João Leite
Paolo Torroni (Eds.)

LNAI 3487

Computational Logic in Multi-Agent Systems

5th International Workshop, CLIMA V
Lisbon, Portugal, September 2004
Revised Selected and Invited Papers

Springer

Lecture Notes in Artificial Intelligence 3487

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

João Leite   Paolo Torroni (Eds.)

Computational Logic in Multi-Agent Systems

5th International Workshop, CLIMA V
Lisbon, Portugal, September 29-30, 2004
Revised Selected and Invited Papers

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

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

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

Library of Congress Control Number: 2005929660

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

ISSN 0302-9743
ISBN-10 3-540-28060-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-28060-6 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11533092 06/3142 5 4 3 2 1 0

Preface

The notion of agency has recently increased its influence in the research and development of computational logic based systems, while at the same time significantly gaining from decades of research in computational logic. Computational logic provides a well-defined, general, and rigorous framework for studying syntax, semantics and procedures, for implementations, environments, tools, and standards, facilitating the ever important link between specification and verification of computational systems.

The purpose of the Computational Logic in Multi-agent Systems (CLIMA) international workshop series is to discuss techniques, based on computational logic, for representing, programming, and reasoning about multi-agent systems in a formal way. Former CLIMA editions were conducted in conjunction with other major computational logic and AI events such as CL in July 2000, ICLP in December 2001, FLoC in August 2002, and LPNMR and AI-Math in January 2004.

The fifth edition of CLIMA was held Lisbon, Portugal, in September 29–30, 2004. We, as organizers, and in agreement with the CLIMA Steering Committee, opted for co-location with the 9th European Conference on Logics in Artificial Intelligence (JELIA 2004), wishing to promote the CLIMA research topics in the broader community of logics in AI, a community whose growing interest in multi-agent issues has been demonstrated by the large number of agent-related papers submitted to recent editions of JELIA.

The workshop received 35 submissions – a sensible increase from the previous edition. The submitted papers showed that the logical foundations of multi-agent systems are felt by a large community to be a very important research topic, upon which classical AI and agent-related issues are to be addressed.

In line with the high standards of previous CLIMA editions, the review process was very selective, the final acceptance rate being below 50%. A Program Committee of 24 top-level researchers from 11 countries and 12 additional reviewers selected 16 papers for presentation, authored by 46 researchers worldwide. The workshop program featured an invited lecture by Alessio Lomuscio (University College London) on Specification and Verification of Multiagent Systems, as well as a panel discussion organized by Marina de Vos (University of Bath) on Logic-Based Multi-agent Systems and Industry. Around 50 delegates attended the two-day event.

This book contains a selection, based on a second round of reviewing, of extended CLIMA V papers, and it starts with an invited contribution by Bożena Woźna and Alessio Lomuscio. The papers are divided into four parts: (i) foundations, (ii) architectures, (iii) interaction, and (iv) planning and applications. There follows a brief overview of the book.

Foundations. In the first paper of this book, *A Logic for Knowledge, Correctness, and Real Time*, Woźna and Lomuscio present and exemplify TCTLKD, a logic for knowledge, correctness and real time interpreted on real-time deontic interpreted systems, and extension to continuous time of deontic interpreted systems.

In *Dynamic Logic for Plan Revision in Intelligent Agents*, van Riemsdijk et al. present, with a sound and complete axiomatization, a dynamic logic for a propositional version of the agent programming language 3APL, tailored to handle the revision of plans.

Grossi et al. present in their paper *Contextual Taxonomies* a characterization of the notion of a taxonomy with respect to specific contexts, addressing problems stemming from the domain of normative system specifications for modelling multi-agent systems.

From Logic Programs Updates to Action Description Updates is where Alferes et al. propose a macro language for the language EVOLP and provide translations from some fragments of known action description languages into the newly defined one.

In *Dynamic Logic Programming: Various Semantics Are Equal on Acyclic Programs*, Homola investigates multi-dimensional dynamic logic programming, establishing some classes of programs for which several known semantics coincide.

Architectures. *Declarative Agent Control*, by Kakas et al., extends the architecture of agents based upon fixed, one-size-fits-all cycles of operation by providing a framework for the declarative specification of agent control in terms of *cycle theories*, which define possible alternative behaviors of agents.

In *Metareasoning for Multi-agent Epistemic Logics*, Arkoudas and Bringsjord present an encoding of a sequent calculus for a multi-agent epistemic logic in Athena, an interactive theorem proving system for many-sorted first-order logic, to enable its use as a metalanguage in order to reason about the multi-agent logic as an object language.

In *Graded BDI Models for Agent Architectures*, Casali et al. propose a general model for a graded BDI agent, specifying an architecture able to deal with the environment uncertainty and with graded mental attitudes.

Interaction. Dastani et al., in their article *Inferring Trust*, extend Liau's logic of Belief, Inform and Trust in two directions: with questions, and with a formalization of topics used to infer trust in a proposition from trust in another proposition.

In *Coordination Between Logical Agents*, Sakama and Inoue investigate on the use of answer set programming for belief representation, namely by addressing the problem of finding logic programs that combine the knowledge from different agents, while preserving some properties, useful to achieve agent coordination.

In *A Computational Model for Conversation Policies for Agent Communication*, Bentahar et al. propose a formal specification of a flexible persuasion protocol between autonomous agents, using an approach based on social commitments and arguments, defined as a combination of a set of conversation policies.

The last paper of this section is *Verifying Protocol Conformance for Logic-Based Communicating Agents*, by Baldoni et al., which describes a method for automatically verifying a form of "structural" conformance by translating AUML sequence diagrams into regular grammars and, then, interpreting the problem of conformance as a problem of language inclusion.

Planning and Applications. In the preliminary report *An Application of Global Abduction to an Information Agent Which Modifies a Plan Upon Failure*, Satoh uses a form of abductive logic programming called global abduction to implement an information agent that deals with the problem of plan modification upon action failure.

In *Planning Partially for Situated Agents*, Mancarella et al. use an abductive variant of the event calculus to specify planning problems as the base of their proposal for a framework to design situated agents capable of computing partial plans.

Han and Barber, in *Desire-Space Analysis and Action Selection for Multiple Dynamic Goals*, use macro actions to transform the state space for the agent's decision problem into the desire space of the agent. Reasoning in the latter allows us to approximately weigh the costs and benefits of each of the agent's goals at an abstract level.

Hirsch et al. conclude this book with the article *Organising Software in Active Environments*, in which they show how logic-based multi-agent systems are appropriate to model active environments. They do so by illustrating how the structuring of the "agent space" can represent both the physical and virtual structures of an application.

We would like to conclude with a glance at the future of this workshop series. The sixth CLIMA edition is being organized by Francesca Toni and Paolo Torroni, and will take place at the City University of London, UK, in June 27–29, 2005, in conjunction with the EU-funded SOCS Project Dissemination Workshop. CLIMA VI will feature a tutorial program and a competition, besides the usual technical content based on the presentation of papers.

We can not miss this opportunity to thank the authors and delegates, who made of CLIMA a very interesting and fruitful event; our generous Program Committee members who did not skimp on time to help us put together a very rich volume after two rounds of reviewing, discussion, and selection; and our sponsoring institutions, Universidade Nova de Lisboa, Fundação para a Ciência e Tecnologia, FBA, and AgentLink III.

April 2005

João Leite
Paolo Torroni

Organization

Workshop Chairs

João Leite, New University of Lisbon, Portugal
Paolo Torroni, University of Bologna, Italy

Program Committee

José Alferes, New University of Lisbon, Portugal
Gerd Brewka, University of Leipzig, Germany
Jürgen Dix, Technical University of Clausthal, Germany
Klaus Fisher, DFKI, Germany
Michael Fisher, The University of Liverpool, UK
James Harland, Royal Melbourne Institute of Technology, Australia
Katsumi Inoue, National Institute of Informatics, Japan
Sverker Janson, Swedish Institute of Computer Science, Sweden
João Leite, New University of Lisbon, Portugal
Yves Lespérance, York University, Canada
John-Jules Ch. Meyer, Utrecht University, The Netherlands
Leora Morgenstern, IBM, USA
Wojciech Penczek, Polish Academy of Sciences, Poland
Jeremy Pitt, Imperial College London, UK
Enrico Pontelli, New Mexico State University, USA
Fariba Sadri, Imperial College London, UK
Ken Satoh, National Institute of Informatics, Japan
Renate Schmidt, The University of Manchester, UK
Tran Cao Son, New Mexico State University, USA
Francesca Toni, University of Pisa, Italy
Wiebe van der Hoek, The University of Liverpool, UK
Paolo Torroni, University of Bologna, Italy
Makoto Yokoo, Kyushu University, Japan
Cees Witteveen, Delft University of Technology, The Netherlands

Additional Reviewers

Federico Banti
Thomas Eiter
Ulle Endriss
Ullrich Hustadt
Magdalena Kacprzak
Olle Olsson

Inna Pivkina
Chiaki Sakama
Kostas Stathis
Maciej Szreter
Gregory Wheeler
Yingqiang Zhang

Secretariat

Filipa Mira Reis
Sílvia Marina Costa

Local Organization

António Albuquerque
Duarte Alvim
Eduardo Barros
Jamshid Ashtari
Joana Lopes
Miguel Maurício
Miguel Morais
Sérgio Lopes

Steering Committee

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

Sponsoring Institutions

Table of Contents

Foundations

Architectures

Interaction

Planning and Applications

A Logic for Knowledge, Correctness, and Real Time*

Bożena Woźna and Alessio Lo uscio

Department of Computer Science,
University College London,
Gower Street, London WC1E 6BT,
United Kingdom
{B.Wozna, A.Lomuscio}@cs.ucl.ac.uk

Abstract. We present TCTLKD, a logic for knowledge, correctness and real time. TCTLKD is interpreted on real time deontic interpreted systems, and extension to continuous time of deontic interpreted systems. We exemplify the use of TCTLKD by discussing a variant of the "railroad crossing system".

1 Introduction

Logic has a long tradition in the area of for al theories for ulti-agent syste s (MAS). Its role is to provide a precise and una biguous specification language to describe, reason about, and predict the behaviour of a syste .

While in the early 80's existing logical for alis s fro other areas such as philosophical logic, concurrency theory, etc., were i ported with little of no odification to the area of MAS, fro the late 80's onwards specific for alis s have been designed, studied, and tailored to the needs of MAS. Of particular note is the case of episte ic logic, or the logic of knowledge.

Focus on episte ic logics in MAS began with the use of the odal logic syste $S5$ developed independently by Hintikka [1] and Au ann [2] in for al logic and econo ics respectively. This starting point for ed the core basis of a nu ber of studies that appeared in the past 20 years, including for alisations of group knowledge [3, 4, 5], co binations of episte ic logic with ti e [6, 7, 8], auto-episte ic logics [9, 10], episte ic updates [11, 12], broadcast syste s and hypercubes [13, 14], etc. Episte ic logic is no longer a re arkable special case of a nor al odal syste , but has now beco e an area of study on its own with regular the atic workshops and conferences.

In particular, co binations of episte ic and te poral logics allow us to reason about the te poral evolution of episte ic states, knowledge of a changing world, etc. Traditionally, this is achieved by co bining a te poral logic for discrete linear ti e [15, 16, 17] with the logic $S5$ for knowledge [18]. Various classes

* The authors acknowledge support from the EPSRC (grant GR/S49353), and the Nuffield Foundation (grant NAL/690/G).

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 1–15, 2005.

of MAS (synchronous, asynchronous, perfect recall, no learning, etc.) can be identified in this framework, and axiomatisations have been provided [19, 20]. More recently, combinations of branching time logic CTL [21, 22, 23] with the epistemic logic $S5$ have been studied, and axiomatisation provided [8].

All efforts above have focused on a discrete model of time, either in its linear or branching versions. While this is useful and adequate in most applications, certain classes of scenarios (notably robotics and networking) require a model of time as a continuous flows of events.

In the area of timed-systems the modal logic TCTL has been suggested as an adequate formalism to model real time. In this paper we propose a logic (which we call TCTLKD) combining the temporal aspects of TCTL with the notions defined by the epistemic logic $S5$, as well as the correctness notion defined in [24]. This combination allows us to reason about the real time evolution of epistemic states, the correct functioning of multi-agent systems with respect to real time, and any combination of these.

Traditionally, the semantics of temporal epistemic logic is defined on variants of interpreted systems to provide an interpretation to the epistemic modalities. These use the notion of *protocol* to provide a basis for the action selection mechanism of the agents. Since we are working on real time, here we shall use the finer grained semantics of timed automata to model the agents' evolution. We then synchronise networks of timed automata to provide a general model of a MAS.

The rest of the paper is organised as follows. In Section 2 we define the concept of interpreted systems on real time by taking the parallel composition of timed automata. In Section 3 we define the logic TCTLKD as an extension to real time of the logic for knowledge and correctness as defined in [24, 25]. In Section 4 we provide a case study analysis to demonstrate its use in applications. We conclude in Section 5 by discussing related and future work on this subject.

2 Interpreted Systems over Real Time

Interpreted systems are traditionally defined as a set of infinite runs on global states [18]. In this model each run is a discrete sentence representing events. At each global state, each agent selects an action according to a (possibly nondeterministic) protocol. In this section we extend (discrete) interpreted systems to real time interpreted systems in two aspects. First, we specify the agents' behaviour by a finer grained semantics: timed automata. Second, by means of parallel composition of timed automata, we define a class of interpreted systems operating on real time.

We begin by recalling the concept of timed automata, as introduced in [26]. Timed automata are extensions of finite state automata with constraints on timing behaviour. The underlying finite state automata are augmented with a set of real time variables.

2.1 Timed Automata

Let $\mathbb{R} = [0, \infty)$ be a set of non-negative real nu bers, $\mathbb{R}_+ = (0, \infty)$ be a set of positive real nu bers, $\mathbb{N} = \{0, 1, 2, \ldots\}$ a set of natural nu bers, and $\mathcal{X}$ a finite set of real variables, called *clocks*. The set of *clock constraints* over $\mathcal{X}$ is defined by the following gra ar:

$$\mathfrak{cc} := true \mid x \sim c \mid \mathfrak{cc} \wedge \mathfrak{cc},$$

where $x \in \mathcal{X}$, $c \in \mathbb{N}$, and $\sim \in \{\leq, <, =, >, \geq\}$. The set of all the clock constraints over $\mathcal{X}$ is denoted by $\mathcal{C}(\mathcal{X})$. A *clock valuation* on $\mathcal{X}$ is a tuple $v \in \mathbb{R}^{|\mathcal{X}|}$. The value of the clock x in v is denoted by $v(x)$. For a valuation v and $\delta \in \mathbb{R}$, $v + \delta$ denotes the valuation v' such that for all $x \in \mathcal{X}$, $v'(x) = v(x) + \delta$. Moreover, let $\mathcal{X}^*$ be the set $\mathcal{X} \cup \{x_0\}$, where x_0 is a clock whose value is always 0, that is, its value does not increase with ti e as the values of the other clocks. Then, an *assignment* $\mathfrak{as}$ is a function fro $\mathcal{X}$ to $\mathcal{X}^*$, and the set of all the assign ents over $\mathcal{X}$ is denoted by $\mathfrak{A}(\mathcal{X})$. By $v[\mathfrak{as}]$ we denote the valuation v' such that for all $x \in \mathcal{X}$, if $\mathfrak{as}(x) \in \mathcal{X}$, then $v'(x) = v(\mathfrak{as}(x))$, otherwise $v'(x) = 0$.

Let $v \in \mathbb{R}^{|\mathcal{X}|}$, the satisfaction relation $\models$ for a clock constraint $\mathfrak{cc} \in \mathcal{C}(\mathcal{X})$ is defined inductively as follows:

$v \models true$,
$v \models (x \sim c)$ iff $v(x) \sim c$,
$v \models (\mathfrak{cc} \wedge \mathfrak{cc}')$ iff $v \models \mathfrak{cc}$ and $v \models \mathfrak{cc}'$.

For a constraint $\mathfrak{cc} \in \mathcal{C}(\mathcal{X})$, by $[\![\mathfrak{cc}]\!]$ we denote the set of all the clock valuations satisfying $\mathfrak{cc}$, i.e., $[\![\mathfrak{cc}]\!] = \{v \in \mathbb{R}^{|\mathcal{X}|} \mid v \models \mathfrak{cc}\}$.

Definition 1 (Timed Automaton). *A* ti ed auto aton *is a tuple* $\mathcal{TA} = (\mathfrak{Z}, L, l^0, \mathcal{X}, E, \mathfrak{I})$, *where*

- $\mathfrak{Z}$ *is a finite set of actions,*
- L *is a finite set of locations,*
- $l^0 \in L$ *is an initial location,*
- $\mathcal{X}$ *is a finite set of clocks,*
- $E \subseteq L \times \mathfrak{Z} \times \mathcal{C}(\mathcal{X}) \times \mathfrak{A}(\mathcal{X}) \times L$ *is a transition relation,*
- $\mathfrak{I} : L \to \mathcal{C}(\mathcal{X})$ *is a function, called a* location invariant, *which assigns to each location* $l \in L$ *a clock constraint defining the conditions under which* $\mathcal{TA}$ *can stay in* l.

Each element e *of* E *is denoted by* $l \xrightarrow{a,\mathfrak{cc},\mathfrak{as}} l'$, *where* l *is a source location,* l' *is a target location,* a *is an action,* $\mathfrak{cc}$ *is the enabling condition for* e, *and* $\mathfrak{as}$ *is the assignment for* e.

Note that we deal with "diagonal-free" auto ata. This is because ulti ately we would like to verify MAS specified in this for alis , and the odel checking ethods for real ti e syste s (based on the Difference Bound Matrices [27], variants of Boolean Decision Diagra s [28, 29], or SAT ethods [30, 31, 32]) are proble atic when the co ponents of the syste s are odelled by "diagonal auto ata".

In order to reason about syste s represented by ti ed auto ata, for a set of propositional variables $\mathcal{PV}$, we define a valuation function $V_{\mathcal{TA}} : L \rightarrow 2^{\mathcal{PV}}$, which assigns propositions to the locations.

Definition 2 (Dense State Space). *The* dense state space *of a timed automaton* $\mathcal{TA} = (\mathfrak{Z}, L, l^0, \mathcal{X}, E, \mathfrak{I})$ *is a structure* $\mathrm{D}(\mathcal{TA}) = (Q, q^0, \rightarrow)$, *where*

- $Q = L \times \mathbb{R}^{|\mathcal{X}|}$ *is the set of all the instantaneous states,*
- $q^0 = (l^0, v^0)$ *with* $v^0(x) = 0$ *for all* $x \in \mathcal{X}$, *is the initial state,*
- $\rightarrow \subseteq Q \times (\mathfrak{Z} \cup \mathbb{R}) \times Q$ *is the transition relation, defined by action- and time-successors as follows:*
  - *for* $a \in \mathfrak{Z}$, $(l, v) \xrightarrow{a} (l', v')$ *iff* $(\exists \mathfrak{cc} \in \mathcal{C}(\mathcal{X}))(\exists \mathfrak{as} \in \mathfrak{A}(\mathcal{X}))$ *such that* $l \xrightarrow{a, \mathfrak{cc}, \mathfrak{as}} l' \in E$, $v \in [\![\mathfrak{cc}]\!]$,$v' = v[\mathfrak{as}]$ *and* $v' \in [\![\mathfrak{I}(l')]\!]$ (action successor),
  - *for* $\delta \in \mathbb{R}$, $(l, v) \xrightarrow{\delta} (l, v + \delta)$ *iff* $v + \delta \in [\![\mathfrak{I}(l)]\!]$ (ti e successor).

For $(l, v) \in Q$, let $(l, v) + \delta$ denote $(l, v + \delta)$. A *q-run* ρ of a $\mathcal{TA}$ is a sequence of instantaneous states: $q_0 \xrightarrow{\delta_0} q_0 + \delta_0 \xrightarrow{a_0} q_1 \xrightarrow{\delta_1} q_1 + \delta_1 \xrightarrow{a_1} q_2 \xrightarrow{\delta_2} \ldots$, where $q_0 = q \in Q$, $a_i \in \mathfrak{Z}$, and $\delta_i \in \mathbb{R}_+$ for each $i \in \mathbb{N}$. A run ρ is said to be *progressive* iff $\Sigma_{i \in \mathbb{N}} \delta_i$ is unbounded. A $\mathcal{TA}$ is *progressive* if all its runs are progressive. For si plicity of presentation, we consider only progressive ti ed auto ata. Note that progressiveness can be checked as in [33].

2.2 Parallel Composition of Timed Automata

In general, we will odel a ulti-agent syste by taking several ti ed auto ata running in parallel and co unicating with each other. These concurrent ti ed auto ata can be co posed into a global ti ed auto aton as follows: the transitions of the ti ed auto ata that do not correspond to a shared action are interleaved, whereas the transitions labelled with a shared action are synchronised.

There are any different definitions of parallel co position. We use a *multi-way synchronisation*, requiring that each co ponent that contains a co unication transition (labelled by a shared action) has to perfor this action.

