3's knowledge, agent 2 is not needed anymore, because it is actually distributed knowledge between 1 and 3 alone whether B is the case, as can be seen from the derivation below:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{y:A,\ldots\Rightarrow\ldots,y:A \qquad y:B,\ldots\Rightarrow\ldots,y:B}{y:A,y:A\supset B,y:B\supset A,xR_1y,xR_3y,x:\mathcal{K}_3A,\ldots\Rightarrow x:\mathcal{D}_{\{1,3\}}\sim B,y:B}{\scriptstyle L\supset}}{y:A\supset B,y:B\supset A,xR_1y,xR_3y,x:\mathcal{K}_3A,\ldots\Rightarrow x:\mathcal{D}_{\{1,3\}}\sim B,y:B}{\scriptstyle L\mathcal{K}_3}}{y:A\supset\subset B,xR_1y,xR_3y,x:\mathcal{K}_3A,\ldots\Rightarrow x:\mathcal{D}_{\{1,3\}}\sim B,y:B}{\scriptstyle L\&}}{xR_1y,xR_3y,x:\mathcal{K}_3A,xR_2x,x:\mathcal{K}_1(A\supset\subset B),\ldots\Rightarrow x:\mathcal{D}_{\{1,3\}}\sim B,y:B}{\scriptstyle L\mathcal{K}_1}}{x:\mathcal{K}_3A,xR_2x,x:\mathcal{K}_1(A\supset\subset B),x:\mathcal{K}_2(\mathcal{K}_3A\vee\mathcal{K}_3\sim A)\Rightarrow x:\mathcal{D}_{\{1,3\}}B,x:\mathcal{D}_{\{1,3\}}\sim B}{\scriptstyle R\mathcal{D}_{\{1,3\}}}}{x:\mathcal{K}_3A,xR_2x,x:\mathcal{K}_1(A\supset\subset B),x:\mathcal{K}_2(\mathcal{K}_3A\vee\mathcal{K}_3\sim A)\Rightarrow x:\mathcal{D}_{\{1,3\}}B\vee\mathcal{D}_{\{1,3\}}\sim B}{\scriptstyle R\vee} \qquad \vdots}{x:\mathcal{K}_3A\vee\mathcal{K}_3\sim A,xR_2x,x:\mathcal{K}_1(A\supset\subset B),x:\mathcal{K}_2(\mathcal{K}_3A\vee\mathcal{K}_3\sim A)\Rightarrow x:\mathcal{D}_{\{1,3\}}B\vee\mathcal{D}_{\{1,3\}}\sim B}{\scriptstyle L\vee}}{xR_2x,x:\mathcal{K}_1(A\supset\subset B),x:\mathcal{K}_2(\mathcal{K}_3A\vee\mathcal{K}_3\sim A)\Rightarrow x:\mathcal{D}_{\{1,3\}}B\vee\mathcal{D}_{\{1,3\}}\sim B}{\scriptstyle L\mathcal{K}_2}}{x:\mathcal{K}_1(A\supset\subset B),x:\mathcal{K}_2(\mathcal{K}_3A\vee\mathcal{K}_3\sim A)\Rightarrow x:\mathcal{D}_{\{1,3\}}B\vee\mathcal{D}_{\{1,3\}}\sim B}{\scriptstyle Ref_2}$$

The branch marked with dots derives the sequent

$$x:\mathcal{K}_3\sim A, xR_2x, x:\mathcal{K}_1(A\supset\subset B), x:\mathcal{K}_2(\mathcal{K}_3A\vee\mathcal{K}_3\sim A)\Rightarrow x:\mathcal{D}_{\{1,3\}}B\vee\mathcal{D}_{\{1,3\}}\sim B$$

It is slightly more complicated because of the negation but is derivable as well. In this example, information provided by a collection of agents was used to find out another collection capable of providing an answer to the original query. The agents in the first group do not know the answer (it is not the case that $\mathcal{D}_{\{1,2\}}B \vee \mathcal{D}_{\{1,2\}} \sim B$), but they know who knows the answer (it is the case that $\mathcal{D}_{\{1,2\}}(\mathcal{D}_{\{1,3\}}B \vee \mathcal{D}_{\{1,3\}} \sim B)$). Agent 2's knowledge was crucial for finding out the set of agents needed to solve the original problem, but the actual query can now be given as a task for agents 1 and 3 to co-operatively solve. Thus, distributed knowledge gives a way of reasoning about informants and their knowledge without requiring that the reasoning agent possesses the actual knowledge. It may be enough if the agent can find out who has the knowledge, as is often the case in real life situations.

5 Conclusions and Future Work

We have here presented a sequent calculus system for formal reasoning in multi-agent epistemic logic with operators for distributed knowledge. Our system enjoys the structural properties that support proof search that starts from the conclusion to be derived. Because the rules are invertible, there is no need for a backtracking mechanism, since if the conclusion is derivable also the premisses are guaranteed to be derivable. Admissibility of the contraction rules guarantees that rules that only produce duplications of the existing formulas need not be considered in the proof search. Finally, admissibility of cut is crucial for delimiting the space of the proof search, because it ensures that no arbitrary new formulas need to be constructed during the search.

Recent literature ([12], [13]) has emphasized the importance of certain properties for what should be appropriately called "distributed knowledge". Among such properties is the principle of full communication: Whenever a statement A is distributed knowledge within a group, it should be possible to derive A from what the agents individually know. As discussed in [12], the syntactic form of the principle roughly corresponds to invertibility of rule 3, which does not hold in general, as a simple Kripke countermodel shows. A characterization of the epistemic models that obey the principle is given in [13]. We can make the principle of full communication part of our system by incorporating the model-theoretic properties in the form of rules for the accessibility relation. First we observe that if the accessibility relations for the modalities of individual knowledge K_a are "canonical," i.e., satisfy

$$xR_a y \text{ iff for all } A(x \Vdash K_a A \text{ implies } y \Vdash A)$$

then the principle of full communication holds. Imposing canonicity amounts to the addition of certain rules such as

$$\frac{x : K_a A, \Gamma \Rightarrow \Delta, y : A}{\Gamma \Rightarrow \Delta, xR_a y}\ {\scriptstyle RR_a}$$

where A is an arbitrary formula not in Γ, Δ, and

$$\frac{y : A, x : K_a A, xR_a y, \Gamma \Rightarrow \Delta}{x : K_a A, xR_a y, \Gamma \Rightarrow \Delta}\ {\scriptstyle LR_a}$$

It is not difficult, although not completely straightforward, to show that the structural properties of the system are maintained by the addition. For instance, for proving admissibility of contraction and cut, a measure of complexity has to be defined in such a way that the weight of relational atoms is greater than the weight of labelled formulas. This is unproblematic as it can be done consistently with the earlier requirements in the proofs of the structural properties of the basic epistemic system. The details will be left for subsequent work.

References

1. Hintikka, J.: Knowledge and Belief: An Introduction to the Logic of the Two Notions. Cornell University Press (1962)
2. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning About Knowledge. MIT Press, Cambridge (1995)
3. Meyer, J.J.C., van der Hoek, W.: Epistemic Logic for AI and Computer Science. Cambridge University Press, Cambridge (1995)
4. Ghidini, C., Serafini, L.: A context-based logic for distributed knowledge representation and reasoning. In: Bouquet, P., Serafini, L., Brézillon, P., Benercetti, M., Castellani, F. (eds.) CONTEXT 1999. LNCS (LNAI), vol. 1688, pp. 159–172. Springer, Heidelberg (1999)

5. Artemov, S.N.: Research problems, http://web.cs.gc.cuny.edu/~sartemov/research_problems.html
6. Negri, S., von Plato, J.: Structural Proof Theory. Cambridge University Press, Cambridge (2001)
7. Negri, S.: Proof analysis in modal logic. Journal of Philosophical Logic 34, 507–544 (2005)
8. Baral, C., Kraus, S., Minker, J.: Combining multiple knowledge bases. IEEE Transactions on Knowledge and Data Engineering 3, 208–220 (1991)
9. Cholvy, L., Garion, C.: Answering queries addressed to several databases according to a majority merging. Journal of Intelligent Information Systems 22, 175–201 (2004)
10. 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)
11. van der Hoek, W., Meyer, J.J.: A complete epistemic logic for multiple agents. In: Bacharach, M., Gérard-Varet, L.A., Mongin, P., Shin, H. (eds.) Epistemic Logic and the Theory of Games and Decisions, pp. 35–68. Kluwer Academic Publishers, Dordrecht (1997)
12. van der Hoek, W., van Linder, B., Meyer, J.J.: Group knowledge is not always distributed (neither is it always implicit). Mathematical Social Sciences 38, 215–240 (1999)
13. Roelofsen, F.: Distributed knowledge. Journal of Applied Non-Classical Logics 17(2), 255–273 (2007)
14. Halpern, J.Y., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artificial Intelligence 54, 319–379 (1992)
15. Fagin, R., Halpern, J.Y., Vardi, M.Y.: What can machines know? On the properties of knowledge in distributed systems. Journal of the ACM 39, 328–376 (1992)
16. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Pearson Education International, London (2003)
17. Negri, S., von Plato, J.: Cut elimination in the presence of axioms. The Bulletin of Symbolic Logic 4, 418–435 (1998)

EVOLP: Tranformation-Based Semantics*

Martin Slota[1,2] and João Leite[2]

[1] Katedra aplikovanej informatiky, Univerzita Komenského, Slovakia
[2] CENTRIA, Universidade Nova de Lisboa, Portugal

Abstract. Over the years, Logic Programming has proved to be a good and natural tool for expressing, querying and manipulating explicit knowledge in many areas of computer science. However, it is not so easy to use in dynamic environments. Evolving Logic Programs (EVOLP) are an elegant and powerful extension of Logic Programming suitable for Multi-Agent Systems, planning and other uses where information tends to change dynamically. In this paper we characterize EVOLP by transforming it into an equivalent normal logic program over an extended language, that serves as a basis of an existing implementation. Then we prove that the proposed transformation is sound and complete and examine its computational complexity.

1 Introduction

Construction of intelligent agents is one of the main matters of artificial intelligence. Computational Logic has shown to be a good tool for both symbolic knowledge representation and reasoning, with fruitful application in Multi-Agent Systems.

Examples of the success of Computational Logic in Multi-Agent Systems include IMPACT [1,2], 3APL [3,4], Jason [5], DALI [6], ProSOCS [7], FLUX [8] and ConGolog [9], to name a few. For a survey on some of these systems, as well as others, see [10,11,12].

Computational Logic, and Logic Programming in particular, can be seen as a good representation language for static knowledge. However, agents must be capable of operating independently in a partially observable environment that may change unexpectedly. Therefore, they need to be able to evolve, both due to self-updates and updates from the environment, and change their model of the world accordingly. If we are to move to such more open and dynamic environments, we must consider ways and means of representing and integrating knowledge updates from external as well as internal sources.

In fact, an agent should not only comprise knowledge about each state, but also knowledge about the transitions between states. The latter may represent the agent's knowledge about the environment's evolution, coupled to its own

* This research has been funded by the European Commission within the 6th Framework Programme project REWERSE number 506779 (cf. http://rewerse.net). It was also supported by the Slovak Agency for Promotion Research and Development under the contract No. APVV-20-P04805.

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 117–136, 2008.

behaviour and evolution. The lack of rich mechanisms to represent and reason about dynamic knowledge and agents i.e. represent and reason about environments where not only some facts about it change, but also the rules that govern it, and where the behaviours of agents also change, is common to the above mentioned systems.

Much research in the last decade has been devoted to finding a good way of updating knowledge bases represented by logic programs [13,14,15,16,17,18,19]. A sequence of logic programs where each program represents a supervenient state of the world was called a Dynamic Logic Program (DLP). Finding a suitable semantics for DLPs became the first step on one of the paths to using Logic Programming in Multi-Agent Systems. Quite a number of semantics with different properties were introduced [16,17,18,19]. We will only mention the Dynamic Stable Model semantics [17] that was later improved and called Refined Dynamic Stable Models [19]. This is also the semantics used throughout this work. For a more comprehensive overview of semantics for DLPs see [18,20,21].

Although Dynamic Logic Programming provides a semantics for a sequence of states of the world expressed as logic programs, it doesn't offer a mechanism for constructing these programs. Update languages like LUPS [22], EPI [23], KUL and KABUL [18] were developed for the purpose of specifying transitions between the states of the world. Each of them defines special types of rules for adding and deleting rules from programs in the sequence. Evolving Logic Programs (EVOLP) [24] also comes from this line of work, but while its predecessors were becoming more and more complicated as more constructs were being added, EVOLP is a simple, yet very powerful extension of traditional logic programming.

EVOLP generalizes Answer-set Programming [25] to allow for the specification of a program's own evolution, in a single unified way. Furthermore, EVOLP also permits, besides internal or self updates, for updates arising from the environment. The resulting language provides a simpler, and more general, formulation of logic program updating, running close to traditional LP doctrine, setting itself on a firm formal basis in which to express, implement, and reason about dynamic knowledge bases, opening up several interesting research topics.

Indeed, EVOLP can adequately express the semantics resulting from successive updates to logic programs, considered as incremental specifications of agents, and whose effect can be contextual. Syntactically, evolving logic programs are just generalized logic programs[1]. But semantically, they permit to reason about updates of the program itself. The language of EVOLP contains a special predicate assert/1 whose sole argument is a full-blown rule. Whenever an assertion assert(r) is true in a model, the program is updated with rule r. The process is then further iterated with the new program.

Whenever the program semantics allows for several possible program models, evolution branching occurs, and several evolution sequences are made possible. This branching can be used to specify the evolution of a situation in the presence of incomplete information. Moreover, the ability of EVOLP to nest rule

[1] Logic programs that allow for rules with default negated literals in their heads.

assertions within assertions allows rule updates to be themselves updated down the line. Furthermore, the EVOLP language can express self-modifications triggered by the evolution context itself, present or future – assert literals included in rule bodies allow for looking ahead on some program changes and acting on that knowledge before the changes occur. In contradistinction to other approaches, EVOLP also automatically and appropriately deals with the possible contradictions arising from successive specification changes and refinements (via Dynamic Logic Programming).

The aim of this work is to provide the basis for an operational semantics for EVOLP, based on a sound and complete transformational semantics for EVOLP, i.e. define a transformation that, given an evolving logic program and a sequence of events, produces an equivalent normal logic program over an extended language. Such a transformation, together with an ASP solver, is the basis of our implementation of EVOLP under the evolution stable model semantics[2]. Currently, the only somehow similar implementation appears in [26] and only for a limited constructive view of EVOLP. More information about the implementation can be found in [27].

We also examine the complexity of the defined transformation. This is performed by inferring both a lower and an upper bound for the size of the transformed program.

The remainder of this work is structured as follows: in Sect. 2 we introduce the syntax and semantics of EVOLP; in Sect. 3 we define the transformation; in Sect. 4 we show that the proposed transformation is sound and complete; in Sect. 5 we examine the complexity of the transformation; in Sect. 6 we conclude and sketch some possible directions of future work.

2 Background: Concepts and Notation

We start with the usual preliminaries: Let $\mathcal{L}$ be a set of propositional atoms. A *default literal* is an atom preceded by **not**. A *literal* is either an atom or a default literal. A *rule* r is an ordered pair $(H(r), B(r))$ where $H(r)$ (dubbed the *head of the rule*) is a literal and $B(r)$ (dubbed the *body of the rule*) is a finite set of literals. A rule with $H(r) = L_0$ and $B(r) = \{ L_1, L_2, \ldots, L_n \}$ will simply be written as

$$L_0 \leftarrow L_1, L_2, \ldots, L_n. \tag{1}$$

If $H(r) = A$ (resp. $H(r) = \mathbf{not}\, A$) then $\mathbf{not}\, H(r) = \mathbf{not}\, A$ (resp. $\mathbf{not}\, H(r) = A$). Two rules r, r' are *conflicting*, denoted by $r \bowtie r'$, iff $H(r) = \mathbf{not}\, H(r')$. We will say a literal L appears in a rule (1) iff the set $\{ L, \mathbf{not}\, L \} \cap \{ L_0, L_1, L_2, \ldots, L_n \}$ is non-empty.

A *generalized logic program* (GLP) over $\mathcal{L}$ is a set of rules. A literal appears in a GLP iff it appears in at least one of its rules.

An *interpretation* of $\mathcal{L}$ is any set of atoms $I \subseteq \mathcal{L}$. An atom A is true in I, denoted by $I \models A$, iff $A \in I$, and false otherwise. A default literal $\mathbf{not}\, A$ is true

[2] A web based demo is available at `http://centria.di.fct.unl.pt/evolp/`

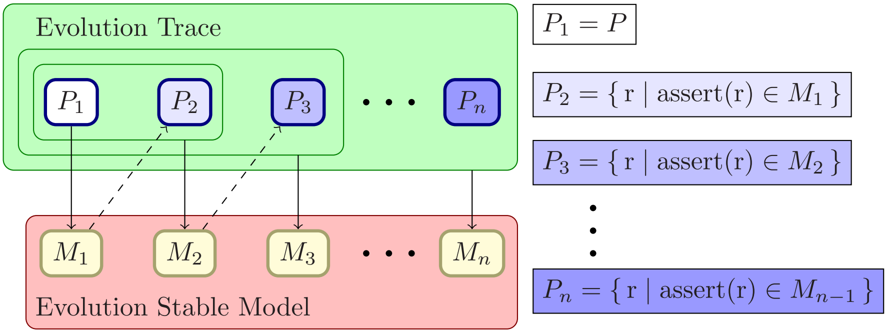

Evolution Trace
P_1
P_2
P_3
$\cdots$
P_n
M_1
M_2
M_3
$\cdots$
M_n
Evolution Stable Model
$P_1 = P$
$P_2 = \{\, \mathrm{r} \mid \mathrm{assert}(\mathrm{r}) \in M_1 \,\}$
$P_3 = \{\, \mathrm{r} \mid \mathrm{assert}(\mathrm{r}) \in M_2 \,\}$
$\vdots$
$P_n = \{\, \mathrm{r} \mid \mathrm{assert}(\mathrm{r}) \in M_{n-1} \,\}$

Fig. 1. Semantics of EVOLP (without events)

in I, denoted by $I \models \textbf{not}\, A$, iff $A \notin I$, and false otherwise. A set of literals B is true in I iff each literal in B is true in I. Given an interpretation I we also define $I^- \overset{\text{def}}{=} \{\, \textbf{not}\, A \mid A \in \mathcal{L} \setminus I \,\}$ and $I^* \overset{\text{def}}{=} I \cup I^-$. An interpretation M is a *stable model* of a GLP P iff $M^* = \mathrm{least}(P \cup M^-)$ where $\mathrm{least}(\cdot)$ denotes the least model of the definite program obtained from the argument program by treating all default literals as new atoms.

Definition 1. *A* dynamic logic program *(DLP) is a sequence of GLPs. Let $\mathcal{P} = (P_1, P_2, \ldots, P_n)$ be a DLP. We use $\rho(\mathcal{P})$ to denote the multiset of all rules appearing in the programs $P_1, P_2, \ldots, P_n$ and $\mathcal{P}^i$ $(1 \leq i \leq n)$ to denote the i-th component of $\mathcal{P}$, i.e. P_i. Given a DLP $\mathcal{P}$ and an interpretation I we define*

$$\mathrm{Def}(\mathcal{P}, I) \overset{\text{def}}{=} \{\, \textbf{not}\, A \mid (\nexists r \in \rho(\mathcal{P}))(H(r) = A \wedge I \models B(r)) \,\} \ , \tag{2}$$

$$\mathrm{Rej}^j(\mathcal{P}, I) \overset{\text{def}}{=} \left\{\, r \in \mathcal{P}^j \;\middle|\; (\exists k, r')\left(k \geq j \wedge r' \in \mathcal{P}^k \wedge r \bowtie r' \wedge I \models B(r')\right) \,\right\} \ , \tag{3}$$

$$\mathrm{Rej}(\mathcal{P}, I) \overset{\text{def}}{=} \bigcup_{i=1}^{n} \mathrm{Rej}^i(\mathcal{P}, I) \ . \tag{4}$$

An interpretation M is a (refined) dynamic stable model *of a DLP $\mathcal{P}$ iff $M^* = \mathrm{least}([\rho(\mathcal{P}) \setminus \mathrm{Rej}(\mathcal{P}, M)] \cup \mathrm{Def}(\mathcal{P}, M))$.*

Definition 2. *Let $\mathcal{L}$ be a set of propositional atoms (not containing the predicate* assert/1*). The extended language $\mathcal{L}_{\mathrm{assert}}$ is defined inductively as follows:*

1. *All propositional atoms in $\mathcal{L}$ are propositional atoms in $\mathcal{L}_{\mathrm{assert}}$.*
2. *If r is a rule over $\mathcal{L}_{\mathrm{assert}}$ then $\mathrm{assert}(r)$ is a propositional atom in $\mathcal{L}_{\mathrm{assert}}$.*
3. *Nothing else is a propositional atom in $\mathcal{L}_{\mathrm{assert}}$.*

An evolving logic program *over a language $\mathcal{L}$ is a GLP over $\mathcal{L}_{\mathrm{assert}}$. An* event sequence *over $\mathcal{L}$ is a sequence of evolving logic programs over $\mathcal{L}$.*

Table 1. Evolution of the program in Example 1 ("assert" is shortened to "ass")

Time	Program	Event	Model
1	P	E_1	$\{\text{no\_coffee}, \text{write\_thesis},$ $\text{ass}(\text{tired} \leftarrow)\}$
2	$\{\,\text{tired} \leftarrow .\,\}$	E_2	$\{\text{tired}, \text{no\_coffee}, \text{make\_coffee}\}$
3	$\emptyset$	E_3	$\{\text{tired}, \text{drink\_coffee}, \text{ass}(\mathbf{not}\,\text{tired} \leftarrow)\}$
4	$\{\,\mathbf{not}\,\text{tired} \leftarrow .\,\}$	E_4	$\{\text{write\_thesis}, \text{ass}(\text{tired} \leftarrow),$ $\text{ass}(\mathbf{not}\,\text{drink\_coffee} \leftarrow),$ $\text{ass}(\text{sleep} \leftarrow \text{tired}),$ $\text{ass}(\text{ass}(\mathbf{not}\,\text{tired} \leftarrow) \leftarrow \text{sleep})\}$
5	$\{\text{tired} \leftarrow .,$ $\mathbf{not}\,\text{drink\_coffee} \leftarrow .,$ $\text{sleep} \leftarrow \text{tired}.,$ $\text{ass}(\mathbf{not}\,\text{tired} \leftarrow) \leftarrow \text{sleep}.\}$	E_5	$\{\text{tired}, \text{sleep}, \text{ass}(\mathbf{not}\,\text{tired} \leftarrow)\}$

Definition 3. *An* evolution interpretation of length n *of an evolving program* P *over* $\mathcal{L}$ *is a finite sequence* $\mathcal{I} = (I_1, I_2, \ldots, I_n)$ *of interpretations of* $\mathcal{L}_{\text{assert}}$. *The* evolution trace *associated with an evolution interpretation* $\mathcal{I}$ *of* P *is the sequence of programs* $(P_1, P_2, \ldots, P_n)$ *where* $P_1 = P$ *and* $P_{i+1} = \{\, r \mid \text{assert}(r) \in I_i \,\}$ *for all* $i \in \{\, 1, 2, \ldots, n-1 \,\}$.

Definition 4. *An evolution interpretation* $\mathcal{M} = (M_1, M_2, \ldots, M_n)$ *of an evolving logic program* P *with evolution trace* $(P_1, P_2, \ldots, P_n)$ *is an* evolution stable model *of* P *given an event sequence* $(E_1, E_2, \ldots, E_n)$ *iff for every* $i \in \{\, 1, 2, \ldots, n \,\}$ M_i *is a dynamic stable model of* $(P_1, P_2, \ldots, P_{i-1}, P_i \cup E_i)$.

Example 1. Consider the following evolving logic program:

$$P: \qquad \text{write\_thesis} \leftarrow \mathbf{not}\,\text{tired}. \tag{5}$$

$$\text{drink\_coffee} \leftarrow \text{tired}, \mathbf{not}\,\text{no\_coffee}. \tag{6}$$

$$\text{make\_coffee} \leftarrow \text{tired}, \text{no\_coffee}. \tag{7}$$

$$\text{assert}(\text{tired} \leftarrow) \leftarrow \text{write\_thesis}. \tag{8}$$

$$\text{assert}(\mathbf{not}\,\text{tired} \leftarrow) \leftarrow \text{drink\_coffee}. \tag{9}$$

P could be an initial program of a simple agent (e.g. Mary) who is trying to write a thesis. Mary can do 3 things: write the thesis, drink coffee or make coffee. She also relies on a sensor that sends her the fact (no_coffee ← .) as an event in case no coffee is available. The meaning of the rules is as follows: Rule (5) says Mary's writing the thesis as long as she's not tired. Rules (6) and (7) tell her what to do when she's tired. Rules (8) and (9) specify whether she will be tired in the next evolution step. If she's writing the thesis, she will get tired. Drinking coffee has an opposite effect. If she's making coffee, no change will take place. Table 1 shows the evolution of P given the sequence of events $\mathcal{E} = (E_1, E_2, E_3, E_4, E_5)$ where $E_1 = E_2 = \{\, \text{no\_coffee} \leftarrow .\, \}$, $E_3 = E_5 = \emptyset$ and

$$\begin{aligned} E_4: \quad & \text{assert}(\textbf{not}\,\text{drink\_coffee} \leftarrow) \leftarrow . \\ & \text{assert}(\text{sleep} \leftarrow \text{tired}) \leftarrow . \\ & \text{assert}(\text{assert}(\textbf{not}\,\text{tired} \leftarrow) \leftarrow \text{sleep}) \leftarrow . \end{aligned}$$

We start off with P and E_1 and compute the first model. It says there is no coffee, Mary is writing her thesis and in the next step she will get tired. We infer the second program from the model, add the second event and compute the second model. Now Mary is tired and makes coffee. This makes the sensor stop complaining in the third step (i.e. $E_3 = \emptyset$) and Mary, still tired, drinks coffee. In the fourth step Mary is writing her thesis again and she is reprogrammed – when she gets tired she will take a nap instead of drinking coffee. In the fifth step the new rules are used – Mary is tired and sleeping.

The previous example is very simple and its main purpose is to demonstrate how the definitions of semantics of EVOLP work. It is by no means exhaustive and doesn't demonstrate the full power of the language. In the next example we will briefly show more complex rules that are a part of a more complex example where EVOLP is used as to implement a fairly sophisticated email agent. For the full version of the example the reader is referred to [28]. Another different example that makes use of an agent architecture based on EVOLP can be found in [26].

Example 2. The email agent example is composed of an evolving logic program P and a sequence of events $(E_1, E_2, \dots, E_16)$. We will only pick some specific rules from the example to demonstrate the constructs that can be specified using EVOLP.

First let's consider the following two rules:

$$\begin{aligned} \text{assert}(\text{in}(M, F_{to}) \leftarrow) &\leftarrow \text{move}(M, F_{from}, F_{to}), \text{in}(M, F_{from}). \\ \text{assert}(\textbf{not}\,\text{in}(M, F_{from}) \leftarrow) &\leftarrow \text{move}(M, F_{from}, F_{to}), \textbf{not}\,\text{in}(M, F_{to}). \end{aligned}$$

They are used in [28] as a part of the base program of an email agent and encode a message moving mechanism. The first rule specifies that if a command received to move a message M from folder F_{from} to folder F_{to} and it is currently in folder F_{from} (i.e. the command is a valid one), then in the next evolution step the message will be in folder F_{to}. The second rule deletes M from folder F_{from} in case it is different from the destination folder F_{to}. Similar rules are used to remember new messages and delete them from folders and remember sent messages.

Another three rules encode an evolving predicate that decides whether a message is spam or not:

$$\begin{aligned} r_1: \quad & \text{spam}(F, S, B) \leftarrow \text{contains}(S, \text{“credit”}). \\ r_2: \quad \textbf{not}\, & \text{spam}(F, S, B) \leftarrow \text{contains}(F, \text{“accountant”}). \\ r_3: \quad & \text{spam}(F, S, B) \leftarrow \text{contains}(S, \text{“credit”}), \text{contains}(S, \text{“Fwd”}). \end{aligned}$$

They are asserted one by one in the events, i.e. $\text{assert}(r_1) \in E_i$, $\text{assert}(r_2) \in E_j$ and $\text{assert}(r_3) \in E_k$ for some $i < j < k$. The first rule defines a spam message as any message having "credit" in its subject. This is further updated by the second rule – messages whose sender contains the word "accountant" are not considered as spam (even if they contain "credit" in their body). The third rule further updates the way messages are classified – messages containing both "credit" and "Fwd" in their subject are considered spam (even if they come from the accountant).

The last rule we are going to mention encodes a more complex behaviour:

$$\begin{aligned}\text{assert}(\text{send}(R, S, B) \leftarrow \text{newmsg}(M, F, S, B), \text{contains}(S, ID), \text{assign}(ID, R)) \leftarrow \\ \text{newmsg}(M, R, ID, B), \text{contains}(B, \text{“accept”}).\end{aligned}$$

The meaning of the rule is as follows: If a message is received from a reviewer R that contains a paper identification ID in the subject and the word "accept" in its body, then all future messages regarding this paper will be forwarded to the reviewer R in case he has been assigned the paper ID. Multiple rules of this kind can be used to configure a simple paper submission system that keeps track of papers, deadlines, authors and reviewers and manages the communication between them.

3 Transformation into a Normal Logic Program

Now we will define a transformation which turns an evolving logic program P together with an event sequence $\mathcal{E}$ of length n into a normal logic program $P_{\mathcal{E}}$ over an extended language. We will prove later that the stable models of $P_{\mathcal{E}}$ are in one-to-one correspondence with the evolution stable models of P given $\mathcal{E}$.

The transformation is essentially a multiple parallel usage of a similar transformation for DLPs introduced in [29]. First we need to define the extended language over which we will construct the resulting program:

$$\begin{aligned}\mathcal{L}_{\text{trans}} \overset{\text{def}}{=} & \left\{\, A^j, A^j_{\text{neg}} \;\middle|\; A \in \mathcal{L}_{\text{assert}} \wedge 1 \le j \le n \,\right\} \\ & \cup \left\{\, \text{rej}(A^j, i), \text{rej}(A^j_{\text{neg}}, i) \;\middle|\; A \in \mathcal{L}_{\text{assert}} \wedge 1 \le j \le n \wedge 0 \le i \le j \,\right\} \\ & \cup \{\, u \,\} \quad .\end{aligned}$$

Atoms of the form A^j and A^j_{neg} in the extended language allow us to compress the whole evolution interpretation (consisting of n interpretations of $\mathcal{L}_{\text{assert}}$, see Def. 3) into just one interpretation of $\mathcal{L}_{\text{trans}}$. Atoms of the form $\text{rej}(A^j, i)$ and $\text{rej}(A^j_{\text{neg}}, i)$ are needed for rule rejection simulation. The atom u will serve to formulate constraints needed to eliminate some unwanted models of $P_{\mathcal{E}}$.

To simplify the notation in the transformation's definition, we'll use the following conventions: Let L be a literal over $\mathcal{L}_{\text{assert}}$, *Body* a set of literals over $\mathcal{L}_{\text{assert}}$ and j a natural number. Then:

- If L is an atom A, then L^j is A^j and L^j_{neg} is A^j_{neg}.
- If L is a default literal **not** A, then L^j is A^j_{neg} and L^j_{neg} is A^j.
- $\mathit{Body}^j = \{\, L^j \mid L \in \mathit{Body} \,\}$.

Definition 5. *Let P be an evolving logic program and $\mathcal{E} = (E_1, E_2, \ldots, E_n)$ an event sequence. By a* transformational equivalent *of P given $\mathcal{E}$ we mean the normal logic program $P_{\mathcal{E}} = P^1_{\mathcal{E}} \cup P^2_{\mathcal{E}} \cup \ldots \cup P^n_{\mathcal{E}}$ over $\mathcal{L}_{\text{trans}}$, where each $P^j_{\mathcal{E}}$ consists of these six groups of rules:*

1. ***Rewritten program rules.*** *For every rule $(L \leftarrow \mathit{Body}.) \in P$, $P^j_{\mathcal{E}}$ contains the rule*
$$L^j \leftarrow \mathit{Body}^j, \mathbf{not}\, \mathrm{rej}(L^j, 1).$$
2. ***Rewritten event rules.*** *For every rule $(L \leftarrow \mathit{Body}.) \in E_j$, $P^j_{\mathcal{E}}$ contains the rule*
$$L^j \leftarrow \mathit{Body}^j, \mathbf{not}\, \mathrm{rej}(L^j, j).$$
3. ***Assertable rules.*** *For every rule $r = (L \leftarrow \mathit{Body}.)$ over $\mathcal{L}_{\text{assert}}$ and all i, $1 < i \leq j$, such that $(\mathrm{assert}(r))^{i-1}$ is in the head of some rule of $P^{i-1}_{\mathcal{E}}$, $P^j_{\mathcal{E}}$ contains the rule*
$$L^j \leftarrow \mathit{Body}^j, (\mathrm{assert}(r))^{i-1}, \mathbf{not}\, \mathrm{rej}(L^j, i).$$
4. ***Default assumptions.*** *For every atom $A \in \mathcal{L}_{\text{assert}}$ such that A^j or A^j_{neg} appears in some rule of $P^j_{\mathcal{E}}$ (from the previous groups of rules), $P^j_{\mathcal{E}}$ also contains the rule*
$$A^j_{\text{neg}} \leftarrow \mathbf{not}\, \mathrm{rej}(A^j_{\text{neg}}, 0).$$
5. ***Rejection rules.*** *For every rule of $P^j_{\mathcal{E}}$ of the form*
$$L^j \leftarrow \mathit{Body}, \mathbf{not}\, \mathrm{rej}(L^j, i).[3]$$
$P^j_{\mathcal{E}}$ also contains the rules
$$\mathrm{rej}(L^j_{\text{neg}}, p) \leftarrow \mathit{Body}. \tag{10}$$
$$\mathrm{rej}(L^j, q) \leftarrow \mathrm{rej}(L^j, i). \tag{11}$$
where:
   (a) $p \leq i$ is the largest index such that $P^j_{\mathcal{E}}$ contains a rule with the literal **not** $\mathrm{rej}(L^j_{\text{neg}}, p)$ *in its body. If no such p exists, then (10) is not in $P^j_{\mathcal{E}}$.*
   (b) $q < i$ is the largest index such that $P^j_{\mathcal{E}}$ contains a rule with the literal **not** $\mathrm{rej}(L^j, q)$ *in its body. If no such q exists, then (11) is not in $P^j_{\mathcal{E}}$.*

[3] It can be a rewritten program rule, a rewritten event rule or an assertable rule (default assumptions never satisfy the further conditions). The set *Body* contains all literals from the rule's body except the **not** $\mathrm{rej}(L^j, i)$ literal.

6. ***Totality constraints.*** *For all $i \in \{1, 2, \ldots, j\}$ and every atom $A \in \mathcal{L}_{\text{assert}}$ such that $P_{\mathcal{E}}^{j}$ contains rules of the form*

$$A^j \leftarrow \mathit{Body}_p, \mathbf{not}\, \text{rej}(A^j, i).$$
$$A^j_{\text{neg}} \leftarrow \mathit{Body}_n, \mathbf{not}\, \text{rej}(A^j_{\text{neg}}, i).$$

$P_{\mathcal{E}}^{j}$ also contains the constraint

$$u \leftarrow \mathbf{not}\, u, \mathbf{not}\, A^j, \mathbf{not}\, A^j_{\text{neg}}.$$

Each $P_{\mathcal{E}}^{j}$ contains rules for simulating the DLP $(P, P_2, P_3, \ldots, P_{j-1}, P_j \cup E_j)$ from the definition of evolution stable model (Definition 4). For the simulation we use the transformational semantics from [29]. We also rewrite all atoms from the original rules as a new set of j-indexed atoms.

The first two groups of rules in $P_{\mathcal{E}}^{j}$ (rewritten program rules and rewritten event rules) contain the rewritten forms of rules from P and E_j. However, we don't know the exact contents of $P_2, P_3, \ldots, P_j$, so the group of assertable rules contains all rules that can possibly occur in those programs. Each of these rules also has an atom of the form $(\text{assert}(r))^{i-1}$ in its body. We will call it the *assertion guard* of the rule and it assures the rule is only used in case it was actually asserted. These atoms are also the only connection between the rules of $P_{\mathcal{E}}^{j}$ and the rules in $P_{\mathcal{E}}^{1} \cup P_{\mathcal{E}}^{2} \cup \ldots \cup P_{\mathcal{E}}^{j-1}$.

The default assumptions are defined similarly as in [29], and they have the same function – they simulate the set of defaults defined in Def. 1.

Rewritten program rules, rewritten event rules, assertable rules and default assumptions also contain a default literal of the form $\mathbf{not}\, \text{rej}(L^j, i)$ in their bodies. We will call this literal the *rejection guard* of the rule and the natural number i the *level* of the rule. Together with the rejection rules, the rejection guard provides a means of rejecting a rule by a higher level rule, similarly as in the set of rejected rules (4).

Rejection rules are responsible for inferring the correct $\text{rej}(L^j, i)$ atoms. The first kind of rules introduces the rejection of rules with a conflicting literal in their head and a level that is the maximum that is also less or equal to i. The second kind of rules takes care of propagating the rejection to rules with an even lower level.

Totality constraints are important in the case that rules of the same level reject each other and no rule of higher level resolves their conflict. An interpretation causing such a situation is not a refined dynamic stable model (more details can be found in [19]). Totality constraints are needed to eliminate the superfluous stable models of $P_{\mathcal{E}}$ originating from these situations.

The following example illustrates how the transformation works:

Example 3. Let's take the evolving logic program

$$P: \qquad \text{assert}(\text{a} \leftarrow) \leftarrow \mathbf{not}\, \text{a}.$$
$$\text{assert}(\mathbf{not}\, \text{a} \leftarrow) \leftarrow \text{a}.$$

and a sequence of two empty events $\mathcal{E}$. The defined transformation would produce the following transformed program:

$$\begin{aligned}
P_{\mathcal{E}} : \quad (\mathrm{assert}(\mathrm{a} \leftarrow))^1 &\leftarrow \mathrm{a}^1_{\mathrm{neg}}, \mathbf{not}\,\mathrm{rej}((\mathrm{assert}(\mathrm{a} \leftarrow))^1, 1). && (12)\\
(\mathrm{assert}(\mathbf{not}\,\mathrm{a} \leftarrow))^1 &\leftarrow \mathrm{a}^1, \mathbf{not}\,\mathrm{rej}((\mathrm{assert}(\mathbf{not}\,\mathrm{a} \leftarrow))^1, 1). && (13)\\
\mathrm{a}^1_{\mathrm{neg}} &\leftarrow \mathbf{not}\,\mathrm{rej}(\mathrm{a}^1_{\mathrm{neg}}, 0). && (14)\\
(\mathrm{assert}(\mathrm{a} \leftarrow))^2 &\leftarrow \mathrm{a}^2_{\mathrm{neg}}, \mathbf{not}\,\mathrm{rej}((\mathrm{assert}(\mathrm{a} \leftarrow))^2, 1). && (15)\\
(\mathrm{assert}(\mathbf{not}\,\mathrm{a} \leftarrow))^2 &\leftarrow \mathrm{a}^2, \mathbf{not}\,\mathrm{rej}((\mathrm{assert}(\mathbf{not}\,\mathrm{a} \leftarrow))^2, 1). && (16)\\
\mathrm{a}^2 &\leftarrow (\mathrm{assert}(\mathrm{a} \leftarrow))^1, \mathbf{not}\,\mathrm{rej}(\mathrm{a}^2, 2). && (17)\\
\mathrm{a}^2_{\mathrm{neg}} &\leftarrow (\mathrm{assert}(\mathbf{not}\,\mathrm{a} \leftarrow))^1, \mathbf{not}\,\mathrm{rej}(\mathrm{a}^2_{\mathrm{neg}}, 2). && (18)\\
\mathrm{a}^2_{\mathrm{neg}} &\leftarrow \mathbf{not}\,\mathrm{rej}(\mathrm{a}^2_{\mathrm{neg}}, 0). && (19)\\
\mathrm{rej}(\mathrm{a}^2_{\mathrm{neg}}, 2) &\leftarrow (\mathrm{assert}(\mathrm{a} \leftarrow))^1. && (20)\\
\mathrm{rej}(\mathrm{a}^2, 2) &\leftarrow (\mathrm{assert}(\mathbf{not}\,\mathrm{a} \leftarrow))^1. && (21)\\
\mathrm{rej}(\mathrm{a}^2_{\mathrm{neg}}, 0) &\leftarrow \mathrm{rej}(\mathrm{a}^2_{\mathrm{neg}}, 2). && (22)\\
\mathrm{u} &\leftarrow \mathbf{not}\,\mathrm{u}, \mathbf{not}\,\mathrm{a}^2, \mathbf{not}\,\mathrm{a}^2_{\mathrm{neg}}. && (23)
\end{aligned}$$

The rules (12) to (14) simulate the first evolution step – they are 2 rewritten program rules and one default assumption. Rules (15) and (16) are rewritten program rules for the second evolution step. In this step, two new rules can be asserted – (17) and (18) are the corresponding assertable rules. (19) is a default assumption, (20) to (22) are rejection rules and (23) is a totality constraint.

$P_{\mathcal{E}}$ has exactly one stable model

$$M = \{\, \mathrm{a}^1_{\mathrm{neg}}, (\mathrm{assert}(\mathrm{a} \leftarrow))^1, \mathrm{a}^2, (\mathrm{assert}(\mathbf{not}\,\mathrm{a} \leftarrow))^2, \mathrm{rej}(\mathrm{a}^2_{\mathrm{neg}}, 2), \mathrm{rej}(\mathrm{a}^2_{\mathrm{neg}}, 0) \,\} \enspace .$$

It directly corresponds to the single evolution stable model $\mathcal{M} = (M_1, M_2)$ of P given $\mathcal{E}$ where $M_1 = \{\, \mathrm{assert}(\mathrm{a} \leftarrow) \,\}$ and $M_2 = \{\, \mathrm{a}, \mathrm{assert}(\mathbf{not}\,\mathrm{a} \leftarrow) \,\}$.

4 Soundness and Completeness

The following 2 theorems show how the stable models of the transformed program correspond to the evolution stable models of the input program. Only sketches of proofs are provided, their full versions can be found in [30].

Theorem 1 (Soundness). *Let P be an evolving logic program, $\mathcal{E} = (E_1, E_2, \dots, E_n)$ an event sequence, N a stable model of $P_{\mathcal{E}}$,*

$$M_i = \{\, A \in \mathcal{L}_{\mathrm{assert}} \mid A^i \in N \,\} \text{ for all } i \in \{\, 1, 2, \dots, n \,\} \enspace .$$

Then $(M_1, M_2, \dots, M_n)$ is an evolution stable model of P given $\mathcal{E}$.

Proof (sketch). Let $(P_1, P_2, \dots, P_n)$ be the evolution trace associated with the evolution interpretation $\mathcal{M} = (M_1, M_2, \dots, M_n)$. According to Def. 4, $\mathcal{M}$ is an evolution stable model of P given $\mathcal{E}$ iff for every $i \in \{\, 1, 2, \dots, n \,\}$ M_i is a dynamic stable model of $(P_1, P_2, \dots, P_{i-1}, P_i \cup E_i)$. Hence we choose one arbitrary but fixed $j \in \{1, 2, \ldots, n\}$ and show that M_j is a dynamic stable model of $\mathcal{P} = (P_1, P_2, \ldots, P_{j-1}, P_j \cup E_j)$.

M_j contains exactly those atoms that have their corresponding j-indexed counterpart inferred by rules in $P^j_{\mathcal{E}}$ as defined in Def. 5. What we need to show is that each rule of $P^j_{\mathcal{E}}$ either corresponds to some rule in $P_1, P_2, \ldots, P_j, E_j$, or is used to simulate the rule-rejection mechanism behind Dynamic Logic Programming, or has no effect on the model.

It can be seen quite easily that rewritten program rules and rewritten event rules correspond to rules in $P_1 = P$ and E_j, respectively. They just contain one extra literal in their body – the rejection guard that is used to block them in case they are rejected.

An assertable rule, added as a rewritten form of an original rule r, can only be fired in case an atom of the form $(\mathrm{assert}(r))^{i-1}$ is true in N. But then $\mathrm{assert}(r)$ is true in M_{i-1} and thus $r \in P_i$. On the other hand, if $r \in P_i$ for some $i \in \{2, 3, \ldots, j\}$, then $\mathrm{assert}(r) \in M_{i-1}$ and hence $(\mathrm{assert}(r))^{i-1} \in N$. So each rewritten program rule, rewritten event rule and assertable rule either corresponds to some rule in the dynamic logic program $\mathcal{P}$, or has no effect on the resulting model because it cannot be fired.

Default assumptions in $P^j_{\mathcal{E}}$ are present for all atoms of the program. They simulate the set of defaults from Def. 1 and contain, just like all the other rules before, the rejection guard in their body that can block their usage in case a higher level rule rejects them by having an opposite literal in its head and its body satisfied in N.

The rejection rules together with the totality constraints can be proved to behave as follows:

1. For each atom A^j appearing in $P^j_{\mathcal{E}}$ they force exactly one of A^j and A^j_{neg} to be a member of N.
2. They infer an atom $\mathrm{rej}(L^j, i)$ with $i > 0$ iff some rule $r \in \mathrm{Rej}^i(\mathcal{P}, M_j)$ has L in its head.
3. They infer an atom $\mathrm{rej}(L^j, 0)$ iff L is a default literal $\mathbf{not}\, A$ and $\mathbf{not}\, A \notin \mathrm{Def}(\mathcal{P}, I)$.

The first point implies that the resulting model will be consistent with respect to the j-indexed versions of original literals. Correct simulation of the rule-rejection mechanism is a consequence of the second point. The third point ensures that only the appropriate subset of default assumptions is used.

Using the propositions from the previous paragraphs, it can be proved (by induction on the number of applications of the immediate consequence operator) that M_j is indeed a dynamic stable model of $\mathcal{P}$. □

Theorem 2 (Completeness). *Let P be an evolving logic program, $\mathcal{E} = (E_1, E_2, \ldots, E_n)$ an event sequence, $\mathcal{M} = (M_1, M_2, \ldots, M_n)$ an evolution stable model of P given $\mathcal{E}$, $(P_1, P_2, \ldots, P_n)$ the evolution trace associated with $\mathcal{M}$ and*

$$\mathcal{P}_i = (P_1, P_2, \ldots, P_{i-1}, P_i \cup E_i) \text{ for all } i \in \{1, 2, \ldots, n\}.$$

Furthermore, let

$$\begin{aligned} N = {} & \{\, L^i \mid i \in \{\, 1, 2, \ldots, n \,\} \wedge M_i \models L \wedge L^i \textit{ appears in } P_{\mathcal{E}} \,\} \\ & \cup \{\, \mathrm{rej}(L^i, k) \mid 1 \leq k \leq i \leq n \wedge (\exists r \in \mathrm{Rej}^k(\mathcal{P}_i, M_i))(H(r) = L) \,\} \\ & \cup \{\, \mathrm{rej}(A^i_{\mathrm{neg}}, 0) \mid i \in \{\, 1, 2, \ldots, n \,\} \wedge \mathbf{not}\, A \notin \mathrm{Def}(\mathcal{P}_i, M_i) \,\} \quad . \end{aligned}$$

Then N is a stable model of $P_{\mathcal{E}}$.

Proof (sketch). Let $R = \mathrm{least}(P_{\mathcal{E}} \cup N^-)$. We need to prove that $N^* = R$. This can be proved in three steps:

1. In the first step we must prove for every literal L of $\mathcal{L}_{\mathrm{assert}}$ and all $j \in \{\, 1, 2, \ldots, n \,\}$ that $L_j \in N \iff L_j \in R$. This can be proved by complete induction on j, using ideas very similar to those in the proof of soundness.
2. The second step is to prove that N and R are identical on the set of atoms of the form $\mathrm{rej}(L^j, i)$ for all $L \in \mathcal{L}_{\mathrm{assert}}$, every $j \in \{\, 1, 2, \ldots, n \,\}$ and every $i \in \{\, 0, 1, \ldots, j \,\}$. If $\mathrm{rej}(L^j, i) \in N$, then some rule $r \in \mathrm{Rej}^i(\mathcal{P}_j, M_j)$ has L in its head. This rule must have been rejected by some other rule r'. $P^j_{\mathcal{E}}$ must contain a rule corresponding to r' that will cause the presence of appropriate rejection rules. Consequently, $\mathrm{rej}(L^j, i)$ will eventually be added to R. A similar idea can be used to prove the converse implication.
3. The last matter that needs to be proved is that none of the totality constraints in $P_{\mathcal{E}}$ has been broken, i.e. that $u \notin R$. This can be proved by contradiction: consider one of the constraints if broken. Then for some atom $A \in \mathcal{L}_{\mathrm{assert}}$ we have $\mathbf{not}\, A^j, \mathbf{not}\, A^j_{\mathrm{neg}} \in R$ and also that both A^j and A^j_{neg} appear in $P_{\mathcal{E}}$. Furthermore, $\mathbf{not}\, A^j, \mathbf{not}\, A^j_{\mathrm{neg}} \in N^-$ and hence $A^j, A^j_{\mathrm{neg}} \notin N$. But then we have both $M_j \not\models A$ and $M_j \not\models \mathbf{not}\, A$ – a contradiction. □

5 Complexity of the Transformation

The computational complexity of the proposed transformation is interesting from multiple viewpoints:

- it directly influences the computational complexity of the implementation of EVOLP that is based on it [27],
- it allows to identify the most time-consuming parts of the transformation which can in turn be optimized to perform better,
- it reveals the branching factor that EVOLP is capable of, i.e. it demonstrates the expressivity of EVOLP.

The rules for generating the transformed program are quite simple, so the algorithm performing the transformation will also be reasonably simple. What really matters is the size and number of rules of the transformed program. The bigger the transformed program will be, the longer it will take to generate it and perform any further processing. We are also interested in which group of rules is the biggest and how it can be made smaller.

The size of each generated rule is either constant (default assumptions, totality constraints and propagating rejection rules) or just constantly bigger than the corresponding original rule. Therefore, we will concentrate on the number of generated rules. First we will derive both a lower and an upper bound for the number of rules of the transformed program. After we have the bounds, we will draw some conclusions. For the rest of this section we will assume P is a finite evolving logic program and $\mathcal{E} = (E_1, E_2, \ldots, E_n)$ is a sequence of finite events.

5.1 Lower Bound

We know the transformed program $P_{\mathcal{E}}$ contains $n|P|$ rewritten program rules and $\sum_{j=1}^{n} |E_j|$ rewritten event rules. So a very simple lower bound for $|P_{\mathcal{E}}|$ is:

$$|P_{\mathcal{E}}| \geq n|P| + \sum_{j=1}^{n} |E_j| \ . \tag{24}$$

Equality can be achieved only if $P = E_1 = E_2 = \ldots = E_n = \emptyset$. Otherwise, $P_{\mathcal{E}}$ will also contain some default assumptions and rejection rules.

5.2 Number of Assertable Rules

In order to derive an upper bound for $|P_{\mathcal{E}}|$, we will first need to make an approximation of the number of assertable rules. Let A be the set of all assertable rules in $P_{\mathcal{E}}$. In Appendix A it is shown that

$$|A| \leq |P| \frac{n^3 - n}{6} + \sum_{j=1}^{n} |E_j| \frac{(n-j)^3 + 5(n-j)}{6} \ . \tag{25}$$

It is also shown that in case we disallow nested asserts (i.e. a rule within an assert atom must not contain another assert atom in its head), we have

$$|A| \leq |P| \frac{n^2 - n}{2} + \sum_{j=1}^{n} (n-j)|E_j| \ . \tag{26}$$

5.3 Upper Bound

We already know the number of rewritten program rules and rewritten event rules in the transformed program and an upper bound for the number of assertable rules. Now we need to deal with the default assumptions, rejection rules and totality constraints.

How many default assumptions can there be? Both P and the events are finite so only a finite set of atoms from $\mathcal{L}_{\text{assert}}$ can be used in them. Let this set be $\mathcal{L}_{P,\mathcal{E}}$. Each atom in this set can generate up to n default assumptions.

Each rewritten program rule, rewritten event rule and assertable rule can generate at most 2 rejection rules. Two of these rules are needed to generate a totality constraint.

Taken together, we have

$$|P_{\mathcal{E}}| \leq \frac{7}{2} \left(n|P| + \sum_{j=1}^{n} |E_j| + |A| \right) + n|\mathcal{L}_{P,\mathcal{E}}| \enspace . \tag{27}$$

If we use the approximation of $|A|$ (25), we get the following inequality:

$$|P_{\mathcal{E}}| \leq \frac{7}{2} \Bigg(n|P| + \sum_{j=1}^{n} |E_j| \\ + |P|\frac{n^3 - n}{6} + \sum_{j=1}^{n} |E_j| \frac{(n-j)^3 + 5(n-j)}{6} \Bigg) + n|\mathcal{L}_{P,\mathcal{E}}|$$

which can be further simplified to

$$|P_{\mathcal{E}}| \leq \frac{7}{2} \left(|P|\frac{n^3 + 5n}{6} + \sum_{j=1}^{n} |E_j| \left(\frac{(n-j)^3 + 5(n-j)}{6} + 1 \right) \right) + n|\mathcal{L}_{P,\mathcal{E}}| \enspace .$$

When n is large and program sizes are considered as parameters, we can use the big-oh notation to get

$$|P_{\mathcal{E}}| = |P| \cdot \mathcal{O}\big(n^3\big) + \sum_{j=1}^{n} |E_j| \cdot \mathcal{O}\big((n-j)^3\big) + n|\mathcal{L}_{P,\mathcal{E}}| \enspace . \tag{28}$$

In case of programs without nested asserts we can use (26) to derive

$$|P_{\mathcal{E}}| \leq \frac{7}{2} \left(|P|\frac{n^2 + n}{2} + \sum_{j=1}^{n} (n-j+1)|E_j| \right) + n|\mathcal{L}_{P,\mathcal{E}}| \enspace ,$$

or, for large n,

$$|P_{\mathcal{E}}| = |P| \cdot \mathcal{O}\big(n^2\big) + \sum_{j=1}^{n} |E_j| \cdot \mathcal{O}(n-j) + n|\mathcal{L}_{P,\mathcal{E}}| \enspace . \tag{29}$$

5.4 Conclusion

In this section we examined how big the transformed program can get. Probably the most obvious and also a very important observation is that the lower bound (24) for $|P_{\mathcal{E}}|$ implies that the transformed program grows with n, no matter how big the events are. Hence for large values of n and small events this can lead to an intractably large transformed program even for tractably large inputs.

The main reason for this is that the transformed program captures all the possible evolutions of the input program. The expressivity EVOLP encompasses, especially the possibility of arbitrary branching based on intermediate models, makes it intractable to compute all the possible evolutions of even small programs. In case we are not interested in all or many of the possible evolutions, this transformation is not suitable as a basis for an implementation. For example when using EVOLP as an executable specification of a Multi-Agent System, a constructive view of the language as taken in [26] is more appropriate.

There are, however, also other possible uses of EVOLP where we do care about the possible evolutions. An example is reasoning about possible futures or on-line planning as a part of a deliberation of an agent. Verification of a Multi-Agent System specified in EVOLP is another. We believe the transformation is appropriate for such applications of EVOLP because the high computational complexity inherent in these problems is delegated to an answer set solver which is already optimized to deal with it.

From the upper bound (28) for $|P_{\mathcal{E}}|$ we can also see that the size of the transformed program depends on the size of the input program, size of events and n at most polynomially. Furthermore, if we use only (or mostly) rules without nested asserts, (29) implies we can lower the power of n that $|P_{\mathcal{E}}|$ grows with.

6 Conclusion and Future Work

We have defined a transformational semantics for Evolving Logic Programs and proved that it is sound and complete. We also examined the computational complexity of the transformation and identified situations in which it is practically applicable. These include reasoning about possible futures, on-line planning and verification of systems specified in EVOLP.

Future work can be devoted to optimizations of the transformation. In particular, the current transformation generates a number of unnecessary default assumptions and rejection rules. This was useful because it made its definition and proofs of soundness and completeness simpler. Now that these proofs are ready, we can concentrate on optimizing the transformation and prove more easily what optimizations are safe to perform. In many situations it is also possible to share rules among evolution steps which is another source of possible future optimizations. The third issue worth examining is the possibility of having a larger transformed program that performs better with the current answer set solvers.

Another line of work that can be followed involves generalizing the transformation. The definition can be extended to a language with classical negation. Another interesting issue is that of identifying a class of evolving logic programs with variables that is groundable with intuitive results and is general enough to be usable in practise.

References

1. Subrahmanian, V.S., Bonatti, P., Dix, J., Eiter, T., Kraus, S., Ozcan, F., Ross, R.: Heterogeneous Agent Systems. MIT Press, Cambridge (2000)
2. Dix, J., Zhang, Y.: IMPACT: a multi-agent framework with declarative semantics. In: Bordini, et al. (eds.) [11], ch. 3

3. Hindriks, K., de Boer, F., van der Hoek, W., Ch. Meyer, J.-J.: Agent programming in 3APL. Int. J. of Autonomous Agents and Multi-Agent Systems 2(4), 357–401 (1999)
4. Dastani, M., van Riemsdijk, M.B., Ch.Meyer, J.-J.: Programming multi-agent systems in 3APL. In: Bordini, et al. (eds.) [11], ch. 2
5. Bordini, R., Hübner, J., Vieira, R.: Jason and the Golden Fleece of agent-oriented programming. In: Bordini, et al. (eds.) [11], ch. 1
6. Costantini, S., Tocchio, A.: A logic programming language for multi-agent systems. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 1–13. Springer, Heidelberg (2002)
7. Bracciali, A., Demetriou, N., Endriss, U., Kakas, A., Lu, W., Stathis, K.: Crafting the mind of a PROSOCS agent. Applied Artificial Intelligence 20(4-5) (2006)
8. Thielscher, M.: Reasoning Robots: The Art and Science of Programming Robotic Agents. Springer, Heidelberg (2005)
9. De Giacomo, G., Lesprance, Y., Levesque, H.J.: ConGolog, a concurrent programming language based on the situation calculus. Artificial Intelligence 121(1-2), 109–169 (2000)
10. Mascardi, V., Martelli, M., Sterling, L.: Logic-based specification languages for intelligent software agents. Theory and Practice of Logic Programming 4(4), 429–494 (2004)
11. Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.): Multi-Agent Programming: Languages, Platforms and Applications. Multiagent Systems, Artificial Societies, and Simulated Organizations, vol. 15. Springer, Heidelberg (2005)
12. Bordini, R.H., Braubach, L., Dastani, M., Seghrouchni, A.E.F., Gomez-Sanz, J.J., Leite, J., O'Hare, G., Pokahr, A., Ricci, A.: A survey of programming languages and platforms for multi-agent systems. Informatica 30(1), 33–44 (2006)
13. Sakama, C., Inoue, K.: Updating extended logic programs through abduction. In: Gelfond, M., Leone, N., Pfeifer, G. (eds.) LPNMR 1999. LNCS (LNAI), vol. 1730. Springer, Heidelberg (1999)
14. Zhang, Y., Foo, N.Y.: Updating logic programs. In: Procs. of ECAI 1998. John Wiley & Sons, Chichester (1998)
15. Eiter, T., Fink, M., Sabbatini, G., Tompits, H.: On properties of update sequences based on causal rejection. Theory and Practice of Logic Programming 2(6) (2002)
16. Leite, J.A., Pereira, L.M.: Generalizing updates: From models to programs. In: Dix, J., Moniz Pereira, L., Przymusinski, T.C. (eds.) LPKR 1997. LNCS (LNAI), vol. 1471. Springer, Heidelberg (1998)
17. Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C.: Dynamic updates of non-monotonic knowledge bases. The Journal of Logic Programming 45(1-3), 43–70 (2000)
18. Leite, J.A.: Evolving Knowledge Bases. IOS Press, Amsterdam (2003)
19. Alferes, J.J., Banti, F., Brogi, A., Leite, J.A.: The refined extension principle for semantics of dynamic logic programming. Studia Logica 79(1), 7–32 (2005)
20. Leite, J.A.: On some differences between semantics of logic program updates. In: Lemaître, C., Reyes, C.A., González, J.A. (eds.) IBERAMIA 2004. LNCS (LNAI), vol. 3315, pp. 375–385. Springer, Heidelberg (2004)
21. Homola, M.: Various semantics are equal on acyclic programs. In: Leite, J.A., Torroni, P. (eds.) Procs. of the 5th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA V). LNCS, vol. 3487, pp. 78–95. Springer, Heidelberg (2004)
22. Alferes, J.J., Pereira, L.M., Przymusinska, H., Przymusinski, T.C.: LUPS – a language for updating logic programs. Artificial Intelligence 138(1&2) (June 2002)

23. Eiter, T., Fink, M., Sabbatini, G., Tompits, H.: A framework for declarative update specifications in logic programs. In: IJCAI 2001, pp. 649–654. Morgan-Kaufmann, San Francisco (2001)
24. Alferes, J.J., Brogi, A., Leite, J.A., Pereira, L.M.: Evolving logic programs. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 50–61. Springer, Heidelberg (2002)
25. Gelfond, M., Lifschitz, V.: Logic programs with classical negation. In: Warren, D., Szeredi, P. (eds.) Proceedings of the 7th international conference on logic programming, pp. 579–597. MIT Press, Cambridge (1990)
26. Leite, J.A., Soares, L.: Adding evolving abilities to a multi-agent system. In: Inoue, K., Satoh, K., Toni, F. (eds.) CLIMA 2006. LNCS (LNAI), vol. 4371, pp. 246–265. Springer, Heidelberg (2007)
27. Slota, M., Leite, J.A.: EVOLP: an implementation. In: Sadri, F., Satoh, K. (eds.) Proceedings of the 8th Workshop on Computational Logic in Multi-Agent Systems (CLIMA VIII) (2008); (System Description, in this volume)
28. Alferes, J.J., Brogi, A., Leite, J.A., Pereira, L.M.: Logic programming for evolving agents. In: Klusch, M., Omicini, A., Ossowski, S., Laamanen, H. (eds.) CIA 2003. LNCS (LNAI), vol. 2782, pp. 281–297. Springer, Heidelberg (2003)
29. Banti, F., Alferes, J.J., Brogi, A.: Operational semantics for DyLPs. In: Bento, C., Cardoso, A., Dias, G. (eds.) EPIA 2005. LNCS (LNAI), vol. 3808, pp. 43–54. Springer, Heidelberg (2005)
30. Slota, M.: Transformational semantics and implementation of evolving logic programs. Master's thesis, Univerzita Komenského (May 2007), http://slotik.info/

A Upper Bound for the Number of Assertable Rules

In this Appendix we derive an upper bound for the number of assertable rules in the transformed program. We will assume P is a finite evolving logic program and $\mathcal{E} = (E_1, E_2, \ldots, E_n)$ is a sequence of finite events. Let A be the set of all assertable rules in the transformational equivalent $P_{\mathcal{E}}$ of P given $\mathcal{E}$. We will need some more declarative characterization of the rules in A in order to work with its cardinality. The following Definition, Lemmas and Theorem provide such characterization:

Definition 6. *Let $E_0 = \emptyset$. We define*

$$A_1 \overset{\text{def}}{=} \{\, r \mid (\exists r_1 \in P)(H(r_1) = \text{assert}(r)) \,\} \quad , \tag{30}$$

for all $i \in \{\, 2, 3, \ldots, n-1 \,\}$

$$\begin{aligned} A_i \overset{\text{def}}{=} & \{\, r \mid (\exists r_1 \in A_{i-1})(H(r_1) = \text{assert}(r)) \,\} \\ & \cup \{\, r \mid (\exists r_2 \in E_{i-1})(H(r_2) = \text{assert}(r_1) \wedge H(r_1) = \text{assert}(r)) \,\} \end{aligned} \tag{31}$$

and for all $j \in \{\, 1, 2, \ldots, n-1 \,\}$ also

$$\overline{A_j} \overset{\text{def}}{=} \bigcup_{i=1}^{j} A_i \cup \{\, r \mid (\exists r_1 \in E_j)(H(r_1) = \text{assert}(r)) \,\} \quad . \tag{32}$$

Remark 1. Let $j \in \{1, 2, \ldots, n\}$. Each assertable rule in $P_{\mathcal{E}}^{j}$ is fully determined by its assertion guard, i.e. if we know that it has the assertion guard $(\mathrm{assert}(r))^{i-1}$ and $r = (L \leftarrow \mathit{Body}.)$, then the assertable rule must be:

$$L^{j} \leftarrow \mathit{Body}^{j}, (\mathrm{assert}(r))^{i-1}, \mathbf{not}\, \mathrm{rej}(L^{j}, i).$$

We will make use of this fact in order to make some formulations simpler.

Lemma 1. *Let $i \in \{1, 2, \ldots, n-1\}$, $j \in \{i, i+1, \ldots, n-1\}$ and $r \in A_i$. Then $P_{\mathcal{E}}^{j+1}$ contains an assertable rule with the assertion guard* $(\mathrm{assert}(r))^{j}$*.*

Proof. We will prove by induction on i.

1 Let $r \in A_1$. Then some rule $r_1 \in P$ exists such that $H(r_1) = \mathrm{assert}(r)$. Let $j \in \{1, 2, \ldots, n-1\}$. Then $P_{\mathcal{E}}^{j}$ must contain a rewritten program rule with $(\mathrm{assert}(r))^{j}$ in its head and therefore $P_{\mathcal{E}}^{j+1}$ must contain an assertable rule with the assertion guard $(\mathrm{assert}(r))^{j}$.

2 We assume the claim holds for i and prove it for $i+1$. Let $r \in A_{i+1}$ and let $j \in \{i+1, i+2, \ldots, n-1\}$. Two cases are possible:

   (a) Some rule $r_1 \in A_i$ exists such that $H(r_1) = \mathrm{assert}(r)$. By the induction hypothesis we have that $P_{\mathcal{E}}^{j}$ contains an assertable rule with the assertion guard $(\mathrm{assert}(r_1))^{j-1}$. This rule has $(\mathrm{assert}(r))^{j}$ in its head. Hence $P_{\mathcal{E}}^{j+1}$ contains an assertable rule with the assertion guard $(\mathrm{assert}(r))^{j}$.

   (b) Some rule $r_2 \in E_i$ exists such that $H(r_2) = \mathrm{assert}(r_1)$ and $H(r_1) = \mathrm{assert}(r)$. Then $P_{\mathcal{E}}^{i}$ contains a rewritten event rule with $(\mathrm{assert}(r_1))^{i}$ in its head. Hence $P_{\mathcal{E}}^{j}$ will contain an assertable rule with the assertion guard $(\mathrm{assert}(r_1))^{i}$ and $(\mathrm{assert}(r))^{j}$ in its head. Therefore $P_{\mathcal{E}}^{j+1}$ must contain an assertable rule with the assertion guard $(\mathrm{assert}(r))^{j}$. □

Lemma 2. *Let $j \in \{1, 2, \ldots, n-1\}$ and $r \in \overline{A_j}$. Then $P_{\mathcal{E}}^{j}$ contains a rule with* $(\mathrm{assert}(r))^{j}$ *in its head.*

Proof. Assume that $r \in \overline{A_j}$. Two cases are possible:

a) $r \in A_i$ for some $i \in \{1, 2, \ldots, j\}$. Then by Lemma 1 we have that $P_{\mathcal{E}}^{j+1}$ contains an assertable rule with the assertion guard $(\mathrm{assert}(r))^{j}$. Hence $P_{\mathcal{E}}^{j}$ must contain a rule with $(\mathrm{assert}(r))^{j}$ in its head.

b) Some rule $r_1 \in E_j$ exists such that $H(r_1) = \mathrm{assert}(r)$. Then $P_{\mathcal{E}}^{j}$ contains a rewritten event rule with $(\mathrm{assert}(r))^{j}$ in its head. □

Lemma 3. *Let $j \in \{1, 2, \ldots, n-1\}$ and let r be a rule over $\mathcal{L}_{\mathrm{assert}}$. If $P_{\mathcal{E}}^{j}$ contains a rule with* $(\mathrm{assert}(r))^{j}$ *in its head, then $r \in \overline{A_j}$.*

Proof. We will prove by complete induction on j.

1 The basis can be inferred from the inductive step with $j = 1$ (the third case doesn't have to be examined because $P_{\mathcal{E}}^{1}$ contains no assertable rules).

2 We assume the proposition holds for all $i \in \{1, 2, \ldots, j-1\}$ and prove it for j. Let's consider three cases:

(a) If $P_{\mathcal{E}}^{j}$ contains a rewritten program rule r_1^* with $(\mathrm{assert}(r))^j$ in its head, then P contains a rule r_1 such that $H(r_1) = \mathrm{assert}(r)$. Hence $r \in A_1 \subseteq \overline{A_j}$.

(b) If $P_{\mathcal{E}}^{j}$ contains a rewritten event rule r_1^* with $(\mathrm{assert}(r))^j$ in its head, then E_j contains a rule r_1 such that $H(r_1) = \mathrm{assert}(r)$. Hence $r \in \overline{A_j}$.