Let $\mathcal{TA}_i = (\mathfrak{Z}_i, L_i, l_i^0, E_i, \mathcal{X}_i, \mathfrak{I}_i)$ be a ti ed auto aton, for $i = 1, \ldots, m$. To define a parallel co position of m ti ed auto ata, we assu e that $L_i \cap L_j = \emptyset$ for all $i, j \in \{1, \ldots, m\}$, and $i \neq j$. Moreover, by $\mathfrak{Z}(a) = \{1 \leq i \leq m \mid a \in \mathfrak{Z}_i\}$ we denote a set of nu bers of the ti ed auto ata containing an action a.

Definition 3 (Parallel Composition). The parallel co position *of* m *timed automata* $\mathcal{TA}_i$ *is a timed automaton* $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$, *where* $\mathfrak{Z} = \bigcup_{i=1}^{m} \mathfrak{Z}_i$, $L = \prod_{i=1}^{m} L_i$, $l^0 = (l_1^0, \ldots, l_m^0)$, $\mathcal{X} = \bigcup_{i=1}^{m} \mathcal{X}_i$, $\mathfrak{I}(l_1, \ldots, l_m) = \bigwedge_{i=1}^{m} \mathfrak{I}_i(l_i)$, *and a transition* $((l_1, \ldots, l_m), a, \mathfrak{cc}, \mathfrak{as}, (l_1', \ldots, l_m')) \in E$ *iff* $(\forall i \in \mathfrak{Z}(a))\ (l_i, a, \mathfrak{cc}_i, \mathfrak{as}_i, l_i') \in E_i$, $\mathfrak{cc} = \bigwedge_{i \in \mathfrak{Z}(a)} \mathfrak{cc}_i$, $\mathfrak{as} = \bigcup_{i \in \mathfrak{Z}(a)} \mathfrak{as}_i$, *and* $(\forall j \in \{1, \ldots, m\} \setminus \mathfrak{Z}(a))\ l_j' = l_j$.

Note that in the above any auto aton is allowed to set a value of any clock, including the ones associated with other agents.

Let PV_i be a set of propositional variables containing the sy bol **true**, $V_{\mathcal{TA}_i} : L_i \rightarrow 2^{\mathcal{PV}_i}$ be a valuation function for the ith auto aton, where $i \in \{1, \ldots, m\}$, and $\mathcal{PV} = \bigcup_{i=1}^{m} \mathcal{PV}_i$. Then, the valuation function $V_{\mathcal{TA}} : L \rightarrow 2^{\mathcal{PV}}$ for the parallel co position of m ti ed auto ata, is defined as follows $V_{\mathcal{TA}}((l_1, \ldots, l_m)) = \bigcup_{i=1}^{m} V_{\mathcal{TA}_i}(l_i)$.

2.3 Real Time Deontic Interpreted System

In line with uch of the ulti-agent syste s literature, we use interpreted syste s as a se antics for a te poral episte ic language. For this, we need to adapt the to work on real ti e: this is why we take ti ed auto ata as the underlying odelling concept (as opposed to the standard protocols of interpreted syste s). To define *real time deontic interpreted systems*, we first partition the set of clock valuations as in [34].

Let $\mathcal{TA}$ be a ti ed auto aton, $\mathcal{C}(\mathcal{TA}) \subseteq \mathcal{C}(\mathcal{X})$ be a non-e pty set containing all the clock constrains occurring in any enabling condition used in the transition relation E or in a state invariant of $\mathcal{TA}$. Moreover, let c_{max} be the largest constant appearing in $\mathcal{C}(\mathcal{TA})$. For $\sigma \in \mathbb{R}$, $frac(\sigma)$ denotes the fractional part of σ, and $\lfloor \sigma \rfloor$ denotes its integral part.

Definition 4 (Equivalence of Clock Valuations). *For two clock valuations v and v' in $\mathbb{R}^{|\mathcal{X}|}$,we say that $v \simeq v'$ iff for all $x, y \in \mathcal{X}$ the following conditions are met:*

1. $v(x) > c_{max}$ *iff* $v'(x) > c_{max}$
2. *if* $v(x) \leq c_{max}$ *and* $v(y) \leq c_{max}$ *then*
   a.) $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$,
   b.) $frac(v(x)) = 0$ *iff* $frac(v'(x)) = 0$, *and*
   c.) $frac(v(x)) \leq frac(v(y))$ *iff* $frac(v'(x)) \leq frac(v'(y))$.

The equivalence classes of the relation $\simeq$ are called *zones*, and denoted by Z, Z' and so on.

Now we are ready to define a *Real Time Deontic Interpreted System* that will be se antics for the logic presented in the next section.

Let $\mathcal{AG}$ be a set of m agents, where each agent is odelled by a ti ed auto aton $\mathcal{TA}_i = (\mathfrak{Z}_i, L_i, l_i^0, E_i, \mathcal{X}_i, \mathfrak{I}_i)$, for $i \in \{1, \ldots, m\}$. Moreover, assu e, in line with [24, 25], that for every agent, its set L_i of local locations is partitioned into "allowed" locations, denoted by $\mathcal{G}_i$, and "disallowed" locations, denoted by $\mathcal{R}_i$ and defined by $\mathcal{R}_i = L_i \setminus \mathcal{G}_i$. We shall call these locations *green* and *red* respectively. Further, assu e that the parallel co position $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$ of all the agents is given[1], and that $l_i : Q \rightarrow L_i$ is a function that returns the location of agent i fro a global state. Then, a *real time deontic interpreted system* is defined as follows.

Definition 5 (Real Time Deontic Interpreted System). *A* real ti e deontic interpreted syste *is a tuple $M_c = (Q, q^0, \rightarrow, \sim_1^K, \ldots, \sim_m^K, R_1^O, \ldots, R_m^O, \mathcal{V}_c)$, where:*

---

[1] Note that the set L, which defines all the possible *global locations*, is defined as the Cartesian product $L_1 \times \ldots \times L_m$, such that $L_1 \supseteq \mathcal{G}_1, \ldots, L_m \supseteq \mathcal{G}_m$.

– *Q, q^0, and $\rightarrow$ are defined as in Definition 2,*
– *$\sim_i^K \subseteq Q \times Q$ is a relation defined by: $(l, v) \sim_i^K (l', v')$ iff $l_i((l, v)) = l_i((l', v'))$ and $v \simeq v'$, for each agent i. Obviously $\sim_i^K$ is an equivalence relation.*
– *$R_i^O \subseteq Q \times Q$ is a relation defined by: $(l, v)R_i^O(l', v')$ iff $l_i((l', v')) \in \mathcal{G}_i$, for each agent i.*
– *$V_c : Q \rightarrow 2^{\mathcal{PV}}$ is a valuation function that extends $V_{\mathcal{TA}}$ as follows $V_c((l, v)) = V_{\mathcal{TA}}(l)$, i.e., V_c assigns the same propositions to the states with the same locations.*

3 The Logic TCTLKD

In this section, we for ally present the syntax and se antics of a *real time computation tree logic for knowledge and correctness* (TCTLKD), which extends the standard TCTL [34], the logic for real ti e, by eans of odal operators for knowledge and correctness.

The language generalises classical propositional logic, and thus it contains the standard propositional connectives $\neg$ (not) and $\vee$ (or); the re aining connectives ($\wedge$ (and), $\rightarrow$ (i plies), $\leftrightarrow$ (if, and only if) are assu ed to be introduced as abbreviations in the usual way. With respect to real ti e te poral connectives, we take as pri itives U_I (for "until within interval I"), and G_I (for "always within interval I"); the re aining operators, F_I (for "eventually within interval I") and R_I (for "release within interval I"), are assu ed to be introduced as abbreviations in the usual way. The language also contains two path quantifiers: A (for "for all the runs") and E (for "there exists a run"). Further, we assu e a set $\mathcal{AG} = \{1, \ldots, m\}$ of agents, and we use the indexed odalities K_i, $\mathcal{O}_i$, and $\hat{\mathrm{K}}_i^j$ to represent the knowledge of agent i, the correct functioning circu stances of agent i, and the knowledge of agent i under assu ption of correct functioning of agent j, respectively. Further ore, we use the indexed odalities D_Γ, C_Γ to represent distributed and co on knowledge in a group of agents $\Gamma \subseteq \mathcal{AG}$, and we use the operator E_Γ to represent the concept "everybody in Γ knows".

3.1 Syntax of TCTLKD

We assu e a set $\mathcal{PV}$ of propositional variables, and a finite set $\mathcal{AG}$ of m agents. Further ore, let I be an interval in $\mathbb{R}$ with integer bounds of the for $[n, n']$, $[n, n')$, $(n, n']$, (n, n'), (n, ∞), and $[n, \infty)$, for $n, n' \in \mathbb{N}$. The set of TCTLKD for ulas is defined inductively as follows:

- every e ber p of $\mathcal{PV}$ is a for ula,
- if α and β are for ulas, then so are $\neg\alpha$, $\alpha \vee \beta$, $\mathrm{EG}_I\alpha$, and $\mathrm{E}(\alpha \mathrm{U}_I \beta)$,
- if α is for ula, then so are $\mathrm{K}_i\alpha$, $\hat{\mathrm{K}}_i^j\alpha$, and $\mathcal{O}_i\alpha$, for $i, j \in \mathcal{AG}$,
- if α is for ula, then so are $\mathrm{D}_\Gamma\alpha$, $\mathrm{C}_\Gamma\alpha$, and $\mathrm{E}_\Gamma\alpha$, for $\Gamma \subseteq \mathcal{AG}$.

The other basic te poral, episte ic, and correctness odalities are defined as follows:

- $\mathrm{EF}_I\varphi \stackrel{def}{=} \mathrm{E}(\mathbf{true}\mathrm{U}_I\varphi)$, $\mathrm{AF}_I\varphi \stackrel{def}{=} \neg\mathrm{EG}_I(\neg\varphi)$, $\mathrm{AG}_I\varphi \stackrel{def}{=} \neg\mathrm{EF}_I(\neg\varphi)$,
- $\mathrm{A}(\alpha\mathrm{U}_I\beta) \stackrel{def}{=} \neg\mathrm{E}(\neg\beta\mathrm{U}_I(\neg\beta \wedge \neg\alpha)) \wedge \neg\mathrm{EG}_I(\neg\beta)$,
- $\mathrm{A}(\alpha\mathrm{R}_I\beta) \stackrel{def}{=} \neg\mathrm{E}(\neg\alpha\mathrm{U}_I\neg\beta)$, $\mathrm{E}(\alpha\mathrm{R}_I\beta) \stackrel{def}{=} \neg\mathrm{A}(\neg\alpha\mathrm{U}_I\neg\beta)$,
- $\overline{\mathrm{K}}_i\alpha \stackrel{def}{=} \neg\mathrm{K}_i\neg\alpha$, $\overline{\mathcal{O}}_i\alpha \stackrel{def}{=} \neg\mathcal{O}_i\neg\alpha$, $\overline{\hat{\mathrm{K}}^{\mathrm{j}}_{\mathrm{i}}}\alpha \stackrel{def}{=} \neg\hat{\mathrm{K}}^j_i\neg\alpha$,
- $\overline{\mathrm{D}}_\Gamma\alpha \stackrel{def}{=} \neg\mathrm{D}_\Gamma\neg\alpha$, $\overline{\mathrm{C}}_\Gamma\alpha \stackrel{def}{=} \neg\mathrm{C}_\Gamma\neg\alpha$, $\overline{\mathrm{E}}_\Gamma\alpha \stackrel{def}{=} \neg\mathrm{E}_\Gamma\neg\alpha$.

3.2 Semantics of TCTLKD

Let $\mathcal{AG}$ be a set of m agents, where each agent is odelled by a ti ed auto aton $\mathcal{TA}_i = (\mathfrak{Z}_i, L_i, l^0_i, E_i, \mathcal{X}_i, \mathfrak{I}_i)$, for $i = \{1, \dots, m\}$, $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$ be their parallel co position, and $M_c = (Q, q^0, \rightarrow, \sim^K_1, \dots, \sim^K_m, R^O_1, \dots, R^O_m, \mathcal{V}_c)$ be a *real time deontic interpreted system*. Moreover, let $\rho = q_0 \stackrel{\delta_0}{\rightarrow} q_0 + \delta_0 \stackrel{a_0}{\rightarrow} q_1 \stackrel{\delta_1}{\rightarrow} q_1 + \delta_1 \stackrel{a_1}{\rightarrow} q_2 \stackrel{\delta_2}{\rightarrow} \dots$ be a run of $\mathcal{TA}$ such that $\delta_i \in \mathbb{R}_+$ for $i \in \mathbb{N}$, and let $f_{\mathcal{TA}}(q)$ denote the set of all such q-runs of $\mathcal{TA}$. In order to give a se antics to TCTLKD, we introduce the notation of a *dense path* π_ρ *corresponding to run* ρ. *A dense path* π_ρ *corresponding to* ρ is a apping fro $\mathbb{R}$ to a set of states[2] such that $\pi_\rho(r) = s_i + \delta$ for $r = \Sigma^i_{j=0}\delta_j + \delta$ with $i \geq 0$ and $0 \leq \delta < \delta_i$. Moreover, as usual, we define the following episte ic relations: $\sim^E_\Gamma = \bigcup_{i\in\Gamma} \sim_i$, $\sim^C_\Gamma = (\sim^E_\Gamma)^+$ (the transitive closure of $\sim^E_\Gamma$), and $\sim^D_\Gamma = \bigcap_{i\in\Gamma} \sim_i$, where $\Gamma \subseteq \mathcal{AG}$.

Definition 6 (Satisfaction of TCTLKD**).** *Let* $M_c, q \models \alpha$ *denote that* α *is true at state* s *in the model* M_c*.* M_c *is omitted, if it is implicitly understood. The relation* $\models$ *is defined inductively as follows:*

$q_0 \models p$ *iff* $p \in \mathcal{V}_c(q_0)$,
$q_0 \models \neg\varphi$ *iff* $q_0 \not\models \varphi$,
$q_0 \models \varphi \vee \psi$ *iff* $q_0 \models \varphi$ *or* $q_0 \models \psi$,
$q_0 \models \varphi \wedge \psi$ *iff* $q_0 \models \varphi$ *and* $q_0 \models \psi$,
$q_0 \models \mathrm{E}(\varphi\mathrm{U}_I\psi)$ *iff* $(\exists\, \rho \in f_{\mathcal{TA}}(q_0))(\exists r \in I)\big[\pi_\rho(r) \models \psi$ *and* $(\forall r' < r)\ \pi_\rho(r') \models \varphi\big]$,
$q_0 \models \mathrm{EG}_I\varphi)$ *iff* $(\exists\, \rho \in f_{\mathcal{TA}}(q_0))(\forall r \in I)\ \pi_\rho(r) \models \varphi$,
$q_0 \models \mathrm{K}_i\alpha$ *iff* $(\forall q' \in Q)((q_0 \sim^K_i q')$ *implies* $q' \models \alpha)$,
$q_0 \models \mathcal{O}_i\alpha$ *iff* $(\forall q' \in Q)(q_0 R^O_i q')$ *implies* $q' \models \alpha)$,
$q_0 \models \hat{\mathrm{K}}^j_i\alpha$ *iff* $(\forall q' \in Q)((q_0 \sim^K_i q'$ *and* $q_0 R^O_j q')$ *implies* $q' \models \alpha)$,
$q_0 \models \mathrm{D}_\Gamma\alpha$ *iff* $(\forall q' \in Q)((q_0 \sim^D_\Gamma q')$ *implies* $q' \models \alpha)$,
$q_0 \models \mathrm{E}_\Gamma\alpha$ *iff* $(\forall q' \in Q)((q_0 \sim^E_\Gamma q')$ *implies* $q' \models \alpha)$,
$q_0 \models \mathrm{C}_\Gamma\alpha$ *iff* $(\forall q' \in Q)((q_0 \sim^C_\Gamma q')$ *implies* $q' \models \alpha)$.

Intuitively, the for ula $\mathrm{E}(\alpha\mathrm{U}_I\beta)$ holds at a state q_0 in a real ti e deontic interpreted syste M_c if there exists a run starting at q_0 such that β holds in so e state in ti e interval I, and until then α always holds. The for ula $\mathrm{EG}_I\alpha$ holds at a state q_0 in a real ti e deontic interpreted syste M_c if there exists a

[2] This can be done because of the assumption that $\delta_i > 0$, i.e., $\delta_i \in \mathbb{R}_+$.

run starting at q_0 such that α holds in all the states on the run in time interval I. The formula $\mathrm{K}_i\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if α holds at all the states that are indistinguishable for agent i from q_0. The formula $\mathcal{O}_i\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if α holds at all the states where agent i is functioning correctly. The formula $\hat{\mathrm{K}}_i^j\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if α holds at all the states that agent i is unable to distinguish from the actual state q_0, and in which agent j is functioning correctly. The formula $\mathrm{E}_\Gamma\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if α is true in all the states that the group Γ of agents is unable to distinguish from the actual state q_0. Note that $\mathrm{E}_\Gamma\alpha$ can be defined by $\bigwedge_{i\in\Gamma}\mathrm{K}_i\alpha$. The formula $\mathrm{C}_\Gamma\alpha$ is equivalent to the infinite conjunction of the formulas $\mathrm{E}_\Gamma^k\alpha$ for $k \geq 1$. So, $C_\Gamma\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if everyone knows α holds at q_0, everyone knows that everyone knows α holds at q_0, etc. The formula $\mathrm{D}_\Gamma\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if the "combined" knowledge of all the agents in Γ implies α. We refer to [34, 18, 24] for more details on the operators above.

A TCTLKD formula φ is *satisfiable* if there exists a real time deontic interpreted system $M_c = (Q, q^0, \rightarrow, \sim_1^K, \ldots, \sim_m^K, R_1^O, \ldots, R_m^O, \mathcal{V}_c)$ and a state q of M_c, such that $M_c, q \models \varphi$. A TCTLKD formula φ is *valid in* M_c (denoted $M_c \models \varphi$) if $M_c, q^0 \models \varphi$, i.e., φ is true at the initial state of the model M_c.

Note that the "full" logic of real time (TCTL) is undecidable [34]. Since real time deontic interpreted systems can be shown to be as expressive as the TCTL-structure of a time graph [34], and the fusion [35] between TCTL, $S5$ for knowledge [18], and $KD45^{i-j}$ for the deontic dimension [24] is a proper extension of TCTL, it follows that problem of satisfiability for the TCTLKD logic will be also undecidable. Still, it is easy to observe that given a TCTLKD formula φ and a real time deontic interpreted system M_c, the problem of deciding whether $M_c \models \varphi$ is decidable. This result is our motivation for introducing TCTLKD. We are not interested in using the whole class of real time deontic interpreted systems, but only to study particular examples by means of this logic. We exemplify this in the next section.

4 Applications

One of the motivations for developing the formalism presented in this paper is that we would like to be able to analyse what epistemic and temporal properties hold, when agents follow or violate their specifications while operating on real time.

As an example of this we discuss the *Railroad Crossing System* (RCS) [36], a well-known example in the literature of real-time verification. Here we analyse the scenario not only by means of temporal operators but also by means of epistemic and correctness modalities. The system consists of three agents: Train, Gate, and Controller running in parallel and synchronising through the events: "*approach*", "*exit*", "*lower*", and "*raise*".

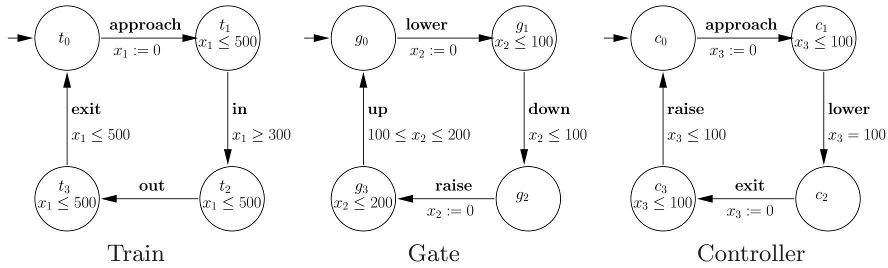


Fig. 1. Agents Train, Gate, and Controller for the correct RCS system

Let us start by considering what we call the *correct RCS*, as modelled by timed automata (Figure 1). The correct RCS operates as follows. When Train approaches the crossing, it sends an *approach* signal to Controller, and enters the crossing between 300 and 500 seconds from this event. When Train leaves the crossing, it sends an *exit* signal to Controller. Controller sends a signal *lower* to Gate exactly 100 seconds after the *approach* signal is received, and sends a *raise* signal within 100 seconds after *exit*. Gate performs the transition *down* within 100 seconds of receiving the request *lower*, and responds to *raise* by moving *up* between 100 and 200 seconds.

Assume the following set of propositional variables: $\mathcal{PV} = \{\mathfrak{p}, \mathfrak{q}, \mathfrak{r}, \mathfrak{s}\}$ with $\mathcal{PV}_{Train} = \{\mathfrak{p}, \mathfrak{q}\}$, $\mathcal{PV}_{Gate} = \{\mathfrak{r}\}$, and $\mathcal{PV}_{Cont} = \{\mathfrak{s}\}$. The proposition $\mathfrak{p}$ represents the fact that an approach signal was sent by Train, $\mathfrak{q}$ that Train is on the cross, $\mathfrak{r}$ that Gate is down, and $\mathfrak{s}$ that Controller sent the signal *lower* to Gate. A real time deontic interpreted system M_{RCS} can be associated with the correct RCS as follows. For the sets $L_1 = \{t_0, t_1, t_2, t_3\}$, $L_2 = \{g_0, g_1, g_2, g_3\}$, and $L_3 = \{c_0, c_1, c_2, c_3\}$ of locations for Train, Gate, and Controller respectively, the set of "green" locations and the dense state space for RCS are defined by $G_1 = L_1$, $G_2 = L_2$, $G_3 = L_3$, and $Q = L_1 \times L_2 \times L_3 \times \mathbb{R}^3$, respectively. The valuation functions for Train ($V_{Train} : L_1 \rightarrow 2^{\mathcal{PV}_{Train}}$), Gate ($V_{Gate} : L_2 \rightarrow 2^{\mathcal{PV}_{Gate}}$), and Controller ($V_{Cont} : L_3 \rightarrow 2^{\mathcal{PV}_{Cont}}$) are defined as follows:

- $V_{Train}(t_1) = \{\mathfrak{p}\}$, $V_{Train}(t_2) = \{\mathfrak{q}\}$, and $V_{Train}(t_0) = V_{Train}(t_3) = \emptyset$.
- $V_{Gate}(g_2) = \{\mathfrak{r}\}$, and $V_{Gate}(g_0) = V_{Gate}(g_1) = V_{Gate}(g_3) = \emptyset$.
- $V_{Cont}(c_2) = \{\mathfrak{s}\}$, and $V_{Cont}(c_0) = V_{Cont}(c_1) = V_{Cont}(c_3) = \emptyset$.

The valuation function $V_{RCS} : L_1 \times L_2 \times L_3 \rightarrow 2^{\mathcal{PV}}$, for the RCS system, is built as follows: $V_{RCS}(l) = V_{Train}(l_1) \cup V_{Gate}(l_2) \cup V_{Cont}(l_3)$, for all $l = (l_1, l_2, l_3) \in L_1 \times L_2 \times L_3$. Thus, according to the definition of the real time deontic interpreted system, the valuation function $V_{M_{RCS}} : L_1 \times L_2 \times L_3 \times \mathbb{R}^3 \rightarrow 2^{\mathcal{PV}}$ of M_{RCS} is defined by $V_{M_{RCS}}(l, v) = V_{RCS}(l)$.

Using the TCTLKD logic, we can specify properties of the correct RCS system that cannot be specified by standard propositional temporal epistemic logic. For example, we consider the following:

$$\mathrm{AG}_{[0,\infty]}(\mathfrak{p} \rightarrow \mathrm{K}_{Controller}(\mathrm{AF}_{[300,\infty]}\mathfrak{q})) \tag{1}$$

$$\mathrm{AG}_{[0,\infty]}\mathrm{K}_{Train}(\mathfrak{p} \rightarrow \mathrm{AF}_{[0,200]}\mathfrak{r}) \tag{2}$$

$$\mathrm{K}_{Controller}(\mathfrak{s} \rightarrow \mathrm{AF}_{[0,100]}\mathfrak{r}) \tag{3}$$

For ula (1) states that forever in the future if an approach signal is sent by agent Train, then agent Controller knows that in so e point after 300 seconds later Train will enter the cross. For ula (2) states that forever in the future agent Train knows that, if it sends an approach signal, then agent Gate will send the signal down within 200 seconds. For ula (3) states that agent Controller knows that if it sends an lower signal, then agent Gate will send the signal down within 100 seconds.

All the for ulas above can be shown to hold on M_{RCS} on the initial state. We can also check that the following properties do not hold on M_{RCS}.

$$\mathrm{AG}_{[0,\infty]}(\mathfrak{p} \rightarrow \mathrm{K}_{Controller}(\mathrm{AF}_{[0,300]}\mathfrak{q})) \tag{4}$$

$$\mathrm{K}_{Train}(\mathrm{AG}_{[0,\infty)}\mathrm{EF}_{[10,90]}\mathfrak{s}) \tag{5}$$

$$\mathrm{K}_{Controller}(\mathfrak{s} \rightarrow \mathrm{AF}_{[0,50]}\mathfrak{r}) \tag{6}$$

For ula (4) states that forever in the future if an approach signal is sent by agent Train, then agent Controller knows that at so e point in the future within 300 seconds Train will enter the crossing. For ula (5) states that agent Train knows that always in the future it is possible that within interval $[10, 90]$ the gate will be down. For ula (6) states that agent Controller knows that if it sends the lower signal, then agent Gate will send the signal down within 50 seconds.

Let us now consider a variant of the RCS syste described above, and let us assu e that agent Controller is faulty. Let us assu e that because of a fault the signal *lower* ay not be sent in the specified interval, and the transition to the faulty state $\overline{c_2}$ ay be triggered. We are allowing for Controller to recover fro the fault once in $\overline{c_2}$ by eans of the action *lower* (see Figure 2).

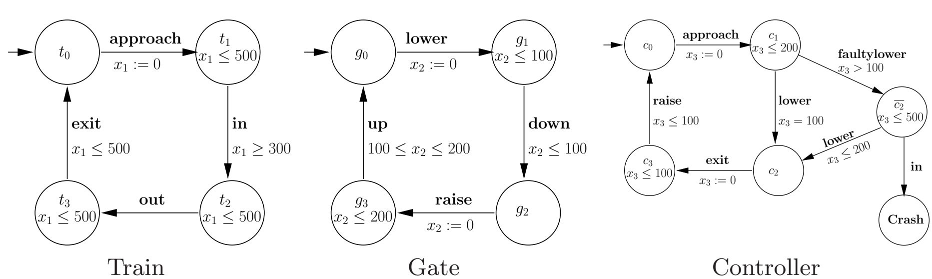


Fig. 2. Agents Train, Gate, and Controller for the faulty RCS system

We exa ine the scenario by considering the following set of propositional variables: $\mathcal{PV} = \{\mathfrak{p}, \mathfrak{q}, \mathfrak{r}, \mathfrak{s}, \mathfrak{crash}\}$ with $\mathcal{PV}_{Train} = \{\mathfrak{p}, \mathfrak{q}\}$, $\mathcal{PV}_{Gate} = \{\mathfrak{r}\}$, and $\mathcal{PV}_{Cont} = \{\mathfrak{s}, \mathfrak{crash}\}$. The propositions $\mathfrak{p}$, $\mathfrak{q}$, $\mathfrak{r}$, and, $\mathfrak{s}$ have the sa e eaning as in the case of the correct RCS syste ; the proposition $\mathfrak{crash}$ represents the fact that Train is on the cross and Gate is still open. A real ti e deontic interpreted syste M_{RCS} can be associated with the faulty RCS syste as follows[3].

For the sets $L_1 = \{t_0, t_1, t_2, t_3\}$, $L_2 = \{g_0, g_1, g_2, g_3\}$, and $L_3 = \{c_0, c_1, c_2, c_3, \overline{c_2}, crash\}$ of locations for Train, Gate, and Controller, the set of "green" locations are defined by $G_1 = L_1$, $G_2 = L_2$, $G_3 = \{c_0, c_1, c_2, c_3\}$, respectively. The dense state space for RCS is defined by $Q = L_1 \times L_2 \times L_3 \times \mathbb{R}^3$. The valuation functions for Train (V_{Train}), Gate (V_{Gate}), and Controller (V_{Cont}) are defined as follows:

- $V_{Train} : L_1 \to 2^{\mathcal{PV}_{Train}}$, and $V_{Train}(t_1) = \{\mathfrak{p}\}$, $V_{Train}(t_2) = \{\mathfrak{q}\}$, and $V_{Train}(t_0) = V_{Train}(t_3) = \emptyset$.
- $V_{Gate} : L_2 \to 2^{\mathcal{PV}_{Gate}}$, and $V_{Gate}(g_0) = V_{Gate}(g_1) = V_{Gate}(g_3) = \emptyset$, and $V_{Gate}(g_2) = \{\mathfrak{r}\}$.
- $V_{Cont} : L_3 \to 2^{\mathcal{PV}_{Cont}}$, and $V_{Cont}(c_0) = V_{Cont}(c_1) = V_{Cont}(c_3) = V_{Cont}(\overline{c_2}) = \emptyset$, $V_{Cont}(c_2) = \{\mathfrak{s}\}$, and $V_{Cont}(crash) = \{\mathfrak{crash}\}$.

The valuation functions $V_{RCS} : L_1 \times L_2 \times L_3 \to 2^{\mathcal{PV}}$, and $V_{M_{RCS}} : L_1 \times L_2 \times L_3 \times \mathbb{R}^3 \to 2^{\mathcal{PV}}$ are defined in the sa e way as in the correct version of the RCS syste .

Using TCTLKD, we can specify the following properties of the faulty RCS syste . These can be checked to hold on the real ti e deontic interpreted syste for the faulty RCS.

$$\mathrm{AG}_{[0,\infty]}\mathrm{K}_{Train}\mathcal{O}_{Controller}(\mathfrak{p} \to \mathrm{AF}_{[0,200]}\mathfrak{r}) \tag{7}$$

$$\mathrm{K}_{Train}\mathcal{O}_{Controller}(\mathfrak{p} \to \mathrm{AF}_{[0,200]}\mathfrak{r}) \tag{8}$$

$$\hat{\mathrm{K}}^{Controller}_{Train}(\mathfrak{p} \to \mathrm{AF}_{[0,200]}\mathfrak{r}) \tag{9}$$

$$\mathrm{AG}_{[0,\infty]}\mathrm{K}_{Train}\mathcal{O}_{Controller}(\neg\mathfrak{crash}) \tag{10}$$

$$\mathrm{AG}_{[0,\infty]}\hat{\mathrm{K}}^{Controller}_{Train}(\neg\mathfrak{crash}) \tag{11}$$

$$\mathrm{AG}_{[0,\infty]}\hat{\mathrm{K}}^{Controller}_{Train}(\mathfrak{p} \to \mathrm{AF}_{[0,100]}\mathfrak{s}) \tag{12}$$

For ula (7) states that forever in the future agent Train knows that whenever agent Controller is functioning correctly, if Train sends the *approach* signal, then agent Gate will send the signal down within 200 seconds. For ula (8) states that agent Train knows that whenever agent Controller is functioning correctly, if the *approach* signal was sent by Train, then at so e point in the future, within 200 second, Gate will be down. For ula (9) states that agent Train knows that under the assu ption of agent Controller functioning correctly, if the *approach* signal was sent by Train, then at so e point in the future, within 200 second, Gate will be down. For ula (10) states that always in the future agent Train knows that whenever agent Controller is functioning correctly under no circu stances

[3] Note that the names of the mathematical objects we use to represent the faulty RCS are the same as the ones employed previously for the correct RCS. Given that these appear in different contexts we trust no confusion arises.

there will be a situation in which Train is on the crossing and Gate is open. For ula (11) states that always in the future agent Train knows that under the assu ption of agent Controller functioning correctly, under no circu stances there will be a situation in which Train is on the crossing and Gate is open. For ula (12) states that always in the future agent Train knows that under the assu ption of agent Controller functioning correctly, if the *approach* signal was sent by Train, then at so e point in the future, within 100 second, the signal *lower* will be sent by Controller.

The following for ulas can be checked not to hold on the faulty RCS.

$$\mathrm{K}_{Train}(\mathfrak{p} \rightarrow \mathrm{AF}_{[0,200]}\mathfrak{r}) \tag{13}$$

$$\mathrm{AG}_{[0,\infty]}\mathrm{K}_{Train}(\neg\mathfrak{crash}) \tag{14}$$

$$\mathrm{AG}_{[0,\infty]}\mathrm{K}_{Train}(\mathfrak{p} \rightarrow \mathrm{AF}_{[0,100]}\mathfrak{s}) \tag{15}$$

For ula (13) states that agent Train knows that, if it sends the *approach* signal, then at so e point in the future, within 200 second, Gate will be down. For ula (14) states that always in the future agent Train knows that under no circu stances there will be a situation where Train is on the cross and Gate is open. For ula (15) states that always in the future agent Train knows that, if it sends the *approach* signal, then at so e point in the future, within 100 second, the signal *lower* will be sent by Controller.

5 Conclusions

In the paper we have proposed TCTLKD, a real ti e logic for knowledge and correctness. TCTLKD is a fusion of three well known logics: TCTL for real ti e [34], S5 for knowledge [18], and $\mathrm{KD45}^{i-j}$ for the correctness di ension [24].

Previous atte pts of co binations of real ti e and knowledge have included [37, 38, 39]. In [37] a technique for deter ining the te poral validity of shared data in real-ti e distributed syste s is proposed. The approach is based on a language consisting of Boolean, episte ic, dyna ic, and real-ti e te poral operators, but the se antics for these is not defined. In [38] a fusion of the branching ti e te poral logic (CTL) and the standard episte ic logic is presented. The se antics of the logic is given over an interpreted syste defined like in [18] with the difference of using runs defined fro real nu bers. This language is used to establish sound and co plete ter ination conditions for o-tion planning of robots, given initial and goal states. [39] presents a fra ework for knowledge-based analysis of clocks synchronisation in syste s with real-ti e constraints. In that work a relation of ti ed precedence as a generalisation of previous work by La port is defined, and it is shown how (inherent) knowledge about ti ed precedences can be applied to synchronise clocks opti ally. Like in [38], the se antics consists of runs that are functions over real ti e. The episte ic relations defined in this work assu e that agents have perfect recall.

Our paper differs fro the approaches above by considering quantitative te poral operators such as $\mathrm{EF}_{[0,10]}$ (eaning "possibly within 10 ti e units"), rather than qualitative operators EF (eaning "possibly in the future", but with no bound), and by not forcing the agents to have perfect recall. In addition, the logic TCTLKD also incorporates a notion of correctness of execution with respect to specifications, a concept not tackled in previous works, and associates a set of clocks to every agent not just to the syste as a whole. While the satisfiability proble for TCTLKD is undecidable, the TCTLKD odel checking proble , i.e., the proble of validity in a given odel, is decidable. Given this, it see s worthwhile to develop odel checking ethods for TCTLKD in the sa e fashion to what has been pursued for the sa e odalities but on discrete ti e [42]. In fact, a preli inary version of the TCTLK[4] bounded odel checking ethod is presented in [40, 41].

References

1. Hintikka, J.: Knowledge and Belief, An Introduction to the Logic of the Two Notions. Cornell University Press, Ithaca (NY) and London (1962)
2. Aumann, R.J.: Agreeing to disagree. Annals of Statistics **4** (1976) 1236–1239
3. Fagin, R., Vardi, M.Y.: Knowledge and implicit knowledge in a distributed environment: Preliminary report. In Halpern, J.Y., ed.: TARK: Theoretical Aspects of Reasoning about Knowledge, San Francisco (CA), Morgan Kaufmann (1986) 187–206
4. Halpern, J., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artificial Intelligence **54** (1992) 319–379
5. van der Hoek, W.: Sytems for knowledge and belief. Journal of Logic and Computation **3** (1993) 173–195
6. Halpern, J.Y., Vardi, M.Y.: The complexity of reasoning about knowledge and time. In: ACM Symposium on Theory of Computing (STOC '86), Baltimore, USA, ACM Press (1986) 304–315
7. Halpern, J.Y., Vardi, M.Y.: The complexity of reasoning about knowledge and time 1: lower bounds. Journal of Computer and System Sciences **38** (1989) 195–237
8. van der Meyden, R., Wong, K.: Complete axiomatizations for reasoning about knowledge and branching time. Studia Logica **75** (2003) 93–123
9. Marek, W., Truszczyński, M.: Autoepistemic logic. Journal of the ACM **38** (1991) 587–618
10. Moore, R.: Possible-world semantics autoepistemic logic. In: Proceedings of Workshop on Non-Monotonic Reasoning, The AAAI Press (1984) 344–354
11. Baltag, A., Moss, L.S., Solecki, S.: The logic of public announcement, common knowledge, and private suspicions. In Gilboa, I., ed.: Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-98), San Francisco, Morgan Kaufmann (1998) 125–132
12. Lomuscio, A., Ryan, M.: An algorithmic approach to knowledge evolution. Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM) **13** (1999)
13. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Knowledge-based programs. Distributed Computing **10** (1997) 199–225

[4] The TCTLK logic is like TCTLKD but it does not contain the correct functioning operator, i.e, the operator $\mathcal{O}_i$ for $i \in \mathcal{AG}$.

14. Lomuscio, A., van der Meyden, R., Ryan, M.: Knowledge in multi-agent systems: Initial configurations and broadcast. ACM Transactions of Computational Logic **1** (2000)
15. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems: Specification. Springer-Verlag (1991)
16. Manna, Z., Pnueli, A.: Temporal Verification of Reactive Systems: Safety. Springer-Verlag (1995)
17. Manna, Z., Pnueli, A.: Completing the temporal picture. In: Selected papers of the 16th international colloquium on Automata, languages, and programming, Elsevier Science (1991) 97–130
18. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
19. Halpern, J., Meyden, R., Vardi, M.Y.: Complete axiomatisations for reasoning about knowledge and time. SIAM Journal on Computing **33** (2003) 674–703
20. van der Meyden, R.: Axioms for knowledge and time in distributed systems with perfect recall. In: Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science, Paris, France, IEEE Computer Society Press (1994) 448–457
21. Ben-Ari, M., Pnueli, A., Manna, Z.: The temporal logic of branching time. Acta Informatica **20** (1983) 207–226
22. Emerson, E.A.: Temporal and modal logic. In van Leeuwen, J., ed.: Handbook of Theoretical Computer Science, Elsevier Science Publishers (1990) 996–1071
23. Emerson, E.A., Halpern, J.Y.: Decision procedures and expressiveness in the temporal logic of branching time. Journal of Computer and System Sciences **30** (1985) 1–24
24. Lomuscio, A., Sergot, M.: Deontic interpreted systems. Studia Logica **75** (2003) 63–92
25. Lomuscio, A., Sergot, M.: Violation, error recovery, and enforcement in the bit transmission problem. Journal of Applied Logic **1**(2): 93–116, 2004
26. Alur, R., Dill, D.: Automata for modelling real-time systems. In: Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP'90). Volume 443 of Lecture Notes in Computer Science., Springer-Verlag (1990) 322–335
27. Dill, D.: Timing assumptions and verification of finite state concurrent systems. In: Automatic Verification Methods for Finite-State Systems. Volume 407 of Lecture Notes in Computer Science., Springer-Verlag (1989) 197–212
28. Behrmann, G., Larsen, K., Pearson, J., Weise, C., Yi, W.: Efficient timed reachability analysis using clock difference diagrams. In: Proceedings of the 11th International Conference on Computer Aided Verification (CAV'99). Volume 1633 of Lecture Notes in Computer Science., Springer-Verlag (1999) 341–353
29. Wang, F.: Efficient data structure of fully symbolic verification of real-time software systems. In: Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'00). Volume 1785 of Lecture Notes in Computer Science., Springer-Verlag (2000) 157–171
30. Penczek, W., Woźna, B., Zbrzezny, A.: SAT-based bounded model checking for the universal fragment of TCTL. Technical Report 947, ICS PAS, Ordona 21, 01-237 Warsaw (2002)
31. Penczek, W., Woźna, B., Zbrzezny, A.: Towards bounded model checking for the universal fragment of TCTL. In: Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'02). Volume 2469 of Lecture Notes in Computer Science., Springer-Verlag (2002) 265–288

32. Seshia, S., Bryant, R.: Unbounded, fully symbolic model checking of timed automata using boolean methods. In: Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03). Volume 2725 of Lecture Notes in Computer Science., Springer-Verlag (2003) 154–166
33. Tripakis, S., Yovine, S.: Analysis of timed systems using time-abstracting bisimulations. Formal Methods in System Design **18** (2001) 25–68
34. Alur, R., Courcoubetis, C., Dill, D.: Model checking in dense real-time. Information and Computation **104** (1993) 2–34
35. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Volume 53 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (2001)
36. Kang, I., Lee, I.: An efficient state space generation for the analysis of real-time systems. In: Proceedings of International Symposium on Software Testing and Analysis. (1996)
37. Anderson, S., Kuster-Filipe, J.: Guaranteeing temporal validity with a real-time logic of knowledge. In: Proceedings of the 1st International Workshop on Data Distribution for Real-Time Systems (DDRTS'03), ICDCS 2003 Workshop, Providence, Rhode Island, USA (2003) 178–183
38. Brafman, R.I., Latombe, J.C., Moses, Y., , Shoham, Y.: Application of a logic of knowledge to motion planning under uncertainty. Journal of the ACM **44** (1997) 633–668
39. Moses, Y., Bloom, B.: Knowledge, timed precedence and clocks. In: Proceedings of the 13th ACM symposium on Principles of Distributed Computing, ACM Press (1994) 274–303
40. Woźna, B., Lomuscio, A., Penczek, W.: Verification of deontic and epistemic properties of multiagent systems and its application to the bit transmission problem with faults. In: Proceedings of the 2nd Workshop on Logic and Communication in Multi-Agent Systems (LCMAS'04). (2004)
41. Lomuscio, A., Woźna, B., Penczek, W.: Bounded model checking for knowledge over real time. In: Proceedings of the International Workshop on Concurrency, Specification and Programming (CS&P'04). Volume 170 of Informatik-Berichte. (2004) 398–414
42. Raimondi, F., Lomuscio, A.: Automatic verification of deontic interpreted systems by model checking via OBDD's. In: Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECAI04). (2004)

Dynamic Logic for Plan Revision in Intelligent Agents

M. Birna van Riemsdijk[1], Frank S. de Boer[1,2,3], and John-Jules Ch. Meyer[1]

[1] ICS, Utrecht University, The Netherlands
[2] CWI, Amsterdam, The Netherlands
[3] LIACS, Leiden University, The Netherlands

Abstract. In this paper, we present a dynamic logic for a propositional version of the agent programming language 3APL. A 3APL agent has beliefs and a plan. The execution of a plan changes an agent's beliefs. Plans can be revised during execution. Due to these plan revision capabilities of 3APL agents, plans cannot be analyzed by structural induction as in for example standard propositional dynamic logic. We propose a dynamic logic that is tailored to handle the plan revision aspect of 3APL. For this logic, we give a sound and complete axiomatization.