(c) If $P_{\mathcal{E}}^{j}$ contains an assertable rule with $(\mathrm{assert}(r))^j$ in its head, then it must be of the form

$$(\mathrm{assert}(r))^j \leftarrow \mathit{Body}^j, (\mathrm{assert}(r_1))^{i-1}, \mathbf{not}\,\mathrm{rej}((\mathrm{assert}(r))^j, i).$$

where $r_1 = (\mathrm{assert}(r) \leftarrow \mathit{Body}.)$ and $i \leq j$. So $P_{\mathcal{E}}^{i-1}$ must contain a rule with $(\mathrm{assert}(r_1))^{i-1}$ in its head and by the induction hypothesis we have $r_1 \in \overline{A_{i-1}}$. Two cases are possible again:

i. $r_1 \in A_h$ for some $h \in \{1, 2, \ldots, i-1\}$. Then $r \in A_{h+1} \subseteq \overline{A_j}$.

ii. Some rule $r_2 \in E_{i-1}$ exists such that $H(r_2) = \mathrm{assert}(r_1)$. We also have $H(r_1) = \mathrm{assert}(r)$. So $r \in A_i \subseteq \overline{A_j}$. □

Theorem 3. *Let $j \in \{1, 2, \ldots, n-1\}$ and let r be a rule over $\mathcal{L}_{\mathrm{assert}}$. $P_{\mathcal{E}}^{j}$ contains a rule with $(\mathrm{assert}(r))^j$ in its head iff $r \in \overline{A_j}$.*

Proof. Follows directly from Lemmas 2 and 3. □

As a consequence of the theorem we have

$$|A| = \sum_{j=1}^{n} (n-j) \left|\overline{A_j}\right| \tag{33}$$

because each rule $r \in \overline{A_j}$ will generate $n-j$ assertable rules, one in each of $P_{\mathcal{E}}^{j+1}, P_{\mathcal{E}}^{j+2}, \ldots, P_{\mathcal{E}}^{n}$. Now we can make an approximation of $|A|$. According to (30), (31) and (32) we have for all $j \in \{1, 2, \ldots, n-1\}$

$$|A_j| \leq |P| + \sum_{i=1}^{j-1} |E_i| \ , \qquad \left|\overline{A_j}\right| \leq j|P| + |E_j| + \sum_{i=1}^{j} (j-i)|E_i| \ .$$

Furthermore, by (33) we have

$$\begin{aligned} |A| = \sum_{j=1}^{n} (n-j) \left|\overline{A_j}\right| &\leq \sum_{j=1}^{n} (n-j) \left(j|P| + |E_j| + \sum_{i=1}^{j} (j-i)|E_i| \right) \\ &= |P| \sum_{j=1}^{n} j(n-j) + \sum_{j=1}^{n} (n-j)|E_j| + \sum_{j=1}^{n} (n-j) \sum_{i=1}^{j} (j-i)|E_i| \ . \end{aligned} \tag{34}$$

First let's solve the first sum:

$$\sum_{j=1}^{n} j(n-j) = n \sum_{j=1}^{n} j - \sum_{j=1}^{n} j^2 = \frac{n^3 - n}{6} \ . \tag{35}$$

The third sum can be simplified as follows:

$$\begin{aligned}\sum_{j=1}^{n}(n-j)\sum_{i=1}^{j}(j-i)|E_i| &= \sum_{i=1}^{n}|E_i|\sum_{j=1}^{n-i}j((n-i)-j) \\ &= \sum_{i=1}^{n}|E_i|\frac{(n-i)^3-(n-i)}{6} \quad .\end{aligned} \tag{36}$$

By (34), (35) and (36) we now have

$$|A| \leq |P|\frac{n^3-n}{6} + \sum_{j=1}^{n}|E_j|\frac{(n-j)^3+5(n-j)}{6} \quad .$$

We can also put some extra restrictions on the input program and then look at the number of assertable rules. For example, if we disallow nested asserts (i.e. a rule within an assert atom must not contain an assert atom in its head), then we have $|A_1| \leq |P|$ and $|A_j| = 0$ for all $j \in \{ 2, 3, \ldots, n-1 \}$. Hence $\left|\overline{A_j}\right| \leq |P|+|E_j|$ for all $j \in \{ 1, 2, \ldots, n-1 \}$ and

$$|A| \leq \sum_{j=1}^{n}(n-j)(|P|+|E_j|) = |P|\frac{n^2-n}{2} + \sum_{j=1}^{n}(n-j)|E_j| \quad .$$

Language Constructs for Multi-agent Programming[⋆]

Louise Dennis, Michael Fisher, and Anthony Hepple

Department of Computer Science, University of Liverpool, Liverpool, UK
{L.A.Dennis,MFisher,A.J.Hepple}@liverpool.ac.uk

Abstract. In this paper we are concerned with proposing, analyzing and implementing simple, yet flexible, constructs for multi-agent programming. In particular, we wish to extend programming languages based on the BDI style of logical agent model with two such constructs, namely *constraints* and *content/context sets*. These two aspects provide sufficient expressive power to allow us to represent, simply and with semantic clarity, a wide range of organisational structures for multi-agent systems. We not only introduce this approach, but provide its formal semantics, through modification of an operational semantics based on the core of AGENTSPEAK, 3APL and METATEM. In addition, we provide illustrative examples by simulating both constraints and content/context sets within the *Jason* interpreter for AGENTSPEAK. In summary, we advocate the use of these simple constructs in many logic-based BDI languages, by appealing to their applicability, simplicity and clear semantics.

1 Introduction

We characterise an agent as an autonomous software component having certain goals and being able to communicate with other agents in order to accomplish these goals [26]. The ability of agents to act independently, to react to unexpected situations and to cooperate with other agents, has made them a popular choice for developing software in a number of areas. At one extreme there are agents that are used to search the INTERNET, navigating autonomously in order to retrieve information; these are relatively lightweight agents, with few goals but significant domain-specific knowledge. At the other end of the spectrum, there are agents developed for independent process control in unpredictable environments. This second form of agent is often constructed using complex software architectures, and they have been applied in areas such as real-time process control [20,14]. Perhaps the most impressive use of such agents is as part of the real-time fault monitoring and diagnosis carried out within the NASA Deep Space One mission [16].

The key reason why an agent-based approach is advantageous in the modelling and programming of autonomous systems is that it permits the clear and concise representation, not just of *what* the autonomous components within the system do, but *why* they do it. This allows us to abstract away from the low-level control aspects and to concentrate on the key feature of autonomy, namely the goals of the component and the choices it makes. Thus, in modelling a system in terms of agents, we often describe each agent's

[⋆] Work partially supported by EPSRC under grant EP/D052548.

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 137–156, 2008.

beliefs and *goals*, which in turn determine the agent's *intentions*. Such agents then make decisions about what action to perform, given their current beliefs and goals/intentions. This kind of approach has been popularised through the influential BDI (Belief-Desire-Intention) model of agent-based systems [20] and, although this representation of behaviour using *mental* notions is initially unusual, it has several benefits. The first is that, ideally, it abstracts away from low-level issues: we simply present some goal that we wish to be achieved, and we expect the agent to act in what we would consider a reasonable, or *rational*, way given such a goal. Secondly, because we are used to understanding and predicting the behaviour of rational agents, the behaviour of autonomous software should be relatively easy for humans to understand and predict too. The modelling of complex systems, even space exploration systems, in terms of rational agents captured within the BDI approach, has been very successful [14,23,22]. Unsurprisingly, this has led to many novel (usually, logic-based) programming languages based (at least in some part) upon this model; these are often termed *BDI Languages*.

When researchers and developers experimented with these languages and used them for a wider variety of applications it became clear that *open* multi-agent systems did not scale well without a further abstraction to capture the working relationships between agents, groups of agents and their environment [10,18]. Furthermore, only a cursory study of human societies is needed to realise that increased levels of productivity and efficiency are realised by societies with effective frameworks that encourage cooperative behaviour amongst its population. The study of agent interaction, cooperation and organisation is therefore of current interest in the agent research community [17,11] but, although a wide variety of BDI languages have been developed [1], few have strong and flexible mechanisms for *organising* multiple agents, and those that do provide no agreement on their organisational mechanisms. Thus, while BDI languages have converged to a common core relating to the activity of individual agents [5], no such convergence is apparent in terms of multi-agent structuring and organisation.

1.1 Agent Organisation

In this section we briefly describe some of the more influential agent-organisational proposals for the purpose of highlighting typical applications of the language constructs we describe in the next section. For a concise but isolated description of each proposal we refer the reader to our companion paper [12].

With a respected philosophical view on agent co-operation, Cohen and Levesque produced a significant paper "Teamwork" [3] in which they persuasively argue that a team of agents should *not* be modelled as an aggregate agent and propose new (logical) concepts of *joint intentions*, *joint commitments* and *joint persistent goals* to ensure that teamwork does not break down due to any divergence of individual team members' beliefs or intentions. Tidhar [24] introduced the concept of *team-oriented programming* with social structure; an agent-centred approach that defines joint goals and intentions for teams but stops short of forcing individual team members to adopt those goals and intentions. Ferber *et al.* [8] present the case for an organisational-centred approach to the design and engineering of complex multi-agent systems. They propose a model for designing multi-agent systems in terms of *agents*, *roles* and *groups*. Agents and groups are proposed as distinct first class entities. Dignum, Esteva, Sierra and Vázquez-Salceda made formal [7] and practical [6,25] contributions to a method of agent organisation that enjoys much current popularity [17]; that of *institutions*. An electronic institution aims to provide an *open* framework in which agents can contribute to the goals of *society* without sacrificing its own self-interest. A key concept is that of institutional norms. Perhaps the most noteworthy aspect of these proposals is the change of focus from the agents themselves onto the interactions that take place between agents.

In this paper we consider extending basic BDI languages with simple, yet powerful, constructs that allow the development of a wide range of organisational structures. Thus, in Section 2,we introduce the concepts behind the new constructs, in particular showing how they relate to typical BDI language semantics. To clarify this further, in Section 3, we provide the core semantics of a subset of AgentSpeak [19,2] incorporating the new concepts; we call this language AgentSpeak$^-$. To show how these concepts can be used, in Section 4 (and in a companion paper [12]), we outline how a variety of organisational structures can be expressed using these simple constructs, present several case studies, and even provide some implementations within AgentSpeak. Finally, in Section 5, we provide concluding remarks. Thus, this paper introduces the concepts and provides semantics, while [12] shows how this approach captures the core of the leading organisational mechanisms.

We begin by introducing the concepts; we do this by first considering the core operational aspects of BDI languages, describe some agent-organisational abstractions and then show how our new concepts affect agent operation.

2 Introducing the Concepts

Although all BDI languages have a family resemblance, their syntax and semantics can vary immensely. We therefore use a loose unifying framework for our discussion into which we believe most BDI languages will fit[1], though not always elegantly.

Our semantic framework assumes that a BDI language specifies the behaviour of an agent in terms of the agent's current state, $\mathcal{S}$, which changes over time and a fixed specification, $\mathcal{SP}$, which does not. Thus, an agent is viewed as a tuple $< \mathcal{S}, \mathcal{SP} >$. $\mathcal{S}$ consists, amongst other things, of a set of beliefs, B. The BDI programming language then has a process for determining whether a given belief b follows from the current state which we will write as $\mathcal{S} \models b$, since these are often logical mechanisms.

The BDI programming language has a specific operation, $\mathbf{select\_instruction}$, which acts on the state according to the specification in order to determine the next instruction to be executed and another, $\mathbf{modify}$, which modifies the state according to the specification and the selected instruction. The execution of an agent can therefore be viewed as repeated application of the transition rule

$$< \mathcal{S}, \mathcal{SP} > \rightarrow < \mathbf{modify}(\mathcal{SP}, \mathcal{S}, \mathbf{select\_instruction}(\mathcal{SP}, \mathcal{S})), \mathcal{SP} > \quad . \qquad (1)$$

We assume that both $\mathcal{S}$ and $\mathcal{SP}$ are made up of a number of sets or stacks (e.g., of beliefs) and use the notation $\mathcal{S}[S_1 \setminus S_2]$ to indicate the state $\mathcal{S}$ in which the set S_1 has been replaced by S_2.

[1] Indeed, in [5] such a framework was used to provide a common semantic basis for 3APL, AgentSpeak, MetateM, etc.

Note. This framework should not be interpreted as assuming that a given BDI language has *explicit* constructs for $\mathbf{select\_instruction}$ and $\mathbf{modify}$, but that most BDI languages can be expressed in terms of the operation of appropriate versions of these functions.

We also assume that a BDI language contains a set of plans (or rules), P, which are used by the $\mathbf{select\_instruction}$ operation. These plans may either be a part of $\mathcal{S}$ or $\mathcal{SP}$. We assume such plans are *triggered* in some fashion by $\mathcal{S}$. In some cases they are triggered by the composition of the beliefs (e.g., METATEM [9]), in some by the goals (e.g., 3APL [13,4]) and in some by explicit trigger events (e.g., *Jason* [2] interpreter for AGENTSPEAK [19]).

To simplify matters, we use an abstraction of a plan, describing it as

$$t \leftarrow \{g\}b \ .$$

Thus, plans comprise; a *trigger*, t; a *guard* (checked against the agent's beliefs), g; and a *body*, b, which specifies an instruction (or sequence of instructions) to be executed. In languages where only beliefs are used to trigger plans this can be written as

$$\top \leftarrow \{g\}b \ .$$

In order to trigger plans, the language requires some component of the current state $\mathcal{S}$ which activates the trigger. We treat this as a set, T, and write the triggering process as $T \models_t t$.

Finally, we will use the notation $Ag \models_a p$ to indicate that a plan, p, is applicable for an agent, Ag. The semantics of this for a basic[2] BDI agent is

$$\frac{\mathbf{app\_cond}(t \leftarrow \{g\}b, Ag)}{Ag \models_a t \leftarrow \{g\}b} \tag{2}$$

where $\mathbf{app\_cond}$ are the agent language's applicability conditions. In most languages

$$\mathbf{app\_cond}(t \leftarrow \{g\}b, Ag) = ((T \models_t t) \wedge (\mathcal{S} \models g)) \ .$$

Notes

- Again we do not necessarily expect these operations associated with plans to be explicit in the languages (e.g., T may be a stack of goals and $T \models_t g$ may be the process of matching the head (or prefix) of this stack).
- There may be other applicability checking processes within the language (e.g., applicability of actions) — we represent all of these within $Ag \models_a$.
- Application of a plan results in an instruction to modify the state either directly (e.g., $+b$ appears in the body of the plan and is an instruction to add b to B) or indirectly (e.g., the body of the plan is integrated into an *intention* or other part of the state which is then used for further planning or to govern subsequent actions and changes of belief).

[2] I.e., a BDI agent whose semantics has not been modified with the constructs we describe later.

Given the above, we below consider the two aspects we wish to introduce. The first, though influenced by the representation of agent groups in METATEM [9], is independent of the underlying language for agents. The only restrictions we put on any underlying language is that, as in most BDI-based languages (and as described above), there are logical mechanisms for explicitly describing *beliefs* and *goals*, and possibly *plans* and *intentions*[3].

2.1 Content and Context Sets

Assuming that the underlying language can describe the behaviour of an agent as above, we now extend the concept of agent with two sets, `Content` and `Context`. Conceptually, the agent's `Content` describes the set of agents it contains, while the agent's `Context` describes a set of agents it is contained within, although in practice the relationship between an agent and its `Content/Context` might vary. For example, an agent's `Content` might describe the set of agents it has some influence over and the agent's `Context` the set of agents it is influenced by. Similarly, an agent's `Content` might be viewed as those agents that it has recruited, and its `Context` those agents it has been recruited by. Alternatively, an agent might represent a location and its `Content` the agents that at that location. The addition of `Content` and `Context` to each agent provides significant flexibility for agent organisation. Agent teams, groups or organisations, which might alternatively be seen as separate entities, are now just agents with non-empty `Content`. This allows these organisations to be hierarchical and dynamic, and so, as we will see later, provides possibilities for a multitude of other coordinated behaviours. Similarly, agents can have several agents within their `Context`. Not only does this allow agents to be part of several organisational structures simultaneously, but it allows the agent to benefit from `Context` representing diverse attributes or behaviours. So an agent might be in a context related to its physical locality (e.g., agents in the same set are 'close' to each other), yet also might be in a context that provides certain roles or abilities. Intriguingly, agents can be within many, overlapping and diverse, contexts. This gives the ability to produce complex organisations, in a way similar to multiple inheritance in traditional object/concept systems. For some sample configurations, see Fig. 1.

An important aspect is that this whole structure is very dynamic. Agents can move in and out of `Content` and `Context`, while new agents (and, hence, organisations) can be spawned easily and discarded. No single agent maintains a representation of the entire structure, allowing for the possibility of a range of organisations, from the *transient* to the *permanent*. From the above it is clear that there is no enforced distinction between an agent and an agent organisation. All are agents; all may be treated similarly.

Our proposals prohibit cyclical structures and require that all structural changes occur with the consent of those agents whose `Content` or `Context` sets are affected. Also, it is essential that the agent's internal behaviour, be it a program or a specification, has direct access to both the `Content` and `Context`, allowing each agent to become more than just a 'dumb' container.

[3] We also require that message-passing between agents is provided; this is standard in most languages.

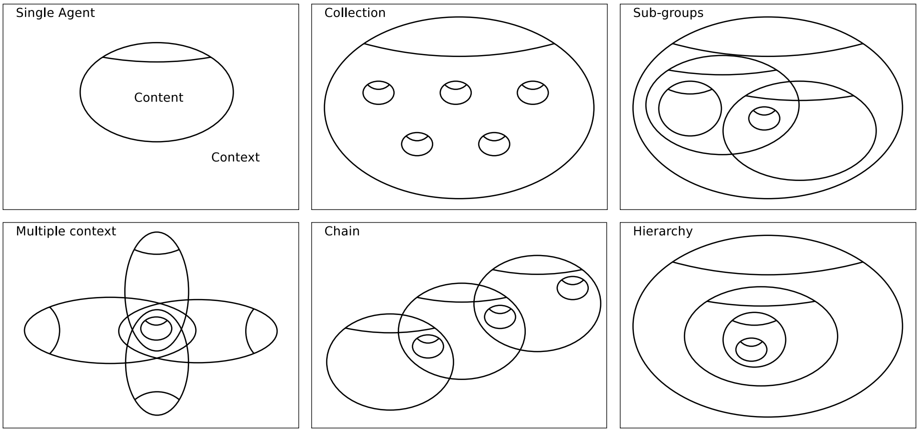


Fig. 1. A selection of potential content/context patterns

Semantics. The simplicity of the above approach allows us to provide a few general operational rules for managing the content and context sets. We extend the agent's state, $\mathcal{S}$, with a content set, (Cn), and a context set, (Cx), and add four new instructions into the language $+ag^{cn}$ (add ag to the content set), $-ag^{cn}$ (remove ag from the content set) and $+ag^{cx}$, $-ag^{cx}$ for adding and removing agents from the context set. We also add four new constructs into the trigger component, T:

$$\begin{array}{ll} entered\_content(ag) & left\_content(ag) \\ entered\_context(ag) & left\_context(ag) \quad . \end{array}$$

Add two new constructs into our language of guards:

$$\begin{array}{l} in\_content(ag) \\ in\_context(ag) \quad . \end{array}$$

We then extend the **modify** operation with the rules:

$$\mathbf{modify}(\mathcal{SP}, \mathcal{S}, +ag^{cn}) = \mathcal{S}[Cn \setminus Cn \cup \{ag\}, T \setminus T \cup entered\_content(ag)] \quad (3)$$

$$\mathbf{modify}(\mathcal{SP}, \mathcal{S}, -ag^{cn}) = \mathcal{S}[Cn \setminus Cn - \{ag\}, T \setminus T \cup left\_content(ag)] \quad (4)$$

and two analogous ones for the context. These rules extend both the state's content/ context and the trigger set, T. This allows plans to be triggered by changes in these sets. (e.g., plans of the form

$$entered\_content(Ag) \leftarrow \{in\_content(Ag)\} send(self, Ag, plan)$$

may be written which are triggered by the addition of a new agent Ag to the content set, into sending that agent a plan).

We also extend the belief inference process to include checking membership of Cn and Cx:

$$\frac{ag \in Cn}{\mathcal{S} \models in\_content(ag)} \quad (5)$$

$$\frac{ag \in Cx}{\mathcal{S} \models in\_context(ag)} \quad . \tag{6}$$

It should be noted that in many languages it may be possible to streamline these extensions (e.g., by merging the triggering of plans and the update of content/context sets – see Section 3).

2.2 Constraints

The second basic component we suggest is necessary for many meaningful multi-agent structures is that of *constraints*. A constraint consists of additional guards that may be appended to plans/rules and actions and is typically provided by an agent's context. This, for example, allows permissions to be modelled.

Semantics. As with groups we extend the agent's state, $\mathcal{S}$, with a constraint set, (C). C is treated as a set of pairs of a trigger and a guard, written $[t \Rightarrow g]$. Depending on the language, it may be desirable to add other pairs to this set, for instance if actions may have guards and there is an applicability process for actions then action/guard pairs may also be useful within constraints. Again, we add new instructions into the language $+new\_constraint^c$ (add $new\_constraint$ to C) and $-new\_constraint^c$ (remove $new\_constraint$ from C), which are analogous to the previous add/remove operators. We then extend our applicability checking process, $Ag \models_a$ to

$$\frac{\forall [t \Rightarrow g'] \in C.\ \mathcal{S} \models g' \quad \mathbf{app\_cond}(t \leftarrow \{g\}b, Ag)}{Ag \models_a t \leftarrow \{g\}b} \quad . \tag{7}$$

So, in many languages, this becomes

$$\frac{\forall [t \Rightarrow g'] \in C.\ \mathcal{S} \models g' \quad T \models_t t \quad \mathcal{S} \models g}{Ag \models_a t \leftarrow \{g\}b} \quad . \tag{8}$$

Similar modifications can be made to the operational semantics of action applicability (internal or external) and any other relevant components of $\mathcal{S}$ and $\mathcal{SP}$.

It should be noted that constraints make relatively little sense in a single agent environment (where guards on plans and actions are sufficient) it is only in a multi-agent environment where a member of `Context` may wish to provide guards to a pre-existing plan or action that such constraints become useful.

Before going on to providing the semantics of a more comprehensive language (in Section 3), we first consider the properties of such semantic extensions.

2.3 Properties of Groups and Constraints

In addition to the generic operational semantics for groups and constraints we present here some properties that ideally any system implementing them should obey. We discuss when these hold in a system that implements these concepts using our suggested rules.

Firstly one agent should believe that another is in its `Content/Context` if, and only if, that agent is *actually* in its `Content/Context`. We express this as:

$$CONTAINS(ag) \Rightarrow BEL(in\_content(ag)) \tag{9}$$
$$CONTAINED\_BY(ag) \Rightarrow BEL(in\_context(ag)) \tag{10}$$
$$BEL(in\_content(ag)) \Rightarrow CONTAINS(ag) \tag{11}$$
$$BEL(in\_context(ag)) \Rightarrow CONTAINED\_BY(ag)\ . \tag{12}$$

For the operational semantics presented above we interpret $CONTAINS(ag)$ as $ag \in Cn$, $CONTAINED\_BY(ag)$ as $ag \in Cx$ and $BEL(\phi)$ as $\mathcal{S} \models \phi$.

Let us assume that the the formulae $in\_content(ag)$ and $in\_context(ag)$ are "reserved" in an implementation, i.e., such formulae can not appear in the belief base either when an agent is initialised or through any belief revision process and that there is no way they can be inferred through belief inference except by the use of (5) and (6). (Many BDI languages have mechanisms for reserving key-words which could be extended for this purpose.) If this is the case then (9–12) follow directly from rules (5) and (6). If it is not possible to restrict the formulae that an agent might believe (e.g., it will accept any formula as a belief if sent it by a trusted external agent) then any system adopting our operational semantics only satisfies (9 and 10), unless additional safeguards are implemented.

Turning to constraints, we would expect any well-behaved system implementing constraints to satisfy

$$(Ag \models_a P) \Rightarrow ((Ag \models_a P) \wedge (C = \emptyset)) \tag{13}$$

i.e., if a plan is applicable given some constraints, then it is also applicable if there are no constraints. In our operational semantics this follows from (7) if we observe that when $C = \emptyset$ the condition $\forall [t \Rightarrow g'] \in C.\ \mathcal{S} \models g'$ reduces to $\top$ and that C is not referred to elsewhere in the rule.

Rao and Georgeff [21] state a number of interesting properties they suggest BDI languages might wish to satisfy and it would be tempting to examine some of these in relation to groups (in particular those relating intentions and beliefs (with $INTEND(\phi)$ interpreted as $\phi \in T$)). However this work assumes that intentions are expressed as temporal formulae and that belief inference includes temporal and causal reasoning. Our triggers are *not* expressed in this way and in fact may include formulae (such as $entered\_content(ag)$) which refer to events that have occurred rather than states of the world the agent wishes to bring about.

As mentioned previously, it is essential that the agent's internal behaviour, be it a program or a specification, has direct access to both the `Content` and `Context`, allowing each agent to become more than just a 'dumb' container. It can provide access to, provide services for, and share information or behaviours with, its `Content`, as is demonstrated by Fig. 2; here agent j moves into the separate context of agents i and k (perhaps i represents an auctioneer agent who provides j with the bidding rules, whilst k is the agent on whose behalf j is bidding). Our proposals encourage the sharing of plans, beliefs and constraints as structural changes take place but also allow the dissemination of new knowledge. Indeed we can state the following, very general, result.

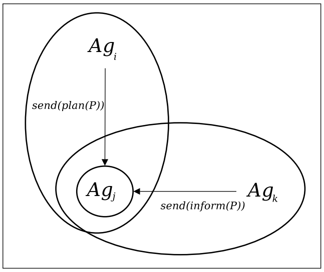


Fig. 2. Sharing plans and information

Theorem 1. *If agent A moves into a new context C and*

- *the context agent, C is willing to send plans/beliefs/constraints/etc to A, and*
- *agent A incorporates these plans/beliefs/constraints/etc sent from its new context,*

then A has the new plans/beliefs/constraints/etc provided by its new context.

Aside. There is an obvious counterpart of Theorem 1 whereby A can *ask* its context for information (plans/beliefs/etc). Once it moves into a new context then A has access to the new information/capabilities provided by its context.

Theorem 1 above has many caveats! However, these mainly cover situations where agents choose not to cooperate. In a cooperative scenario, where an agent provides plans/beliefs/constraints/etc to any new members of its content, and where agents accept those items from their new context, then Theorem 1 says that an agent effectively has the information and capabilities provided by its context (in addition to its own).

Importantly, this is seamless. The particular example of *constraints* is informative. Constraints effectively prohibit certain planning choices. Thus, through Theorem 1 we know that an agent with certain choices (e.g. of how to achieve a goal) will inherit the constraints (restrictions) from its context. If the agent is in multiple contexts, the agent must make choices satisfying *all* the constraints received from its contexts. Effectively, the agent is constrained by the union of all its contexts and so its behaviour must follow the intersection of behaviours allowed by each context.

This aspect is exhibited in the *cookery* example in Section 4.3, but is also closely linked to organisational aspects such as *norms* in that the agent's choices are modified by the contexts (organisations) in which it finds itself.

3 A Simple BDI Language: AGENTSPEAK$^-$

We will conclude our discussion of formal semantics with a simple example showing how our framework provides a practical methodology for extending existing BDI languages. Let us consider an extremely simple agent programming language based on AGENTSPEAK [19,2]; we will call this language AGENTSPEAK$^-$.

Syntax. Our language uses ground first-order formulae for beliefs, actions and goals. A plan is a triple of a goal, a guard and a stack of instructions (called here *deeds* following AIL [5]). An Agent is a triple of a set of beliefs, a stack of deeds and a set of plans. This is shown in Fig. 3.

$$\begin{aligned}
belief &:= \text{Ground first-order formula}\\
action &:= \text{Ground first-order formula}\\
goal &:= \text{Ground first-order formula}\\
plan &:= goal : set(belief) \leftarrow stack(deed)\\
agent &:= < set(belief), stack(deed), set(plan) >\\
deed &:= action \mid +belief \mid -belief \mid +!goal
\end{aligned}$$

Fig. 3. Syntax of AGENTSPEAK$^-$

Operational Semantics. An operational semantics for AGENTSPEAK$^-$ is provided in the form of the four transition rules in Fig. 4. In these semantics $do(a)$ is an operation in an agent's interface that causes it to perform the action, a, and then returns a set of messages in the form of deeds, $+!received(sender, \phi)$, which instruct the agent to handle the message ϕ from agent $sender$. In this language, therefore, perception has to be handled by an explicit $perception$ action which then returns messages from the environment as if from another agent. Finally, ';' represents the $cons$ function on stacks, '@' represents the join function, and '$random$' indicates random selection of an element from a set.

$$\frac{do(a) = msg}{< B, a; D, P > \rightarrow < B, msg@D, P >} \tag{14}$$

$$\frac{}{< B, +b; D, P > \rightarrow < B \cup \{b\}, D, P >} \tag{15}$$

$$\frac{}{< B, -b; D, P > \rightarrow < B - \{b\}, D, P >} \tag{16}$$

$$\frac{body = random(\{b \mid g : G \leftarrow b \in P \wedge G \subseteq B\})}{< B, +!g; D, P > \rightarrow < B, body@D, P >} \tag{17}$$

Fig. 4. Operational Semantics of AGENTSPEAK$^-$

Note. This is not intended as a practical example of a BDI language. For a start the language is entirely ground and makes no use of unification. Secondly the rather crude use of the deed stack to organise both planning and message handling/perception is likely to cause quite strange behaviour in any real agent setting.

Framework	-	AGENTSPEAK$^-$
$\mathcal{SP}$	-	P
$\mathcal{S}$	-	$< B, D >$
T	-	D
$\mathcal{S} \models b$	-	$b \subseteq B$
$T \models_t t$	-	$t = hd(D)$
$\textbf{app\_cond}(gl : g \leftarrow b)$	-	$g \subseteq B$
$\textbf{modify}((B, D), P, +b)$	-	$(B \cup \{b\}, D)$
$\textbf{modify}((B, D), P, -b)$	-	$(B - \{b\}, D)$
$\textbf{modify}((B, D), P, ds)$	-	$(B, ds@D)$
$\textbf{select\_instruction}((B, a; D), P)$	-	$do(a)$
$\textbf{select\_instruction}((B, +b; D), P)$	-	$+b$
$\textbf{select\_instruction}((B, -b; D), P)$	-	$-b$
$\textbf{select\_instruction}((B, +!g; D), P)$	-	$random(\{b \mid p \in P \wedge Ag \models_a p\})$

Fig. 5. Mapping our Framework to AGENTSPEAK$^-$

Extension to the Simple BDI Language. Fig. 5 shows how this language fits into our earlier framework. Modifying these semantics according to our content/context and constraints framework now gives us the language semantics shown in Fig. 6

In fact this extension can be improved upon based on the details of our languages. For instance we can omit the $entered\_content()$ and $left\_content()$ and use $+ag^{cn}$ and $-ag^{cn}$ as plan triggers if we like, changing (3) to

$$\frac{body = random(\{b| + ag^{cn} : G \leftarrow b \in P \wedge G \subseteq B \wedge \forall [+ag^{cn} \Rightarrow G'] \in C.\, G' \subseteq B\}}{< B, +ag^{cn}; D, Cn, Cx, C, P > \rightarrow < B, body@D, Cn \cup \{ag\}, Cx, C, P >} \tag{28}$$

4 Using the Concepts

We will briefly discuss some illustrative examples of the use of constraints and content/context sets (sometimes termed *groups*) in organisational and multi-agent settings. A group, G, is simply an agent (again called G) whose `Contents` are the set of agents within the group.

We begin by considering a few common aspects of agent organisations, and then examine two case studies in more detail. Note that a more comprehensive review of how many agent organisational approaches can be modelled using our constructs is provided in [12].

4.1 Shared Beliefs

Being a member of all but the least cohesive groups/organisations requires that some shared beliefs exist between the members. Making the (contentious) assumption that all agents are honest and that joining a group is both individual rational and group rational, let agent i hold a belief set BS_i and assume the programming language contains the instruction $addBelief(Beliefs)$ with the semantics

$$\textbf{modify}(\mathcal{SP}, \mathcal{S}, addBelief(Bs)) = \mathcal{S}[B \setminus B \cup Bs].$$

$$\frac{do(a) = msg \qquad \forall[a \Rightarrow G] \in C.\, G \subseteq B}{< B, a; D, Cn, Cx, C, P > \rightarrow < B, msg@D, Cn, Cx, C, P >} \tag{18}$$

$$\frac{}{< B, +b; D, Cn, Cx, C, P > \rightarrow < B \cup \{b\}, D, Cn, Cx, C, P >} \tag{19}$$

$$\frac{}{< B, -b; D, Cn, Cx, C, P > \rightarrow < B - \{b\}, D, Cn, Cx, C, P >} \tag{20}$$

$$\frac{}{< B, +c^c; D, Cn, Cx, C, P > \rightarrow < B, D, Cn, Cx, C \cup \{c\}, P >} \tag{21}$$

$$\frac{}{< B, -c^c; D, Cn, Cx, C, P > \rightarrow < B, D, Cn, Cx, C - \{c\}, P >} \tag{22}$$

$$\frac{body = random(\{b \mid g : G \leftarrow b \in P \wedge G \subseteq B \wedge \forall[g \Rightarrow G'] \in C.\, G' \subseteq B\})}{< B, +!g; D, Cn, Cx, C, P > \rightarrow < B, body@D, Cn, Cx, C, P >} \tag{23}$$

$$\frac{}{< B, +ag^{cn}; D, Cn, Cx, C, P > \rightarrow < B, +!entered\_content(ag); D, Cn \cup \{ag\}, Cx, C, P >} \tag{24}$$

$$\frac{}{< B, -ag^{cn}; D, Cn, Cx, C, P > \rightarrow < B, +!left\_content(ag); D, Cn - \{ag\}, Cx, C, P >} \tag{25}$$

$$\frac{}{< B, +ag^{cx}; D, Cn, Cx, C, P > \rightarrow < B, +!entered\_context(ag); D, Cn, Cx \cup \{ag\}, C, P >} \tag{26}$$

$$\frac{}{< B, -ag^{cx}; D, Cn, Cx, C, P > \rightarrow < B, +!left\_context(ag); D, Cn, Cx - \{ag\}, C, P >} \tag{27}$$

Fig. 6. AGENTSPEAK$^-$ extended to multi-agents

Suppose a (group) agent i has the plan:

$$entered\_content(Ag) \leftarrow \{in\_content(Ag)\}send(i, Ag, inform(BS_i))$$

and agent j has the plan:

$$received(Ag, j, inform(BS_i)) \leftarrow \{in\_context(Ag)\}addBelief(BS_i)$$

taken together these plans mean that if j joins the `Content` of i it gets sent the beliefs BS_i which it adds to its own belief base. This allows shared beliefs to be established.

The agent in receipt of the new beliefs may or may not disseminate them to the agents in its `Content`, depending on the nature and purpose of the group structure. Once held, beliefs are retained until contradicted or revised (for example, on leaving the group). It is worth noting here that these behaviours are merely suggestions of how our proposals can be used to implement shared beliefs, providing the developer has authorship of all agents.

4.2 Permissions and Obligations

A number of multi-agent proposals include concepts of permissions and obligations [1]. An agent within a group setting may or may not have the permission to perform a particular action or communicate in a particular fashion. This can be easily represented using constraints: for instance if agents in group, G, may not perform action a then the constraint $[a \Rightarrow \bot]$ can be communicated to them when they join G's `Content`.

It should be noted that in order for such a message to be converted into an actual constraint, the receiving agent would also need the plan:

$$received(Ag, i, constrain([a \Rightarrow g])) \leftarrow \{in\_context(Ag)\} + [a \Rightarrow g]^c \ .$$

This design deliberately allows varying degrees of autonomy among agents to be handled by the programmer.

Obligations are where a group member is *obliged* to behave in a particular fashion. This can be modelled if plans are treated as modifiable by the underlying agent language. Obligations can then be communicated as new plans.

```
/*---------------- initial beliefs ---------------- */
cooperative.

/*---------------- rules ------------------------- */
check_constraint(Plan, Arg)
  :-  not constraint_fails(Plan,Arg).

/* --------------- basic plans -------------------- */

/* How an agent responds to a group membership invitations */
+!join(Group)[source(Group)] : cooperative
  <-   .my_name(Me);
       +context(Group);
       .println("I believe I have the context of ", Group);
       .send(Group, achieve, accept(Me, Group)).
```

Fig. 7. A simple cooperative agent defined in AgentSpeak

4.3 Case Study 1: Cookery Agents

We now describe a case study which we have implemented in AGENTSPEAK using Jason. It concerns a simple *cook* agent who is provided with a number of plans by a chef agent, each for cooking a different meal. The cook's choice of plan is constrained by the `Context` in which it cooks.

Scenario. The chef of a restaurant hires a cook and provides a list of dishes from which the cook is free to choose when asked to prepare a meal. As diners arrive, their preferences are noted and the cook endeavours to choose a meal that satisfies all of the diners. Our cook was implemented as a simple, cooperative agent with the ability to enter the Context of other agents but without any domain abilities — it can't cook — see Fig. 7.

When hired, the cook agent receives plans for making risotto, steak and pizza. AgentSpeak code defining this behaviour is shown below.

```
+content(Agent)[source(self)]
  <-  .print("Sending ", Agent, " plans...");
      .send(Agent, tellHow, "+!cook(risotto)
                                : check_constraint(cook,risotto)
                                  <- make(risotto).");
      .send(Agent, tellHow, "+!cook(steak)
                                : check_constraint(cook,steak)
                                  <- make(steak).");
      .send(Agent, tellHow, "+!cook(pizza)
                                : check_constraint(cook,pizza)
                                  <- make(pizza).").
```

(Note that this sending of plans is triggered by the cook entering its Content.) When asked to prepare a meal without the constraints of any diners it prepares risotto; see Fig. 8(b). A meat eating diner then imposes their dislike for risotto by providing the cook with the constraint

```
constraint_fails(cook,risotto).
```

Now acting in the context of this meat eater, rather than making risotto, the chef prepares steak; see Fig. 8(c). Finally, a vegetarian diner invites the chef to join its Content and imposes the constraint constraint_fails(cook,steak), see Fig. 8(d). The agent, now a member of three contexts, must decide an appropriate course of action within the supplied constraints — it must not commit to cooking risotto or steak! Thus it is constrained to choose to prepare pizza; see below.

```
+content(Agent)[source(self)]
  <-  .print("Sending ", Agent, " my constraints");
      .send(Agent, tell, constraint_fails(cook,steak)).
```

Full execution output for this example is given below.

```
[chef] saying: inviting cook to join my content
[cook] saying: I believe I have the context of chef
[chef] saying: Sending cook plans...
[chef] saying: I consider cook to be in my content
[cook] doing: make(risotto)
***cook is making risotto***
[meatEater] saying: inviting cook to join my content
[cook] saying: I believe I have the context of meatEater
[meatEater] saying: Sending cook my constraints
[meatEater] saying: I consider cook to be a member of my content
[cook] doing: make(steak)
***cook is making steak***
[vegetarian] saying: inviting cook to join my group
[cook] saying: I believe I have the context of vegetarian
```

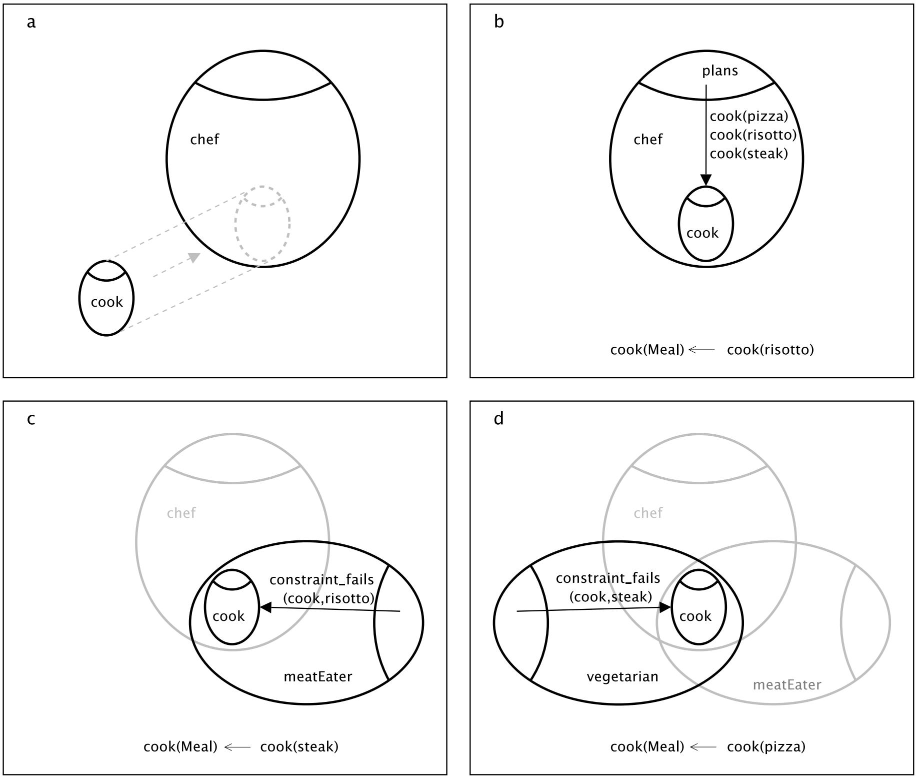


Fig. 8. A cook with multiple constraints

```
[vegetarian] saying: Sending cook my constraints
[vegetarian] saying: I consider cook to be in my content
[cook] doing: make(pizza)
***cook is making pizza***
```

4.4 Case Study 2: Self Deploying Agents

This example demonstrates the potential for software services that migrate across geographical spaces and deploy themselves in their new location.

Scenario. Co-ordination of disaster and rescue missions is a challenging problem for the authorities involved [15]. The deployment location, the number and nature of agencies (commissioned or voluntary) involved cannot be foreseen and speed of deployment is critical. Establishing fast and reliable communication channels between all parties, no matter what their individual resources are, is essential for effective co-ordination.

In our example, disaster recovery head quarters has a co-ordination agent, hq, that is mobilised to a wired network in the proximity of the disaster. hq has domain knowledge but no local knowledge or resources — it does not know which agencies are on the scene and cannot communicate outside of its host network. In order to effectively co-ordinate the rescue effort hq must seek help from a variety of *helper* agents that can carry communication to the operational agencies and provide information about local resources. Examples of help provided by such agents might be: WiFi communication; environmental sensors; public display points; media communications; and utility providers. The suitability of these agents might be determined by proximity, ability or cost.

On arrival hq broadcasts a '*services needed*' message requesting that agents with certain capabilities offer their services. The following code snippet illustrates an agent's generic recruitment plan, used to broadcast requests for services to the entire agent space, along with the plan to recruit a WiFi service.

```
/* Broadcast for local services*/
+!recruit(Service)
  <-  .broadcast(askIf, has_ability(Agent, Service)).
      ...
      !recruit(wifi).
```

Co-operative agents respond to hq's plea for help by sending a reply stating their abilities and confirming their willingness to join the *group* rescue effort. Below, we show an agent's plan for responding to requests for help.

```
/* Confirm ability and willingness to join */
+!help(Group, Service)
  : .my_name(Me) and has_ability(Me, Service)
  <-  .send(Group, tell, has_ability(Me, Service));
      .send(Group, achieve, accept(Me, Group)).
      ...
      !help(hq, wifi).
```

The plan has a guard that ensures only genuinely able agents respond, it confirms its ability and requests group membership. Note that in this case, the helper agent does not consider itself to be a member of the group until the group itself directly informs it of its membership — a hierarchical structure whereby membership is controlled by the group is appropriate in this scenario but our proposals also allow agents to control their own `Context`, as shown in Fig. 7.

On acknowledgement of group membership hq holds the belief $content(wifi)$, $wifi$ holds the belief $context(hq)$ and $wifi$ is provided with authentication procedures to apply to incoming connections; see below.

```
+!accept(Agent, Group)[source(Agent)]
  : is_useful(Agent,_)[source(self)]
  <-  +content(Agent);
      .send(Agent, tell, context(Group)).

+content(Agent)[source(self)]
  <-  is_useful(Agent, communicator);
      .send(Agent, tellHow, authentication).
```

Broadcasts of this nature are unavoidable when an agent has no knowledge of the system ahead of deployment. However the context/content mechanism provides a convenient and intuitive alternative that enables more efficient multi-cast communication; for example, our agent hq may have recruited a number of communicator agents to whom it wants to broadcast information, by creating a new agent to act as a container for the communicator agents, hq is able to send a message to all *communicators* — using the container agent as a proxy — with the $send(group, broadcast(message))$ message, where the agent $group$ receiving the $broadcast(message)$ message distributes it to all members of its `Content`. Once structures are formed, multi-cast communication of the following type become the norm:

```
send(communicators, broadcast(found(Survivor, Location))).
send(locators, broadcast(is_clear(Zone))).
```

Fig. 9 shows some of the structural changes that take place during deployment of our simple disaster management system.

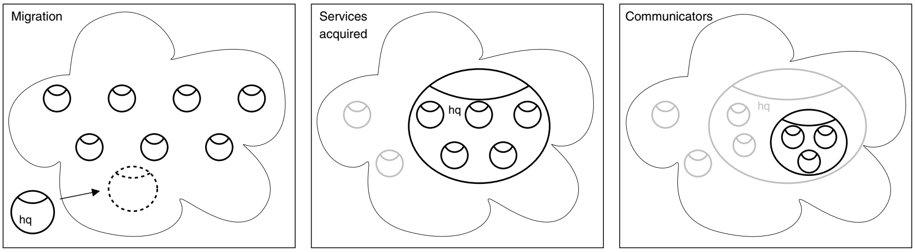


Fig. 9. The structural view during deployment

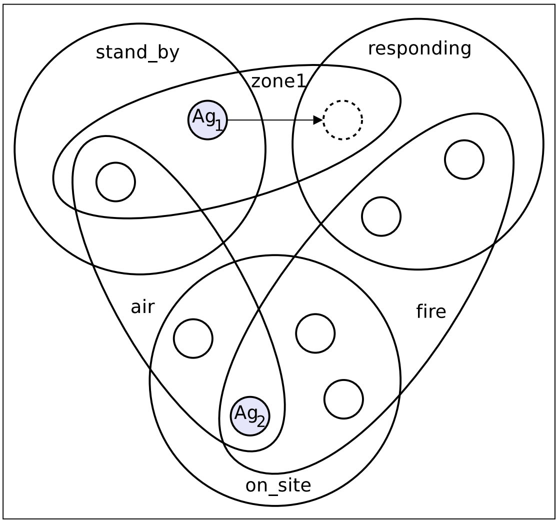


Fig. 10. The dynamic nature of search and rescue

One of the difficulties of disaster management where life saving rescue is required, is the prioritisation of rescue attempts and subsequent allocation of resources, particularly when the number, location and needs of victims changes throughout the rescue mission. Continuous re-assessment of the mission's priorities must take place yet pragmatic decisions must be made to ensure rescue teams are effectively deployed and do not, for example, waste time travelling between rescue sites. The context of a rescue team's current activity, their specialisms and location must be considered before allocating them to a rescue site. Our grouping constructs provide the flexibility to model the dynamic nature of these contexts and provides a useful bound for reasoning — reducing the search space for suitable rescue teams. Fig. 10 illustrates how our proposal intuitively deals with this situation. The diagram shows rescue agent Ag_1 standing by in $zone1$ ready to be deployed and its subsequent change of context if it were to respond to a call. Another agent Ag_2 that has both air and fire specialisms is currently attending a rescue site. Using this formalism it is easy to express autonomous behaviour on behalf of the rescue agents;

```
constraint_fails(respond, _) :-
  in_context(responding), in_context(on_site).
```

Giving the agents the above rule prevents them from responding to rescue requests whilst either on route to, or at the scene of a rescue, when combined with the plan below.

```
+!respond(Rescue) : check_constraint(respond, Rescue)
  <-  !attend(Rescue).
```

5 Concluding Remarks

In this paper we have proposed a simple extension to BDI languages that permits the development of complex multi-agent organisations. We have shown how the addition of both content and context sets, and constraints is semantically simple and appealing. The key aspect, particularly with contexts and constraints is that an agents behaviour may be modified, seamlessly, when the agent moves between contexts.

Although we provided a semantic definition for a simple BDI language, we gave this only for illustrative purposes. We expect that developers' favourite logical agent language could be extended in this way. Importantly, the semantic rules show how this logical extension can be added (relatively easily) to any appropriate BDI language.

Finally, we provided some simple examples here, and yet more examples in a companion paper [12], to illustrate and justify our statement that many agent organisational aspects can be modelling using our two simple concepts. These examples demonstrate how leading organisational and team-working concepts such as roles, joint-intentions and groups fit within our framework, a framework that, if incorporated into BDI languages will enable a consistent agent-organisation semantics across languages.

References

1. Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.): Multi-Agent Programming: Languages, Platforms and Applications. Springer, Heidelberg (2005)
2. Bordini, R.H., Hübner, J.F., Vieira, R.: Jason and the Golden Fleece of Agent-Oriented Programming. In: Bordini, et al. (ed.) [1], vol. 1, pp. 3–37

3. Cohen, P.R., Levesque, H.J.: Teamwork. Technical Report 504, SRI International, California, USA (1991)
4. Dastani, M., van Riemsdijk, M.B., Meyer, J.-J.C.: Programming Multi-Agent Systems in 3APL. In: Bordini, et al. (eds.) [1]
5. Dennis, L.A., Farwer, B., Bordini, R.H., Fisher, M., Wooldridge, M.: A Common Semantic Basis for BDI Languages. In: Proc. Seventh International Workshop on Programming Multiagent Systems (ProMAS). LNCS (LNAI). Springer, Heidelberg (to appear, 2007)
6. Esteva, M., de la Cruz, D., Sierra, C.: ISLANDER: an electronic institutions editor. In: Proc. 1st International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 1045–1052. ACM, New York (2002)
7. Esteva, M., Rodríguez-Aguilar, J.A., Sierra, C., Garcia, P., Arcos, J.L.: On the Formal Specification of Electronic Institutions. In: Sierra, C., Dignum, F.P.M. (eds.) AgentLink 2000. LNCS (LNAI), vol. 1991, pp. 126–147. Springer, Heidelberg (2001)
8. Ferber, J., Gutknecht, O.: A Meta-model for the Analysis and Design of Organizations in Multi-agent Systems. In: Proc. 3rd International Conference on Multi-Agent Systems (ICMAS), pp. 128–135. IEEE, Los Alamitos (1998)
9. Fisher, M.: MetateM: The Story so Far. In: Bordini, R.H., Dastani, M., Dix, J., Seghrouchni EI Fallah, A. (eds.) PROMAS 2005. LNCS (LNAI), vol. 3862, pp. 3–22. Springer, Heidelberg (2006)
10. Fox, M.S.: An Organizational View of Distributed Systems. IEEE Transactions on Systems, Man, and Cybernetics 11 (1981)
11. Grossi, D., Dignum, F., Dastani, M., Royakkers, L.: Foundations of Organizational Structures in Multiagent Systems. In: Proc. 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 690–697. ACM, New York (2005)
12. Hepple, A., Dennis, L., Fisher, M.: A Common Basis for Agent Organisations in BDI Languages. In: Proc. First International Workshop on Languages, Methodologies and Development Tools for Multi-Agent Systems (LADS). LNCS. Springer, Heidelberg (to appear, 2008)
13. Hindriks, K.V., de Boer, F.S., van der Hoek, W., Meyer, J.-J.C.: Agent Programming in 3APL. Autonomous Agents and Multi-Agent Systems 2(4), 357–401 (1999)
14. Jennings, N.R., Wooldridge, M.: Applications of Agent Technology. In: Agent Technology: Foundations, Applications, and Markets. Springer, Heidelberg (1998)
15. Kitano, H., Tadokoro, S.: RoboCup Rescue: A Grand Challenge for Multiagent and Intelligent Systems. AI Magazine 22(1), 39–52 (2001)
16. Muscettola, N., Nayak, P.P., Pell, B., Williams, B.: Remote Agent: To Boldly Go Where No AI System Has Gone Before. Artificial Intelligence 103(1-2), 5–48 (1998)
17. Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.): Coordination, Organization, Institutions and Norms in agent systems II (COIN). LNCS, vol. 4386. Springer, Heidelberg (2007)
18. Prietula, M., Carley, K., Gasser, L. (eds.): Simulating Organizations: Computational Models of Institutions and Groups. MIT Press, Cambridge (1998)
19. Rao, A.: AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In: Proc. 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW). LNCS, vol. 1038, pp. 42–55. Springer, Heidelberg (1996)
20. Rao, A.S., Georgeff, M.: BDI Agents: from theory to practice. In: Proc. First International Conference on Multi-Agent Systems (ICMAS 1995), pp. 312–319 (June 1995)
21. Rao, A.S., Georgeff, M.P.: Decision Procedures for BDI Logics. Journal of Logic and Computation 8(3), 293–343 (1998)
22. Sierhuis, M.: Multiagent Modeling and Simulation in Human-Robot Mission Operations (2006), http://ic.arc.nasa.gov/ic/publications

23. Sierhuis, M., Bradshaw, J.M., Acquisti, A., Hoof, R.V., Jeffers, R., Uszok, A.: Human-Agent Teamwork and Adjustable Autonomy in Practice. In: Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Nara, Japan (2003)
24. Tidhar, G.: Team-Oriented Programming: Preliminary Report. Technical Report 1993-41, Australian Artificial Intelligence Institute, Melbourne, Australia (1993)
25. Vázquez-Salceda, J., Dignum, V., Dignum, F.: Organizing Multiagent Systems. Journal of Autonomous Agents and Multi-Agent Systems 11(3), 307–360 (2005)
26. Wooldridge, M., Jennings, N.R.: Intelligent Agents: Theory and Practice. The Knowledge Engineering Review 10(2), 115–152 (1995)

$\mathcal{B}$-Tropos

Agent-Oriented Requirements Engineering Meets Computational Logic for Declarative Business Process Modeling and Verification*

Volha Bryl[1], Paola Mello[2], Marco Montali[2],
Paolo Torroni[2], and Nicola Zannone[1]

[1] DISI, University of Trento – Via Sommarive 14, 38100 Povo (TN), Italy
{bryl,zannone}@disi.unitn.it
[2] DEIS, University of Bologna – V.le Risorgimento 2, 40136 Bologna, Italy
{pmello,mmontali,ptorroni}@deis.unibo.it

Abstract. The work presented in this paper stands at the intersection of three diverse research areas: agent-oriented early requirements engineering, business process requirements elicitation and specification, and computational logic-based specification and verification. The analysis of business requirements and the specification of business processes are fundamental steps in the development of information systems. The first part of this paper presents $\mathcal{B}$-Tropos as a way to combine business goals and requirements with the business process model. $\mathcal{B}$-Tropos enhances a well-known agent-oriented early requirements engineering framework with declarative business process-oriented constructs, inspired by the DecSerFlow and ConDec languages. In the second part of the paper, we show a mapping of $\mathcal{B}$-Tropos onto $\mathcal{S}$CIFF, a computational logic-based framework for properties and conformance verification.

1 Introduction

This work proposes an integration of different techniques for information systems engineering, with the aim to reconcile requirements elicitation with declarative specification, prototyping, and analysis inside a single unified framework.

In tackling the requirements elicitation part, we take an agent-oriented perspective. Modeling and analyzing requirements of IT systems in terms of agents and their goals is an increasingly popular approach [20] which helps understanding the organizational setting where a system operates, as well as modeling the stakeholders' strategic interests, and finally documenting the rationale behind the design choices made. After system requirements elicitation is complete, one must define a corresponding business process. A very important issue that must be addressed at this stage is how to link the "strategic" business goals and

* This work has been partially funded by EU SENSORIA and SERENITY projects, by the national MIUR-FIRB TOCAI.IT and MIUR-PRIN 2005-011293 projects, and by the PAT MOSTRO project.

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 157–176, 2008.

requirements with the business process model [23]. Many problems arise from organizational theory and strategic management perspectives due to limits on particular resources (e.g., cost, time, etc.). Business strategies have a fundamental impact on the structure of enterprises leading to efficiency in coordination and cooperation within economic activities.

For our purpose, we have chosen Tropos [8], an agent-oriented software engineering methodology which uses the concepts of agent and goal from the early phases of the system development. Tropos has a number of interesting features, such as its goal- and agent-orientation, intuitive and expressive modeling notation, etc., which have made it to become popular. However, a drawback of Tropos and a number of similar methodologies is that they do not clearly define how to move from a requirements model to a business process model. For example, Tropos does not allow the modeling of temporal and data constraints between tasks assigned to agents: this means that when developing a business process, the corresponding Tropos model does not have enough information to define a temporal ordering between activities. Likewise, start and completion times, triggering events, deadlines, and many other aspects not necessarily related to the temporal dimension are essential elements in the description of a business process model, but they are not represented in Tropos models.

How to enhance Tropos with information that can be automatically used in the generation of a business process model is one of the aspects we address in this work. In particular, we have extended Tropos with declarative business process-oriented constructs, inspired by two recent graphical languages: DecSerFlow [34] and ConDec [33]. We enhance the characteristic goal-oriented approach of Tropos agents by introducing a high-level reactive, process-oriented dimension. We refer to the extended framework as to $\mathcal{B}$-Tropos. Furthermore, we show how both these complementary aspects could be mapped onto the $\mathcal{S}$CIFF language [4], which sits at the basis of a computational logic-based framework for the specification and verification of interaction protocols in open multi-agent systems. In the presentation of this work, we discuss the issue of time (ordering, deadlines, etc.) because it is an essential part of business process modeling, and because it is easy to explain by intuitive examples. However, $\mathcal{B}$-Tropos is not only a temporal extension of Tropos, but it covers also the treatment of conditions on process input/output data and other constraints.

The marriage of $\mathcal{B}$-Tropos with $\mathcal{S}$CIFF sets a link between specification, prototyping and analysis: in fact, $\mathcal{S}$CIFF specifications can be used to implement and animate logic-based agents [1], as well as to perform different verification tasks, such as properties verification [2] and conformance verification of a given execution trace [4]. Prototyping (animation) and analysis (properties and conformance verification) add value to $\mathcal{B}$-Tropos and can make it appealing to a large set of potential users. Early requirements engineers and process engineers will be able to test their models directly and get an immediate picture of the system being developed. Engineers testing the properties of the models will not have to resort to ad-hoc, error-prone translations of high-level models into the languages used to feed specifications into model checkers, since $\mathcal{B}$-Tropos can directly generate $\mathcal{S}$CIFF programs. Managers who need to monitor the correct behavior of a running system will have a $\mathcal{S}$CIFF specification of the system generated out of a $\mathcal{B}$-Tropos model automatically, and based on this specification they will be able to automatically check the compliance of the system using the SOCS-SI runtime and offline checking facilities [3].

In this work, we focus on specific aspects of this global vision. We define $\mathcal{B}$-Tropos and the mapping of $\mathcal{B}$-Tropos constructs onto the $\mathcal{S}$CIFF framework. To make the discussion more concrete, the proposed approach is applied to modeling and analyzing an intra-enterprise organizational model, focusing on the coordination of economic activities among different units of an enterprise collaborating to produce a specific product. The organizational model is an excerpt of a large case study under consideration within the national FIRB TOCAI.IT project.[1]

The structure of the paper is as follows. Section 2 briefly presents the Tropos methodology. Section 3 describes $\mathcal{B}$-Tropos. The $\mathcal{S}$CIFF framework is presented in Section 4, whereas Section 5 defines the mapping of $\mathcal{B}$-Tropos concepts to $\mathcal{S}$CIFF specifications. The paper ends with the overview of related work in Section 6 and conclusive remarks in Section 7.

2 The Tropos Methodology

Tropos [8] is an agent-oriented software engineering methodology tailored to describe and analyze socio-technical systems along the whole development process from requirements analysis up to implementation. One of its main advantages is the importance given to early requirements analysis. This allows one to capture *why* a piece of software is developed, behind *what* or *how*.

The methodology is founded on models that use the concepts of actors (i.e., agent and roles), goals, tasks, resources, and social dependencies between two actors. An *actor* is an active entity that has strategic goals and performs actions to achieve them. A *goal* represents a strategic interest of an actor. A *task* represents a particular course of actions that produce a desired effect. A *resource* represents a physical or an informational entity without intentionality. A *dependency* between two actors indicates that one actor depends on another in order to achieve some goal, execute some task, or deliver some resource. The former actor is called the *depender*, while the latter is called the *dependee*. The object around which the dependency centers, which can be a goal, a task, or a resource, is called the *dependum*. In the graphical representation, actors are represented as circles; goals, tasks and resources are respectively represented as ovals, hexagons and rectangles; and dependencies have the form *depender* → *dependum* → *dependee*.

From a methodological perspective, Tropos is based on the idea of building a model of a system that is incrementally refined and extended. Specifically, *goal*

[1] The TOCAI.IT project (RBNE05BFRK, `http://www.dis.uniroma1.it/~tocai/`) is a three-year, 4.5 Ml euro project on “Knowledge-oriented technologies for enterprise aggregation in Internet.” It involves a consortium of 11 Italian universities, the National Research Council, and three industrial partners in the ICT, engineering, and manufacturing sectors.

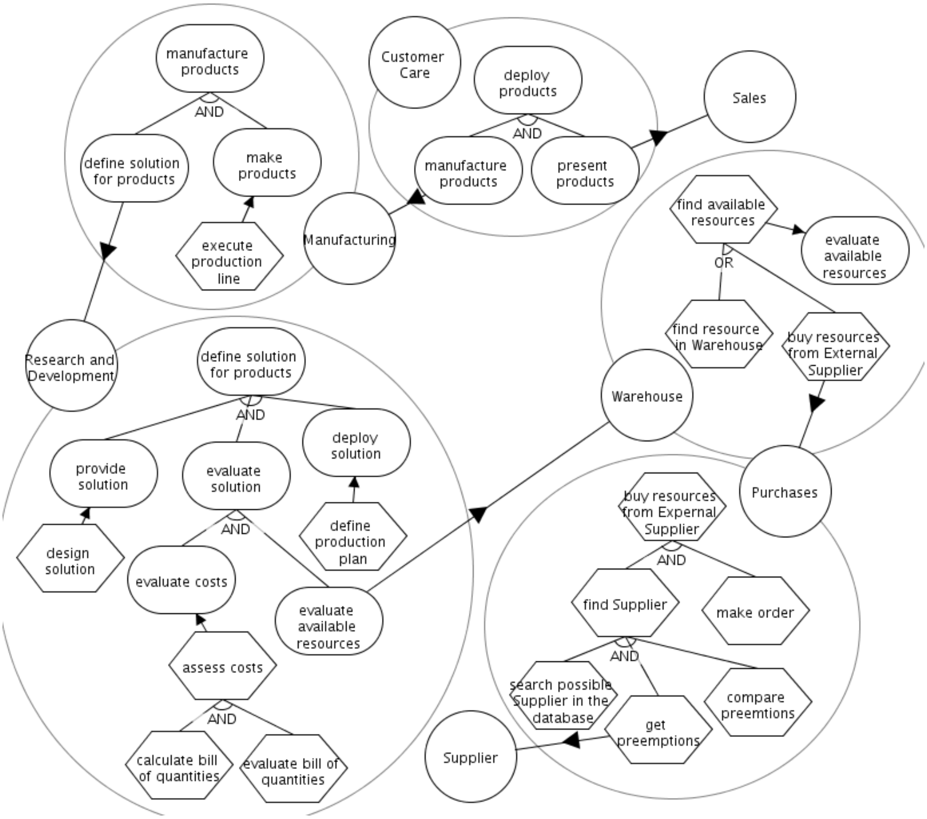


Fig. 1. Product development process

analysis consists of refining goals and eliciting new social relationships among actors. Goal analysis is conducted from the perspective of single actors using means-end analysis and AND/OR decomposition. *Means-end analysis* aims at identifying tasks to be executed in order to achieve a goal. Means-end relations are graphically represented as arrows without any label on them. *AND/OR decomposition* combines AND and OR refinements of a root goal or a root task into subparts. In essence, AND-decomposition is used to define the high-level process for achieving a goal or a task, whereas OR-decomposition defines alternatives for achieving a goal or executing a task. Fig. 1 presents the Tropos diagram representing an excerpt of the product development process studied in the course of the TOCAI project.

Example 1. Different divisions of a company have to cooperate in order to produce a specific product. The Customer Care division is responsible for deploying products to customers, which it refines into subgoals manufacture product, for which Customer Care depends on the Manufacturing division, and present product, for which it depends on the Sales division. In turn, Manufacturing decomposes the appointed goal into subgoals define solution for product, for which it depends on the Research & Development (R&D) division, and make product that it achieves through task execute production line. To achieve goal define solution for product, R&D has to achieve goals provide solution, which it achieves by executing task design solution, evaluate solution, and deploy solution, which it achieves through task define production plan. The evaluation of the solution is performed in terms of costs and available resources. In order to evaluate costs, R&D executes task assess costs, which consists of calculate bill of quantities and evaluate bill of quantities. Moreover, this division depends on the Warehouse for the goal evaluate available resources. The Warehouse either queries the databases to find available resources or asks the Purchases division to buy resources from external Supplier. The Purchases division searches in company's databases for possible Suppliers and selects the one who provides the best offer.

3 Towards Declarative Process-Oriented Annotations

How business processes can be obtained from requirements analysis is an urgent issue for the development of a system. Unfortunately, Tropos is not able to cope with this issue mainly due to the lack of temporal constructs. In this section we discuss how Tropos can be extended in order to deal with high-level process-oriented aspects. The proposed extensions intend to support designers in defining durations, absolute time and domain-based constraints for goals and tasks, as well as declaratively specifying relations between them. These extensions are based on the DecSerFlow [34] and ConDec [33] graphical languages for the declarative representation of service flows and flexible business processes. The enhanced Tropos is called $\mathcal{B}$-Tropos.

3.1 Some Definitions

For the sake of clarity, we now give some informal definitions, which will be used to describe the Tropos extensions introduced in this section.

Definition 1 (Time interval). *A time interval is a definite length of time marked off by two (non negative) instants (T_{min} and T_{max}), which can be considered both in an exclusive or inclusive manner. As usually, we use parentheses ($(\ldots)$) to indicate exclusion and square brackets ($[\ldots]$) to indicate inclusion.*

Definition 2 (Relative time interval). *A time interval is relative if initial instant and final instant are defined in function of another instant. Given a time interval TI marked off by T_{min} and T_{max} and a time instant T, two relative time intervals could be defined w.r.t. T*

- *TI^{+T} to denote the time interval marked off by $T + T_{min}$ and $T + T_{max}$;*
- *TI^{-T} to denote the time interval marked off by $T - T_{max}$ and $T - T_{min}$.*

For example, $[10, 15)^{+T_1} \equiv [T_1 + 10, T_1 + 15)$ and $(0, 7]^{-T_2} \equiv (T_2 - 7, T_2]$.

Definition 3 (Absolute time constraint). *An absolute time constraint is a constraint of the form* T $\boldsymbol{OP}$ *Date, where* T *is a time variable, Date is a date and* $\boldsymbol{OP} \in \{at, after, after\_or\_at, before, before\_or\_at\}$ *(with their intuitive meaning).*

Definition 4 (Domain-based constraint). *A domain-based constraint formalizes specific application domain requirements and is specified using CLP constraints [21] (e.g., >, <, =, etc.) or Prolog predicates.*

Definition 5 (Condition). *A condition is a conjunction of domain-based and absolute time constraints.*

For example, condition $T\ before\_or\_at\ 11.26.2007 \wedge workingDay(T)$ states that T has 26 November 2007 as deadline and that it must be a working day.

3.2 Tasks/Goals Extension

In order to support the modeling and analysis of process-oriented aspects of systems, we have annotated goals and tasks with temporal information such as *start* and *completion* times (the notation is shown in Fig. 2). Each task/goal can also be described in terms of its allowed *duration* ($[Dmin, Dmax]$ in Fig. 2). This allows one to constrain, for instance, the completion time to the start time, i.e., $completion\ time \in [Dmin, Dmax]^{+start\ time}$. Additionally, absolute temporal constraints can be used to define start and completion times of goals and tasks.

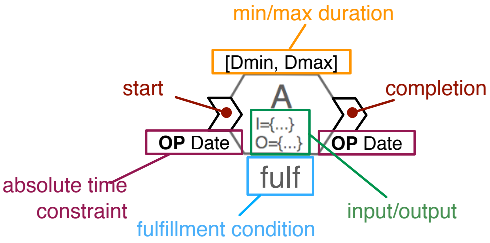


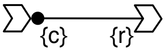
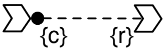
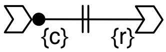
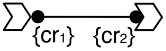
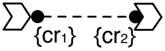
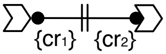
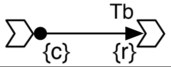
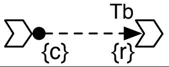
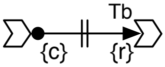
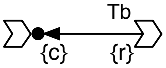
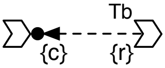
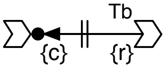
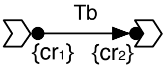
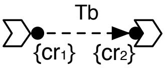
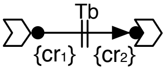
Fig. 2. Extended notation for tasks and goals

A goal/task can also be described in terms of the resources needed and produced by the goal/task itself. We represent the resources needed by a goal/task through attribute *input* and the resources produced by a goal/task through attribute *output*. Finally, tasks can be annotated with a *fulfillment condition*, which defines when they are successfully executed.

3.3 Process-Oriented Constraints

To refine a requirements model into a high-level and declarative process-oriented view, we have introduced different connections between goals and tasks, namely *relation*, *weak relation*, and *negation* (see Table 1). These connections allow

Table 1. Tropos extensions to capture process-oriented constraints (grouped negation connections share the same intended meaning, as described in [34])

	relation	weak relation	negation
responded presence			
co-existence			
response			
precedence			
succession			

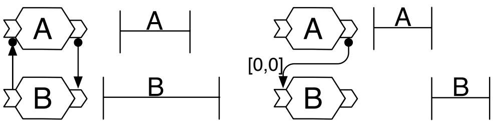
designers to specify partial orderings between tasks under both temporal and domain-based constraints. To make the framework more flexible, connections are not directly linked to tasks but to their start and completion times. This solution, for instance, enables the representation of interleaving concurrency. A small circle is used to denote the connection source, which determines when the triggering condition is satisfied (co-existence and succession connections associate the circle to both end-points, since they are bi-directional).

Relation and negation connections are based on DecSerFlow [34] and ConDec [33] template formulas, extended with constraints on execution times (e.g., deadlines) as well as domain-based and absolute time constraints. Conditions can be specified on both start and completion times and are delimited by curly braces ($\{c\}$, $\{r\}$ and $\{cr_i\}$ in Table 1); the source condition is a triggering condition, whereas the target imposes restrictions on time and/or data.

The intended meaning of a *responded presence* relation is: if the source happens such that c is satisfied, then the target should happen and satisfy r. The *co-existence* relation applies the responded presence relation in both directions, by imposing that the two involved tasks, when satisfying cr_1 and cr_2, should co-exist (namely either none or both are executed).

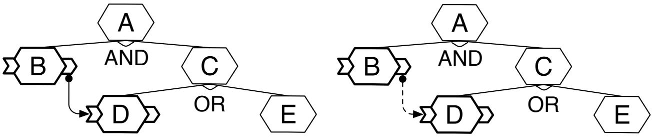
Other relation connections extend the responded presence relation by specifying a temporal ordering between source and target events; optionally, a relative time interval (denoted with T_b in Table 1) could be attached to these connections, bounding when the target is expected to happen with respect to the time at which the source happened.[2] In particular, the *response* relation constrains the target to happen *after* the source. If T_b is specified, the minimum and maximum times are respectively treated as a delay and a deadline, that is, the target should occur between the minimum and the maximum time after the source ($target\ time \in T_b^{+source\ time}$). The *precedence* relation is opposite to response relation, in the sense that it constrains the target to happen *before* the

[2] If T_b is not specified, the default interval is $(0, \infty)$.



(a) *A* during *B* (b) *A* meets *B*

Fig. 3. Representation of two simple Allen's intervals in $\mathcal{B}$-Tropos



(a) Response relation (b) Weak response relation

Fig. 4. Integrating process-oriented and goal-directed dimensions in $\mathcal{B}$-Tropos

source. A *succession* relation is used to mutually specify that two tasks are the response and precedence of each other. By mixing different relation connections, we can express complex temporal dependencies and orderings, such as Allen's intervals [5] (Fig. 3). For example, Allen's *duration* relation is formalized by imposing that A's start should happen after B's start and A's completion should happen before B's completion (Fig. 3(a)), whereas *meets* relation is formalized by imposing that A's completion should be equal to B's start (Fig. 3(b)).

As in DecSerFlow and ConDec, we assume an open approach. Therefore, we have to explicitly specify not only what is expected, but also what is forbidden. These "negative" dependencies are represented by negation connections, the counter-part of relation connections. For example, negation co-existence between two tasks states that when one task is executed, the other task shall never be executed, neither before nor after the source.

Summarizing, through relation and negation connections designers can add a horizontal declarative and high level process-oriented dimension to the vertical goal-directed decomposition of goals and tasks. It is worth noting that, in presence of OR decompositions, adding connections may affect the semantics of the requirements model. The decomposition of task A in Fig. 4(a) shows that its subtask C can be satisfied by satisfying D or E. On the contrary, the response relation between B's completion and D's start makes D mandatory (B has to be performed because of the AND-decomposition, hence D is expected to be performed after B). This kind of interaction is not always desirable. Therefore, we have introduced *weak relation* connections with the intent of relaxing relation connections. Their intended meaning is: whenever both the source and the target happen and the trigger condition is satisfied, the target must satisfy the restriction condition. The main difference between relations and weak relations is that in weak relations the execution is constrained a posteriori, after both source and target have happened. Differently from Fig. 4(a), in Fig. 4(b) the response constraint between B and D should be satisfied only if D is executed.