1 Introduction

An agent is commonly seen as an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives [1]. Programming these flexible computing entities is not a trivial task. An important line of research in this area, is research on *cognitive* agents. These are agents endowed with high-level mental attitudes such as beliefs, desires, goals, plans, intentions, norms and obligations. Intelligent cognitive agents should be able to reason with these mental attitudes in order to exhibit the desired flexible problem solving behavior.

The very concept of (cognitive) agents is thus a complex one. It is imperative that programmed agents be amenable to precise and formal specification and verification, at least for some critical applications. This is recognized by (potential) appliers of agent technology such as NASA, which organizes specialized workshops on the subject of formal specification and verification of agents [2, 3].

In this paper, we are concerned with the verification of agents programmed in (a simplified version of) the cognitive agent programming language *3APL*[1] [4, 5, 6]. This language is based on theoretical research on cognitive notions [7, 8, 9, 10]. In the latest version [6], a 3APL agent has a set of beliefs, a plan and a set of goals. The idea is, that an agent tries to fulfill its goals by selecting appropriate plans, depending on its beliefs about the world. Beliefs should thus represent the world or environment of the agent; the goals represent the state of

[1] 3APL is to be pronounced as "triple-a-p-l".

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 16–32, 2005.

the world the agent wants to realize and plans are the means to achieve these goals.

As explained, cognitive agent programming languages are designed to program flexible behavior using high-level mental attitudes. In the various languages, these attitudes are handled in different ways. An important aspect of 3APL is the way in which plans are dealt with. A plan in 3APL can be executed, resulting in a change of the beliefs of the agent[2]. Now, in order to increase the possible flexibility of agents, 3APL [4] was endowed with a mechanism with which the programmer can program agents that can *revise* their plans during execution of the agent. This is a distinguishing feature of 3APL compared to other agent programming languages and architectures [11, 12, 13, 14]. The idea is, that an agent should not blindly execute an adopted plan, but it should be able to revise it under certain conditions. As this paper focusses on the plan revision aspect of 3APL, we consider a version of the language with only beliefs and plans, i.e., without goals. We will use a propositional and otherwise slightly simplified variant of the original 3APL language as defined in [4].

In 3APL, the plan revision capabilities can be programmed through plan revision rules. These rules consist of a head and a body, both representing a plan. A plan is basically a sequence of so-called basic actions. These actions can be executed. The idea is, informally, that an agent can apply a rule if it has a plan corresponding to the head of this rule, resulting in the replacement of this plan by the plan in the body of the rule. The introduction of these capabilities now gives rise to interesting issues concerning the characteristics of plan execution, as will become clear in the sequel. This has implications for reasoning about the result of plan execution and therefore for the formal verification of 3APL agents, which we are concerned with in this paper.

To be more specific, after defining (a simplified version of) 3APL and its semantics (section 2), we propose a dynamic logic for proving properties of 3APL plans in the context of plan revision rules (section 3). For this logic, we provide a sound and complete axiomatization (section 4).

As for related work, verification of agents programmed in an agent programming language has for example been addressed in [15]. This paper addresses model checking of the agent programming language AgentSpeak. A sketch of a dynamic logic to reason about 3APL agents has been given in [5]. This logic however is designed to reason about a 3APL interpreter or deliberation language, whereas in this paper we take a different viewpoint and reason about plans. In [16], a programming logic (without axiomatization) was given for a fragment of 3APL without plan revision rules. Further, the operational semantics of plan revision rules is similar to that of procedures in procedural programming. In fact, plan revision rules can be viewed as an extension of procedures. Logics and semantics for procedural languages are for example studied in De Bakker [17]. Although the operational semantics of procedures and plan revision rules are similar, techniques for reasoning about procedures cannot be used for plan

[2] A change in the environment is a possible “side effect” of the execution of a plan.

revision rules. This is due to the fact that the introduction of these rules results in the semantics of the sequential composition operator no longer being compositional (see section 3). This issue has also been considered from a semantic perspective in [18, 19]. In [20], a framework for planning in dynamic environments is presented in a logic programming setting. The approach is based on hierarchical task network planning. The motivation for that work is similar to the motivation for the introduction of plan revision rules.

To the best of our knowledge, this is the first attempt to design a logic and deductive system for plan revision rules or similar language constructs. Considering the semantic difficulties that arise with the introduction of this type of construct, it is not a priori obvious that it would be possible at all to design a deductive system to reason about these constructs. The main aim of this work was thus to investigate whether it is possible to define such a system and in this way also to get a better theoretical understanding of the construct of plan revision rules. Whether the system presented in this paper is also practically useful to verify 3APL agents, remains to be seen and will be subject to further research.

2 3APL

2.1 Syntax

Below, we define belief bases and plans. A belief base is a set of propositional formulas. A plan is a sequence of basic actions and abstract plans. Basic actions can be executed, resulting in a change to the beliefs of the agent. An abstract plan can, in contrast with basic actions, not be executed directly in the sense that it updates the belief base of an agent. Abstract plans serve as an abstraction mechanism like procedures in procedural programming. If a plan consists of an abstract plan, this abstract plan could be transformed into basic actions through the application of plan revision rules, which will be introduced below[3].

In the sequel, a language defined by inclusion shall be the smallest language containing the specified elements.

Definition 1. *(belief bases)* Assume a propositional language $\mathcal{L}$ with typical formula q and the connectives $\wedge$ and $\neg$ with the usual meaning. Then the set of belief bases Σ with typical element σ is defined to be $\wp(\mathcal{L})$.[4]

Definition 2. *(plans)* Assume that a set $\mathsf{BasicAction}$ with typical element a is given, together with a set $\mathsf{AbstractPlan}$ with typical element p. Then the set of plans Π with typical element π is defined as follows:

- $\mathsf{BasicAction} \cup \mathsf{AbstractPlan} \subseteq \Pi$,
- if $c \in (\mathsf{BasicAction} \cup \mathsf{AbstractPlan})$ and $\pi \in \Pi$ then $c\,;\pi \in \Pi$.

[3] Abstract plans could also be modelled as non-executable basic actions.
[4] $\wp(\mathcal{L})$ denotes the powerset of $\mathcal{L}$.

Basic actions and abstract plans are called atomic plans and are typically denoted by c. For technical convenience, plans are defined to have a list structure, which means strictly speaking, that we can only use the sequential composition operator to concatenate an atomic plan and a plan, rather than concatenating two arbitrary plans. In the following, we will however also use the sequential composition operator to concatenate arbitrary plans π_1 and π_2 yielding $\pi_1; \pi_2$. The operator should in this case be read as a function taking two plans that have a list structure and yielding a new plan that also has this structure. The plan π_1 will thus be the prefix of the resulting plan.

We use ϵ to denote the empty plan, which is an empty list. The concatenation of a plan π and the empty list is equal to π, i.e., $\epsilon; \pi$ and $\pi; \epsilon$ are identified with π.

A plan and a belief base can together constitute a so-called configuration. During computation or execution of the agent, the elements in a configuration can change.

Definition 3. *(configuration)* Let Σ be the set of belief bases and let Π be the set of plans. Then $\Pi \times \Sigma$ is the set of configurations of a 3APL agent.

Plan revision rules consist of a head π_h and a body π_b. Informally, an agent that has a plan π_h, can replace this plan by π_b when applying a plan revision rule of this form.

Definition 4. *(plan revision (PR) rules)* The set of PR rules $\mathcal{R}$ is defined as follows: $\mathcal{R} = \{\pi_h \rightsquigarrow \pi_b \mid \pi_h, \pi_b \in \Pi, \pi_h \neq \epsilon\}$.[5]

Take for example a plan $a; b$ where a and b are basic actions, and a PR rule $a; b \rightsquigarrow c$. The agent can then either execute the actions a and b one after the other, or it can apply the PR rule yielding a new plan c, which can in turn be executed. A plan p consisting of an abstract plan cannot be executed, but can only be transformed using a procedure-like PR rule such as $p \rightsquigarrow a$.

Below, we provide the definition of a 3APL agent. The function $\mathcal{T}$, taking a basic action and a belief base and yielding a new belief base, is used to define how belief bases are updated when a basic action is executed.

Definition 5. *(3APL agent)* A 3APL agent $\mathcal{A}$ is a tuple $\langle \mathsf{Rule}, \mathcal{T} \rangle$ where $\mathsf{Rule} \subseteq \mathcal{R}$ is a finite set of PR rules and $\mathcal{T} : (\mathsf{BasicAction} \times \sigma) \to \sigma$ is a partial function, expressing how belief bases are updated through basic action execution.

2.2 Semantics

The semantics of a programming language can be defined as a function taking a statement and a state, and yielding the set of states resulting from executing the

[5] In [4], PR rules were defined to have a guard, i.e., rules were of the form $\pi_h \mid \phi \rightsquigarrow \pi_b$. For a rule to be applicable, the guard should then hold. For technical convenience and because we want to focus on the plan revision aspect of these rules, we however leave out the guard in this paper. The results could be extended for rules with a guard.

initial state ent in the initial state. In this way, a state ent can be viewed as a transfor ation function on states. In 3APL, plans can be seen as state ents and belief bases as states on which these plans operate. There are various ways of defining a se antic function and in this paper we are concerned with the so-called *operational* se antics (see for exa ple De Bakker [17] for details on this subject).

The operational se antics of a language is usually defined using transition syste s [21]. A transition syste for a progra ing language consists of a set of axio s and derivation rules for deriving transitions for this language. A transition is a transfor ation of one configuration into another and it corresponds to a single co putation step. Let $\mathcal{A} = \langle \mathsf{Rule}, \mathcal{T} \rangle$ be a 3APL agent and let $\mathsf{BasicAction}$ be a set of basic actions. Below, we give the transition syste $\mathsf{Trans}_{\mathcal{A}}$ for our si -plified 3APL language, which is based on the syste given in [4]. This transition syste is specific to agent $\mathcal{A}$.

There are two kinds of transitions, i.e., transitions describing the execution of basic actions and those describing the application of a plan revision rule. The transitions are labelled to denote the kind of transition. A basic action at the head of a plan can be executed in a configuration if the function $\mathcal{T}$ is defined for this action and the belief base in the configuration. The execution results in a change of belief base as specified through $\mathcal{T}$ and the action is re oved fro the plan.

Definition 6. *(action execution)* Let $a \in \mathsf{BasicAction}$.

$$\frac{\mathcal{T}(a, \sigma) = \sigma'}{\langle a; \pi, \sigma \rangle \rightarrow_{exec} \langle \pi, \sigma' \rangle}$$

A plan revision rule can be applied in a configuration if the head of the rule is equal to a prefix of the plan in the configuration. The application of the rule results in the revision of the plan, such that the prefix equal to the head of the rule is replaced by the plan in the body of the rule. A rule $a; b \rightsquigarrow c$ can for exa ple be applied to the plan $a; b; c$, yielding the plan $c; c$. The belief base is not changed through plan revision.

Definition 7. *(rule application)* Let $\rho : \pi_h \rightsquigarrow \pi_b \in \mathsf{Rule}$.

$$\langle \pi_h; \pi, \sigma \rangle \rightarrow_{app} \langle \pi_b; \pi, \sigma \rangle$$

In the sequel, it will be useful to have a function taking a PR rule and a plan, and yielding the plan resulting fro the application of the rule to this given plan. Based on this function, we also define a function taking a set of PR rules and a plan and yielding the set of rules applicable to this plan.

Definition 8. *(rule application)* Let $\mathcal{R}$ be the set of PR rules and let Π be the set of plans. Let $\rho : \pi_h \rightsquigarrow \pi_b \in \mathcal{R}$ and $\pi, \pi' \in \Pi$. The partial function $apply : (\mathcal{R} \times \Pi) \rightarrow \Pi$ is then defined as follows.

$$apply(\rho)(\pi) = \begin{cases} \pi_b; \pi' & \text{if } \pi = \pi_h; \pi', \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The function *applicable* : $(\wp(\mathcal{R}) \times \mathit{\Pi}) \to \wp(\mathcal{R})$ yielding the set of rules applicable to a certain plan, is then as follows: $\mathit{applicable}(\mathsf{Rule}, \pi) = \{\rho \in \mathsf{Rule} \mid \mathit{apply}(\rho)(\pi)$ is defined$\}$.

Using the transition syste , individual transitions can be derived for a 3APL agent. These transitions can be put in sequel, yielding transition sequences. Fro a transition sequence, one can obtain a *computation sequence* by re oving the plan co ponent of all configurations occurring in the transition sequence. In the following definitions, we for ally define co putation sequences and we specify the function yielding these sequences, given an initial configuration.

Definition 9. *(computation sequences)* The set $\mathit{\Sigma}^+$ of finite co putation sequences is defined as $\{\sigma_1, \ldots, \sigma_i, \ldots, \sigma_n \mid \sigma_i \in \mathit{\Sigma}, 1 \leq i \leq n, n \in \mathbb{N}\}$.

Definition 10. *(function for calculating computation sequences)* Let $x_i \in \{\mathit{exec}, \mathit{app}\}$ for $1 \leq i \leq m$. The function $\mathcal{C}^{\mathcal{A}} : (\mathit{\Pi} \times \mathit{\Sigma}) \to \wp(\mathit{\Sigma}^+)$ is then as defined below.

$$\mathcal{C}^{\mathcal{A}}(\pi, \sigma) = \{\sigma, \ldots, \sigma_m \in \mathit{\Sigma}^+ \mid \theta = \langle \pi, \sigma \rangle \to_{x_1} \ldots \to_{x_m} \langle \epsilon, \sigma_m \rangle \text{ is a finite sequence of transitions in } \mathsf{Trans}_{\mathcal{A}}\}.$$

Note that we only take into account successfully ter inating transition sequences, i.e., those sequences ending in a configuration with an e pty plan. Using the function defined above, we can now define the operational se antics of 3APL.

Definition 11. *(operational semantics)* Let $\kappa : \wp(\mathit{\Sigma}^+) \to \wp(\mathit{\Sigma})$ be a function yielding the last ele ents of a set of finite co putation sequences, which is defined as follows: $\kappa(\mathit{\Delta}) = \{\sigma_n \mid \sigma_1, \ldots, \sigma_n \in \mathit{\Delta}\}$. The operational se antic function $\mathcal{O}^{\mathcal{A}} : \mathit{\Pi} \to (\mathit{\Sigma} \to \wp(\mathit{\Sigma}))$ is defined as follows:

$$\mathcal{O}^{\mathcal{A}}(\pi)(\sigma) = \kappa(\mathcal{C}^{\mathcal{A}}(\pi, \sigma)).$$

We will so eti es o it the superscript $\mathcal{A}$ fro functions as defined above, for reasons of presentation. The exa ple below is used to explain the definition of the operational se antics.

Example 1. Let $\mathcal{A}$ be an agent with PR rules $\{p; a \leadsto b, p \leadsto c\}$, where p is an abstract plan and a, b, c are basic actions. Let σ_a be the belief base resulting fro the execution of a in σ, i.e., $\mathcal{T}(a, \sigma) = \sigma_a$, let be σ_{ab} the belief resulting fro executing first a and then b in σ, etc.

Then $\mathcal{C}^{\mathcal{A}}(p; a)(\sigma) = \{(\sigma, \sigma, \sigma_b), (\sigma, \sigma, \sigma_c, \sigma_{ca})\}$, which is based on the transition sequences $\langle p; a, \sigma \rangle \to_{app} \langle b, \sigma \rangle \to_{exec} \langle \epsilon, \sigma_b \rangle$ and $\langle p; a, \sigma \rangle \to_{app} \langle c; a, \sigma \rangle \to_{exec} \langle a, \sigma_c \rangle \to_{exec} \langle \epsilon, \sigma_{ca} \rangle$. We thus have that $\mathcal{O}^{\mathcal{A}}(p; a)(\sigma) = \{\sigma_b, \sigma_{ca}\}$.

3 Dynamic Logic

In programming language research, an important area is the specification and verification of programs. Program logics are designed to facilitate this process. One such logic is dynamic logic [22, 23], with which we are concerned in this paper. In dynamic logic, programs are explicit syntactic constructs in the logic. To be able to discuss the effect of the execution of a program π on the truth of a formula ϕ, the modal construct $[\pi]\phi$ is used. This construct intuitively states that in all states in which π halts, the formula ϕ holds.

Programs in general are constructed from atomic programs and composition operators. An example of a composition operator is the sequential composition operator (;), where the program $\pi_1; \pi_2$ intuitively means that π_1 is executed first, followed by the execution of π_2. The semantics of such a compound program can in general be determined by the semantics of the parts of which it is composed. This compositionality property allows analysis by structural induction (see also [24]), i.e., analysis of a compound statement by analysis of its parts. Analysis of the sequential composition operator by structural induction can in dynamic logic be expressed by the following formula, which is usually a validity: $[\pi_1; \pi_2]\phi \leftrightarrow [\pi_1][\pi_2]\phi$. For 3APL plans on the contrary, this formula does not always hold. This is due to the presence of PR rules.

We will informally explain this using the 3APL agent of example 1. As explained, the operational semantics of this agent, given initial plan $p; a$ and initial state σ, is as follows: $\mathcal{O}(p; a)(\sigma) = \{\sigma_b, \sigma_{ca}\}$. Now compare the result of first "executing"[6] p in σ and then executing a in the resulting belief base, i.e., compare the set $\mathcal{O}(a)(\mathcal{O}(p)(\sigma))$. In this case, there is only one successfully terminating transition sequence and it ends in σ_{ca}, i.e., $\mathcal{O}(a)(\mathcal{O}(p)(\sigma)) = \{\sigma_{ca}\}$. Now, if it would be the case that $\sigma_{ca} \models \phi$ but $\sigma_b \not\models \phi$, the formula $[p; a]\phi \leftrightarrow [p][a]\phi$ would not hold[7].

Analysis of plans by structural induction in this way thus does not work for 3APL. In order to be able to prove correctness properties of 3APL programs however, one can perhaps imagine that it is important to have *some* kind of induction. As we will show in the sequel, the kind of induction that can be used to reason about 3APL programs, is induction on the *number of PR rule applications in a transition sequence*. We will introduce a dynamic logic for 3APL based on this idea.

3.1 Syntax

In order to be able to do induction on the number of PR rule applications in a transition sequence, we introduce so-called *restricted plans*. These are plans,

[6] We will use the word "execution" in two ways. Firstly, as in this context, we will use it to denote the execution of an arbitrary plan in the sense of going through several transition of type *exec* or *app*, starting in a configuration with this plan and resulting in some final configurations. Secondly, we will use it to refer to the execution of a basic action in the sense of going through a transition of type *exec*.

[7] In particular, the implication would not hold from right to left.

annotated with a natural number[8]. Informally, if the restriction parameter of a plan is n, the number of rule applications during execution of this plan cannot exceed n.

Definition 12. *(restricted plans)* Let Π be the language of plans and let $\mathbb{N}^- = \mathbb{N} \cup \{-1\}$. Then, the language Π_r of restricted plans is defined as $\{\pi\restriction_n \mid \pi \in \Pi, n \in \mathbb{N}^-\}$.

Below, we define the language of dynamic logic in which properties of 3APL agents can be expressed. In the logic, one can express properties of restricted plans. As will become clear in the sequel, one can prove properties of the plan of a 3APL agent by proving properties of restricted plans.

Definition 13. *(plan revision dynamic logic (*PRDL*))* Let $\pi\restriction_n \in \Pi_r$ be a restricted plan. Then the language of dynamic logic $\mathcal{L}_{\mathsf{PRDL}}$ with typical element ϕ is defined as follows:

- $\mathcal{L} \subseteq \mathcal{L}_{\mathsf{PRDL}}$,
- if $\phi \in \mathcal{L}_{\mathsf{PRDL}}$, then $[\pi\restriction_n]\phi \in \mathcal{L}_{\mathsf{PRDL}}$,
- if $\phi, \phi' \in \mathcal{L}_{\mathsf{PRDL}}$, then $\neg\phi \in \mathcal{L}_{\mathsf{PRDL}}$ and $\phi \wedge \phi' \in \mathcal{L}_{\mathsf{PRDL}}$.

3.2 Semantics

In order to define the semantics of PRDL, we first define the semantics of restricted plans. As for ordinary plans, we also define an operational semantics for restricted plans. We do this by defining a function for calculating computation sequences, given an initial restricted plan and a belief base.

Definition 14. *(function for calculating computation sequences)* Let $x_i \in \{exec, app\}$ for $1 \leq i \leq m$. Let $N_{app}(\theta)$ be a function yielding the number of transitions of the form $s_i \rightarrow_{app} s_{i+1}$ in the sequence of transitions θ. The function $\mathcal{C}_r^{\mathcal{A}} : (\Pi_r \times \Sigma) \rightarrow \wp(\Sigma^+)$ is then as defined below.

$$\begin{aligned}\mathcal{C}_r^{\mathcal{A}}(\pi\restriction_n, \sigma) = \{\sigma, \ldots, \sigma_m \in \Sigma^+ \mid \theta = \langle\pi, \sigma\rangle \rightarrow_{x_1} \ldots \rightarrow_{x_m} \langle\epsilon, \sigma_m\rangle \\ \text{is a finite sequence of transitions in } \mathsf{Trans}_{\mathcal{A}} \text{ where } 0 \leq N_{app}(\theta) \leq n\}\end{aligned}$$

As one can see in the definition above, the computation sequences $\mathcal{C}_r^{\mathcal{A}}(\pi\restriction_n, \sigma)$ are based on transition sequences starting in configuration $\langle\pi, \sigma\rangle$. The number of rule applications in these transition sequences should be between 0 and n, in contrast with the function $\mathcal{C}^{\mathcal{A}}$ of definition 10, in which there is no restriction on this number.

Based on the function $\mathcal{C}_r^{\mathcal{A}}$, we define the operational semantics of restricted plans by taking the last elements of the computation sequences yielded by $\mathcal{C}_r^{\mathcal{A}}$. The set of belief bases is empty if the restriction parameter is equal to -1.

[8] Or with the number -1. The number -1 is introduced for technical convenience and it will become clear in the sequel why we need this.

Definition 15. *(operational semantics)* Let κ be as in definition 11. The operational se antic function $\mathcal{O}_r^{\mathcal{A}} : \Pi_r \to (\Sigma \to \wp(\Sigma))$ is defined as follows:

$$\mathcal{O}_r^{\mathcal{A}}(\pi\lceil_n)(\sigma) = \begin{cases} \kappa(\mathcal{C}_r^{\mathcal{A}}(\pi\lceil_n, \sigma)) & \text{if } n \geq 0, \\ \emptyset & \text{if } n = -1. \end{cases}$$

In the following proposition, we relate the operational se antics of plans and the operational se antics of restricted plans.

Proposition 1.

$$\bigcup_{n \in \mathbb{N}} \mathcal{O}_r(\pi\lceil_n)(\sigma) = \mathcal{O}(\pi)(\sigma)$$

Proof. I ediate fro definitions 15, 14, 11 and 10.

Using the operational se antics of restricted plans, we can now define the se antics of the dyna ic logic.

Definition 16. *(semantics of* PRDL*)* Let $q \in \mathcal{L}$ be a propositional for ula, let $\phi, \phi' \in \mathcal{L}_{\mathsf{PRDL}}$ and let $\models_{\mathcal{L}}$ be the entail ent relation defined for $\mathcal{L}$ as usual. The se antics $\models_{\mathcal{A}}$ of $\mathcal{L}_{\mathsf{PRDL}}$ is then as defined below.

$$\begin{array}{lcl} \sigma \models_{\mathcal{A}} q & \Leftrightarrow & \sigma \models_{\mathcal{L}} q \\ \sigma \models_{\mathcal{A}} [\pi\lceil_n]\phi & \Leftrightarrow & \forall \sigma' \in \mathcal{O}_r^{\mathcal{A}}(\pi\lceil_n)(\sigma) : \sigma' \models_{\mathcal{A}} \phi \\ \sigma \models_{\mathcal{A}} \neg\phi & \Leftrightarrow & \sigma \not\models_{\mathcal{A}} \phi \\ \sigma \models_{\mathcal{A}} \phi \wedge \phi' & \Leftrightarrow & \sigma \models_{\mathcal{A}} \phi \text{ and } \sigma \models_{\mathcal{A}} \phi' \end{array}$$

As $\mathcal{O}_r^{\mathcal{A}}$ is defined in ter s of agent $\mathcal{A}$, so is the se antics of $\mathcal{L}_{\mathsf{PRDL}}$. We use the subscript $\mathcal{A}$ to indicate this. Let $\mathsf{Rule} \subseteq \mathcal{R}$ be a finite set of PR rules. If $\forall \mathcal{T}, \sigma : \sigma \models_{\langle \mathsf{Rule}, \mathcal{T} \rangle} \phi$, we write $\models_{\mathsf{Rule}} \phi$.

In the dyna ic logic PRDL, one can express properties of restricted plans, rather than of ordinary 3APL plans. The operational se antics of ordinary plans $\mathcal{O}$ and of restricted plans $\mathcal{O}_r$ are however related (proposition 1). As the se antics of the construct $[\pi\lceil_n]\sigma$ is defined in ter s of $\mathcal{O}_r$, we can use this construct to specify properties of 3APL plans, as shown by the following corollary.

Corollary 1.

$$\forall n \in \mathbb{N} : \sigma \models_{\mathcal{A}} [\pi\lceil_n]\phi \Leftrightarrow \forall \sigma' \in \mathcal{O}^{\mathcal{A}}(\pi)(\sigma) : \sigma' \models_{\mathcal{A}} \phi$$

Proof. I ediate fro proposition 1 and definition 16.

4 The Axiom System

In order to prove properties of restricted plans, we propose a deductive syste for PRDL in this section. Rather than proving properties of restricted plans, the ai is however to prove properties of 3APL plans. We thus want to prove properties of the for $\forall n \in \mathbb{N} : [\pi\lceil_n]\phi$, as these are directly related to 3APL by corollary 1. The idea now is, that these properties can be proven by induction on n. We will explain this in ore detail after introducing the axio syste for restricted plans.

Definition 17. *(axiom system (AS$_{\mathsf{Rule}}$))* Let BasicAction be a set of basic actions, AbstractPlan be a set of abstract plans and Rule $\subseteq \mathcal{R}$ be a finite set of PR rules. Let $a \in$ BasicAction, let $p \in$ AbstractPlan, let $c \in$ (BasicAction $\cup$ AbstractPlan) and let ρ range over $applicable(\mathsf{Rule}, c; \pi)$. The following are then the axio s of the syste AS$_{\mathsf{Rule}}$.

(PRDL1) $[\pi\restriction_{-1}]\phi$
(PRDL2) $[p\restriction_0]\phi$
(PRDL3) $[\epsilon\restriction_n]\phi \leftrightarrow \phi$ if $0 \leq n$
(PRDL4) $[c; \pi\restriction_n]\phi \leftrightarrow [c\restriction_0][\pi\restriction_n]\phi \wedge \bigwedge_\rho [apply(\rho, c; \pi)\restriction_{n-1}]\phi$ if $0 \leq n$

(PL) axio s for propositional logic
(PDL) $[\pi\restriction_n](\phi \rightarrow \phi') \rightarrow ([\pi\restriction_n]\phi \rightarrow [\pi\restriction_n]\phi')$

The following are the rules of the syste AS$_{\mathsf{Rule}}$.

(GEN)

$$\frac{\phi}{[\pi\restriction_n]\phi}$$

(MP)

$$\frac{\phi_1,\ \phi_1 \rightarrow \phi_2}{\phi_2}$$

As the axio syste is relative to a given set of PR rules Rule, we will use the notation $\vdash_{\mathsf{Rule}} \phi$ to specify that ϕ is derivable in the syste AS$_{\mathsf{Rule}}$ above.

The idea is that properties of the for $\forall n \in \mathbb{N} : \vdash_{\mathsf{Rule}} [\pi\restriction_n]\phi$ can be proven by induction on n as follows. If we can prove $[\pi\restriction_0]\phi$ and $\forall n \in \mathbb{N} : ([\pi\restriction_n]\phi \vdash_{\mathsf{Rule}} [\pi\restriction_{n+1}]\phi)$, we can conclude the desired property. These pre ises should be proven using the axio syste above. Consider for exa ple an agent with a PR rule $a \rightsquigarrow a; a$ and assu e that $\mathcal{T}$ is defined such that $[a\restriction_0]\phi$. One can then prove $\forall n : [a\restriction_n]\phi$ by proving $[a\restriction_n]\phi \vdash_{\mathsf{Rule}} [a\restriction_{n+1}]\phi$, for arbitrary n.

We will now explain the PRDL axio s of the syste . The other axio s and the rules are standard for propositional dyna ic logic (PDL) [22]. We start by explaining the ost interesting axio : (PRDL4). We first observe that there are two types of transitions that can be derived for a 3APL agent: action execution and rule application (see definitions 6 and 7). Consider a configuration $\langle a; \pi, \sigma \rangle$ where a is a basic action. Then during co putation, possible next configurations are $\langle \pi, \sigma' \rangle$[9] (action execution) and $\langle apply(\rho, a; \pi), \sigma \rangle$ (rule application) where ρ ranges over the applicable rules, i.e., $applicable(\mathsf{Rule}, a; \pi)$[10]. We can thus analyze the plan $a; \pi$ by analyzing π after the execution of a, and the plans resulting fro applying a rule, i.e., $apply(\rho, a; \pi)$[11]. The execution of an action can be

9 Assuming that $\mathcal{T}(a, \sigma) = \sigma'$.
10 See definition 8 for the definitions of the functions *apply* and *applicable*.
11 Note that one could say we analyze a plan $a; \pi$ partly by structural induction, as it is partly analyzed in terms of a and π.

represented by the nu ber 0 as restriction para eter, yielding the first ter of the right-hand side of (PRDL4): $[a\upharpoonright_0][\pi\upharpoonright_n]\phi$[12]. The second ter is a conjunction of $[apply(\rho, c; \pi)\upharpoonright_{n-1}]\phi$ over all applicable rules ρ. The restriction para eter is $n-1$ as we have "used" one of our n per itted rule applications. The first three axio s represent basic properties of restricted plans. (PRDL1) can be used to eli inate the second ter on the right-hand side of axio (PRDL4), if the left-hand side is $[c; \pi\upharpoonright_0]\phi$. (PRDL2) can be used to eli inate the first ter on the right-hand side of (PRDL4), if c is an abstract plan. As abstract plans can only be transfor ed through rule application, there will be no resulting states if the restriction para eter of the abstract plan is 0, i.e., if no rule applications are allowed. (PRDL3) states that if ϕ is to hold after execution of the e pty plan, it should hold "now". It can be used to derive properties of an ato ic plan c, by using axio (PRDL4) with the plan $c; \epsilon$.

Example 2. Let $\mathcal{A}$ be an agent with one PR rule, i.e., $\mathsf{Rule} = \{a; b \rightsquigarrow c\}$ and let $\mathcal{T}$ be such that $[a\upharpoonright_0]\phi$, $[b\upharpoonright_0]\phi$ and $[c\upharpoonright_0]\phi$. We now want to prove that $\forall n : [a; b\upharpoonright_n]\phi$. We have $[a; b\upharpoonright_0]\phi$ by using that this is equivalent to $[a\upharpoonright_0][b\upharpoonright_0]\phi$ by proposition 3 (section 4.1). The latter for ula can be derived by applying (GEN) to $[b\upharpoonright_0]\phi$. We prove $\forall n \in \mathbb{N} : ([a; b\upharpoonright_n]\phi \vdash_{\mathsf{Rule}} [a; b\upharpoonright_{n+1}]\phi)$ by taking an arbitrary n and proving that $[a; b\upharpoonright_n]\phi \vdash_{\mathsf{Rule}} [a; b\upharpoonright_{n+1}]\phi$. Using (PRDL4) and (PRDL3), we have the following equivalences. In order to apply (PRDL4) to the conjunct $[c\upharpoonright_{n-1}]\phi$, n has to be greater than 0. This is however not a proble , as the result was proven separately for $n = 0$.

$$\begin{aligned}[a; b\upharpoonright_n]\phi &\leftrightarrow [a\upharpoonright_0][b\upharpoonright_n]\phi \wedge [c\upharpoonright_{n-1}]\phi\\ &\leftrightarrow [a\upharpoonright_0][b\upharpoonright_0][\epsilon\upharpoonright_n]\phi \wedge [c\upharpoonright_0][\epsilon\upharpoonright_{n-1}]\phi\\ &\leftrightarrow [a\upharpoonright_0][b\upharpoonright_0]\phi \wedge [c\upharpoonright_0]\phi\end{aligned}$$

Si ilarly, we have the following equivalences for $[a; b\upharpoonright_{n+1}]\phi$, yielding the desired result.

$$\begin{aligned}[a; b\upharpoonright_{n+1}]\phi &\leftrightarrow [a\upharpoonright_0][b\upharpoonright_{n+1}]\phi \wedge [c\upharpoonright_n]\phi\\ &\leftrightarrow [a\upharpoonright_0][b\upharpoonright_0][\epsilon\upharpoonright_{n+1}]\phi \wedge [c\upharpoonright_0][\epsilon\upharpoonright_n]\phi\\ &\leftrightarrow [a\upharpoonright_0][b\upharpoonright_0]\phi \wedge [c\upharpoonright_0]\phi\end{aligned}$$

4.1 Soundness and Completeness

The axio syste of definition 17 is sound.

Theorem 1. *(soundness)* Let $\phi \in \mathcal{L}_{\mathsf{PRDL}}$. Let $\mathsf{Rule} \subseteq \mathcal{R}$ be an arbitrary finite set of PR rules. Then the axio syste $\mathsf{AS}_{\mathsf{Rule}}$ is sound, i.e.:

$$\vdash_{\mathsf{Rule}} \phi \Rightarrow \models_{\mathsf{Rule}} \phi.$$

Proof. We prove soundness of the PRDL axio s of the syste $\mathsf{AS}_{\mathsf{Rule}}$.
(PRDL1) The proof is through observing that $\mathcal{O}_r(\pi\upharpoonright_{-1})(\sigma) = \emptyset$ by definition 15.

[12] In our explanation, we consider the case where c is a basic action, but the axiom holds also for abstract plans.

(PRDL2) The proof is analogous to the proof of axio (PRDL1), with p for π and 0 for -1 and using definition 6 to derive that $\mathcal{O}_r^{\mathcal{A}}(p\lceil_0)(\sigma) = \emptyset$.
(PRDL3) The proof is through observing that $\kappa(\mathcal{C}_r(\epsilon\lceil_n, \sigma)) = \{\sigma\}$ by definition 14.
(PRDL4) Let $\pi \in \Pi$ be an arbitrary plan and $\phi \in \mathcal{L}_{\mathsf{PRDL}}$ be an arbitrary PRDL for ula.
To prove: $\forall \mathcal{T}, \sigma : \sigma \models_{\langle \mathsf{Rule}, \mathcal{T}\rangle} [c;\pi\lceil_n]\phi \leftrightarrow [c\lceil_0][\pi\lceil_n]\phi \wedge \bigwedge_\rho [apply(\rho, c;\pi)\lceil_{n-1}]\phi$, i.e.:

$$\forall \mathcal{T}, \sigma : \sigma \models_{\langle \mathsf{Rule}, \mathcal{T}\rangle} [c;\pi\lceil_n]\phi \Leftrightarrow \forall \mathcal{T}, \sigma : \sigma \models_{\langle \mathsf{Rule}, \mathcal{T}\rangle} [c\lceil_0][\pi\lceil_n]\phi \text{ and } \forall \mathcal{T}, \sigma : \sigma \models_{\langle \mathsf{Rule}, \mathcal{T}\rangle} \bigwedge_\rho [apply(\rho, c;\pi)\lceil_{n-1}]\phi.$$

Let $\sigma \in \Sigma$ be an arbitrary belief base and let $\mathcal{T}$ be an arbitrary belief update function. Assu e $c \in \mathsf{BasicAction}$ and further ore assu e that $\langle c;\pi,\sigma\rangle \rightarrow_{execute} \langle \pi, \sigma_1\rangle$ is a transition in $\mathsf{Trans}_{\mathcal{A}}$, i.e., $\kappa(\mathcal{C}_r^{\mathcal{A}}(c\lceil_0, \sigma)) = \{\sigma_1\}$ by definition 14. Let ρ range over $applicable(\mathsf{Rule}, c;\pi)$. Now, observe the following by definition 14:

$$\kappa(\mathcal{C}_r^{\mathcal{A}}(c;\pi\lceil_n, \sigma)) = \kappa(\mathcal{C}_r^{\mathcal{A}}(\pi\lceil_n, \sigma_1)) \cup \bigcup_\rho \kappa(\mathcal{C}_r^{\mathcal{A}}(apply(\rho, c;\pi)\lceil_{n-1}, \sigma)). \tag{1}$$

If $c \in \mathsf{AbstractPlan}$ or if a transition of the for $\langle c;\pi,\sigma\rangle \rightarrow_{execute} \langle \pi, \sigma_1\rangle$ is not derivable, the first ter of the right-hand side of (1) is e pty.

($\Rightarrow$) Assu e $\sigma \models_{\mathsf{Rule}} [c;\pi\lceil_n]\phi$, i.e., by definition 16 $\forall \sigma' \in \mathcal{O}_r^{\mathcal{A}}(c;\pi\lceil_n, \sigma) : \sigma' \models_{\mathsf{Rule}} \phi$, i.e., by definition 15:

$$\forall \sigma' \in \kappa(\mathcal{C}_r^{\mathcal{A}}(c;\pi\lceil_n, \sigma)) : \sigma' \models_{\mathsf{Rule}} \phi. \tag{2}$$

To prove: (A) $\sigma \models_{\mathsf{Rule}} [c\lceil_0][\pi\lceil_n]\phi$ and (B) $\sigma \models_{\mathsf{Rule}} \bigwedge_\rho [apply(\rho, c;\pi)\lceil_{n-1}]\phi$.
(A) If $c \in \mathsf{AbstractPlan}$ or if a transition of the for $\langle c;\pi,\sigma\rangle \rightarrow_{execute} \langle \pi, \sigma_1\rangle$ is not derivable, the desired result follows i ediately fro axio (PRDL2) or an analogous proposition for non executable basic actions. If $c \in \mathsf{BasicAction}$, we have the following fro definitions 16 and 15.

$$\begin{aligned} \sigma \models_{\mathsf{Rule}} [c\lceil_0][\pi\lceil_n]\phi &\Leftrightarrow \forall \sigma' \in \mathcal{O}_r^{\mathcal{A}}(c\lceil_0, \sigma) : \sigma' \models_{\mathsf{Rule}} [\pi\lceil_n]\phi \\ &\Leftrightarrow \forall \sigma' \in \mathcal{O}_r^{\mathcal{A}}(c\lceil_0, \sigma) : \forall \sigma'' \in \mathcal{O}_r^{\mathcal{A}}(\pi\lceil_n, \sigma') : \sigma'' \models_{\mathsf{Rule}} \phi \\ &\Leftrightarrow \forall \sigma' \in \kappa(\mathcal{C}_r^{\mathcal{A}}(c\lceil_0, \sigma)) : \forall \sigma'' \in \kappa(\mathcal{C}_r^{\mathcal{A}}(\pi\lceil_n, \sigma')) : \sigma'' \models_{\mathsf{Rule}} \phi \\ &\Leftrightarrow \forall \sigma'' \in \kappa(\mathcal{C}_r^{\mathcal{A}}(\pi\lceil_n, \sigma_1)) : \sigma'' \models_{\mathsf{Rule}} \phi \end{aligned} \tag{3}$$

Fro 1, we have that $\kappa(\mathcal{C}_r^{\mathcal{A}}(\pi\lceil_n, \sigma_1)) \subseteq \kappa(\mathcal{C}_r^{\mathcal{A}}(c;\pi\lceil_n, \sigma))$. Fro this and assu p- tion (2), we can now conclude the desired result (3).

(B) Let $c \in (\mathsf{BasicAction} \cup \mathsf{AbstractPlan})$ and let $\rho \in applicable(\mathsf{Rule}, c;\pi)$. Then we want to prove $\sigma \models_{\mathsf{Rule}} [apply(\rho, c;\pi)\lceil_{n-1}]\phi$. Fro definitions 16 and 15, we have the following.

$$\begin{aligned} \sigma \models_{\mathsf{Rule}} [apply(\rho, c;\pi)\lceil_{n-1}]\phi &\Leftrightarrow \forall \sigma' \in \mathcal{O}_r^{\mathcal{A}}(apply(\rho, c;\pi)\lceil_{n-1}, \sigma) : \sigma' \models_{\mathsf{Rule}} \phi \\ &\Leftrightarrow \forall \sigma' \in \kappa(\mathcal{C}_r^{\mathcal{A}}(apply(\rho, c;\pi)\lceil_{n-1}, \sigma)) : \sigma' \models_{\mathsf{Rule}} \phi \end{aligned} \tag{4}$$

Fro 1, we have that $\kappa(\mathcal{C}_r^{\mathcal{A}}(apply(\rho, c; \pi)\restriction_{n-1}, \sigma)) \subseteq \kappa(\mathcal{C}_r^{\mathcal{A}}(c; \pi\restriction_n, \sigma))$. Fro this and assu ption (2), we can now conclude the desired result (4).