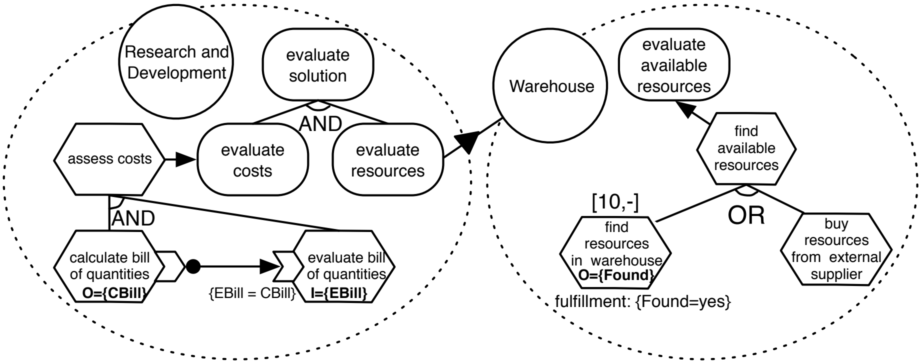


Fig. 5. Process-oriented extensions applied on a fragment of Fig. 1

Finally, $\mathcal{B}$-Tropos permits to constrain non-leaf tasks, leading to the possibility of expressing some process-oriented patterns [35]. For instance, a relation connection whose source is the completion of a task, which is AND-decomposed into two subtasks, triggers when both subtasks have been executed. Therefore, the connection resembles the concept of a synchronizing merge on the leaf tasks.

To show how process-oriented constraints could be added to a Tropos model, we extend a fragment of the diagram represented in Fig. 1; the result is shown in Fig. 5. The first extension concerns the decomposition of task assess costs: the bill of quantities can be evaluated only after having been calculated. Such a constraint could be modeled in $\mathcal{B}$-Tropos by (1) indicating that the calculation produces a bill of quantities, whereas the evaluation takes a bill as an input, and (2) attaching a response relation connection between the completion of task calculate bill of quantities and the start of task evaluate bill of quantities. The second extension has the purpose of better detailing task find resources in Warehouse, namely representing that (1) task duration is at least 10 time units, (2) the task produces as an output a datum (called *Found*), which describes whether or not resources have been found in the Warehouse, and (3) the task is considered fulfilled only if resources have been actually found, that is, *Found* is equal to *yes*. Finally, one can notice the absence of constraints between goals evaluate costs and evaluate resources. Such an absence enables the two sets of activities aimed at achieving those goals to be executed concurrently.

4 $\mathcal{S}$CIFF

$\mathcal{S}$CIFF [4] is a formal framework based on abductive logic programming [22], developed in the context of the SOCS project[3] for specifying and verifying interaction protocols in an open multi-agent setting. $\mathcal{S}$CIFF introduces the concept of event as an atomic observable and relevant occurrence triggered at execution

[3] SOcieties of heterogeneous ComputeeS,EU-IST-2001-32530 (home page`http://lia.deis.unibo.it/research/SOCS/`).

time. The designer has the possibility to decide what has to be considered as an event; this generality allows him to decide how to model the target domain at the desired abstraction level, and to exploit $\mathcal{S}$CIFF for representing any evolving process where activities are performed and information is exchanged.

We distinguish between the description of an *event*, and the fact that an event has happened. Happened events are represented as atoms $\mathbf{H}(Ev, T)$, where Ev is a *term* and T is an integer, representing the discrete time point at which the event happened. The set of all the events happened during a protocol execution constitutes its log (or execution trace). Furthermore, the $\mathcal{S}$CIFF language supports the concept of *expectation* as first-class object, pushing the user to think of an evolving process in terms of reactive rules of the form *"if A happened, then B is expected to happen"*. Expectations about events come in form $\mathbf{E}(Ev, T)$ where Ev and T are variables, eventually grounded to a particular term/value.

The binding between happened events and expectations is given by means of *Social Integrity Constraints* ($\mathcal{IC}$s). Such constraints are forward rules of the form $Body \rightarrow Head$, where *Body* can contain literals and (conjunctions of happened and expected) events and *Head* can contain (disjunctions of) conjunctions of expectations. CLP constraints and Prolog predicates can be used to impose relations or restrictions on any of the variables, for instance, on time (e.g., by expressing orderings or deadlines). Intuitively, $\mathcal{IC}$ allows the designer to define how an interaction should evolve, given some previous situation represented in terms of happened events; the static knowledge of the target domain is instead formalized inside the $\mathcal{S}$CIFF Knowledge Base. Here we find pieces of knowledge on the interaction model as well as the global organizational goal and/or objectives of single participants. Indeed, $\mathcal{S}$CIFF considers interaction as goal-directed, i.e., it envisages environments in which each actor as well as the overall organization could have some objective only achievable through interaction; by adopting such a vision, the same interaction protocol could be seamlessly exploited for achieving different strategic goals. This knowledge is expressed in the form of clauses (i.e., a logic program); a clause body may contain expectations about the behavior of participants, defined literals, and constraints, while their heads are atoms. As advocated in [17], this vision reconciles in a unique framework forward reactive reasoning with backward, goal-oriented deliberative reasoning.

In $\mathcal{S}$CIFF an interaction model is interpreted in terms of an Abductive Logic Program (ALP) [22]. In general, an ALP is a triple $\langle P, A, IC \rangle$, where P is a logic program, A is a set of predicates named *abducibles*, and IC is a set of Integrity Constraints. Roughly speaking, the role of P is to define predicates, the role of A is to fill in the parts of P that are unknown, and the role of IC is to control the way elements of A are hypothesized, or "abducted". Reasoning in abductive logic programming is usually goal-directed, and accounts for finding a set of abducted hypotheses Δ built from predicates in A such that $P \cup \Delta \models G$ (being G a goal) and $P \cup \Delta \models IC$. The idea underlying $\mathcal{S}$CIFF is to adopt abduction to dynamically *generate* the expectations and to perform the *conformance checking* between expectations and happened events (to ensure that they are following the interaction model). Expectations are defined as abducibles: the framework makes hypotheses about how participants should behave. Conformance is verified by trying to confirm the hypothesized expectations: a concrete running interaction is evaluated as conformant if it fulfills the specification. Operationally, expectations are generated and verified by the $\mathcal{S}$CIFF proof procedure,[4] a transition system which has been proved sound and complete with respect to the declarative semantics [4]. The proof procedure is embedded within SOCS-SI [3], a JAVA-based tool capable of accepting different event sources (or previously collected execution traces) and checking if the actual behavior is conformant with respect to a given $\mathcal{S}$CIFF specification.

5 Mapping $\mathcal{B}$-Tropos Concepts to the $\mathcal{S}$CIFF Framework

In this section we present the mapping of $\mathcal{B}$-Tropos concepts into $\mathcal{S}$CIFF specifications, briefly describing how the obtained formalization is used to implement the skeleton of logic-based agents. The idea behind the mapping is to define a formal statement in $\mathcal{S}$CIFF for each $\mathcal{B}$-Tropos graphical element. This allows for the automatic generation of $\mathcal{S}$CIFF specifications from $\mathcal{B}$-Tropos models.

Table 2 summarizes the formalization of the goal-oriented part of $\mathcal{B}$-Tropos in $\mathcal{S}$CIFF. This part represents the static knowledge of the application domain, so it is modeled inside the $\mathcal{S}$CIFF knowledge base. Two fundamental concepts are *goal achievement* and *task execution*. These concepts are modeled in $\mathcal{S}$CIFF using the 6-ary predicates *achieve* and *execute*. Intuitively, $achieve(x, g, t_i, t_f, i, o)$ is true if actor x achieves goal g where t_i is the start time and t_f is the completion time. $execute(x, a, t_i, t_f, i, o)$ holds if actor x executes task a where t_i and t_f are start and completion time, respectively. Parameters i and o represent the resources respectively needed and produced by the execution of the task or achievement of a goal. Start and completion times should satisfy both duration and absolute time constraints (ac in Table 2) eventually associated to a goal/task.

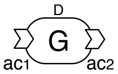
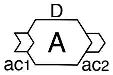
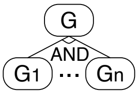
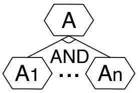
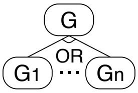
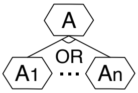
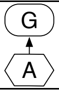
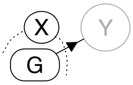
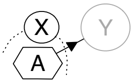
The execution of tasks is also determined by the satisfaction of fulfillment conditions and the generation of task start and completion events. These events are represented using literals of the form $event(ev, x, a, r)$ where $ev \in \{start, end\}$, a is the task that has generated the event, x is the actor who has executed the task, and r is a list of resources. In particular, resources associated with start events represent the input of the task, whereas resources associated with completion events refer to the output.

In some cases the designer may prefer to keep the model at an abstract level, so goals can be neither refined nor associated to tasks. Abduction allows us to face such a lack of information by reasoning on goal achievement in a hypothetical way. In particular, we have introduced a new abducible called **achieved** to hypothesize that the actor has actually reached the goal.

Tropos relations are then formalized in $\mathcal{S}$CIFF as rules on the basis of the following concepts:

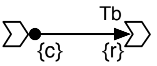
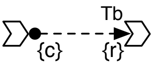
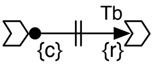
[4] Available at http://lia.deis.unibo.it/research/sciff/

Table 2. Mapping of the goal-oriented proactive part of $\mathcal{B}$-Tropos onto $\mathcal{S}$CIFF

Leaf goal		$achieve(X, G, T_i, T_f, I, O) \leftarrow$ $\mathbf{achieved}(X, G, T_i, T_f, I, O),$ $T_f \in [D_{min}, D_{max}]^{+T_i}, ac_1, ac_2.$
Leaf task		$execute(X, A, T_i, T_f, I, O) \leftarrow$ $\mathbf{E}(event(start, X, A, I), T_i),$ $\mathbf{E}(event(end, X, A, O), T_f),$ $T_f \in [D_{min}, D_{max}]^{+T_i}, ac_1, ac_2,$ $fulfillment\_condition.$
AND decomposition		$achieve(X, G, T_i, T_f, I, O) \leftarrow$ $achieve(X, G_1, T_{i1}, T_{f1}, I_1, O_1), \ldots,$ $achieve(X, G_n, T_{in}, T_{fn}, I_n, O_n),$ $T_i = \min\{T_{i1}, \ldots, T_{in}\}, T_f = \max\{T_{f1}, \ldots, T_{fn}\},$ $I = I_1 \cup \ldots \cup I_n, O = O_1 \cup \ldots \cup O_n.$
		$execute(X, A, T_i, T_f, I, O) \leftarrow$ $execute(X, A_1, T_{i1}, T_{f1}, I_1, O_1), \ldots,$ $execute(X, A_n, T_{in}, T_{fn}, I_n, O_n),$ $T_i = \min\{T_{i1}, \ldots, T_{in}\}, T_f = \max\{T_{f1}, \ldots, T_{fn}\},$ $I = I_1 \cup \ldots \cup I_n, O = O_1 \cup \ldots \cup O_n.$
OR decomposition		$achieve(X, G, T_i, T_f, I, O) \leftarrow achieve(X, G_1, T_i, T_f, I, O).$ $\ldots$ $achieve(X, G, T_i, T_f, I, O) \leftarrow achieve(X, G_n, T_i, T_f, I, O).$
		$execute(X, A, T_i, T_f, I, O) \leftarrow execute(X, A_1, T_i, T_f, I, O).$ $\ldots$ $execute(X, A, T_i, T_f, I, O) \leftarrow execute(X, A_n, T_i, T_f, I, O).$
Means-end		$achieve(X, G, T_i, T_f, I, O) \leftarrow execute(X, A, T_i, T_f, I, O).$
Goal dependency		$achieve(X, G, T_i, T_f, I, O) \leftarrow$ $\mathbf{E}(delegate(X, Y, G, T_f), T_d),$ $T_d > T_i, T_d < T_f.$
Task dependency		$execute(X, A, T_i, T_f, I, O) \leftarrow$ $\mathbf{E}(delegate(X, Y, A, T_f), T_d),$ $T_d > T_i, T_d < T_f.$

- AND/OR-decompositions and means-end are trivially translated to $\mathcal{S}$CIFF.
- In goal (task) dependencies, it is expected that the depender appoints the dependee to achieve a goal (execute a task) before a certain time instant. To this end, we have introduced event $delegate(x, y, g, t)$ to indicate that actor x delegates the achievement of goal g to actor y and y have to achieve g by time t. A delegation is observable and so it is kept trace of in the execution trace.

Table 3. Mapping of $\mathcal{B}$-Tropos response connections onto $\mathcal{S}$CIFF

Response		$\mathbf{hap}(event(Ev, X_1, A_1, R_1), T_1) \wedge c$ $\rightarrow \mathbf{exp}(event(Ev, X_2, A_2, R_2), T_2) \wedge r \wedge T_2 \in T_b^{+T_1}.$
Weak Response		$\mathbf{hap}(event(Ev, X_1, A_1, R_1), T_1) \wedge c$ $\wedge \mathbf{hap}(event(Ev, X_2, A_2, R_2), T_2) \rightarrow r \wedge T_2 \in T_b^{+T_1}.$
Negation Response		$\mathbf{hap}(event(Ev, X_1, A_1, R_1), T_1) \wedge c$ $\wedge \mathbf{hap}(event(Ev, X_2, A_2, R_2), T_2) \wedge r \wedge T_2 \in T_b^{+T_1} \rightarrow \perp .$

The reactive part of $\mathcal{B}$-Tropos encompasses both the reaction to a dependency and process-oriented constraints. As already pointed out, process-oriented constraints are inspired by DecSerFlow/ConDec template formulas, for which a preliminary mapping to $\mathcal{S}$CIFF has been already established [11]. Connections are translated using $\mathcal{IC}$s. For the sake of space, we refer to [4] for a detailed description on how $\mathcal{S}$CIFF handles constraints. Here we present some examples of how process-oriented constraints are formalized (Table 3). Such formulas specify the informal description given in Section 3. Response connection constraint states that if the source, $event(Ev, X_1, A_1, R_1)$, happens and the trigger condition, c, is satisfied, then the target, $event(Ev, X_2, A_2, R_2)$, is expected to happen and the restriction condition imposed on the target, r, must be satisfied. In addition, the target is expected to occur within $T_b^{+T_1}$. Weak response constraints are verified a posteriori. In particular, when the connected events happen and the triggering condition is satisfied, the restriction imposed by the target must be satisfied. Similarly to response connections, the constraint is verified if the target event occurs within $T_b^{+T_1}$. Negative response constraints spot an inconsistency when the connected events happen and all conditions are satisfied.

We remark that the framework allows one to constrain non-leaf tasks and goals, but only start and completion events of leaf tasks are considered as observable events. To address this issue, we have introduced intensional predicates **hap** and **exp** to represent the happening and expectation of (possibly) composite events. For instance, a leaf task starts (or is completed) only if there is evidence for it (i.e., the corresponding event happened). Accordingly, for a leaf-task A:

$$\mathbf{hap}(event(Ev, X, A, R), T) \leftarrow \mathbf{H}(event(Ev, X, A, R), T).$$
$$\mathbf{exp}(event(Ev, X, A, R), T) \leftarrow \mathbf{E}(event(Ev, X, A, R), T).$$

Composite events recursively follow the goal analysis approach:

- the start/completion of an OR-decomposed task happen (resp. is expected to happen) when one of its (sub)tasks start/completion happens (resp. is expected to happen);
- the start of an AND-decomposed task happens (resp. is expected to happen) when its first (sub)task starts (resp. expected to start);
- the completion of an AND-decomposed task happens (resp. is expected to happen) when its last (sub)task is completed (expected to be completed).

To model the reaction to a dependency, we assume that when a dependee Y receives from a depender X a request for achieving a goal G, Y reacts by undertaking the commitment of achieving G:[5]

$$\mathbf{H}(delegate(X,Y,G,T_f),T_d) \rightarrow achieve(Y,G,T_i,T_f,I,O) \wedge T_i > T_d.$$

The provided formalization can be used to directly implement the skeleton of logic-based agents, as for example the ones described in [1]. Such agents follow the Kowalsky-Sadri cycle for intelligent agents, by realizing the *think* phase with the $\mathcal{S}$CIFF proof-procedure and the *observe* and *act* phases in JADE. The proof-procedure embedded in $\mathcal{S}$CIFF-agents is equipped with the possibility to transform expectations about the agent into happened events, and with a selection rule for choosing a behavior when several choices are available. In particular, each actor represented in a $\mathcal{B}$-Tropos model can be mapped into a $\mathcal{S}$CIFF-agent whose deliberative pro-active part (formalized in the agent's knowledge base) is driven by the goal/task decomposition of its root goal, and whose reactive behavior (formalized as a set of $\mathcal{IC}$s) is determined by the delegation mechanism and the process-oriented constraints. The agent that wants to achieve the global goal (e.g., Customer Care in Fig. 1) starts by decomposing it, whereas other agents wait until an incoming request is observed. In this case, the dependency reactive rule of the agent is triggered, and the agent attempts to achieve the assigned goal. This goal may be either decomposed or delegated to other agents until expectations proving its achievement are generated. Such expectations thus are transformed to happened events, that is, actions performed by the agent.

Figure 6 presents the $\mathcal{S}$CIFF formalization corresponding to the $\mathcal{B}$-Tropos diagram of Fig. 5. Here Research & Development and Warehouse are respectively represented as $r\&d$ and wh, and symbol $=$ is used to denote unification. In that figure one can see how the formalized $\mathcal{S}$CIFF specification is assigned to the Warehouse and R&D units. To have an intuition about how the two agents act and interact, let us consider the case in which the R&D unit intends to achieve the goal assigned by the Manufacturing division. The unit decomposes goal evaluate solution in its subparts until a set of expectations, which lead to the achievement of the goal, is determined. Below we list a possible set of expectations:

$$\begin{aligned}
&\mathbf{E}(event(start, r\&d, calc\_bill, []), T_{scb}), \ldots,\\
&\mathbf{E}(event(end, r\&d, calc\_bill, [Bill]), T_{ccb}), T_{ccb} > T_{scb},\\
&\mathbf{E}(event(start, r\&d, eval\_bill, [Bill]), T_{seb}), T_{seb} > T_{ccb},\\
&\mathbf{E}(event(end, r\&d, eval\_bill, []), T_{ceb}), T_{ceb} > T_{seb},\\
&\mathbf{E}(delegate(r\&d, wh, eval\_resources, T_{cer}), T_{ser}).
\end{aligned}$$

This set of expectations can be read as an execution plan, consisting of two concurrent parts: (1) a sequence of events related to start/completion of leaf tasks, ordered by the response relation which constrains the bill calculation and evaluation; (2) the delegation of resources evaluation, which should be communicated

[5] For the sake of brevity we do not present here the reaction rule for task dependency that has the same intuition as the one for goal dependency.

$$\begin{array}{rl}
KB_{r\&d}: achieve(r\&d, eval\_solution, T_i, T_f, I, O) \leftarrow & achieve(r\&d, eval\_costs, T_{i1}, T_{f1}, I_1, O_1), \\
& achieve(r\&d, eval\_resources, T_{i2}, T_{f2}, I_2, O_2), \\
& \min(T_i, [T_{i1}, T_{i2}]), \max(T_f, [T_{f1}, T_{f2}]), \\
& I = I_1 \cup I_2, O = O_1 \cup O_2. \\
achieve(r\&d, eval\_costs, T_i, T_f, I, O) \leftarrow & execute(r\&d, assess\_costs, T_i, T_f, I, O). \\
execute(r\&d, assess\_costs, T_i, T_f, I, O) \leftarrow & execute(r\&d, calc\_bill, T_{i1}, T_{f1}, I_1, O_1), \\
& execute(r\&d, eval\_bill, T_{i2}, T_{f2}, I_2, O_2), \\
& \min(T_i, [T_{i1}, T_{i2}]), \max(T_f, [T_{f1}, T_{f2}]), \\
& I = I_1 \cup I_2, O = O_1 \cup O_2. \\
execute(r\&d, calc\_bill, T_i, T_f, [], [CBill]) \leftarrow & \mathbf{E}(event(start, r\&d, calc\_bill, []), T_i), \\
& \mathbf{E}(event(end, r\&d, calc\_bill, [CBill]), T_f), T_f > T_i. \\
execute(r\&d, eval\_bill, T_i, T_f, [EBill], []) \leftarrow & \mathbf{E}(event(start, r\&d, eval\_bill, [EBill]), T_i), \\
& \mathbf{E}(event(end, r\&d, eval\_bill, []), T_f), T_f > T_i. \\
achieve(r\&d, eval\_resources, T_i, T_f, I, O) \leftarrow & \mathbf{E}(delegate(r\&d, wh, eval\_resources, T_f), T_d), \\
& achieve(wh, eval\_resources, T_d, T_f, I, O), \\
& T_d > T_i, T_d < T_f. \\
KB_{wh}: achieve(wh, eval\_resources, T_i, T_f, I, O) \leftarrow & execute(wh, find\_resources, T_i, T_f, I, O). \\
execute(wh, find\_resources, T_i, T_f, I, O) \leftarrow & execute(wh, find\_in\_wh, T_i, T_f, I, O). \\
execute(wh, find\_resources, T_i, T_f, I, O) \leftarrow & execute(wh, buy, T_i, T_f, I, O). \\
execute(wh, find\_resources, T_i, T_f, [], [Found]) \leftarrow & \mathbf{E}(event(start, wh, find\_in\_wh, []), T_i), \\
& \mathbf{E}(event(end, wh, find\_in\_wh, [Found]), T_f), \\
& T_f \geq T_i + 10, Found = yes. \\
execute(wh, buy, T_i, T_f, [], []) \leftarrow & \mathbf{E}(event(start, wh, buy, []), T_i), \\
& \mathbf{E}(event(end, wh, buy, []), T_f), T_f > T_i.
\end{array}$$

$$\begin{array}{rl}
\mathcal{IC}s_{r\&d}: \mathbf{hap}(event(end, r\&d, calc\_bill, [CBill]), T_1) \rightarrow & \mathbf{exp}(event(start, r\&d, eval\_bill, [EBill]), T_2) \\
& \wedge T_2 > T_1 \wedge EBill = CBill. \\
\mathcal{IC}s_{wh}: \mathbf{H}(delegate(r\&d, wh, eval\_resources, T_f), T_d) \rightarrow & achieve(wh, eval\_resources, T_i, T_f, I, O) \\
& \wedge T_i > T_d.
\end{array}$$

Fig. 6. Formalization of the $\mathcal{B}$-Tropos model fragment shown in Fig. 5

to the Warehouse. In particular, when the expectation about the delegation is transformed to a happened event by the R&D agent, the Warehouse agent is committed to achieve the delegated goal inside the time interval (T_{ser}, T_{cer}). It is worth noting that the framework can identify inconsistencies in temporal and/or data requirements specification by means of unsatisfiable constraints. This is, for instance, the case in which the R&D unit requires an evaluation of the availability of resources, e.g., in 5 time units, whereas the Warehouse needs at least 10 time units to verify the presence of resources. In these situations, the designer needs either to relax constrains (e.g., extending the time) or to adopt new solutions for increasing the performance of the system (e.g., providing the Warehouse with a more efficient search application).

Besides the implementation of logic-based agents, $\mathcal{S}$CIFF can also be used to perform different kinds of verification, namely performance verification and conformance verification. Performance verification aims at proving that stakeholders can achieve their strategic goals in a given time. Such verification can also be used to evaluate different design alternatives in terms of system performance. For example, one could ask $\mathcal{S}$CIFF to verify whether an execution exists such that the top goal of one of the stakeholders is achieved within a given deadline. $\mathcal{S}$CIFF will then try to generate such an execution, by means of an intensional (i.e., partially specified) execution trace; generally speaking, this is achieved by transforming expectations into happened events. Conformance verification [4] is related to the auditing measures that can be adopted for monitoring the activities performed by actors within the system. The idea underlying conformance verification is to analyze system logs and compare them with the design of the system, to verify whether the actual behavior of a system effectively complies with model expectations. This allows system administrators to understand whether or not stakeholders have achieved their goals and, if it is not the case, to predict future actions.

6 Related Work

While the literature on single aspects of the framework is huge (many references can be found to the papers describing Tropos, $\mathcal{S}$CIFF, and DecSerFlow/CondDec), not much work has been done at the intersection of the corresponding domains. Several formal frameworks have been developed to support the Tropos methodology. For instance, Giorgini et al. [19] proposed a formal framework based on logic programming for the analysis of security requirements. However, the framework does not take into account temporal aspects of the system. In [9] a planning approach has been proposed to analyze and evaluate design alternatives. Though this framework explores the space of alternatives and determines a (sub-)optimal plan, that is, a sequence of actions, to achieve the goals of stakeholders, it is limited in defining temporal constraints among tasks. Fuxman et al. [18] proposed Formal Tropos that extends Tropos with annotations that characterize the temporal evolution of the system, describing, for instance, how the network of relationships evolves over time. Formal Tropos provides a temporal logic-based specification language for representing Tropos concepts together with temporal constructs, which are verified using a model-checking technique such as the one implemented in NuSMV. This framework has been used to verify the consistency of requirements models [18] as well as business processes against business requirements and strategic goal model [23]. However, Formal Tropos does not support abduction, and thus, it is not able to generate expectations and perform conformance checking between expectations and happened events. Finally, we mention the work by Cares et al [10], who proposed to implement software agents in Prolog starting from Tropos models. In particular, they proposed to specify the programming activation time through four implementation attributes, namely at begin, at end, at call, and always. The difference with our proposal lies in the generation of implementation besides the employed temporal constructs. Actually, they do not provide an encoding of Tropos models into Prolog so that the implementation is manual.

The last years have seen the need for bridging the gap between requirements engineering and business process design by providing support for developing business processes on top of requirements models and verifying whether a business process actually meets its business goals. For instance, Lapochnian et al. [25] proposed a systematic requirements-driven approach for business process design and configuration management, which adopts goal models to capture alternative process configurations. Differently from our work, they do not consider the relationships between agents so that framework is inadequate to describe business processes spanning across multi-agent systems. Frankova et al. [16] have used the SI* modeling language [27], an extension of Tropos addressing security and privacy issues, as a basis for the definition of Secure BPEL, a specification language that extends WS-BPEL [6] for modeling secure business processes. The objective of this framework is to assist business process analysts in deriving the skeleton of secure business processes from early requirements analysis. Finally, López et al. [26] presented a reasoning method for verifying the consistency between SI* models and BPMN specifications [7]. In particular, the authors have investigated the connection between business processes and requirements models, introducing the notion of goal equivalence based on trace semantics.

Several works also attempt to define a formal semantics underlying graphical business process models and to design agent systems. In the business process domain, Wong et al. [37] provided a formal semantics for a subset of BPMN in terms of the process algebra CSP [30], whereas Dijkman et al. [15] used Petri Nets [28]. Their objective is to formally analyze and compare business process models. We differ from these proposals since the objective of our work is to provide a requirements-driven framework for business process and agent system design. The use of computational logic for the flexible specification and rigorous verification of agent interaction is adopted by many proposals. While other works (e.g., [36]) use temporal logic to model the temporal dimension of interaction, $\mathcal{S}$CIFF exploits a constraint solver and adopts an explicit representation of time.

Event Calculus [24] was introduced by Kowalsky and Sergot as a logic programming formalism for representing events and their effects. This formalism explicitly reasons upon properties (fluents) holding during time intervals. Differently from Event Calculus, our framework treats time like other variables, in association with domains, which makes it possible to express constraints (e.g., deadlines) and to exploit an underlying constraint solver. Among the works based on Event Calculus, we cite the work by Shanahan [32], who proposed the abductive event calculus that includes the concept of expectation, and the work by Cicekli et al. [12], who formalized workflows using Event Calculus. In Shanahan's work events and expectations are of the same nature and both are abduced, while our expectations should match the actual events. This is due to the different underlying assumptions and, consequently, the different focus: while we assume that the history is known, Shanahan proposes to abduce events. Similarly to [15,37], Cicekli et al. focus on the execution of business processes, whereas the reconciliation between a business process and the business goals that have motivated the process definition are completely ignored.

Finally we mention that a mapping of DecSerFlow into Linear Temporal Logic (LTL) [29] has been proposed in [34]. It can be used to verify or enforce conformance of service flows and also to directly enact their execution. The advantages of using

SCIFF instead of LTL is that SCIFF can handle time and data in an explicit and quantitative way, exploiting CLP to define temporal and data-related constraints.

7 Conclusions

In this work we have proposed to integrate a number of techniques for information systems engineering, with the aim to reconcile requirements elicitation with specification, prototyping and analysis, inside a single unified framework. We have presented $\mathcal{B}$-Tropos, an extension of Tropos with declarative process-oriented constraints, and its mapping into the $\mathcal{S}$CIFF language. We have mainly focused on the modeling and mapping of aspects related to declarative business processes using connections inspired by DecSerFlow and ConDec languages. Augmenting a Tropos model with such constraints has the effect that both the proactive and the reactive, process-oriented agent behavior could be captured within the same diagram.

The mapping of $\mathcal{B}$-Tropos onto $\mathcal{S}$CIFF makes it possible to directly implement logic-based agents starting from the enhanced Tropos model, as well as to perform different kinds of verification, namely to check if the model satisfies a given property and to monitor if the execution trace of a real system is actually compliant with the model.

The work presented here is a first step towards the integration of a business process in the requirements model. We are currently running experiments on prototyping as well as on property and conformance verification. Some results are presented in [13], where $\mathcal{B}$-Tropos models are also used to generate possible executions traces, and to animate agents in the context of the CLIMA Contest Food Collection problem [14], in line with the aforementioned work by Cares and colleagues [10]. We are also investigating in depth the formal properties of our proposed mapping, and are trying to understand how to better exploit the underlying $\mathcal{S}$CIFF constraint solver by introducing more complex scheduling and resource constraints so as to capture more detailed business requirements and agent interactions. As a future activity, we plan to investigate the generation of executable business process specifications (such as WS-BPEL) from $\mathcal{B}$-Tropos models. Another direction under investigation concerns business process compliance. In particular, we are interested in the problem of the interplay between business and control objectives during business process design [31]. Finally, we intend to conduct empirical studies on large scale, industrial size case studies for a practical evaluation of the framework.

References

1. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P.: A Verifiable Logic-Based Agent Architecture. In: Esposito, F., Raś, Z.W., Malerba, D., Semeraro, G. (eds.) ISMIS 2006. LNCS (LNAI), vol. 4203, pp. 188–197. Springer, Heidelberg (2006)
2. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Security Protocols Verification in Abductive Logic Programming: A Case Study. In: Dikenelli, O., Gleizes, M.-P., Ricci, A. (eds.) ESAW 2005. LNCS (LNAI), vol. 3963, pp. 106–124. Springer, Heidelberg (2006)

3. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Compliance verification of agent interaction: a logic-based software tool. Applied Artificial Intelligence 20(2-4), 133–157 (2006)
4. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable Agent Interaction in Abductive Logic Programming: the SCIFF framework. TOCL (accepted for publication, 2007)
5. Allen, J., Ferguson, G.: Actions and events in interval temporal logic. Journal of Logic and Computation 4(5) (1995)
6. Web Services Business Process Execution Language (WS-BPEL) Version 2.0. OASIS Standard, April 11, 2007, Organization for the Advancement of Structured Information Standards (2007)
7. Business Process Modeling Notation (BPMN) Specification. OMG Final Adopted Specification, February 6, 2006, Object Management Group (2006)
8. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: TROPOS: An Agent-Oriented Software Development Methodology. JAAMAS 8(3), 203–236 (2004)
9. Bryl, V., Massacci, F., Mylopoulos, J., Zannone, N.: Designing Security Requirements Models through Planning. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 33–47. Springer, Heidelberg (2006)
10. Cares, C., Franch, X., Mayol, E.: Extending Tropos for a Prolog Implementation: A Case Study Using the Food Collecting Agent Problem. In: Toni, F., Torroni, P. (eds.) CLIMA 2005. LNCS (LNAI), vol. 3900, pp. 396–405. Springer, Heidelberg (2006)
11. Chesani, F., Mello, P., Montali, M., Storari, S.: Towards a DecSerFlow declarative semantics based on computational logic. Technical Report DEIS-LIA-07-001, University of Bologna, Italy, LIA Series no. 79 (January 2007)
12. Cicekli, N.K., Yildirim, Y.: Formalizing Workflows Using the Event Calculus. In: Ibrahim, M., Küng, J., Revell, N. (eds.) DEXA 2000. LNCS, vol. 1873, pp. 222–231. Springer, Heidelberg (2000)
13. Corapi, D.: Traduzione di un linguaggio per l'ingegneria dei requisiti orientato agli agenti in logica computazionale. Technical Report DEIS-LIA-07-007, University of Bologna, Italy, Master's Thesis. LIA Series no. 85 (October 2007) (in Italian)
14. Dastani, M., Dix, J., Novák, P.: The First Contest on Multi-agent Systems Based on Computational Logic. In: Toni, F., Torroni, P. (eds.) CLIMA 2005. LNCS (LNAI), vol. 3900, pp. 373–384. Springer, Heidelberg (2006)
15. Dijkman, R., Dumas, M., Ouyang, C.: Formal Semantics and Automated Analysis of BPMN Process Models. Preprint 7115, Queensland University of Technology (2007)
16. Frankova, G., Massacci, F., Seguran, M.: From Early Requirements Analysis towards Secure Workflows. In: Proc. of IFIPTM 2007, The full version appears as Technical Report, DIT-07-036 (2007), http://eprints.biblio.unitn.it/archive/00001220/
17. Fung, T.H., Kowalski, R.A.: The IFF proof procedure for abductive logic programming. Journal of Logic Programming 33(2), 151–165 (1997)
18. Fuxman, A., Liu, L., Mylopoulos, J., Pistore, M., Roveri, M., Traverso, P.: Specifying and Analyzing Early Requirements in Tropos. REJ 9(2), 132–150 (2004)
19. Giorgini, P., Massacci, F., Zannone, N.: Security and Trust Requirements Engineering. In: Aldini, A., Gorrieri, R., Martinelli, F. (eds.) FOSAD 2005. LNCS, vol. 3655, pp. 237–272. Springer, Heidelberg (2005)
20. Henderson-Sellers, B., Giorgini, P. (eds.): Agent-Oriented Methodologies. Idea Group Publishing (2005)

21. Jaffar, J., Maher, M.: Constraint logic programming: a survey. Journal of Logic Programming 19-20, 503–582 (1994)
22. Kakas, A.C., Kowalski, R.A., Toni, F.: Abductive Logic Programming. Journal of Logic and Computation 2(6), 719–770 (1993)
23. Kazhamiakin, R., Pistore, M., Roveri, M.: A Framework for Integrating Business Processes and Business Requirements. In: EDOC 2004, pp. 9–20. IEEE Computer Society Press, Los Alamitos (2004)
24. Kowalski, R., Sergot, M.: A logic-based calculus of events. New Gen. Comput. 4(1), 67–95 (1986)
25. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-Driven Design and Configuration Management of Business Processes. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 246–261. Springer, Heidelberg (2007)
26. López, H.A., Massacci, F., Zannone, N.: Goal-Equivalent Secure Business Process Re-engineering. In: Proc. of SeMSoC 2007 (2007)
27. Massacci, F., Mylopoulos, J., Zannone, N.: An Ontology for Secure Socio-Technical Systems. In: Handbook of Ontologies for Business Interaction. The IDEA Group (2007)
28. Peterson, J.L.: Petri Nets. ACM Comput. Surv. 9(3), 223–252 (1977)
29. Pnueli, A.: The Temporal Semantics of Concurrent Programs. In: Proc. of the International Symposium on Semantics of Concurrent Computation, pp. 1–20. Springer, Heidelberg (1979)
30. Roscoe, A.W., Hoare, C.A.R., Bird, R.: The Theory and Practice of Concurrency. Prentice-Hall, Englewood Cliffs (1997)
31. Sadiq, S.W., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
32. Shanahan, M.: Reinventing shakey. In: Logic-based artificial intelligence, pp. 233–253. Kluwer Academic Publishers, Dordrecht (2000)
33. van der Aalst, W.M.P., Pesic, M.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)
34. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: Towards a truly declarative service flow language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
35. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
36. Venkatraman, M., Singh, M.P.: Verifying Compliance with Commitment Protocols. JAAMAS 2(3), 217–236 (1999)
37. Wong, P., Gibbons, J.: A Process Semantics for BPMN. Oxford. University Computing Laboratory (preprint, 2007), http://web.comlab.ox.ac.uk/

A Heuristic Approach to P2P Negotiation

Stefania Costantini, Arianna Tocchio, and Panagiota Tsintza

Dip. di Informatica, Università di L'Aquila, Coppito 67100, L'Aquila, Italy
{stefcost,tocchio,panagiota.tsintza}@di.univaq.it

Abstract. In this paper, we present a formal and executable approach to automated multi-issue negotiation between competitive agents. In particular, this approach is based on reasoning in terms of projections in convex regions of admissible values and is an extension of previous work by Marco Cadoli in the area of proposal-based negotiation. Our goal is to develop a heuristic strategy to flexibly compute the offers and counter-offers so as to fulfill each agent's objectives and minimize the number of agents' interactions. The proposed algorithm aims at improving a fundamental parameter of the negotiation process: the interaction complexity in the average case.

1 Introduction

Automated negotiation among software agents is becoming increasingly important as a consequence of the rapid development of web-based transactions and e-commerce. Negotiation is an important subject of study in the branch of Distributed Artificial Intelligence (DAI) and MAS (Multi-Agent Systems), as discussed for instance in [1,2,3,4].

A negotiation process can be defined as a particular form of interaction between two or more agents. As discussed, e.g., in [5] and [6], negotiation is a particular type of interaction in which a group of agents, with a desire to cooperate but with conflicting interests, work together in aim to reach a common goal, or to achieve an agreement that is acceptable by all parties in the process. More formally, negotiation can be defined as "a distributed research in a space of potential agreements" ([7]). In this sense, each participant involves its individual area of interest (also called *negotiation space* or *feasibility region*), and intends to reach agreements in that area. Negotiation spaces can be represented by a set of constraints. Then, finding an agreement can be modelled as a *constraint satisfaction problem* (CSP). In particular, in multi-agent systems the process of negotiation can be represented as a *distributed constraint satisfaction problem* (DCSP), since the constraints are distributed among different agents ([8]).

In proposal-based negotiation, the information exchanged between the parties is in the form of offers (internal points of the negotiation spaces) rather than constraints, preferences or argumentation. Each agent is able to compute the points to offer in order to reach an agreement. Negotiation terminates successfully whenever the participants on the process, find a point, in the space of negotiation, that is mutually acceptable. That point has therefore to be included

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 177–192, 2008.

in the common area of all negotiation spaces: i.e., in the intersection of the areas of interests [3].

Agents involved in the process of negotiation need to interact. Usually they are *self-interested*, since each one has different constraints to satisfy and different benefits, in terms of utility functions, to maximize (or minimize). The utility functions can be represented by new constraints on the agents' knowledge.

The research work reported here is an extension of previous work by Marco Cadoli, introduced in ([9]), and presents an heuristic strategy for proposal-based negotiation. The goal is to minimize the number of the interactions between the automated agents involved in the process and thus speed-up the search of an agreement - note that the speed of the process, or time complexity, largely depends on the particular negotiation strategy adopted by each agent. In this approach, negotiation spaces are considered to be convex, i.e., all points between two acceptable points are acceptable as well. The admissible offers are internal points of the negotiation areas, and those will be the only exchangeable information among the involved agents. The participating agents are capable of logical reasoning and are able to reason in means of projections. As discussed below (section 3), reasoning by means of projection can help the agents compute subsequent offers as each one can exclude certain points of the individual negotiation areas.

Both the original Marco Cadoli's approach and the proposed extension are of interest in Computational Logic because involved agents are assumed to be perfect logical reasoners, and then find a natural realization in logic-based agent-oriented languages. In fact, as discussed in Section 6 we have implemented the extended approach in one of these languages, namely in the DALI language. The rest of this paper is structured as follows. Section 2 is an overview of related work. In section 3, we present the theoretical background and the basic approach to negotiation that we adopt, introduced by Marco Cadoli. In section 4, we discuss our motivations for extending this basic approach. Section 5 is devoted to the presentation of the features of the extended negotiation model. In section 6 we present the implementation of the proposed strategy in DALI. In Section 7 we conclude and outline future work.

2 An Overview

Numerous strategies have been proposed in order to improve the efficiency, completeness and robustness of the process of negotiation (e.g., [9,3,10,11,12,13,14,15,16,17,18,19]). In [10], a number of agent strategies designed for the 2002 TAC (*Trading Agent Competition*) are reported and compared. These techniques include machine learning, adapted, planning and hybrid agents as well as heuristic-based strategies. The aim of [11], instead, is to determine how an agent (with firm deadlines) can select an optimal strategy based on an incomplete information about his opponent. STRATUM, reported in [12], is a methodology for guiding strategies for negotiating agents in non-game-theoretic domains.

[13] analyzes how a group of agents can deal in presence of *externalities*. In economics an externality is a cost or benefit resulting from a transaction that is borne or received by third parties, thus affecting social welfare. This has been done by adding to the worth of a bilateral coalition the amount of the negative externalities, that is created for the excluded player. The equilibrium values are respectively increased or decreased in the presence of negative or positive externalities. In [14], the line of research has produced a number of results on the hardness of dealing with positive and negative externalities while aiming at maximization of the social welfare. In [17] Matos et al. present an empirical evaluation of a number of negotiation strategies in different kinds of automated agents and environments. Rahwan et. al in [18] have developed a protocol-independent theory of strategy. In this approach, various factors that influence the generation of strategies are defined and used by the automated agents. In [19], instead, a negotiation model and two types of negotiation strategies, called *concession* and *problem solving* strategies, are presented.

The process of negotiation is considered as a constraint-based model in [9],[16] and [3]. In [16] and [3] negotiation is considered as a distributed constraint satisfaction problem without however considering complexity issues. Instead, the nature of the offers and the selection of variables assignment in the approach of [9] are mainly aimed at obtaining a reasonable complexity in terms of number of interactions steps. The speed of negotiation is tackled in [20], where however only boolean variables are considered.

3 Theoretical Background: The Approach by Marco Cadoli

In this paper, we discuss and extend the approach to reasoning by means of projections introduced by Marco Cadoli and reported in ([9]). In this section, we first present the approach and then discuss why some extensions are needed and are useful.

In Marco Cadoli's approach, negotiation is considered as a distributed constraint satisfaction problem. The assumptions made by Marco Cadoli [9] are the following. (i) Negotiation involves two or more parties, that exchange proposals until either an agreement is found (i.e., the last proposal is acceptable by all the parties involved) or there is an evidence of the fact that no agreement is possible. (ii) Negotiation involves variables (also called negotiation issues). (iii) A proposal (or "offer") is a communication act possibly containing the assignment of values to the involved variables ("variable assignment"). (iv) Negotiation is restricted with no loss of generality to involve only two variables. The approach however could be easily generalized. (v) The negotiation space (also called negotiation region or feasibility region or area of interest) associated to each party coincides with the set of variable assignments that are considered to be acceptable, i.e., where the value assigned to each variable is within the range that the party considers to be acceptable. (vi) As only two variables are involved, negotiation spaces are restricted to be regions in the cartesian plane. (vii) A possible proposal is in principle any point of the negotiation space. (viii) Negotiation spaces are restricted to be convex regions, i.e., all points included within the boundaries of each individual region belong to the region itself and thus are equally acceptable as potential agreements. Therefore, each negotiation space can be described by a set of constraints which describe the region perimeter by describing the acceptable range of values for each variable. (ix) Negotiation spaces are considered to be polyhedral (thus, negotiation spaces admit a finite number of vertices). (x) For one of the two agents, who wishes to minimize the number of negotiation steps, possible proposals are restricted to be the vertices of negotiation spaces.

It is assumed (again without loss of generality) that negotiation is restricted to two parties, that have agreed in advance on the issues which are involved. Of course, the parties (that later on we will call "agents") associate a meaning to the variables, that in their view may represent prices, time, number of items, etc. A negotiation starts when one of the two agents makes a proposal. The other one will respond with a counter-proposal, and the process will go on in subsequent *steps* (where each agent responds to the other one's last proposal). The other party is also called the "opponent". The negotiation process will end either because an agreement has been found, or if there is an evidence that no agreement is possible. Since negotiation spaces are considered as convex regions, a necessary condition for an agreement to be reached is that the intersection of the feasibility regions is not empty.

As offers are for one of the two parties restricted to be vertices of a polyhedral region, in Marco Cadoli's approach the process will necessarily end whenever this party has no more vertices to offer. This means that each negotiation process always converges to an end in a finite number of steps. However, the number of vertices can be, in the worst case, exponential in the number of variable. Thus, the process has a worst-case exponential complexity in the number of negotiation steps. Therefore, the approach defines a negotiation protocol aimed at obtaining in the average case large savings in terms of number of proposed vertices. The underlying assumptions about the participating agents are the following. (a) Agents are perfect logical reasoners. (b) Agents communicate only by exchanging proposals and there is no other form of shared knowledge. When a

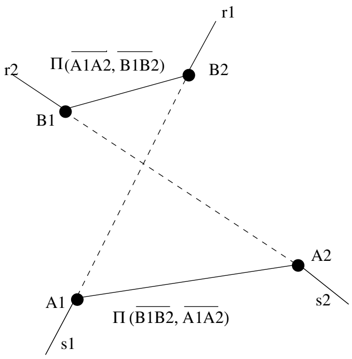

r1
r2
$\Pi(\overline{A1A2}, \overline{B1B2})$
B2
B1
A2
A1
s2
$\Pi(\overline{B1B2}, \overline{A1A2})$
s1

Fig. 1. Lines segments projections

proposal has been issued, it becomes common knowledge for all involved parties. (c) Agents are partially cooperative, in the sense that they are aware of the negotiation protocol that they apply faithfully, i.e., they do not make offers that they are not really willing to accept. (d) Agents do not cheat, i.e., they do not make proposals that are not implied by the protocol at that step and, (e) during a negotiation, they aim at minimizing the number of steps.

The main point of the approach is that agents are able to reason by means of projections. This kind of reasoning is illustrated below. In this setting, we consider the projection of a line segment over another one. We recall this concept by means of an example (in Figure 1). In the example, we consider the four points A_1, A_2, B_1, B_2. The figure highlights two projections:

1. The projection of the segment $\overline{A_1A_2}$ over $\overline{B_1B_2}$ (denoted with $\Pi(\overline{B_1B_2}, \overline{A_1A_2})$ and delimited by $r_2\overline{B_1B_2}r_1$), and
2. The projection of $\overline{B_1B_2}$ over $\overline{A_1A_2}$ (denoted with $\Pi(\overline{A_1A_2}, \overline{B_1B_2})$ and delimited by $s_1\overline{A_1A_2}s_2$).

Suppose for instance, referring to Figure 1, that points A_1 and A_2 are offers made by agent A, while B_1 and B_2 are offers made by agent B. Suppose that the steps, i.e., the order in which proposal are exchanged, are as follows: Agent A offers A_1, agent B counter-offers B_1, agent A replies with A_2 and agent B with B_2. Agent A (being a perfect logical reasoner and being aware of convexity and projections and being aware that the other agent has the same potential) is now able to perform the following reasoning. From proposals $A1$ and $A2$ and from the convexity hypothesis, agent B knows that the whole segment $\overline{A_1A_2}$ belongs to A's negotiation area. Then, as B aims at minimizing the number of steps, if B had a point of its region on that segment, it would have offered it. As B instead has offered B_2, agent A is allowed to conclude that the intersection between B's region and segment $\overline{A_1A_2}$ is empty.

Also, A can consider that if B had any vertex on: either the line $s1$ that goes beyond segment $\overline{B_2A_1}$ in the direction of A_1 or on the line $s2$ that goes beyond segment $\overline{B_1A_2}$ in the direction of A_2 then, again by convexity, the segment $\overline{A_1A_2}$ would necessarily belong to B's region: in fact, any polyhedral region including as vertices B_1, B_2 and these two hypothetical points would also include segment $\overline{A_1A_2}$, which is a contradiction. Therefore, A is able to exclude the whole projection area delimited by segment $\overline{A_1A_2}$ and by the two above-mention lines. In fact, no point in there can be possibly acceptable for B, and thus A will choose no such point as an offer. This area can be obtained, as observed before, by making the *projection* of segment $\overline{B_1B_2}$ over $\overline{A_1A_2}$.

Marco Cadoli's approach adopts the strategy of concluding the process in the minimum possible number of interactions: excluding portions of the feasibility region actually leads to excluding many potential offers and thus reducing the number of steps. In fact, agent B can perform a similar reasoning if A responds to offer B_2 with a counter-offer that does not constitute an agreement. The reasoning can then be repeated by both agents after subsequent offers.

In [9] it has been proved that reasoning in terms of projections leads to a protocol that always converges and in some cases allows for large savings in terms of number of proposed vertices (in the worst case, the number of agent's interactions is exponential in the number of variables and in the number of constraints). It is important to notice that the assumption of **only one party** being restricted to offer vertices is essential. Otherwise, going back to the example we may notice the following: if B were restricted to offer vertices then B might offer a point on segment $\overline{A_1A_2}$ only if B had a vertex there. Therefore, vertex-based reasoning would be approximate and, though often leading to a speed-up, might exclude possible agreements. Each agent is assumed to be self-interested, which means that it ignores the other agents preferences. However, as discussed below, if the agent considers *its own* preferences the potential speed-up can go lost. This is why we devised to extend this approach.

For the sake of clarity, before discussing the extensions we consider a more complete example. Let us consider as before a bilateral peer-to-peer negotiation process involving two issues. Let us introduce the two involved agents, say *Seller* and *Buyer*, represented by the respective negotiation areas reported in Figure 2. In this example, the negotiation area (indicated as R_A) of agent *Seller* can be described by the set of constraints $C_A = \{X \succeq 4, X \preceq 20, Y \succeq 13, Y \preceq 40, X \succeq -3Y+49, X \preceq Y+4, X \succeq (3/2)Y-50, X \preceq -(3/4)Y+47\}$. The negotiation area (indicated as R_B) of the agent *Buyer* can be described by the set $C_B = \{X \succeq 15, X \preceq 40, Y \succeq 10, Y \preceq 25\}$. Let $V_A = \{A_1 = (10,13), A_2 = (17,40), A_3 = (4,15), A_4 = (17,13), A_5 = (20,16), A_6 = (20,36), A_7 = (4,36), A_8 = (10,40)\}$ be the sets of possible proposals (set of vertices of the negotiation space) of the agents *Seller* and $V_B = \{B_1 = (15,10), B_2 = (40,25), B_3 = (40,10), B_4 = (15,25)\}$ be the possible proposals of the agent *Buyer*. The intersection area $I = R_A \bigcap R_B$ is clearly not empty, and therefore there is a potential agreement between the two agents.

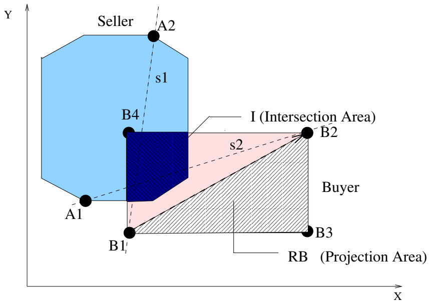


Fig. 2. An example

We assume that the negotiation process starts with a proposal from the agent *Buyer* and that the sequence of proposals is as follows: B_1, A_1, B_2, A_2. Each interaction has the side-effect of updating the knowledge base of the agents by storing all proposals (both made and received) and possible new constraints. The subsequent steps of the negotiation are determined as follows:

- Since agent *Buyer* has received as counter-offer point A_1, not included in its negotiation area, it rejects the proposal.
- After two interactions, the agents have exchanged four proposals, namely B_1, A_1, B_2, A_2 and none of them has been accepted. At this point, the agent *Buyer* computes a projection area by connecting the couples of points (B_1, A_2), (B_2, A_1) and (B_1, B_2) and by adding to its knowledge base the new linear constraints that represent the new lines s_1 and s_2. The agent is able to conclude that the intersection of the other part negotiation area with the projection area (delimited by the lines s_1, s_2 and by the segment (B_1, B_2)) is empty: therefore, no point in the projection area can be accepted from the counter-part. In fact, agent *Seller* knows that the buyer having offered points B_1 and B_2 implies (by the convexity hypothesis) that the whole segment connecting the two points belongs to the buyer's negotiation area. Then, had the seller's negotiation area admitted a non-empty intersection with this segment, the seller would have offered a vertex either on the segment or beyond it. Since the buyer instead receives point A_2, it becomes aware that no such intersection exists and, consequently, that no point in the aforementioned projection belongs to the seller's area.

  The agent exploits the updated knowledge for selecting the next offer to make, having excluded all points that belong to the projection area. Thus, the agent *Buyer* understands that its vertex B_3 cannot be accepted by the opposer: it excludes this point from the set of proposals and proceeds by offering its point B_4.
- Finally, this offer belongs to the negotiation area of the agent *Seller* and therefore this proposal will be accepted. In this case, we say that the negotiation process terminated successfully.

  In the case where the last offer does not belong to the negotiation area of the opposer, the agent *Buyer* would conclude that there is no further point to propose and would terminate the negotiation process, having an evidence of the fact that the intersection of the two negotiation areas is empty and therefore there is no possible agreement.

4 Limits of the Original Approach

The reasons why we propose an extension to the basic approach are summarized in the following points:

- We intend to allow the two agent to both follow the same protocol. If so, limiting the possible proposals to vertices inducts problematic trades: in particular, this happens whenever the intersection area is not empty but includes no vertices. For example, in Figure 3 there are two agents, *Seller* and *Buyer*, whose individual negotiation areas are expressed through convex regions. It can be seen that there is a potential agreement amongst the agents, since the intersection area includes various points. However in this case, after six interactions (namely, the sequence of proposals is $B_1, A_1, B_2, A_2, B_3, A_3$) the seller agent understands that there is no other vertex to propose: in fact, it has previously excluded vertex A_4. Thus, it concludes the process of negotiation with a proof that there no possible agreement. This problem is due to the fact that only vertices can be proposed.

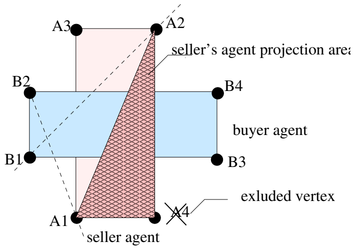


Fig. 3. A first problematic interaction

- The "flat" nature of proposals (where all vertices are equally considered) may lead, in real applications, to other problematic situations. In particular, it may frequently happen that one or more issues considered by an agent have greater priority than others. Let us assume that the agents try to reach an optimal point with respect to an objective function. This objective function can be chosen according to the particular context. In figure 4, for example, we consider two agents, *Business* and *Client*. We assume that *Business* wishes to maximize the issue Y and that the sequence of interactions starts with a proposal (from *Client*) of point B_1. The interaction proceeds with A_1, B_2 and A_2. In this case, the approach of reasoning by means of projections does not allow us to obtain savings in terms of number of proposed vertices.

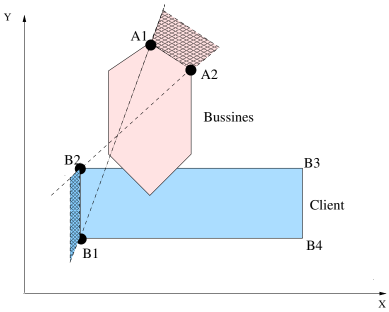


Fig. 4. A second problematic interaction

- As mentioned in [9], another problem of the approach of reasoning by means of projections is that, since the agents have to remember all proposals (both made and received), it is hard to find algorithms and data structures which allow agents to store the entire sequence of proposals in polynomial space.

– In [9] another problem is mentioned: the limit case where negotiation spaces are convex (and finite) regions represented by circles. In this case, agents are unable to select their offers since there are not vertices and there is an infinite number of possible choices.

We have tried to overcome some of the problems discussed above. In particular, we assume proposals to be not only vertices but also internal points of the convex regions. Those points will not be randomly selected. Their selection, instead, will be based on recent proposals by the same agent.

5 Proposed Extension

In this Section we illustrate the proposed extended approach by means of an example and then we formally define the new negotiation algorithm.

5.1 An Example

Consider two agents A and B, represented by their negotiation areas denoted as R_A and R_B. In general, as discussed above, the two bidding agents can reach an agreement only in the case where $R_A \bigcap R_B \neq \emptyset$.

We propose a change in the nature of the proposals. In fact, in the extended algorithm agents are allowed to make proposals that are internal points of the feasibility regions, rather than just vertices. The strategy still takes advantage from reasoning by means of projections: projection areas will be created dynamically during negotiation, so that agents will be able to exclude many internal points of the individual feasibility region. This implies a better accuracy in the selection of offers.

Assume that the two negotiation areas R_A and R_B are those represented in figure 5. The process of negotiation initiates with a proposal, for example by agent A. The first proposal is assumed to be a random point of the individual convex region of the agent. The second proposal of agent A, instead, is identified as follows: A computes the circumference whose center corresponds to the point of the first proposal and whose radius is R = δ, where the choice of the margin δ will depend on each specific application context (his margin can be chosen,

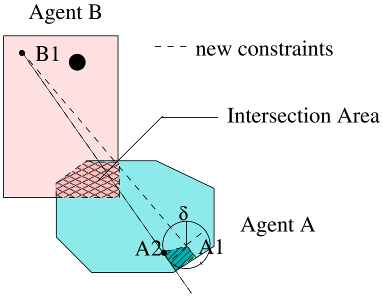


Fig. 5. The trade-off strategy in multi-variable regions

for example, as in [21].). The proposal will be selected as a random point of the semi-circumference closer to the opposer's first proposal.

Clearly, we require it to be included on the individual feasibility region. In this way, the agent tries to approach the opposer's offer by proposing a point that is more likely to be accepted, in an attempt to adapt its individual profile to the one that can be assumed for the opposer. To do this, the agent has to add new constraints to its knowledge base. If no such point is found, then the next proposal will be a new random point of the feasibility region. After that, each agent uses the reasoning on projection in the same way as in [9]. The points included in the projection areas will be stored in the agents memory, by adding the new constraints that represent each projection area to its knowledge base.

All subsequent proposals will be selected in the following way: each agent computes the segment that connects the two last offers of the opposer. Note that, as discussed before, all points of this segment are necessarily included in the opposer's area. Therefore, if agent A area admits a non-empty intersection with sg, being a perfect reasoner A will offer a point therein, thus reaching an agreement. This constitutes an enhancement w.r.t. the basic approach where, being each agent bound to offer vertices, we have the following two possibilities: the case where agent A has a vertex on sg that it can propose, and thus the agreement is found; the unfortunate case where it has no vertex on sg, and then it can just offer a vertex *beyond* the segment, without being sure of that vertex still belonging to the intersection.

If instead agent A area has empty intersection with sg, then the next proposal will be selected in the same way as the second one, where however the center of the new circumference will be A's last proposal. Clearly, any the new proposal must not be included in the projections made so far. The projection areas may be described in terms of a set of linear inequations. In this way, the agent will find the new points to offer by solving a new, or extended, DCSP problem.

An advantage of the new algorithm is that agents do not have to store all the potential proposals. Rather, the only information that an agent needs in order to construct the new projections (i.e., the constraints), consists of the two most recent proposals made by the two parties. If a satisfactory contract has not been found yet, then the agent continues with the next proposal and so on.

5.2 Definition of the Algorithm

In this section we are going to illustrate the precise definition of the algorithm. In the following, we consider $proposal_{J_i}$ as the i^{th} offer of the agent J ($J \in \{A, B\}$). Each agent maintains a knowledge base KBJ. More precisely, we will denote by KBA_i the knowledge base of agent A after the i^{th} interaction. In the description, expressions such as *offer(.)*, *accept(.)* and *reply* enable agents communication. Notice that an agent which performs a communication act considers it as an *action* while the receiver perceives communications as *external events.*

```
Algorithm                                                    AgentA
if ($reply$ = $accept$)
 then make($contract$);
else
  if (interest & $proposal_{B_0}$)                  /* first proposal to make
   then
    (a) identify $proposal_{A_1}$ as random point of the feasibility region;
    (b) $offer(proposal_{A_1})$;
    (c) wait for a $reply$;
  else
    if ($reply = (proposal_{B_i} \in R_A)$)
     then
      $accept(proposal_{B_i})$;
    else
     if (i = 1)                                     /* second proposal to make
      then
       (a) find a random point $A_2 = (x_2, y_2)$ of the semi-circumference
             closer to the $proposal_{B_1}$ (radius = $\delta$);
       (b) construct the projections;
       (c) add the new constraints which represent the projection
             area to the agent's knowledge base in $KBA_2$;
       (d) $offer(proposal_{A_2})$;
       (e) wait for a $reply$;
     else                                           /* $i > 1$ (subsequent proposals)
      if ($\overline{B_{i-1}B_i} \bigcap R_A \neq \emptyset$)
       then
        (a) find common point $A_{i+1}$ of $\overline{B_{i-1}B_i} \bigcap R_A$;
        (b) $offer(proposal_{A_{i+1}})$;
      else
       do
        (a) find random point $A_{i+1}$ (with (($A_{i+1} \in R_A$) &
              ($A_{i+1} \notin KBA_i$))) upon the semi-circumference closer to
              the $proposal_{B_i}$;
        (b) construct the projections;
        (c) add the new constraints which represent the projection
              area to the agent's knowledge base in $KBA_{i+1}$;
        (d) $offer(proposal_{A_{i+1}})$;
        (e) wait for a $reply$;
       while ($proposal_{B_i} \notin R_A$)              /*no agreement is found
```

Since there is a huge (infinite, in principle) number of points included in an agent feasibility region, convergence of the proposed algorithm is not guaranteed. The simplest way of coping with this problem is an upper bound to the number of allowed interactions. A (bounded) number of random restarts can also be in order. However, in [22] we show that some additional assumptions can guarantee convergence, at the expense of less precision in the identified solution.

The extended approach is not in contrast to the vertex-based basic one: rather, it is complementary in many potential ways. A first possible integration may apply the extended algorithm whenever there is no more vertex to offer. A smarter integration, which copes with the problem of local preferences/objectives, may assume that an agent wishes to find an agreement which is as close as possible to a *preferred* vertex: the extended approach can be applied starting from this vertex (instead of starting randomly) so as to approach step-by-step the opposer while staying as long as possible in the preferred subpart of the region. In this sense, the new approach can be exploited to define "local search" variants of the original one.

5.3 Performance of the Proposed Algorithm

In [22] we present many experiments to evaluate the performance of our approach, also in comparison to the original one, where performance is here intended in terms of the number of iterations necessary to conclude the negotiation process. The number of iterations in our case is no more proportional in the number of vertices, contrary to [9]. Rather, it heavily depends upon the dimension of the intersection area and upon the distance between the first two offers. However, tests have shown that our new approach results to be in average case even more efficient than the original one.

Parameters that influence the number of iterations are at least: i) the size of the intersection area, ii) the size of the negotiation areas, iii) the (Euclidean) distance between the first two proposals and iv) the δ parameter. Experiments have shown that the size of the intersection area and the number of iterations will tend to be in inverse proportion. Vice versa, the size of the negotiation areas is proportional to the number of iterations which means that when the first one increases the second one increases as well. The Euclidean distance between the first proposals influences the algorithm performance since the more distant they are the greater number of iterations is necessary to conclude the process.

The choice of the δ parameter deserves some discussion: with very small δ's each agent has to make a long way to approach the opposer's proposal, requiring many "steps". If instead δ is too big, even bigger than the intersection area, then some steps can "jump" the intersection and force to randomly walk around the intersection itself. For the particular case of rectangular feasibility regions, the best value for δ appears to be a bit less that the smallest edge. In fact, even starting from a random point there is a good chance to meet the intersection quickly.

6 Implementation

The proposed approach has been implemented in the DALI language. DALI ([23,24,25,26,27]) is an Active Logic Programming language designed for executable specification of logical agents. DALI is a prolog-like logic programming language with a prolog-like declarative and procedural semantics [28]. In order to introduce reactive and proactive capabilities, the basic logic language has been syntactically, semantically and procedurally enhanced by introducing several kinds of *events*, managed by suitable *reactive rules*. In this section, we present a snapshot of the code developed for the implementation of the above-discussed negotiation algorithm. The implementation is quite straightforward, as the proposed algorithm basically consists in communication and logical reasoning, which find a natural and direct implementation in DALI.

All constraints (and therefore the representation of the negotiation area) as well as the margin δ to be used during the interaction are stored in the agent profile and are loaded at runtime. This makes the implementation elaboration-tolerant w.r.t. changes to the negotiation parameters. The proposals of the counter-part is received by the agent by means of a DALI reactive rule: $\mathit{offerE}(X, Y, A)\ :>\ \mathit{once}(\mathit{reconsider}(X, Y, A))$. Whenever an agent receives an offer it checks if the offer is included in the negotiation area and responds accordingly, by accepting the proposal. This is implemented via the following rule:

```
reconsider(X,Y,A):- area(X,Y),!,clause(agent(A),_),
       messageA(clientnew,send_message(accept_pr(X,Y,A),A)).
```

Otherwise, by using the rule $\mathit{call\_random\_semicycle}(X1, Y1, X, Y)$, the agent selects a (random) point of the semi-circumference closer to the opposer's proposal, provided that it belongs to the negotiation area and does not belong to the projection areas constructed so far. After that, the agent sends a message containing the counter-offer by means of an action of the kind *messageA*. Finally, it updates its knowledge base by adding the new constraints that represent the projection areas and by updating the last four proposals (both made and received).

```
reconsider(X,Y,A):- out_of_area(X,Y),(Az >X,Bz>Y,out(_,_,_,_,X,Y)),
  call_random_semicycle(X1,Y1,X,Y),
        messageA(clientnew,send_message(new_offer(X1,Y1,A),A)),
  clause(agent(A),_), update_offer1(offer1(_,_),Ac,Bc),
  update_offer2(offer2(_,_),X,Y),update_proposal1(proposal1(_,_),Az,Bz),
  update_proposta2(proposta2(_,_),X1,Y1),clause(offers(L),_),
  append([Y],L,L1), append([X],L1,L2),assert(offers(L2)),
  retractall(offers(L)),clause(proposals(Lp),_),append([Y1],Lp,L3),
  append([X1],L3,L4),assert(proposals(L4)),retractall(proposals(Lp)).
```

As an example of the pro-active capabilities of the agent, we show how the agent checks whether a point is included in the projection areas. This check employs an *internal event*, represented by a pair of DALI rules. The conclusion of the first rule is automatically attempted from time to time. If at some point it becomes true (i.e., it can been proved), possibly returning some values for the output variables, then the (instantiated) body of the second rule (the reactive one) is executed.

```
update_history(X,Y):- offerP(X,Y,_).
update_historyI(X,Y):>
```

```
   [_,_,Xo2,Yo2,Xo1,Yo1|L1]=Lo,[_,_,Xp2,Yp2,Xp1,Yp1|L2]=Lp,
   new_condition(X,Y,Xp2,Yp2,Xo2,Yo2,Xp1,Yp1,Xo1,Yo1),
   update_constraints([Xo2,Yo2,Xo1,Yo1|L1],[Xp2,Yp2,Xp1,Yp1|L2],X,Y).

new_condition(X,Y,X1,Y1,X2,Y2,X3,Y3,X4,Y4):-
   X4>X1,X2>=X4,Y2>=Y4,X1=X3,!,X1>=X,
 coeff2(X1,Y1,X2,Y2,X3,Y3,X4,Y4,M2),Cost2 is Y4-(M2*X4),(M2*X)+Cost2>=Y,
 coeff3(X1,Y1,X2,Y2,X3,Y3,X4,Y4,M3),Cost3 is Y2-(M3*X2),Y>=(M3*X)+Cost3.
```

Here, $\mathit{update\_historyI}(X, Y))$ is an internal event that is triggered each time the agent has received a new offer (recorded as a past event, suffix P). The procedure $\mathit{new\_condition}(X, Y, Xp2, Yp2, Xo2, Yo2, Xp1, Yp1, Xo1, Yo1)$ builds the new projection (by constructing the new constraints) while $\mathit{update\_constraints}$ $([Xo2, Yo2, Xo1, Yo1|L1], [Xp2, Yp2, Xp1, Yp1|L2], X, Y)$, updates the knowledge base of the agent by adding new constraints.

7 Concluding Remarks and Future Work

The extension proposed in this paper to the original approach by Marco Cadoli is based upon adopting a heuristic algorithm which considers not only the vertices as possible offers but also internal points of the feasibility regions. By comparing the proposed algorithm with the one reported in [9] we conclude that our work overcomes some problems, even though in our case the number of interactions is no more proportional to the number of vertices. In fact, the proposed algorithm has no requirements on the nature of the negotiation spaces (i.e., we relax the limitation to polyhedric areas). The new approach can be usefully integrated with the basic approach in many ways, thus trying to keep the efficiency of vertex-based interaction whenever possible, and exploiting the additional flexibility when deemed useful. The proposed extension in general produces a more accurate solution, as it is able to consider all points included in the negotiation areas. The performance of the extended algorithm is worse in case of negotiation spaces with a limited number of vertices, but is better, in average, in the opposite case (high number of vertices).

Even though the algorithm complexity, in some extreme cases, can be considered high we claim that the granularity of the search space can justify this fact. Moreover, the additional complexity is a reasonable price to pay for the extra features and for the possibility of other extensions. In fact, the approach can be further extended: we may add new protocols, objective and utility functions. We have been studying the possibility of considering as negotiation spaces not only convex areas but also non-convex ones, by converting a non-convex region into a convex one and by excluding all points that are not part of the original negotiation area [29].

Acknowledgments

This work started as a cooperation with Marco Cadoli. Unfortunately, Marco left us in November 2006 after a long illness. This paper is dedicated to him.

References

1. Guttman, R.H., Moukas, A.G., Maes, P.: Agent-mediated electronic commerce: a survey. In: Knowl. Eng. Rev., vol. 13(2), pp. 147–159. Cambridge University Press, New York (1998)
2. Guttman, R.H., Maes, P.: Cooperative vs. competitive multi-agent negotiations in retail electronic commerce. In: Klusch, M., Weiss, G. (eds.) CIA 1998. LNCS (LNAI), vol. 1435, pp. 135–147. Springer, Heidelberg (1998)
3. Kowalczyk, R., Bui, V.: On constraint-based reasoning in e-negotiation agents. In: Proceedings of Agent-Mediated Electronic Commerce III, Current Issues in Agent-Based Electronic Commerce Systems, London, UK, pp. 31–46. Springer, Heidelberg (2001)
4. Bozzano, M., Delzanno, G., Martelli, M., Mascardi, V., Zini, F.: Logic programming and multi-agent system: A synergic combination for applications and semantics. In: Logic Programming and Multi-Agent System: A Synergic Combination for Applications and Semantics, The Logic Programming Paradigm - A 25-Year Perspective, pp. 5–32. Springer, Heidelberg (1999)
5. Walton, D.N., Krabbe, E.C.W.: Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning. State University of New York Press, Albany (1995)
6. Governatori, G., ter Hofstede, A.H., Oaks, P.: Defeasible logic for automated negotiation. In: Proceedings of CollECTeR, Deakin University (2000)
7. Jennings, N.R., Faratin, P., Lomuscio, A.R., Parsons, S., Sierra, C., Wooldridge, M.: Automated negotiation: prospects, methods and challenges. International Journal of Group Decision and Negotiation 10(2), 199–215 (2001)
8. Yokoo, M., Durfee, E.H., Ishida, T., Kuwabara, K.: The distributed constraint satisfaction problem: Formalization and algorithms. Knowledge and Data Engineering 10(5), 673–685 (1998)
9. Cadoli, M.: Proposal-based negotiation in convex regions. In: Proceedings of Cooperative Information Agents VII (CIA), pp. 93–108 (2003)
10. Greenwald, A.: The 2002 trading agent competition: an overview of agent strategies. AI Magazine 24(1), 83–91 (2003)
11. Fatima, S.S., Wooldridge, M., Jennings, N.R.: Optimal negotiation strategies for agents with incomplete information. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), vol. 2333, pp. 53–68. Springer, Heidelberg (2002)
12. Rahwan, I., Sonenburg, L., Jennings, N.R., McBurney, P.: Stratum: A methodology for designing heuristic agent negotiation strategies. International Journal of Applied Artificial Intelligence 21(6), 489–527 (2007)
13. Gomes, A.R.: Valuations and dynamics of negotiations. Rodney L. White Center for Financial Research Working Paper, 21–99 (2004)
14. Conitzer, V., Sandholm, T.: Expressive negotiation in settings with externalities. In: Proceedings of the 20^{th} National Conference on Artificial Intelligence (AAAI 2005), Pittsburgh, Pennsylvania, USA, pp. 255–260 (2005)
15. Kazmer, D., Zhu, L., Hatch, D.: Process window derivation with an application to optical media manufacturing. Journal of Manufacturing Science and Engineering 123(2), 301–311 (2001)
16. Rahwan, I., Kowalczyk, R., Pham, H.H.: Intelligent agents for automated one-to-many e-commerce negotiation. In: Proceedings of the twenty-fifth Australasian conference on Computer science (ACSC 2002), Darlinghurst, Australia, pp. 197–204. Australian Computer Society, Inc. (2002)

17. Matos, N., Sierra, C., Jennings, N.R.: Determining successful negotiation strategies: An evolutionary approach. In: Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS 1998), Paris, France, pp. 182–189. IEEE Computer Society Press, Los Alamitos (1998)
18. Rahwan, I., McBurney, P., Sonenberg, L.: Towards a theory of negotiation strategy. In: Proceedings of the 5th Workshop on Game Theoretic and Decision Theoretic Agents (GTDT-2003), pp. 73–80 (2003)
19. Lopes, F., Mamede, N.J., Novais, A.Q., Coelho, H.: Negotiation strategies for autonomous computational agents. In: Proceedings European Conference on AI (ECAI), pp. 38–42 (2004)
20. Wooldridge, M., Parson, S.: Languages for negotiation. In: Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI 2000), pp. 393–397 (2000)
21. Somefun, K., Gerding, E., Bohte, S., La, H.: Automated negotiation and bundling of information goods. In: Proceedings of the 5^{th} Workshop on AgentMediated Electronic Commerce (AMEC V), Melbourne, Australia (2003)
22. Costantini, S., Tocchio, A., Tsintza, P.: Experimental evaluation of heuristic approach to p2p negotiation. A journal, draft available from the authors, preliminary version; In: Proc. of RCRA 2007, Intl. Workshop on experimantal evaluation of Algorithms for Solving Problems with Combinatorial Explosion, 2007 (2008)
23. Costantini, S., Tocchio, A.: A logic programming language for multi-agent systems. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 1–13. Springer, Heidelberg (2002)
24. Costantini, S., Tocchio, A.: The dali logic programming agent-oriented language. In: Alferes, J.J., Leite, J.A. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 685–688. Springer, Heidelberg (2004)
25. Tocchio, A.: Multi-agent systems in computational logic. Ph.D. Thesis, Dipartimento di Informatica, Universitá degli Studi di L'Aquila (2005)
26. Costantini, S., Tocchio, A.: About declarative semantics of logic-based agent languages. In: Baldoni, M., Endriss, U., Omicini, A., Torroni, P. (eds.) DALT 2005. LNCS (LNAI), vol. 3904, pp. 106–123. Springer, Heidelberg (2006)
27. Costantini, S., Tocchio, A., Verticchio, A.: A game-theoretic operational semantics for the dali communication architecture. In: Proceedings of WOA 2004, Turin, Italy, pp. 13–21 (December 2004); ISBN 88-371-1533-4
28. Lloyd, J.W.: Foundations of Logic Programming, 2nd edn. Springer, Berlin (1987)
29. Siegel, A.: A historical review of the isoperimetric theorem in 2-d, and its place in elementary plane geometry (2003), http://www.cs.nyu.edu/faculty/siegel/sciam.pdf

Towards Context Sensitive Defeasible Rules

Armin Hezart, Abhaya Nayak, and Mehmet Orgun

Intelligent Systems Group, Department of Computing
Division of ICS, Macquare University, Sydney, NSW 2109, Australia
{ahezart,abhaya,mehmet}@ics.mq.edu.au

Abstract. Defeasible argumentation systems are used to model commonsense and defeasible reasoning. Current argumentation systems assume that an argument that appears to be justified also satisfies our expectation in relation to the correct outcome, and, vice versa. In this paper we present an alternative representation of defeasible rules, tailored for argumentation based defeasible reasoning, that is free of such an assumption. We provide a mapping between our argumentation system and Dung's abstract argumentation theory to show its efficacy.

1 Prelude

Arguments and Argumentations, as far as human activities go, are quite ancient. Arguments are used to draw conclusions that are supported by available premises. Argumentation, on the other hand, is the interactive process used to determine what conclusions are to be ultimately drawn when multiple arguments are available that directly or indirectly, support or oppose a conclusion. From the point of view of argumentation, arguments are defeasible – a seemingly good argument can fall prey to an opposing argument if the latter is justified, and remains so. This aspect of argumentation, that it deals with arguments that are only *prima facie* justified, is exploited in connecting argumentation theory with defeasible reasoning.

The defeasibility of reasoning is captured in different ways in different frameworks. Very roughly, in the framework of default logic it is captured by assuming that the rules are defeasible, allowing for alternative extensions depending on which set of defaults get activated.[1] Often rules of thumb such as *specificity* [19,10,25],[2] are used to break the tie in case of conflicting rules. In circumscription [14], it is achieved by minimizing the extension of "abnormal" predicates. In the case of argumentation, it is achieved by allowing some arguments to *defeat* other arguments.

In this paper we introduce an alternative representation of defeasible rules that is context sensitive. Effectively, we assume that a mechanism exists that given an arbitrary rule, tells us whether in a given situation the rule is applicable. As already mentioned

[1] Informally, a default rule is of the form: *If A is known as a matter of fact, and B can be assumed without courting inconsistency, then C may be inferred.* Thus, given a knowledge base, the rule itself tells, as it were, whether or not it can be "fired".

[2] *Given two rules, applicable to a given context, the one that makes use of more specific information takes precedence over the other.* Thus, if we know Tweety is a penguin, and given the rules that birds in general fly and penguins don't, we should conclude that Tweety does not fly.

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 193–213, 2008.

there are already a few representations of defeasible rules such as Reiter's default rules or McCarthy's circumscription rules. However, most of the proposed rules are context independent, in the sense that the condition that makes a rule applicable is context independent. In this paper we refer to these context independent (abnormality) conditions conclusive defeaters of a rule.

In the next section we provide a short background of argumentation systems, setting up the context for two running examples that we use as our motivation. In section 2 we elaborate on how these defeasible rules are represented. In section 3 we develop an argumentation theory based on the attack and reinstatement relationships between arguments. In section 4 we discuss the semantics of our argumentation system by providing a translation from our system to Dung's abstract argumentation framework. We also argue that in case of conclusive defeaters, based on the chosen semantics our argumentation system give the same outcome as systems based on conventional default rules. Finally, we conclude with a short summary.

2 Background

From the outset we assume a propositional language $\mathcal{L}$ composed of countably many literals (both positive and negative) and a set of defeasible inference rules $\mathcal{R}$. Technically a rule is a relation between a set of literals called premises and another literal called a conclusion.

> **Notation.** An inference rule, d, is represented as $d : a_1, a_2, ..., a_n \rightarrow a$ where $a_1, a_2, ..., a_n, a \in \mathcal{L}$. We call $bd(d) = \{a_1, a_2, ..., a_n\}$ the *body* of the rule, and, $hd(d) = a$ its *head*.

An argument is usually defined as a sequence of inferences from known premises (the *contingent knowledge*) to a conclusion. Alternatively it is represented as an inference-tree-structure embedded in the premises, or, as a pair of premises and conclusion. We use all these three representations dictated by convenience.

We note that unlike the truth-based classical logical systems, argumentation systems are founded upon justification [15]. An argument is accepted in the absence of a justified counter-argument. The counter-argument against an argument is called the attacker [9] or the defeater [16] of an argument. We will be using the terms "defeat" and "attack" pretty much interchangeably. We say that a defeated argument is *reinstated* if its defeater gets defeated by an accepted argument.

There are generally two types of defeat relationship between arguments, *Rebuttal* and *Undercutting*. Rebutting defeat is a symmetric notion. Two arguments that have contradictory conclusions are each other's *rebutting defeater*. In this case neither of the arguments is outrightly defeated, hence, in some literatures they are assigned the status *d*efensible [28] or provisionally defeated [17]. On the other hand, an *undercutting defeater* is an argument that attacks the underlying reasoning of the argument under attack. Undercutting attacks are not necessarily symmetrical. An argument that is defeated by an undercutting attack of a justified argument is *outrightly-defeated* or *overruled*.

In general, accepted arguments in argumentation systems that are used for commonsense reasoning are subject to two general constraints:

1. An acceptable argument (including its conclusion) should meet our expectation in regard to the available information,
2. An acceptable argument should be justified within the logic of the employed argumentation system.

Constraints (1) and (2) imply that if our expectation says that an argument is acceptable then the argument either has no defeaters, or, from our body of knowledge we are able to construct other justified arguments that defeat all its defeaters, or, there are other arguments that independently support the same conclusion.

We offer two motivating examples to address the constraint that this requirement puts on the body of knowledge. In the first example we advocate the need to expand the notion of reinstatement of argument to allow arguments to be reinstated without defeating the defeater. In the second example we argue the need to allow asymmetrical-provisional-defeat relationship. We use the result of these examples to introduce the representation of defeasible rules that we advocate. We argue that our representation while simple, provides a more explanatory model of a *real world* situation. Simplicity of the rules allows an argumentation system meet one of the main objectives of argumentation reasoning, namely to provide an explanation in line with human reasoning [22,23,26]. We note in passing that since we have not yet introduced our own definition of an argument, in the following examples, we represent arguments as sequences of inferences from premises, standard in the literature. We also depict arguments as triangles where the base represent premises and top vertex conclusion of the argument. The attack relationship is shown by an arrow form attacking to the attacked argument.

Example 1. Imagine a real world system, involving, for instance, secretion of hormones and enzymes in presence of other hormones and enzymes. Let us assume that states of this system are all describable in terms of atoms a, b, c, v, x, y, z where a, b, c mnemonically stands for enzyme A, enzyme B, enzyme C is present in the system, and , v, x, y, z for hormone V, hormone X, hormone Y, and hormone Z is present in the system.

Tables 1(a) and 1(b) represent results of careful experimentation for two alternative scenarios. Each table has two sections: *Known Facts* and *Can be Believed*. For instance, the row one in Table 1(a) states that *if all we know is that the enzyme A is present then we are allowed to believe that hormone Z is present, as well.* Furthermore, the only difference between the two scenarios is that in scenario (a) all hormones are detectable whereas in scenario (b) the hormone Y is not detectable. Our question is: *Can we model both these scenarios in terms on an argumentation system?*

Furthermore, the knowledge *if all we know is that enzyme A is present then we are allowed to believe hormone Z is present too* is interpreted as *presence of enzyme A is the primary explanation for secretion of hormone Z* (Enzymes generally act as catalysts), and is represented in terms of defeasible inference rule $a \rightarrow z$. The second row of table 1(a) is interpreted as the presence of hormone Y acts as a suppressant for secretion of hormone Z. In argumentation terms, the presence of hormone Y is therefore interpreted as an undercutting defeater for the reasoning: from a and $a \rightarrow z$ to z.

The result of construction of such defeasible inference rules and their associated undercutting defeaters is given in Table 1(c). It can be shown that the observed system, as expressed in Table 1(a) can be modeled in terms of an argumentation system using the rules in table 1(c). An argumentation system carries as follows:

Table 1. (a) y: detectable. (b) y: not detectable. (c) Rules in 1(a) and 1(b).

(a)

Known Facts					Can be Believed				
a					a	z			
a	y				a	y			
b					b	y			
a	b				a	b	y		
a	b	x			a	b	x	z	
a	b	c			a	b	c	x	z
.	.	.	.	.	.	.	.	.	.

(b)

Known Facts					Can be Believed				
a					a	z			
b					b				
a	b				a	b			
a	b	x			a	b	x	z	
a	b	c			a	b	c	x	z
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

(c)

Visualization of Tables 1(a), 1(b) in terms of defeasible rules			
Table	Name	Rule	Defeater
Table 1(a)	d_1	$a \rightarrow z$	y
	d_2	$b \rightarrow y$	x
	d_3	$c \rightarrow x$	
Table 1(b)	d_1'	$a \rightarrow z$	b
	d_2'		
	d_3'	$c \rightarrow x$	

1. An argument is justified if it has no defeater, or, all its defeaters are defeated by justified arguments,
2. An argument that has a justified defeater is overruled,
3. Only the conclusions supported by justified arguments are justified.

In the second scenario we assume that the hormone Y is not detectable. The result of experimentation for this scenario is shown in Table 1(b). Based on Table 1(b), the presence of enzyme B now acts as the undercutting defeater for the rule $a \rightarrow z$. The construction of defeasible rules for Table 1(b) is given in Table 1(c). We would now like to ask the same central question *could we still model this system in terms of an argumentation system*?

The answer this time is far from obvious. For instance, if the contingent knowledge is $\{a, b, c\}$, i.e. the enzymes A, B, C are known to be present, are we allowed to believe in presence of hormone Z?

The arguments[3] in relation to the secretion of hormone Z are:

1. *enzyme A is present* represented as $arg_0 = \langle a \rangle$.
2. *enzyme A is present and since enzyme A is the reason for secretion of hormone Z, so hormone Z is present* represented as $arg_1 = \langle a,\ a \rightarrow z,\ z \rangle$.
3. *enzyme B is present* represented as $arg_2 = \langle b \rangle$.
4. *enzyme C is present and since enzyme C is the reason for secretion of hormone X, so hormone X is present* represented as $arg_3 = \langle c,\ c \rightarrow x,\ x \rangle$.

The only attack relationship is arg_2 undercutting arg_1 i.e. presence of enzyme B undercuts the reasoning $a \rightarrow z$. Therefore, arg_1 is defeated by arg_2 (fig. 1). Furthermore, since arg_2 has no defeater (note: arg_2 is an observation so cannot have any defeaters) arg_1 stays defeated. Yet, the table 1(b) indicates that despite the presence of enzyme B, if enzyme C is present we are allowed to believe that enzyme A results in secretion of hormone Z. As it can be seen in fig. 1, the problem lies in arg_3 being unable to reinstate arg_4. The only way to reinstate arg_4 is to defeat arg_2. But, as it is already noted

[3] An argument is represented as a sequence $\langle s_1,\ s_2, ...,\ s_n \rangle$ of statements where the last statement, s_n is the conclusion. The sequence $\langle s_n \rangle$ is an argument with an empty set of premises representing a single fact.

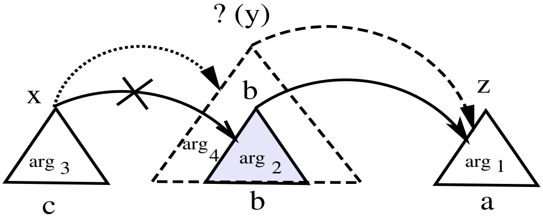


Fig. 1. Arguments interaction for the scenario ($\{a, b, c\}$ is given facts) in example 1

arg_2 is in essence an observation and cannot be defeated. It is as if there is a missing argument arg_4 as shown in figure 1 by dotted lines (arg_4 can be constructed from table 1(a)) where arg_4 attacks arg_1 and arg_3 reinstates arg_1 against this attack.

To fix this problem we could introduce another defeasible rule $(a, c) \to z$ to independently derive z. However, $(a, c) \to z$ is an artificial construct; the explanation for belief in z lies only in a. Hence, unless we introduce this rather artificial rule, our argumentation system will fall short of matching the real system. To sum up, we would like to allow arguments to be reinstated through context (not necessarily by attacking the attacker). We will also use the idea of missing arguments in our translation to Dung's Argumentation framework. ■

Our next example is based on one used in [16]. The idea is, under normal circumstances, an object that appears to be red can be taken to be red; however one has to be more cautious if she knows that the object in question is illuminated by a red light, whereby *red-lighting* acts as an undercutting defeater against the argument supporting *the object being red* based on its appearance. Contrary to what is taken for granted in the current argumentation systems, this example shows that undercutting attacks by acceptable arguments may result only in provisional defeat.

Example 2. In this example we argue that the common representation of this scenario in terms of an argumentation system, fig. 2(a), does not yield the expected outcome. However, an alternative representation, fig. 2(b), gives the expected outcome. We take the contingent knowledge and rule base to be $\{A_{red}, L_{red}\}$ and $\{A_{red} \to I_{red}\}$ where A stands for *appears*, I for *is* and L for *lighting*.

The argument for *object is red* in both depiction of this scenario, fig. 2(a) and 2(b), is $A_1 = \langle A_{red}, A_{red} \to I_{red}, I_{red} \rangle$. In the first representation, fig. 2(a), the argument

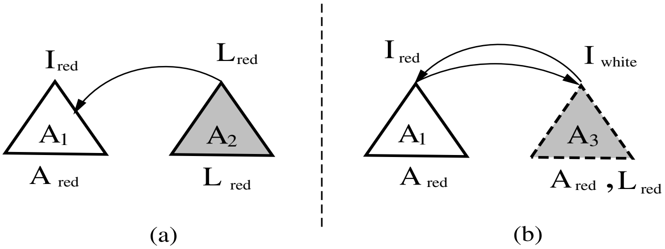


Fig. 2. Argumentation representation of the two alternative scenarios in example 2

A_1 is undercut by the argument $A_2 = \langle L_{red} \rangle$, and, consequently I_{red} gets status *overruled* [16]. Yet, the expected answer is I_{red}: *defensible* (*defensible* is also referred to as *provisionally defeated*) as object is either red or white.

In an alternative representation where rule $(A_{red} \wedge L_{red}) \rightarrow I_{white}$ (or alternatively $(A_{red} \wedge L_{red}) \rightarrow \neg I_{red}$) is added to the rule base, the resulting argument interaction, fig. 2(b) gives the expected outcome, i.e I_{red}: *defensible*. In fig.2(b) the two arguments A_1 and $A_3 = \langle A_{red}, L_{red}, (A_{red} \wedge L_{red}) \rightarrow I_{white}, I_{white} \rangle$ rebut each other leading to both arguments being *provisionally defeated*. The argument for *object being white* in fig. 2(b), shown by dotted lines, could again be viewed as a missing argument in the first depiction of this scenario. We therefore would like to allow undercutting attacks (or asymmetrical attacks in general) result in provisional defeat. ■

We take an inference rule to be like a black-box[4] with some *underlying explanation* similar to the notion of *conveyance* given in [13]. In every rule, antecedents are considered as the primary reason for belief in the consequent. In addition, there are other ancillary reasons that either strengthen or weaken a given rule.

In relation to how a rule works a justification function is provided that describes the conditions under which a rule gets activated. The justification function maps a given context (represented as a set of literals) into the operability space $\{0, 1/2, 1\}$, the values in question signaling, respectively, whether the rule is *acceptable right away*, is *provisionally defeated*, or is *outright defeated*.

As formal theories of argumentation get matured there is a growing interest to adapt these theories for modeling various forms of human reasoning. We believe our approach is in line with this goal. One approach is *modeling of the natural language argumentation schemes* [20,22,23]. This approach involves:

1. Characterization and classification of stereotypical patterns of reasoning. The characterization and classification of stereotypical patterns of reasoning is theorized in form of argumentation schemes [23].
2. The translation of formal arguments into natural language dialectic arguments [22].

In order to adapt formal argumentation theory to model argumentation schemes, the proposed approaches extend the current theory [1,29]. Amgoud and Cayrol [1] propose a preference-based argumentation framework that augments preferences among premises with the attack relationship in Dung's framework, while Wooldridge *et al.* [29] propose a hierarchical meta-logical argumentation framework. Our approach, too, is an extension of current argumentation theory with a central theme that each inference rule should have an underlying explanation. Explanations are a supposition in argumentation schemes [13] and a requirement for translation into natural language arguments. Argumentation schemes are a top down approach, while our approach is a bottom up approach.

3 Defeasible Reasoning System

By a defeasible reasoning system we mean a pair $\langle \mathcal{L}, \mathcal{R} \rangle$ where $\mathcal{L}$ is formal language, and $\mathcal{R}$ is a set of inference rules [6].

[4] A black-box view of inference rule allows for an element of intentionality in the rules [4] as longs as there is an underlying explanation or a notion of conveyance.

In the last section we stated that if $p \rightarrow q$ is an inference rule then our belief in p is the primary reason for our belief in q and the circumstances affecting our belief in applicability of $p \rightarrow q$ are the ancillary reasons.

Consider a rule d. If a context/circumstance do not affect d then d is applicable by default. However, if the circumstance affects the applicability of d then we should determine its effect. We represent each circumstance C_i as a set of literals. Suppose $C_1, C_2, ..., C_n$ are the circumstances that affect applicability of d. Now, not all literals in a circumstance C_i affect the applicability of d. Let for each circumstance C_i, its subset $J_i \subseteq C_i$, be the set of literals that affects rule d. Then $J = \bigcup_{i=1}^{n} J_i$ is the set of all literals that affects the applicability of rule d. We would therefore like to define a justification function for defeasible rule d by partitioning $\wp(J)$ into three equivalence classes (equivalence w.r.t. the degree of *acceptability* of the rule d). The degree of *acceptability* of the rule d is represented by values in the operability space $\{0,\ 1/2,\ 1\}$.

3.1 Defeasible Inference Rule

Accordingly, we assume every rule d is associated with three families of $\mathbf{T}_d, \mathbf{U}_d$, and $\mathbf{F}_d$ of sets of literals such that:

1. $\mathbf{T}_d$, $\mathbf{U}_d$, and $\mathbf{F}_d$ partition a set $\wp(\mathbf{J}_d)$ where $\mathbf{J}_d$ is called justification domain of d. This assumption says that the three families, $\mathbf{T}_d, \mathbf{U}_d$ and $\mathbf{F}_d$ jointly exhaust all possible "observable states" that determine applicability of rule d.
2. x and $\neg x$ are not both in $\mathbf{J}_d$.
3. if $a \in bd(d)$, then $a \notin \mathbf{J}_d$. The antecedent of a rule must already be believed for a rule to be fired. Therefore, there is no special need to have the antecedent in $\mathbf{J}_d$.

Definition 1. *Let d be a rule with the three associated sets,* $(\mathbf{T}_d, \mathbf{U}_d, \mathbf{F}_d)$[5]*, given above. The* **justification function** *of a rule d is a function* $\mathbf{H}_d : \wp(\mathcal{L}) \longrightarrow \{0, 1/2, 1\}$ *where*

$$\mathbf{H}_d(X) = \begin{cases} 1 & \textit{if} \quad \mathbf{R}_d(X) = \emptyset \textit{ or } \mathbf{R}_d(X) \in \mathbf{T}_d \\ 0 & \textit{if} \quad \mathbf{R}_d(X) \in \mathbf{F}_d \\ 1/2 & \textit{if} \quad \mathbf{R}_d(X) \in \mathbf{U}_d \end{cases}$$

and the **relevance factor** *of X w.r.t. the rule d, denoted* $R_d(X)$*, is the largest subset of X that is also a member of* $(\mathbf{T}_d \cup \mathbf{U}_d \cup \mathbf{F}_d)$ *i.e.* $R_d(X) = X \cap \mathbf{J}_d$*.*

The parameter X is intended to represent a circumstance. A tabular representation of justification function is called *justification matrix* of the rule. It can be seen that if a context has no relevance to the applicability of a rule then the rule is applicable. In other words, if $R_d(X) = \emptyset$ then $\mathbf{H}_d(X) = 1$. We next define what it means to say whether a rule is accepted or defeated, as well as, classifying rules based on the justification function.

Definition 2. *Let* $A \subseteq \mathcal{L}$ *be a set of sentences, and* $d \in \mathcal{R}$ *a rule.*
1. A is said to
(a) ***accept*** *d iff* $\mathbf{H}_d(A) = 1$.
(b) ***outright defeat*** *d iff* $\mathbf{H}_d(A) = 0$.
(c) ***provisionally defeat*** *d iff* $\mathbf{H}_d(A) = 1/2$.
(d) ***conclusively defeat*** *d iff* $\mathbf{H}_d(A) = 0$ *and if* $A \subseteq B$ *then* $\mathbf{H}_d(B) = 0$.

[5] While determining $\mathbf{T}_d$, $\mathbf{U}_d$, and $\mathbf{F}_d$ require some effort, it is no more arduous than assigning strength to arguments. However, in case of *Justification Function* there is the advantage of having a point of reference, i.e. the *circumstance*.

2. *A rule $d : a_1, a_2, ..., a_n \rightarrow a$ in $\mathcal{R}$ is a **normal** rule iff $\{\neg a\}$ conclusively defeats d.*
3. *A normal rule $d \in \mathcal{R}$ is a **default** rule iff $\mathbf{U}_d = \emptyset$, and, every $A \in \mathbf{F}_d$ conclusively defeats d. In addition we call $\mathbf{F}^k_d = \{A \mid A \textit{ is the minimal set in } \mathbf{F}_d\}$ the **justification-base** of default rule.*
4. *A rule $d \in \mathcal{R}$ is an **indefeasible** rule iff $\mathbf{F}_d = \mathbf{T}_d = \mathbf{U}_d = \emptyset$.*

The terms outright-defeat and provisional-defeat are adopted from [17,18]. An indefeasible rule d is always acceptable, i.e. $\mathbf{H}_d(X) = 1$ is always true. This makes indefeasible rules synonymous to the material conditionals or the necessary knowledge in other argumentation systems [10,25].

Conclusive defeat is an important property that is implicitly assumed in other argumentation systems. In a *Conclusive defeater* the defeat condition is context independent viz. a rule is always inapplicable in presence of a conclusive defeater.

Conclusive defeater is used to define normal rules. A normal rule is always defeated in light of contrary evidence to its conclusion. Normal rules allow: 1) implicitly capture *rebuttal attacks*, and 2) ensure no two arguments with contradictory conclusions are simultaneously justified. Default rules above are a special class of normal rules. In default rules all defeat conditions are conclusive defeat conditions. It can be argued that within our semantics default rules would be equivalent to Reiter's default rules.

Observation 1. *Given any set of sentences $A \subseteq \mathcal{L}$ and rule $d \in \mathcal{R}$*

1. *$\mathbf{H}_d(A)$ has one and only one value.*
2. *A normal rule d with $\mathbf{U}_d = \emptyset$ is a default rule iff no A in $\mathbf{F}_d$ is a subset of B in $\mathbf{T}_d$.*
3. *If d is a default rule and $\mathbf{F}^k_d$ its justification-base then:*
   (a) for every $A \in \mathbf{F}^k_d$, A is either a singleton or $A = \{a \mid a \in B \text{ and } B \in \mathbf{T}_d\}$.
   (b) for $X \subseteq \mathcal{L}$, $\mathbf{H}_d(X) = 0$ iff $\exists A \in \mathbf{F}^k_d$ s.t. $A \subseteq X$.

The obs. 1.1 is the first step to ensure that the proposed argumentation theory is well-defined. The obs. 1.2 and 1.3 draw a parallel between default rules (as defined in this work) and conventional default rules e.g. Reiter default rules. Obs. 1.3 states that if a context includes any member of $\mathbf{F}^k_d$ then rule d is in applicable. Hence, one can say that members of $\mathbf{F}^k_d$ are similar to negation of grounded justification assumptions in Reiter default rules. For instance, if $d : b \rightarrow f$ and $\mathbf{F}^k_d = \{\neg f, p, e\}$, then the equivalent Reiter default rule is $d^r = \frac{b:f,\neg p,\neg e}{f}$ (as shown in example 3).

Example 3. (the standard example in non-monotonic reasoning) *A Bird can usually fly unless it is a penguin or an emu.* Let literals $b,\ f,\ p,\ e$ mnemonically stand for Tweety is a bird, can fly, is a penguin, is an emu. The example can be represented as a defeasible rule d: $b \rightarrow f$ is a default rule with the following justification function.

$$\begin{aligned} &\mathbf{T}_d = \mathbf{U}_d = \emptyset \\ &\mathbf{F}_d = \{\{p\},\{e\},\{\neg f\},\{p,e\},\{p,\neg f\},\{e,\neg f\},\{p,e,\neg f\}\} \\ &\mathbf{F}^k_d = \{\{p\},\{e\},\{\neg f\}\}. \end{aligned}$$ ■

Example 4. Sam's friends usually like ethnic foods unless they are hot and spicy. Though, Thai green curry is hot and spicy, they still like it. Let the literals $sf,\ ef,\ hs,\ tg$ mnemonically represent Sam's friends, like ethnic food, food is hot and spicy, and food is Thai green curry. *Sam's friends usually like ethnic foods* could therefore be represented as, d: $sf \rightarrow ef$ with the justification function:

$\mathbf{U}_d = \emptyset \quad \mathbf{T}_d = \{\{tg\}, \{hs, tg\}\}$
$\mathbf{F}_d = \{\{hs\}, \{\neg ef\}, \{\neg ef, tg\}, \{\neg ef, tg, hs\}\}.$ ■

Example 5. The behavior of the Table 1(b), to the extent it is specified, can be captured by assuming two inference rules (d_1 is a normal rule and d_2 is a default rule):

$d_1 : a \rightarrow z$, and
$d_2 : c \rightarrow x$,

where the respective justification functions (see Table 2) are determined by:

$\mathbf{T}_{d1} = \{\{x\}, \{x, b\}\}$
$\mathbf{U}_{d1} = \emptyset$
$\mathbf{F}_{d1} = \{\{b\}, \{\neg z\}, \{\neg z, b\},$
$\{\neg z, x\}, \{\neg z, x, b\}\}$
$\mathbf{T}_{d2} = \emptyset$
$\mathbf{U}_{d2} = \emptyset$, and
$\mathbf{F}_{d2} = \{\{v\}, \{\neg x\}, \{v, \neg x\}\}.$

Tables 2. The justification matrices of inference rules for Table 1(b)

$d_1 : a \rightarrow z$		
$\mathbf{T}_{d1}$ (1)	$\mathbf{U}_{d1}$ (1/2)	$\mathbf{F}_{d1}$ (0)
x		b
x, b		$\neg z$
		$\neg z, b$
		$\neg z, x$
		$\neg z, x, b$

$d_2 : c \rightarrow x$		
$\mathbf{T}_{d2}$ (1)	$\mathbf{U}_{d2}$ (1/2)	$\mathbf{F}_{d2}$ (0)
		v
		$\neg x$
		$v, \neg x$

■

3.2 Argument

We define an argument by a set of contingent facts, a set of inference rules and conclusion of the argument. This definition of an argument is in line with those given in [25] and [27]. Though, unlike the latter we do not include the length and the size of an argument as its properties. The size of an argument might indicate its strength [26], but, our position is that information regarding strength of an argument should be within its inference rules.

Definition 3. *Let $D \subseteq \mathcal{R},\ A \subseteq \mathcal{L}$ and $a \in \mathcal{L}$, An argument $\mathring{A}$ is a tuple $\langle A, D, a\rangle$ such that there exists a sequence of rules $d_1, \ldots, d_m \in D$ where:*

1. *$a = hd(d_m)$, and*
2. *$\forall d_{i,\ 0<i\leq m}$, either*
   (a) $bd(d_i) = \emptyset$, or
   (b) $\forall a_j \in bd(d_i)$, either $a_j \in A$ or there exists $d_{k,\ 0<k<j}$ such that $a_j = hd(d_k)$.
3. *No proper sub-sequence of $A' \subset A$ and $D' \subset D$ satisfy the two conditions above.*

We denote A, D and a by $A_{\mathring{A}}, D_{\mathring{A}}, a_{\mathring{A}}$; and call $A_{\mathring{A}}$ the evidence and $a_{\mathring{A}}$ the conclusion of the argument. Furthermore, we say an argument $\mathring{A}_2 = \langle A_2, D_2, a_2\rangle$ is a *subargument* of $\mathring{A}_1 = \langle A_1, D_1, a_1\rangle$, denoted by $\mathring{A}_2 \sqsubseteq \mathring{A}_1$, if and only if $D_2 \subseteq D_1$.

In our system, arguments interact with other arguments indirectly through context. Arguments create the context in which other arguments are accepted or rejected. The natural contribution of an argument to a context is its conclusion. On the other hand, when we accept an argument we implicitly accept all its subarguments. Therefore, in a set of arguments, the effective contribution of an argument to context is the conclusions of all its subarguments.

The set $Cn(\text{Å}) = \{x \in \mathcal{L} \mid x = a_{\text{Å}_i}, \text{Å}_i \sqsubseteq \text{Å}\}$ is called the *consequences* of the argument Å. If $\mathcal{A}$ is a set of arguments then $Cn(\mathcal{A}) = \bigcup_i Cn(\text{Å}_i)$ where $\text{Å}_i \in \mathcal{A}$.

We are now in a position to extend the concept of justification function to that of defeasible arguments. The justification function of an argument is defined by applying the *weakest link principle* to its inference rules.

Definition 4. *Given a defeasible argument* $\mathring{A} = \langle A, D, a\rangle$*, its justification function* $G_{\mathring{A}} : \wp(\mathcal{L}) \longrightarrow \{0, 1/2, 1\}$ *is defined as:* $G_{\mathring{A}}(X) = \min_{d \in D}(\mathbf{H}_d(X))$.

Furthermore, since we deal with argument sets we need to have a justification function *w.r.t.* a set of arguments. Moreover, it can be seen that the meaning of $\mathbf{H}_{\text{Å}}(X)$ is preserved whether X is a set of sentences or consequences of a set of arguments.

Definition 5. *Let* $\mathring{A}$ *be an argument and* $\mathcal{X} \subseteq \mathcal{A}$ *a set of arguments. The (induced) justification function of an argument* $\mathring{A}$ *is defined as* $\mathbf{H}_{\mathring{A}}(\mathcal{X}) : \wp(\mathcal{A}) \longrightarrow \{0, 1/2, 1\}$, $\mathbf{H}_{\mathring{A}}(\mathcal{X}) = \mathbf{G}_{\mathring{A}}(Cn(\mathcal{X}))$ *where* $\mathbf{G}_{\mathring{A}}$ *is the justification function of* $\mathring{A}$*.*

Observation 2. *Given a defeasible reasoning system* ($\mathcal{L}$, $\mathcal{R}$)*,* $X \subseteq \mathcal{L}$*,* $\mathcal{A}$ *a set of arguments and* $\mathring{A} = \langle A, D, a\rangle$ *an argument in* ($\mathcal{L}$, $\mathcal{R}$)*:*

1. $\mathbf{G}_{\mathring{A}}(X)$*, and* $\mathbf{H}_{\mathring{A}}(X)$ *have one and only one value.*
2. *if* $B \subseteq X = Cn(\mathcal{A})$ *conclusively defeats* $d \in D$ *then* $\mathbf{H}_{\mathring{A}}(\mathcal{A}) = \mathbf{G}_{\mathring{A}}(X) = 0$.

4 Defeat and Reinstatement Relationships

The defeasiblity of arguments is captured by *Defeat Relationship* between arguments. From presented defeat relationships, we are interested in undercutting attacks and rebuttal attacks. We capture rebuttal attacks through defeasible property of normal inference rules, without explicitly defining rebuttal attacks.

Unlike most argumentation systems where defeat is a direct binary relationship between individual arguments, in this system a group of arguments can cause or remove the defeat-condition for an argument, indirectly, via context. This property makes defeat a binary relationship between a group of arguments and an argument.

The phenomenon of separate arguments with same conclusion reenforcing each other is called accrual of arguments. Whether accrual of arguments is a valid argumentation concept or not is debatable [18]. Nonetheless, since our defeat and reinstatement relationship is between a group of arguments and an argument the intended meaning of accrual of arguments [26] can be easily represented in this model of defeat relationship.

In a set of arguments $\mathcal{A}$, context is set by consequences (conclusions of all subarguments) of all arguments in $\mathcal{A}$. In order to show an argument set $\mathcal{A}_c$ attacks an argument Å in $\mathcal{A}$, we need to establish given an initial context $Cn(\mathcal{A}')$ where $\mathcal{A}' \subset \mathcal{A}$, addition of $Cn(\mathcal{A}_c)$ results in Å being defeated. The notion of defeat is connected to a decrease in degree of acceptability of Å that is a decrease in $\mathbf{H}_{\text{Å}}(\mathcal{A})$. Accordingly, if $\mathbf{H}_{\text{Å}}(\mathcal{A}')$ is reduced to 0 it is said $\mathcal{A}_c$ outrightly defeats Å, and, if reduced to 1/2 provisionally defeats Å. In the same token, in order to reinstate Å, $\mathcal{A}_c$ (or $\{\text{Å}_1\}$) has to increase the degree of acceptability of Å.

An example of attack and reinstatement is shown in Fig. 3. The large arrows from arguments to context show contribution of arguments (their consequences) to the context. Let assume an initial scenario where $\mathcal{A}' = \{\mathring{A}, \mathring{A}_1\}$, and Å is acceptable *w.r.t.* $\mathcal{A}'$. If we add $\mathring{A}_2$ to $\mathcal{A}'$ a portion of $Cn(\mathcal{A}_{c1}) \subset Cn(\mathcal{A}')$ where $\mathcal{A}_{c1} = \{\mathring{A}_1, \mathring{A}_2\}$ attacks Å (shown by a circle). Yet, if add $\mathring{A}_3$ to the mix, Å becomes acceptable *w.r.t.* the new argument set $\mathcal{A}$. It is as if a portion of context of $\mathcal{A}_{c2}$ = $\{\mathring{A}_2, \mathring{A}_3\}$ reinstates Å against $\mathcal{A}_{c1}$.

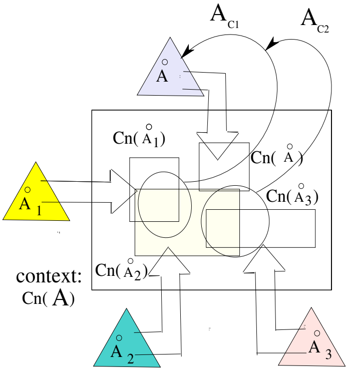


Fig. 3. A schematic of arguments interaction

Definition 6. *Let $\mathcal{A}$, $\mathcal{A}_c = \{\mathring{A}_1, \mathring{A}_2, ..., \mathring{A}_n\} \subseteq \mathcal{A}$ be sets of arguments, and $\mathring{A} \in \mathcal{A}$ an argument. We will say that the argument set $\mathcal{A}_c$ defeats the argument Å in $\mathcal{A}$*

1. **outright** *iff $\exists \mathcal{A}' \subseteq \mathcal{A}$ s.t. $\mathcal{A}_c \subseteq \mathcal{A}'$ and $\mathbf{H}_{\mathring{A}}(\mathcal{A}') = 0$, and $\mathcal{A}_c$ is a maximal subset of $\mathcal{A}'$ where $\mathbf{H}_{\mathring{A}}(\mathcal{A}' \setminus \{\mathring{A}_j\}) \neq 0$ for all $\mathring{A}_j \in \mathcal{A}_c$,*
2. **provisionally** *iff $\exists \mathcal{A}' \subseteq \mathcal{A}$ s.t. $\mathcal{A}_c \subseteq \mathcal{A}'$ and $\mathbf{H}_{\mathring{A}}(\mathcal{A}') = 1/2$, and $\mathcal{A}_c$ is a maximal subset of $\mathcal{A}'$ where $\mathbf{H}_{\mathring{A}}(\mathcal{A}' \setminus \{\mathring{A}_j\}) = 1$ for all $\mathring{A}_j \in \mathcal{A}_c$.*
3. *Furthermore, we will call:*
   (a) *$\mathcal{A}_c$ a* **defeat-scenario** *for Å in $\mathcal{A}$.*
   (b) *The argument set $\mathcal{A}' \setminus \mathcal{A}_c$ the* **defeat-context** *of the defeat relationship.*
   (c) *A defeat-scenario $\mathcal{A}_c$ is a* **conclusive-defeater** *of Å in $\mathcal{A}$ iff for every $\mathcal{A}'' \subseteq \mathcal{A}$ if $\mathcal{A}_c \subseteq \mathcal{A}''$ then $\mathbf{H}_{\mathring{A}}(\mathcal{A}'') = 0$. If $\mathcal{A}_c$ is conclusive defeater in any arbitrary $\mathcal{A}$ then it is called* **TConclusive-defeater** *of Å* [6] *. If $\mathcal{A}_c$ outrightly defeats Å and is not a conclusive defeater, we called it a* **non-conclusive-defeater**.

In definition above, the condition $\mathbf{H}_{\mathring{A}}(\mathcal{A}') = 0$ might have been enough to say $\mathcal{A}'$ defeats Å. However there is a section of $\mathcal{A}'$ that is responsible for defeat and there is a section of $\mathcal{A}'$ that acts as the context for a particular defeat scenario. We name the first part the defeater and the second part the context for defeat. Now we are in a position to show the following results.

Observation 3. *Let $\mathcal{A}$, $\mathcal{A}'$ be two argument sets, $\mathcal{A} \subseteq \mathcal{A}'$, Å, $\mathring{A}' \in \mathcal{A}$, Å a subargument of $\mathring{A}'$, and $\mathcal{A}_c$ a defeat-scenario for Å in $\mathcal{A}$ then:*

1. *$\mathcal{A}_c$ is also a defeat-scenario for Å in $\mathcal{A}'$.*
2. *if $\mathcal{A}_c$ is a conclusive-defeater of Å then $\mathcal{A}_c$ is also a conclusive defeater of $\mathring{A}'$ in $\mathcal{A}$.*
3. *If D, the set of all rules of arguments in $\mathcal{A}$, is comprised of only indefeasible and default rules then all defeaters of arguments in $\mathcal{A}$ are TConclusive defeaters where the context of defeat is $\emptyset$. In addition, if both $\mathcal{A}_c$ is the minimal set of arguments where $F \in \mathbf{F}^k_d$, $F \subseteq Cn(\mathcal{A}_c)$, and, F is singleton set, then $\mathcal{A}_c$ is singleton set.*

[6] 'T' for True in every argument set or universal defeater.

4. *If $Cn(\mathcal{A}_c)$ conclusively defeats at least one defeasible rule in argument Å then $\mathcal{A}_c$ is a conclusive defeater of Å (though, the reverse is not necessarily true.)*

Observations 3.3 and 3.4 are continuation of our attempt to draw a parallel between argumentation systems that are built upon conventional default rules and this argumentation system. In most argumentation systems Defeat relationship is a static relationship. Observation 3.1 states that the defeat relationship between a defeat scenario $\mathcal{A}_c$ and an argument Å is static. However, this claim is contrary to our original claim that defeat relationship is subject to context. The reason for this apparent conflict is we want to keep the proposed argumentation system in line with conventional argumentation theory. In order to account for the influence of context over defeat relationship we define a reinstatement by context relationship.

In current argumentation systems an argument is reinstated only when its defeater is defeated. In our system arguments can reinstate other arguments by context without defeating their defeaters. The parts 1(a) and 1(b) in the following definition are the conventional method of reinstatement whereas parts 1(b) and 2(b) are exclusive to our system representing *reinstatement by context*. It can be seen that in case of conclusive defeat scenarios there is no reinstatement by context.

Definition 7. *Let $\mathcal{A}$, $\mathcal{A}_{c1}$=$\{Å_{11}, Å_{12}, \ldots, Å_{1n}\} \subseteq \mathcal{A}$ and $\mathcal{A}_{c2}$=$\{Å_{21}, Å_{22}, \ldots, Å_{2m}\} \subseteq \mathcal{A}$ be three argument sets, argument Å $\in \mathcal{A}$, and $\mathcal{A}_{c2}$ be a defeat scenario for Å in $\mathcal{A}$.*

1. $\mathcal{A}_{c1}$ **outrightly reinstates** *Å in $\mathcal{A}$ against $\mathcal{A}_{c2}$ iff either:*
   (a) $\exists Å_i \in \mathcal{A}_{c2}$ such that $\mathcal{A}_{c1}$ is an outright-defeat-scenario for $Å_i$ in $\mathcal{A}$, or
   (b) both
      i. $\exists \mathcal{A}' \subseteq \mathcal{A}$ such that $\mathcal{A}_{c1}$, $\mathcal{A}_{c2} \subseteq \mathcal{A}'$; $\mathbf{H}_{Å}(\mathcal{A}') = 1$ and
      ii. $\mathcal{A}' \setminus (\mathcal{A}_{c2} \cup \{Å_j\})$ is a defeat-context for $\mathcal{A}_{c2}$ defeating Å for all $Å_j \in \mathcal{A}_{c1}$.
2. $\mathcal{A}_{c1}$ **provisionally reinstates** *Å in $\mathcal{A}$ against $\mathcal{A}_{c2}$ iff: $\mathcal{A}_c$ is not an outright-defeat-scenario for any argument $Å_i \in \mathcal{A}_{c2}$ and either:*
   (a) $\exists Å_i \in \mathcal{A}_{c2}$ s.t. $\mathcal{A}_{c1}$ is a provisional-defeat-scenario for $Å_i \in \mathcal{A}_{c2}$ in $\mathcal{A}$, or
   (b) both
      i. $\exists \mathcal{A}' \subseteq \mathcal{A}$ such that $\mathcal{A}_{c2}, \mathcal{A}_{c1} \subseteq \mathcal{A}'$; $\mathbf{H}_{Å}(\mathcal{A}') = 1/2$, and
      ii. $\mathcal{A}' \setminus (\mathcal{A}_{c2} \cup \{Å_j\})$ is a defeat-context for $\mathcal{A}_{c2}$ outright-defeating Å, $\forall Å_j \in \mathcal{A}_{c1}$.

We define an argumentation theory comprised of a set of arguments and all types of attack and reinstatement relationships given above. In order to interpret this argumentation theory into a Dung's Argumentation framework, we require to identify our system with all classes of attack and reinstatement relationships. The reason for this requirement is given in the next section.

Definition 8. *Given a defeasible reasoning system ($\mathcal{L}, \mathcal{R}$), an* **argumentation theory** *is a tuple AT = ($\mathcal{A}, \otimes, \oplus$) where $\mathcal{A}$ is a set of arguments constructed in ($\mathcal{L}, \mathcal{R}$), and $\otimes, \oplus$ are the defeat and reinstatement relationships between a set of arguments and an argument as defined above. Furthermore, given (A, D) in ($\mathcal{L}, \mathcal{R}$), if $\mathcal{A}$ is all the possible arguments that can be constructed in (A, D) then the argumentation theory AT = ($\mathcal{A}, \otimes, \oplus$) is called an* **induced argumentation theory** *from (A, D). In addition, AT is called* **context insensitive** *if all defeat scenarios $\mathcal{A}_c \subseteq \mathcal{A}$ are conclusive defeaters.*

Observation 4. *Let $AT = (\mathcal{A}, \otimes, \oplus)$ be an argumentation theory and D the set of all rules in all arguments in $\mathcal{A}$. Then, if D is comprised of only indefeasible and default rules, AT will be context insensitive.*

The next example is to show attack and reinstatement relations at work. This example is a modified version of example given in [25,10]. This example also shows the role of primary and ancillary reasons in an inference rule. We continue this example in example 7 where we show which arguments are justified and which are overruled.

Example 6. Let $\mathcal{L} = \{a, \neg a, s, \neg s, r, \neg r, c, \neg c, e, \neg e\}$ with the following readings:

a : *Tom is a mature adult*; s : *Tom is a student*;
r : *Tom has very rich parents*; c : *Tom has a car*; and e : *Tom is employed.*

All the inference rules are normal, constituting

$$\mathcal{R} = \{d_1 : s \to \neg a,\ d_2 : s \to \neg e,\ d_3 : a \to \neg s,\ d_4 : a \to c,\ d_5 : a \to e,\ d_6 : c \to e,\ d_7 : \neg e \to \neg c,\ d_8 : r \to c,\ d_9 : e \to c\}.$$

The rules d_1, d_3, d_8, and d_9 are all default rules with no other defeater except the negation of their consequents. The rule d_7 is also a default rule, but having an additional conclusive defeater, namely the scenario $\{r\}$. The justification matrices of rest of the rules are provided in the tables below.

Table 3. The justification matrices of inference rules d_2, d_4, d_5, d_6 in example 6

d_2: $s \to \neg e$		
1	1/2	0
$\{r\}$, $\{c, r\}$	$\{c\}$, $\{a, r\}$, $\{a, c, r\}$	$\{a\}$, $\{a, c\}$

d_4: $a \to c$		
1	1/2	0
$\{r\}$, $\{e\}$, $\{s, r\}$,$\{s, e\}$, $\{r, e\}$, $\{s, r, a\}$	$\{s\}$	

d_5: $a \to e$		
1	1/2	0
$\{s\}$,$\{c\}$, $\{r\}$, $\{c, s\}$, $\{c, r\}$	$\{s, r\}$, $\{s, r, c\}$	

d_6: $c \to e$		
1	1/2	0
$\{a\}$, $\{r\}$, $\{s, a\}$, $\{a, r\}$	$\{s\}$, $\{s, r\}$, $\{s, a, r\}$	

Let $\mathring{A}_1 = \langle\{s\}, \emptyset, s\rangle$, $\mathring{A}_2 = \langle\{r\}, \emptyset, r\rangle$, $\mathring{A}_3 = \langle\{a\}, \{d_5\}, e\rangle$, and $\mathring{A}_4 = \langle\{c\}, \{d_6\}, e\rangle$ be arguments in an induced argumentation theory derived from $(\mathcal{L}, \mathcal{R})$. Then $\mathcal{A}_{c1} = \{\mathring{A}_1, \mathring{A}_2\}$ is a provisional-defeat scenario for $\mathring{A}_3$ but not for $\mathring{A}_4$ and $\mathcal{A}_{c2} = \{\mathring{A}_1\}$ is a provisional-defeat scenario for $\mathring{A}_4$ but not for $\mathring{A}_3$. ■

5 Semantics

The semantics of a argumentation system is determined by the rules of interaction between arguments. There are a number of approaches to provide the semantics of argumentation systems [28], e.g., assigning status to arguments [17,18], defining the acceptable set(s) of arguments [9,6] and using dialectic argumentation trees [8]. While there are minor differences, the approaches are driven by the same intuition where a definition in one can be an observation in another [9,21]. For instance in [9] it is shown the set of justified arguments in [16] is equivalent to the grounded-extension in [9]. Dung's Argumentation Framework is used as basis in a number of argumentation systems. We adopt Dung's framework in order to give an anatomical picture of this system's behavior. We first translate proposed argumentation theory into a Dung argumentation framework and then apply Dung's semantics to the interpreted arguments.

Definition 9 (Dung's Argumentation Framework (Dung 1995))

1. *A Dung argumentation framework is a pair* $AF = \langle AR, R\rangle$ *where* AR *is a set of arguments and R, the attack relationship, is a binary relation between arguments i.e.* $R \subseteq AR \times AR$.
2. *An argument* $a \in AR$ *is attacked by a set of arguments* S *iff* $\exists b \in S$ *s.t.* bRa.
3. *An argument* $a \in AR$ *is acceptable with respect to a set* S *of arguments iff* $\forall b \in AR$*: if* b R a *then* S *attacks* b.
4. *A set* S *of arguments is said to be conflict-free iff* $\forall a, b \in S$ *there is no* aRb.
5. *A conflict-free set of arguments S is admissible iff each argument in* S *is acceptable with respect to* S.
6. *A preferred extension of an argumentation framework AF is a maximal (w.r.t. set inclusion) admissible set of* AF.

In general, systems that are built upon Dung's system (with exception of [1]) deal with conclusive defeats, and reinstatement of arguments is by defeating of their defeaters, and the provisional-defeat is an interpretation of multiple preferred extensions.

Figure 4. shows the underlying idea in translation to Dung's AF. Let AT be an argumentation theory where $\mathring{A}$ attacks $\mathring{A}_2$ and $\mathring{A}_3$; and $\mathring{A}_1$ reinstates $\mathring{A}_3$ against $\mathring{A}$. For simplicity we use arguments instead of argument sets. If we translate AT by 1-to-1 mapping between arguments in AT to arguments in AF, we get $\mathring{A}_1$ reinstating both $\mathring{A}_3$ and $\mathring{A}_2$, fig. 4(a) ($\mathring{A}_1$ defending $\mathring{A}_3$ against $\mathring{A}$, leads to defending $\mathring{A}_2$ as well). But, this translation is incorrect since $\mathring{A}_1$ should only reinstate $\mathring{A}_3$. To obtain the desired translation we borrow the idea of *missing arguments* from examples 1 and 2. We assume there is an imaginary argument $\mathring{A}_4$, $\mathring{A} \sqsubset \mathring{A}_4$ (shown by dotted lines in fig. 4(b)). It is $\mathring{A}_4$ that attacks $\mathring{A}_3$ and $\mathring{A}_1$ reinstates $\mathring{A}_3$ by attacking $\mathring{A}_4$. This time a 1-to-1 translation to a Dung's AF would yield $\mathring{A}_3$ reinstated and $\mathring{A}_2$ defeated,fig. 4(b). This result is the intended result. Hence, we need to distinguish between various types of defeats and reinstatement in our translation.

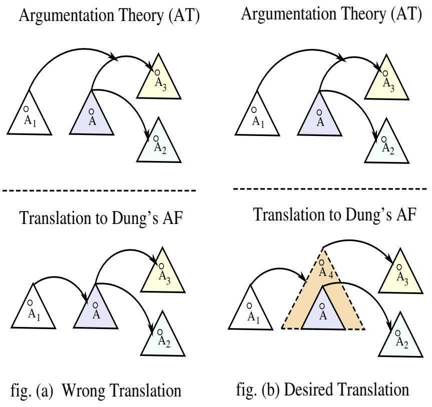


fig. (a) Wrong Translation fig. (b) Desired Translation

Fig. 4. Translation to Dung's Argumentation Framework

Definition 10 (Translation). [7] *Let $AF = \langle AR, R\rangle$ be a Dung argumentation framework, and $AT = (\mathcal{A}, \otimes, \oplus)$ an argumentation theory in $\langle \mathcal{L}, \mathcal{R}\rangle$. AF is a translation of AT iff:*

1. *there is a surjective function $\mathbf{M1}$: $\wp(\mathcal{A}) \longrightarrow AR$ such that:*
   (a) *For every $\mathring{A} \in \mathcal{A}$ there is an $\alpha \in AR$ for $\{\mathring{A}\}$,*
   (b) *For every $\mathcal{A}_c$ where $\mathcal{A}_c \subseteq \mathcal{A}$ is a defeat-scenario or reinstatement-scenario for $\mathring{A} \in \mathcal{A}$ there is an $\alpha \in AR$ (if $\mathcal{A}_c$ is singleton then α is the same α in 1(a)),*
   (c) *if $\mathcal{A}_c$ is a reinstatement by context scenario or a non-conclusive defeat-scenario there is one additional $\alpha^i_j \in AR$ for each defeat or reinstatement case (indexes i and j denote $\mathcal{A}_c$ and individual case j).*
2. *Given all $\alpha, \alpha^i_j \in AR$ as specified above then there is a surjective funcion $\mathbf{M2}$: $\wp(\mathcal{A}) \times \wp(\mathcal{A}) \longrightarrow R$ where R is the relation R in AF such that:*
   (a) *if $\mathcal{A}_{c1}$ is an outright defeat scenario for an argument $\mathring{A}_k$, and β standing for any β mapped under $\mathbf{M1}$ for $\mathring{A}_k$ or any $\mathcal{A}_{ck}$ that $\mathring{A}_k$ is a member of then:*
      i. *if $\mathcal{A}_{c1}$ is a conclusive defeat scenario, then $\alpha R \beta$ where α is the mapped $\alpha \in AR$ for $\mathcal{A}_{c1}$,*
      ii. *if $\mathcal{A}_{c1}$ is a non-conclusive defeat scenario then $\alpha^i_j R\beta$ where α^i_j is the mapped α^i_j for the corresponding $\mathcal{A}_{c1}$ defeating $\mathring{A}_k$,*
      iii. *if $\mathcal{A}_{c1}$ is a provisional defeat scenario for an argument $\mathring{A}_k \in \mathcal{A}_{c2}$ then $\alpha^i_j R\beta$ and $\beta R\alpha^i_j$ where α^i_j is the mapped α^i_j for the corresponding $\mathcal{A}_{c1}$ provisionally defeating $\mathring{A}_k$.*
   (b) *if $\mathcal{A}_{c1}$ is a reinstatement by context scenario for an argument $\mathring{A}_k$ against the defeat scenario $\mathcal{A}_{c2}$, and α^i_j, β^i_j are the mapped arguments in AR for $\mathcal{A}_{c1}$ and $\mathcal{A}_{c2}$ for this defeat-reinstatement scenario then:*
      i. *if $\mathcal{A}_{c1}$ is an outright-reinstatement scenario, then $\alpha^i_j R\beta^i_j$,*
      ii. *if $\mathcal{A}_{c1}$ is a provisional-reinstatement scenario then $\alpha^i_j R\beta^i_j$ and $\beta^i_j R\alpha^i_j$.*

To define operability space we implicitly adopted values of status of arguments in [17,18,28]. Assigning status to arguments is part of semantics in [17,18]. We defined semantics based on Dung-Argumentation semantics. We are yet to interpret semantics to values in operability space. The relation between BDKT-argumentation [6] semantics, and Pollock-argumentation [17,18] semantics is given in [21]. BDKT-argumentation semantics closely follows semantics given in [9].

Definition 11. *Let $AT = (\mathcal{A}, \otimes, \oplus)$ be an argumentation theory in a defeasible reasoning system $\langle \mathcal{L}, \mathcal{R}\rangle$ and $AF = \langle AR, R\rangle$ its interpreted Dung-Argumentation framework, the status assignment function* [8] $\mathbf{E}$: $\mathcal{A} \longrightarrow \{0, 1/2, 1\}$ *is:*

[7] In relation to odd-defeat-cycles we agree with [5] differing from Dung's semantics. Our intuitive reason is arguments in odd-defeat-cycles indirectly attack themselves making them a class of self-defeating arguments. In order to comply with [5] we need to change *Conclusive Defeat* to *TConclusive Defeat* in our translation. However, for our purpose the difference is not important as semantics is the same except when there are odd-defeat-cycles.

[8] Instead of *preferred semantics* we could have given admissible semantics by saying $\mathbf{E}(\mathring{A}) = 1$ *if X_p is in an admissible-extension where no attacker of X_p is in any admissible-extensions*, and $\mathbf{E}(\mathring{A}) = 1/2$ *if at least one attacker of X_p is also in a admissible-extensions.*

$$\mathbf{E}(\mathring{A}) = \begin{cases} 1 & \textit{if } X_p \textit{ is in all preferred-extensions in AF} \\ 1/2 & \textit{if } X_p \textit{ is in at least one, but not all the preferred-extensions in AF} \\ 0 & \textit{if } X_p \textit{ is in none of the preferred-extensions in AF} \end{cases}$$

where X_p is the mapped argument α in AR for $\{\mathring{A}\}$. Furthermore, The status of a literal $x \in \mathcal{L}$ is given by the status function $\mathbf{S}(x) = \max_{\mathring{A}\in\mathcal{A}}(\mathbf{E}(\mathring{A}))$ where $x \in Cn(\mathring{A})$, for any $x \in \bigcup_{\mathring{A}\in\mathcal{A}} Cn(\mathring{A})$, otherwise $S(x)$ is assigned 0 i.e. $S(x) = 0$. The values $1, 1/2$ and 0 stand for JUSTIFIED, DEFENSIBLE *and* OVERRULED.

The following observation ensures that semantics of an argumentation theory AT is well-defined.

Observation 5. *In an argumentation theory AT, every argument $\mathring{A}$ and literal $a \in \mathcal{L}$ has one and only one status.*

The following example shows the status of arguments in the induced argumentation theory given in Example 6.

Example 7. We extend example 6 as follows. Let $A = \{s, a, r\}$ and $AT = (\mathcal{A}, \otimes, \oplus)$ be an induced argumentation theory from $(A, \mathcal{R})$ that is,

$$\begin{aligned}
&\mathcal{A} = \{\ \mathring{A}_1 = \langle\ \{s\},\ \emptyset,\ s\rangle,\ \mathring{A}_2 = \langle\ \{a\},\ \emptyset,\ a\rangle, \mathring{A}_3 = \langle\ \{r\},\ \emptyset,\ r\rangle,\\
&\mathring{A}_{11} = \langle\ \{s\},\ \{d_1\},\ \neg a\rangle,\ \ \mathring{A}_{12} = \langle\ \{s\},\ \{d_2\},\ \neg e\rangle,\ \ \mathring{A}_{121} = \langle\ \{s\},\ \{d_2, d_7\},\ \neg c\rangle,\\
&\mathring{A}_{21} = \langle\ \{a\},\ \{d_3\},\ \neg s\rangle,\ \ \mathring{A}_{22} = \langle\ \{a\},\ \{d_4\},\ c\rangle,\ \ \mathring{A}_{221} = \langle\ \{a\},\ \{d_4, d_6\},\ e\rangle,\\
&\mathring{A}_{23} = \langle\ \{a\},\ \{d_5\},\ e\rangle,\quad \mathring{A}_{231} = \langle\ \{a\},\ \{d_5, d_9\},\ c\rangle,\\
&\mathring{A}_{31} = \langle\ \{r\},\ \{d_8\},\ c\rangle,\quad \mathring{A}_{311} = \langle\ \{r\},\ \{d_8, d_6\},\ e\rangle\ \ \}
\end{aligned}$$

The calculated state of $\mathring{A} \in \mathcal{A}$ is:

$$\mathbf{E}(\mathring{A}) = \begin{cases} 1 & \text{if } \mathring{A} \in \mathcal{A}_2 \\ 1/2 & \text{if } \mathring{A} \in \mathcal{A}_3 \\ 0 & \text{if } \mathring{A} \in (\mathcal{A} \setminus (\mathcal{A}_2 \cup \mathcal{A}_3)) \end{cases}$$

where $\mathcal{A}_2 = \{\mathring{A}_1, \mathring{A}_2, \mathring{A}_3, \mathring{A}_{22}, \mathring{A}_{221}, \mathring{A}_{23}, \mathring{A}_{231}, \mathring{A}_{31}\}$,
and $\mathcal{A}_3 = \{\mathring{A}_{12}, \mathring{A}_{23}, \mathring{A}_{231}, \mathring{A}_{311}\}$. ■

We define consistency in an argumentation theory based on its Dung's AF translation. For an argumentation theory to be consistent, no two justified arguments should have contradictory conclusions. Since the acceptability of an argument is captured through admissible set(s) then no two arguments in any given admissible set should have contradictory consequences.

Definition 12. *An argumentation theory $AT = \langle\mathcal{A}, \otimes, \oplus\rangle$ is said to be consistent iff there is no $a \in \mathcal{L}$ s.t. $a \in Cn(\mathring{A}_1)$, $\neg a \in Cn(\mathring{A}_2)$, $\mathring{A}_1, \mathring{A}_2 \in \mathcal{A}$, and, the corresponding mapped arguments α_1, α_2 in the translation AF belong to the same preferred extension.*

Observation 6. *Let $AT = \langle\mathcal{A}, \otimes, \oplus\rangle$ be an argumentation theory then*

1. *If AT is consistent then $\forall a \in \mathcal{L}$ if $\mathbf{S}(a) = 1$ then $\mathbf{S}(\neg a) \neq 1$ (note: $\neg(\neg a) = a$).*
2. *Let $D = D_1 \cup D_2$ be the set of defeasible rules of all arguments in $\mathcal{A}$ where D_1 is the set of indefeasible and D_2 the set of normal rules; and, C be the set of consequences of all arguments in $\mathcal{A}$ (i.e. $C = \bigcup_{\mathring{A}\in\mathcal{A}} Cn(\mathring{A})$). Then*

(a) if the induced argumentation theory AT' from (C, D_1) is consistent then AT is consistent.

(b) The mapping in translation to AF will be bijective; and, if $\mathbf{F}_d^k$ of all the rules $d \in D_2$ are composed of singletons then AF is isomorphic to the structure $\langle \mathcal{A}, R^* \rangle$ where $R^* = \{(\mathring{A}_1, \mathring{A}_2) \in \mathcal{A} \times \mathcal{A} \mid \{\mathring{A}_1\} defeats \mathring{A}_2\}$.

Observation 6.2(a) states that given an argumentation theory AT, if the indefeasible part of AT, i.e. the propositional part of AT, is consistent then if the defeasible part of AT constitutes of normal rules, AT will be consistent. Observation 6.2(b) is our last result on the parallel between our argumentation system and argumentation systems based on default rules. The bijective mapping in obs. 6.2(b) means that there are no imaginary arguments in translation AF. Furthermore, if defeaters in default rules are singleton defeaters then AT would have the same structure as its Dung's translation.

6 A Short Comparison with Other Defeasible Reasoning Systems

In this section we briefly discuss how the defeasible rules we proposed relate to other rule-based defeasible reasoning systems, namely BDKT abstract assumption-based framework [6], and the argumentation based Defeasible Logic [11]. These defeasible rules are essentially assumption based defeasible rules applicable in absence of ay contrary evidence in $\mathbf{F}_d$ and $\mathbf{U}_d$. Our argumentation theory is therefore a form of assumption based default reasoning theory. The language $\mathcal{L}$ and defeasible rules are similar to the language and rules in Defeasible Logic where language, body and head of rules are comprised of literals.

In relation to BDKT, assuming that $\mathbf{U}_d$ in all rules is empty, expanding $\mathcal{L}$ to include sentences with logical conjunction '$\wedge$' makes the proposed defeasible rules a form of grounded Reiter defaults where the assumption is the negation of elements of $\mathbf{F}_d$. It is already shown that an argumentation theory based on this modified version of rules can be captured by BDKT framework [6]. However, in BDKT there is no direct means of addressing the $\mathbf{U}_d$ part of a rule. In other words, in BDKT an acceptable set of assumptions can not to provisionally defeat an assumption in a direct fashion.

In relation to Defeasible Logic, assumption based rules and the corresponding undercutting attacks on the assumptions of rules can be represented in two ways depending on whether the assumptions are explicitly or implicitly expressed. In our approach assumptions are implicitly expressed. For a rule $r : bd(r) \Rightarrow hd(r)$ with an implicit assumption a, r is divided into two rules $r_1 : bd(r) \Rightarrow inf(r)$ and $r_2 : inf(r) \Rightarrow hd(r)$ [9]. The undercutting attack on the assumption of the rule can be expressed by $r_{dft} : \neg a \rightsquigarrow \neg inf(r)$ (or, alternatively $r_{dft} : \neg a \Rightarrow \neg inf(r)$) [10] . Though undercutting attacks can be expressed in Defeasible Logic, attacks themselves are invariant. So, we cannot represent reinstatement of a rule by context unless we expand the rule-base by additional rules as discussed in example 1. Furthermore, even if we assume that there is no reinstatement by context(effectively making attacks invarient,) a translation of $\mathbf{U}_d$ in terms

[9] Splitting rules in this fashion is originally proposed to express superiority relation among conflicting rules [2].

[10] The reverse-translation, i.e. translating a Defeasible-Logic rule to a Default-Logic rule, is given in [3].

of undercutting defeater would only make sense for ambiguity blocking semantics [11] of Defeasible Logic. There is no direct means of addressing $\mathbf{U}_d$ in terms of a defeater for an ambiguity propagating semantics [11].

As it can be seen the two reasons why this argumentation system cannot be directly expressed in BDKT framework or in argumentation based Defeasible Logic are $\mathbf{U}_d$ and reinstatement by context. In light of the results and discussion above, it can be argued that with appropriate semantics, an induced argumentation theory from the proposed default rules can indeed be embedded in both BDKT framework and Defeasible Logic. It is possible to envisage a schema for translating a non-default rule to a set of new default rules, effectively constructing an argumentation theory that consists of only indefeasible and default rules. The translated argumentation theory will be equivalent to the original theory *w.r.t.* the status of literals in $\mathcal{L}$. Such translation allows us to capture a given argumentation theory in BDKT framework (or in Defeasible Logic.) The schema is similar to the method used for translation to Dung's Framework [11].

7 Discussion and Future Direction

In this paper we proposed a simple representation of defeasible rules consisting of only literals. Each defeasible rule is associated with a justification function. The justification function effectively describes under what circumstances a rule cannot be applied. The antecedents of a rule are the primary reasons for believing the consequent. The literals in the justification function are taken to be the ancillary reasons that strengthen or weaken a rule. Unlike most argumentation systems, in this system arguments attack or reinstate other arguments indirectly via context. The context is the collection of consequences of arguments in a set of arguments. We also provided a translation from this system to Dung's abstract argumentation theory in way of validating our approach.

Our investigation into defeasible rules in the context of argumentation systems is programmatic in character. There are some important issues that have not yet been addressed, including:

1. Our system shares a problem regarding the non-normal Reiter defaults [19] in relation to two seemingly acceptable arguments that are built upon contradictory assumptions. We will address this problem along the lines suggested in [6] where the conceded assumptions are explicitly stated.
2. In general, the contrapositives of default rules are not automatically allowed in defeasible reasoning systems [6,19] although it has been argued that the contrapositives help avoid certain counterintuitive results [7].

[11] We provide only the basic idea behind the schema. For every rule d, we construct all possible sequences of the form $J_1, J_2, \cdots, J_n$ where $J_i \subseteq \mathbf{J}_d$, $J_1 \in \mathbf{U}_d \; or \mathbf{F}_d$, $J_i \subset J_{i+1}$, and J_{i+1} is the minimum J_k *w.r.t.* $\subseteq$ that is not in the same class as J_i (class in terms of $\mathbf{T}_d, \mathbf{U}_d, \mathbf{F}_d$). For every distinct $J_{i+1} \setminus J_i$ we construct a rule d_i^* where $bd(d_i^*) = J_{i+1} \setminus J_i$, $hd(d_i^*) = a_i^*$, $\mathbf{J}_{d_i^*} = \mathbf{F}_{d_i^*} = \{b_i^*\}$, and $a_i^* = b_{i-1}^*$. In addition, $bd(d_1^*) = J_1$ and $\mathbf{J}_{d_n^*} = \mathbf{F}_{d_n^*} = \emptyset$; and, if $J_{i+1} \in \mathbf{U}_d$ then $a_{i-1}^* = b_i^*$. This way we extend $(\mathcal{L}, \mathcal{R})$ to $(\mathcal{L}^*, \mathcal{R}^*)$ by newly introduced a_i^*, b_i^* and d_i^*. It can be shown that an argumentation theory from substitution of non-default rules with the corresponding set of default rules is equivalent to the original theory *w.r.t.* $\mathcal{L}$.

In our future work we will show how a rule can be explained by other rules, including expressing non-default rules as a set of default rules. We will address introduction of logical connectives in the antecedent of a rule, as well as, giving a more in depth comparison with other defeasible reasoning systems.