($\Leftarrow$) Assu e $\sigma \models_{\mathsf{Rule}} [c\restriction_0][\pi\restriction_n]\phi$ and $\sigma \models_{\mathsf{Rule}} \bigwedge_\rho [apply(\rho, c; \pi)\restriction_{n-1}]\phi$, i.e., $\forall \sigma' \in \kappa(\mathcal{C}_r^{\mathcal{A}}(\pi\restriction_n, \sigma_1)) : \sigma' \models_{\mathsf{Rule}} \phi$ (3) and $\forall \sigma' \in \kappa(\mathcal{C}_r^{\mathcal{A}}(apply(\rho, c; \pi)\restriction_{n-1}, \sigma)) : \sigma' \models_{\mathsf{Rule}} \phi$ (4).
To prove: $\sigma \models_{\mathsf{Rule}} [c; \pi\restriction_n]\phi$, i.e., $\forall \sigma' \in \kappa(\mathcal{C}_r^{\mathcal{A}}(c; \pi\restriction_n, \sigma)) : \sigma' \models_{\mathsf{Rule}} \phi$ (2). If $c \in \mathsf{AbstractPlan}$ or if a transition of the for $\langle c; \pi, \sigma\rangle \rightarrow_{execute} \langle \pi, \sigma_1\rangle$ is not derivable, we have that $\kappa(\mathcal{C}_r^{\mathcal{A}}(c; \pi\restriction_n, \sigma)) = \bigcup_\rho \kappa(\mathcal{C}_r^{\mathcal{A}}(apply(\rho, c; \pi)\restriction_{n-1}, \sigma))$ (1). Fro this and the assu ption, we have the desired result.

If $c \in \mathsf{BasicAction}$ and a transition of the for $\langle c; \pi, \sigma\rangle \rightarrow_{execute} \langle \pi, \sigma_1\rangle$ is derivable, we have (1). Fro this and the assu ption, we again have the desired result.

In order to prove co pleteness of the axio syste , we first prove proposition 2, which says that any for ula fro $\mathcal{L}_{\mathsf{PRDL}}$ can be rewritten into an equivalent for ula where all restriction para eters are 0. This proposition is proven by induction on the size of for ulas. The size of a for ula is defined by eans of the function $size : \mathcal{L}_{\mathsf{PRDL}} \to \mathbb{N}^3$. This function takes a for ula fro $\mathcal{L}_{\mathsf{PRDL}}$ and yields a triple $\langle x, y, z\rangle$, where x roughly corresponds to the su of the restriction para eters occurring in the for ula, y roughly corresponds to the su of the length of plans in the for ula and z is the length of the for ula.

Definition 18. *(size)* Let the following be a lexicographic ordering on tuples $\langle x, y, z\rangle \in \mathbb{N}^3$:

$$\begin{aligned}\langle x_1, y_1, z_1\rangle < \langle x_2, y_2, z_2\rangle \text{ iff } & x_1 < x_2 \text{ or} \\ & (x_1 = x_2 \text{ and } y_1 < y_2) \text{ or } (x_1 = x_2 \text{ and } y_1 = y_2 \text{ and } z_1 < z_2).\end{aligned}$$

Let max be a function yielding the axi u of two tuples fro $\mathbb{N}^3$ and let f and s respectively be functions yielding the first and second ele ent of a tuple. Let l be a function yielding the nu ber of sy bols of a syntactic entity and let $q \in \mathcal{L}$. The function $size : \mathcal{L}_{\mathsf{PRDL}} \to \mathbb{N}^3$ is then as defined below.

$$\begin{aligned}
size(q) &= \langle 0, 0, l(q)\rangle \\
size([\pi\restriction_n]\phi) &= \begin{cases} \langle n + f(size(\phi)), l(\pi) + s(size(\phi)), l([\pi\restriction_n]\phi)\rangle & \text{if } n > 0 \\ \langle f(size(\phi)), s(size(\phi)), l([\pi\restriction_n]\phi)\rangle & \text{otherwise} \end{cases} \\
size(\neg\phi) &= \langle f(size(\phi)), s(size(\phi)), l(\neg\phi)\rangle \\
size(\phi \wedge \phi') &= \langle f(max(size(\phi), size(\phi'))), s(max(size(\phi), size(\phi'))), l(\phi \wedge \phi')\rangle
\end{aligned}$$

In the proof of proposition 2, we use the following le a. The first clause specifies that the right-hand side of axio (PRDL4) is s aller than the left-hand side. This axio will usually be used by applying it fro left to right to prove a for ula such as $[\pi\restriction_n]\phi$. Intuitively, the fact that the for ula will get "s aller" as specified through the function $size$, suggests convergence of the deduction process.

Lemma 1. Let $\phi \in \mathcal{L}_{\mathsf{PRDL}}$, let $c \in (\mathsf{BasicAction} \cup \mathsf{AbstractPlan})$, let ρ range over $applicable(\mathsf{Rule}, c; \pi)$ and let $n > 0$. The following then holds:

1. $size([c\restriction_0][\pi\restriction_n]\phi \wedge \bigwedge_\rho [apply(\rho, c; \pi)\restriction_{n-1}]\phi) < size([c; \pi\restriction_n]\phi)$,
2. $size(\phi) < size(\phi \wedge \phi')$ and $size(\phi') < size(\phi \wedge \phi')$.

Proof. The proof is si ply by applying definition 18.

Proposition 2. Any for ula $\phi \in \mathcal{L}_{\mathsf{PRDL}}$ can be rewritten into an equivalent for ula ϕ_{PDL} where all restriction para eters are 0, i.e.:

$$\forall \phi \in \mathcal{L}_{\mathsf{PRDL}} : \exists \phi_{\mathsf{PDL}} \in \mathcal{L}_{\mathsf{PRDL}} : size(\phi_{\mathsf{PDL}}) = \langle 0, 0, l(\phi_{\mathsf{PDL}})\rangle \text{ and } \vdash_{\mathsf{Rule}} \phi \leftrightarrow \phi_{\mathsf{PDL}}.$$

Proof. The fact that a for ula ϕ has the property that it can be rewritten as specified in the proposition, will be denoted by $\mathsf{PDL}(\phi)$ for reasons that will beco e clear in the sequel. The proof is by induction on $size(\phi)$.

- $\phi \equiv q$
  $size(q) = \langle 0, 0, l(q)\rangle$ and let $q_{\mathsf{PDL}} = q$, then $\mathsf{PDL}(q)$.
- $\phi \equiv [\pi\restriction_n]\phi'$
  If $n = -1$, we have that $[\pi\restriction_n]\phi'$ is equivalent with $\top$ (PRDL1). As $\mathsf{PDL}(\top)$, we also have $\mathsf{PDL}([\pi\restriction_n]\phi')$ in this case.
  Let $n = 0$. We then have that $size([\pi\restriction_n]\phi') = \langle f(size(\phi')), s(size(\phi')), l([\pi\restriction_n]\phi')\rangle$ is greater than $size(\phi') = \langle f(size(\phi')), s(size(\phi')), l(\phi')\rangle$. By induction, we then have $\mathsf{PDL}(\phi')$, i.e., ϕ' can be rewritten into an equivalent for ula ϕ'_{PDL}, such that $size(\phi'_{\mathsf{PDL}}) = \langle 0, 0, l(\phi'_{\mathsf{PDL}})\rangle$. As $size([\pi\restriction_n]\phi'_{\mathsf{PDL}}) = \langle 0, 0, l([\pi\restriction_n]\phi'_{\mathsf{PDL}})\rangle$, we have $\mathsf{PDL}([\pi\restriction_n]\phi'_{\mathsf{PDL}})$ and therefore $\mathsf{PDL}([\pi\restriction_n]\phi')$.
  Let $n > 0$. Let $\pi \equiv \epsilon$. By le a 1, we have $size(\phi') < size([\epsilon\restriction_n]\phi')$. Therefore, by induction, $\mathsf{PDL}(\phi')$. As $[\epsilon\restriction_n]\phi'$ is equivalent with ϕ' by axio (PRDL3), we also have $\mathsf{PDL}([\epsilon\restriction_n]\phi')$. Now let $\pi \equiv c; \pi'$ and let $L = [c; \pi'\restriction_n]\phi'$ and $R = [c\restriction_0][\pi'\restriction_n]\phi' \wedge \bigwedge_\rho [apply(\rho, c; \pi')\restriction_{n-1}]\phi'$. By le a 1, we have that $size(R) < size(L)$. Therefore, by induction, we have $\mathsf{PDL}(R)$. As R and L are equivalent by axio (PRDL4), we also have $\mathsf{PDL}(L)$, yielding the desired result.
- $\phi \equiv \neg\phi'$
  We have that $size(\neg\phi') = \langle f(size(\phi')), s(size(\phi')), l(\neg\phi')\rangle$, which is greater than $size(\phi')$. By induction, we thus have $\mathsf{PDL}(\phi')$ and $size(\phi'_{\mathsf{PDL}}) = \langle 0, 0, l(\phi'_{\mathsf{PDL}})\rangle$. Then, $size(\neg\phi'_{\mathsf{PDL}}) = \langle 0, 0, l(\neg\phi'_{\mathsf{PDL}})\rangle$ and thus $\mathsf{PDL}(\neg\phi'_{\mathsf{PDL}})$ and therefore $\mathsf{PDL}(\neg\phi')$.
- $\phi \equiv \phi' \wedge \phi''$
  By le a 1, we have $size(\phi') < size(\phi' \wedge \phi'')$ and $size(\phi'') < size(\phi' \wedge \phi'')$. Therefore, by induction, $\mathsf{PDL}(\phi')$ and $\mathsf{PDL}(\phi'')$ and therefore $size(\phi'_{\mathsf{PDL}}) = \langle 0, 0, l(\phi'_{\mathsf{PDL}})\rangle$ and $size(\phi''_{\mathsf{PDL}}) = \langle 0, 0, l(\phi''_{\mathsf{PDL}})\rangle$. Then, $size(\phi'_{\mathsf{PDL}} \wedge \phi''_{\mathsf{PDL}}) = \langle 0, 0, l(\phi'_{\mathsf{PDL}} \wedge \phi''_{\mathsf{PDL}})\rangle$ and therefore $size((\phi' \wedge \phi'')_{\mathsf{PDL}}) = \langle 0, 0, l((\phi' \wedge \phi'')_{\mathsf{PDL}})\rangle$ and we can conclude $\mathsf{PDL}((\phi' \wedge \phi'')_{\mathsf{PDL}})$ and thus $\mathsf{PDL}(\phi' \wedge \phi'')$.

Although structural induction is not possible for plans in general, it *is* possible if we only consider action execution, i.e., if the restriction para eter is 0. This is specified in the following proposition, fro which we can conclude that a for ula ϕ with $size(\phi) = \langle 0, 0, l(\phi)\rangle$ satisfies all standard PDL properties.

Proposition 3. *(sequential composition)* Let $\mathsf{Rule} \subseteq \mathcal{R}$ be a finite set of PR rules. The following is then derivable in the axiom system $\mathsf{AS}_{\mathsf{Rule}}$.

$$\vdash_{\mathsf{Rule}} [\pi_1; \pi_2 \upharpoonright_0]\phi \leftrightarrow [\pi_1 \upharpoonright_0][\pi_2 \upharpoonright_0]\phi$$

Proof. The proof is through repeated application of axiom (PRDL4), first from left to right and then from right to left (also using axiom (PRDL1) to eliminate the rule application part of the axiom).

Theorem 2. *(completeness)* Let $\phi \in \mathcal{L}_{\mathsf{PRDL}}$ and let $\mathsf{Rule} \subseteq \mathcal{R}$ be a finite set of PR rules. Then the axiom system $\mathsf{AS}_{\mathsf{Rule}}$ is complete, i.e.:

$$\models_{\mathsf{Rule}} \phi \;\Rightarrow \vdash_{\mathsf{Rule}} \phi.$$

Proof. Let $\phi \in \mathcal{L}_{\mathsf{PRDL}}$. By proposition 2 we have that a formula ϕ_{PDL} exists such that $\vdash_{\mathsf{Rule}} \phi \leftrightarrow \phi_{\mathsf{PDL}}$ and $size(\phi_{\mathsf{PDL}}) = \langle 0, 0, l(\phi_{\mathsf{PDL}})\rangle$ and therefore by soundness of $\mathsf{AS}_{\mathsf{Rule}}$ also $\models_{\mathsf{Rule}} \phi \leftrightarrow \phi_{\mathsf{PDL}}$. Let ϕ_{PDL} be a formula with these properties.

$$\begin{aligned} \models_{\mathsf{Rule}} \phi \Leftrightarrow \models_{\mathsf{Rule}} \phi_{\mathsf{PDL}} \quad & (\models_{\mathsf{Rule}} \phi \leftrightarrow \phi_{\mathsf{PDL}}) \\ \Rightarrow \vdash_{\mathsf{Rule}} \phi_{\mathsf{PDL}} \quad & (\text{completeness of } \mathsf{PDL}) \\ \Leftrightarrow \vdash_{\mathsf{Rule}} \phi \quad & (\vdash_{\mathsf{Rule}} \phi \leftrightarrow \phi_{\mathsf{PDL}}) \end{aligned}$$

The second step in this proof needs some justification. The general idea is, that all PDL axioms and rules are applicable to a formula ϕ_{PDL} and moreover, these axioms and rules are contained in our axiom system $\mathsf{AS}_{\mathsf{Rule}}$. As PDL is complete, we have $\models_{\mathsf{Rule}} \phi_{\mathsf{PDL}} \Rightarrow \vdash_{\mathsf{Rule}} \phi_{\mathsf{PDL}}$. There are however some subtleties to be considered, as our action language is not exactly the same as the action language of PDL, nor is it a subset (at first sight).

In particular, the action language of PDL does not contain abstract plans or the empty action ϵ. These are axiomatized in the system $\mathsf{AS}_{\mathsf{Rule}}$ and the question is, how these axioms relate to the axiom system for PDL. It turns out, that the semantics of $p \upharpoonright_0$ and $\epsilon \upharpoonright_0$ (or $\epsilon \upharpoonright_n$, for that matter) correspond respectively to the special PDL actions `fail` (no resulting states if executed) and `skip` (the identity relation). These actions are respectively defined as **0**? and **1**?. Filling in these actions in the axiom for test ($[\psi?]\phi \leftrightarrow (\psi \to \phi)$), we get the following, corresponding exactly with the axioms (PRDL2) and (PRDL3).

$$\begin{aligned} [\mathbf{0}?]\phi \leftrightarrow (\mathbf{0} \to \phi) \;\Leftrightarrow\; [\mathbf{0}?]\phi \quad & \Leftrightarrow\; [\mathtt{fail}]\phi \\ [\mathbf{1}?]\phi \leftrightarrow (\mathbf{1} \to \phi) \;\Leftrightarrow\; [\mathbf{1}?]\phi \leftrightarrow \phi \quad & \Leftrightarrow\; [\mathtt{skip}]\phi \leftrightarrow \phi \end{aligned}$$

Our axiom system is complete for formulas ϕ_{PDL}, because it contains the PDL axioms and rules that are applicable to these formulas, that is, the axiom for sequential composition, the axioms for `fail` and `skip` as stated above, the axiom for distribution of box over implication and the rules (MP) and (GEN). The axiom for sequential composition is not explicitly contained in $\mathsf{AS}_{\mathsf{Rule}}$, but is derivable for formulas ϕ_{PDL} by proposition 3. Axiom (PRDL3), i.e., the more general version of $[\epsilon \upharpoonright_0]\phi \leftrightarrow \phi$, is needed in the proof of proposition 2, which is used elsewhere in this completeness proof.

5 Conclusion and Future Research

In this paper, we presented a dyna ic logic for reasoning about 3APL agents, tailored to handle the plan revision aspect of the language. As we argued, 3APL plans cannot be analyzed by structural induction. Instead, we proposed a logic of restricted plans, which should be used to prove properties of 3APL plans by doing induction on the restriction para eter.

Being able to do structural induction is usually considered an essential property of progra s in order to reason about the . As 3APL plans lack this property, it is not at all obvious that it should be possible to reason about the , especially using a clean logic with sound and co plete axio atization. The fact that we succeeded in providing such a logic, thus at least de onstrates this possibility.

We did so e preli inary experi ents in actually using the logic to prove properties of certain 3APL agents. More research is however needed to establish the practical usefulness of the logic to prove properties of 3APL agents and the possibility to do for exa ple auto ated theore proving. In this light, incorporation of interaction with an environ ent in the se antics is also an i portant issue for future research.

References

1. Wooldridge, M.: Agent-based software engineering. IEEE Proceedings Software Engineering **144** (1997) 26–37
2. Rash, J., Rouff, C., Truszkowski, W., Gordon, D., Hinchey, M., eds.: Formal Approaches to Agent-Based Systems (Proceedings of FAABS'01). Volume 1871 of LNAI., Berlin, Springer (2001)
3. Hinchey, M., Rash, J., Truszkowski, W., Rouff, C., Gordon-Spears, D., eds.: Formal Approaches to Agent-Based Systems (Proceedings of FAABS'02). Volume 2699 of LNAI., Berlin, Springer (2003)
4. Hindriks, K.V., de Boer, F.S., van der Hoek, W., Meyer, J.J.Ch.: Agent programming in 3APL. Int. J. of Autonomous Agents and Multi-Agent Systems **2** (1999) 357–401
5. van Riemsdijk, M.B., van der Hoek, W., Meyer, J.J.Ch.: Agent programming in Dribble: from beliefs to goals using plans. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03), Melbourne (2003) 393–400
6. Dastani, M., van Riemsdijk, M.B., Dignum, F., Meyer, J.J.Ch.: A programming language for cognitive agents: goal directed 3APL. In: Programming multiagent systems, First International Workshop (ProMAS'03). Volume 3067 of LNAI. Springer, Berlin (2004) 111–130
7. Bratman, M.E.: Intention, plans, and practical reason. Harvard University Press, Massachusetts (1987)
8. Cohen, P.R., Levesque, H.J.: Intention is choice with commitment. Artificial Intelligence **42** (1990) 213–261

9. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In Allen, J., Fikes, R., Sandewall, E., eds.: Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91), Morgan Kaufmann (1991) 473–484
10. van der Hoek, W., van Linder, B., Meyer, J.J.Ch.: An integrated modal approach to rational agents. In Wooldridge, M., Rao, A.S., eds.: Foundations of Rational Agency. Applied Logic Series 14. Kluwer, Dordrecht (1998) 133–168
11. Rao, A.S.: AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In van der Velde, W., Perram, J., eds.: Agents Breaking Away (LNAI 1038), Springer-Verlag (1996) 42–55
12. Shoham, Y.: Agent-oriented programming. Artificial Intelligence **60** (1993) 51–92
13. Giacomo, G.d., Lespérance, Y., Levesque, H.: *ConGolog*, a Concurrent Programming Language Based on the Situation Calculus. Artificial Intelligence **121** (2000) 109–169
14. Evertsz, R., Fletcher, M., Jones, R., Jarvis, J., Brusey, J., Dance, S.: Implementing Industrial Multi-Agent Systems Using JACK™. In: Proceedings of the first international workshop on programming multiagent systems (ProMAS'03). Volume 3067 of LNAI. Springer, Berlin (2004) 18–49
15. Bordini, R.H., Fisher, M., Pardavila, C., Wooldridge, M.: Model checking AgentSpeak. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03), Melbourne (2003) 409–416
16. Hindriks, K.V., de Boer, F.S., van der Hoek, W., Meyer, J.J.Ch.: A programming logic for part of the agent language 3APL. In: Proceedings of the First Goddard Workshop on Formal Approaches to Agent-Based Systems (FAABS'00). (2000)
17. de Bakker, J.: Mathematical Theory of Program Correctness. Series in Computer Science. Prentice-Hall International, London (1980)
18. van Riemsdijk, M.B., Meyer, J.J.Ch., de Boer, F.S.: Semantics of plan revision in intelligent agents. In Rattray, C., Maharaj, S., Shankland, C., eds.: Proceedings of the 10th International Conference on Algebraic Methodology And Software Technology (AMAST04). Volume 3116 of LNCS, Springer-Verlag (2004) 426–442
19. van Riemsdijk, M.B., Meyer, J.J.Ch., de Boer, F.S.: Semantics of plan revision in intelligent agents. Technical report, Utrecht University, Institute of Information and Computing Sciences (2003) UU-CS-2004-002.
20. Hayashi, H., Cho, K., Ohsuga, A.: A new HTN planning framework for agents in dynamic environments. In: Proceedings of the Fourth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA-IV). (2003) 108–133
21. Plotkin, G.D.: A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus (1981)
22. Harel, D.: First-Order Dynamic Logic. Lectures Notes in Computer Science 68. Springer, Berlin (1979)
23. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. The MIT Press, Cambridge, Massachusetts and London, England (2000)
24. van Emde Boas, P.: The connection between modal logic and algorithmic logics. In: Mathematical foundations of computer science 1978. Volume 64 of LNCS. Springer, Berlin (1978) 1–15

Contextual Taxonomies

Davide Grossi, Frank Dignum, and John-Jules Ch. Meyer

Utrecht University,
The Netherlands
{davide, dignum, jj}@cs.uu.nl

Abstract. We provide a formal characterization of a notion of contextual taxonomy, that is to say, a taxonomy holding only with respect to a specific context. To this aim, a new proposal for dealing with "contexts as abstract mathematical entities" is set forth, which is geared toward solving some problems arising in the area of normative system specifications for modeling multi-agent systems. Contexts are interpreted as sets of description logic models for different languages, and a number of operations on contexts are defined. Using this framework, a simple scenario taken from the legal domain is modeled, and a formal account of the so called open-texture of legal terms is provided characterizing the notions of "core" and "penumbra" of the meaning of a concept.