References

1. Amgoud, L., Cayrol, C.: A Reasoning Model Based on the Production of Acceptable Arguments. Ann. Math. Artif. Intell. 34(1-3), 197–215 (2002)
2. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Trans. Comput. Log. 2(2), 255–287 (2001)
3. Antoniou, G., Billington, D.: Relating Defeasible and Default Logic. Australian Joint Conference on Artificial Intelligence 2001, 13–24 (2001)
4. Asher, N., Pelletier, J.: Generics and Defaults. In: van Bentham, J., ter Meulen, A. (eds.) Handbook of Logic and Language 1997. Elsevier, Amsterdam (1997)
5. Baroni, P., Giacomin, M.: Solving Semantic Problems with Odd-Length Cycles in Argumentation. In: Nielsen, T.D., Zhang, N.L. (eds.) ECSQARU 2003. LNCS (LNAI), vol. 2711, pp. 440–451. Springer, Heidelberg (2003)
6. Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An Abstract, Argumentation- Theoretic Approach to Default Reasoning. Artif. Intell. 93, 63–101 (1997)
7. Caminada, M., Amgoud, L.: An Axiomatic Account of Formal Argumentation. In: AAAI 2005, pp. 608–613 (2005)
8. Chesnevar, C.I., Simari, G.R.: A Lattice-Based Approach to ComputingWarranted Beliefs in Skeptical Argumentation Frameworks. In: IJCAI 2007, pp. 280–285 (2007)
9. Dung, P.M.: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artif. Intell. 77(2), 321–358 (1995)
10. Dung, P.M., Son, T.C.: An argument-based approach to reasoning with specificity. Artif. Intell. 133(1-2), 35–85 (2001)
11. Governatori, G., Maher, M.J., Antoniou, G., Billington, D.: Argumentation Semantics for Defeasible Logic. J. Log. Comput. 14(5), 675–702 (2004)
12. Horty, J.: Skepticism and floating conclusions. Artif. Intell. 135(1-2), 55–72 (2002)
13. Katzav, J., Reed, C.: On Argumentation Schemes and the Natural Classification of Arguments. Argumentation 18(2), 239–259 (2004)
14. McCarthy, J.: Applications of Circumscription to Formalizing Common-Sense Knowledge. Artif. Intell. 28(1), 89–116 (1986)
15. Nute, D.: Defeasible Logic. In: Bartenstein, O., Geske, U., Hannebauer, M., Yoshie, O. (eds.) INAP 2001. LNCS (LNAI), vol. 2543, pp. 151–169. Springer, Heidelberg (2003)
16. Pollock, J.L.: Defeasible Reasoning. Cognitive Science 11(4), 481–518 (1987)
17. Pollock, J.L.: Justification and Defeat. Artif. Intell. 67(2), 377–407 (1994)
18. Pollock, J.L.: Defeasible reasoning with variable degrees of justification. Artif. Intell. 133(1-2), 233–282 (2001)
19. Poole, D.: The Effect of Knowledge on Belief: Conditioning, Specificity and the Lottery Paradox in Default Reasoning. Artif. Intell. 49(1-3), 281–307 (1991)
20. Prakken, H., Reed, C., Walton, D.N.: Argumentation Schemes and Generalizations in Reasoning about Evidence. In: ICAIL 2003, pp. 32–41 (2003)
21. Quoc, B.V., Foo, N.Y., Thurbon, J.: Semantics for a theory of defeasible reasoning. Ann. Math. Artif. Intell. 44(1-2), 87–119 (2005)
22. Reed, C., Grasso, F.: Computational Models of Natural Language Argument. In: International Conference on Computational Science, vol. 1(2000), pp. 999–1008 (2001)

23. Reed, C., Walton, D.: Towards a Formal and Implemented Model of Argumentation Schemes in Agent Communication. Autonomous Agents and Multi-Agent Systems 11(2), 173–188 (2005)
24. Reiter, R.: A Logic for Default Reasoning. Artif. Intell. 13(1-2), 81–132 (1980)
25. Simari, G.R., Loui, R.P.: A Mathematical Treatment of Defeasible Reasoning and its Implementation. Artif. Intell. 53(2-3), 125–157 (1992)
26. Verheij, B.: Accrual of arguments in defeasible argumentation. In: Proceedings of the Second Workshop on Dutch/German Workshop on Nonmonotonic Reasoning, pp. 217–224 (1995)
27. Vreeswijk, G.: Abstract Argumentation Systems. Artif. Intell. 90(1-2), 225–279 (1997)
28. Vreeswijk, G., Prakken, H.: Logical systems for defeasible argumentation. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, 2nd edn., vol. 4, pp. 219–318. Kluwer, Dordrecht (2002)
29. Wooldridge, M., McBurney, P., Parsons, S.: On the meta-logic of arguments. In: AAMAS 2005, pp. 560–567 (2005)

Proof of Observations

Proof. Observation 1

1. $\mathbf{T}_d, \mathbf{U}_d, \mathbf{F}_d$ are mutually exclusive, therefore $\mathbf{H}_d(A)$ has one and only one value.
2. (→) If $\exists A \in \mathbf{F}_d$ that A is a subset of $B \in \mathbf{T}_d$ then not all $A \in \mathbf{F}_d$ are conclusive defeaters of d which is contradictory to the assumption.
   (←) If no $A \in \mathbf{F}_d$ is a subset of $B \in \mathbf{T}_d$ then for $\forall A \in \mathbf{F}_d$, A is a conclusive defeater of d making d a default rule.
3. The members in $\mathbf{F}_d$ create a partial order w.r.t set inclusion. Since every $A \in \mathbf{F}_d^k$ is the minimal set in $\mathbf{F}_d$ then no proper subset of A is in $\mathbf{F}_d$.
   (a) Therefore, if A is not a singleton then all its members should belong to some $B \in \mathbf{T}_d$. Otherwise, it would contradict the assumption.
   (b) (→) $\exists C \in \mathbf{F}_d$ s.t. $C \subseteq X$ otherwise $\mathbf{H}_d(X) \neq 0$. So, there is $A \in \mathbf{F}_d^k$ s.t. $A \subseteq C$.
   (←) Suppose, there is $A \in \mathbf{F}_d^k$ s.t. $A \subseteq X$ then by definition of the conclusive defeater $\mathbf{H}_d(X) = 0$. □

Proof. Observation 2

1. The function $\min_{d \in D}(\mathbf{H}_d(X))$ has one and only one value for a given argument.
2. Definition of conclusive defeat implies that defeat is context independent so $\min_{d \in D}$ $(\mathbf{H}_d(X))$ = 0 is context independent. □

Proof. Observation 3

1. By definition of defeat relationship $\exists \mathcal{A}_1 \subseteq \mathcal{A}$ such that given defeat conditions are satisfied (for outright or provisional defeat). Now, since $\mathcal{A}_1$ is also subset of $\mathcal{A}'$, the same defeat conditions are still satisfied in relation to $\mathcal{A}'$.
2. By definition of conclusive-defeat-scenario, $\forall \mathcal{A}''$ and $\mathcal{A}_c \subseteq \mathcal{A}'' \subseteq \mathcal{A}$ then $\exists d \in D_{\mathring{A}}$ s.t. $\mathbf{H}_d(Cn(\mathcal{A}''))$ = 0. Moreover, since $\mathring{A} \sqsubset \mathring{A}'$ we have $d \in D_{\mathring{A}'}$. Hence, $\mathbf{H}_{\mathring{A}'}(\mathcal{A}'')$ = 0 or $\mathcal{A}_c$ is a conclusive defeat scenario for $\mathring{A}'$ in $\mathcal{A}$.

3. Let $\mathcal{A}_c$ be a defeater of Å in $\mathcal{A}$ then by def. 6 $Cn(\mathcal{A}_c)$ are defeaters of a rule $d \in D_{\mathring{A}}$ and by obs. 1.3(b) $F \subseteq Cn(\mathcal{A}_c)$ where $F \in \mathbf{F}^k_d$. Therefore, $\mathcal{A}_c$ is a TConclusive defeat scenario (obs. 2.2 and definition TConclusive defeat scenario). Next, if $\mathcal{A}_c$ is not the minimal set where $F \subseteq Cn(\mathcal{A}_c)$ then there is $\mathcal{A}'_c \subset \mathcal{A}_c$ s.t. $F \subseteq Cn(\mathcal{A}'_c)$. This means $\mathbf{H}_{\mathring{A}}(\mathcal{A}' \setminus \{\mathring{A}_j\}) = 0$ contradicting defeat condition in def. 6.1 ($\mathcal{A}'$ is the $\mathcal{A}'$ in def. 6.1 and $\mathring{A}_j \in (\mathcal{A}_c \setminus \mathcal{A}'_c)$. For the same reason (i.e. $\mathcal{A}_c$ being the maximal set in def. 6.1) the context of defeat is $\mathcal{A}' \setminus \mathcal{A}_c = \emptyset$, and, if $F \in \mathbf{F}^k_d$ is singleton then $\mathcal{A}_c$ will be singleton.
4. $Cn(\mathcal{A}_c)$ defeating a rule d implies $\mathbf{H}_{\mathring{A}}(\mathcal{A}'') = 0$ for any $\mathcal{A}''$ s.t. $\mathcal{A}_c \subseteq \mathcal{A}''$. □

Proof. Observation 4
If D is only comprised of indefeasible or default rules then either arguments have no defeaters or any argument set $\mathcal{A}_c$ that defeats an argument in $\mathcal{A}$ is its conclusive defeater (Observation 3.3). Furthermore, any attack by a conclusive defeater can not be reinstated by context ($\mathbf{H}_{\mathring{A}}(\mathcal{A}')$ where $\mathcal{A}_c \subseteq \mathcal{A}'$ is always zero). □

Proof. Observation 5
The Status function $\mathbf{E}$ is sum of three partial functions with exclusive domains. Moreover, from definition of Translation, every $\{\mathring{A}\}$ is mapped to one and only one argument $\alpha \in AR$ of AF. Hence, every argument has one and only one status. The status function $\mathbf{S}$ is a *max function*, therefore, every literal also has one and only one status. □

Proof. Observation 6

1. If $\mathbf{S}(a) = 1$ and $\mathbf{S}(\neg a) = 1$ then there are two justified arguments $\mathring{A}_1, \mathring{A}_2 \in \mathcal{A}$ where $a \in Cn(\mathcal{A}_1)$ and $\neg a \in Cn(\mathcal{A}_2)$. Thus, their mapped arguments α_1, α_2 in Dung's AF belong to all preferred extensions contradicting the initial assumption.
2. (a) If AT is not consistent then there are two arguments $\mathring{A}_1, \mathring{A}_2 \in \mathcal{A}$ such that $a \in Cn(\mathring{A}_1)$ and $\neg a \in Cn(\mathring{A}_2)$. Hence, there are two rules $d_1, d_2 \in D$ such that $hd(d_1) = a$ and $hd(d_2) = \neg a$. Now, if one of the d_1 or d_2 is a normal rule then the corresponding argument should have been conclusively defeated by the other argument and so not to be accepted. Therefore, d_1 and d_2 should both be indefeasible rules. But, if d_1 and d_2 are both indefeasible rules then the induced argumentation theory AT' from $(bd(d_1) \cup bd(d_2), \{d_1, d_2\})$ would be inconsistent which is contradictory to the initial assumption.
   (b) According to obs. 4, AT is context insensitive, therefore, rules 1(c), 2(a)(ii), 2(a)(iii), 2(b) of translation are not applied. Hence, the mapping $\mathbf{M1}$ and $\mathbf{M2}$ are 1-to-1 and consequently bijective mappings. Now, if all members of $\mathbf{F}^k_d$ are singletons then all $\mathcal{A}_c$ in AT are singletons (obs. 3.3). Thus, by rule 1(b) of translation, arguments in AF are bijective mappings of singleton argument sets in AT. Hence, $\mathbf{M1}$ acts like a bijective function from $\mathcal{A}$ to AR. Moreover, since all attack relationship in R are bijective mappings from $\{singleton\_argument\_set\} \times \{singleton\_argument\_set\}$ (all $\mathcal{A}_c$ are singletons), $\mathbf{M2}$ acts like a bijective function from R^* to R. Hence, $\langle \mathcal{A}, R^* \rangle$ and $\langle AR, R \rangle$ are two isomorphic structures. □

Fuzzy Argumentation for Trust

Ruben Stranders, Mathijs de Weerdt, and Cees Witteveen

Delft University of Technology
rs06r@ecs.soton.ac.uk, {M.M.deWeerdt,C.Witteveen}@tudelft.nl

Abstract. In an open Multi-Agent System, the goals of agents acting on behalf of their owners often conflict with each other. Therefore, a personal agent protecting the interest of a single user cannot always rely on them. Consequently, such a personal agent needs to be able to reason about trusting (information or services provided by) other agents. Existing algorithms that perform such reasoning mainly focus on the immediate utility of a trusting decision, but do not provide an explanation of their actions to the user. This may hinder the acceptance of agent-based technologies in sensitive applications where users need to rely on their personal agents.

Against this background, we propose a new approach to trust based on argumentation that aims to expose the rationale behind such trusting decisions. Our solution features a separation of opponent modeling and decision making. It uses possibilistic logic to model behavior of opponents, and we propose an extension of the argumentation framework by Amgoud and Prade [1] to use the fuzzy rules within these models for well-supported decisions.

1 Introduction

An open Multi-Agent System (MAS) is characterized by an agent's freedom to enter and exit the system as it pleases, and the lack of central regulation and control of behavior. In such a MAS, agents are often not only dependent upon each other, as for example in Computer-Supported Cooperative Work (CSCW) [2], web services [3], e-Business [4,5], and Human-Computer interaction [6], but their goals may also be in conflict. As a consequence, agents in such a system are not reliable or trustworthy by default, and an agent needs to take into account the *trustworthiness* of other agents when planning how to satisfy its owner's demands.

Several algorithms have been devised to confront this problem of estimating trustworthiness by capturing past experiences in one or two values to estimate future behavior (e.g. see the survey by Dash et al. [7]). These algorithms, however, primarily focus on improving the immediate success of an agent. Less emphasis is laid on discovering patterns in the behavior of other agents, or—more challenging—their incentives. Moreover, the *rationale* of a decision often eludes the user: in most approaches it is 'hidden' in a large amount of numerical data, or simply incomprehensible. At any rate, these approaches do not provide human-readable information about these decisions, and were indeed not designed to do this.

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 214–230, 2008.

The following example illustrates the importance of the rationale behind the agent's decision. Suppose a user instructs a personal agent to buy a painting for his collection. When an interesting painting is offered, this agent estimates its value by requesting the opinion from a number of experts. To obtain a good estimate, it then assigns weights to the various received appraisals. When the user plans to buy a very valuable painting, he is not just interested in the final estimate of this agent, or in the retrieved estimates and their weights. When so much is at stake, he wants to know where these weights come from. Why, for example, is the weight for this famous expert so low? If the agent told him that this is because this expert is known to misrepresent his estimate in cases where he is interested in buying himself, and this may be such a case, would not this agent be much more useful than an agent that simply assigns a number to the trustworthiness of the expert?

The lack of such explanations can severely hamper the acceptance of agent-based technology, especially in areas where users rely on agents to perform sensitive tasks. Without the availability of these explanations, the user almost needs to have blind faith in his agent's ability to trust other agents. We believe that the state of the art in dealing with trust in Multi-Agent Systems has not sufficiently addressed this issue. Therefore, we are interested in an approach that lays more emphasis on the rationale of trusting decisions, and in this paper we work towards a proof-of-concept of such an approach.

Due to the uncertainty of information in Multi-Agent Systems, this setting gives rise to some specific requirements of the opponent model an agent should be able to build: (i) The model should be able to represent inherently uncertain, ambiguous, and incomplete knowledge about other agents, and (ii) it should support an argumentation framework capable of making decisions and explaining them. This implies that the opponent model should support logical rules.

We put forward such a model in Section 2, where the core idea of our approach is presented: a unique combination of a fuzzy rule opponent modeling technique and a solid argumentation framework applied to the process of making trust decisions. In this section we also explain how the argumentation framework by Amgoud and Prade [1] can be extended to deal with situations with not only possibilistic rules, but also where the rules themselves are not always fully applicable to a given situation. In Section 3 we show how this model can be applied within the context of an art appraisal domain, as described in the Agent Reputation and Trust (ART) testbed [8]. The final section summarizes the benefits of an argumentation-based approach to explaining trusting decisions, discusses related work, and gives some interesting ways of extending the ideas given in this paper.

2 An Architecture for Fuzzy Argumentation

The goal of the approach presented in this paper is to capture uncertain knowledge about other agents in logical rules, and to use this knowledge to derive not only good decisions, but also arguments to support these decisions. In this section

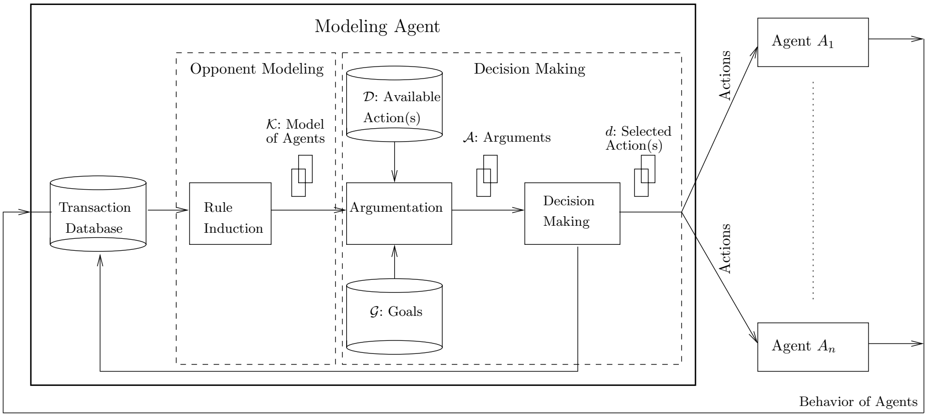


Fig. 1. The architecture for the Modeling Agent

we describe the global architecture of our approach, the formal argumentation framework for making the decisions, and the opponent-modeling algorithm we used in our proof of concept.

2.1 High-level Architecture

Figure 1 shows the architecture of our proposed approach, and introduces some of the terminology used throughout the rest of the paper. The two main components of our framework are *opponent modeling* and *decision making*. The opponent modeling component is responsible for modeling the behavior of other agents, based on past experiences with these agents. Data from these past experiences are stored in a *transaction database*. From this data a knowledge base of rules is induced that models the behavior of each agent. The details of opponent modeling are discussed in Section 2.2.

The decision making component (details in Section 2.3) is responsible for making the actual decisions. Decisions may be supported by *arguments*. An argument relates to a prediction of future behavior of opponents, and is obtained using the opponent models. The extent to which an argument supports a decision is expressed in terms of its *strength*. The strength of an argument is composed of the argument's *weight*, which defines the desirability of the predicted result of this decision, and of its *level*, which is the amount of confidence in the accuracy of the prediction. After having executed the decision that is supported by the strongest argument, the *actual* outcomes are observed and recorded in the transaction database. These new results are subsequently used to refine the model of the opposing agents once again, completing the circle.

2.2 Opponent Modeling

In order to explain an argument for trust to a user, the agent first needs to possess knowledge about other agents. The format in which the knowledge is expressed should be capable of capturing the inherent vagueness and ambiguousness of information in a trust domain. Fuzzy (or possibilistic) logic [9] is an adequate tool to tackle this modeling problem, because it provides a natural way of translating back and forth between logical rules describing the expected (or learned) behavior of other agents, and uncertain numerical data.

For brevity and clarity, we omit the details of the specific variety of fuzzy logic used in the knowledge bases of our agent, and instead focus on the intuition and the ideas behind our approach. Therefore, it is sufficient to know that a fuzzy proposition is a statement of the form "property x is high", meaning that x is a member of the fuzzy set "high". A formula in our fuzzy language $\mathcal{L}$ can be composed of such elementary statements using the fuzzy logic operators that intuitively extend the semantics of standard propositional logic to the fuzzy domain.

Now, the knowledge base to model the other agents consists of such fuzzy formulas that each describe a specific aspect of another agent's behavior. However, since such knowledge is constructed based on past interactions with other agents, not all of these learned formulas will have the same status; the inherent unreliability and unpredictability of other agents might cause our agent to add imprecise or even incorrect rules to its knowledge base. Therefore, we also add a confidence value to each formula in the knowledge base to represent the certainty with which the formula has been learned.

Definition 1. *A* knowledge base $\mathcal{K}$ *is a set of tuples* (k_i, ρ_i) *where* $k_i \in \mathcal{L}$ *is a fuzzy formula, and* $\rho_i \in [0, 1]$ *is the confidence the agent has in* k_i*.*

The valuation of a fuzzy formula depends on a given state of the world w. Such a state w is a description of the current state of the environment by a set of propositions. In our application, a world state represent the actions of our agent towards other agents in the past, which might influence their behavior in future interactions. Given a world state, the extent to which a fuzzy formula is valid can be determined using a valuation function, which assigns a measure of applicability to each formula.

Definition 2. *Given a world state w, the* valuation function $v_w : \mathcal{L} \rightarrow [0, 1]$ *gives the applicability of a fuzzy formula in the world w.*

In most situations, the knowledge base consists of fuzzy *rules*, i.e. a material implication from an observation (condition) to an expected/learned effect (conclusion). Such a rule can be *partially* applicable in a particular world state, instead of just being fully applicable or not at all. If k_i is a fuzzy rule, we say that $v_w(k_i)$ is the *match strength* of k_i. Consider the following example of such a fuzzy rule.

Example 1. Suppose we own a (possibly) very valuable painting, and we would like to have it appraised by taking a weighted average over a set of appraisal agents. Each of these agents not only gives the appraisal itself, but also a claim on its certainty about this appraisal. To know which agents to trust, we look at their behavior in the past. Such previous interactions have led us to believe that the following rule k_1 accurately describes one of the agent's (a) behavior: "**if** a says it is certain with level c_{high} (very certain) **then** agent a's appraisal error ae_{low} (low)". This rule should be interpreted as follows: if agent a's certainty is a member of the fuzzy set c_{high}, which contains all high values of certainty, its appraisal error will be a member of the fuzzy set ae_{low}, containing all low values of appraisal error. So, if this agent claims it is very certain about its appraisal, we conclude that its appraisal error is low, and we will base our own estimate strongly on this agent's appraisal.

As hinted above, *membership* of a fuzzy set is not just true or false, but can take on a range of values between 0 and 1. Suppose that in a certain world w agent a's certainty c is not exactly c_{high}, but slightly lower. In that case c's membership of c_{high} is less than 1. Rule k_1 still applies, but its match strength $v_w(k_1)$ will also be less than one. In this case, we say the rule fires *partially*, and consequently we cannot predict that the appraisal error will be exactly ae_{low}. To be more precise, the membership of the actual appraisal error in ae_{low} is less than 1. In plain English, this implies that we should expect an appraisal error that is not low, but slightly higher.

At this point, it is important to note the difference between the confidence ρ_i in a rule k_i, and its match strength $v_w(k_i)$ for a certain world state w. The former represents the *validity* of the rule in describing a *certain system or agent*, and the latter represents the *applicability* of a rule to the system or agent in a particular *state of the world*. In the previous example, rule k_1 might not be valid at all for describing a's behavior. Put differently, the rule could be wrong, in which case the confidence ρ_1 should be close to 0. On the other hand, given a certain scenario (for example, in which certainty equals c), rule k_1 could be used to predict the behavior of the agent, *provided that* c is a member of c_{high}. Otherwise, the preconditions of the rule are not met, and the rule does not apply to the world state w. As a result, the rule's match strength $v_w(k_i)$ is zero.

Keeping in mind the requirements identified in the introduction (the ability to model uncertainty and at the same time support an argumentation framework), we decided to use a simple theory revision algorithm called Fuzzy Rule Learner (FURL) [10] to construct such a knowledge base containing the observed behavior of the other agents. Taking observations from the environment as input, FURL is capable of creating a rule base of fuzzy rules. FURL's output consists of a multi-level rule base known as a Hierarchical Prioritized Structure [11] and, for each rule, the prediction error it causes on past observations (the training set). Rules in each level can be thought of as an exception to rules in the layer below it. For our application, however, we can think of the result just as a (flat) rule base with fuzzy rules and their prediction error where the confidence values are taken to be the inverse of the prediction error of the rule according to FURL.

2.3 Decision Making

In this section we introduce the argumentation framework used in the decision making component. The work by Amgoud and Prade [1,12] was considered to be a good basis for such a framework, because it inherently supports reasoning under uncertainty with fuzzy logic. This framework uses the agent's knowledge base $\mathcal{K}$, a set of its goals $\mathcal{G}$, and a set of possible decisions (or actions) $\mathcal{D}$. An argument A in favor of a decision $d \in \mathcal{D}$ is then defined as follows [1,12].

Definition 3. *Given an agent $\langle \mathcal{K}, \mathcal{G}, \mathcal{D} \rangle$, an* argument A in favor *of a decision $d \in \mathcal{D}$ is a triple $A = \langle S, C, d \rangle$, where*

- *$S \subseteq \mathcal{K}$ is the support of the argument, containing the knowledge from the agent's knowledge base $\mathcal{K}$ used to predict the consequences of decision d,*
- *$C \subseteq \mathcal{G}$ are the consequences of the argument, i.e. the goals reached by decision d, and*
- *$d \in \mathcal{D}$ is the conclusion argument A recommends.*

Moreover, $S \cup \{d\}$ should entail C, S should be minimal, and C maximal among the sets satisfying the above conditions.

The original framework proposed in [1] requires that the support S should be *consistent* with d. That is, applying d should not result in a contradiction with previously acquired knowledge. However, the original framework is based on propositional logic, whereas our method uses fuzzy logic. In contrast to propositional logic, applying a decision d on a fuzzy knowledge base $\mathcal{K}$, will not result in a contradiction, regardless of the contents of $\mathcal{K}$. The consistency requirement is therefore no longer relevant. This is due to the fact that a fuzzy rulebase is inherently capable of resolving inconsistencies. More specifically, when multiple rules fire at the same time, with different outputs, these outputs are fused together and converted into a scalar using a process called *defuzzification* [13].

The set $\mathcal{A}$ gathers all arguments that can be constructed from $\mathcal{K}$, $\mathcal{G}$, and $\mathcal{D}$ as follows: for each decision $d \in \mathcal{D}$, the consequences $C \subseteq \mathcal{G}$ are predicted using a subset S of the knowledge base $\mathcal{K}$, resulting in an argument $\langle S, C, d \rangle$. Subsequently, a decision is made by selecting the argument(s) with the highest strength.

The process of reaching a decision can be determined in four steps:

1. The *level* of the argument is calculated based on the confidence in support S. Remember from the previous section that the confidence in a rule depends on how well it models another agent's behavior.
2. The *weight* (or desirability) of the outcomes C is evaluated in light of the goals of the agent (or those of its owner).
3. The *level* and the *weight* of each argument are combined in its *strength.* Strength can be considered as a summary of the argument's validity and the desirability of the predicted outcomes of the decision it supports.
4. The decision supported by the argument with the highest strength is selected.

We will now discuss each step in more detail.

In the original framework [1,12], the *level* of an argument solely referred to the amount of *confidence* in the rules and facts in the support of the argument: $\mathit{Level}\,(\langle S, C, d \rangle) = \min \{\rho_i \mid (k_i, \rho_i) \in S\}$. However, in our model, rules in the knowledge base cannot be applied regardless of the state of the world. Often, their precondition matches only partially with the facts in this state (as in Example 1). Therefore, our definition of the level of an argument needs to take care of the balance between this match strength in an environment w and the confidence of the rules in the knowledge base:

1. For equal confidence levels ρ, the knowledge rule with the highest match strength should determine the *Level* of the argument. The higher the match strength, the more the knowledge applies to the current world state, and the more reliable it is in this particular case.
2. For equal match strengths v_w, the knowledge rule with the lowest level of confidence should determine the *Level* of the argument. This is consistent with the argumentation framework presented in [1].
3. In cases where a rule is fully matched, or not matched at all (e.g. $v_w(k) \in \{0, 1\}$), our definition should reduce to the definition of *Level* in the original framework.

Therefore we base the level of an argument on the weakest part of the argument. In our case, the weakest rule in a support set S given a world state w has not only a low confidence, but also a high match strength.

Definition 4. *Given a world state w, and a support set S, the weakest rule (k_j, ρ_j) is obtained by:*

$$(k_j, \rho_j) = \arg\min_{(k_i,\rho_i)\in S} \left\{ \frac{\rho_i}{v_w(k_i)} \;\middle|\; v_w(k_i) \neq 0 \right\} . \tag{1}$$

We define the *level* of an argument as the product of the confidence and the match strength of this weakest rule.

Definition 5. *Given a current world state w, the* level of an argument $A = \langle S, C, d\rangle$ *is defined by $Level_w(A) = \rho_j \cdot v_w(k_j)$, where (k_j, ρ_j) is the weakest rule.*

It is easy to check that this definition meets all three of the requirements stated above: the rule with the lowest confidence level and the highest match strength is selected, and the resulting level is the confidence level of this rule times multiplied by its match strength.

The *Weight* of an argument A depends on the goals that can be reached. The goals are given as tuples (g_j, λ_j) in the set $\mathcal{G}$. Like an element from the knowledge base, a goal g_j is a fuzzy rule or fact. The attached value $0 \leq \lambda_j \leq 1$ denotes the preference of the goal. In the original definition [1], weight is inversely proportional to the preference of the most important goal that is not satisfied. However, when using fuzzy logic, the predicate *satisfied* becomes fuzzy as well, making this definition very difficult to apply. We therefore chose to re-establish a similar relation between weight on the one hand, and preference and goal satisfaction on the other hand. One of the key properties of such a relation is that the more important a goal is, the more detrimental the reduction in weight when the goal becomes less satisfied. This is realized by the following definition.

Definition 6. *Given a current world state w, the* weight of an argument $A = \langle S, C, d\rangle$ *is defined by*

$$Weight_w(A) = \sum_{(g_j,\lambda_j)\in C} v_{w\cup S}\,(g_j)\cdot\lambda_j. \tag{2}$$

This definition ensures that the weight of the argument is proportional to the utility of the expected consequences of the decision. More specifically, if a goal g with preference λ is 50% true, we expect the utility to increase with $\lambda/2$. We sum over all goals of the agent to obtain the weight of the argument. As Section 3.2 shows, this also brings about a more intuitive trade-off between (possibly conflicting) goals.

Finally, the *Weight* and *Level* of an argument are combined into its the strength. Here we just follow the original definition [1].

Definition 7. *Given a current world state w, the* strength of an argument *is defined by $Strength_w(A) = Level_w(A)\cdot Weight_w(A)$.*

Such a value of *Strength* can then be used to determine which argument is more preferred.

Definition 8. *Let A and B be two arguments in $\mathcal{A}$. Argument A* is preferred to *B iff $Strength(A) \leq Strength(B)$.*

The upcoming section illustrates how an agent built according to this architecture operates in a simple problem domain.

3 Examples

In this section, we would like to investigate how an agent based on our approach behaves in a simple art appraisal environment. We assume the environment is inhabited with other agents with fixed strategies, and show that it is capable of explaining its decisions in terms of aggregated observations (rules).

The Agent Reputation and Trust (ART) testbed provides a simple environment to do our experiments [8]. ART is becoming the *de facto* standard for experimenting with trust algorithms and evaluating their performance. In this environment our personal agent is put in competition with N other agents to estimate the true value v of a painting. Each agent has its own area of expertise for which it can give good opinions to others. Consequently, it is often wise to consult other agents for an estimate of the value of the painting. Each other agent i can send a tuple (c_i, e_i) to our agent upon request where e_i is the estimate, and c_i is the *claimed* certainty of this estimate (c_1 being a low certainty and c_6 a high certainty). Our agent then should combine these estimates in its own appraisal by submitting a weight vector $\mathbf{w} = \{w_1, w_2, \ldots, w_N\}$ to the testbed, where $w_i \geq 0$, and $\sum_i w_i = 1$. The weight for an agent i should not only depend on its claimed certainty c_i, but also on its trustworthiness. Our slightly modified version of the testbed[1] subsequently calculates the weighted average of the estimates to obtain the final appraisal $a = \sum_{i=1}^{N} w_i e_i$.

As agents are rewarded based on the accuracy of their appraisals, they should aim to find the weight vector $\mathbf{w}$ that minimizes the appraisal error for a painting with true value v:

$$\mathbf{w} = \arg\min_{\mathbf{w} \in \mathbb{R}^N} \left| v - \sum_{i=1}^{N} w_i e_i \right| \tag{3}$$

To determine a suitable $\mathbf{w}$, our agent attempts to find a relation between the claimed certainty c_i, and the accuracy $|v - e_i|$ for each agent i. Now, since agents compete with each other for a number of rounds (appraising different paintings), it may be worthwhile to deceive other agents misrepresenting the claimed certainty at some point. Needless to say, this creates an issue of trust. Knowing when and whom to trust is therefore a prerequisite for success in this domain.

In the two scenarios that follow, we study the decision making process of our agent while in competition with two other agents: Honest and Reciprocal. Honest is an agent that honestly asserts a certainty c proportional to its expected accuracy, i.e. $c_{\textsc{Honest}} \propto |v - e_{\textsc{Honest}}|$.

Reciprocal's behavior is somewhat more complicated. When an opponent has misrepresented its expertise by overstating its certainty, Reciprocal responds in kind by being dishonest as well. However, if Reciprocal's opponent is honest, Reciprocal behavior towards that opponent is identically to that of Honest.

In each of the following scenarios, our agent has interacted with both agents in 200 transactions. From the observations made during these 200 transactions, we used FURL to build an opponent model which constitutes the knowledge base $\mathcal{K}$. The knowledge bases use three different fuzzy domains: c_0 to c_5 denote very low to very high certainty, ae_0 to ae_7 denote eight different levels of the appraisal error from low to high, and d_0 to d_5 denote the levels of dishonesty of our agent in the previous round, also from low to high.

To give an example, Figure 2 shows the accuracy of the predictions made by the model learned by FURL from observing Reciprocal's behavior. During this run, our agent 'tests' Reciprocal by alternating between honest and dishonest behavior towards it. As can be observed from Figure 2, FURL is reasonably capable of learning the effect of dishonesty on Reciprocal's behavior. At the end of the run, the learned model contains multiple fuzzy if-then rules describing the behavior, together with a confidence measure. To show what an opponent model looks like, Tables 1 and 2 contain a selection of rules from the opponent models of Reciprocal and Honest after 200 transactions.

[1] In our preliminary experiments, estimates are generated deterministically, instead of being drawn from a probability distribution. That way, we could significantly reduce the length of each competition run and still obtain useful results on the explanatory power of the arguments generated by our agent.

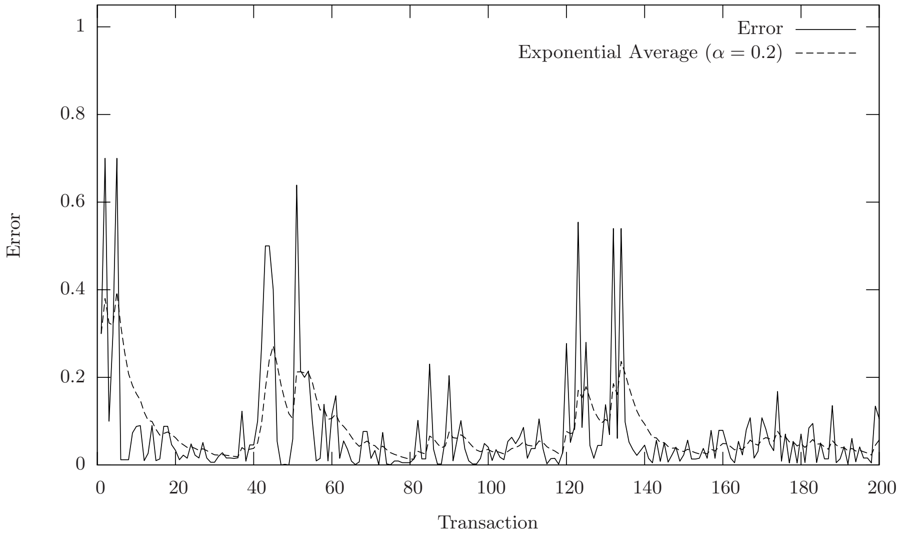


Fig. 2. Prediction errors of the learned opponent model for RECIPROCAL

Table 1. Model of Honest's behavior after 200 interactions. These learned rules describe the fact that if Honest claims that its certainty is low (c_0), its error is usually quite high (ae_5), and vice-versa.

#	Rule	Confidence
1	**if** certainty is c_0 **then** appraisal-error is ae_5	0.00381
	...	
6	**if** certainty is c_5 **then** appraisal-error is ae_0	0.00520

Table 2. Model of Reciprocal's behavior after 200 interactions. These learned rules describe how Reciprocal works approximately. For example, rule 13 describes that if the claimed certainty is moderate, and our agent was honest itself in the previous round (d_0), then the appraisal error is quite low (ae_0).

#	Rule	Confidence
1	**if** certainty is c_0 **then** appraisal-error is ae_7	0.09824
7	**if** certainty is c_6 **then** appraisal-error is ae_2	0.01403
12	**if** certainty is c_1 **and** dishonesty is d_6 **then** appraisal-error is ae_6	0.05282
13	**if** certainty is c_2 **and** dishonesty is d_0 **then** appraisal-error is ae_0	0.03136
26	**if** certainty is c_3 **and** dishonesty is d_6 **then** appraisal-error is ae_5	0.04653

Using the opponent model, the agent is able to make a decision about trusting its opponents in the next transaction. More specifically, it chooses a weight vector **w** to determine how the opponents appraisals are combined. Obviously, the optimal decision is to assign all the weight to the agent that is most skilled at appraising the painting (in which case $w_{i^*} = 1 \Leftrightarrow i^* = \arg\min_i |v - e_i|$, and $w_i = 0 \Leftrightarrow i \neq i^*$). However, because of possible noise in the environment or suboptimal learning capabilities, determining i^* is not a trivial task. Using the argumentation framework, our agent is able to find a balance between utility of the outcome of a decision, and the confidence in the knowledge used to predict these outcomes.

3.1 Scenario 1: Requester Role

In this example scenario our agent consults Honest and Reciprocal to appraise one of its paintings. For each agent, it constructs arguments to support the desirability of obtaining an estimate from both Honest and Reciprocal. The strengths of these arguments are used to determine the delegation weights $\mathbf{w} = \{w_{\text{Honest}}, w_{\text{Reciprocal}}\}$. In what follows, we focus on our agent's goals, and its available decisions. These, combined with the actual observations during a transaction determine the strength of the arguments supporting the decisions and subsequently the delegation weights.

Because it is in our agent's interest to appraise the painting as accurately as possible, its goal set $\mathcal{G}$ contains a single goal $g_1 =$ (appraisal-error is *acceptable*, 1), where *acceptable* is a fuzzy set with a membership function that is inversely proportional to the relative appraisal error $ae_i = \frac{|v-e_i|}{v}$. Put differently, goal g_1 states that our agent favors accurate appraisals from its opponents. Since g_1 is the only goal, it has maximal relative priority.

As mentioned before, the claimed certainty c is a key indicator of the expertise of agent i. In this example, suppose that Honest returns a numerical value that is 100% member of fuzzy set c_1, meaning that it is quite uncertain (see Figure 3 for the fuzzy partitioning of the certainty domain), while Reciprocal replies that it can appraise the painting with a certainty between c_4 and c_5. Also, we know that in the previous round, our agent has somewhat misrepresented its certainty towards Reciprocal (dishonesty was a member of the fuzzy set d_3).

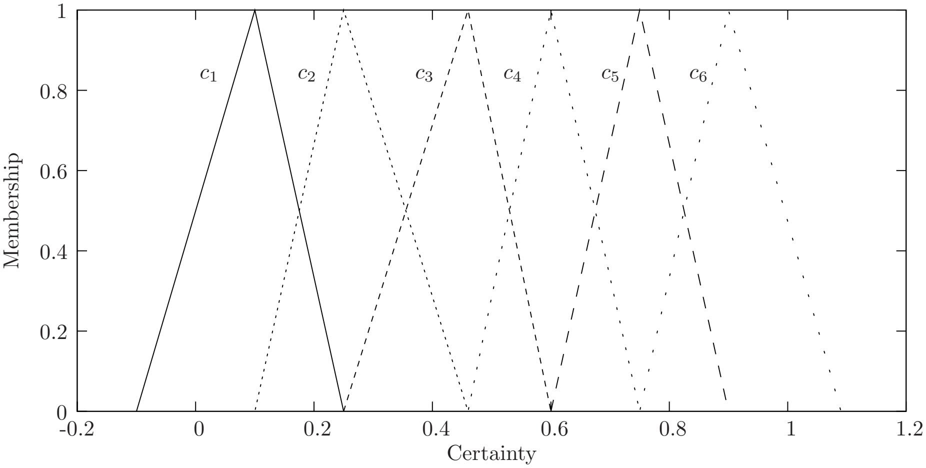


Fig. 3. Example fuzzy partitioning of the *certainty* domain

Table 3. The support for Argument $A_{\textsc{Honest}}$

Knowledge	Match	Confidence
certainty is c_1	100%	0.00832
if certainty is c_1 **then** appraisal-error is ae_4	100%	
appraisal-error is ae_4	100%	

Table 4. The consequences, and the *Level* and *Weight* calculations of argument $A_{\textsc{Honest}}$

Goal	Match	Preference
g_1	0.25	1

(a) The consequences of Argument $A_{\textsc{Honest}}$

Property	Calculation	Result
Level $(A_{\textsc{Honest}})$	1×0.00832	0.00832
Weight $(A_{\textsc{Honest}})$	1×0.25	0.25

(b) The *Level* and *Weight* calculations of Argument $A_{\textsc{Honest}}$

Our agent then has to consider two possible decisions in the set of decisions $\mathcal{D} = \{d_{\textsc{Honest}}, d_{\textsc{Reciprocal}}\}$: to accept the estimate from Honest, and to accept the estimate from Reciprocal. Because our agent can weigh the estimates received from both agents, these decisions are not mutually exclusive. For example, our agent can decide to weigh the appraisals from both agents equally, resulting in a final appraisal that is the average of both agent's appraisals.

Based on the claimed certainties from both agents, and the models in Tables 1 and 2, we see that our agent expects a poor appraisal from Honest. On the other hand, Reciprocal's certainty is very high, but our agent has to take into account its own dishonesty towards Reciprocal. Using the goals $\mathcal{G}$, the knowledge base $\mathcal{K}$ (containing the model and the observations), and decisions $\mathcal{D}$, our agent generates two arguments. The first argument $A_{\textsc{Honest}}$ supports decision $d_{\textsc{Honest}}$, the second argument $A_{\textsc{Reciprocal}}$ supports decision $d_{\textsc{Reciprocal}}$.

The support of $A_{\textsc{Honest}}$ consists of parts of the opponent model of Honest relevant to this particular transaction. This is summarized in Table 3. The *consequences* of $A_{\textsc{Honest}}$ relate to the desirability of the consequences of decision $d_{\textsc{Honest}}$ in terms of the agent's goals. For a certainty of c_1, a single rule in the opponent model fires, and predicts an appraisal error of ae_4 (last row in Table 3). Given this prediction, we can determine the utility in terms of goal g_1 (see Table 4(a)). When we defuzzify ae_4, we obtain a numerical value of 0.75.[2] Using the membership function of *acceptable*, we determine that goal g_1 is only 25% satisfied. From the information in Tables 3 and 4(a), we can now calculate the *Level,* relating to the desirability of the consequences, and *Weight,* relating to the confidence of argument $A_{\textsc{Honest}}$ (see Equation 2): Table 4(b) lists the steps for this calculation. From this, our agent can now determine the strength of the argument for Honest: $0.00832 \times 0.25 = 0.00208$ (see Definition 8).

[2] Defuzzification is a mapping from membership of one or more fuzzy sets to the original domain. There are a couple of ways to do this, but often the center of gravity of the membership functions is taken [9].

Table 5. The support for Argument $A_{\textsc{Reciprocal}}$

Knowledge	**Match**	**Confidence**
certainty is c_4	50%	
certainty is c_5	50%	
dishonesty is d_3	40%	
if certainty is c_4 **then** appraisal-error is ae_3	50%	0.00876
if certainty is c_5 **then** appraisal-error is ae_2	50%	0.01042
if certainty is c_4 **and** dishonesty is d_3 **then** appraisal-error is ae_0	40%	0.01640
if certainty is c_4 **and** dishonesty is d_3 **then** appraisal-error is ae_1	40%	0.01640

Table 6. The consequences, and the *Level* and *Weight* calculations for argument $A_{\textsc{Reciprocal}}$

Goal	**Match**	**Preference**
g_1	0.75	1

(a) The consequences of Argument $A_{\textsc{Reciprocal}}$

Property	**Calculation**	**Result**
Level($A_{\textsc{Reciprocal}}$)	0.5×0.00876	0.00438
Weight($A_{\textsc{Reciprocal}}$)	1×0.75	0.75

(b) The *Level* and *Weight* calculations of Argument A_2

Next, our agent performs the same steps for Reciprocal. For determining the support and consequences of argument $A_{\textsc{Reciprocal}}$, we follow the same procedure as above. These are summarized in Tables 5 and 6(a), respectively. This time, four rules fire based on the certainty received from Reciprocal and our agent's dishonesty towards it in the previous round. The resulting appraisal error is expected to be somewhere between ae_0 and ae_3. This corresponds with a 75% satisfaction of goal g_1. Table 6(b) shows the calculation of the *Level* and *Weight* of this argument. Based on these measures, we now calculate the strength of the argument: $0.00438 \times 0.75 = 0.00329$.

In the final step, our agent compares the strengths of both arguments. This is done in Table 7. After normalizing these strengths, we obtain the weight vector $\mathbf{w}$. From Table 7 it can be seen that Reciprocal determines 61% of the appraisal. Evidently, our agent favors a low appraisal error, while taking the reduced confidence of the knowledge of Reciprocal's behavior for granted.

In this scenario, it has been demonstrated that our agent is able to make a trade-off between an agent whose behavior can be reliably predicted (Honest) and an agent for which a less reliable opponent model is available, but which probably provides a more accurate appraisal (Reciprocal). The strengths of the arguments supporting both decision reflect this trade-off. In the end, the lower predicted appraisal error for Reciprocal proved to be decisive.

Table 7. The delegation weights for Honest and Reciprocal in scenario 1

Agent	**Level**	**Weight**	**Strength**	**Delegation weight**
Honest	0.00832	0.25	0.00208	0.39
Reciprocal	0.00438	0.75	0.00329	0.61

3.2 Scenario 2: Provider Role

In the previous scenario, we focused on the appraisals received from the opponents. Now, we also include another type of decision. Other agents may ask our agent for an appraisal. In such a situation, our agent needs to decide how truthfully it should report its estimate of the value of their paintings. To this end, we add a new goal, and apply the decision making procedure to the appraisals generated by *our agent*. The new goal, called g_2, essentially encourages our agent to be as deceptive as possible towards other agents (by overstating the accuracy of the provided estimate). However, when other agents discover this behavior, they may give our agent worse estimates as well. Deceiving other agents must not negatively influence the accuracy of appraisals received from those agents *too much*. Consequently, we must find a balance between the members of the new goal base $\mathcal{G} = \{g_1, g_2\}$.

Deciding the extent of the deception towards an opponent differs from deciding delegation weights in scenario 1 in that the certainty variable is not relevant. In the previous scenario, the agent wanted to predict the appraisal error based on the claimed certainty of its opponents. Now, the agent attempts to predict the effect of its own dishonesty on the appraisal error in the *next* round. Therefore, the opponent models in Tables 1 and 2 need first to be made independent of the certainty variable.

This is done by generating a set of arguments for each available decision for a number of *certainty* values.[3] This way, we effectively factored out the *certainty* variable from the opponent model, while the relation between *dishonesty* and *appraisal error* remains intact. The *Level* and *Weight* of each of these arguments are averaged to obtain an aggregated *Level* and *Weight*. The recommended decision is then calculated in the normal fashion. Of course, deciding on the amount of deception towards HONEST is trivial, because HONEST does not respond to the behavior of its opponents.[4] Because of this, our agent is capable of being totally dishonest with this agent, without surrendering accuracy. In what follows, we therefore illustrate this process by calculating the best level of deception towards RECIPROCAL.

In addition to goal g_1 from scenario 1, goal g_2=(dishonesty is *deceptive*, 0.5) is included in the goal base $\mathcal{G}$ of our agent. The membership of the fuzzy set *deceptive* is proportional to the extent to which the agent misrepresents its expertise by overstating its certainty. Note that goal g_2 has a lower priority than goal g_1.

We consider five different decisions: d_A, i.e. dishonesty is 0.0, d_B, i.e. dishonesty is 0.25, ..., and d_E, i.e. dishonesty is 1.0. Table 8 shows the arguments generated for each decision. We see that the extent of our agent's dishonesty towards RECIPROCAL influences the average appraisal error. Of course, due to the nature of RECIPROCAL, this is to be expected, because it punishes dishon-

[3] More specifically, we generated an argument for 100 equally spaced values of 'certainty' between 0 and 1.

[4] This is reflected in Table 1, which shows only a relation between certainty and the appraisal error.

Table 8. Properties of the set of arguments supporting different values of dishonesty towards Reciprocal

	Dishonesty	**Appraisal Error**	**Goal Satisfaction**		**Level**	**Weight**
			g_1	g_2		
d_A	0.00	0.63	0.37	0.00	1.49	0.37
d_B	0.25	0.75	0.25	0.25	1.49	0.38
d_C	0.50	0.85	0.15	0.50	1.49	0.40
d_D	0.75	0.87	0.13	0.75	1.49	0.51
d_E	1.00	0.85	0.15	1.00	1.49	0.65

esty by responding in kind. Consequently, increasing dishonesty while keeping the certainty constant, the appraisal error increases.

The interesting aspect of this scenario is the trade-off between goals g_1 and g_2. Our agent has to decide what it values most: an accurate appraisal *from*, or its deception *towards* Reciprocal. With this particular goal base and its associated priorities, we conclude from Table 8 that our agent favors the latter. Decision d_E is preferred based on the fact that it has the highest weight.

4 Discussion

In this paper we showed how arguments can be based on fuzzy rules. This generalization of Amgoud and Prade's argumentation framework [1] is able to come up with a reasoning for each of the possible decisions. We showed how the confidence and match strength of the underlying rules, and the priority of the decisions influence the decisions of our agent. Combined with a fuzzy rule learner this argumentation framework forms a unique method for agents to reason about trust, and provide a logical explanation for the actions (to be) taken.

Existing work on opponent modeling in the context of trust uses scalars or small vectors to represent trust. For example, in FIRE [14] the *quality* and the *reliability* of past transaction-results are derived and used for future decision making. An application of the Dempster-Shafer theory collects evidence of trustworthiness [15], and another approach using probabilistic reciprocity captures utility provided to and received from an agent [16], or the probability that task delegation towards an agent is successful [17]. Because of the limited amount of information present in these models, much of the information gathered during interacting with an opponent is lost. Consequently, the decision models they support are quite limited.

An example where the model of trust is more elaborate can be found in the work by Castelfranchi et al. [6,18], where trust is decomposed in distinct beliefs. Such a more complex model would open up the possibility of implementing different intervention strategies, depending on the precise composition of trust, instead of just having a binary choice: delegation or non-delegation. However in their approach the reasons *why* an agent is trusted are still not very clear. An owner of an agent that uses a so-called fuzzy cognitive map is confronted with a list of specific beliefs on parts of the model of the other agent, such as the other's competence, intentions, and reliability. It is not clear where these beliefs come from, and no method is given to learn such beliefs from past interactions. For this, we need to trace back the process that established a certain decomposition of trust for a specific agent. We believe that our approach forms a good basis to include such a more elaborate model of trust, but this may require a more advanced fuzzy rule learning algorithm.

Improving the opponent modeling algorithm is one of our goals for future work. The FURL algorithm we used in our approach has a number of limitations. Most importantly, FURL is incapable of detecting relatively complex behavior. It is not able to accurately model data sets with a large number of input variables as can be seen from the extensive experiments in our technical report [19].

In contrast to the decision model of Castelfranchi et al., the modified doxastic logic for Belief, Information acquisition, and Trust (BIT) [20] is more capable of explaining why certain facts are believed. For example, using BIT, an agent could be able to present the rationale of the decision to trust another. In terms of our aim, this is very appealing. However, due to the inherent uncertain, vague and continuous nature of observations in a Multi-Agent System it is not trivial to translate these to BIT. In this paper we showed how to make such a translation to fuzzy logic. Modal logic has no 'native' support for *directly* representing such observations, but possibly the ideas of our architecture can be reproduced in the context of modal logic.

As a final note, in the current work we have only used arguments in favor of a decision. The framework, however, also allows for contra-arguments, allowing for much more complex argumentation. Maybe even more interesting would be to add support for reputation in our approach. This would involve broadening our model, designing new algorithms to select agents from which reputation information is requested, and developing an algorithm to aggregate these reputations.

Acknowledgements

This work is partially supported by the Technology Foundation STW, applied science division of NWO, and the Ministry of Economic Affairs of the Netherlands.

References

1. Amgoud, L., Prade, H.: Using arguments for making decisions: a possibilistic logic approach. In: AUAI 2004: Proceedings of the 20th conference on Uncertainty in artificial intelligence, pp. 10–17. AUAI Press (2004)
2. Barthés, J.A., Tacla, C.: Agent-supported portals and knowledge management in complex R&D projects. In: Sixth International Conference on CSCW in Design (CSCWD 2001), pp. 287–292 (2001)
3. Maximilien, E., Singh, M.: Reputation and endorsement for web services. ACM SIGEcom Exchanges, 24–31 (2002)

4. Resnick, P., Zeckhauser, R.: Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. In: Working Paper for the NBER Workshop on Empirical Studies of Electronic Commerce (2000)
5. Guttman, R., Moukas, A., Maes, P.: Agent-mediated electronic commerce: a survey. The Knowledge Engineering Review, 147–159 (1998)
6. Castelfranchi, C., Falcone, R.: Social trust: A cognitive approach. In: Castelfranchi, C., Tan, Y. (eds.) Trust and Deception in Virtual Societies, pp. 55–90. Kluwer Academic Publishers, Dordrecht (2001)
7. Dash, R.K., Ramchurn, S.D., Jennings, N.R.: Trust-based mechanism design. In: AAMAS 2004, pp. 748–755 (2004)
8. Fullam, K., Sabater, J., Barber, K.S.: Toward a testbed for trust and reputation models. Trusting Agents for Trusting Electronic Societies, 95–109 (2005)
9. Zadeh, L.: Fuzzy sets. In: Zadeh, L.A. (ed.) Fuzzy sets, fuzzy logic, and fuzzy systems, pp. 19–34. World Scientific Publishing Co., Inc., River Edge (1996)
10. Rozich, R., Ioerger, T., Yager, R.: FURL - a theory revision approach to learning fuzzy rules. In: Proceedings of the IEEE International Conference on Fuzzy Systems, pp. 791–796 (2002)
11. Yager, R.R.: On the hierarchical structure for fuzzy modeling and control. IEEE Transactions on Fuzzy Systems 23, 1189–1197 (1993)
12. Amgoud, L., Prade, H.: Explaining qualitative decision under uncertainty by argumentation. In: Proceedings of the AAAI. AAAI Press, Menlo Park (2006)
13. Harris, C., Moore, C., Brown, M.: Intelligent Control: Aspects of Fuzzy Logic and Neural Networks. Robotics and Automated Systems. World Scientific Press, Singapore (1993)
14. Huynh, D., Jennings, N.R., Nigel, R., Shadbolt.: Developing an integrated trust and reputation model for open multi-agent systems. In: Proceedings of 7th International Workshop on Trust in Agent Societies, pp. 62–77 (2004)
15. Yu, B., Singh, M.P.: An evidential model of distributed reputation management. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems, pp. 294–301. ACM Press, New York (2002)
16. Sen, S., Dutta, P.S.: The evolution and stability of cooperative traits. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems, pp. 1114–1120. ACM Press, New York (2002)
17. Mui, L., Mohtashemi, M., Halberstadt, A.: Notions of reputation in multi-agents systems: A review. In: First International Conference on Autonomous Agents and MAS, Bologna, Italy, pp. 280–287 (July 2002)
18. Falcone, R., Pezzulo, G., Castelfranchi, C.: A fuzzy approach to a belief-based trust computation. In: Falcone, R., Singhr, M., Tan, Y.H. (eds.) Trust, reputation, and security: theories and practice, pp. 73–86. Springer, Heidelberg (2003)
19. Stranders, R.: Argumentation based decision making for trust in multi-agent systems. Master's thesis, Delft University of Technology (2006)
20. Liau, C.J.: Belief, information acquisition, and trust in multi-agent systems: a modal logic formulation. Artificial Intelligence 149(1), 31–60 (2003)

Assumption-Based Argumentation for Selection and Composition of Services

Francesca Toni

Department of Computing, Imperial College London, UK

Abstract. We present an argumentation-based approach to design and realise agents that can support the selection and composition of services in distributed environments, such as service-oriented architectures and grids. The choice of services (for selection or for composition) is equated to decisions. The agents are equipped with *beliefs*, in the form of (possibly conflicting) defeasible rules, *goals* and alternative *decisions*. Beliefs, goals, decisions may be ranked according to specified preferences. We show how beliefs and preferences can be taken into account to support the decision-making process of the agent, in order to achieve its goals. We deal with conflicts and preferences by using *assumption-based argumentation*, an existing computational-logic-based argumentation framework, that has already been proven to be an effective tool for many applications of argumentation.

1 Introduction

In recent years, under the influence of the World Wide Web, many standalone software tools have evolved into locally managed but globally accessible applications within heavily distributed environments, such as the grid and service-oriented architectures. Users are then posed the problem of selecting and/or composing these tools into complex applications, in order to fulfil specified users' needs. The use of agent technology offers a powerful solution to dynamic service composition in distributed settings such as the grid [11]. Different services can be associated with autonomous agents that can identify and negotiate, on behalf of service requestors and providers, implementation plans that take into account the requirements of both sides.

We present an argumentation-based approach to design and realise agents that can support the selection and composition of services in such distributed environments. The choice of services (for selection or for composition) is equated to decisions. The agents are equipped with

- *beliefs*, in the form of (possibly conflicting) defeasible rules and (defeasible) preferences (e.g. expressing the credibility of beliefs);
- *goals*(needs/objectives), possibly incompatible and ranked according to users' preferences (e.g. expressing how important they are to the agents); and
- alternative (and thus conflicting) *decisions*, possibly ranked according to users' preferences (e.g. expressing the cost of decisions).

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 231–247, 2008.

Beliefs and preferences need to be taken into account when supporting the decision-making process of the (user) agent, in order to achieve its (most preferred) goals. We deal with conflicts and (qualitative) preferences by using argumentation, and in particular *assumption-based argumentation* [3,6,8,9,10]. This is a general-purpose framework for argumentation whereby, differently from [7] and work following it, arguments and attack relation are not primitive concepts, but are defined instead: arguments are *backward deductions* (using sets of *rules* in an underlying logic) supported by sets of *assumptions*, and the notion of attack amongst arguments is reduced to that of *contrary* of assumptions. Assumption-based argumentation has been proven to generalise many forms of non-monotonic reasoning [3] and legal reasoning [14]. It is equipped with correct computational counterparts [8,9,10] and implementations [12], heavily based upon logic programming.

By using assumption-based argumentation, we basically define argumentative agents capable of solving multi-attribute decision-making problems, where the attributes are seen as the agents' goals, and where information (beliefs) about alternatives (decisions) is not clear-cut, and there are arguments in favour and against attributes taking one or another value. These decisions/goals need to be "weighted" against one another. Moreover, there may be "uncertainty" as to the source/validity of information, or about the current state of the world. Thus the "weight" of arguments may need to take the level of "uncertainty" into account. We represent "weights" and "uncertainty" by using qualitative preferences. We do so inpired by a number of concrete scenarios, borrowed from the ARGUGRID project [1]. These scenarios consider the need for service selection and composition to support business migration and to realise complex earth observation products. These scenarios indicate the need for dealing with possibly conflicting information (epistemic beliefs), decisions linking to goals/objectives, preferences over beliefs, goals, and decisions. In this paper we will show how all these features can be formalised and dealt with by means of assumption-based argumentation.

The paper is organised as follows. In section 2 we describe the motivating scenarios. In section 3 we give some background on assumption-based argumentation. In sections 4, 5, 6 we show how to deal with reasoning with beliefs, alternative decisions, and goals (respectively and incrementally) in assumption-based argumentation. In section 7 we conclude, summarising our contribution, related work and avenues for future research.

2 Motivating Scenarios

We consider here two example scenarios where service selection and composition are core. These scenarios are central in the ARGUGRID project [2], that aims at developing a framework supporting service selection and composition over

[1] `http://www.argugrid.eu`

[2] ARGUGRID also considers e-market places scenarios, omitted here for lack of space. Further details on all ARGUGRID scenarios can be found in [19].

the grid and/or service-oriented architectures, with the help of argumentative agents. These scenarios identify the need for a framework capable of handling

- defeasible, conflicting information (beliefs)
- preferences over beliefs
- mutually exclusive decisions for the achievement of goals
- preferences over decisions and goals.

We will see that the framework proposed in this paper will be capable of dealing with this kind of scenarios.

2.1 Business Migration

Suppose an investor decides to migrate an existing business to a new location, so as to improve profit while fulfilling certain requirements. The investor will typically have (or will be able to get hold of) some information about various alternative locations ("services") and their characteristics.

The choice (selection) of location will typically depend on a number of factors, including ease of accessibility of the location, permits, regulations, and taxes by the national/local governments, local markets and access to global markets, competitors, access to any construction contractors, access to suppliers and materials, availability of additional information or assistance, etc. For example, the investor may require that the proposed location is "easily" accessible, that the tax rate in the country of the chosen location is a maximum of 30%, that there are at least three suppliers of some required materials in the vicinity of the location, and that there are "good" assistance centres for new businesses in the chosen area.

Typically, these factors cannot be assessed with certainty for any given location, as they depend on information that is partial and resulting from sources whose reliability varies. For example, a location may be deemed "easily" accessible because close to a river of sufficient depth for water transportation, according to some source of information. However, if there is no port as yet on that river, the location would be unsuitable for water-transportation, and thus not "easily" accessible. Moreover, according to some other source, the depth of the river may be insufficient at points.

The investor may have some preferences over the requirements and the alternative locations, and these preferences may be conditional. For example, the investor may think that accessibility is a more important factor than availability of materials, as materials can be transported cheaply once the transportation network is set up. Moreover, the investor may prefer a location in a country believed (not) to have signed the Kyoto agreement for climate change control, if (not) environmentally conscious.

Once a location is selected, amongst a number of alternative candidates, several "services" need to be combined, for example a construction contractor and some suppliers. The composition of these services corresponds to a concrete plan of action for the set-up of the business.

2.2 Earth Observation

Satellites can be seen as earth observation services, useful in a number of applications (ranging from environmental monitoring, meteorology and map-making) because of their wide area observation capability, the fact that they can provide observations non-intrusively, their capability to provide measurements rapidly and continuosly.

Different earth observation satellites exist, varying, e.g., according to their orbit, according to the kind of instruments they carry on board and, for each instrument, the kind of sensors the instruments are equipped with. The instruments may allow to collect, save, and transmit data about the earth. The sensors may be classified as radar and optical. For example, some satellites (e.g. Meteosat, MGS, Eutelsat) are geostationary (namely apparently stationary wrt a given location on earth), whereas others (e.g. MetOp, Radarsat, JERS_1) are polar (revolving around earth, and passing above both poles at each resolution). Moreover, JERS_1 has radar sensors whereas Meteosat has optical sensors.

Different types of satellites have different characteristics, in terms e.g. of the resolution of images they can record and the frequency with which they can record them, as well as their cost. For example, optical sensors are guaranteed to give higher resolution images than radar sensors, but they are heavily weather-dependent, whereas radar sensors are weather-independent but limited in detecting shapes or surface variations. Also, the RADARSAT-1 satellite would charge 1000$ for emergency programming and 100$ for basic programming.

Given a specific image requirement by a user, in the simplest cases a specific satellite needs to be selected, and, in the more complex cases, a combination of satellites needs to be sought to provide a number of different images. In both cases, the choice needs to take into account the large number of available satellites and their many characteristics, as well as the requirements of the user.

For example, if the image is needed in order to study an oil spill then a single satellite would suffice, as soon as the response time is very quick and the image resolution is high. The choice amongst satellites with radar or optical sensors largely depends on the user's beliefs about the weather. Finally, the choice of the satellite depends on the funds availability of the user and the cost of the satellite. The user may have different preferences concerning its requirements (e.g. an environment-concerned user would definitely give top priority to speed over cost, in the case of an oil spill). Moreover, the user may have (possibly conflicting) beliefs about the time and location of the spill, possibly coming from different sources with different degrees of reliability.

As another example, in case of a fire monitoring scenario, several images possibly from different satellites may be required to control the spreading of a fire over a large region, in conjunction with information about weather conditions. This is an example requiring the combination of services.

3 Background: Abstract and Assumption-Based Argumentation

Definition 1. *An* abstract argumentation framework *is a pair* $(Arg, attacks)$ *where* Arg *is a finite set, whose elements are referred to as* arguments*, and* $attacks \subseteq Arg \times Arg$ *is a binary relation over* Arg*. Given sets* $X, Y \subseteq Arg$ *of arguments,* X *attacks* Y *iff there exists* $x \in X$ *and* $y \in Y$ *such that* $(x, y) \in attacks$*.*

Given an abstract argumentation framework, several notions of *"acceptable" sets of arguments* can be defined [7].

Definition 2. *A set* X *of arguments is*

- conflict-free *iff it does not attack itself;*
- admissible *iff* X *is conflict-free and* X *attacks every set of arguments* Y *such that* Y *attacks* X*;*
- preferred *iff* X *is maximally admissible;*
- sceptically preferred *iff* X *is the intersection of all preferred sets of arguments;*
- complete *iff* X *is admissible and* X *contains all arguments* x *such that* X *attacks all attacks against* x*;*
- grounded *iff* X *is minimally complete;*
- ideal *iff* X *is admissible and it is contained in every preferred set of arguments.*

The last notion was not in the original [7], but has been proposed recently [9,10] as an alternative, less sceptical semantics than the grounded semantics.

The abstract view of argumentation does not deal with the problem of actually finding arguments and attacks amongst them. Typically, arguments are built by connecting rules in the belief set of the proponent of arguments, and attacks arise from conflicts amongst such arguments. In *assumption-based argumentation*, arguments are (implicitly meant to be) obtained by reasoning backwards with a given set of inference rules (the belief set), from conclusions to premises that are assumptions, and attacks are defined in terms of a notion of "contrary" of assumptions. Belief set and backward reasoning are defined in terms of a *deductive system*:

Definition 3. *A* deductive system *is a pair* $(\mathcal{L}, \mathcal{R})$ *where*

- $\mathcal{L}$ *is a formal language consisting of countably many sentences, and*
- $\mathcal{R}$ *is a countable set of inference rules of the form*

$$\frac{x_1, \ldots, x_n}{x}$$

where $x \in \mathcal{L}$ *is called the* conclusion *and* $x_1, \ldots, x_n \in \mathcal{L}$ *are called the* premises *of the inference rule, and* $n \geq 0$*.*

If $n = 0$, then the inference rule represents an axiom. Note that a deductive system does not distinguish between domain-independent axioms/rules, which belong to the specification of the logic, and domain-dependent axioms/rules, which represent a background theory. For notational convenience, throughout the paper we write $x \leftarrow x_1, \ldots, x_n$ instead of $\dfrac{x_1, \ldots, x_n}{x}$ and x instead of $x \leftarrow$.

Definition 4. *Given a deductive system* $(\mathcal{L}, \mathcal{R})$ *and a* selection function [3] f, *a* (backward) deduction *of a conclusion* x *based on (or supported by) a set of premises* P *is a sequence of multi-sets* $S_1, \ldots, S_m$, *where* $S_1 = \{x\}$, $S_m = P$, *and for every* $1 \leq i < m$, *where* y *is the sentence occurrence in* S_i *selected by* f*:*

1. *If* y *is not in* P *then* $S_{i+1} = S_i - \{y\} \cup S$ *for some inference rule of the form* $y \leftarrow S \in \mathcal{R}$. [4]
2. *If* y *is in* P *then* $S_{i+1} = S_i$.

Each S_i *is referred to as a* step *in the deduction.*

In the remainder of this paper we will use the following notation: $P \vdash c$ will stand for "there exists a deduction of c supported by P". This notation is simplistic as it does not allow to distinguish different deductions to the same conclusions and supported by the same premises, but it is a useful shorthand when the steps in the deduction are not of interest.

Deductions are the basis for the construction of arguments in assumption-based argumentation, but to obtain an argument from a backward deduction we restrict the premises to ones that are *assumptions*. Moreover, to specify when one argument attacks another, we need to specify *contraries* of assumptions.

Definition 5. *An* assumption-based argumentation framework *is a tuple* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ *where*

- $(\mathcal{L}, \mathcal{R})$ *is a deductive system.*
- $\mathcal{A} \subseteq \mathcal{L}$, $\mathcal{A} \neq \{\}$. $\mathcal{A}$ *is referred to as the set of* assumptions.
- *If* $x \in \mathcal{A}$, *then there is no inference rule of the form* $x \leftarrow x_1, \ldots, x_n \in \mathcal{R}$.
- $\overline{}$ *is a (total) mapping from* $\mathcal{A}$ *into* $\mathcal{L}$. $\overline{x}$ *is referred to as the* contrary *of* x.

Note that assumption-based frameworks are still abstract, in the sense that in order to be deployed they need to be instantiated. Several instances have been studied already [3,14]. In this paper we study some additional instances, for epistemic and practical reasoning. Note that, by the third bullet, following [8] we restrict ourselves to *flat* frameworks [3], whose assumptions do not occur as

[3] A selection function is any function from sets of elements to elements. The definition of backward deduction relies upon some chosen selection function. However, note that if a backward deduction for a conclusion exists for some selection function, then a backward deduction for that conclusion will exist for any other selection function. This result follows from the analogous result for SLD-resolution for Horn clauses.

[4] The same symbols are used for multi-set membership, union, intersection and subtraction as for ordinary sets.

conclusions of inference rules. Flat frameworks are restricted but still interesting and general, as, for example, they admit default logic and logic programming as concrete instances [3], as well as all the instances we will consider in this paper.

In the assumption-based approach to argumentation, arguments are deductions to conclusions, based solely upon assumptions, and the attack relationship between arguments depends solely on the contrary of assumptions.

Definition 6. *A* set of assumptions *A* attacks a set of assumptions *B iff there exists an assumption $x \in B$ and a deduction $A' \vdash \overline{x}$ such that $A' \subseteq A$: if this is the case, we say that A* attacks *B* on *x.*

This notion of attack between sets of assumptions implicitly gives a notion of attack between arguments supported by sets of assumptions: the attacking argument needs to have as conclusion the contrary of an assumption in the support of the attacked argument.

Within assumption-based argumentation, implicitly, a set of assumptions stands for the set of all arguments whose premises are contained in the given set of assumptions. Thus, the computation of "acceptable" sets of arguments amounts to computing "acceptable" sets of assumptions:

Definition 7. *A set X of assumptions is*

- conflict-free *iff X does not attack itself;*
- admissible *iff X conflict-free and X attacks every set of assumptions Y that attacks X;*
- preferred *iff it is maximally admissible;*
- sceptically preferred *iff X is the intersection of all preferred sets of assumptions;*
- complete *iff it is admissible and contains all assumptions x such that X attacks all attacks against $\{x\}$;*
- grounded *iff it is minimally complete;*
- ideal *iff X is admissible and it is contained in every preferred set of assumptions.*

4 Reasoning about Beliefs

Reasoning about beliefs may be performed within a framework consisting of defeasible and strict rules and facts [17], some of which may express preferences over the application of rules and the use of facts (thus some of these preferences may be themselves defeasible). The use of rules to represent preferences rather than fixed partial orders is advocated by many, e.g. [4,18], driven by the requirements of applications, for example in a legal domain. We define here frameworks for reasoning about beliefs (referred to as *epistemic frameworks*) based upon defeasible rules and preferences, ignoring for simplicity strict rules. [5] We start by giving some preliminary notions.

[5] Strict rules require special attention to guarantee "closedness" and "consistency" of epistemic reasoning [5]. For a treatment of strict rules in epistemic frameworks built upon assumption-based argumentation see [20].

Definition 8

- *A* language $\mathcal{L}_{ag}$ *is a set of ground literals, which can be atoms a or negations of atoms $\neg a$. We will refer to these literals as* basic literals.
- *A* naming $\mathcal{N}$ *is a bijective function associating a distinguished name $\mathcal{N}(x)$ to any element x in a given domain X. For any given X, we will refer to the set of all such names as $\mathcal{N}(X)$.*
- *A* preference literal (wrt X and $\mathcal{N}$) *is of the form $n_1 \succ n_2$ where n_1, n_2 are (different) names in $\mathcal{N}(X)$.*
- *A* literal *is either a basic or a preference literal.*

Intuitively, $\mathcal{L}_{ag}$ is the language underlying the agent's reasoning problem, in the chosen domain. When X is a given set of rules over $\mathcal{L}_{ag}$, $n_1 \succ n_2$ stands for "the rule named n_1 is preferred to the rule named n_2". In the remainder of this paper, given a basic literal l, with an abuse of notation, $\neg l$ will stand for the complement of l, namely $\neg l$ if l is an atom, and a if l is a negative literal $\neg a$. Moreover, given a preference literal l of the form $n_1 \succ n_2$, $\neg l$ will stand for $n_2 \succ n_1$.

Definition 9. *Given a language $\mathcal{L}_{ag}$ and a naming $\mathcal{N}$:*

- *A* basic rule *(wrt $\mathcal{L}_{ag}$) is of the form $P \rightarrow c$ where $P = l_1, \ldots, l_n$ and $c, l_1, \ldots, l_n$ are basic literals in $\mathcal{L}_{ag}$ and $n \geq 0$.*
- *A* preference rule (wrt some given X and $\mathcal{N}$) *is of the form $P \rightarrow c$ where c is a preference literal (wrt X and $\mathcal{N}$), $P = l_1, \ldots, l_n$ and $l_1, \ldots, l_n$ are basic literals in $\mathcal{L}_{ag}$ or preference literals (wrt X and $\mathcal{N}$), and $n \geq 0$.*
- *A* (defeasible) rule *is either a basic rule or a preference rule.*

Given a rule $P \rightarrow c$, c is referred to as the conclusion *and P as the* premises. *When $n = 0$ the rule may be referred to as a* fact.

Definition 10. *Let $\mathcal{L}_{ag}$ be a language and $\mathcal{N}$ be a naming. An* epistemic framework *ϵ is a set E of* defeasible *rules that can be partitioned into sets $E_1, \ldots, E_d$, $d \geq 0$, such that*

- *E_1 is a set of basic rules (wrt $\mathcal{L}_{ag}$);*
- *for each $i \geq 1$, E_i is a set of preference rules wrt $X = \cup_{j=1,\ldots,i-1} E_j$ and $\mathcal{N}$.*

Intuitively, defeasible rules may or may not be chosen by a rational reasoner, depending on the emergence of conflicts. A rational reasoner needs to avoid these conflicts in its chosen "reasoning lines". Conflicts in epistemic frameworks arise from "deriving" complementary conclusions from sets of chosen (defeasible) rules, either of the form a and $\neg a$ or of the form $n_1 \succ n_2$ and $n_2 \succ n_1$. The semantics of epistemic frameworks needs to resolve these conflicts. In the remainder of this section, we will show how to provide this semantics for epistemic frameworks by means of assumption-based argumentation, by first considering frameworks without preference rules.

Below, we will assume given an epistemic framework $\epsilon{=}E$ wrt a language $\mathcal{L}_{ag}$ and a naming $\mathcal{N}$. When we restrict attention to epistemic frameworks without preference rules, $\mathcal{L}_{ag}$ will consist solely of basic literals (and $\mathcal{N}$ will be ignored).

Definition 11. *The assumption-based framework corresponding to an epistemic framework without preferences $\epsilon{=}E$ is $\langle \mathcal{L}_\epsilon, \mathcal{R}_\epsilon, \mathcal{A}_\epsilon, \overline{} \rangle$ whereby*

- *$\mathcal{A}_\epsilon$ is a set of literals not already in $\mathcal{L}_{ag}$ such that there exists a bijective mapping α from rules in E into $\mathcal{A}_\epsilon$;*
- *$\mathcal{L}_\epsilon{=}\mathcal{L}_{ag} \cup \mathcal{A}_\epsilon$;*
- *$\mathcal{R}_\epsilon = \{c \leftarrow P, \alpha(P \rightarrow c) | P \rightarrow c \in E\}$*
- *$\overline{\alpha(P \rightarrow c)} = \neg c$.*

Intuitively, any assumption in $\mathcal{A}_\epsilon$ correspond to the applicability of the corresponding rule, which is opposed by the complement of the conclusion of that rule being "derivable": this is expressed by the definition of contrary.

Example 1. Consider $E = \{q; q \rightarrow p; r; r \rightarrow \neg p\}$. In the context of the business migration scenario of section 2, p may represent that some given country of interest has signed the Tokyo protocol. There is evidence from some source q that this is so ($q \rightarrow p$), and evidence from some other source r that this is not so ($r \rightarrow \neg p$). The sources (q and r) are defeasible, and so are the reports claiming that these sources provided support for p and $\neg p$ (that the given country signed/did not sign the Tokyo protocol respectively).

Given this E, we can choose $\mathcal{A}_\epsilon{=}\{a_1, a_2, a_3, a_4\}$[6], and then $\mathcal{R}_\epsilon{=}\{q \leftarrow a_1;\ p \leftarrow q, a_2; r \leftarrow a_3;\ \neg p \leftarrow r, a_4\}$, and $\overline{a_1} = \neg q$, $\overline{a_2} = \neg p$, $\overline{a_3} = \neg r$, and $\overline{a_4} = p$.

By virtue of the formulation in definition 11, any notion of acceptable set of assumptions may be adopted to provide a semantics to E. For instance, in example 1, $\{a_1, a_2\}$ is an admissible set of assumptions with $\{q, p\}$ the corresponding "output" beliefs, and $\{a_1, a_3\}$ is the grounded set of assumptions with $\{q, r\}$ the corresponding "output" beliefs.

Let us now consider epistemic frameworks with preference rules. We will use the following notation: given a set of rule X, a literal L is *defined* in X if a rule in X has L or $\neg L$ as its conclusion.

Definition 12. *The assumption-based framework corresponding to an epistemic framework with preferences $\epsilon{=}E$ is $\langle \mathcal{L}_\epsilon, \mathcal{R}_\epsilon, \mathcal{A}_\epsilon, \overline{} \rangle$ whereby*

- *$\mathcal{A}_\epsilon$ is a set of literals not already in $\mathcal{L}_{ag}$ such that there exists a bijective mapping α from rules in E into $\mathcal{A}_\epsilon$;*
- *$\mathcal{L}_\epsilon{=}\mathcal{L}_{ag} \cup \mathcal{A}_\epsilon \cup \mathcal{B}_\epsilon \cup \mathcal{C}_\epsilon$ where $\mathcal{B}_\epsilon$ and $\mathcal{C}_\epsilon$ are distinct sets of literals not already in $\mathcal{L}_{ag} \cup \mathcal{A}_\epsilon$ such that*
  - *there exists a bijective mapping β from rules in E into $\mathcal{B}_\epsilon$;*
  - *there exists a bijective mapping χ from assumptions in $\mathcal{A}_\epsilon$ into $\mathcal{C}_\epsilon$;*

[6] Note that the specification of $\mathcal{R}_\epsilon$ depends on the choice of $\mathcal{A}_\epsilon$, and for each different choice of $\mathcal{A}_\epsilon$ a different $\mathcal{R}_\epsilon$ needs to be given. However, all alternative choices are isomorphic.

- $\mathcal{R}_\epsilon = \{c \leftarrow \beta(P \rightarrow c) | P \rightarrow c \in E\} \cup$
  $\{\beta(P \rightarrow c) \leftarrow P, \alpha(P \rightarrow c) | P \rightarrow c \in E\} \cup$
  $\{\chi(a) \leftarrow n' \succ n, \beta(P' \rightarrow \neg c) |\, a = \alpha(P \rightarrow c), P \rightarrow c \in E, P' \rightarrow \neg c \in E,$
  $n = \mathcal{N}(P \rightarrow c),\ n' = \mathcal{N}(P' \rightarrow \neg c),$
  $n' \succ n \textit{ is defined in } E\} \cup$
  $\{\chi(a) \leftarrow \beta(P' \rightarrow \neg c) |\, a = \alpha(P \rightarrow c), P \rightarrow c \in E, P' \rightarrow \neg c \in E,$
  $\mathcal{N}(P \rightarrow c) \succ \mathcal{N}(P' \rightarrow \neg c) \textit{ is not defined in } E\}$;
- $\overline{a} = \chi(a)$.

Intuitively, assumptions in $\mathcal{A}_\epsilon$, as in the case of no preference rules, correspond to the applicability of the corresponding rules, sentences in $\mathcal{B}_\epsilon$ correspond to the actual application of the corresponding rules, and sentences in $\mathcal{C}_\epsilon$ correspond to objecting to the application of a rule, by a rule with higher preference and conflicting conclusion being "derivable" (if any) or simply by a rule with conficting conclusion (if no preference is defined): this is expressed by the definition of contrary.

Example 2. Consider $E = \{q \rightarrow p; q; \neg p; \neg q; r \rightarrow n_1 \succ n_3; r; \neg r\}$, where $n_1 = \mathcal{N}(q \rightarrow p)$ and $n_3 = \mathcal{N}(\neg p)$. Here, similarly to example 1, p may represent, in the context of the business migration scenario of section 2, that some given country of interest has signed the Tokyo protocol, with $q \rightarrow p$ representing that there is evidence that some source q claims that this is indeed so. The fact that the source q is reliably making this claim is debatable (both q and $\neg q$ are rules). The rule $\neg p$ may represent a defeasible belief that the country of interest has not signed the Tokyo protocol, and the preference rule $r \rightarrow n_1 \succ n_3$ may represent that there is evidence from some source r supporting that the rule leading to conclude p is stronger (more reliable than) the rule concluding $\neg p$.

Given this E, $\mathcal{A}_\epsilon$ may be $\{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$, $\mathcal{R}_\epsilon$ is

$$\begin{aligned}
\{\ & p \leftarrow b_1;\ \ b_1 \leftarrow q, a_1;\ \ c_1 \leftarrow n_3 \succ n_1, b_3;\\
& q \leftarrow b_2;\ \ b_2 \leftarrow a_2;\ \ c_2 \leftarrow b_4;\\
& \neg p \leftarrow b_3;\ \ b_3 \leftarrow a_3;\ \ c_3 \leftarrow n_1 \succ n_3, b_1;\\
& \neg q \leftarrow b_4;\ \ b_4 \leftarrow a_4;\ \ c_4 \leftarrow b_2;\\
& n_1 \succ n_3 \leftarrow b_5;\ \ b_5 \leftarrow r, a_5;\\
& r \leftarrow b_6;\ \ b_6 \leftarrow a_6;\ \ c_6 \leftarrow b_7;\\
& \neg r \leftarrow b_7;\ \ b_7 \leftarrow a_7;\ \ c_7 \leftarrow b_6\}
\end{aligned}$$

and $\overline{a_i} = c_i$, for $i = 1, \ldots, 7$.

By virtue of the formulation in definition 12, any notion of "acceptable" set of assumptions may be adopted to provide a semantics for E. For instance, in example 2, $\{a_1, a_2\}$ is an admissible set of assumptions, with "output" beliefs $\{p, q\}$. Indeed, this set of assumptions is conflict-free (there is no backward deduction from any of its subsets supporting any of c_1, c_2). Moreover, it attacks (by means of a deduction supporting c_4) the set of assumptions $\{a_4\}$ that attacks it (by means of a deduction supporting c_2). Note that the assumption a_1 is not attacked by any set of assumptions, as there is no deduction supporting c_1. The set of assumptions $\{a_3, a_4\}$ is also admissible, as it is conflict-free and it attacks all attacks against it: it is attacked by

- $\{a_1, a_2, a_5, a_6\}$ (supporting a deduction for c_3), counter-attacked by $\{a_4\}$,
- $\{a_2\}$ (supporting a deduction for c_4), also counter-attacked by $\{a_4\}$.

5 Reasoning about Decisions

Definition 13. *Given $\mathcal{L}_{ag}$, a* practical framework *is a tuple $\langle \epsilon, \mathcal{D}, \mathcal{G}, D \rangle$ where*

- *ϵ is an epistemic framework wrt $\mathcal{L}_{ag}$;*
- *$\mathcal{D} \subseteq \mathcal{L}_{ag}$ is the set of* potential decisions *such that none of its elements occurs in the conclusion of any rule in ϵ;*
- *$\mathcal{G} \subseteq \mathcal{L}_{ag}$ is the set of* goals*;*
- *D is a set of preference rules wrt $\mathcal{D}$ and some naming (and wrt $\mathcal{L}_{ag}$).*

In the remainder of the paper, we will assume that the naming is the identity function, associating d to each decision $d \in \mathcal{D}$.

Intuitively, the agent's task is to choose amongst its potential decisions in $\mathcal{D}$ so that its goals in $\mathcal{G}$ are achieved. These decisions are intended to be mutually exclusive alternatives, that can be understood as possible plans of actions. The goals are objectives/constraints of the agent, namely features that the agent would like the decisions to exhibit.

The agent's preferences amongst these decisions are represented within D. These preferences may be expressing a partial order, when D only consists of facts (and special care is taken so that antisymmetry holds).More generally, D is a set of preference rules, e.g. $P \rightarrow d_1 \succ d_2$, expressing that a decision d_1 is preferred to another decision d_2 under certain circumstances P. These circumstances are represented by means of literals in $\mathcal{L}_{ag}$, and will typically be beliefs to be evaluated wrt ϵ.

From now on we will assume given $\langle \epsilon, \mathcal{D}, \mathcal{G}, D \rangle$ wrt $\mathcal{L}_{ag}$. We will first consider the case where D=$\{\}$, and then the general case.

Frameworks for practical reasoning can be modelled naturally within *generalised assumption-based frameworks* [7], whereby contrary is a (total) mapping from assumptions into *sets* of sentences in $\mathcal{L}$ (rather than individual sentences in $\mathcal{L}$). Given such a generalised framework, the notion of attack between sets of assumptions is modified as follows

- a set of assumptions X attacks a set of assumptions Y iff there exists an assumption $x \in Y$, a sentence $y \in \overline{x}$ and an argument $X' \vdash y$ such that $X' \subseteq X$.

[7] Note that this generalisation is strictly speaking not necessary, and practical frameworks could be expressed also within the orginal form of assumption-based frameworks, by introducing new sentences and rules in the underlying deductive system, but somewhat less naturally.

Definition 14. *The (generalised) assumption-based framework corresponding to* $\pi{=}\langle\epsilon, \mathcal{D}, \mathcal{G}, \{\}\rangle$ *is* $\langle\mathcal{L}_\pi, \mathcal{R}_\pi, \mathcal{A}_\pi, \overline{}\rangle$ *whereby, given that* $\langle\mathcal{L}_\epsilon, \mathcal{R}_\epsilon, \mathcal{A}_\epsilon, \overline{}\rangle$ *is the assumption-based framework corresponding to* ϵ*,*

- $\mathcal{A}_\pi = \mathcal{A}_\epsilon \cup \mathcal{D}$*;*
- $\mathcal{L}_\pi{=}\mathcal{L}_\epsilon$*;*
- $\mathcal{R}_\pi{=}\mathcal{R}_\epsilon$*;*
- *if* $x \in \mathcal{A}_\epsilon$*, then* $\overline{x} = \{y\}$ *where* y *is the contrary of* x *in* $\langle\mathcal{L}_\epsilon, \mathcal{R}_\epsilon, \mathcal{A}_\epsilon, \overline{}\rangle$*; if* $x \in \mathcal{D}$*, then* $\overline{x} = \mathcal{D} - \{x\}$*.*

Intuitively, the decisions correspond to new assumptions, reflecting the "abductive" nature of decision-making. The mutual exclusion amongst decisions is achieved by setting the contrary of decisions to all other decisions.

Below, whenever $\overline{x} = \{y\}$, for $x, y \in \mathcal{L}_{ag}$, we will write simply $\overline{x} = y$.

Example 3. Consider $\pi{=}\langle\epsilon, \mathcal{D}, \mathcal{G}, \{\}\rangle$ where $\epsilon{=}E = \{p; d_1, p \rightarrow q; d_2 \rightarrow s; s \rightarrow t; t \rightarrow \neg p; d_1 \rightarrow \neg p\}$ and $\mathcal{D} = \{d_1, d_2\}$. Here, again in the context of the business migration scenario of section 2, d_1 and d_2 may represent two alternative locations for the migration of a business, each having benefits (s for d_2, $\neg p$ for d_1 and, if the belief p is held, also q for d_1; each of these benefits may have repercussions). Then, in $\langle\mathcal{L}_\pi, \mathcal{R}_\pi, \mathcal{A}_\pi, \overline{}\rangle$, $\mathcal{R}_\pi$ is [8]:

$$p \leftarrow a_1; \quad q \leftarrow p, d_1, a_2; \quad s \leftarrow d_2, a_3; \quad t \leftarrow s, a_4; \quad \neg p \leftarrow t, a_5; \quad \neg p \leftarrow d_1, a_6$$

where $\mathcal{A}_\pi = \{a_1, \ldots, a_6, d_1, d_2\}$ and $\overline{a_1} = \neg p$, $\overline{a_2} = \neg q$, $\overline{a_3} = \neg s$, $\overline{a_4} = \neg t$, $\overline{a_5} = \overline{a_6} = p$, $\overline{d_1} = d_2$, $\overline{d_2} = d_1$.

By virtue of this formulation, any notion of "acceptable" set of assumptions may be adopted to provide a semantics to $\langle\epsilon, \mathcal{D}, \mathcal{G}, D\rangle$. For instance, in example 3, $\{a_1, a_2, d_1\}$ and $\{a_3, a_4, a_5, d_2\}$ are both admissible sets of assumptions, with corresponding "outputs" $\{p, q\}$ and $\{s, t, \neg p\}$, respectively.

Amongst all acceptable sets of arguments, we want to consider solely those having the goals in $\mathcal{G}$ in their "output".

Definition 15. *Given* $\langle\mathcal{L}_\pi, \mathcal{R}_\pi, \mathcal{A}_\pi, \overline{}\rangle$*, an "acceptable" set of assumptions* Δ *wrt* $\langle\mathcal{L}_\pi, \mathcal{R}_\pi, \mathcal{A}_\pi, \overline{}\rangle$ *is* desired *iff* $\mathcal{G} \subseteq O(\Delta)$*, where* $O(\Delta) = \{x \in \mathcal{L}_\pi | \Delta' \vdash_\pi x, \Delta' \subseteq \Delta\}$*, with* $\Delta' \vdash_\pi x$ *standing for "there exists a deduction of* x *supported by* Δ'*" wrt* $\langle\mathcal{L}_\pi, \mathcal{R}_\pi, \mathcal{A}_\pi, \overline{}\rangle$*.*

Thus, practical reasoning may be realised within assumption-based argumentation by identifying acceptable sets of assumptions that contain a support for the desired goals. For instance, given $\mathcal{G} = \{\neg p\}$ in example 3, $\{a_3, a_4, a_5, d_2\}$ is desired admissible, whereas $\{a_1, a_2, d_1\}$ is not.

[8] We adopt here the simpler translation given in section 4, as there are no preference rules in ϵ. Note that the specification of $\mathcal{R}_\pi$ depends on the choice of $\mathcal{A}_\pi$, and for each different choice of $\mathcal{A}_\pi$ a different $\mathcal{R}_\pi$ needs to be given. However, all alternative choices are isomorphic.

Note that $\mathcal{G}$ may contain positive and negative literals. Thus, implicitly, $\mathcal{G}$ specifies properties that need to be fulfilled (the positive literals) and properties that need to be avoided (the negative literals).

Let us consider now the general case of practical frameworks $\langle \epsilon, \mathcal{D}, \mathcal{G}, D \rangle$ with any, possibly non-empty D. The translation of definition 14 can be generalised so that rules in D are treated in a similar manner as preference rules in ϵ. This is illustrated by the following example.

Example 4. Consider $\pi{=}\langle \epsilon, \mathcal{D}, \mathcal{G}, D \rangle$ where ϵ and $\mathcal{D}$ are as in example 3, and $D = \{u \rightarrow d_1 \succ d_2; v \rightarrow d_2 \succ d_1\}$, where u, v are new atoms of $\mathcal{L}_{ag}$. Then, in $\langle \mathcal{L}_\pi, \mathcal{R}_\pi, \mathcal{A}_\pi, \overline{} \rangle$, $\mathcal{R}_\pi$ is

$$
\begin{array}{l}
p \leftarrow a_1; \quad q \leftarrow p, d_1, a_2; \quad s \leftarrow d_2, a_3; \quad t \leftarrow s, a_4; \quad \neg p \leftarrow t, a_5; \quad \neg p \leftarrow d_1, a_6; \\
d_1 \succ d_2 \leftarrow b_7; \quad b_7 \leftarrow u, a_7; \quad c_7 \leftarrow b_8; \\
d_2 \succ d_1 \leftarrow b_8 \quad b_8 \leftarrow v, a_8; \quad c_8 \leftarrow b_7; \\
e_1 \leftarrow d_2 \succ d_1; \quad e_1 \leftarrow d_2; \\
e_2 \leftarrow d_1 \succ d_2; \quad e_2 \leftarrow d_1
\end{array}
$$

where $\mathcal{A}_\pi = \{a_1, \ldots, a_8, d_1, d_2\}$ and $\overline{a_1} = \neg p$, $\overline{a_2} = \neg q$, $\overline{a_3} = \neg s$, $\overline{a_4} = \neg t$, $\overline{a_5} = \overline{a_6} = p$, $\overline{a_7} = c_7$, $\overline{a_8} = c_8$, $\overline{d_1} = e_2$, $\overline{d_2} = e_1$. The translation introduces new literals (e_1, e_2) for the contraries of decisions and new definitions for these contraries (last two lines in the description of $\mathcal{R}_\pi$) preventing decisions being made if more preferred decisions can be made too, and in any case preventing incompatible decisions. In the example, if u is an additional fact in ϵ, then d_1 is the only decision possible to achieve the goal $\neg p$ according to any argumentation semantics for $\langle \mathcal{L}_\pi, \mathcal{R}_\pi, \mathcal{A}_\pi, \overline{} \rangle$.

Due to lack of space, we omit here the formal translation into assumption-based argumentation for $\langle \epsilon, \mathcal{D}, \mathcal{G}, D \rangle$ with a non-empty D.

So far we have assumed that decisions are all in a single pool $\mathcal{D}$, but in general there could be sets of sets of potential decisions, and composite decisions resulting from choosing one element for each set. For example, for service composition, two services may be needed, and for each of these services a number of possible choices may be available. This can be accommodated in a straightforward manner, as follows:

- $\mathcal{D}$ is partitioned into subsets $\mathcal{D}_1, \ldots, \mathcal{D}_n$, $n \geq 1$;
- in $\langle \mathcal{L}_\pi, \mathcal{R}_\pi, \mathcal{A}_\pi, \overline{} \rangle$, all elements of $\mathcal{D}_i$ are assumptions in $\mathcal{A}_\pi$;
- in $\langle \mathcal{L}_\pi, \mathcal{R}_\pi, \mathcal{A}_\pi, \overline{} \rangle$, for each i, for each $x \in \mathcal{D}_i$, the contrary of x is set to $\mathcal{D}_i - \{x\}$;
- the preference rules in D may be given only for decisions in the same element of the partition of $\mathcal{D}$.

For example, in the context of the business migration scenario of section 2, there may $\mathcal{D}_1$ and $\mathcal{D}_2$, the first representing a number of alternatives for building constructors, the second a number of alternatives for suppliers of materials. We omit further details due to lack of space.

6 Reasoning about Goals

So far we have assumed that all goals in $\mathcal{G}$ that the decisions aim at achieving are equally important. Also, in some cases no desired "acceptable" decision (set of assumptions) may exist, e.g. if the goals are incompatible. This may happen in example 3, for instance, given $\mathcal{G} = \{\neg p, q\}$. The use of stratification or, more generally, user-defined preferences over $\mathcal{G}$ will help in general with identifying desired "acceptable" sets. For example, if $\mathcal{G} = \{\neg p, q\}$ and $\neg p \succ q$ (namely the goal $\neg p$ is more preferred than the goal q), then $\{a_3, a_4, a_5, d_2\}$ is desired admissible, if instead $q \succ \neg p$ (namely the goal q is more preferred than the goal $\neg p$), then $\{a_1, a_2, d_1\}$ is desired admissible.

Preferences may be seen as providing several layers of importance for goals (from "must have" to various degrees of "wish to have"). In the case of preferences providing a fixed partial order (given by means of facts, with appropriate antysymmetry), the preferences provide a partition/stratification whereby the highest layer correspond to properties whose achievement/avoidance is a must for the agent, and the remaining layers correspond to properties whose achievement/avoidance is a wish for the agent, but that can be overlooked in favour of properties in higher layers.

In the sequel, we will refine the practical frameworks of definition 13 to allow for reasoning about goals, taking into account preferences.

Definition 16. *Given $\mathcal{L}_{ag}$, a* (full) practical framework *is a tuple $\langle \epsilon, \mathcal{D}, \mathcal{G}, D, G \rangle$ where*

- *ϵ, $\mathcal{D}$, $\mathcal{G}$ and D are as in definition 13 and*
- *G is a set of preference rules wrt $\mathcal{G}$ and some naming (and wrt $\mathcal{L}_{ag}$).*

In the remainder of the paper, we will assume that the naming is the identity function, associating g to each goal $g \in \mathcal{G}$. Moreover, for simplicity, until section 7, we will ignore preferences over decisions and assume that D=$\{\}$. The translation of definition 14 can be generalised so that rules in G are treated in a similar manner as preference rules in ϵ as in definition 12. This is illustrated by the following example.

Example 5. Consider π=$\langle \epsilon, \mathcal{D}, \mathcal{G}, \{\}, G \rangle$ where ϵ=$\{d_1 \rightarrow p; d_2 \rightarrow \neg p\}$, $\mathcal{D}$=$\{d_1, d_2\}$, $\mathcal{G} = \{p, \neg p\}$ and $G = \{u \rightarrow p \succ \neg p; v \rightarrow \neg p \succ p\}$, where $\mathcal{L}_{ag}$=$\{p, \neg p, u, v, d_1, d_2\}$. For example, in the context of the business migration scenario of section 2, p may represent ease of accessibility by land, guaranteed for location d_1 ($d_1 \rightarrow p$) but not true for location d_2 ($d_2 \rightarrow \neg p$). The investor would prefer p or $\neg p$ depending on whether it will transport goods by air (u) or by land (v), respectively.

Then, in $\langle \mathcal{L}_\pi, \mathcal{R}_\pi, \mathcal{A}_\pi, \overline{} \rangle$, $\mathcal{R}_\pi$ is

$$\begin{array}{llll}
p \leftarrow b_1; & b_1 \leftarrow d_1, a_1; & c_1 \leftarrow \neg p \succ p; & c_1 \leftarrow b_2 \\
\neg p \leftarrow b_2; & b_2 \leftarrow d_2, a_2; & c_2 \leftarrow p \succ \neg p; & c_2 \leftarrow b_1 \\
p \succ \neg p \leftarrow b_3; & b_3 \leftarrow u, a_3; & c_3 \leftarrow b_4; & \\
\neg p \succ p \leftarrow b_4 & b_4 \leftarrow v, a_4; & c_4 \leftarrow b_3; &
\end{array}$$

where $\mathcal{A}_\pi = \{a_1, \dots, a_4, d_1, d_2\}$ and $\overline{a_i} = c_i$, for all $i = 1, \dots, 4$. The translation prevents goals for being achieved (by making the appropriate decisions) when more preferred goals can also be achieved. It also prevents, in case no preferences can be applied, that conflicting goals can be achieved. In the example, if u is an additional fact in ϵ, then p is the only goal to be pursued, and decision d_1 needs to be made, according to any argumentation semantics for $\langle \mathcal{L}_\pi, \mathcal{R}_\pi, \mathcal{A}_\pi, \overline{} \rangle$.

So far we have assumed that the set of goals $\mathcal{G}$ is fixed. However, in general agents may need to adopt goals dynamically, e.g. as a reaction to observations that the agent makes in its environment. For example, in the case of business relocation, a financial crisis may force the agent to reconsider its investment goals. This could be accommodated, e.g. following [13], by replacing $\mathcal{G}$ with a set of defeasible rules (whose conclusions would be potential goals) with preferences, allowing to determine the most important goals for the agent to adopt at any given time.

7 Conclusions

We have shown how assumption-based argumentation can support reasoning with defeasible (uncertain) conflicting beliefs, alternative decisions, and possibly incompatible goals, all ranked using dynamic, qualitative preferences. This reasoning is core for service selection and composition in general, and in the motivating scenarios (business migration and earth observation applications) in particular.

The problem we have analysed is an example of argumentation for practical reasoning, trying to answer questions such as: "how to achieve a given purpose best, given some circumstances?" [2]. Since the value of the required goals cannot be evaluated with certainty, conventional multi-issue decision making techniques cannot be directly applied. Our approach realies upon (various mappings onto) assumption-based argumentation. As a by-product of these mappings, the computational techniques [8,9,10,12] for assumption-based argumentation can be directly deployed to provide a computational counterpart of our approach and realise argumentative agents for service selection and composition.

We have focused on the decision-making for selection and composition of services given some information (beliefs) available to the agents. Instead, within the larger ARGUGRID picture and in other work (e.g. [16,15]), agents communicate with one another in order to exchange/evaluate information acquired dynamically. In particular, it would be interesting to study the possibility for arguments to be exchanged amongst agents and with the users, as justifications for the agents' decisions. The communicatin with the user would be important, e.g. for business migration where it is important that the investor is presented with an appropriate analysis of the advantages and disadvantages of potential choices/decisions.

Our work also has a number of other limitations. First, we see decisions as full plans. However, they could be *high-level* decisions, which could in turn be definable in terms of *atomic decisions*, namely the actions composing the plans, in a hierachical fashion. Moreover, we ignore the possibility that in general actions may be definiable in terms of other actions, as in “counts-as” rules. Furthermore, we have not studied the interaction between preferences of goals and decisions: e.g., how to deal with cases where one goal is preferred to another, but the decision to achieve the first is less preferred than the one to achieve the second?

There are a number of approaches for reasoning about beliefs and goals, we will focus our comparison here on approaches using argumentation. For example, [1] uses argumentation in a similar context as the one we have considered here, but considering stratification over goals and beliefs. [2] also sees decisions as assumptions, but uses an accrual mechanism to determine the best decision. To the best of our knowledge, no existing approach considers dynamic preferences for beliefs, decisions, goals, of the type we have considered here, although [13] does consider dynamic preferences over goals. Finally, some work exists in dealing with service composition in an agent-oriented setting. For example, [15] proposes to use games for service composition, focusing on interactions amongst agents and in particular evaluating their competence to provide services.

Acknowledgements

The author has been supported by a UK Royal Academy of Engineering/ Leverhulme Trust senior fellowship and by the Sixth Framework IST programme of the EC, under the 035200 ARGUGRID project.

References

1. Amgoud, L.: A unified setting for inference and decision: an argumentation-based approach. In: Proc. IJCAI Workshop on Computational Models of Natural Arguments (2005)
2. Bench-Capon, T., Prakken, H.: Justifying actions by accruing arguments. In: 1st International Conference on Computational Models of Argument (COMMA) (2006)
3. Bondarenko, A., Dung, P., Kowalski, R., Toni, F.: An abstract, argumentation-theoretic framework for default reasoning. Artificial Intelligence 93(1-2), 63–101 (1997)
4. Brewka, G.: Well-founded semantics for extended logic programs with dynamic preferences. Journal of Artificial Intelligence Research 4, 19 (1996)
5. Caminada, M., Amgoud, L.: An axiomatic account of formal argumentation. In: Proc. AAAI (2005)
6. Dimopoulos, Y., Nebel, B., Toni, F.: On the computational complexity of assumption-based argumentation for default reasoning. Artificial Intelligence 141, 57–78 (2002)
7. Dung, P.: The acceptability of arguments and its fundamental role in non-monotonic reasoning and logic programming and n-person game. Artificial Intelligence 77, 321–357 (1995)
8. Dung, P., Kowalski, R., Toni, F.: Dialectic proof procedures for assumption-based, admissible argumentation. Artificial Intelligence 170, 114–159 (2006)
9. Dung, P., Mancarella, P., Toni, F.: A dialectic procedure for sceptical, assumption-based argumentation. In: 1st International Conference on Computational Models of Argument (COMMA) (2006)

10. Dung, P., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artificial Intelligence - Special Issue on Argumentation in Artificial Intelligence 171(10-15), 642–674 (2007)
11. Foster, I., Jennings, N., Kesselman, C.: Brain meets brawn: why grid and agents need each other. In: Proc. AAMAS (2004)
12. Gaertner, D., Toni, F.: A credulous and sceptical argumentation system. In: Proc. of ArgNMR (2007)
13. Kakas, A.C., Moraitis, P.: Argumentation based decision making for autonomous agents. In: Proc. AAMAS, pp. 883–890 (2003)
14. Kowalski, R.A., Toni, F.: Abstract argumentation. Journal of Artificial Intelligence and Law, Special Issue on Logical Models of Argumentation 4(3-4), 275–296 (1996)
15. Stathis, K., Lekeas, G.K., Kloukinas, C.: Competence checking for the global e-service society using games. In: Engineering Societies in the Agents World (ESAW 2006). Springer, Heidelberg (2007)
16. Morge, M.: A dialectics multiagent system in which argumentative agents play and arbitrate to reach an agreement. In: Proc.1st Workshop on Argumentation in Artificial Intelligence and Law (2005)
17. Nute, D.: Defeasible reasoning. In: Fetzer, J.H. (ed.) Aspects of Artificial Intelligence, pp. 251–288. Kluwer Academic Publishers, Dordrecht (1987)
18. Prakken, H., Sartor, G.: Argument-based extended logic programming with defeasible priorities. Journal of Applied Non-Classical Logics 7(1), 25–75 (1997)
19. Stournaras, T., Dimitrelos, D., Tabasco, A., Barba, J., Pedrazzani, D., Yage, M., An, T., Dung, P., Hung, N., Khoi, V.D., Thang, P.M.: e-business application scenarios. In: Stournaras, T. (ed.) ARGUGRID deliverable D.1.2 (2007)
20. Toni, F.: Assumption-Based Argumentation for Closed and Consistent Defeasible Reasoning. In: Satoh, K., Inokuchi, A., Nagao, K., Kawamura, T. (eds.) JSAI 2007. LNCS (LNAI), vol. 4914, pp. 390–402. Springer, Heidelberg (2008)

A Complete Quantified Epistemic Logic for Reasoning about Message Passing Systems

Francesco Belardinelli[1] and Alessio Lomuscio[2]

[1] Scuola Normale Superiore, Pisa
belardinelli@sns.it
[2] Department of Computing
Imperial College London, UK
A.Lomuscio@imperial.ac.uk

Abstract. We investigate quantified interpreted systems, a semantics to model multi-agent systems in which the agents can reason about individuals, their properties, and relationships among them. The semantics naturally extends interpreted systems to first-order by introducing a domain of individuals. We present a first-order epistemic language interpreted on this semantics and prove soundness and completeness of the quantified modal system $QS5_n^D$, an axiomatisation for these structures. Finally, we exemplify the use of the logic by modeling message passing systems, a relevant class of interpreted systems analysed in epistemic logic.

1 Introduction

Modal epistemic logic has been widely studied in multi-agent systems (MAS) both on its own and in combination with other modalities, very often temporal ones. The typical language extends propositional logic by adding n modalities K_i representing the knowledge of agent i, as well as other modalities representing different mental states for the agents (distributed and common knowledge, beliefs, etc) and/or the temporal flow of time [10,25].

The use of modal propositional logic as a specification language requires little justification: it is a rather expressive language, well-understood from a theoretical point of view. Still, it is hard to counterargue the remark, often raised by practitioners in Software Engineering, that quantification in specifications is so natural and convenient that it really should be brought explicitly into the language. Even when working with finite domains of individuals, without quantification one is forced to introduce ad-hoc propositions to emulate basic relations among individuals. Not always quantification is simply syntactic sugar: certain expressivity needs do require infinite domains (e.g., see section 4 below).

In multi-agent systems the power of first-order logic is required every time agents reason about:

- relational statement, as in *agent i knows that message μ has been sent by a to b*, or formally
$$K_i Sent(a, b, \mu)$$

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 248–267, 2008.

- functional dependency or identity: *agent i knows that message μ is the encryption of message μ' with key k*, formally

$$K_i(\mu = enc(k, \mu'))$$

- an infinite domain of individuals, or a finite domain whose cardinality cannot be bounded in advance: *agent i has to read all e-mails before deleting them*, or formally

$$\forall \mu (Delete(i, \mu) \rightarrow Read(i, \mu))$$

- quantification on agents' indexes [21]: *all agents knows..., at least one agent knows...*

$$\forall i K_i \ldots, \exists i K_i \ldots$$

Further, epistemic modalities can be combined with quantifiers to express concepts such as knowledge *de re/de dicto* [12].

Irrespective of the above, the use of first-order modal logic in MAS specifications is normally frowned upon by theoreticians. Why should we use an undecidable language when a decidable one does the job reasonably well already? Is the price that quantification brings in justified? While these objections are certainly sensible, we believe that their strength has been increasingly weakened by recent progress in the verification of MAS by model checking [13,28,26,23]. In the model checking approach [8] the decision problem is tackled not by checking validity but simply model satisfaction. In other words, we do not check whether a formula representing a specification is satisfiable, but simply whether it is true on the model representing all possible evolutions of the system. While the former problem is undecidable for first-order modal logic, the latter is decidable at least in some suitable fragments, such as the *monodic* fragments studied in [16,32,33,35]. Moreover, we have specification languages supporting first-order interval temporal logic [29,30]. Recently, first-order modal logic has been applied to the analysis of security protocols [1,5,9]. Finally, we have some preliminary works on first-order model checking [31,34].

This paper takes inspiration from the considerations above and aims at making progress on the subject of first-order epistemic logic. The main contribution of the paper is the axiomatisation in section 5, where a sound and complete system for quantified interpreted systems (QIS) is presented. We argue that QIS are the natural extension to first-order of Interpreted Systems semantics, the usual formalism for epistemic logic in MAS [10,25].

While completeness results for quantified modal logic are customarily proved with respect to Kripke semantics [12,18], we should state clearly that QML has been discussed in MAS settings before. In [10] quantified epistemic logic is briefly discussed, along with its Kripke semantics and some significant validities; in [21] the authors introduce a quantified logic of belief, in which the doxastic modalities are indexed to terms of a first-order language; in [2] a limited form of quantification is added to Coalition Logic. However, in most of the works above completeness is not tackled. This may be due to the technical difficulties associated with QML and the relatively poor status of the metatheoretical investigation in comparison with the propositional case. We hope this contribution will be the first in a line of work in which a systematic analysis of these logics is provided.

Scheme of the paper. In section 2 we present two classes of first-order structures: systems of global states and Kripke frames. In section 3 we introduce the first-order modal language $\mathcal{L}_n^D$ which is interpreted on quantified interpreted systems, a valued version of the systems of global states. In section 4 we exemplify syntax and semantics by describing three formal models for multi-agent systems and discuss some specification patterns. In section 5 we introduce the first-order modal system $Q.S5_n^D$, and prove the main result of this paper: $Q.S5_n^D$ is a sound and complete axiomatisation of the validities in the structures of global states. Finally, section 6 outlines some extensions of the present formalism.

2 Systems of Global States and Kripke Frames

In this section we introduce the systems of global states and Kripke frames in a first-order setting. While the first ones are used in computer science to model the behaviour of MAS [10,14,25], Kripke frames are best employed to get a deeper understanding of the formal properties of these systems [6,7]. Technically, we extend the corresponding propositional structures to first-order. This extension is not trivial, as there are many ways of performing it: for instance, we can choose a single domain of quantification or several domains for each agent and/or for each computational state, not to mention domains of intensional objects [4]. In this paper we consider the simplest construction, where we have just a single quantification domain D common to all the agents and states, which contains all possible objects. We leave other options for further work. In what follows we assume a set of agents $A = \{1, \ldots, n\}$.

2.1 Systems of Global States

This paper is primarily concerned with the representation of knowledge in MAS, not their temporal evolution. Given this, we adopt the "static" perspective on the systems of global states [22], rather than the "dynamic" version [10]. So, while we assume that the states of the system result from the evolution given by protocols and transitions, for the time being we do not consider them explicitly. More formally, consider a set L_i of local states $l_i, l'_i, \ldots$, for each agent $i \in A$, and a set L_e containing the local states of the environment $l_e, l'_e, \ldots$. We define a system of global states as follows:

Definition 1 (SGS). *A system of global states $\mathcal{S}$ is a couple $\langle S, D \rangle$ such that $S \subseteq L_e \times L_1 \times \ldots \times L_n$ is a non-empty set of global states, and D is a non-empty domain of individuals. $\mathcal{SGS}$ is the class of the systems of global states.*

This definition of SGS is based on two assumptions. First, the domain D of individuals is the same for every agent i, so that all agents effectively reason about the same objects. This choice is justified by the *external account of knowledge* usually adopted in the framework of interpreted systems. If knowledge is ascribed to agents by an external observer, it seems natural to focus on a unique set of individuals: the ones assumed to exist by the external observer. Second, the domain D is assumed to be the same for every global state, i.e., no individual appears nor disappears in moving from one state to another. This also is consistent with the external account of knowledge: all individuals are supposed to be existing from the observer's viewpoint. We discuss further options in section 6. Finally, it can be the case that $A \subseteq D$. This means that the agents can reason about themselves, their properties, and relationships.

2.2 Kripke Frames

While Kripke frames are less intuitive than interpreted systems to model MAS, they are more convenient for the purpose of formal analysis, namely completeness investigations. We work with frames with equivalence relations, so we take the following definition:

Definition 2. *A Kripke frame $\mathcal{F}$ is a $n+2$-tuple $\langle W, \sim_1, \dots, \sim_n, D\rangle$ such that W is a non-empty set; for $i \in A$, $\sim_i$ is an equivalence relation on W; D is a non-empty set of individuals. $\mathcal{K}$ is the class of all Kripke frames.*

Now we have systems of global states modelling MAS and Kripke frames. In order to axiomatise $\mathcal{SGS}$, it is useful to map SGS into Kripke frames.

2.3 Maps between $\mathcal{SGS}$ and $\mathcal{K}$

We explore the relationship between these structures by means of two maps f and g from $\mathcal{SGS}$ to $\mathcal{K}$ and viceversa. We show that every SGS $\mathcal{S}$ is isomorphic to $g(f(\mathcal{S}))$, that is, there is a one-to-one correspondence onto the sets of global states and the domains of individuals. Further, we prove that every Kripke frame $\mathcal{F} = \langle W, \sim_1, \dots, \sim_n, D\rangle$ is isomorphic to $f(g(\mathcal{F})) = \langle W', \sim'_1, \dots, \sim'_n, D'\rangle$, that is, there are bijections between W and W' and between D and D'; in addition $w \sim_i w'$ iff $(f \circ g)(w) \sim'_i (f \circ g)(w')$. As a result, every sound and complete axiomatisation of Kripke frames is also an axiomatisation of SGS.

We start with the map $f : \mathcal{SGS} \rightarrow \mathcal{K}$. Let $\mathcal{S} = \langle S, D\rangle$ be an SGS, define $f(\mathcal{S})$ as the $n+2$-tuple $\langle S, \sim_1, \dots, \sim_n, D\rangle$, where S is the set of possible states and D is the domain of individuals. Moreover, for each $i \in A$, the relation $\sim_i$ on S such that $\langle l_e, l_1, \dots, l_n\rangle \sim_i \langle l'_e, l'_1, \dots, l'_n\rangle$ iff $l_i = l'_i$ is an equivalence relation. So $f(\mathcal{S})$ is a Kripke frame.

For the converse map $g : \mathcal{K} \rightarrow \mathcal{SGS}$, let $\mathcal{F} = \langle W, \sim_1, \dots, \sim_n, D\rangle$ be a Kripke frame. For every epistemic state $w \in W$, let the equivalence class $[w]_{\sim_i} = \{w' | w \sim_i w'\}$ be a local state for agent i, and W is the set of local states for the environment. Define $g(\mathcal{F}) = \langle S, D\rangle$, where S contains all the $n+1$-tuples $\langle w, [w]_{\sim_1}, \dots, [w]_{\sim_n}\rangle$, for $w \in W$, while D is defined as above. The structure $g(\mathcal{F})$ is trivially an SGS.

We prove that the composition of the two maps gives isomorphic structures.

Lemma 1. *Every Kripke frame $\mathcal{F}$ is isomorphic to $f(g(\mathcal{F}))$.*

Proof. If $\mathcal{F} = \langle W, \sim_1, \ldots, \sim_n, D\rangle$ is a Kripke frame, then $f(g(\mathcal{F})) = \langle W', \sim'_1, \ldots, \sim'_n, D\rangle$ is such that W' is the set of n+1-tuples $\langle w, [w]_{\sim_1}, \ldots, [w]_{\sim_n}\rangle$, for $w \in W$. The composition $f \circ g$ is a bijection between W and W': it is one-to-one as if $w, w' \in W'$ and $w = w'$, then in particular the first components of w and w' are equal. It is onto as the first component w_1 of $w \in W'$ is such that $w_1 \in W$ and $f(g(w_1)) = w$. Also, the identity on D is a bijection. Finally, $w \sim_i w'$ iff $[w]_{\sim_i} = [w']_{\sim_i}$ iff $\langle w, [w]_{\sim_1}, \ldots, [w]_{\sim_n}\rangle \sim'_i \langle w', [w']_{\sim_1}, \ldots, [w']_{\sim_n}\rangle$. Thus, the two structures are isomorphic. □

By lemma 1 we will show in section 5 that a sound and complete axiomatisation of Kripke frames is adequate also with respect to SGS.

3 Syntax and Semantics

In this section we introduce the first-order multi-modal language $\mathcal{L}_n^D$ containing individual variables and constants, as well as quantifiers, n epistemic operators, the distributed knowledge operator, and identity. The language $\mathcal{L}_n^D$ is interpreted on models based on Kripke frames. Finally, we present the quantified interpreted systems, a valued version of the systems of global states.

3.1 Syntax

The first-order multi-modal language $\mathcal{L}_n^D$ contains individual variables $x_1, x_2, \ldots$, n-ary functors $f_1^n, f_2^n, \ldots$ and n-ary predicative letters $P_1^n, P_2^n, \ldots$, for $n \in \mathbb{N}$, the identity predicate $=$, the propositional connectives $\neg$ and $\rightarrow$, the universal quantifier $\forall$, the epistemic operators K_i, for $i \in A$, and the distributed knowledge operator D_G, for $G \subseteq A$.

Definition 3. *Terms and formulas in the language $\mathcal{L}_n^D$ are defined in the Backus-Naur form as follows:*

$$t ::= x \mid f^k(t_1, \ldots, t_k)$$
$$\phi ::= P^k(t_1, \ldots, t_k) \mid t = t' \mid \neg\phi \mid \phi \rightarrow \psi \mid K_i\phi \mid D_G\phi \mid \forall x\phi$$

Intuitively, the formula $K_i\phi$ means that *agent i knows ϕ*, while $D_G\phi$ is read as *ϕ is distributed knowledge among the agents in G*. The symbols $\bot$, $\wedge$, $\vee$, $\leftrightarrow$ and $\exists$ are defined by means of the other logical constants; we refer to the 0-ary functors as individual constants $c_1, c_2, \ldots$ A closed term v is a term where no variable appears, the closed terms are only constants and terms obtained by applying functors to closed terms. Finally, by $t[\vec{y}]$ (resp. $\phi[\vec{y}]$) we mean that $\vec{y} = y_1, \ldots, y_n$ are all the free variables in t (resp. ϕ); while $t[\vec{y}/\vec{t}]$ (resp. $\phi[\vec{y}/\vec{t}]$) denotes the term (resp. formula) obtained by simultaneously substituting some, possibly all, free occurrences of $\vec{y}$ in t (resp. ϕ) with $\vec{t} = t_1, \ldots, t_n$, renaming bounded variables if necessary.

3.2 Semantics

In order to assign a meaning to the formulas in $\mathcal{L}_n^D$ we make use of Kripke models. We then define validity on quantified interpreted systems in terms of validity on Kripke models.

Definition 4 (model). *A Kripke model $\mathcal{M}$ - or simply K-model - based on a Kripke frame $\mathcal{F}$, is a couple $\langle \mathcal{F}, I \rangle$ where I is an interpretation such that:*

- *if f^k is a k-ary functor, then $I(f^k)$ is a function from D^k to D;*
- *if P^k is a k-ary predicative letter and $w \in W$, then $I(P^k, w)$ is a k-ary relation on D, i.e. $I(P^k, w) \subseteq D^k$;*
- *the interpretation $I(=, w)$ of the identity $=$ in w is the equality on D.*

Note that function symbols are interpreted rigidly, that is, for every $w, w' \in W$ the interpretation of a functor f^k in w is the same as the interpretation of f^k in w'. Given that our approach is the one of the external observer, rigid designators seem appropriate.

Now let σ be an assignment, i.e., any function from the set of variables in $\mathcal{L}_n^D$ to the domain D, the valuation $I^\sigma(t)$ of a term t is defined as $\sigma(y)$ for $t = y$, and $I^\sigma(t) = I(f^k)(I^\sigma(t_1), \ldots, I^\sigma(t_k))$, for $t = f^k(t_1, \ldots, t_k)$. In particular, the valuation $I^\sigma(d)$ of a constant d is an individual $I(d)$ in D. The variant $\sigma\binom{x}{a}$ of the assignment σ differs from σ at most on x and assigns element $a \in D$ to x. Now we are able to define the truth conditions for the formulas in $\mathcal{L}_n^D$.

Definition 5 (Satisfaction). *The satisfaction relation $\models$ for a formula $\phi \in \mathcal{L}_n^D$, a world $w \in \mathcal{M}$ and an assignment σ is inductively defined as follows:*

$$\begin{array}{lll}
(\mathcal{M}^\sigma, w) \models P^k(\vec{t}) & \text{iff} & \langle I^\sigma(t_1), \ldots, I^\sigma(t_k) \rangle \in I(P^k, w) \\
(\mathcal{M}^\sigma, w) \models t = t' & \text{iff} & I^\sigma(t) = I^\sigma(t') \\
(\mathcal{M}^\sigma, w) \models \neg\psi & \text{iff} & (\mathcal{M}^\sigma, w) \not\models \psi \\
(\mathcal{M}^\sigma, w) \models \psi \to \psi' & \text{iff} & (\mathcal{M}^\sigma, w) \not\models \psi \text{ or } (\mathcal{M}^\sigma, w) \models \psi' \\
(\mathcal{M}^\sigma, w) \models K_i\psi & \text{iff} & \text{for } w' \in W,\ w \sim_i w' \text{ implies } (\mathcal{M}^\sigma, w') \models \psi \\
(\mathcal{M}^\sigma, w) \models D_G\psi & \text{iff} & \text{for } w' \in W,\ (w, w') \in \bigcap_{i \in G} \sim_i \text{ implies } (\mathcal{M}^\sigma, w') \models \psi \\
(\mathcal{M}^\sigma, w) \models \forall x\psi & \text{iff} & \text{for all } a \in D, (\mathcal{M}^{\sigma\binom{x}{a}}, w) \models \psi
\end{array}$$

The truth conditions for the formulas containing the symbols $\perp$ $\wedge$, $\vee$, $\leftrightarrow$ and $\exists$ are standardly defined from those above. Further, a formula ϕ in $\mathcal{L}_n^D$ is said to be *true at a world* w iff it is satisfied at w by every assignment σ; ϕ is *valid on a model* $\mathcal{M}$ iff it is true at every world in $\mathcal{M}$; ϕ is *valid on a frame* $\mathcal{F}$ iff it is valid on every model on $\mathcal{F}$; ϕ is *valid on a class* $\mathcal{C}$ *of frames* iff it is valid on every frame in $\mathcal{C}$.

Now we have all preliminary definitions to introduce quantified interpreted systems.

Definition 6 (QIS). *A quantified interpreted systems $\mathcal{P}$ based on an SGS $\mathcal{S}$, is a couple $\langle \mathcal{S}, I \rangle$ such that I is an interpretation of $\mathcal{L}_n^D$ in the Kripke frame $f(\mathcal{S})$.*

The notions of satisfaction, truth and validity are defined as above, i.e., let $\mathcal{P}_f = \langle f(\mathcal{S}), I \rangle$ be the Kripke model for the quantified interpreted system $\mathcal{P} = \langle \mathcal{S}, I \rangle$, then $(\mathcal{P}^\sigma, s) \models \phi$ iff $(\mathcal{P}_f^\sigma, s) \models \phi$. In particular, the present definition of satisfaction agrees with the usual definition for interpreted systems:

$$\begin{array}{lll} (\mathcal{P}^\sigma, s) \models P^k(t_1, \ldots, t_k) & \text{iff} & \langle I^\sigma(t_1), \ldots, I^\sigma(t_k) \rangle \in I(P^k, s) \\ (\mathcal{P}^\sigma, s) \models K_i\psi & \text{iff} & l_i(s) = l_i(s') \text{ implies } (\mathcal{P}^\sigma, s') \models \psi \\ (\mathcal{P}^\sigma, s) \models D_G\psi & \text{iff} & l_i(s) = l_i(s') \text{ for all } i \in G \text{ implies } (\mathcal{P}^\sigma, s') \models \psi \\ (\mathcal{P}^\sigma, s) \models \forall x\psi & \text{iff} & \text{for all } a \in D,\ (\mathcal{P}^{\sigma\binom{a}{x}}, s') \models \psi \end{array}$$

Moreover, a formula $\phi \in \mathcal{L}_n^D$ is valid on a quantified interpreted systems $\mathcal{P}$ iff ϕ is valid on $\mathcal{P}_f$, or more formally:

Definition 7 (Validity on QIS). *If ϕ is a formula in $\mathcal{L}_n^D$ and $\mathcal{P}$ is a quantified interpreted systems, then $\mathcal{P} \models \phi$ iff $\mathcal{P}_f \models \phi$.*

Thus, we can reason about a multi-agent system by using the expressiveness of QIS, but rely on Kripke models to prove formal properties of the system.

3.3 Some Validities

Clearly, the language $\mathcal{L}_n^D$ is very expressive. We can specify the knowledge agents have of facts about individuals, as in the following specification: *agent i knows that someone sent him a message when he receives it*,

$$\forall j, \mu(Recd(i, j, \mu) \rightarrow K_i Sent(j, i, \mu)) \tag{1}$$

This specification can be expressed also in some propositional modal languages. However, in $\mathcal{L}_n^D$ we can make more subtle distinctions as in *if agent i receives a message, then he knows that someone sent it to him*:

$$\forall \mu(\exists j Recd(i, j, \mu) \rightarrow K_i\ \exists j' Sent(j', i, \mu)) \tag{2}$$

The latter specification is weaker than the former, as (2) says nothing about the identity of the sender, while (1) requires that the receiver knows the identity of the sender.

We briefly explore the semantics of QIS by considering the traditional Barcan formulas [12]. There has been much discussion on these principles and their soundness in epistemic contexts. Given that the domain of quantification is the same for every global state, both the Barcan formula and its converse are valid on the class $\mathcal{QIS}$ of all QIS, i.e., they hold in every quantified interpreted system:

$$\begin{array}{lll} \mathcal{QIS} \models \forall x K_i\phi \rightarrow K_i\forall x\phi & \mathcal{QIS} \models \forall x D_G\phi \rightarrow D_G\forall x\phi & BF \\ \mathcal{QIS} \models K_i\forall x\phi \rightarrow \forall x K_i\phi & \mathcal{QIS} \models D_G\forall x\phi \rightarrow \forall x D_G\phi & CBF \end{array}$$

We remarked that these formulas are direct consequences of having fixed domains, which were justified by the external account of knowledge usually adopted in epistemic logic: the domain of quantification consists of the individuals considered by the designer. Also, we deem these validities in line with the bird's eye approach of epistemic logic. By BF if agent i knows that a is ϕ for each individual $a \in D$, then she knows that all the individuals are ϕ. In fact, in any epistemic alternative considered by agent i *at most* the individuals she currently considers (i.e., those in D) are present. In other words, agents are assumed to be able to generalise their knowledge, at least when this is considered from an external point of view. By CBF if agent i knows that all the individuals are ϕ, then she knows that a is ϕ, for each individual $a \in D$. This happens because in any epistemic alternative considered by agent i *at least* the individuals she currently considers (again those in D) are present. In other words, agents are assumed to be able to particularise their knowledge.

We stress the fact that in this interpretation the formula $K_i \forall x \phi$ does not mean that *agent i knows that all the individuals she considers are ϕ*, but rather *agent i knows that all the individuals (considered by the external designer) are ϕ*. Further, following the external account of knowledge typical in epistemic logic, the truth of $K_i \forall x \phi$ does not imply that agent i has to be aware of all the individuals considered by the designer. As it is the case in propositional epistemic logic, the formula $\forall x \phi$ expresses the knowledge attributed by the external observer to agent i, rather than the explicit knowledge possessed by i.

We have also generalised versions of the Barcan formula and its converse, for arbitrary strings of epistemic operators:

$$\mathcal{QIS} \models \forall x E_{j_1} \dots E_{j_m} \phi \to E_{j_1} \dots E_{j_m} \forall x \phi \qquad BF_{j_1,\dots,j_m}$$
$$\mathcal{QIS} \models E_{j_1} \dots E_{j_m} \forall x \phi \to \forall x E_{j_1} \dots E_{j_m} \phi \qquad CBF_{j_1,\dots,j_m}$$

where each E_{j_k} is either K_i or D_G. Even if these principles seem quite strong, by considering an external notion of knowledge they do not appear problematic.

For what concerns identity, the following principles hold:

$$\mathcal{QIS} \models (t = t') \to K_i(t = t') \qquad \mathcal{QIS} \models (t \neq t') \to K_i(t \neq t')$$
$$\mathcal{QIS} \models (t = t') \to D_G(t = t') \qquad \mathcal{QIS} \models (t \neq t') \to D_G(t \neq t')$$

These validities result from rigid designation and require further explanation. Suppose message μ' is the encryption of message μ with key k, i.e. $\mu' = enc(k, \mu)$, then by the principles above any agent i should know this identity, that is, for each i, we have $K_i(\mu' = enc(k, \mu))$. But this seems to imply that we cannot represent encryption in this formalism. However, if we assume the *de re* interpretation of modality, we can reconcile encryption and the validities above. In fact, if μ' and $enc(k, \mu)$ are one and the same message, then any agent i knows that this message is identical to itself, which is the *de re* interpretation of $K_i(\mu' = enc(k, \mu))$. Still, agent i may not have explicit, *de dicto* knowledge of the fact that message μ' is obtained by encrypting μ with key k.

4 Message-Passing QIS

In this section we show how to model message-passing systems [10] in the framework of QIS. A m.p. system is a multi-agent system where agents communicate by exchanging messages, so the most relevant events are sending and receiving messages. The formalism of message passing systems is useful to model a wide range of MAS. For instance, a network of computers, such as the Internet, can be seen as a m.p. system. In general, any multi-agent system is a m.p. system if the message transmission delay is not negligible. In a m.p. system the local state of each agent contains information about its initial state, the messages it has sent or received, as well as the internal actions it has taken.

In what follows we show that m.p. systems can be defined as a particular class of SGS satisfying a finite number of specifications in the first-order modal language $\mathcal{L}_n^D$. Our main result consists in showing that Proposition 4.4.3 in [10], concerning the knowledge of the ordering of events in m.p. systems, can be restated as a validity on the class of QIS modeling m.p. systems. Thus, the formalism of QIS is powerful enough to deal with the theory of m.p. systems. Throughout the section we refer to [10], par. 4.4.5-6, for the details of m.p. systems.

More formally, we introduce a set *Act* of actions $\alpha_1, \alpha_2, \ldots$, and a set MSG of messages $\mu_1, \mu_2, \ldots$ For each agent $i \in A$, we consider a set Σ_i of initial events $init(i, \alpha)$, and a set INT_i of internal events $int(i, \alpha)$. We define the local state l_i for agent i as a *history* over Σ_i, INT_i and MSG, that is, a sequence of events whose first element is in Σ_i, and whose following elements either belong to INT_i or are events of the form $send(i, j, \mu)$, $rec(i, j, \mu)$ for $j \in A$, $\mu \in MSG$ and $\alpha \in Act$. Intuitively, $init(i, \alpha)$ represents the event where *agent i performs the initial action α*, $send(i, j, \mu)$ represents the event where *agent i sends message μ to j*, while the intended meaning of $rec(i, j, \mu)$ is that *agent i receives message μ from j*. Finally, $int(i, \alpha)$ means that *agent i performs the internal action α*.

A global state s is a tuple $\langle l_e, l_1, \ldots, l_n \rangle$, where $l_1, \ldots, l_n$ are local states as above and l_e contains all the events in $l_1, \ldots, l_n$. We define a reflexive, transitive and anti-symmetric relation $\leq$ on the local states of agent i such that $l_i \leq l'_i$ iff l_i is a prefix of l'_i. This order extends to global states, so that $s \leq s'$ iff $l_i \leq l'_i$ for every $i \in A$. We can define message passing systems as a special class of quantified interpreted systems by considering the class of QIS $\mathcal{P} = \langle S, D, I \rangle$ where S is a non-empty set of global states; the domain D of individuals includes all agents in A, the messages in MSG, the actions in *Act*, and the events $e_1, e_2, \ldots$; I is the interpretation for $\mathcal{L}_n^D$. Intuitively, each m.p. QIS models the evolution of a m.p. system: starting from an initial state, the m.p. QIS contains the states reachable during the execution of the m.p. system. The temporal evolution of a m.p. QIS can be represented as a sequence $s_0, s_1, \ldots$ of global states such that $s_0 = \langle init(e, \alpha_e), init(1, \alpha_1), \ldots, init(n, \alpha_n) \rangle$, and for every $n \in \mathbb{N}$, either s_{n+1} is identical to s_n or there is an i such that $l_i(s_n) \leq l_i(s_{n+1})$ but $l_i(s_n) \neq l_i(s_{n+1})$. Note that a single m.p. QIS can contain several temporal evolutions of the same m.p. system.

We assume that our language has terms and predicative letters for representing the objects in the domain D and the relations among them. In particular, $e_1, e_2, \ldots$ are metaterms ranging over events: we write $\forall e \phi[e]$ as a shorthand for

$$\forall i, j, \mu, \alpha \; \phi[send(i, j, \mu)] \wedge \phi[rec(i, j, \mu)] \wedge \phi[init(i, \alpha)] \wedge \phi[int(i, \alpha)]$$

In fact, any event is either a send or a receive event, or an initial action or an internal action. We use the same notation for the objects in the model and the syntactic elements, as the ones mirror the others; the distinction will be made clear by the context. We immediately give some examples of the expressiveness of our formalism. In $\mathcal{L}_n^D$ we can define events by formulas which are provably valid in every m.p. QIS (the existence of a unique individual $\exists!$ can be defined by means of $=$):

$$\forall e \exists! i, j, \mu, \alpha \; (i \neq j) \wedge \; (e = send(i,j,\mu) \vee e = rec(i,j,\mu) \vee e = init(i,\alpha) \vee e = int(i,\alpha))$$

$$\begin{aligned}\forall i, j, \mu, \alpha \exists! e_1, e_2, e_3, e_4 (send(i,j,\mu) = e_1 \wedge rec(i,j,\mu) = e_2 \wedge init(i,\alpha) = e_3 \wedge int(i,\alpha) = e_4 \wedge \\ \wedge e_1 \neq e_2 \wedge e_1 \neq e_3 \wedge e_1 \neq e_4 \wedge e_2 \neq e_3 \wedge e_2 \neq e_4 \wedge e_3 \neq e_4).\end{aligned}$$

The first specification expresses the fact that every event is either a *send* or a *receive* event, where the sender is different from the receiver, or an initial action, or an internal action. The second specification says that every *send* or *receive* event, initial action, and internal action are distinct events. Thus, we cannot have $send(i,j,\mu) = e = rec(i',j',\mu')$. It is easy to check that our definition of m.p. QIS validates these specifications.

In [10], p. 132 the authors list three constraints on m.p. systems, the third one involves the notion of *run* on an SGS. Nonetheless, we can restate the first two without introducing runs:

MP1 a local state l_i for agent i is a *history* over Σ_i, INT_i and MSG;
MP2 for every event $rec(i,j,\mu)$ in $l_i(s)$ there exists an event $send(j,i,\mu)$ in $l_j(s)$.

Further, the following simplifying assumption is considered.

* all events in a given agent's local state are distinct, an agent can never perform the same action twice.

Note that this does not restrict our analysis as identical actions can be timestamped. We show how to formalise these specifications in the language $\mathcal{L}_n^D$ of m.p. QIS. First, we introduce a predicate H for *happened* such that $(\mathcal{P}^\sigma, s) \models H(e,i)$ iff e is an event in $l_i(s)$. The formula $H(e)$ is a shorthand for $\exists i H(e,i)$. By the definition of m.p. system, we can show that $(\mathcal{P}^\sigma, s) \models H(e)$ iff e is an event in s. Further, we define an order $Prec$ on events as follows:

$$\begin{aligned}(\mathcal{P}^\sigma, s) \models Prec(e,e',i) \text{ iff } & (\mathcal{P}^\sigma, s) \models H(e,i) \wedge H(e',i) \text{ and} \\ & \text{for all } s' \leq s, \; (\mathcal{P}^\sigma, s') \models H(e',i) \rightarrow H(e,i).\end{aligned}$$

The definition of $Prec(e,e')$ is similar, with $H(e)$ instead of $H(e,i)$. We can force the events in a global state s to be partially ordered by specifying that $Prec(e,e')$ is a reflexive and transitive relation on the set of past events:

$$H(e) \rightarrow Prec(e,e) \tag{3}$$

$$Prec(e,e') \wedge Prec(e',e'') \rightarrow Prec(e,e'') \tag{4}$$

As an example, we show that (4) holds. Suppose that $(\mathcal{P}^\sigma, s) \models Prec(e,e') \wedge Prec(e',e'')$. This means that $(\mathcal{P}^\sigma, s) \models \exists i H(e,i) \wedge \exists j H(e'',j)$. Moreover, we have that for all $s' \leq s$, $(\mathcal{P}^\sigma, s') \models \exists i H(e', i) \to \exists i H(e, i)$ and $(\mathcal{P}^\sigma, s') \models \exists j H(e'', j) \to \exists j H(e', j)$. By renaming bounded variables and the transitivity of implication, we obtain that for all $s' \leq s$, $(\mathcal{P}^\sigma, s') \models \exists i H(e'', i) \to \exists i H(e, i)$. As a result, $(\mathcal{P}^\sigma, s) \models Prec(e, e'')$. Further, $Prec(e, e', i)$ can be defined as a linear order on the events in l_i, i.e., it is also anti-symmetric and total:

$$Prec(e, e', i) \land Prec(e', e, i) \to (e = e') \tag{5}$$

$$H(e, i) \land H(e', i) \to Prec(e, e', i) \lor Prec(e', e, i) \tag{6}$$

We define $Linear(Prec(e, e', i))$ as the conjunction of (3)–(6) above, expressing the fact that the relation $Prec(e, e', i)$ is linear. Also, we define the first event as the minimal one with respect to $Prec(e, e', i)$, that is,

$$Fst(e, i) ::= H(e, i) \land \forall e'(e' \neq e \to (H(e', i) \to \neg Prec(e', e, i)))$$

Finally, the formulas $Sent(i, j, \mu)$, $Recd(i, j, \mu)$, $Init(i, \alpha)$, and $Int(i, \alpha)$ are shorthands for $H(send(i, j, \mu))$, $H(rec(i, j, \mu))$, $H(init(i, \alpha))$, and $H(int(i, \alpha))$ respectively. Now we can formalise the specifications MP1-2 and * as follows:

MP1* $Linear(Prec(e, e', i)) \land$
$\land \exists! e(Fst(e, i) \land \exists \alpha(e = init(i, \alpha))) \land \forall e(H(e, i) \land \neg Fst(e, i) \to$
$\to \exists j, \alpha, \mu(e = int(i, \alpha) \lor e = send(i, j, \mu) \lor e = rec(i, j, \mu)))$
MP2' $\forall i, j, \mu(Recd(i, j, \mu) \to Sent(j, i, \mu))$

MP1* forces the local state of any agent i to satisfy MP1 and *; while by MP2' specification MP2 is satisfied. MP1*-2 are the basic specifications for m.p. QIS, but we can consider further constraints on message passing system. A property often required in the framework of m.p. systems is *channel reliability*. Modified from [10], a m.p. system is reliable if every sent message is eventually received, or more formally:

MP4 if $send(i, j, \mu)$ is in $l_i(s)$, then there exists a global state s' such that $rec(j, i, \mu)$ is in $l_j(s')$.

In $\mathcal{L}_n^D$ we can formalise this specification as follows:

MP4' $\forall j, \mu(Sent(i, j, \mu) \to \neg K_i \neg Recd(j, i, \mu))$

In fact, if $send(i, j, \mu)$ is in $l_i(s)$, by MP4' $(\mathcal{P}^\sigma, s) \models \neg K_i \neg Recd(j, i, \mu)$, this means that there exists a global state s' such that $(\mathcal{P}^\sigma, s') \models Recd(j, i, \mu)$, that is, $rec(j, i, \mu) \in l_j(s')$. Thus, MP4 holds. Note that MP4' is stronger than MP4 as the former requires that the local states of agent i in s and s' are identical. This can be considered a limit of our epistemic language, due to the lack of temporal operators. Further, a relevant property of m.p. systems concerns *authentication*: if agent i receives a message μ from agent j, then i knows that μ was actually sent by j. This specification can be expressed as

$$\forall j, \mu(Recd(i, j, \mu) \to K_i Sent(j, i, \mu))$$

Finally, we may require that agents have *perfect recall*, that is, they know everything that has happened to them:

$$\forall e(H(e,i) \rightarrow K_i H(e,i))$$

It is easy to show that by the way they are defined, m.p. QIS satisfy authentication and perfect recall but not channel reliability. We remark that all the specifications introduced are defined by means of only the predicative constants $H(e,i)$ and $Prec(e,e',i)$.

We now prove the main result of this section: Proposition 4.4.3 in [10] can be restated as a validity on the class of m.p. QIS satisfying MP1, MP2, and *. We do not give the full statement, but we note that this metatheoretical result can be restated as a formula in the first-order modal language $\mathcal{L}_n^D$. First, we introduce a relation $\mapsto_G$ of *potential causality* between events, as discussed in [20]. This relation is intended to capture the intuition that event e might have caused event e'.

Fix a subset G of A, the relation $\mapsto_G$ holds between events e, e' at a state s iff both e and e' appears in s, and

1. for some $i, j \in G$, e' is a *receive* event and e is the corresponding *send* event;
2. for some $i \in G$, events e, e' are both in $l_i(s)$ and either $e = e'$ or e comes earlier than e' in $l_i(s)$;
3. for some event e'', we have that $e \mapsto_G e''$ and $e'' \mapsto_G e'$ hold at s.

Note that $\mapsto_G$ is a partial order on events. We say that $(\mathcal{P}^\sigma, s) \models e \mapsto_G e'$ if $e \mapsto_G e'$ hold at s (we use the same notation for semantic and syntactic elements).