1 Introduction

The motivation of this work lies in problems stemming from the domain of normative system specifications for modeling multi-agent systems ([1, 2]). In [3, 4, 5] contexts have been advocated to play a central role in the specification of complex normative systems. The notion of context has obtained attention in AI researches since the seminal work [6], and much work has been carried out with regard to the logical analysis of this notion (see [7, 8] for an overview). With this work, we intend to pursue this research line providing a logical framework for dealing with a conception of context specifically derived from the aforementioned application domain. We nevertheless deem that the formal analysis we are going to present may give valuable insights for understanding contexts in general, also outside our specific domain of interest.

In general, the purpose of the present work is to propose a framework for grounding a new formal semantics of expressions such as: "A counts as B ([9]) in institution c", or "B supervenes A in institution c" ([10]), or "A conventionally generates B in institution c" ([11]), or "A translates (means) B in institution c" ([5]). These expressions, known in legal theory as *constitutive rules*, will be interpreted essentially as contextualized subsumption relations establishing taxonomies which hold only with respect to a specific (institutional) context. We came to a notion of *contextual taxonomy* through the analysis of some well known problems of underspecification, or more technically open-texture ([12]), typical of legal terminologies. These vagueness-related issues constitute, more concretely,

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 33–51, 2005.

the direct target of the work. We quote here an excerpt from [13] neatly exposing this type of problems.

> [Suppose a] legal rule forbids you to take a vehicle into the public park. Plainly this forbids an automobile, but what about bicycles, roller skates, toy automobiles? What about airplanes? Are these, as we say, to be called "vehicles" for the purpose of the rule or not? If we are to communicate with each other at all, and if, as in the most elementary form of law, we are to express our intentions that a certain type of behavior be regulated by rules, then the general words we use like "vehicle" in the case I consider must have some standard instance in which no doubts are felt about its application. There must be a *core* of settled meaning, but there will be, as well, a *penumbra* of debatable cases in which words are neither obviously applicable nor obviously ruled out. [...] We may call the problems which arise outside the hard core of standard instances or settled meaning "problems of the penumbra"; they are always with us whether in relation to such trivial things as the regulation of the use of the public park or in relation to the multidimensional generalities of a constitution.

Given a general (regional) rule not allowing vehicles within public parks, there might be a municipality allowing bicycles in its parks, and instead another one not allowing them in. What counts as a vehicle according to the first municipality, and what counts as a vehicle according to the second one then? This type of problems has been extensively approached especially from the perspective of the formalization of defeasible reasoning: the regional rule "all vehicles are banned from public parks" is defeated by the regulation of the first municipality stating that "all vehicles that are bicycles are allowed in the park" and establishing thus an *exception* to the general directive. The formalization of norms via non-monotonic techniques (see [14] for an overview) emphasizes the existence of exceptions to norms while understanding abstract terms in the standard way (all instances of bicycles are always vehicles). It has also been proposed to view the inclusion rules themselves as defaults: "normally, if something is a bicycle, then it is a vehicle" (for example [15, 5]). We deem these approaches, despite being effective in capturing the reasoning patterns involved in these scenarios, to be not adequate for analyzing problems related with the *meaning* of the terms that trigger those reasoning patterns. Those reasoning patterns are defeasible because the meaning of the terms involved is not definite, it is vague, it is -and this is the thesis we hold here- context dependent[1]. We propose therefore to analyze these "*problems of the penumbra*" in terms of the notion of context: according to (in the context of) the public parks regulation of the first municipality bicycles are not vehicles, according to (in the context of) the public parks regulation of the second one bicycles are vehicles. This reading will be interpreted as follows: "the subsumption of the concept `bicycle` under the concept `vehicle` holds in the context of the first municipality, but not in the context of the second one".

[1] The issue of the relationship between contextuality and defeasibility has been raised also in [7].

A defeasible reasoning analysis leads to a quite different reading, which flattens the eaning of concepts and handles its variations by eans of the notion of exception: "every exceptional instance of `bicycle` is not an instance of `vehicle`". Bringing contexts into play will instead allow for a neat characterization of the notions of "core" and "penu bra" of the eaning of a concept, a characterization which is not obtainable via the use of a notion of exception.

The re ainder of this paper is structured in accordance with the following outline. In Section 2 we will introduce the notion of *contextual taxonomy* aking use of a concrete scenario; in Section 3 we will provide a for al fra ework based on a very si ple type of description logic which accounts for this concept; in Section 4 we will provide a for alization of the scenario introduced, and we will for ally characterize the notions of conceptual "core" and "penu bra"; in Section 5 we will discuss relations with other work; finally, in Section 6, so e conclusive re arks are ade.

2 Contextualizing Taxonomies

Let us now depict a si ple scenario in order to state in clear ter s the exa ple used in the introduction.

Example 1. **(The public park scenario)** In the regulation governing access to public parks in region R it is stated that: "vehicles are not allowed within public parks". In this regulation no ention is ade of (possible) subconcepts of the concept vehicle, e.g., cars, bicycles, which ay help in identifying an instance of vehicle. In unicipal regulations subordinated to this regional one, specific subconcepts are instead handled. In unicipality M1, the following rule holds: "bicycles are allowed to access public parks". In M2 instead, it holds that: "bicycles are not allowed to access public parks". In both M1 and M2 it holds that: "cars are not allowed in public parks".

In this scenario the concept of `vehicle` is clearly open-textured. Instances of `car` (w.r.t. the taxono ies presupposed by M1 and M2) are "core" instances of `vehicle`, while instances of `bicycle` lay in the "penu bra" of `vehicle`. We will constantly refer back to this exa ple in the re aining of the work. In fact, our first ai will be to provide a for al fra ework able to account for scenarios for ally analogous to the afore entioned one[2].

Since the state ent about the need for addressing "contexts as abstract athe atical entities" in [6], any for alizations of the notion have been proposed (see [7] or [8] for an overview). Our proposal pursues the line of developing a se antic approach to the notion of context according to what was originally presented in [16]. In that work contexts are for alized as sets of first order logic odels. They are then connected via a relation called *compatibility relation*,

[2] Note that this scenario hides a typical form of contextual reasoning called "categorization" ([8]), or "perspective" ([7]).

which requires the sets of models constituting the different contexts to satisfy sets of domain specific inter-contextual inference rules (*bridge rules*). This theory has been variously used in work on specification of agent architectures ([17, 18]) where the stress lies in how contexts influence each other at a proof-theoretical level rather than at a semantical one (what can can be inferred in this context, given that something holds in some other context?). We follow the basic intuition of understanding contexts as sets of models. Nevertheless, since we are mainly interested in taxonomies, much simpler models will be used here[3]. Moreover, we will partly depart from the proposal in [16] trying to characterize also a set of operations meaningfully definable on contexts. In fact, what we are interested in is also an articulate characterization of the interplay between contexts: how can contexts be joined, abstracted, etc. Instead of focusing on bridge rules, which have to be introduced outside and separately from the contexts, we will define some operations on contexts such that all possible compatibility relations will be generated by the semantics of the contexts alone. This will provide intrinsic boundaries within which other bridge rules may later be defined.

To summarize, we will expose an approach to contexts which is driven by intuitions stemming from the analysis of normative terminologies, and which is based on description logic semantics.

3 A Formal Framework

The main requirements of the formal framework that we will develop are the following ones.

1. It should enable the possibility of expressing lexical differences. A much acknowledged characteristic of contextual reasoning is, indeed, that contexts should be specified on different languages ([19, 20, 21, 22]). The context of the national regulation about access to public parks should obviously be specified on a vocabulary that radically differs from the vocabulary used to specify the context of regulations about, for instance, immigration law: public park regulations do not talk about immigrants. Moreover, in Example 1, we observed that more concrete contexts make actually use of richer vocabularies: talking about vehicles comes down to talk about cars, bicycles, etc. In a nutshell, different contexts mean different ontologies and therefore different languages.
2. It should provide a formal semantics (as general as possible) for contextualized subsumption expressions, that is to say, for contextual taxonomies.
3. It should enable the possibility of describing operations between contexts.

Following these essential guidelines, a language and a semantics are introduced in what follows. The language will make use of part of description logic syntax, as regards the concept constructs, and will make use of a set of operators aimed at capturing the interplay of contexts. In particular, we will introduce:

[3] Basically models for description logic languages without roles. See Section 3.

– A *contextual conjunction* operator. Intuitively, it will yield a composition of contexts: the contexts “dinosaurs” and “contemporary reptiles” can be intersected on a language talking about crocodiles generating a common less general context like “crocodiles”.
– A *contextual disjunction* operator. Intuitively, it will yield a union of contexts: the contexts “viruses” and “bacterias” can be unified on a language talking about microorganisms generating a more general context like “viral or bacterial microorganisms”.
– A *contextual negation* operator. Intuitively, it will yield the context obtained via subtraction of the context negated: the negation of the context “viruses” on the language talking about microorganisms generates a context like “non viral microorganisms”.
– A *contextual abstraction* operator. Intuitively, it will yield the context consisting of some information extracted from the context to which the abstraction is applied: the context “crocodiles”, for instance, can be obtained via abstraction of the context “reptiles” on the language talking only about crocodiles. In other words, the operator prunes the information contained in the context “reptiles” keeping only what is expressible in the language which talks about crocodiles and abstracting from the rest.

Finally, also *maximum* and *minimum* contexts will be introduced: these will represent the most general, and respectively the less general, contexts on a language. As it appears from this list of examples, operators will need to be indexed with the language where the operation they denote takes place. The point is that contexts always belong to a language, and so do operations on them[4].

These intuitions about the semantics of context operators will be clarified and made more rigorous in Section 3.2 where the semantics of the framework will be presented, and in Section 4.1 where an example will be formalized.

3.1 Language

The language we are interested in defining is nothing but a formal metalanguage for talking about sets of subsumption relations, i.e., what in description logic are called terminological boxes (TBoxes). In fact, we consider only TBoxes specified on very simple languages containing just atomic concepts and boolean operators[5]. We decided to keep the syntax of these languages poor mainly for two reasons: firstly, because the use of boolean concept descriptions alone is enough

[4] Note that indexes might be avoided considering operators interpreted on operations taking place on one selected language, like the largest common language of the languages of the two contexts. However, this would result in a lack of expressivity that we prefer to avoid for the moment.

[5] In fact, we are going to extend the language of propositional logic. Nevertheless, the semantics we are going to use in Section 3.2 is not the semantics of propositional logic, and it is instead of a description logic kind. For this reason we deem instructive to refer to these simple languages also as description logic languages of the type $\mathcal{ALC}$ ([23]) but with an empty set of roles.

to . odel the scenario depicted in Exa ple 1; secondly, because this is still a preli inary proposal with which we ai to show how contextual reasoning and reasoning about vague notions are a enable to being handled on the basis of co putationally appealing logics. On this basis it will be natural, in future, to consider also richer languages.

The alphabet of the language $\mathcal{L}^{CT}$ (*language for contextual taxonomies*) contains therefore the alphabets of a fa ily of languages $\{\mathcal{L}_i\}_{0 \leq i \leq n}$. This fa ily is built on the alphabet of a given "global" language $\mathcal{L}$ which contains all the ter s occurring in the ele ents of the fa ily. Moreover, we take $\{\mathcal{L}_i\}_{0 \leq i \leq n}$ to be such that, for each non-e pty subset of ter s of the language $\mathcal{L}$, there exist a $\mathcal{L}_i$ which is built on that set and belongs to the fa ily. Each $\mathcal{L}_i$ contains a non-e pty finite set $\mathbf{A}_i$ of ato ic concepts (A), the zeroary operators $\perp$ (botto concept) and $\top$ (top concept), the unary operator $\neg$, and the binary operators $\sqcap$ and $\sqcup$[6].

Besides, the alphabet of $\mathcal{L}^{CT}$ contains a finite set of context identifiers $\mathbf{c}$, two fa ilies of zeroary operators $\{\perp_i\}_{0 \leq i \leq n}$ (ini u contexts) and $\{\top_i\}_{0 \leq i \leq n}$ (axi u contexts), two fa ilies of unary operators $\{abs_i\}_{0 \leq i \leq n}$ (context abstraction operators) and $\{\neg_i\}_{0 \leq i \leq n}$ (context negation operators), two fa ilies of binary operators $\{\curlywedge_i\}_{0 \leq i \leq n}$ (context conjunction operators) and $\{\curlyvee_i\}_{0 \leq i \leq n}$ (context disjunction operators), one context relation sy bol $\preccurlyeq$ (context c_1 "is at . ost as general as" context c_2) and a contextual subsu ption relation sy bol " . : . $\sqsubseteq$." (within context c, concept A_1 is a subconcept of concept A_2), finally, the sentential connectives $\sim$ (negation) and $\wedge$ (conjunction)[7]. Thus, the set Ξ of context constructs (ξ) is defined through the following BNF:

$$\xi ::= c \mid \perp_i \mid \top_i \mid \neg_i\, \xi \mid abs_i\, \xi \mid \xi_1 \curlywedge_i \xi_2 \mid \xi_1 \curlyvee_i \xi_2.$$

Concepts and concept constructors are then defined in the usual way. The set Γ of concept descriptions (γ) is defined through the following BNF:

$$\gamma ::= A \mid \perp \mid \top \mid \neg\gamma \mid \gamma_1 \sqcap \gamma_2 \mid \gamma_1 \sqcup \gamma_2.$$

The set $\mathcal{A}$ of assertions (α) is then defined through the following BNF:

$$\alpha ::= \xi : \gamma_1 \sqsubseteq \gamma_2 \mid \xi_1 \preccurlyeq \xi_2 \mid \sim \alpha \mid \alpha_1 \wedge \alpha_2.$$

Technically, a *contextual taxonomy* in $\mathcal{L}^{CT}$ is a set of subsu ption relation expressions which are contextualized with respect to the sa e context, e.g.: $\{\xi : \gamma_1 \sqsubseteq \gamma_2, \xi : \gamma_2 \sqsubseteq \gamma_3\}$. This kind of sets of expressions are what we are interested in. Assertions of the for $\xi_1 \preccurlyeq \xi_2$ provide a for alization of the notion

---

[6] It is worth stressing again that, in fact, a language $\mathcal{L}_i$, as defined here, is just a sub-language of languages of the type $\mathcal{ALC}$. As we will see later in this section, to represent contextual TBoxes the subsumption symbol is replaced by a set of contextualized subsumption symbols.

[7] It might be worth remarking that language $\mathcal{L}^{CT}$ is, then, an expansion of each $\mathcal{L}_i$ language.

of *generality* often touched upon in context theory (see for exa ple [6, 24]). In Section 4.1 the following sy bol will be also used " . : . ⊏ ." (within context c, concept A_1 is a proper subconcept of concept A_2). It can be defined as follows:

$$\xi : \gamma_1 \sqsubset \gamma_2 \; =_{def} \; \xi : \gamma_1 \sqsubseteq \gamma_2 \wedge \; \sim \xi : \gamma_2 \sqsubseteq \gamma_1.$$

A last category of expressions is also of interest, na ely expressions representing what a concept eans in a given context: for instance, recalling Exa ple 1, "the concept `vehicle` in context M1". These expressions, as it will be shown in Section 3.2, are particularly interesting fro a se antic point of view. Let us call the *contextual concept descriptions* and let us define their set $\mathcal{D}$ through the following BNF:

$$\delta ::= \xi : \gamma.$$

As we will see in Section 3.2, contextual concept descriptions $\mathcal{D}$ play an i portant role in the se antics of contextual subsu ption relations.

3.2 Semantics

In order to provide a se antics for $\mathcal{L}^{CT}$ languages, we will proceed as follows. First we will define a class of structures which can be used to provide a for al eaning to those languages. We will then characterize the class of operations and relations on contexts that will constitute the se antic counterpart of the operators and relation sy bols introduced in Section 3.1. Definitions of the for al eaning of our expressions will then follow.

Before pursuing this line, it is necessary to recollect the basic definition of a description logic odel for a language $\mathcal{L}_i$ ([23]).

Definition 1. (Models for $\mathcal{L}_i$'s)
A model m for a language $\mathcal{L}_i$ is defined as follows:

$$m = \langle \Delta_m, \mathcal{I}_m \rangle$$

where:

- Δ_m *is the (non empty) domain of the model;*
- $\mathcal{I}_m$ *is a function* $\mathcal{I}_m : \mathbf{A}_i \longrightarrow \mathcal{P}(\Delta_m)$*, that is, an interpretation of (atomic concepts expressions of)* $\mathcal{L}_i$ *on* Δ_m*. This interpretation is extended to complex concept constructs via the following inductive definition:*

$$\begin{aligned}
\mathcal{I}_m(\top) &= \Delta_m \\
\mathcal{I}_m(\bot) &= \emptyset \\
\mathcal{I}_m(\neg A) &= \Delta_m \setminus \mathcal{I}_m(A) \\
\mathcal{I}_m(A \sqcap B) &= \mathcal{I}_m(A) \cap \mathcal{I}_m(B) \\
\mathcal{I}_m(A \sqcup B) &= \mathcal{I}_m(A) \cup \mathcal{I}_m(B).
\end{aligned}$$

Out of technicalities, what a odel m for a language $\mathcal{L}_i$ does, is to assign a denotation to each ato ic concept (for instance the set of ele ents of Δ_m that instantiate the concept `bicycle`) and, accordingly, to each co plex concept (for instance the set of ele ents of Δ_m that instantiate the concept `vehicle` ⊓ ¬ `bicycle`).

3.3 Models for $\mathcal{L}^{CT}$

We can now define a notion of *contextual taxonomy model* (ct- odel) for languages $\mathcal{L}^{CT}$.

Definition 2. (ct-models)
A ct-model $\mathbb{M}$ *is a structure:*

$$\mathbb{M} = \langle \{\mathbf{M}_i\}_{0 \leq i \leq n}, \mathbb{I} \rangle$$

where:

- $\{\mathbf{M}_i\}_{0 \leq i \leq n}$ *is the family of the sets of models* $\mathbf{M}_i$ *of each language* $\mathcal{L}_i$*. That is,* $\forall m \in \mathbf{M}_i$*,* m *is a model for* $\mathcal{L}_i$*.*
- $\mathbb{I}$ *is a function* $\mathbb{I} : \mathbf{c} \longrightarrow \mathcal{P}(\mathbf{M}_0) \cup \ldots \cup \mathcal{P}(\mathbf{M}_n)$*. In other words, this function associates to each atomic context identifier in* $\mathbf{c}$ *a subset of the set of all models in some language* $\mathcal{L}_i$*:* $\mathbb{I}(c) = M$ *with* $M \subseteq \mathbf{M}_i$ *for some* i *s.t.* $0 \leq i \leq n$*. Function* $\mathbb{I}$ *can be seen as labeling sets of models on some language* i *via atomic context identifiers. Notice that* $\mathbb{I}$ *fixes, for each atomic context identifier, the language on which the context denoted by the identifier is specified. We could say that it is* $\mathbb{I}$ *itself which fixes a specific index for each atomic context identifier* c*.*
- $\forall m', m'' \in \bigcup_{0 \leq i \leq n} \mathbf{M}_i$*,* $\Delta_{m'} = \Delta_{m''}$*. That is, the domain of all models* m *is unique. We assume this constraint simply because we are interested in modeling different (taxonomical) conceptualizations of a same set of individuals.*

This can be clarified by eans of a si ple exa ple. Suppose the alphabet of $\mathcal{L}^{CT}$ to be the set of ato ic concepts $\{$`allowed`, `vehicle`, `car`, `bicycle`$\}$ and the set of ato ic context identifiers $\{c_{M1}, c_{M2}, c_R\}$. The nu ber of possible languages $\mathcal{L}_i$ given the four afore entioned concepts is obviously $2^4 - 1$. A ct- odel for this $\mathcal{L}^{CT}$ language would have as do ain the set of the sets of all odels for each of the $2^4 - 1$ $\mathcal{L}_i$ languages, and as interpretation a function $\mathbb{I}$ which assigns to each c_{M1}, c_{M2} and c_R a subset of an ele ent of that set, i.e., a set of odels for one of the $\mathcal{L}_i$ languages. We will co e back to this specific language in Section 4.1, where we discuss the for alization of the public park scenario.

The key feature of this se antics is that contexts are characterized as sets of odels for the sa e language. This perspective allows for straightforward odel theoretical definitions of operations on contexts.

3.4 Operations on Contexts

Before getting to this, let us first recall a notion of *domain restriction* ($\rceil$) of a function f w.r.t. a subset C of the do ain of f. Intuitively, a do ain restriction of a function f is nothing but the function $C \rceil f$ having C as do ain and s.t. for each ele ent of C, f and $C \rceil f$ return the sa e i age. The exact definition is the following one: $C \rceil f = \{\langle x, f(x) \rangle \mid x \in C\}$.

Definition 3. (Operations on contexts)
Let M' and M'' be sets of models:

$$\rceil_i M' = \{m \mid m = \langle \Delta_{m'}, \mathbf{A}_i \rceil \mathcal{I}_{m'} \rangle \ \& \ m' \in M'\} \tag{1}$$
$$M' \Cap_i M'' = \rceil_i M' \cap \rceil_i M'' \tag{2}$$
$$M' \Cup_i M'' = \rceil_i M' \cup \rceil_i M'' \tag{3}$$
$$-_i M' = \mathbf{M}_i \setminus \rceil_i M'. \tag{4}$$

Intuitively, the operations have the following meaning: operation 1 allows for abstracting the relevant content of a context with respect to a specific language; operations 2 and 3 express basic set-theoretical composition of contexts; finally, operation 4 returns, given a context, the most general of all the remaining contexts. Let us now provide some technical observations. First of all notice that operation $\rceil_i$ yields the empty context when it is applied to a context M' the language of which is not an elementary expansion of $\mathcal{L}_i$. This is indeed very intuitive: the context obtained via abstraction of the context "dinosaurs" on the language of, say, "botanics" should be empty. Empty contexts can be also obtained through the $\Cap_i$ operation. In that case the language is shared, but the two contexts simply do not have any interpretation in common. This happens, for example, when the members of two different football teams talk about their opponents: as a matter of fact, no interpretation of the concept `opponent` can be shared without jeopardizing the fairness of the match. The following propositions can be proved with respect to the operations on contexts.