Now we prove that the potential causality relation $\mapsto_G$ respects the order $Prec$ of events by showing that the following validity holds in the class of m.p. QIS. This means that if event e is the "cause" of event e', then it is distributed knowledge among the agents that e happened before e'. Note that this is the right to left implication of Proposition 4.4.3 in [10]:

$$m.p.\ QIS \models \forall e, e'((e \mapsto_G e') \rightarrow D_G Prec(e,e'))$$

Proof. Assume that $(\mathcal{P}^\sigma, s) \models e \mapsto_G e'$. If e' is a *receive* event and e is the corresponding *send* event, then $l_i(s) = l_i(s')$ for all $i \in G$ implies $(\mathcal{P}^\sigma, s') \models H(e) \wedge H(e')$, and for $s'' \leq s'$, $(\mathcal{P}^\sigma, s'') \models Recd(i,j,\mu) \rightarrow Sent(j,i,\mu)$ by MP2'. Thus, $(\mathcal{P}^\sigma, s) \models D_G Prec(e,e')$.

If e, e' are both in $l_i(s)$ and either $e = e'$ or e comes earlier than e' in $l_i(s)$, then $l_i(s) = l_i(s')$ implies that $(\mathcal{P}^\sigma, s') \models H(e) \wedge H(e')$, and for $s'' \leq s'$, $(\mathcal{P}^\sigma, s'') \models H(e') \rightarrow H(e)$. It follows that $(\mathcal{P}^\sigma, s) \models K_i Prec(e,e')$, and by $D1$, $D2$, $(\mathcal{P}^\sigma, s) \models D_G Prec(e,e')$.

Finally, if there exists some event e'' such that $e \mapsto_G e''$ and $e'' \mapsto_G e'$, we can assume without loss of generality that this happens because we are in either the first or second case above. In both cases we have that $(\mathcal{P}^\sigma, s) \models D_G Prec(e,e'') \wedge D_G Prec(e'',e')$. Therefore, for every s', $l_i(s) = l_i(s')$ for all $i \in G$

implies $(\mathcal{P}^\sigma, s') \models H(e) \wedge H(e')$, and for $s'' \leq s'$, $(\mathcal{P}^\sigma, s'') \models H(e'') \rightarrow H(e) \wedge H(e') \rightarrow H(e'')$. By transitivity, $(\mathcal{P}^\sigma, s'') \models H(e') \rightarrow H(e)$. Thus, $(\mathcal{P}^\sigma, s) \models D_G Prec(e, e')$. □

The analysis of message-passing systems carried out in this section clearly shows the advantages of first-order modal formalisms in comparison with propositional ones. By means of language $\mathcal{L}_n^D$ we are able to formalise various constraints on m.p. systems, thereby signaling the general correctness of the approach. Most importantly, the right to left implication of Proposition 4.4.3 in [10] turned out out be a validity on the class of QIS modelling m.p. systems.

In the second part of this paper we will show that this expressivity gain is obtained while still retaining completeness of the logical formalism.

5 Axiomatisation

In this section we provide a sound and complete axiomatisation of systems of global states. Note that while it is customary in modal logic to axiomatise unvalued structures (hence our choice of $\mathcal{SGS}$), the same result applies to $\mathcal{QIS}$ as well. Technically, we first prove the completeness of the first-order multi-modal system $Q.S5_n^D$ with respect to Kripke frames. Then, by lemma 1 the completeness of $Q.S5_n^D$ with respect to $\mathcal{SGS}$ follows.

In [19] Kripke proved the completeness of monomodal $Q.S5$ (see also [12,18]). The novelty of this section consists in showing that the techniques in [11] for the completeness of propositional $S5_n^D$ can be straightforwardly extended to first-order for proving the completeness of $Q.S5_n^D$. Also, note that an independent completeness proof for $S5_n^D$ appeared in [24].

5.1 The System $Q.S5_n^D$

The system $Q.S5_n^D$ on the language $\mathcal{L}_n^D$ is a first-order multi-modal version of the propositional system $S5$. Although tableaux proof systems and natural deduction calculi are more suitable for automated theorem proving, Hilbert-style systems are easier to handle for the completeness proof. Hereafter we list the postulates of $Q.S5_n^D$; note that $\Rightarrow$ is the inference relation between formulas.

Definition 8. *The system $Q.S5_n^D$ on $\mathcal{L}_n^D$ contains the following schemes of axioms and inference rules:*

Taut	*every instance of classic propositional tautologies*
MP	$\phi \rightarrow \psi, \phi \Rightarrow \psi$
Dist	$K_i(\phi \rightarrow \psi) \rightarrow (K_i\phi \rightarrow K_i\psi)$
T	$K_i\phi \rightarrow \phi$
4	$K_i\phi \rightarrow K_iK_i\phi$
5	$\neg K_i\phi \rightarrow K_i\neg K_i\phi$
Nec	$\phi \Rightarrow K_i\phi$

Dist	$D_G(\phi \to \psi) \to (D_G\phi \to D_G\psi)$
T	$D_G\phi \to \phi$
4	$D_G\phi \to D_G D_G\phi$
5	$\neg D_G\phi \to D_G\neg D_G\phi$
D1	$D_{\{i\}}\phi \leftrightarrow K_i\phi$
D2	$D_G\phi \to D_{G'}$, *for* $G \subseteq G'$
Nec	$\phi \Rightarrow D_G\phi$
Ex	$\forall x\phi \to \phi[x/t]$
Gen	$\phi \to \psi[x/t] \Rightarrow \phi \to \forall x\psi$, *where* x *is not free in* ϕ
Id	$t = t$
Func	$t = t' \to (t''[x/t] = t''[x/t'])$
Subst	$t = t' \to (\phi[x/t] \to \phi[x/t'])$

We consider the standard definitions of *proof* and *theorem*: $\vdash \phi$ means that $\phi \in \mathcal{L}_n^D$ is a theorem in $Q.S5_n^D$. Moreover, $\phi \in \mathcal{L}_n^D$ is *derivable* in $Q.S5_n^D$ from a set Δ of formulas in $\mathcal{L}_n^D$ - $\Delta \vdash \phi$ in short - iff there are $\phi_1, \ldots, \phi_n \in \Delta$ such that $\vdash \phi_1 \wedge \ldots \wedge \phi_n \to \phi$. It is easy to check that the axioms of $Q.S5_n^D$ are valid on every Kripke frame and the inference rules preserve validity. As a consequence, we have the following soundness result.

Lemma 2 (Soundness). *The system $Q.S5_n^D$ is sound with respect to the class $\mathcal{K}$ of Kripke frames.*

By lemma 2 and the definition of validity on SGS, these implications hold:

$$Q.S5_n^D \vdash \phi \quad \Rightarrow \quad \mathcal{K} \models \phi \quad \Rightarrow \quad \mathcal{SGS} \models \phi$$

Thus, we have soundness also for the systems of global states.

Corollary 1 (Soundness). *The system $Q.S5_n^D$ is sound with respect to the class $\mathcal{SGS}$ of systems of global states.*

In the next paragraph we show that the axioms in $Q.S5_n^D$ are not only necessary, but also sufficient to prove all the validities on $\mathcal{SGS}$. In conclusion we show that the converse of the Barcan formula is provable in $Q.S5_n^D$. For a proof of BF, we refer to [12], p.138.

1. $\forall x\phi \to \phi$	Ex
2. $K_i(\forall x\phi \to \phi)$	from 1 by Nec
3. $K_i(\forall x\phi \to \phi) \to (K_i\forall x\phi \to K_i\phi)$	$Dist$
4. $K_i\forall x\phi \to K_i\phi$	from 2, 3 by MP
5. $K_i\forall x\phi \to \forall x K_i\phi$	from 4 by Gen

5.2 Completeness

We prove the completeness of $Q.S5_n^D$ by extending to first-order the proof for the propositional system $S5_n^D$ in [11]. The novelty of our result consists in showing that this method can be straightforwardly applied to first-order Kripke frames.

Specifically, we show that if $Q.S5_n^D$ does not prove a formula $\phi \in \mathcal{L}_n^D$, then the canonical model $\mathcal{M}^{Q.S5_n^D}$ for $Q.S5_n^D$ does not pseudo-validate ϕ. It is not guaranteed that the notion of pseudo-validity (to be defined below) coincides with plain validity, but by results in [11] we can obtain from $\mathcal{M}^{Q.S5_n^D}$ a Kripke model $\mathcal{M}'$ such that $\mathcal{M}^{Q.S5_n^D}$ pseudo-validates ϕ iff $\mathcal{M}' \models \phi$. Thus completeness follows.

In order to show the first part of the completeness result we rely on two lemmas: the *saturation lemma* and the *truth lemma*, whose statements require the following definitions: let Λ be a set of formulas in $\mathcal{L}_n^D$,

Λ is *consistent* iff for every $\phi \in \Lambda$, $\nvdash \neg\phi$;
Λ is *maximal* iff for every $\phi \in \mathcal{L}_n^D$, $\phi \in \Lambda$ or $\neg\phi \in \Lambda$;
Λ is *max-cons* iff Λ is consistent and maximal;
Λ is *rich* iff $\exists x\phi \in \Lambda$ implies $\phi[x/d] \in \Lambda$, for some constant $d \in \mathcal{L}_n^D$;
Λ is *saturated* iff Λ is max-cons and rich.

Assume that $Q.S5_n^D$ does not prove ϕ, then the set $\{\neg\phi\}$ is consistent, and by the saturation lemma below $\{\neg\phi\}$ can be extended to a saturated set:

Lemma 3 (Saturation [18]). *If Δ is a consistent set of formulas in $\mathcal{L}_n^D$, then it can be extended to a saturated set Π of formulas on some expansion $\mathcal{L}_n^{D+}$ obtained by adding an infinite set of new individual constants to $\mathcal{L}_n^D$.*

Now we introduce the canonical model for $Q.S5_n^D$. Note that $\wp^+(A)$ is the set of non-empty sets of agents.

Definition 9 (Canonical model). *The canonical model for $Q.S5_n^D$ on the language $\mathcal{L}_n^D$, with an expansion $\mathcal{L}_n^{D+}$, is a tuple $\mathcal{M}^{Q.S5_n^D} = \langle W, \{R_j\}_{j \in A \cup \wp^+(A)}, D, I\rangle$ such that*

- *W is the set of saturated sets of formulas in $\mathcal{L}_n^{D+}$;*
- *for $i \in A$, $w, w' \in W$, wR_iw' iff $\{\phi | K_i\phi \in w\} \subseteq w'$;*
- *for $G \subseteq A$, $w, w' \in W$, wR_Gw' iff $\{\phi | D_G\phi \in w\} \subseteq w'$;*
- *D is the set of equivalence classes $[v] = \{v' | v = v' \in w\}$, for every closed term $v \in \mathcal{L}_n^{D+}$;*
- *$I(f^k)([v_1], \ldots, [v_k]) = [f^k(v_1, \ldots, v_k)]$;*
- *$\langle [v_1], \ldots, [v_k]\rangle \in I(P^k, w)$ iff $P^k(v_1, \ldots, v_k) \in w$.*

If we assume that $Q.S5_n^D \nvdash \phi$, by the saturation lemma there exists a saturated set $w \supseteq \{\neg\phi\}$, so the set W of possible worlds is non-empty. Further, by definition of R_i and R_G, and axioms *Func* and *Subst*, we can show that the definition of $[v]$ is independent from w, so D is well defined. Since T, 4 and 5 are all axioms of $Q.S5_n^D$, the various R_i and R_G are equivalence relations. Moreover, from $D1$ and $D2$ it follows that $R_{\{i\}}$ is equal to R_i, and $R_G \subseteq \bigcap_{i \in G} R_i$. However, in general it is not the case that $R_G = \bigcap_{i \in G} R_i$. This remark gives the *rationale* for introducing the pseudo-satisfaction relation $\models^p$, defined as $\models$ but for the distributed knowledge operator D_G (in what follows we simply write $\mathcal{M}$ for $\mathcal{M}^{Q.S5_n^D}$):

$$(\mathcal{M}^\sigma, w) \models^p D_G\psi \text{ iff for every } w' \in W,\ wR_Gw' \text{ implies } (\mathcal{M}^\sigma, w') \models^p \psi$$

Now we can prove the *truth lemma* for the pseudo-satisfaction relation $\models^p$. In order to obtain this result we observe that for an assignment σ such that $\sigma(y_i) = [v_i]$, for $1 \leq i \leq n$, we have that $I^\sigma(t[\vec{y}]) = [t[\vec{y}/\vec{v}]]$.

Lemma 4 (Truth lemma). *For every $w \in \mathcal{M}$, $\phi \in \mathcal{L}_n^{D+}$, for $\sigma(y_i) = [v_i]$,*

$$(\mathcal{M}^\sigma, w) \models^p \phi[\vec{y}] \textit{ iff } \phi[\vec{y}/\vec{v}] \in w$$

Proof. The proof is by induction on the structure of $\phi \in \mathcal{L}_n^{D+}$.

$\phi = P^k(t_1, \ldots, t_k)$. By the definitions of pseudo-satisfaction and canonical interpretation $(\mathcal{M}^\sigma, w) \models^p P^k(t_1[\vec{y}], \ldots, t_k[\vec{y}])$ iff $\langle I^\sigma(t_1[\vec{y}]), \ldots, I^\sigma(t_k[\vec{y}])\rangle \in I(P^k, w)$ iff $\langle [t_1[\vec{y}/\vec{v}]], \ldots, [t_k[\vec{y}/\vec{v}]]\rangle \in I(P^k, w)$ iff $P^k(t_1[\vec{y}/\vec{v}], \ldots, t_k[\vec{y}/\vec{v}]) \in w$.

$\phi = \neg\psi, \psi \to \psi', \forall x\psi$. The cases for the propositional connectives follows by the maximality and consistency of the worlds in the canonical model; whereas for the universal quantifier, the induction step is proved by the richness of w.

$\phi = K_i\psi$. $\Leftarrow$ Assume that $K_i\psi[\vec{y}/\vec{v}] \in w$ and wR_iw'. By definition of R_i, $\psi[\vec{y}/\vec{v}] \in w'$ and by the induction hypothesis $(\mathcal{M}^\sigma, w') \models^p \psi[\vec{y}]$. Therefore $(\mathcal{M}^\sigma, w) \models^p K_i\psi[\vec{y}]$.

$\Rightarrow$ Assume that $K_i\psi[\vec{y}/\vec{v}] \notin w$. Note that the set $\{\phi | K_i\phi \in w\} \cup \{\neg\psi[\vec{y}/\vec{v}]\}$ is consistent. By standard techniques [12,18] we can extend it to a saturated set w' such that $\{\phi | K_i\phi \in w\} \cup \{\neg\psi[\vec{y}/\vec{v}]\} \subseteq w'$. This means that wR_iw' and $(\mathcal{M}^\sigma, w') \models^p \neg\psi[\vec{y}]$ by the induction hypothesis. Hence $(\mathcal{M}^\sigma, w) \not\models^p K_i\psi[\vec{y}]$.

$\phi = D_G\psi$. Similar to the previous case. □

We remarked that the canonical model might not satisfy $\bigcap_{i\in G} R_i = R_G$. However, it can be unwound to get a structure $\mathcal{M}'$ in such a way that the same formulas are valid [11]. More formally, given the canonical model $\mathcal{M} = \langle W, R, D, I\rangle$, there is another structure $\mathcal{M}^* = \langle W^*, R^*, D, I^*\rangle$ and a surjective function $h : W^* \to W$ such that (i) $\mathcal{M}^*$ is a tree, that is, for all $w, w' \in W^*$, there is at most one *reduced* path from w to w', (ii) wR_i^*w' implies $h(w)R_ih(w')$ and wR_G^*w' implies $h(w)R_Gh(w')$, and (iii) $\langle a_1, \ldots, a_k\rangle \in I^*(P^k, w)$ iff $\langle a_1, \ldots, a_k\rangle \in I(P^k, h(w))$.

In order to define $\mathcal{M}^*$ and h we need more definitions. Let w, w' be worlds in W, a *path from w to w'* is a sequence $\langle w_1, i_1, w_2, i_2, \ldots, i_{k-1}, w_k\rangle$ such that:

1. $w = w_1$ and $w' = w_k$;
2. $w_1, \ldots, w_k \in W$;
3. each i_j is either an agent or a set of agents;
4. $\langle w_j, w_{j+1}\rangle \in R^*_{i_j}$.

The *reduction* of a path $\langle w_1, i_1, w_2, i_2, \ldots, i_{k-1}, w_k\rangle$ is obtained by replacing each maximal consecutive subsequence $\langle w_q, i_q, w_{q+1}, i_{q+1}, \ldots, i_{r-1}, w_r\rangle$ where

$i_q = i_{q+1} = \ldots = i_{r-1}$ by $\langle w_q, i_q, w_r\rangle$. A path is said to be *reduced* is it is equal to its reduction.

We define W^* by induction. Let W_1^* be W, and define W_{k+1}^* as the set of worlds $v_{w,i,w'}$ such that $w \in W_k^*$, $w' \in W$ and i is an agent or group of agents. Let $W^* = \bigcup_{k\in\mathbb{N}} W_k^*$, then define $h : W^* \to W$ by letting $h(w) = w$, for $w \in W_1^*$ and $h(v_{w,i,w'}) = w'$, for $w \in W_k^*$. Further, R_i^* is the reflexive, transitive and symmetric closure of the relation defined for $w, w' \in W^*$ iff $w' = v_{w,i,w''}$ for some $w'' \in W$, and $h(w)R_i h(w')$. Finally, $I^*(P^k, w) = I(P^k, h(w))$. It can be checked that $\mathcal{M}^*$ and h satisfy (i)-(iii) above, we omit the proof for reasons of space and refer to [11] for the details. In particular, we can show what follows:

Lemma 5. *For $w \in W^*$, $\phi \in \mathcal{L}_n^D$,*

$$(\mathcal{M}^{*\sigma}, w) \models^p \phi \textit{ iff } (\mathcal{M}^\sigma, h(w)) \models^p \phi$$

Proof. The proof is by induction on the length of ϕ. If ϕ is an atomic formula, then the coimplication follows by the definition of I^*. The cases for the propositional connectives and the universal quantifier are straightforward.

$\phi = K_i\psi$. $\Leftarrow$ Suppose that $(\mathcal{M}^{*\sigma}, w) \not\models^p K_i\psi$, then there is a world $w' \in W^*$ such that wR_i^*w' and $(\mathcal{M}^{*\sigma}, w') \not\models^p \psi$. This means that $h(w)R_i h(w')$ and $(\mathcal{M}^\sigma, h(w')) \not\models^p \psi$ by induction hypothesis. Thus $(\mathcal{M}^\sigma, h(w)) \not\models^p K_i\psi$.

$\Rightarrow$ If $(\mathcal{M}^\sigma, h(w)) \not\models^p K_i\psi$, then there is a world $w' \in W$ such that $h(w)R_i w'$ and $(\mathcal{M}^\sigma, w') \not\models^p \psi$. By construction $v_{w,i,w'} \in W^*$, $h(v_{w,i,w'}) = w'$ and $wR_i^* v_{w,i,w'}$. By induction hypothesis $(\mathcal{M}^{*\sigma}, v_{w,i,w'}) \not\models^p \psi$, hence $(\mathcal{M}^{*\sigma}, w) \not\models^p K_i\psi$.

$\phi = D_G\psi$. Similar to the previous case. □

Now we make use of the structure $\mathcal{M}^*$ to define a Kripke model $\mathcal{M}'$ that does not validate the unprovable formula $\phi \in \mathcal{L}_n^D$. Define $\mathcal{M}' = \langle W', R', D', I'\rangle$ as follows:

- $W' = W^*$, $D' = D$ and $I' = I^*$;
- R_i' is the transitive closure of $R_i^* \cup \bigcup_{i\in G} R_G^*$.

Since the various R_i^* and R_G^* are reflexive and symmetric, it follows that R_i' is an equivalence relation, and therefore $\mathcal{M}'$ is based on a Kripke frame. Further, we can prove the following result:

Lemma 6. *For $\phi \in \mathcal{L}_n^D$,*

$$(\mathcal{M}'^\sigma, w) \models \phi \textit{ iff } (\mathcal{M}^{*\sigma}, w) \models^p \phi$$

Proof. Also this proof is by induction on the length of ϕ. If ϕ is an atomic formula, then the coimplication follows because $I' = I^*$. The cases for the propositional connectives are straightforward.

For $\phi = K_i\psi$ or $\phi = D_G\psi$, the inductive step goes as in the propositional case; we refer to [11] for a detailed proof.

$\phi = \forall x\psi$. If $(\mathcal{M}'^\sigma, w) \models \phi$, then for all $a \in D'$, $(\mathcal{M}'^{\sigma\binom{a}{x}}, w) \models \psi$. By induction hypothesis $(\mathcal{M}^{*\sigma\binom{a}{x}}, w) \models^p \psi$, and since $D' = D$, $(\mathcal{M}^{*\sigma}, w) \models^p \phi$. □

In conclusion, if $\phi \in \mathcal{L}_n^D$ is not provable in $Q.S5_n^D$, then the canonical model $\mathcal{M}$ pseudo-satisfies $\neg\phi$ by lemma 4. By lemma 5 also $\mathcal{M}^*$ pseudo-satisfies $\neg\phi$, and by the last result above $\mathcal{M}'$ does not validate ϕ. Thus, we state the following completeness result.

Theorem 1 (Completeness). *The system $Q.S5_n^D$ is complete with respect to the class $\mathcal{K}$ of Kripke frames.*

As a consequence, we have completeness also with respect to the systems of global states. In fact, if $\nvdash \phi$ then by Theorem 1 there exists a K-model $\mathcal{M} = \langle \mathcal{F}, I \rangle$, based on a Kripke frame $\mathcal{F}$, which falsifies ϕ. In order to prove that $\mathcal{SGS} \not\models \phi$ we have to find a quantified interpreted system $\mathcal{P}$ falsifying ϕ. Define $\mathcal{P}$ as $\langle g(\mathcal{F}), I \rangle$: by the definition of validity in QIS, $\mathcal{P} \models \phi$ iff $\mathcal{P}_f = \langle f(g(\mathcal{F})), I \rangle$ models ϕ, but by lemma 1 $f(g(\mathcal{F}))$ is isomorphic to $\mathcal{F}$. Hence $\mathcal{P} \not\models \phi$.

As a result, we have the following implications and a further completeness result:

$$\mathcal{SGS} \models \phi \quad \Rightarrow \quad \mathcal{K} \models \phi \quad \Rightarrow \quad Q.S5_n^D \vdash \phi$$

Corollary 2 (Completeness). *The system $Q.S5_n^D$ is complete with respect to the class $\mathcal{SGS}$ of systems of global states.*

By combining together the soundness and completeness theorems we compare directly the axiomatisation $Q.S5_n^D$ and the systems of global states, so we state our main result:

Corollary 3 (Soundness and Completeness). *A formula $\phi \in \mathcal{L}_n^D$ is valid on the class $\mathcal{SGS}$ of systems of global states iff ϕ is provable in $Q.S5_n^D$.*

6 Conclusions

As we argued in the Introduction, first-order modal formalisms offer expressivity advantages over propositional ones. But the cited explorations already carried out on this subject in MAS and, more in general, in knowledge representation and Artificial Intelligence, have so far fallen short of a deep and systematic analysis of the machinery even in the case of static epistemic logic.

In this paper we believe we have made a first attempt in this direction: the axiomatisation presented, even if limited to the static case, shows that the popular system $S5_n^D$ extends naturally to first-order. In carrying out this exercise we tried to remain as close as possible to the original semantics of interpreted systems, so that fine grained specifications of MAS may be expressed, as recent work on model checking interpreted systems demonstrates [13,28].

Different extensions of the present framework seem worth pursuing. First of all, it seems interesting to relax the assumption on the domain of quantification and admit a different domain $d(w)$ for every state w. Further, we could assume a different domain of quantification $d_a(w)$ for each agent a in a state w. In this case quantification would be agent-indexed, i.e. we would be using a different quantifier $\forall_a$ for every agent $a \in A$. In such an extended framework we should check whether the validities on m.p. QIS in section 4 still hold, and how to modify the completeness proof for $Q.S5_n^D$. Also, it would be of interest to explore the completeness issues resulting from term-indexing epistemic operators as in [21].

In an orthogonal dimension to the above, another significant extension would be to add temporal operators to the formalism. This would open the way for an exploration of axiomatisations for temporal/epistemic logic for MAS. While as reported in the Introduction we are not so concerned with the satisfiability problem, in doing so attention will have to be paid to the results in [16].

Acknowledgments

The research described in this paper was supported by the EC Framework 6 funded project CONTRACT (IST Project Number 034418). The authors would like to thank Wiebe van der Hoek for useful comments on an earlier version of this paper, as well as CLIMA's reviewers for their suggestions.

References

1. Armando, A., et al.: The Avispa tool for the automated validation of internet security protocols and applications. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 281–285. Springer, Heidelberg (2005)
2. Ågotnes, T., van der Hoek, W., Wooldridge, M.: Quantified Coalition Logic. In: Proceedings of IJCAI, Hyderabad, India, January 6-12, 2007, pp. 1181–1186 (2007)
3. Belardinelli, F., Lomuscio, A.: A Quantified Epistemic Logic to reason about Multi-Agent Systems. In: Proceedings of AAMAS 2007, Honolulu, Hawaii (2007)
4. Belardinelli, F., Lomuscio, A.: Quantified Epistemic Logics with Flexible Terms. In: LORI workshop on Logic, Rationality and Interaction, Beijing, August 5-9 (2007)
5. Bieber, P.: A logic of communication in hostile environments. In: CSFW, pp. 14–22 (1990)
6. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge, UP (2001)
7. Chagrov, A., Zakharyaschev, M.: Modal Logic. Oxford University Press, Oxford (1997)
8. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (1999)
9. Dams, M., Cohen, M.: A complete axiomatisation of knowledge and cryptography. In: LICS (2007)
10. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
11. Fagin, R., Halpern, J., Vardi, M.: What can machines know? on the properties of knowledge in distributed systems. J. ACM 39(2), 328–376 (1992)
12. Fitting, M., Mendelsohn, R.: First-order Modal Logic. Kluwer, Dordrecht (1999)
13. Gammie, P., van der Meyden, R.: MCK: Model checking the logic of knowledge. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 479–483. Springer, Heidelberg (2004)
14. Halpern, J., Fagin, R.: Modelling knowledge and action in distributed systems. Distributed Computing 3(4), 159–179 (1989)
15. Garson, J.: Quantification in modal logic. Handbook of Philosophical Logic, vol. 2, pp. 249–307 (1984)

16. Hodkinson, I., Wolter, F., Zakharyaschev, M.: Decidable fragment of first-order temporal logics. Ann. Pure Appl. Logic 106(1-3), 85–134 (2000)
17. van der Hoek, W., Wooldridge, M.: Tractable multiagent planning for epistemic goals. In: Proceedings of AAMAS 2002, pp. 1167–1174. ACM Press, New York (2002)
18. Hughes, G., Cresswell, M.: A New Introduction to Modal Logic. Routledge (1996)
19. Kripke, S.: A Completeness Theorem in Modal Logic. J. Sym. Log. 24, 1–14 (1959)
20. Lamport, L.: Time, Clocks, and the Ordering of Events in a Distributed System. Communication of the ACM 21(7), 558–565 (1978)
21. Lomuscio, A., Colombetti, M.: QLB: a quantified logic for belief. In: Jennings, N.R., Wooldridge, M.J., Müller, J.P. (eds.) ECAI-WS 1996 and ATAL 1996. LNCS, vol. 1193. Springer, Heidelberg (1996)
22. Lomuscio, A., Ryan, M.: On the relation between interpreted systems and kripke models. In: Wobcke, W., Pagnucco, M., Zhang, C. (eds.) Agents and Multi-Agent Systems Formalisms, Methodologies, and Applications. LNCS (LNAI), vol. 1441. Springer, Heidelberg (1997)
23. Lomuscio, A., Penczek, W., Wozna, B.: Bounded model checking knowledge and real time. Artificial Intelligence (to appear)
24. Meyer, J.-J., van der Hoek, W.: Making some issues of implicit knowledge explicit. Int. J. of Foundations of Computer Science 3(2), 193–223 (1992)
25. Meyer, J.-J.C., van der Hoek, W.: Epistemic Logic for AI and Computer Science. Cambridge University Press, Cambridge (1995)
26. Penczek, W., Lomuscio, A.: Verifying Epistemic Properties of multi-agent systems via bounded model checking. Fund. Inform. 55(2), 167–185 (2003)
27. Quine, W.v.O.: Quantifiers and Propositional Attitudes. Journal of Philosophy 53, 177–187 (1956)
28. Raimondi, F., Lomuscio, A.: Automatic verification of multi-agent systems by model checking via OBDDs. Journal of Applied Logic 5(2), 235–251 (2007)
29. Solanki, M.: A Compositional Framework for the Specification, Verification and Runtime Validation of Reactive Web Service. PhD thesis (2005)
30. Solanki, M., Cau, A., Zedan, H.: ASDL: a wide spectrum language for designing web services. In: WWW, pp. 687–696. ACM, New York (2006)
31. Spoletini, P.: Verification of Temporal Specification via Model Checking. PhD thesis, Politecnico di Milano, Dipartimento di Elettronica e Informatica (2006)
32. Sturm, H., Wolter, F., Zakharyaschev, M.: Monodic epistemic predicate logic. In: Brewka, G., Moniz Pereira, L., Ojeda-Aciego, M., de Guzmán, I.P. (eds.) JELIA 2000. LNCS (LNAI), vol. 1919, pp. 329–344. Springer, Heidelberg (2000)
33. Sturm, H., Wolter, F., Zakharyaschev, M.: Common knowledge and quantification. Economic Theory 19, 157–186 (2002)
34. Viganó, F.: A Framework for Model Checking Institutions. In: Edelkamp, S., Lomuscio, A. (eds.) MoChArt IV. LNCS (LNAI), vol. 4428, pp. 129–145. Springer, Heidelberg (2007)
35. Wolter, F., Zakharyaschev, P.: Decidable fragments of first-order modal logics. J. Symb. Log. 66(3), 1415–1438 (2001)

Analytic Cut-Free Tableaux for Regular Modal Logics of Agent Beliefs

Rajeev Goré[1],* and Linh Anh Nguyen[2],**

[1] The Australian National University and NICTA
Canberra ACT 0200, Australia
Rajeev.Gore@anu.edu.au
[2] Institute of Informatics, University of Warsaw
ul. Banacha 2, 02-097 Warsaw, Poland
nguyen@mimuw.edu.pl

Abstract. We present a sound and complete tableau calculus for a class $\mathcal{BReg}$ of extended regular modal logics which contains useful epistemic logics for reasoning about agent beliefs. Our calculus is cut-free and has the analytic superformula property so it gives a decision procedure. Applying sound global caching to the calculus, we obtain the first optimal (EXPTime) tableau decision procedure for $\mathcal{BReg}$. We demonstrate the usefulness of $\mathcal{BReg}$ logics and our tableau calculus using the wise men puzzle and its modified version, which requires axiom (5) for single agents.

1 Introduction

Context-free grammar logics are normal multimodal logics characterised by "inclusion axioms" of the form $[t]\varphi \supset [s_1]\dots[s_k]\varphi$, where $[t]$ and $[s_i]$ are modalities indexed by members t and s_i from some fixed set $\mathcal{MOD}$ of indices. Such logics are useful for modelling interactions between agents and groups of agents when indices from $\mathcal{MOD}$ denote agents/groups of agents. The general satisfiability problem of context-free grammar logics is undecidable [4], so researchers paid attention also to regular grammar logics, which are context-free grammar logics whose set of the corresponding grammar rules $t \to s_1 \dots s_k$ forms either a left/right linear grammar or is specified by finite automata [4,6,7,9]. Note that a left/right linear grammar can be transformed in polynomial time to an equivalent finite automaton, and vice versa, and checking whether a context-free grammar generates a regular language is undecidable (see, e.g., [18]). To avoid ambiguity, we refer to regular grammar logics specified by finite automata as *regular modal logics*. (Due to the polynomial transformation, there is no big difference between the two notions.)

* NICTA is funded by the Australian Government's Dept of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Centre of Excellence program.

** This work has been supported by Polish Ministry of Science and Higher Education under grant N N206 399334.

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 268–287, 2008.

Regular modal logics cannot be directly used for reasoning about belief due to the lack of axioms (D) and (5). It is commonly assumed that modal logics of belief invariably utilise the following:

Belief Consistency: Since $\langle t\rangle\varphi \equiv \neg[t]\neg\varphi$, axiom $(D) : [t]\varphi \supset \langle t\rangle\varphi$ states that agents cannot believe both φ and $\neg\varphi$.

Positive Introspection: Axiom $(4) : [t]\varphi \supset [t][t]\varphi$ states that agents are aware of what they believe.

Negative Introspection: Axiom $(5) : \langle t\rangle\varphi \supset [t]\langle t\rangle\varphi$, or alternatively $\neg[t]\psi \supset [t]\neg[t]\psi$, states that agents are aware of what they do not believe.

In [22], Nguyen studied a multimodal logic $KD4I_g5_a$ for reasoning about belief and common belief of agents in multi-agent systems. He adopted axioms (D) and (4) for all agents and groups of agents, and axiom $(I) : [t]\varphi \supset [s]\varphi$ for any (proper) super-group t of a group (or a single agent) s, but adopted axiom $(5) : \langle t\rangle\varphi \supset [t]\langle t\rangle\varphi$ only for single agents t. If t is a non-singleton group and s is a single agent belonging to t, then the contra-positive of $[t]\varphi \supset [s]\varphi$ gives us $\langle s\rangle\varphi \supset \langle t\rangle\varphi$. If axiom (5) were present for the proper group t then $\langle s\rangle\varphi$ would give us $[t]\langle t\rangle\varphi$. But $\langle s\rangle\varphi \supset [t]\langle t\rangle\varphi$ states that the belief of a single agent s leads to a belief among the whole super-group t about φ. Conversely, the contra-positive $\langle t\rangle[t]\varphi \supset [s]\varphi$ states that if the group jointly does not believe that it does not jointly believe φ, then single agent s believes φ, which seems equally absurd. The logic $KD4I_g5_a$ formalises the most important properties of belief and common belief but does not give an exact formulation of common belief. It is similar to the well-known modal logic with common belief $KD45_n^C$ [14] and the modal logic with mutual belief [1] but it lacks the induction rule for common belief.

In this paper, we study the class $\mathcal{BR}eg$ of *regular modal logics of agent beliefs*, which are regular modal logics extended with axioms (D) and (5), where axiom (5) is adopted only for modal indices with the "terminal KD45-condition" as for the case of single agents in $KD4I_g5_a$ (see Section 2.2 for a formal definition). We extend our tableau calculus for regular modal logics [9] to a tableau calculus for $\mathcal{BR}eg$ logics. Our calculus for $\mathcal{BR}eg$ is sound, complete, cut-free and has the analytic superformula property. Applying sound global caching [9,11] to it, we obtain the first optimal (EXPTime) tableau decision procedure for $\mathcal{BR}eg$. Note that standard tableau algorithms without global caching for regular modal logics and $\mathcal{BR}eg$ logics do not belong to EXPTime (but NEXPTime or 2NEXPTime).

The rest of this paper is structured as follows. In the end of this section, we present motivational examples and mention related works. In Section 2, we formally specify $\mathcal{BR}eg$ logics, introduce automaton-modal operators, and give definitions for tableau calculi. In Section 3, we present our tableau calculus for $\mathcal{BR}eg$, prove its soundness, and present "closed" tableaux for the motivational examples. In Section 4, we prove completeness of our tableau calculus. In Section 5, we present an EXPTime decision procedure with global caching for $\mathcal{BR}eg$. Section 6 concludes this work.

1.1 Motivational Examples

We will study two motivational examples about reasoning about beliefs and common beliefs of agents using our tableau calculus for $\mathcal{BR}eg$. The first one is the wise men puzzle, which is a famous benchmark introduced by McCarthy [19] for AI and has previously been studied in a considerable number of works (see [22] for some of them). The puzzle can be stated as follows (cf. [17]). A king wishes to know whether his three advisors (a, b, c) are as wise as they claim to be. Three chairs are lined up, all facing the same direction, with one behind the other. The wise men are instructed to sit down in the order a, b, c. Each of the men can see the backs of the men sitting before them (e.g. c can see a and b). The king informs the wise men that he has three cards, all of which are either black or white, at least one of which is white. He places one card, face up, behind each of the three wise men, explaining that each wise man must determine the colour of his own card. Each wise man must announce the colour of his own card as soon as he knows what it is. All know that this will happen. The room is silent; then, after a while, wise man a says "My card is white!".

For $x \in \{a, b, c\}$, let $[x]\varphi$ stand for "the wise man x believes in φ" and let p_x stand for "the card of x is white". Let g denote the group $\{a, b, c\}$ and let $[g]$ informally stand for a certain operator of "common belief" of the group g. Let L_{wmp} be the $\mathcal{BR}eg$ logic which extends K_n ($n = 4$ and $\mathcal{MOD} = \{g, a, b, c\}$) with the following axioms:

$$\begin{array}{llll} [x]\varphi \supset \langle x\rangle\varphi & \text{and} & [x]\varphi \supset [x][x]\varphi & \text{for } x \in \{g, a, b, c\}, \\ [g]\varphi \supset [x]\varphi & \text{and} & \langle x\rangle\varphi \supset [x]\langle x\rangle\varphi & \text{for } x \in \{a, b, c\}. \end{array}$$

The wise men puzzle can be formalised as follows:

- If y sits behind x then either x's card is white or y knows that x's card is not white: $\varphi_1 = [g](p_a \vee [b]\neg p_a)$, $\varphi_2 = [g](p_a \vee [c]\neg p_a)$, $\varphi_3 = [g](p_b \vee [c]\neg p_b)$.
- At least one of the men has a white card: $\varphi_4 = [g](p_a \vee p_b \vee p_c)$.
- As b and c say nothing, all of the wise men have a common belief that each of b and c does not believe that his own card is white, which is written in the negation normal form as: $\varphi_5 = [g]\langle b\rangle\neg p_b$, $\varphi_6 = [g]\langle c\rangle\neg p_c$.
  Here, we omit the temporal aspect to make the formalisation simple.
- The question is whether a believes that his card is white ($[a]p_a$). That is, whether $(\varphi_1 \wedge \ldots \wedge \varphi_6) \supset [a]p_a$ is L_{wmp}-valid. This is equivalent to whether the formula set $\Gamma_{wmp} = \{\varphi_1, \ldots, \varphi_6, \langle a\rangle\neg p_a\}$ is L_{wmp}-unsatisfiable.

As we will see, the wise men puzzle is solvable in a regular modal logic without axioms (D) and (5). More specifically, Γ_{wmp} is unsatisfiable in the logic L'_{wmp} obtained from L_{wmp} by discarding axioms (D) and (5). So, we introduce a modified version of the wise men puzzle for which axiom (5) is necessary[1]: "... The king places one card, face up, behind each of the three wise men, explaining that as soon as b or c knows that his own card is white or that the card of the man behind is white he must inform the man in the front about that[2], and as soon as

[1] It can be shown that the formula set Δ_{wmp} given below is L'_{wmp}-satisfiable.

[2] It does not matter whether the 3rd man notices this or not.

a knows who has a white card he must announce that. The question is whether a will know who has a white card." To formulate the new puzzle, we discard the formulae φ_5 and φ_6 and add to the formula set the following formulae:

$$\varphi'_5 = [g]([c]p_c \supset [b]p_c) \quad \varphi'_6 = [g]([b]p_b \supset [a]p_b) \quad \varphi'_7 = [g]([b]p_c \supset [a]p_c)$$

The new formula set is thus $\Delta_1 = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi'_5, \varphi'_6, \varphi'_7\}$. The question is whether $\Delta_1 \supset [a]p_a \lor [a]p_b \lor [a]p_c$ is L_{wmp}-valid, or equivalently, whether $\Delta_{wmp} = \Delta_1 \cup \{\langle a\rangle\neg p_a, \langle a\rangle\neg p_b, \langle a\rangle\neg p_c\}$ is L_{wmp}-unsatisfiable.

1.2 Related Works

In the previous work [9], we gave an analytic tableau calculus for regular modal logics and presented an EXPTime decision procedure for such logics. The class $\mathcal{BR}eg$ studied in this paper extends the class of regular modal logics with useful epistemic logics for reasoning about agent beliefs.

Regular grammar logics extended with axiom (5) belong to the class of regular grammar logics with converse [7], but such a statement with (5) replaced by (D) is not true. Adding axiom (D) to regular grammar logics with converse results in a class larger than $\mathcal{BR}eg$. In [7], Demri and de Nivelle gave a translation of the satisfiability problem for grammar logics with converse into the two-variable guarded fragment GF^2 of first-order logic, and showed that the general satisfiability problem for regular grammar logics with converse is in EXPTime. Assuming that it is easy to extend the translation with axiom (D) to cope with $\mathcal{BR}eg$, we cannot compare efficiency of the two approaches (for $\mathcal{BR}eg$) yet, but our tableau decision procedure for $\mathcal{BR}eg$ is certainly worth studying and experimenting.

Our tableau decision procedure for $\mathcal{BR}eg$ uses global caching, which we have used for regular modal logics [9], description logics $\mathcal{ALC}$ [10] and $\mathcal{SHI}$ [11]. Our adaptation of global caching for $\mathcal{BR}eg$ only slightly differs from the one used for the other logics. We include Section 5 on global caching for $\mathcal{BR}eg$ to increase readability and do not count it as a main contribution of this paper.

Other related works are works on regular grammar logics [4,6], works on PDL-like logics (e.g. [15,5,2]), and works on epistemic logics (e.g. [14,1]). However, the first two groups often lack axioms (D) and (5) and are not devoted to reasoning about epistemic states of agents, while the third group often adopts only some specific axioms but not the wide range of inclusion axioms. The class of "incestual multimodal logics" studied by Baldoni in [3] is large and contains the class $\mathcal{BR}eg$, but the general satisfiability problem for it is undecidable. Notice that such a problem is EXPTime-complete for regular grammar/modal logics [6], regular grammar logics with converse [7], PDL and converse-PDL (see, e.g., [15]), and $KD45^C_n$ with $n \geq 2$ [14]. As $\mathcal{BR}eg$ contains all regular modal logics, our EXPTime decision procedure presented in this paper for $\mathcal{BR}eg$ is optimal.

2 Preliminaries

2.1 Definitions for Multimodal Logics

Our modal language is built from two disjoint sets: $\mathcal{MOD}$ is a finite set of modal indices and $\mathcal{PROP}$ is a set of primitive propositions. We use p and q for (arbitrary) elements of $\mathcal{PROP}$ and use t and s for (arbitrary) elements of $\mathcal{MOD}$. *Formulae* of our *primitive language* are recursively defined using the BNF grammar: $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \supset \varphi \mid [t]\varphi \mid \langle t \rangle\varphi$.

A *Kripke frame* is a tuple $\langle W, \tau, (R_t)_{t \in \mathcal{MOD}} \rangle$, where W is a nonempty set of possible worlds, $\tau \in W$ is the current world, and each R_t is a binary relation on W, called the accessibility relation for $[t]$ and $\langle t \rangle$.[3] If $R_t(w, u)$ holds then we say that the world w sees world u via R_t.

A *Kripke model* is a tuple $\langle W, \tau, (R_t)_{t \in \mathcal{MOD}}, h \rangle$, where $\langle W, \tau, (R_t)_{t \in \mathcal{MOD}} \rangle$ is a Kripke frame and h is a function mapping worlds to sets of primitive propositions. For $w \in W$, the set of primitive propositions "true" at w is $h(w)$.

A *model graph* is a tuple $\langle W, \tau, (R_t)_{t \in \mathcal{MOD}}, H \rangle$, where $\langle W, \tau, (R_t)_{t \in \mathcal{MOD}} \rangle$ is a Kripke frame and H is a function mapping worlds to formula sets. We sometimes treat model graphs as models with the range of H restricted to $\mathcal{PROP}$.

Given a Kripke model $M = \langle W, \tau, (R_t)_{t \in \mathcal{MOD}}, h \rangle$ and a world $w \in W$, the *satisfaction relation* $\models$ is defined as usual for the classical connectives with two extra clauses for the modalities as below:

$$\begin{array}{lll} M, w \models [t]\varphi & \text{iff} & \forall v \in W.\ R_t(w, v) \text{ implies } M, v \models \varphi \\ M, w \models \langle t \rangle\varphi & \text{iff} & \exists v \in W.\ R_t(w, v) \text{ and } M, v \models \varphi. \end{array}$$

We say that φ *is satisfied at* w *in* M if $M, w \models \varphi$. We say that M *satisfies* φ and call M a *model of* φ if $M, \tau \models \varphi$.

If we consider only Kripke models, with no restrictions on R_t, we obtain a normal multimodal logic with a standard Hilbert-style axiomatisation K_n.

Note: We now assume that, if not stated otherwise, formulae are in *negation normal form*, where $\supset$ is translated away and $\neg$ occurs only directly before primitive propositions. It is well known that every formula φ has a logically equivalent formula φ' which is in negation normal form. We treat a finite set of formulae as the conjunction of its formulae.

2.2 A Class $\mathcal{BReg}$ of Regular Modal Logics of Agent Beliefs

A $\mathcal{BReg}$ logic is a normal multimodal logic L extending K_n with:

Inclusion Axioms: a set $IA(L)$ of inclusion axioms $[t]\varphi \supset [s_1] \ldots [s_k]\varphi$ with $k \geq 0$ whose corresponding grammar rules $t \to s_1 \ldots s_k$ jointly form a grammar $RG(L)$ specified by finite automata $(A_s)_{s \in \mathcal{MOD}}$ such that A_s for $s \in \mathcal{MOD}$ recognises the set of words derivable from s using the rules of $RG(L)$;[4]

[3] By writing $(R_t)_{t \in \mathcal{MOD}}$ instead of using R as a function that maps t to a binary relation R_t, we emphasize that the frame is for a multimodal logic.

[4] If $k = 0$ then the right hand side of $t \to s_1 \ldots s_k$ stands for the empty word ε. $\mathcal{MOD}$ is used both as the set of terminal symbols and the set of grammar variables.

Seriality Axioms: a set of seriality axioms $[t]\varphi \supset \langle t\rangle\varphi$ for every t of a set $\mathcal{DI}(L) \subseteq \mathcal{MOD}$ of D-indices;

Terminal KD45-Condition: a set of axioms $[t]\varphi \supset [t][t]\varphi \in IA(L)$ and $\langle t\rangle\varphi \supset [t]\langle t\rangle\varphi$ for every t of a set $\mathcal{EI}(L) \subseteq \mathcal{DI}(L)$ of E-indices, with the condition that $IA(L)$ contains no other inclusion axioms of the form $[t]\varphi \supset [s_1]\ldots[s_k]\varphi$ for $t \in \mathcal{EI}(L)$.

Recall that a *finite automaton* A is a tuple $\langle \Sigma, Q, I, \delta, F\rangle$, where: Σ is the alphabet (for our case $\Sigma = \mathcal{MOD}$); Q is a finite set of states; $I \subseteq Q$ is the set of initial states; $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation; and $F \subseteq Q$ is the set of accepting states. A *run* of A on a word $s_1 \ldots s_k$ is a finite sequence of states $q_0, q_1, \ldots, q_k$ such that $q_0 \in I$ and $\delta(q_{i-1}, s_i, q_i)$ holds for every $1 \leq i \leq k$. It is an *accepting run* if $q_k \in F$. We say that A *accepts* word w if there exists an accepting run of A on w. The set of all words accepted/recognised by A is $\mathcal{L}(A)$.

We do not require axiom (D) for every modal index in order to allow $\mathcal{BReg}$ to contain all regular modal logics. But, if every modal index denotes either an agent or a group of agents, then we can assume that $\mathcal{DI}(L) = \mathcal{MOD}$. We allow axiom (5) only for modal indices which satisfy the terminal KD45-condition. This restriction can be justified from practical considerations stated for $KD4I_g5_a$ in the introduction. It can be shown that the multimodal logics of belief $KDI4$, $KDI4_s$, $KD4I_g$, $KD4I_g5_a$ studied by Nguyen in [21,22], as well as $KD45_{(m)}$, belong to $\mathcal{BReg}$. The monomodal logics K, KD, T, $K4$, $KD4$, $S4$, $KD45$ belong to $\mathcal{BReg}$, but KB, KDB, B, $K5$, $KD5$, $K45$, $KB5$ ($KB4$), $S5$ do not.

Given two binary relations R_1 and R_2 over W, their relational composition $R_1 \circ R_2 = \{(x, z) \mid \exists y \in W.R_1(x, y) \;\&\; R_2(y, z)\}$ is also a binary relation over W.

The *L-frame restrictions* for a $\mathcal{BReg}$ logic L are the following restrictions:

- $R_{s_1} \circ \ldots \circ R_{s_k} \subseteq R_t$ if $s_1 \ldots s_k$ is accepted by A_t, where $t \in \mathcal{MOD}$ and $(A_s)_{s\in\mathcal{MOD}}$ are the finite automata specifying $RG(L)$;
- R_t is serial (i.e. $\forall u \exists w\, R_t(u, w)$) for each D-index $t \in \mathcal{DI}(L)$;
- R_t is transitive and euclidean (i.e. $\forall u, v, w\, R_t(u, v) \wedge R_t(v, w) \rightarrow R_t(u, w)$ and $\forall u, v, w\, R_t(u, v) \wedge R_t(u, w) \rightarrow R_t(w, v)$) for each E-index $t \in \mathcal{EI}(L)$.

A Kripke model is an *L-model* if its frame satisfies all L-frame restrictions. A formula φ is *L-satisfiable* if there exists an L-model satisfying it. A formula φ is *L-valid* if every L-model satisfies it. It can be shown that for a $\mathcal{BReg}$ logic L, a formula φ is L-valid iff it is derivable using the axiomatisation of L. (See [25] for the correspondence theory.)

2.3 Some Properties of $\mathcal{BReg}$ Logics

Let L be a $\mathcal{BReg}$ logic. For $t \in \mathcal{EI}(L)$, logic L contains the axiom $\langle t\rangle\varphi \supset [t]\langle t\rangle\varphi$, which can also be written as $\langle t\rangle[t]\psi \supset [t]\psi$. This latter axiom implies the inclusion axiom $[t][t]\psi \supset [t]\psi$ (because $t \in \mathcal{EI}(L) \subseteq \mathcal{DI}(L)$ and $[t][t]\psi \supset \langle t\rangle[t]\psi$ is L-valid), which corresponds to the grammar rule $tt \rightarrow t$. Let $eRG(L)$ be the grammar extending $RG(L)$ with rules $tt \rightarrow t$ for $t \in \mathcal{EI}(L)$. We call $eRG(L)$ the *extended grammar* of L. Syntactically, $eRG(L)$ is not a regular grammar.

Let L be a $\mathcal{BR}eg$ logic and let $(A_t)_{t\in\mathcal{MOD}}$ be the automata specifying the regular grammar $RG(L)$. An s-path from state q_0 to state q_n in A_t is a sequence of transitions (q_0, s, q_1), (q_1, s, q_2), $\ldots$, (q_{n-2}, s, q_{n-1}), (q_{n-1}, s, q_n) in δ_t, with $n \geq 1$. For each $t \in \mathcal{MOD} \setminus \mathcal{EI}(L)$, let A'_t be the automaton obtained from A_t by the following modification: for every $s \in \mathcal{EI}(L)$ and every s-path from state q_0 to state q_n in A_t, add the transition (q_0, s, q_n) to A'_t. We call the resulting automata A'_t, for $t \in \mathcal{MOD} \setminus \mathcal{EI}(L)$, the *automata specifying* $eRG(L)$. It should be clear that, for $t \in \mathcal{MOD} \setminus \mathcal{EI}(L)$, the word $s_1 \ldots s_k$ is accepted by A'_t iff $s_1 \ldots s_k$ is derivable from t using the grammar $eRG(L)$ since the modification simply adds "s-transitivity" for every $s \in \mathcal{EI}(L)$. Thus $eRG(L)$ can be treated as a regular grammar for starting symbols outside $\mathcal{EI}(L)$.

Example 1. The logic L_{wmp} specified in the introduction is a $\mathcal{BR}eg$ logic with $\mathcal{DI}(L_{wmp}) = \{g, a, b, c\}$, $\mathcal{EI}(L_{wmp}) = \{a, b, c\}$, and the extended grammar $eRG(L_{wmp})$ specified by the following finite automaton

$$A_g = \langle \mathcal{MOD}, \{0, 1\}, \{0\}, \{(0, x, 0), (0, x, 1) \mid x \in \mathcal{MOD}\}, \{1\}\rangle,$$

which accepts all non-empty finite words over $\mathcal{MOD}$.

Example 2. The logic $S4$ is a $\mathcal{BR}eg$ logic with $\mathcal{DI}(S4) = \mathcal{MOD} = \{a\}$, $\mathcal{EI}(S4) = \emptyset$, and the regular grammar $RG(S4)$ specified by the following finite automaton

$$A_a = \langle \{a\}, \{0\}, \{0\}, \{(0, a, 0)\}, \{0\}\rangle,$$

which accepts all finite sequences of a (in particular ε, and therefore is not an automaton for $K4$).

Lemma 1. *Let L be a $\mathcal{BR}eg$ logic and $t \in \mathcal{MOD} \setminus \mathcal{EI}(L)$. Then, for every $n \geq 1$, and every $s_1, s_2, \cdots, s_n \in \mathcal{MOD}$, if the word $s_1 \ldots s_n$ is derivable from t using the grammar $eRG(L)$ then the formula $[t]\varphi \supset [s_1] \ldots [s_n]\varphi$ is L-valid.*

Proof. By induction on the length of the derivation of $s_1 \ldots s_k$ from t using $eRG(L)$.

For a set Q of states of automaton A, the pair (A, Q) can be treated as the automaton obtained from A by replacing the set of initial states by Q. Thus, $\mathcal{L}(A, Q)$ denotes the language generated by (A, Q).

Let $A = \langle \mathcal{MOD}, Q_A, I_A, \delta_A, F_A\rangle$ and let ε denote the empty word. For any $Q \subseteq Q_A$, any $t \in \mathcal{MOD}$, and any word α over alphabet $\mathcal{MOD}$, define:

$$\begin{aligned} \delta_A(Q, t) &= \{q' \mid \exists q \in Q.(q, t, q') \in \delta_A\}, \\ \widetilde{\delta_A}(Q, \varepsilon) &= Q, \\ \widetilde{\delta_A}(Q, \alpha t) &= \delta_A(\widetilde{\delta_A}(Q, \alpha), t). \end{aligned}$$

For $A_s = \langle \mathcal{MOD}, Q_s, I_s, \delta_s, F_s\rangle$, we write δ_s (resp. $\widetilde{\delta_s}$) instead of δ_{A_s} (resp. $\widetilde{\delta_{A_s}}$).

Lemma 2. *Let L be a $\mathcal{BR}eg$ logic, $(A_t)_{t\in\mathcal{MOD}\setminus\mathcal{EI}(L)}$ the automata specifying $eRG(L)$, $s \in \mathcal{MOD} \setminus \mathcal{EI}(L)$, $A_s = \langle \mathcal{MOD}, Q_s, I_s, \delta_s, F_s\rangle$, and $Q = \widetilde{\delta_s}(I_s, \alpha_1) \cup \ldots \cup \widetilde{\delta_s}(I_s, \alpha_h)$ for some words $\alpha_1, \ldots \alpha_h$ over alphabet $\mathcal{MOD}$. Then:*

1. *If* $t \to s_1 \dots s_k$ *is a rule of* $RG(L)$ *then*
   $\mathcal{L}(A_s, \widetilde{\delta_s}(Q,t)) \subseteq \mathcal{L}(A_s, \widetilde{\delta_s}(Q, s_1 \dots s_k))$.
2. *If* $t \in \mathcal{EI}(L)$ *then* $\mathcal{L}(A_s, \widetilde{\delta_s}(Q,t)) = \mathcal{L}(A_s, \widetilde{\delta_s}(Q, tt))$.
3. *If* $\mathcal{L}(A_s, Q') \subseteq \mathcal{L}(A_s, Q'')$ *then* $\mathcal{L}(A_s, \widetilde{\delta_s}(Q',t)) \subseteq \mathcal{L}(A_s, \widetilde{\delta_s}(Q'',t))$.

Proof. Since $\delta_s(Q' \cup Q'', t) = \delta_s(Q',t) \cup \delta_s(Q'',t)$ for all $Q', Q'' \subseteq Q_s$, for assertions 1 and 2, we can assume $Q = \widetilde{\delta_s}(I_s, \alpha)$ for some word α.

1: Suppose β is a word over alphabet $\mathcal{MOD}$, and $\beta \in \mathcal{L}(A_s, \widetilde{\delta_s}(Q,t))$. Thus $\alpha t \beta \in \mathcal{L}(A_s)$. If $t \to s_1 \dots s_k$ is a rule of $RG(L)$, it follows that $\alpha s_1 \dots s_k \beta \in \mathcal{L}(A_s)$. Hence $\beta \in \mathcal{L}(A_s, \widetilde{\delta_s}(Q, s_1 \dots s_k))$.

2: Since $t \to tt$ is a rule of $RG(L)$ for all $t \in \mathcal{EI}(L)$, the first assertion gives one half of the inclusion. It therefore suffices to show that $\mathcal{L}(A_s, \widetilde{\delta_s}(Q,tt)) \subseteq \mathcal{L}(A_s, \widetilde{\delta_s}(Q,t))$. Let $\beta \in \mathcal{L}(A_s, \widetilde{\delta_s}(Q,tt))$. Thus $\alpha tt\beta \in \mathcal{L}(A_s)$. Because $t \in \mathcal{EI}(L)$, we have $tt \to t$ as a grammar rule of $eRG(L)$. It follows that t is derivable from tt using the grammar $eRG(L)$. Since A_s recognises the language derivable from s using $eRG(L)$, it follows that $\alpha t \beta \in \mathcal{L}(A_s)$. Hence $\beta \in \mathcal{L}(A_s, \widetilde{\delta_s}(Q,t))$.

3: The third assertion clearly holds.

2.4 Automaton-Modal Formulae

If A is a finite automaton, Q is a subset of the states of A, and φ is a formula in the primitive language then we call $[A,Q]$ a (universal) *automaton-modal operator* and $[A,Q]\varphi$ a formula in the extended language. Note that an automaton-modal operator can appear only at the beginning of a formula. Similar constructions were previously used in [15,16,9].[5]

Given a Kripke model $M = \langle W, \tau, (R_t)_{t \in \mathcal{MOD}}, h \rangle$ and $w_0 \in W$, define that $M, w_0 \models [A,Q]\varphi$ if $M, w_k \models \varphi$ for every path $w_0 R_{s_1} w_1 \dots w_{k-1} R_{s_k} w_k$ with $k \geq 0$ and $\widetilde{\delta_A}(Q, s_1 \dots s_k) \cap F_A \neq \emptyset$ (i.e. there exists a state of Q such that A accepts the word $s_1 \dots s_k$ when starting from that state).

From now on, by a *formula* we mean either a formula in the primitive language (as defined in Section 2.1) or an automaton-modal formula.

2.5 Definitions for Tableau Calculi

As in our previous works on tableau calculi [8,20], our tableaux trace their roots to Hintikka via [24]. A *tableau rule* σ consists of a numerator N above the line and a (finite) list of denominators $D_1, D_2, \dots, D_k$ (below the line) separated by vertical bars. The numerator is a finite formula set, and so is each denominator. As we shall see later, each rule is read downwards as "if the numerator is L-satisfiable, then so is one of the denominators". The numerator of each tableau

[5] In [9] we used the notation $(A,Q).\varphi$ instead of $[A,Q]\varphi$. We change the notation because the modal operator is a "universal" modal operator according to the next definition, and one can use $\langle A,Q \rangle\varphi$ for the dual existential modal operator.

Table 1. Tableau Rules for $\mathcal{BR}eg$ Logics

$$(\bot)\ \frac{X; p; \neg p}{\bot} \qquad (\wedge)\ \frac{X; \varphi \wedge \psi}{X; \varphi \wedge \psi; \varphi; \psi} \qquad (\vee)\ \frac{X; \varphi \vee \psi}{X; \varphi \vee \psi; \varphi \mid X; \varphi \vee \psi; \psi}$$

$$(D)\ \frac{X}{X; \langle t\rangle \top}\ \text{if } t \in \mathcal{DI}(L) \qquad (5)\ \frac{X; \langle t\rangle\varphi}{X; \langle t\rangle\varphi; [t]\langle t\rangle\varphi}\ \text{if } t \in \mathcal{EI}(L)$$

$$(\mathsf{aut})\ \frac{X; [t]\varphi}{X; [t]\varphi; [A_t, I_t]\varphi}\ \text{if } t \notin \mathcal{EI}(L) \qquad (\mathsf{add})\ \frac{X; [A_t, Q]\varphi}{X; [A_t, Q]\varphi; \varphi}\ \text{if } Q \cap F_t \neq \emptyset$$

$$(\mathsf{trans})\ \frac{X; \langle t\rangle\varphi}{\mathsf{trans}(X, t); \varphi}\ \text{if } t \notin \mathcal{EI}(L) \qquad (\mathsf{trans}_4)\ \frac{X; [t]Y; \langle t\rangle\varphi}{\mathsf{trans}(X, t); Y; [t]Y; \varphi}\ \text{if } t \in \mathcal{EI}(L)$$

rule contains one or more distinguished formulae called the *principal formulae*. A *tableau calculus* $\mathcal{C}L$ for a logic L is a finite set of tableau rules.

A $\mathcal{C}L$-tableau for a finite set X of formulae is a tree with root X whose nodes carry finite formula sets obtained from their parent nodes by instantiating a tableau rule with the proviso that if a child s carries a set Z and Z has already appeared on the branch from the root to s then s is an *end node*.

Let Δ be a set of tableau rules. We say that Y is *obtainable from* X *by applications of rules from* Δ if there exists a tableau for X which uses only rules from Δ and has a node that carries Y. A node to which no rule is applicable is also an end-node. A branch in a tableau is *closed* if its end node carries only $\bot$. A tableau is *closed* if every one of its branches is closed. A tableau is *open* if it is not closed. A finite formula set X is $\mathcal{C}L$-*consistent* if every $\mathcal{C}L$-tableau for X is open. If there is a closed $\mathcal{C}L$-tableau for X then X is $\mathcal{C}L$-*inconsistent*.

A tableau calculus $\mathcal{C}L$ is *sound* if for all finite formula sets X in the primitive language, X is L-satisfiable implies X is $\mathcal{C}L$-consistent. It is *complete* if for all finite formula sets X in the primitive language, X is $\mathcal{C}L$-consistent implies X is L-satisfiable. We say that a rule σ of $\mathcal{C}L$ is sound w.r.t. L if for every instance σ' of σ, if the numerator of σ' is L-satisfiable then so is one of the denominators of σ'. Any calculus $\mathcal{C}L$ containing only rules sound w.r.t. L is sound.

3 A Tableau Calculus for $\mathcal{BR}eg$ Logics

Fix a $\mathcal{BR}eg$ logic L and let $(A_t = \langle \mathcal{MOD}, Q_t, I_t, \delta_t, F_t\rangle)_{t \in \mathcal{MOD} \setminus \mathcal{EI}(L)}$ be the automata specifying $eRG(L)$. Recall that formulae are in negation normal form. We use X, Y to denote formula sets, use $[t]X$ to denote the set $\{[t]\varphi \mid \varphi \in X\}$, and use $\top$ to denote the truth constant with the usual semantics. We write $X; Y$ for $X \cup Y$, write $X; \varphi$ for $X \cup \{\varphi\}$, and $\varphi; \psi$ for $\{\varphi, \psi\}$.

The transfer of X through $\langle t\rangle$, denoted by $\mathsf{trans}(X, t)$, is:

$$\mathsf{trans}(X, t) = \{[A_s, \delta_s(Q, t)]\psi \mid [A_s, Q]\psi \in X\}.$$

The tableau calculus $\mathcal{C}L$ is given in Table 1. For intuition of the use of automaton-modal formulae in tableaux, we refer the reader to Section 2.3 of [9].

The last two rules (trans) and (trans_4) of the calculus are *transitional* rules, while the remaining rules except $(\bot)$ are *static* rules. The intuition of this sorting is that static rules keep us in the same world of the Kripke model under construction, while transitional rules take us to a new Kripke successor world.

Note that we include the principal formula of the static rules in their denominators.[6] Thus, the numerator of any static rule is a subset of every one of its denominators. A set X is *closed* w.r.t. a tableau rule if applying that rule to X gives back X as one of the denominators. We implicitly assume that a static rule is applied to X only when X is not closed w.r.t. that rule and treat this as an (additional) condition for applying the rule.

In comparison with the calculus given in [9] for regular modal logics, our calculus for $\mathcal{BR}eg$ contains 3 more rules (D), (5), (trans_4); the rule $(\vee)$ slightly changes as described in the above paragraph; and the rule (aut) is the same as (label). Note, however, that the automata used for (trans), (aut), (add) are now the ones specifying the extended grammar $eRG(L)$ but not $RG(L)$.

A tableau calculus $\mathcal{C}$ has the *analytic superformula* property iff to every finite set X we can assign a finite set $X^*_{\mathcal{C}}$ which contains all formulae that may appear in any tableau for X. We write $Sf(\varphi)$ for the set of all subformulae of φ, and $Sf(X)$ for the set $\bigcup_{\varphi \in X} Sf(\varphi) \cup \{\bot\}$. Our calculus $\mathcal{C}L$ has the analytic superformula property, with $X^*_{\mathcal{C}L} = Sf(X) \cup \{[A_t, Q]\varphi \mid [t]\varphi \in Sf(X) \;\&\; Q \subseteq Q_t\}$.

Lemma 3. *The tableau calculus $\mathcal{C}L$ is sound.*

Proof. We show that $\mathcal{C}L$ contains only rules sound w.r.t. L as follows. Suppose that the numerator of the considered rule is satisfied at a world w in an L-model $M = \langle W, \tau, (R_t)_{t \in \mathcal{MOD}}, h\rangle$. We have to show that at least one of the denominators of the rule is also L-satisfiable. For the static rules, we show that some denominator is satisfied at w itself. For the transitional rules (trans) and (trans_4), we show that its denominator is satisfied at some world reachable from w via R_t in the same L-model.

$(\bot), (\wedge), (\vee), (D), (5)$: These cases are obvious.

(aut): Suppose that $M, w \models X; [t]\varphi$. Let $w_0 = w, w_1, \ldots, w_k$ be worlds of M such that $R_{s_i}(w_{i-1}, w_i)$ holds for $1 \leq i \leq k$ and $s_1 \ldots s_k$ is accepted by A_t. By Lemma 1, $[t]\psi \supset [s_1]\ldots[s_k]\psi$ is L-valid. Hence $M, w_k \models \varphi$. Thus, $M, w \models [A_t, I_t]\varphi$.

(add): Suppose that $M, w \models X; [A_t, Q]\varphi$ and $Q \cap F_t \neq \emptyset$. Since $\widetilde{\delta_t}(Q, \varepsilon) = Q$, we have that $\widetilde{\delta_t}(Q, \varepsilon) \cap F_t \neq \emptyset$. Since $M, w \models [A_t, Q]\varphi$, it follows that $M, w \models \varphi$.

(trans): Suppose that $M, w \models X; \langle t\rangle\varphi$. Then there exists some u such that $R_t(w, u)$ holds and $M, u \models \varphi$. For each $[A_s, Q]\psi \in X$, we have $M, w \models [A_s, Q]\psi$, and by the semantics of automaton-modal formulae, it follows that $M, u \models [A_s, \delta_s(Q, t)]\psi$. Hence, the denominator is satisfied at u.

(trans_4): The proof for this case is similar to the proof for the case of (trans), with an additional justification that $[t]\psi \supset [t][t]\psi$ is an axiom of L when $t \in \mathcal{EI}(L)$.

[6] This allows an easier proof for soundness of global caching.

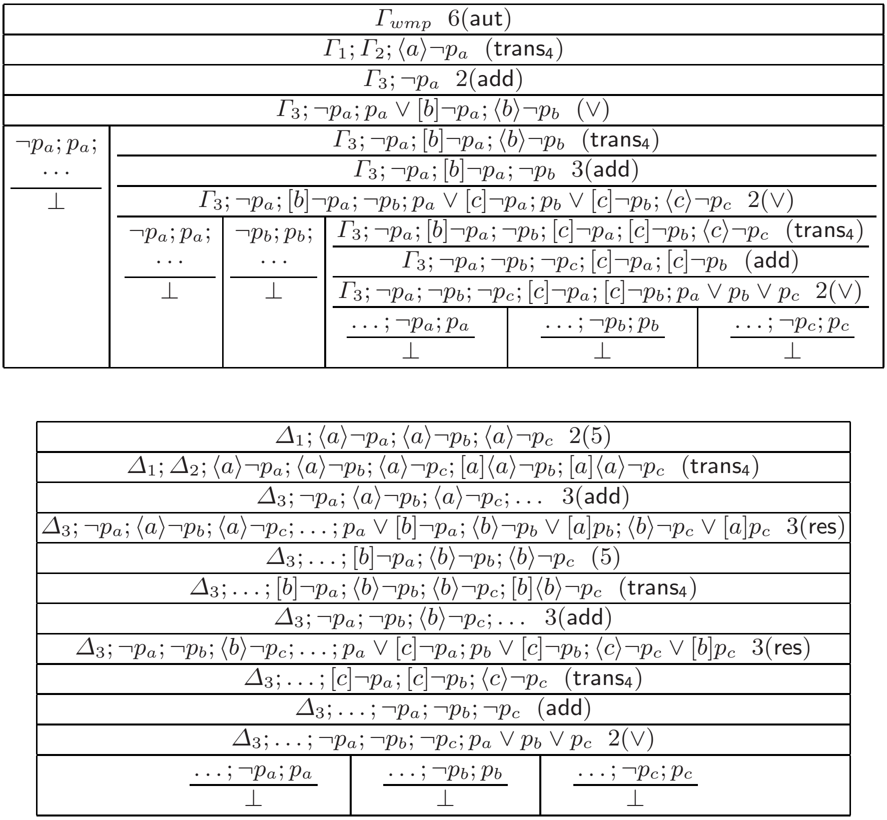

Γ_{wmp} 6(aut)
$\Gamma_1; \Gamma_2; \langle a\rangle\neg p_a$ (trans$_4$)
$\Gamma_3; \neg p_a$ 2(add)
$\Gamma_3; \neg p_a; p_a \vee [b]\neg p_a; \langle b\rangle\neg p_b$ ($\vee$)
$\neg p_a; p_a; \ldots$
$\bot$
$\Gamma_3; \neg p_a; [b]\neg p_a; \langle b\rangle\neg p_b$ (trans$_4$)
$\Gamma_3; \neg p_a; [b]\neg p_a; \neg p_b$ 3(add)
$\Gamma_3; \neg p_a; [b]\neg p_a; \neg p_b; p_a \vee [c]\neg p_a; p_b \vee [c]\neg p_b; \langle c\rangle\neg p_c$ 2($\vee$)
$\neg p_a; p_a; \ldots$
$\bot$
$\neg p_b; p_b; \ldots$
$\bot$
$\Gamma_3; \neg p_a; [b]\neg p_a; \neg p_b; [c]\neg p_a; [c]\neg p_b; \langle c\rangle\neg p_c$ (trans$_4$)
$\Gamma_3; \neg p_a; \neg p_b; \neg p_c; [c]\neg p_a; [c]\neg p_b$ (add)
$\Gamma_3; \neg p_a; \neg p_b; \neg p_c; [c]\neg p_a; [c]\neg p_b; p_a \vee p_b \vee p_c$ 2($\vee$)
$\ldots; \neg p_a; p_a$
$\bot$
$\ldots; \neg p_b; p_b$
$\bot$
$\ldots; \neg p_c; p_c$
$\bot$
$\Delta_1; \langle a\rangle\neg p_a; \langle a\rangle\neg p_b; \langle a\rangle\neg p_c$ 2(5)
$\Delta_1; \Delta_2; \langle a\rangle\neg p_a; \langle a\rangle\neg p_b; \langle a\rangle\neg p_c; [a]\langle a\rangle\neg p_b; [a]\langle a\rangle\neg p_c$ (trans$_4$)
$\Delta_3; \neg p_a; \langle a\rangle\neg p_b; \langle a\rangle\neg p_c; \ldots$ 3(add)
$\Delta_3; \neg p_a; \langle a\rangle\neg p_b; \langle a\rangle\neg p_c; \ldots; p_a \vee [b]\neg p_a; \langle b\rangle\neg p_b \vee [a]p_b; \langle b\rangle\neg p_c \vee [a]p_c$ 3(res)
$\Delta_3; \ldots; [b]\neg p_a; \langle b\rangle\neg p_b; \langle b\rangle\neg p_c$ (5)
$\Delta_3; \ldots; [b]\neg p_a; \langle b\rangle\neg p_b; \langle b\rangle\neg p_c; [b]\langle b\rangle\neg p_c$ (trans$_4$)
$\Delta_3; \neg p_a; \neg p_b; \langle b\rangle\neg p_c; \ldots$ 3(add)
$\Delta_3; \neg p_a; \neg p_b; \langle b\rangle\neg p_c; \ldots; p_a \vee [c]\neg p_a; p_b \vee [c]\neg p_b; \langle c\rangle\neg p_c \vee [b]p_c$ 3(res)
$\Delta_3; \ldots; [c]\neg p_a; [c]\neg p_b; \langle c\rangle\neg p_c$ (trans$_4$)
$\Delta_3; \ldots; \neg p_a; \neg p_b; \neg p_c$ (add)
$\Delta_3; \ldots; \neg p_a; \neg p_b; \neg p_c; p_a \vee p_b \vee p_c$ 2($\vee$)
$\ldots; \neg p_a; p_a$
$\bot$
$\ldots; \neg p_b; p_b$
$\bot$
$\ldots; \neg p_c; p_c$
$\bot$

Fig. 1. Closed $\mathcal{CL}$-Tableaux for the Wise Men Puzzle

3.1 Examples

In this subsection, we present closed tableaux for the formula sets formalising the wise men puzzle. Let L be the $\mathcal{BR}eg$ logic L_{wmp} defined in the introduction. Recall that the following automaton A_g specifies $eRG(L)$:

$$A_g = \langle \mathcal{MOD}, \{0,1\}, \{0\}, \{(0,x,0), (0,x,1) \mid x \in \mathcal{MOD}\}, \{1\}\rangle$$

In Figure 1, we give a closed $\mathcal{CL}$-tableau for the formula set Γ_{wmp}, which was specified in the introduction for formalising the wise men puzzle. In that tableau, for $1 \leq i \leq 6$, φ_i is the formula as in the introduction, ψ_i is the subformula of φ_i such that $\varphi_i = [g]\psi_i$, $\Gamma_1 = \{\varphi_1, \ldots, \varphi_6\}$, $\Gamma_2 = \{[A_g, \{0\}]\psi_1, \ldots, [A_g, \{0\}]\psi_6\}$, and $\Gamma_3 = \{[A_g, \{0,1\}]\psi_1, \ldots, [A_g, \{0,1\}]\psi_6\}$. Since the tableau calculus $\mathcal{CL}$ is sound, it follows that Γ_{wmp} is L-unsatisfiable.

For the modified version of the wise men puzzle, let $\varphi_1, \ldots, \varphi_4, \varphi_5', \varphi_6', \varphi_7'$ be the formulae as in the introduction. In the negation normal form, we have that:

$$\varphi_5' = [g](\langle c\rangle\neg p_c \vee [b]p_c) \quad \varphi_6' = [g](\langle b\rangle\neg p_b \vee [a]p_b) \quad \varphi_7' = [g](\langle b\rangle\neg p_c \vee [a]p_c)$$

For $1 \leq i \leq 7$, let ψ_i be the formula such that $\varphi_i = [g]\psi_i$ if $1 \leq i \leq 4$, and $\varphi'_i = [g]\psi_i$ if $i \in \{5, 6, 7\}$. Let $\Delta_1 = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi'_5, \varphi'_6, \varphi'_7\}$, $\Delta_2 = \{[A_g, \{0\}]\psi_i \mid 1 \leq i \leq 7\}$, and $\Delta_3 = \{[A_g, \{0, 1\}]\psi_i \mid 1 \leq i \leq 7\}$. Let $\overline{\varphi}$ denote the negation normal form of $\neg\varphi$. Note that the following rule is "derivable" using the rules of $\mathcal{C}L$:

$$(\mathsf{res})\ \frac{X; \varphi \vee \psi; \overline{\varphi}}{X; \varphi \vee \psi; \overline{\varphi}; \psi} \quad \text{or} \quad \frac{X; \psi \vee \varphi; \overline{\varphi}}{X; \psi \vee \varphi; \overline{\varphi}; \psi}$$

In Figure 1, we also give a closed $\mathcal{C}L$-tableau using the tableau rule (res) for the formula set $\Delta_{wmp} = \Delta_1 \cup \{\langle a \rangle \neg p_a, \langle a \rangle \neg p_b, \langle a \rangle \neg p_c\}$, which was specified in the introduction for formalising the modified version of the wise men puzzle. Since the tableau calculus $\mathcal{C}L$ is sound, it follows that Δ_{wmp} is L-unsatisfiable.

4 Completeness

4.1 Proving Completeness Via Model Graphs

Let L be a $\mathcal{BR}eg$ logic. We prove completeness of our calculus via model graphs following [24,8,20,9] by giving an algorithm that accepts a finite $\mathcal{C}L$-consistent formula set X in the primitive language and constructs an L-model graph (defined below) for X that satisfies each of its formulae at the appropriate world.

For a finite $\mathcal{C}L$-consistent formula set X, a formula set Y is called a *$\mathcal{C}L$-saturation of* X if Y is a $\mathcal{C}L$-consistent set obtainable from X by applications of the static rules of $\mathcal{C}L$ and closed w.r.t. the static rules of $\mathcal{C}L$.

Lemma 4. *Let X be a finite $\mathcal{C}L$-consistent formula set and Y a $\mathcal{C}L$-saturation of X. Then $X \subseteq Y \subseteq X^*_{\mathcal{C}L}$. Furthermore, there is an effective procedure that, given a finite $\mathcal{C}L$-consistent formula set X, constructs some $\mathcal{C}L$-saturation of X.*

Proof. Clearly, $X \subseteq Y \subseteq X^*_{\mathcal{C}L}$. Observe that if a static rule of $\mathcal{C}L$ is applicable to Y, then one of the corresponding instances of the denominators is $\mathcal{C}L$-consistent. Since Y is a $\mathcal{C}L$-saturation, Y is closed w.r.t. the static rules of $\mathcal{C}L$.

We construct a $\mathcal{C}L$-saturation of X as follows: let $Y = X$; while some static rule of $\mathcal{C}L$ is applicable to Y and has a corresponding denominator instance Z which is $\mathcal{C}L$-consistent and strictly contains Y, set $Y = Z$. At each iteration, $Y \subset Z \subseteq X^*_{\mathcal{C}L}$, so this process always terminates. Clearly, the resulting set Y is a $\mathcal{C}L$-saturation of X.

A model graph is an *L-model graph* if its frame is an L-frame. An L-model graph $\langle W, \tau, (R_t)_{t \in \mathcal{MOD}}, H \rangle$ is *saturated* if every $w \in W$ satisfies:

- if $\varphi \wedge \psi \in H(w)$ then $\{\varphi, \psi\} \subseteq H(w)$;
- if $\varphi \vee \psi \in H(w)$ then $\varphi \in H(w)$ or $\psi \in H(w)$;
- if $[t]\varphi \in H(w)$ and $R_t(w, u)$ holds then $\varphi \in H(u)$;
- if $\langle t \rangle \varphi \in H(w)$ then $\exists u \in W$ with $R_t(w, u)$ and $\varphi \in H(u)$.

A saturated model graph is *consistent* if no world contains $\perp$, and no world contains a pair of the form $\{p, \neg p\}$. Our model graphs are merely a data structure, while Rautenberg's are required to be saturated and consistent [24].

Lemma 5. *If $M = \langle W, \tau, (R_t)_{t \in \mathcal{MOD}}, H\rangle$ is a consistent saturated L-model graph, then M satisfies all formulae in the primitive language of $H(\tau)$.*

Proof. By proving $\varphi \in H(w)$ implies $M, w \models \varphi$ via induction on the length of φ.

Given a finite $\mathcal{C}L$-consistent set X in the primitive language, we construct a consistent saturated L-model graph $M = \langle W, \tau, (R_t)_{t \in \mathcal{MOD}}, H\rangle$ such that $X \subseteq H(\tau)$, thereby giving an L-model for X.

4.2 Constructing Model Graphs

Given X, the compact form $\mathsf{compact}(X)$ of X is the smallest set such that:

- if $\varphi \in X$ and φ is not of the form $[A_t, Q]\psi$ then $\varphi \in \mathsf{compact}(X)$;
- if $[A_t, Q]\psi \in X$ and $Q_1, \ldots, Q_k$ are all the sets such that $[A_t, Q_i]\psi \in X$ for $1 \leq i \leq k$, then $[A_t, Q_1 \cup \ldots \cup Q_k]\psi \in \mathsf{compact}(X)$.

Observe that the compact form does not affect the essence of $\mathcal{C}L$-tableaux. More specifically, if applying a $\mathcal{C}L$-tableau rule to X gives denominators $Y_1, \ldots, Y_k$, then applying that rule to $\mathsf{compact}(X)$ gives denominators $Z_1, \ldots, Z_k$ such that $\mathsf{compact}(Z_i) = \mathsf{compact}(Y_i)$ for $1 \leq i \leq k$. In particular, "compacting" preserves $\mathcal{C}L$-consistency and $\mathcal{C}L$-inconsistency.

For $t \in \mathcal{EI}(L)$ and $\langle t\rangle\varphi \in X$, define

$$\mathsf{trans}_4(X, \langle t\rangle\varphi) = \mathsf{trans}(X, t) \cup \{\psi, [t]\psi \mid [t]\psi \in X\} \cup \{\varphi\}.$$

For $t \in \mathcal{EI}(L)$, define

$$\begin{aligned}\mathsf{core}_5(X, t) = {} & \{[t]\varphi \mid [t]\varphi \in X\} \cup \{\langle t\rangle\varphi \mid \langle t\rangle\varphi \in X\} \cup \\ & \{[A_s, Q]\varphi \mid \exists \alpha, Q'.[A_s, Q']\varphi \in X \text{ and } Q = \widetilde{\delta_s}(I_s, \alpha t) \subseteq Q'\}.\end{aligned}$$

As shown in the next lemma, $\mathsf{core}_5(X, t)$ can be treated as the subset of X consisting of formulae that are preserved when travelling through edges of R_t, including edges forced by the euclidean frame restriction.

Lemma 6. *Let X be a $\mathcal{C}L$-saturation of some formula set and Y be a $\mathcal{C}L$-saturation of $\mathsf{trans}_4(X, \langle t\rangle\varphi)$ for some $\langle t\rangle\varphi \in X$ with $t \in \mathcal{EI}(L)$. Then $\mathsf{core}_5(X, t) \subseteq \mathsf{core}_5(Y, t)$.*

Proof. Due to the static rule (5), it suffices to show that if $[A_s, Q]\xi \in \mathsf{core}_5(X, t)$ then $[A_s, Q]\xi \in \mathsf{core}_5(Y, t)$. Suppose that $[A_s, Q]\xi \in \mathsf{core}_5(X, t)$. Thus, there exist α and Q' s.t. $Q = \widetilde{\delta_s}(I_s, \alpha t) \subseteq Q'$ and $[A_s, Q']\varphi \in X$. By definition of the set trans_4, there exists $[A_s, Q'']\varphi \in Y$ s.t. $\delta_s(Q', t) \subseteq Q''$. It follows that $\widetilde{\delta_s}(I_s, \alpha tt) \subseteq Q''$. By Lemma 2, $\widetilde{\delta_s}(I_s, \alpha t) = \widetilde{\delta_s}(I_s, \alpha tt)$, hence $[A_s, Q]\xi \in \mathsf{core}_5(Y, t)$.

A $\mathcal{CL}$-consistent set X is $\mathsf{core}_5(t)$-*saturated* if for every $\langle t\rangle\varphi \in X$ and every $\mathcal{CL}$-saturation Y of $\mathsf{trans}_4(X, \langle t\rangle\varphi)$ we have $\mathsf{core}_5(Y, t) = \mathsf{core}_5(X, t)$.

Algorithm 1 given below constructs a consistent saturated L-model graph for a finite $\mathcal{CL}$-consistent set X. In this algorithm, for each $t \in \mathcal{EI}(L)$: we find a $\mathsf{core}_5(t)$-saturated set U which is obtainable from $H(w)$ by applications of static $\mathcal{CL}$-rules and rule (trans_4) with the principal formula of the form $\langle t\rangle\psi$; we then create successors of w via R'_t to satisfy $\langle t\rangle$-formulae using $\mathsf{core}_5(U, t)$ as the content of w. But we do this in two different ways depending upon whether w has an R'_t-predecessor at this iteration.

To prove correctness of Algorithm 1, we use a data structure denoted by core^*_5 to store $\mathsf{core}_5(U, t)$ in $\mathsf{core}^*_5(w, t)$. Note that core_5 is a function, while core^*_5 is a table. In the algorithm, the worlds of the constructed model graph are marked either as *unresolved* or as *resolved*.

Algorithm 1

Input: a finite $\mathcal{CL}$-consistent set X of primitive language formulae.
Output: an L-model graph $M = \langle W, \tau, (R_t)_{t\in\mathcal{MOD}}, H\rangle$ of X.

```
1. Let $W = \{\tau\}$ and $R'_t = \emptyset$ for all $t \in \mathcal{MOD}$.
   Let $Y$ be a $\mathcal{CL}$-saturation of $X$ and let $H(\tau) = \mathsf{compact}(Y)$.
   Mark $\tau$ as unresolved.
2. While there are unresolved worlds, take one, say $w$, and do:
   (a) For every formula $\langle t\rangle\varphi$ in $H(w)$ with $t \notin \mathcal{EI}(L)$:
         i. Let $U = \mathsf{trans}(H(w), t) \cup \{\varphi\}$ be the result of applying rule $(\mathsf{trans})$ to
            $H(w)$, let $Y$ be a $\mathcal{CL}$-saturation of $U$, and let $Z = \mathsf{compact}(Y)$.
        ii. If $\exists u \in W$ on the path from the root to $w$ with $H(u) = Z$, then add the
            pair $(w, u)$ to $R'_t$. Otherwise, add a new world $u$ with content $Z$ to $W$,
            mark it as unresolved, and add the pair $(w, u)$ to $R'_t$.
   (b) For every $t \in \mathcal{EI}(L)$ such that $R'_t(v, w)$ does not hold for any $v$:
         i. Let $U$ be a $\mathcal{CL}$-saturation of $\mathsf{trans}_4(H(w), \langle t\rangle\top)$.
        ii. While there exist $\langle t\rangle\varphi \in U$ and a $\mathcal{CL}$-saturation $V$ of $\mathsf{trans}_4(U, \langle t\rangle\varphi)$ such
            that $\mathsf{core}_5(U, t) \subset \mathsf{core}_5(V, t)$, let $U = V$.
       iii. Let $\mathsf{core}^*_5(w, t) = \mathsf{core}_5(U, t)$.
        iv. For every $\langle t\rangle\varphi \in \mathsf{core}^*_5(w, t)$:
            Let $Y$ be a $\mathcal{CL}$-saturation of $\mathsf{trans}_4(\mathsf{core}^*_5(w, t), \langle t\rangle\varphi)$,
            let $Z = \mathsf{compact}(Y)$, and do the same as Step 2(a)ii.
   (c) For every $t \in \mathcal{EI}(L)$ such that $R'_t(v, w)$ holds for some $v$:
       Let $\mathsf{core}^*_5(w, t) = \mathsf{core}_5(H(w), t)$.
   (d) Mark $w$ as resolved.
3. Let $(R_t)_{t\in\mathcal{MOD}}$ be the least extension of $(R'_t)_{t\in\mathcal{MOD}}$ for $t \in \mathcal{MOD}$ such that
   $\langle W, \tau, (R_t)_{t\in\mathcal{MOD}}\rangle$ is an $L$-frame (note that the seriality conditions are cared by
   the tableau rule $(D)$ and need not to be considered here).
```

This algorithm always terminates: eventually, for every w, either w contains no $\langle t\rangle$-formulae, or there exists an ancestor with $H(u) = Z$ at Step 2(a)ii because all $\mathcal{CL}$-saturated sets are drawn from the finite and fixed set $X^*_{\mathcal{CL}}$.

4.3 Completeness Proof

Lemma 7. *The following assertions are invariants during execution of Step 3 of Algorithm 1 (when $(R'_t)_{t \in \mathcal{MOD}}$ are extended to $(R_t)_{t \in \mathcal{MOD}}$).*

1. *If $R_t(w, u)$ holds and $t \in \mathcal{EI}(L)$ then*
   $\mathsf{core}^*_5(w, t) = \mathsf{core}^*_5(u, t) = \mathsf{core}_5(H(u), t)$.
2. *If $R_t(w, u)$ holds then for every formula $[A_s, Q]\varphi \in H(w)$, there exists $[A_s, Q']\varphi \in H(u)$ such that $\mathcal{L}(A_s, \delta_s(Q, t)) \subseteq \mathcal{L}(A_s, Q')$.*

Proof. We first prove that if $t \in \mathcal{EI}(L)$ then the first assertion implies the second one. Hence, we need to prove the second assertion only for the case $t \notin \mathcal{EI}(L)$.

Suppose $t \in \mathcal{EI}(L)$, that the first assertion holds, and $[A_s, Q]\varphi \in H(w)$. Hence there exist words $\alpha_1, \ldots, \alpha_k$ such that $Q = \widetilde{\delta_s}(I_s, \alpha_1) \cup \ldots \cup \widetilde{\delta_s}(I_s, \alpha_k)$. By the computation of $\mathsf{core}^*_5(w, t)$, we have $[A_s, \widetilde{\delta_s}(I_s, \alpha_i t)]\varphi \in \mathsf{core}^*_5(w, t)$, for $1 \leq i \leq k$. Hence $[A_s, \widetilde{\delta_s}(I_s, \alpha_i t)]\varphi \in \mathsf{core}_5(H(u), t)$ for every $1 \leq i \leq k$. It follows that there exists $[A_s, Q']\varphi \in H(u)$ such that $\delta_s(Q, t) \subseteq Q'$, and thus $\mathcal{L}(A_s, \delta_s(Q, t)) \subseteq \mathcal{L}(A_s, Q')$.

We prove the assertions of the lemma by induction on the number of steps executed when extending R'_t for $t \in \mathcal{MOD}$ to R_t.

Consider the base case, when $R'_t(w, u)$ holds. For the first assertion, assume that $t \in \mathcal{EI}(L)$. Hence u must have been created from w via Step 2b. We have that $\mathsf{core}^*_5(w, t) = \mathsf{core}_5(H(u), t)$, because $\mathsf{core}^*_5(w, t)$ is $\mathsf{core}_5(t)$-saturated and u is created from w via R'_t using $\mathsf{core}^*_5(w, t)$ as the content of w. When u is resolved, we have that $\mathsf{core}^*_5(u, t) = \mathsf{core}_5(H(u), t)$ due to Step 2c. Hence the first assertion holds. The second assertion clearly holds for the case $t \notin \mathcal{EI}(L)$.

Consider the inductive step for the first assertion. If $R_t(w, u)$ is created from $R_t(w, v)$ and $R_t(v, u)$ then, by the inductive assumption, $\mathsf{core}^*_5(w, t) = \mathsf{core}^*_5(v, t)$ and $\mathsf{core}^*_5(v, t) = \mathsf{core}^*_5(u, t) = \mathsf{core}_5(H(u), t)$, which imply the first assertion. If $R_t(w, u)$ is created from $R_t(v, w)$ and $R_t(v, u)$ then, by the inductive assumption, $\mathsf{core}^*_5(v, t) = \mathsf{core}^*_5(w, t)$ and $\mathsf{core}^*_5(v, t) = \mathsf{core}^*_5(u, t) = \mathsf{core}_5(H(u), t)$, which imply the first assertion.

Consider the inductive step for the second assertion and the case when $t \notin \mathcal{EI}(L)$. Suppose that $R_t(w, u)$ is created from edges $R_{s_i}(w_{i-1}, w_i)$ with $1 \leq i \leq k$, $w = w_0$, $u = w_k$, due to an inclusion $R_{s_1} \circ \ldots \circ R_{s_k} \subseteq R_t$. Let $[A_s, Q]\varphi \in H(w)$. By Lemma 2(1), $\mathcal{L}(A_s, \delta_s(Q, t)) \subseteq \mathcal{L}(A_s, \widetilde{\delta_s}(Q, s_1 \ldots s_k))$. Let $Q_0 = Q$. For $i = 1, \ldots, k$, by the inductive assumption, there exists $[A_s, Q_i]\varphi \in H(w_i)$ such that $\mathcal{L}(A_s, \delta_s(Q_{i-1}, s_i)) \subseteq \mathcal{L}(A_s, Q_i)$. For $i = 2 \ldots k$, by Lemma 2(3), $\mathcal{L}(A_s, \widetilde{\delta_s}(Q, s_1 \ldots s_i)) \subseteq \mathcal{L}(A_s, Q_i)$ since $\mathcal{L}(A_s, \widetilde{\delta_s}(Q, s_1 \ldots s_{i-1})) \subseteq \mathcal{L}(A_s, Q_{i-1})$ and $\widetilde{\delta_s}(Q, s_1 \ldots s_i) = \delta_s(\widetilde{\delta_s}(Q, s_1 \ldots s_{i-1}), s_i)$. Hence $\mathcal{L}(A_s, \widetilde{\delta_s}(Q, s_1 \ldots s_k)) \subseteq \mathcal{L}(A_s, Q_k)$. It follows that $\mathcal{L}(A_s, \delta_s(Q, t)) \subseteq \mathcal{L}(A_s, Q_k)$. Choose $Q' = Q_k$.

Lemma 8. *Let X be a finite $\mathcal{C}L$-consistent set of formulae in the primitive language and $M = \langle W, \tau, (R_t)_{t \in \mathcal{MOD}}, H \rangle$ be the model graph for X constructed by Algorithm 1. Then M is a consistent saturated L-model graph satisfying X.*

Proof. It is clear that M is an L-model graph and for any $w \in W$, the set $H(w)$ is $\mathcal{C}L$-consistent. We want to show that M is a saturated model graph. It suffices to show that:

1. For all $w, u \in W$, if $[t]\varphi \in H(w)$ and $R_t(w,u)$ holds then $\varphi \in H(u)$.
2. For every $w \in W$, if $\langle t\rangle\varphi \in H(w)$ and $t \in \mathcal{EI}(L)$ then there exists $u \in W$ such that $R_t(w,u)$ holds and $\varphi \in H(u)$.

For the first assertion, suppose $[t]\varphi \in H(w)$ and $R_t(w,u)$ holds.

Case $t \notin \mathcal{EI}(L)$: Since $[t]\varphi \in H(w)$, there exists $[A_t, Q]\varphi \in H(w)$ with $Q \supseteq I_t$. By Lemma 7, there exists $[A_t, Q']\varphi \in H(u)$ such that $\mathcal{L}(A_t, \delta_t(I_t, t)) \subseteq \mathcal{L}(A_t, Q')$. Since $t \in \mathcal{L}(A_t)$, we have that $\varepsilon \in \mathcal{L}(A_t, \delta_t(I_t, t))$. Hence $\varepsilon \in \mathcal{L}(A_t, Q')$, which means $Q' \cap F_t \neq \emptyset$. Since $[A_t, Q']\varphi \in H(u)$, it follows that $\varphi \in H(u)$ by rule (add).

Case $t \in \mathcal{EI}(L)$: Since $[t]\varphi \in H(w)$, we have that $[t]\varphi \in \mathsf{core}_5^*(w,t)$. Since $R_t(w,u)$ holds, there exists v such that $R'_t(v,u)$ holds. By Lemma 7, $\mathsf{core}_5^*(w,t) = \mathsf{core}_5^*(u,t) = \mathsf{core}_5^*(v,t)$. Hence $[t]\varphi \in \mathsf{core}_5^*(v,t)$. Since $R'_t(v,u)$ holds, it follows that $\varphi \in H(u)$.

We now prove the second assertion. Suppose $\langle t\rangle\varphi \in H(w)$ and $t \in \mathcal{EI}(L)$. If $R'_t(v,w)$ does not hold for any v when w is resolved then w is connected via R'_t to a world u with $\varphi \in H(u)$ at Step 2b since $\langle t\rangle\varphi \in \mathsf{core}_5^*(w,t)$. Alternatively, suppose $R'_t(v,w)$ does hold for some v when w is resolved (at Step 2c). Since $\langle t\rangle\varphi \in H(w)$, we have $\langle t\rangle\varphi \in \mathsf{core}_5(H(w),t) = \mathsf{core}_5^*(v,t)$ by Lemma 7. Now v must have been considered at Step 2b in a previous iteration since this is the only way that an edge like $R'_t(v,w)$ is created. Since $\langle t\rangle\varphi \in \mathsf{core}_5^*(v,t)$, this iteration must also create a world u with $R'_t(v,u)$ such that $\varphi \in H(u)$. Then $R_t(w,u)$ must hold after Step 3 by euclideaness.

Theorem 1. *The calculus $\mathcal{C}L$ for $\mathcal{BR}eg$ logics is sound and complete.*

This theorem immediately follows from Lemmas 3 and 8.

We use Algorithm 1 only to prove completeness of the calculus $\mathcal{C}L$ for a $\mathcal{BR}eg$ logic L. It assumes that the input set X is $\mathcal{C}L$-consistent and is inefficient due to the naive computation of saturations and the limited caching. In the next section, we present Algorithm 2 with global caching for checking $\mathcal{C}L$-consistency of formula sets. Since the calculus $\mathcal{C}L$ is sound and complete, $\mathcal{C}L$-consistency coincides with L-satisfiability. Algorithm 2 explores the search space by building an and-or graph using the tableau rules of $\mathcal{C}L$. The content (label) of a node in the graph is a formula set in the compact form. Global caching means that for each possible content, at most one node with that content in the search space is expanded, and such an expansion is done at most once for that content. Global caching is one of the most useful optimisations for tableau decision procedures for modal logics [23]. Due to global caching and the compact form of nodes, Algorithm 2 has the optimal EXPTime complexity.

5 An EXPTime Decision Procedure with Global Caching for $\mathcal{BReg}$ Logics

In this section L denotes an $\mathcal{BReg}$ logic. In Figure 2 we give an algorithm for checking $\mathcal{C}L$-consistency which creates an and-or graph using the tableau rules of $\mathcal{C}L$ and global caching. A node in the constructed graph is a record with three attributes:

content: the formula set carried by the node
status: $\{\texttt{unexpanded}, \texttt{expanded}, \texttt{cons}, \texttt{incons}\}$
kind: $\{\texttt{and-node}, \texttt{or-node}\}$

Algorithm 2

```
Input: a finite set $X$ of primitive language formulae and an $\mathcal{BReg}$ logic $L$ with finite
    automata $(A_t)_{t \in \mathcal{MOD} \setminus \mathcal{EI}(L)}$ specifying the extended grammar $eRG(L)$
Output: an and-or graph $G = \langle V, E \rangle$ with $\tau \in V$ as the initial node such that
    $\tau.status = \texttt{cons}$ iff $X$ is $\mathcal{C}L$-consistent
Remark: We use "rule" to refer to a $\mathcal{C}L$-tableau rule.

1. create a new node $\tau$ with $\tau.content := X$ and $\tau.status := \texttt{unexpanded}$;
   let $V := \{\tau\}$ and $E := \emptyset$;
2. while $\tau.status \notin \{\texttt{cons}, \texttt{incons}\}$ and we can choose an unexpanded node $v \in V$ do:
   (a) $\mathcal{D} := \emptyset$;
   (b) if no rule is applicable to $v.content$ then $v.status := \texttt{cons}$
   (c) else if the rule $(\bot)$ is applicable to $v.content$ then $v.status := \texttt{incons}$
   (d) else if some static rule with only one denominator is applicable to $v.content$
       giving denominator $Y$ then $v.kind := \texttt{and-node}$, $\mathcal{D} := \{Y\}$
   (e) else if the rule $(\vee)$ is applicable to $v.content$ giving denominators $Y_1$ and $Y_2$
       (both different from $v.content$) then $v.kind := \texttt{or-node}$, $\mathcal{D} := \{Y_1, Y_2\}$
   (f) else
         i. $v.kind := \texttt{and-node}$,
        ii. for every transitional rule applicable to $v.content$ and for every possible
            application of the rule to $v.content$ giving denominator $Y$, add $Y$ to $\mathcal{D}$;
   (g) for every denominator $Y \in \mathcal{D}$ do
         i. let $Z = \mathsf{compact}(Y)$,
        ii. if some $w \in V$ has $w.content = Z$ then add edge $(v, w)$ to $E$
       iii. else let $w$ be a new node, set $w.content := Z$, $w.status := \texttt{unexpanded}$,
            add $w$ to $V$, and add edge $(v, w)$ to $E$;
   (h) if ($v.kind = \texttt{or-node}$ and one of the successors of $v$ has status $\texttt{cons}$)
       or ($v.kind = \texttt{and-node}$ and all the successors of $v$ have status $\texttt{cons}$) then
            $v.status := \texttt{cons}$, $propagate(G, v)$
   (i) else if ($v.kind = \texttt{and-node}$ and one of the successors of $v$ has status $\texttt{incons}$)
       or ($v.kind = \texttt{or-node}$ and all the successors of $v$ have status $\texttt{incons}$) then
            $v.status := \texttt{incons}$, $propagate(G, v)$
   (j) else $v.status := \texttt{expanded}$;
3. if $\tau.status \notin \{\texttt{cons}, \texttt{incons}\}$ then
       for every node $v \in V$ with $v.status \neq \texttt{incons}$, set $v.status := \texttt{cons}$;
```

Fig. 2. Checking $\mathcal{C}L$-Consistency Using Global Caching

```
Procedure propagate$(G, v)$
Parameters: an and-or graph $G = \langle V, E\rangle$ and $v \in V$ with $v.status \in \{$cons, incons$\}$
Returns: a modified and-or graph $G = \langle V, E\rangle$

1. $queue := \{v\}$;
2. while $queue$ is not empty do
3. (a) extract $x$ from $queue$;
   (b) for every $u \in V$ with $(u, x) \in E$ and $u.status =$ expanded do
         i. if ($u.kind =$ or-node and one of the successors of $u$ has status cons)
            or ($u.kind =$ and-node and all the successors of $u$ have status cons) then
               $u.status :=$ cons,  $queue := queue \cup \{u\}$
        ii. else if ($u.kind =$ and-node and one of the successors of $u$ has status incons)
            or ($u.kind =$ or-node and all the successors of $u$ have status incons) then
               $u.status :=$ incons,  $queue := queue \cup \{u\}$;
```

Fig. 3. Propagating Consistency and Inconsistency Through an And-Or Graph

To check whether a given finite formula set X is $\mathcal{C}L$-consistent, the initial node τ has content X and status `unexpanded`. The main while-loop continues processing nodes until the status of τ is determined to be in $\{$`cons`, `incons`$\}$, or until every node is expanded, whichever happens first.

The algorithm gives a preference to the rule $(\bot)$, then any one of the static unary rules, then the static binary rule $(\vee)$. If none of these are applicable, then it applies the transitional rules simultaneously. When a rule is applied, the algorithm categorises the numerator as either an `or-node` or an `and-node` with an `or-node` being inconsistent if every child is inconsistent and an `and-node` being inconsistent if at least one child is inconsistent.

The main difference with traditional methods appears at Step 2g: here, for every denominator, we first check whether an already existing node can act as a proxy for that denominator. If so, then we do not create that denominator, but merely insert an edge from the numerator to the existing proxy.

If these steps cannot determine the status of v as `cons` or `incons`, then its status is set to `expanded`. But if these steps do determine the status of a node v to be `cons` or `incons`, this information is itself propagated to the predecessors of v in the and-or graph G via the routine *propagate*(G, v), explained shortly.

The main loop ends when the status of the initial node τ becomes `cons` or `incons` or all nodes of the graph have been expanded. In the latter case, all nodes with status $\neq$ `incons` are given status `cons` (effectively giving the status *open* to tableau branches which loop).

The procedure *propagate* used in the above algorithm is specified in Figure 3. As parameters, it accepts an and-or graph G and a node v with (irrevocable) status `cons` or `incons`. The purpose is to propagate the status of v through the and-or graph and alter G to reflect the new information.

Initially, the queue contains only v. While the queue is not empty: a node x is extracted; the status of x is propagated to each predecessor u of x in an appropriate way; and if the status of u becomes (irrevocably) `cons` or `incons` then u is inserted into the queue for further propagation.

This construction thus uses both caching and propagation techniques. See [12] for the proofs of the following results.

Theorem 2. *Let L be an $\mathcal{BR}eg$ logic whose extended grammar is specified by finite automata $(A_t)_{t \in \mathcal{MOD} \setminus \mathcal{EI}(L)}$, X a finite set of primitive language formulae, and $G = \langle V, E \rangle$ the graph constructed by Algorithm 2 for X using $\mathcal{C}L$, with $\tau \in V$ as the initial node. Then X is $\mathcal{C}L$-consistent iff $\tau.status = \texttt{cons}$.*

Corollary 1. *Algorithm 2 is an EXPTime decision procedure for $\mathcal{BR}eg$ logics.*

6 Conclusions

We have given an analytic cut-free tableau calculus for a large class $\mathcal{BR}eg$ of epistemic logics for reasoning about agent beliefs. As demonstrated for the wise men puzzle and its modified version, $\mathcal{BR}eg$ logics are very useful for reasoning about mutual beliefs of agents. The class $\mathcal{BR}eg$ essentially extends the class of regular modal logics by allowing axioms (D) and (5) which are useful and sometimes necessary for practical applications.

Our tableau calculus for $\mathcal{BR}eg$ seems a simple extension of our tableau calculus for regular modal logics [9] using standard tableau rules to deal with axioms (D) and (5). Note, however, that non-trivial complications of the extension w.r.t. [9] lie in the use of finite automata specifying the extended grammar $eRG(L)$ instead of $RG(L)$ and the way of combining "regularity" of the used logic with axiom (5).

Applying global caching to our calculus, we obtain the first optimal (EXPTime) tableau decision procedure for $\mathcal{BR}eg$, which does not use cut rules.[7] Furthermore, it is easy to show that most of the well-known optimisation techniques for tableau decision procedures (as discussed in [13,23]) are applicable to this decision procedure.

Acknowledgements. Our thanks to the reviewers for their useful comments.

References

1. Aldewereld, H., van der Hoek, W., Meyer, J.-J.C.: Rational teams: Logical aspects of multi-agent systems. Fundamenta Informaticae 63(2–3), 159–183 (2004)
2. Baader, F.: Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In: Proceedings of IJCAI 1991, pp. 446–451 (1991)
3. Baldoni, M.: Normal multimodal logics with interaction axioms. In: Basin, D., D'Agostino, M., Gabbay, D.M., Viganò, L. (eds.) Labelled Deduction, pp. 33–57. Kluwer Academic Publishers, Dordrecht (2000)
4. Baldoni, M., Giordano, L., Martelli, A.: A tableau for multimodal logics and some (un)decidability results. In: de Swart, H. (ed.) TABLEAUX 1998. LNCS (LNAI), vol. 1397, pp. 44–59. Springer, Heidelberg (1998)

[7] We are not aware of other tableau decision procedures for $\mathcal{BR}eg$.

5. De Giacomo, G., Massacci, F.: Combining deduction and model checking into tableaux and algorithms for Converse-PDL. Information and Computation 117-137, 87–138 (2000)
6. Demri, S.: The complexity of regularity in grammar logics and related modal logics. Journal of Logic and Computation 11(6), 933–960 (2001)
7. Demri, S., de Nivelle, H.: Deciding regular grammar logics with converse through first-order logic. Journal of Logic, Language and Information 14(3), 289–329 (2005)
8. Goré, R.: Tableau methods for modal and temporal logics. In: D'Agostino, et al. (eds.) Handbook of Tableau Methods, pp. 297–396. Kluwer, Dordrecht (1999)
9. Goré, R., Nguyen, L.A.: A tableau system with automaton-labelled formulae for regular grammar logics. In: Beckert, B. (ed.) Proceedings of TABLEAUX 2005. LNCS (LNAI), vol. 3702, pp. 138–152. Springer, Heidelberg (2005)
10. Goré, R., Nguyen, L.A.: EXPTIME tableaux for ALC using sound global caching. In: Calvanese, D., et al. (eds.) Proceedings of Description Logics 2007, pp. 299–306 (2007)
11. Goré, R., Nguyen, L.A.: EXPTIME tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies. In: Olivetti, N. (ed.) TABLEAUX 2007. LNCS (LNAI), vol. 4548, pp. 133–148. Springer, Heidelberg (2007)
12. Goré, R., Nguyen, L.A.: The long version of this paper (2007), http://www.mimuw.edu.pl/~nguyen/papers.html
13. Goré, R., Nguyen, L.A.: Optimised EXPTIME tableaux for $\mathcal{ALC}$ using sound global caching, propagation and cutoffs. Manuscrip (2007), http://www.mimuw.edu.pl/~nguyen/papers.html
14. Halpern, J.Y., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artificial Intelligence 54, 319–379 (1992)
15. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
16. Horrocks, I., Sattler, U.: Decidability of SHIQ with complex role inclusion axioms. Artif. Intell. 160(1-2), 79–104 (2004)
17. Konolige, K.: Belief and incompleteness. Technical Report 319, SRI Inter. (1984)
18. Mateescu, A., Salomaa, A.: Formal languages: an introduction and a synopsis. Handbook of Formal Languages, vol. 1, pp. 1–40. Springer, Heidelberg (1997)
19. McCarthy, J.: First order theories of individual concepts and propositions. Machine Intelligence 9, 120–147 (1979)
20. Nguyen, L.A.: Analytic tableau systems and interpolation for the modal logics KB, KDB, K5, KD5. Studia Logica 69(1), 41–57 (2001)
21. Nguyen, L.A.: Multimodal logic programming. Theoretical Computer Science 360, 247–288 (2006)
22. Nguyen, L.A.: Reasoning about epistemic states of agents by modal logic programming. In: Toni, F., Torroni, P. (eds.) Proceedings of CLIMA VI. LNCS (LNAI), vol. 3900, pp. 37–56. Springer, Heidelberg (2006), http://www.mimuw.edu.pl/~nguyen/papers.html
23. Nguyen, L.A.: A modern tableau decision procedure with global caching for ALC. The paper together with the implemented prover (called TGC for Tableaux with Global Caching (March 2008), http://www.mimuw.edu.pl/~nguyen/papers.html
24. Rautenberg, W.: Modal tableau calculi and interpolation. JPL 12, 403–423 (1983)
25. van Benthem, J.: Correspondence theory. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic. II, pp. 167–247. Reidel, Dordrecht (1984)

EVOLP: An Implementation[⋆]

Martin Slota[1,2] and João Leite[2]

[1] Katedra aplikovanej informatiky, Univerzita Komenského, Slovakia
[2] CENTRIA, Universidade Nova de Lisboa, Portugal

Abstract. In this paper we present an implementation of EVOLP under the Evolution Stable Model semantics, based on the transformation defined in [1]. We also discuss optimizations used in the implementation.

1 Introduction

Evolving Logic Programming (EVOLP) [2] is a generalization of Answer Set Programming [3] to allow for the specification of a program's own evolution, in a single unified way, by permitting rules to indicate assertive conclusions in the form of program rules. Furthermore, EVOLP also permits, besides internal or self updates, for updates arising from the environment. The resulting language provides a simple and general formulation of logic program updating, particularly suited for Multi-Agent Systems [4,5].

The language of Evolving Logic Programs contains a special predicate assert/1 whose sole argument is a full-blown rule. Whenever an assertion assert(r) is true in a model, the program is updated with rule r. The process is then further iterated with the new program. Whenever the program semantics allows for several possible program models, evolution branching occurs, and several evolution sequences are made possible. This branching can be used to specify the evolution of a situation in the presence of incomplete information. Moreover, the ability of EVOLP to nest rule assertions within assertions allows rule updates to be themselves updated down the line. The ability to include assert literals in rule bodies allows for looking ahead on some program changes and acting on that knowledge before the changes occur. EVOLP also automatically and appropriately deals with the possible contradictions arising from successive specification changes and refinements (via Dynamic Logic Programming).

Elsewhere [1], we present a transformation that takes an evolving logic program P and a sequence of events $\mathcal{E}$ as input and outputs an equivalent normal logic program $P_{\mathcal{E}}$. The aim of this work is to present an implementation of EVOLP based on this transformation. The implementation can be easily integrated with other existing multi-agent programming frameworks such as Jason [6], 2APL [7] and 3APL [8], among others.

[⋆] This research has been funded by the European Commission within the 6th Framework Programme project REWERSE number 506779 (cf. http://rewerse.net). It was also supported by the Slovak Agency for Promotion Research and Development under the contract No. APVV-20-P04805.

F. Sadri and K. Satoh (Eds.): CLIMA VIII, LNAI 5056, pp. 288–298, 2008.

The remainder of this work is structured as follows: in Sect. 2 we introduce the syntax and semantics of EVOLP and the transformation from [1]; in Sect. 3 we present the implementation; in Sect. 4 we conclude and discuss future work.

2 Preliminaries

First we present the syntax and semantics of Dynamic Logic Programs and Evolving Logic Programs (EVOLP) and also a simple example that shows how EVOLP can be used to program a simple agent.

Let $\mathcal{L}$ be a set of propositional atoms. A *default literal* is an atom preceded by **not**. A *literal* is either an atom or a default literal. A *rule* r is an ordered pair $(H(r), B(r))$ where $H(r)$ (dubbed the *head of the rule*) is a literal and $B(r)$ (dubbed the *body of the rule*) is a finite set of literals. A rule with $H(r) = L_0$ and $B(r) = \{ L_1, L_2, \ldots, L_n \}$ will simply be written as

$$L_0 \leftarrow L_1, L_2, \ldots, L_n. \tag{1}$$

If $H(r) = A$ (resp. $H(r) = \textbf{not}\, A$) then $\textbf{not}\, H(r) = \textbf{not}\, A$ (resp. $\textbf{not}\, H(r) = A$). Two rules r, r' are *conflicting*, denoted by $r \bowtie r'$, iff $H(r) = \textbf{not}\, H(r')$. We will say a literal L appears in a rule (1) iff the set $\{ L, \textbf{not}\, L \} \cap \{ L_0, L_1, L_2, \ldots, L_n \}$ is non-empty.

A *generalized logic program* (GLP) over $\mathcal{L}$ is a set of rules. A literal appears in a GLP iff it appears in at least one of its rules.

An *interpretation* of $\mathcal{L}$ is any set of atoms $I \subseteq \mathcal{L}$. An atom A is true in I, denoted by $I \models A$, iff $A \in I$, and false otherwise. A default literal $\textbf{not}\, A$ is true in I, denoted by $I \models \textbf{not}\, A$, iff $A \notin I$, and false otherwise. A set of literals B is true in I iff each literal in B is true in I. Given an interpretation I we also define $I^- \overset{\text{def}}{=} \{ \textbf{not}\, A \mid A \in \mathcal{L} \setminus I \}$ and $I^* \overset{\text{def}}{=} I \cup I^-$. An interpretation M is a *stable model* of a GLP P iff $M^* = \text{least}(P \cup M^-)$ where $\text{least}(\cdot)$ denotes the least model of the definite program obtained from the argument program by treating all default literals as new atoms.

Definition 1. *A* dynamic logic program *(DLP) is a sequence of GLPs. Let $\mathcal{P} = (P_1, P_2, \ldots, P_n)$ be a DLP. We use $\rho(\mathcal{P})$ to denote the multiset of all rules appearing in the programs $P_1, P_2, \ldots, P_n$ and $\mathcal{P}^i$ $(1 \leq i \leq n)$ to denote the i-th component of $\mathcal{P}$, i.e. P_i. Given a DLP $\mathcal{P}$ and an interpretation I we define*

$$\text{Def}(\mathcal{P}, I) \overset{\text{def}}{=} \{ \textbf{not}\, A \mid (\nexists r \in \rho(\mathcal{P}))(H(r) = A \land I \models B(r)) \} \enspace , \tag{2}$$

$$\text{Rej}^j(\mathcal{P}, I) \overset{\text{def}}{=} \left\{ r \in \mathcal{P}^j \;\middle|\; (\exists k, r') \left(k \geq j \land r' \in \mathcal{P}^k \land r \bowtie r' \land I \models B(r')\right) \right\}, \tag{3}$$

$$\text{Rej}(\mathcal{P}, I) \overset{\text{def}}{=} \bigcup_{i=1}^{n} \text{Rej}^i(\mathcal{P}, I) \enspace . \tag{4}$$

An interpretation M is a (refined) dynamic stable model *of a DLP $\mathcal{P}$ iff $M^* = \text{least}([\rho(\mathcal{P}) \setminus \text{Rej}(\mathcal{P}, M)] \cup \text{Def}(\mathcal{P}, M))$.*

Definition 2. *Let $\mathcal{L}$ be a set of propositional atoms (not containing the predicate* assert/1*). The extended language $\mathcal{L}_{\text{assert}}$ is defined inductively as follows: – All propositional atoms in $\mathcal{L}$ are propositional atoms in $\mathcal{L}_{\text{assert}}$; – If r is a rule over $\mathcal{L}_{\text{assert}}$ then* $\text{assert}(r)$ *is a propositional atom in $\mathcal{L}_{\text{assert}}$; – Nothing else is a propositional atom in $\mathcal{L}_{\text{assert}}$. An* evolving logic program *over a language $\mathcal{L}$ is a GLP over $\mathcal{L}_{\text{assert}}$. An* event sequence *over $\mathcal{L}$ is a sequence of evolving logic programs over $\mathcal{L}$.*

Definition 3. *An* evolution interpretation of length *n of an evolving program P over $\mathcal{L}$ is a finite sequence $\mathcal{I} = (I_1, I_2, \ldots, I_n)$ of interpretations of $\mathcal{L}_{\text{assert}}$. The* evolution trace *associated with an evolution interpretation $\mathcal{I}$ of P is the sequence of programs $(P_1, P_2, \ldots, P_n)$ where $P_1 = P$ and $P_{i+1} = \{\, r \mid \text{assert}(r) \in I_i \,\}$ for all $i \in \{\, 1, 2, \ldots, n-1 \,\}$.*

Definition 4. *An evolution interpretation $\mathcal{M} = (M_1, M_2, \ldots, M_n)$ of an evolving logic program P with evolution trace $(P_1, P_2, \ldots, P_n)$ is an* evolution stable model *of P given an event sequence $(E_1, E_2, \ldots, E_n)$ iff for every $i \in \{\, 1, 2, \ldots, n \,\}$ M_i is a dynamic stable model of $(P_1, P_2, \ldots, P_{i-1}, P_i \cup E_i)$.*

Example 1. Let's consider a simple agent that fills glasses with water. If it receives a request from the environment (e.g. when somebody presses a button), it starts filling a glass with water. When the glass is full, it stops filling it. This evolving logic program encodes the described behaviour[1]: $P = \{\, \text{assert}(\text{fill} \leftarrow) \leftarrow \text{request}., \text{assert}(\textbf{not}\, \text{fill} \leftarrow) \leftarrow \text{full}. \,\}$. In each step the agent also receives an event from the environment. Let's consider a sequence of four events $\mathcal{E} = (E_1, E_2, E_3, E_4)$ where $E_1 = \{\, \text{request} \leftarrow . \,\}$, $E_2 = \emptyset$, $E_3 = \{\, \text{full} \leftarrow . \,\}$ and $E_4 = \emptyset$. The semantics of P w.r.t. $\mathcal{E}$ is a single sequence of four models (M_1, M_2, M_2, M_4) where $M_1 = \{\, \text{request}, \text{assert}(\text{fill} \leftarrow) \,\}$, $M_2 = \{\, \text{fill} \,\}$, $M_3 = \{\, \text{fill}, \text{full}, \text{assert}(\textbf{not}\, \text{fill} \leftarrow) \,\}$ and $M_4 = \emptyset$. Its meaning is that in the first step the agent receives a request in E_1, in the second it starts filling the glass, in the third it receives the signal to stop filling it and in the fourth it does nothing, i.e. it stops filling it.

Now we will complicate things a bit. Let's say the water was too warm and a cooling system was installed into the agent. Its behaviour also needs to be changed – it should ignore the request button until the water is cold enough. In EVOLP events can be used to program the agents. In this particular case the reprogramming is done through the event $E_5 = \{\, \text{assert}(\textbf{not}\, \text{assert}(\text{fill} \leftarrow) \leftarrow \textbf{not}\, \text{cold}) \leftarrow . \,\}$. The rule asserted in E_5 disallows filling the glass if the agent doesn't receive a signal that the water is cold enough. If, for example, the agent further receives the events $E_6 = \{\, \text{request} \leftarrow . \,\}$, $E_7 = \{\, \text{request} \leftarrow ., \text{cold} \leftarrow . \,\}$ and $E_8 = \emptyset$, then the corresponding models will be $M_6 = \{\, \text{request} \,\}$, $M_7 = \{\, \text{request}, \text{cold}, \text{assert}(\text{fill} \leftarrow) \,\}$ and $M_8 = \{\, \text{fill} \,\}$. In other words, the first request was ignored while the second was accepted because the water was already cold.

[1] With a small difference: the agent's reactions are always delayed one step.

The previous example is just a very simple one. The rules can be much more complex and can be used to capture very complicated behaviours. Evolution branching may also occur, allowing us to reason about incomplete information. For more examples the reader is referred to [4,5].

Now we will present the transformation which turns an evolving logic program P together with an event sequence $\mathcal{E}$ of length n into a normal logic program $P_{\mathcal{E}}$ over an extended language. In [1] we show that there is a one-to-one correspondence between the stable models of $P_{\mathcal{E}}$ and the evolution stable models of P given $\mathcal{E}$. The computational complexity of the transformation is also examined there.

First we need to define the extended language over which we will construct the resulting program:

$$\begin{aligned}\mathcal{L}_{\text{trans}} \overset{\text{def}}{=} & \left\{ A^j, A^j_{\text{neg}} \;\middle|\; A \in \mathcal{L}_{\text{assert}} \wedge 1 \le j \le n \right\} \\ & \cup \left\{ \text{rej}(A^j, i), \text{rej}(A^j_{\text{neg}}, i) \;\middle|\; A \in \mathcal{L}_{\text{assert}} \wedge 1 \le j \le n \wedge 0 \le i \le j \right\} \\ & \cup \{ u \} .\end{aligned}$$

Atoms of the form A^j and A^j_{neg} in the extended language allow us to compress the whole evolution interpretation (consisting of n interpretations of $\mathcal{L}_{\text{assert}}$, see Def. 3) into just one interpretation of $\mathcal{L}_{\text{trans}}$. Atoms of the form $\text{rej}(A^j, i)$ and $\text{rej}(A^j_{\text{neg}}, i)$ are needed for rule rejection simulation. The atom u will serve to formulate constraints needed to eliminate some unwanted models of $P_{\mathcal{E}}$.

To simplify the notation in the transformation's definition, we'll use the following conventions: Let L be a literal over $\mathcal{L}_{\text{assert}}$, *Body* a set of literals over $\mathcal{L}_{\text{assert}}$ and j a natural number. Then:

- If L is an atom A, then L^j is A^j and L^j_{neg} is A^j_{neg}.
- If L is a default literal **not** A, then L^j is A^j_{neg} and L^j_{neg} is A^j.
- $\mathit{Body}^j = \{ L^j \mid L \in \mathit{Body} \}$.

Definition 5. *Let P be an evolving logic program and $\mathcal{E} = (E_1, E_2, \ldots, E_n)$ an event sequence. By a* transformational equivalent *of P given $\mathcal{E}$ we mean the normal logic program $P_{\mathcal{E}} = P^1_{\mathcal{E}} \cup P^2_{\mathcal{E}} \cup \ldots \cup P^n_{\mathcal{E}}$ over $\mathcal{L}_{\text{trans}}$, where each $P^j_{\mathcal{E}}$ consists of these six groups of rules:*

1. ***Rewritten program rules.*** *For every rule $(L \leftarrow \mathit{Body}.) \in P$, $P^j_{\mathcal{E}}$ contains the rule*
$$L^j \leftarrow \mathit{Body}^j, \textbf{not}\, \text{rej}(L^j, 1).$$
2. ***Rewritten event rules.*** *For every rule $(L \leftarrow \mathit{Body}.) \in E_j$, $P^j_{\mathcal{E}}$ contains the rule*
$$L^j \leftarrow \mathit{Body}^j, \textbf{not}\, \text{rej}(L^j, j).$$
3. ***Assertable rules.*** *For every rule $r = (L \leftarrow \mathit{Body}.)$ over $\mathcal{L}_{\text{assert}}$ and all $i \in \{ 1, 2, \ldots, j-1 \}$ such that $(\text{assert}(r))^i$ is in the head of some rule of $P^i_{\mathcal{E}}$, $P^j_{\mathcal{E}}$ contains the rule*
$$L^j \leftarrow \mathit{Body}^j, (\text{assert}(r))^i, \textbf{not}\, \text{rej}(L^j, i+1).$$

4. ***Default assumptions.*** *For every atom* $A \in \mathcal{L}_{\mathrm{assert}}$ *such that* A^j *or* A^j_{neg} *appears in some rule of* $P^j_{\mathcal{E}}$ *(from the previous groups of rules),* $P^j_{\mathcal{E}}$ *also contains the rule*
$$A^j_{\mathrm{neg}} \leftarrow \mathbf{not}\,\mathrm{rej}(A^j_{\mathrm{neg}}, 0). \tag{5}$$
5. ***Rejection rules.*** *For every rule of* $P^j_{\mathcal{E}}$ *of the form*
$$L^j \leftarrow \mathit{Body}, \mathbf{not}\,\mathrm{rej}(L^j, i).$$[2]
$P^j_{\mathcal{E}}$ *also contains the rules*
$$\mathrm{rej}(L^j_{\mathrm{neg}}, p) \leftarrow \mathit{Body}. \tag{6}$$
$$\mathrm{rej}(L^j, q) \leftarrow \mathrm{rej}(L^j, i). \tag{7}$$
where:
(a) $p \leq i$ *is the largest index such that* $P^j_{\mathcal{E}}$ *contains a rule with the literal* $\mathbf{not}\,\mathrm{rej}(L^j_{\mathrm{neg}}, p)$ *in its body. If no such* p *exists, then* (6) *is not in* $P^j_{\mathcal{E}}$.
(b) $q < i$ *is the largest index such that* $P^j_{\mathcal{E}}$ *contains a rule with the literal* $\mathbf{not}\,\mathrm{rej}(L^j, q)$ *in its body. If no such* q *exists, then* (7) *is not in* $P^j_{\mathcal{E}}$.
6. ***Totality constraints.*** *For all* $i \in \{1, 2, \ldots, j\}$ *and every atom* $A \in \mathcal{L}_{\mathrm{assert}}$ *such that* $P^j_{\mathcal{E}}$ *contains rules of the form*
$$A^j \leftarrow \mathit{Body}_p, \mathbf{not}\,\mathrm{rej}(A^j, i).$$
$$A^j_{\mathrm{neg}} \leftarrow \mathit{Body}_n, \mathbf{not}\,\mathrm{rej}(A^j_{\mathrm{neg}}, i).$$
$P^j_{\mathcal{E}}$ *also contains the constraint*
$$u \leftarrow \mathbf{not}\,u, \mathbf{not}\,A^j, \mathbf{not}\,A^j_{\mathrm{neg}}.$$

Let's take a closer look at $P_{\mathcal{E}}$. It consists of n subprograms $P^1_{\mathcal{E}}, P^2_{\mathcal{E}}, \ldots, P^n_{\mathcal{E}}$. Each $P^j_{\mathcal{E}}$ is used to simulate the j-th evolution step on a separate set of atoms. In the first evolution step, the only rules that need to be simulated are the rules of P and E_1. They are simulated in the groups of rewritten program rules and rewritten event rules. Each occurrence of a literal L is written as L^1.

In the further steps there are more rules coming into play. In order to simulate the j-th evolution step, we need to simulate rules of P and E_j and also rules that could have been asserted in some previous evolution step. So apart from the rewritten program rules and rewritten event rules we also have assertable rules – whenever an atom $(\mathrm{assert}(r))^i$ appears in the head of some rule of $P^i_{\mathcal{E}}$ for some $i \in \{1, 2, \ldots, j-1\}$, the rule r is included in $P^j_{\mathcal{E}}$ in a special rewritten form. Apart from rewriting each literal L as L^j, we need to make sure the rule can only be used in case $(\mathrm{assert}(r))^i$ is actually inferred by some rule of $P^i_{\mathcal{E}}$. This is achieved by adding $(\mathrm{assert}(r))^i$ to the body of the rewritten form of r.

[2] It can be a rewritten program rule, a rewritten event rule or an assertable rule (default assumptions never satisfy the further conditions). The set *Body* contains all literals from the rule's body except the $\mathbf{not}\,\mathrm{rej}(L^j, i)$ literal.

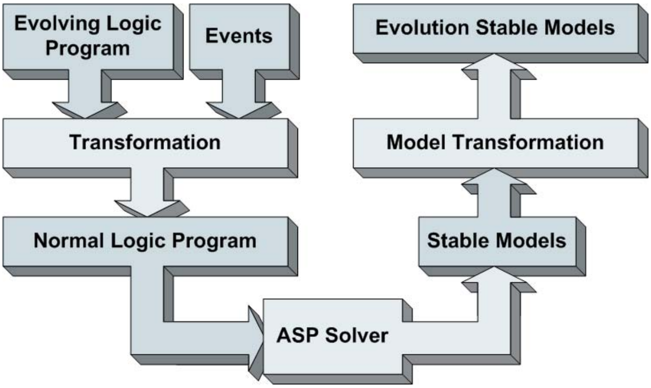


Fig. 1. Implementation of EVOLP using the transformation

Moreover, each of the mentioned rules contains one extra literal of the form **not** rej(L^j, i) in its body that marks it as a rule of level i. Rules with smaller levels can be rejected by higher level rules in case a conflict between their heads arises. Rules originating directly from P are of level 1 and rules from E_j are of level j, the highest level in $P^j_{\mathcal{E}}$. If a rule r was added because $(\text{assert}(r))^i$ appears in the head of some rule of $P^i_{\mathcal{E}}$ (i.e. the added rule is an assertable rule), it is marked with level $i+1$ because it can be asserted into the $(i+1)$-th step.

The other three groups of rules (default assumptions, rejection rules and totality constraints) are used to simulate the rule rejection mechanism behind the Refined Dynamic Stable Model semantics for Dynamic Logic Programming [9]. For more information on the transformation the reader is referred to [1].

3 Implementation of EVOLP

The described transformation, together with an ASP solver, can be used to implement EVOLP. Figure 1 shows how we can do this – we use the transformation to turn an evolving logic program and a sequence of events into an equivalent normal logic program. Then we use an ASP solver to find its stable models and reconstruct the evolution stable models of the original input.

One of the objectives was to make the implementation easy to integrate with existing multi-agent programming frameworks. Since many of them are written in Java, EVOLP is also implemented in Java. The implementation, however, uses Lparse as an external grounder and Smodels[3] as an external ASP solver.

[3] http://www.tcs.hut.fi/Software/smodels/

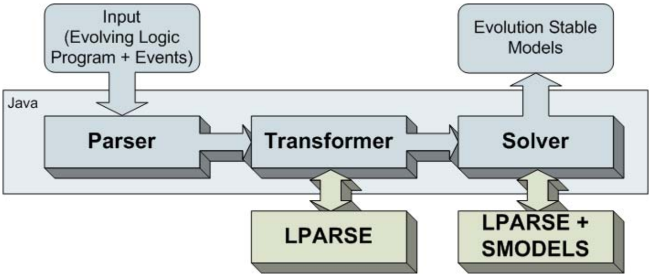


Fig. 2. Data processing steps

Variables are supported and restricted in the same way as in Lparse. More specifically, every variable must be bound in the positive part of the rule's body by some domain predicate[4].

As illustrated in Fig. 2, the input program is processed in three steps, each implemented by a separate Java class:

1. The input text is parsed and an object representation of the evolving logic program and events is created.
2. An equivalent normal logic program is produced according to a slightly more optimized version of the transformation defined in Def. 5. Lparse is used during the transformation to ground any variables in the input program. A more detailed description of this step is given below.
3. Lparse and Smodels are executed on the transformed program in a separate process. The resulting stable models are parsed and evolution stable models of the original input are reconstructed.

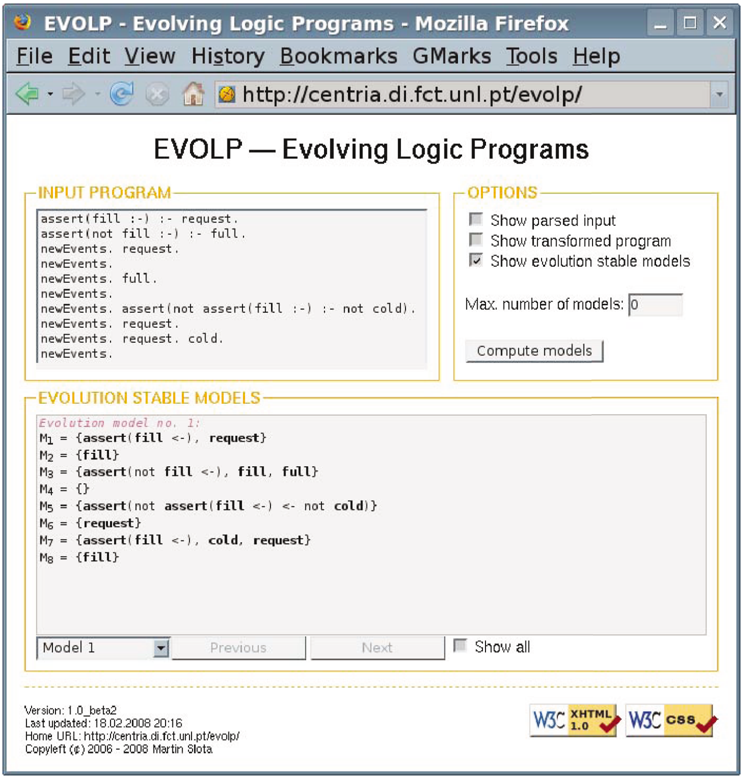
The implementation can currently run as a simple web application[5]. It can be used to enter an evolving logic program and compute some or all of its evolution stable models. Figure 3 shows a screenshot of the web application with the source and computed evolution stable models of the program from Example 1.

3.1 Implementation of the Transformation

The transformed program is constructed in incrementally and the partially constructed program is always used to construct the next part of the result. Let's assume we already constructed the programs $P^1_{\mathcal{E}}, P^2_{\mathcal{E}}, \ldots, P^{j-1}_{\mathcal{E}}$. In order to construct $P^j_{\mathcal{E}}$, we need to know what assertable rules to include. These can be inferred from a grounded version of $P^1_{\mathcal{E}} \cup P^2_{\mathcal{E}} \cup \ldots \cup P^{j-1}_{\mathcal{E}}$. Consequently, $P^j_{\mathcal{E}}$ can be constructed and the process can be iterated.

[4] Domain predicates are those that are defined without recursion or using positive recursion only. A more detailed description with an example can be found in the Lparse Manual which is included in the Lparse source package available at `http://www.tcs.hut.fi/Software/smodels/`.

[5] A demo runs at `http://centria.di.fct.unl.pt/evolp/`


EVOLP - Evolving Logic Programs - Mozilla Firefox
File Edit View History Bookmarks GMarks Tools Help
http://centria.di.fct.unl.pt/evolp/
Compute models
Model 1
Previous
Next
W3C XHTML 1.0
W3C CSS

EVOLP — Evolving Logic Programs

INPUT PROGRAM

```
assert(fill :-) :- request.
assert(not fill :-) :- full.
newEvents. request.
newEvents.
newEvents. full.
newEvents.
newEvents. assert(not assert(fill :-) :- not cold).
newEvents. request.
newEvents. request. cold.
newEvents.
```

OPTIONS

□ Show parsed input
□ Show transformed program
☑ Show evolution stable models

Max. number of models: 0

EVOLUTION STABLE MODELS

```
Evolution model no. 1:
M_1 = {assert(fill <-), request}
M_2 = {fill}
M_3 = {assert(not fill <-), fill, full}
M_4 = {}
M_5 = {assert(not assert(fill <-) <- not cold)}
M_6 = {request}
M_7 = {assert(fill <-), cold, request}
M_8 = {fill}
```

□ Show all

Version: 1.0_beta2
Last updated: 18.02.2008 20:16
Home URL: http://centria.di.fct.unl.pt/evolp/
Copyleft (ɔ) 2006 - 2008 Martin Slota

Fig. 3. Screenshot of the web application with the program from Example 1. The program source is showed in the bottom part and the single evolution stable model is listed in the upper part.

However, from a practical point of view, it is not possible to use Lparse to perform the transformation exactly as described. The problem is that the predicate rej/2 might be a non-domain predicate and still might contain variables in some of the second type of rejection rules (7). Fortunately, there is a better solution of the whole problem – it avoids the mentioned problem and is also more efficient. Instead of constructing the whole $P^j_{\mathcal{E}}$ before grounding it, only the first three groups of rules can be constructed and changed syntactically so that the grounder produces an appropriate grounded version. This way we only ground a part of $P^j_{\mathcal{E}}$ instead of grounding $P^1_{\mathcal{E}} \cup P^2_{\mathcal{E}} \cup \ldots \cup P^j_{\mathcal{E}}$. On the other hand, we need to take care of remembering the level of rules when they come out of the grounder and also their origin – whether they are assertable or not. This can be performed by adding dummy literals to their bodies before they are given to Lparse and filtering them out in their grounded versions.

3.2 Optimizations

The presented implementation includes some simple optimizations that prevent the generation of some unnecessary rules, even though these rules should be

```
class EvolpTest extends EvolpVarProcessor {
    protected void showMessage(String message) {
        // uncomment to see logging output
        // System.out.println("-- LOG MESSAGE: " + message);
    }

    public static void main(String [] args) {
        EvolpTest et = new EvolpTest();
        // set the program and events from Example 1 as input
        et.setInput(new StringReader(
            "assert(fill :-) :- request." +
            "assert(not fill :-) :- full." +
            "newEvents. request." +
            "newEvents." +
            "newEvents. full." +
            "newEvents." +
            "newEvents." +
                "assert(not assert(fill :-) :- not cold)." +
            "newEvents. request." +
            "newEvents. request. cold." +
            "newEvents."));
        // compute evolution stable models and give them one
        // by one to the TestConsumer.compute() method
        et.computeModels(new TestConsumer());
    }

    static class TestConsumer
        extends AbstractConsumer<EvolutionStableModel> {
        private int count = 0;

        public void consume(EvolutionStableModel model) {
            count++;
            System.out.print("Evolution stable model no. ");
            System.out.println(count);
            int step = 0;
            for (StableModel m : model) {
                step++;
                System.out.print("Step " + step + ": ");
                for (LpAtom a : m)
                    System.out.print(a.toString() + ", ");
                System.out.println();
            }
            System.out.println();
        }
    }
}
```

Fig. 4. Simple use of the implementation directly from Java

generated according to the formal definition of the transformation. In particular, we don't generate all default assumptions as in the definition, we generate a default assumption (5) only in case the atom A^j_{neg} appears in a body of some rewritten program rule, rewritten event rule or assertable rule.

Another optimization involves the second type of rejection rules (7). They are only generated if they are "reachable", i.e. a rule $\text{rej}(L^j, q) \leftarrow \text{rej}(L^j, i)$ is generated only in case $P_{\mathcal{E}}$ contains another rejection rule (6) of the form $\text{rej}(L^j, p) \leftarrow \mathit{Body}$ for some $p \geq i$.

There are also other ways of optimizing the resulting program, namely by sharing rules among evolution steps when possible. This can significantly reduce the size of the transformed program. We plan to include such an optimization in later versions of the implementation. Other possible optimizations include minimizing the amount of data transmitted between Lparse and Java by giving shorter (numeric) names to predicates and experimenting with modifications of the transformation that would produce equivalent normal programs that perform better with the current answer set solvers. We also plan to design a set of benchmark tests and perform them with different versions of the implementation.

3.3 Using the Implementation as a Library

Our implementation can easily be used as an external library from Java. If the binary .jar file from `http://centria.di.fct.unl.pt/evolp/` is included in the CLASSPATH, then the easiest way to use the implementation is to override the `lp.ui.EvolpVarProcessor` class. Figure 4 shows a complete source code of a Java class that uses the library to print out the evolution stable model of the program from Example 1. It produces the following output containing the expected evolution stable model:

```
Evolution stable model no. 1
Step 1: assert(fill <-), request,
Step 2: fill,
Step 3: assert(not fill <-), fill, full,
Step 4:
Step 5: assert(not assert(fill <-) <- not cold),
Step 6: request,
Step 7: assert(fill <-), cold, request,
Step 8: fill,
```

4 Conclusion and Future Work

We presented an implementation of EVOLP that is based on a transformation into an equivalent normal logic program. We also discussed some basic optimizations performed in the implementation.

We are currently extending the implementation to support strong negation, domain declarations, weight constraints, arithmetic predicates and other practical features. Apart from that, we plan to examine further optimizations of the implementation and perform some benchmark tests.

References

1. Slota, M., Leite, J.A.: EVOLP: Transformation-based semantics. In: Sadri, F., Satoh, K. (eds.) Proceedings of the 8th Workshop on Computational Logic in Multi-Agent Systems (CLIMA VIII) (2008) (in this volume)
2. Alferes, J.J., Brogi, A., Leite, J.A., Pereira, L.M.: Evolving logic programs. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 50–61. Springer, Heidelberg (2002)
3. Gelfond, M., Lifschitz, V.: Logic programs with classical negation. In: Warren, D., Szeredi, P. (eds.) Proceedings of the 7th international conference on logic programming, pp. 579–597. MIT Press, Cambridge (1990)
4. Leite, J.A., Soares, L.: Adding evolving abilities to a multi-agent system. In: Satoh, K., Inoue, K., Toni, F. (eds.) CLIMA 2006. LNCS (LNAI), vol. 4371, pp. 246–265. Springer, Heidelberg (2007)
5. Alferes, J.J., Brogi, A., Leite, J.A., Pereira, L.M.: Logic programming for evolving agents. In: Klusch, M., Ossowski, S., Omicini, A., Laamanen, H. (eds.) CIA 2003. LNCS (LNAI), vol. 2782, pp. 281–297. Springer, Heidelberg (2003)
6. Bordini, R.H., Wooldridge, M., Hübner, J.F.: Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology). John Wiley & Sons, Chichester (2007)
7. Dastani, M., Hobo, D., Meyer, J.-J.C.: Practical extensions in agent programming languages. In: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), ACM Press, New York (2007)
8. Hindriks, K., de Boer, F., van der Hoek, W., Meyer, J.-J.C.: Agent programming in 3APL. Int. J. of Autonomous Agents and Multi-Agent Systems 2(4), 357–401 (1999)
9. Alferes, J.J., Banti, F., Brogi, A., Leite, J.A.: The refined extension principle for semantics of dynamic logic programming. Studia Logica 79(1), 7–32 (2005)

Author Index