Proposition 1. (Structure of contexts on a given language)
The structure of contexts $\langle \mathcal{P}(\mathbf{M}_i), \Cup_i, \Cap_i, -_i, \mathbf{M}_i, \emptyset \rangle$ on a language $\mathcal{L}_i$ is a Boolean Algebra.

Proof. The proof follows straightforwardly from Definition 3. ■

Proposition 2. (Abstraction operation on contexts)
Operation $\rceil_i$ is surjective and idempotent.

Proof. That $\rceil_i$ is surjective can be proved per absurdum. First notice that this operation is a function of the following type: $\rceil_i : \mathcal{P}(\mathbf{M}_0) \cup \ldots \cup \mathcal{P}(\mathbf{M}_n) \longrightarrow \mathcal{P}(\mathbf{M}_i)$ with $1 \leq i \leq n$. If it is not surjective then $\exists M'' \subseteq \mathbf{M}_i$ s.t. $\forall M'$ in the domain of $\rceil_i$, $\rceil_i M' \neq M''$. This means that $\forall M'$ in the domain of $\rceil_i$, $\{m \mid m = \langle \Delta_{m'}, \mathbf{A}_i \rceil \mathcal{I}_{m'} \rangle \ \& \ m' \in M'\} \neq M''$, which is impossible because we have at least that $\rceil_i M'' = M''$. The proof of the equation for idempotency $\rceil_i(\rceil_i M) = \rceil_i M$ is straightforward. ■

These propositions clarify the type of conception of context we hold here: contexts are sets of models on different taxonomical languages; on each language the set of possible contexts is structured in a boolean algebra; the operation of abstraction allows for shifting from richer to simpler languages and it is, as we would intuitively expect, idempotent (abstracting from an abstraction yields the same first abstraction) and surjective (every context, even the empty one, can be seen as an abstraction of a different richer context, in the most trivial case, an abstraction of itself).

3.5 Formal Meaning of $\boldsymbol{\Xi}$, $\boldsymbol{\mathcal{D}}$, and $\boldsymbol{\mathcal{A}}$

In Definition 2 ato ic contexts are interpreted as sets of odels on so e language $\mathcal{L}_i$ for $0 \leq i \leq n$: $\mathbb{I}(c) = M \in \mathcal{P}(\mathbf{M}_0) \cup \ldots \cup \mathcal{P}(\mathbf{M}_n)$. The se antics of contexts constructs Ξ can be defined via inductive extension of that definition.

Definition 4. (Semantics of contexts constructs)
The semantics of context constructors is defined as follows:

$$\begin{aligned}
\mathbb{I}(\bot_i) &= \emptyset \\
\mathbb{I}(\top_i) &= \mathbf{M}_i \\
\mathbb{I}(\xi_1 \curlywedge_i \xi_2) &= \mathbb{I}(\xi_1) \Cap_i \mathbb{I}(\xi_2) \\
\mathbb{I}(\xi_1 \curlyvee_i \xi_2) &= \mathbb{I}(\xi_1) \Cup_i \mathbb{I}(\xi_2) \\
\mathbb{I}(\neg_i\ \xi) &= -_i\mathbb{I}(\xi) \\
\mathbb{I}(abs_i\ \xi) &= \rceil_i\mathbb{I}(\xi).
\end{aligned}$$

The $\bot_i$ context is interpreted as the e pty context (the sa e on each language); the $\top_i$ context is interpreted as the greatest, or ost general, context on $\mathcal{L}_i$; the binary $\curlywedge_i$-co position of contexts is interpreted as the greatest lower bound of the restriction of the interpretations of the two contexts on $\mathcal{L}_i$; the binary $\curlyvee_i$-co position of contexts is interpreted as the lowest upper bound of the restriction of the interpretations of the two contexts on $\mathcal{L}_i$; context negation is interpreted as the co ple ent with respect to the ost general context on that language; finally, the unary abs_i operator is interpreted just as the restriction of the interpretation of its argu ent to language $\mathcal{L}_i$.

Se antics for the contextual concept description $\mathcal{D}$ and for the assertions $\mathcal{A}$ in $\mathcal{L}^{CT}$ is based on the function $\mathbb{I}$.

Definition 5. (Semantics of contextual concept descriptions: $||\ .\ ||_{\mathbb{M}}$)
The semantics of contextual concept descriptions is defined as follows:

$$||\xi : \gamma||_M = \{D \mid \langle \gamma, D \rangle \in \mathcal{I}_m \ \&\ m \in \mathbb{I}(\xi)\}.$$

The eaning of a concept γ in a context ξ is the set of denotations D attributed to that concept by the odels constituting that context.

It is worth noticing that if concept γ is not expressible in the language of context ξ, then $||\xi : \gamma||_{\mathbb{M}} = \emptyset$, that is, concept γ gets no denotation at all in context ξ. This happens si ply because concept γ does not belong to the do ain of functions $\mathcal{I}_m$, and there therefore exists no interpretation for that concept in the odels constituting ξ. This shows also how Definition 5 allows to capture the intuitive distinction between concepts which lack denotation ($||\xi : \gamma||_{\mathbb{M}} = \emptyset$), and concepts which have a denotation which is e pty ($||\xi : \gamma||_{\mathbb{M}} = \{\emptyset\}$): a concept that lacks denotation is for exa ple the concept `immigrant` in the context of public park access regulation; in the sa e context, a concept with e pty denotation is for exa ple the concept `car`$\sqcap\neg$`car`.

In what follows we will often use the notation $\mathbb{I}(\xi : \gamma)$ instead of the heavier $||\xi : \gamma||_{\mathbb{M}}$.

Definition 6. (Semantics of assertions: $\models$)
The semantics of assertions is defined as follows:

$$\begin{aligned}
\mathbb{M} \models \xi : \gamma_1 \sqsubseteq \gamma_2 \;\; & \textit{iff} \;\; \mathbb{I}(\xi : \gamma_1), \mathbb{I}(\xi : \gamma_2) \neq \emptyset \text{ and } \forall m \in \mathbb{I}(\xi),\ \mathcal{I}_m(\gamma_1) \subseteq \mathcal{I}_m(\gamma_2) \\
\mathbb{M} \models \xi_1 \preccurlyeq \xi_2 \;\; & \textit{iff} \;\; \mathbb{I}(\xi_1) \subseteq \mathbb{I}(\xi_2) \\
\mathbb{M} \models \sim \alpha \;\; & \textit{iff} \;\; \text{not } \mathbb{M} \models \alpha \\
\mathbb{M} \models \alpha_1 \wedge \alpha_2 \;\; & \textit{iff} \;\; \mathbb{M} \models \alpha_1 \text{ and } \mathbb{M} \models \alpha_2.
\end{aligned}$$

A contextual subsumption relation between γ_1 and γ_2 holds iff $\mathbb{I}(\xi)$ makes the meaning of γ_1 and γ_2 not empty and all models m of $\mathbb{I}(\xi)$ interpret γ_1 as a subconcept of γ_2. Note that this is precisely the clause for the validity of a subsumption relation in standard description logics, but together with the fact that the concepts involved are actually meaningful in that context. The $\preccurlyeq$ relation between context constructs is interpreted as a standard subset relation: $\xi_1 \preccurlyeq \xi_2$ means that context denoted by ξ_1 contains at most all the models that ξ_2 contains, that is to say, ξ_1 is *at most as general as* ξ_2. Note that this relation, being interpreted on the $\subseteq$ relation, is reflexive, antisymmetric and transitive. In [5] a generality ordering with similar properties was imposed on the set of context identifiers, and analogous properties for a similar relation have been singled out also in [11]. The interesting thing is that such an ordering is here emergent from the semantics. Note also that this relation holds only between contexts specified on the same language. Clauses for boolean connectives are the obvious ones.

The satisfaction clause of contextual subsumption relations deserves some more remarks. We observed that the satisfaction is conditioned to the meaningfulness of the terms involved with respect to the context. This condition is necessary because our contexts have different languages. Another way to deal with this would be to impose syntactic constraints on the formation of $\xi : \gamma_1 \sqsubseteq \gamma_2$ expressions, in order to distinguish the well-formed ones from the ill-formed ones. However, this would determine a dependence of the definition of well-formed expressions of $\mathcal{L}^{CT}$ on the models $\mathbb{M}$ of the language itself. Alternatively, the satisfaction relation itself might be restricted to consider only those subsumptions between concepts that, given the interpretation of the context, are interpreted as meaningful. Nevertheless, this option too determines a weird dependence, namely between the definition of the satisfaction relation and the models: the scope of the satisfaction would vary according to the models[8]. We chose for yet another solution, exploiting the possibility that our semantics enables of distinguishing meaningless concepts from concepts with empty extension (see Definition 5). By means of this feature it is possible to constrain the satisfaction of $\xi : \gamma_1 \sqsubseteq \gamma_2$ formulas, in such a way that, for them to be true, concepts γ_1 and γ_2 have

[8] Though in a completely different formal setting, this way is pursued in [21, 22].

to be eaningful in context ξ. Intuitively, we interpret contextual subsu ption relations as inherently presupposing the eaningfulness of their ter s.

4 Contextual Taxonomies, "Core" and "Penumbra"

4.1 Formalizing an Example

We are now able to provide a for alization of the si ple scenario introduced in Exa ple 1 based on the for al se antic achinery just exposed.

Example 2. **(The public park scenario formalized)** To for alize the public park scenario within our setting a language $\mathcal{L}^{CT}$ is needed, which contains the following ato ic concepts: `allowed`, `vehicle`, `car`, `bicycle`. Three ato ic contexts are at issue here: the context of the ain regulation R, let us call it c_R; the contexts of the unicipal regulations M1 and M2, let us call the c_{M1} and c_{M2} respectively. These contexts should be interpreted on two relevant languages. A language $\mathcal{L}_0$ for c_R s.t. $\mathbf{A}_0 = \{\mathtt{allowed}, \mathtt{vehicle}\}$; and a language $\mathcal{L}_1$ for c_{M1} and c_{M2} s.t. $\mathbf{A}_1 = \mathbf{A}_0 \cup \{\mathtt{car}, \mathtt{bicycle}\}$ (an abstract language concerning only vehicles and objects allowed to get into the park, and a ore concrete one concerning, besides this, also cars and bicycles). A for alization of the scenario by eans of $\mathcal{L}^{CT}$ for ulas is the following one:

$$abs_0(c_{M1}) \curlyvee_0 abs_0(c_{M2}) \preccurlyeq c_R \tag{5}$$

$$c_R : \mathtt{vehicle} \sqsubseteq \neg\mathtt{allowed} \tag{6}$$

$$c_{M1} \curlyvee_1 c_{M2} : \mathtt{car} \sqsubset \mathtt{vehicle} \tag{7}$$

$$c_{M1} : \mathtt{bicycle} \sqsubset \mathtt{vehicle} \tag{8}$$

$$c_{M2} : \mathtt{bicycle} \sqsubseteq \neg\mathtt{vehicle} \tag{9}$$

$$c_{M1} \curlyvee_1 c_{M2} : \mathtt{bicycle} \sqsubset \mathtt{vehicle} \sqcup \mathtt{allowed}. \tag{10}$$

For ula (5) plays a key role, stating that the two contexts c_{M1}, c_{M2} are concrete variants of context c_R. It tells this by saying that the context obtained by joining the two concrete contexts on language $\mathcal{L}_0$ (the language of c_R) is at ost as general as context c_R. As we will see in discussing the logical consequences of this set of for ulas, for ula (5) akes c_{M1}, c_{M2} inherit what holds in c_R. For ula (6) for alizes the abstract rule to the effect that vehicles belong to the category of objects not allowed to access public parks. For ula (7) states that in both contexts cars count as vehicles. For ulas (8) and (9) state the two different conceptualizations of the concept of bicycle holding in the two concrete contexts at issue. These for ulas show where the two contextual taxono ies diverge. For ula (10), finally, tells that bicycles either are vehicles or should be allowed in the park. Indeed, it ight be seen as a clause avoiding "cheating" classifications such as: "bicycles counts as cars".

It is worth listing and discussing so e straightforward logical consequences of the for alization.

$$(5),(6) \models c_{M1} : \texttt{vehicle} \sqsubseteq \neg\texttt{allowed}$$
$$(5),(6),(7) \models c_{M1} : \texttt{car} \sqsubset \neg\texttt{allowed}$$
$$(5),(6),(8) \models c_{M1} : \texttt{bicycle} \sqsubset \neg\texttt{allowed}$$

$$(5),(6) \models c_{M2} : \texttt{vehicle} \sqsubseteq \neg\texttt{allowed}$$
$$(5),(6),(7) \models c_{M2} : \texttt{car} \sqsubset \neg\texttt{allowed}$$
$$(5),(6),(9),(10) \models c_{M2} : \texttt{bicycle} \sqsubseteq \texttt{allowed}$$

These are indeed the for ulas that we would intuitively expect to hold in our scenario. The list displays two sets of for ulas grouped on the basis of the context to which they pertain. They for alize the two contextual taxono ies at hands in our scenario. Let us have a closer look. The first consequence of each group results fro the generality relation expressed in (5), by eans of which the content of (6) is shown to hold also in the two concrete contexts: in si ple words, contexts c_{M1} and c_{M2} inherit the general rule stating that vehicles are not allowed to access public parks. Via this inherited rule, and via (7), it is shown that, in all concrete contexts, cars are also not allowed to access the park. As to cars then, all contexts agree. Where differences arise is in relation with how the concept of bicycle is handled. In context c_{M1}, since bicycles count as vehicles (8), bicycles are also not allowed. In context c_{M2}, instead, bicycles constitute an allowed class because they are not considered to be vehicles (9) and there is no bicycle which does not count as a vehicle and which does not belong to that class of allowed objects (10). In the following section we show in so e ore detail how a odel for the for alization just exposed looks like.

4.2 A Model of the Formalization

For ulas (5)-(10) constrain ct- odels in the following way:

$$\rceil_0\mathbb{I}(c_{M1}) \cup \rceil_0\mathbb{I}(c_{M2}) \subseteq \mathbb{I}(c_R)$$
$$\forall m \in \mathbb{I}(c_R),\ \mathcal{I}_m(\texttt{vehicle}) \subseteq \Delta_1 \setminus \mathcal{I}_m(\texttt{allowed})$$
$$\mathbb{I}(c_R : \texttt{vehicle}), \mathbb{I}(c_R : \texttt{allowed}) \neq \emptyset$$
$$\forall m \in \mathbb{I}(c_{M1}) \cup \mathbb{I}(c_{M2}),\ \mathcal{I}_m(\texttt{car}) \subset \mathcal{I}_m(\texttt{vehicle})$$
$$\mathbb{I}(c_{M1} \curlyvee_1 c_{M2} : \texttt{car}), \mathbb{I}(c_{M1} \curlyvee_1 c_{M2} : \texttt{vehicle}) \neq \emptyset$$
$$\forall m \in \mathbb{I}(c_{M1}),\ \mathcal{I}_m(\texttt{bicycle}) \subset \mathcal{I}_m(\texttt{vehicle})$$
$$\mathbb{I}(c_{M1} : \texttt{bicycle}), \mathbb{I}(c_{M1} : \texttt{vehicle}) \neq \emptyset$$
$$\forall m \in \mathbb{I}(c_{M2}),\ \mathcal{I}_m(\texttt{bicycle}) \subseteq \Delta_1 \setminus \mathcal{I}_m(\texttt{vehicle})$$
$$\mathbb{I}(c_{M2} : \texttt{bicycle}), \mathbb{I}(c_{M2} : \texttt{vehicle}) \neq \emptyset$$
$$\forall m \in \mathbb{I}(c_{M1}) \cup \mathbb{I}(c_{M2}),\ \mathcal{I}_m(\texttt{bicycle}) \subseteq \mathcal{I}_m(\texttt{vehicle}) \cup \mathcal{I}_m(\texttt{allowed})$$
$$\mathbb{I}(c_{M1} \curlyvee_1 c_{M2} : \texttt{bicycle}), \mathbb{I}(c_{M1} \curlyvee_1 c_{M2} : \texttt{allowed}) \neq \emptyset.$$

Besides the ones above, a odel of the scenario can be thought of requiring two ore constraints. Although the for al language as it is defined in 3.1 cannot express the , we show that they can be perfectly captured at a se antic level and therefore that new appropriate sy bols ight be accordingly added to the syntax.

- $\mathbb{I}(c_{M1} : \texttt{bicycle}) = \mathbb{I}(c_{M2} : \texttt{bicycle}) = \{\{a, b\}\}$[9] ($c_{M1}$ and c_{M2} agree on the interpretation of `bicycle`, say, the set of objects $\{a, b\}$);
- $\mathbb{I}(c_{M1} : \texttt{car}) = \mathbb{I}(c_{M2} : \texttt{car}) = \{\{c\}\}$[10] ($c_{M1}$ and c_{M2} agree on the interpretation of `car`, say, the singleton $\{c\}$).

Let us stipulate that the odels m that will constitute our interpretation of contexts identifiers consist of a do ain $\Delta_m = \{a, b, c, d\}$ and let us call the sets of all odels for $\mathcal{L}_0$ and $\mathcal{L}_1$ on this do ain respectively $\mathbf{M}_0$ and $\mathbf{M}_1$. Given the restrictions, a ct- odel of the scenario can consist then of the do ain $\mathbf{M}_0 \cup \mathbf{M}_1$ and of the function $\mathbb{I}$ s.t.:

- $\mathbb{I}(c_{M1}) = \{m_1, m_2\} \subseteq \mathbf{M}_1$ s.t. $\mathcal{I}_{m_1}(\texttt{allowed}) = \{d\}$, $\mathcal{I}_{m_1}(\texttt{vehicle}) = \{a, b, c\}$, $\mathcal{I}_{m_1}(\texttt{bicycle}) = \{a, b\}$, $\mathcal{I}_{m_1}(\texttt{car}) = \{c\}$ and $\mathcal{I}_{m_2}(\texttt{allowed}) = \emptyset$, $\mathcal{I}_{m_2}(\texttt{vehicle}) = \{a, b, c, d\}$, $\mathcal{I}_{m_2}(\texttt{bicycle}) = \{a, b\}$, $\mathcal{I}_{m_2}(\texttt{car}) = \{c\}$.
  In c_{M1} concepts `allowed` and `vehicle` are interpreted in two possible ways; notice that odel m_2 akes no object allowed to access the park;
- $\mathbb{I}(c_{M2}) = \{m_3\} \subseteq \mathbf{M}_1$ s.t. $\mathcal{I}_{m_3}(\texttt{allowed}) = \{a, b\}$, $\mathcal{I}_{m_3}(\texttt{vehicle}) = \{c, d\}$, $\mathcal{I}_{m_3}(\texttt{car}) = \{c\}$, $\mathcal{I}_{m_3}(\texttt{bicycle}) = \{a, b\}$.
  In c_{M2}, which is constituted by a single odel, the concept `vehicle` strictly contains `car`, and excludes `bicycle`. Notice also that `bicycle` coincide with `allowed`.
- $\mathbb{I}(c_R) = \{m \mid \mathcal{I}_m(\texttt{vehicle}) \subseteq \Delta_1 \setminus \mathcal{I}_m(\texttt{allowed})\}$.
  In c_R, concepts `vehicle` and `allowed` get all possible interpretations that keep the disjoint.

We can now get to the ain for al characterizations at which we have been ai ing in this work.

4.3 Representing Conceptual "Core" and "Penumbra"

What is the part of a denotation of a concept which re ains context independent? What is the part which varies instead? "Core" and "penu bral" eaning are for alized in the two following definitions.

Definition 7. **($\mathfrak{Core}(\gamma, \xi_1, \xi_2)$)**
The "core meaning" of concept γ w.r.t. contexts ξ_1, ξ_2 on language $\mathcal{L}_i$ is defined as:

$$\mathfrak{Core}(\gamma, \xi_1, \xi_2) =_{def} \bigcap(\mathbb{I}(\xi_1 : \gamma) \cup \mathbb{I}(\xi_2 : \gamma)).$$

[9] It might be worth recalling that the meaning of a concept in a context is a set of denotations, which we assume to be here, for the sake of simplicity (and in accordance with our intuitions about the scenario), a singleton.

[10] See previous footnote.

Intuitively, the definition takes just the conjunction of the union of the interpretations of γ in the two contexts. Referring back to Exa ple 2, we have that $\mathfrak{Core}(\texttt{vehicle}, c_{M1}, c_{M2}) = \{c\}$, that is, the core of the concept `vehicle` coincides, in those contexts, with the denotation of the concept `car`. The notion of "penu bra" is now easily definable.

Definition 8. **($\mathfrak{Penumbra}(\gamma, \xi_1, \xi_2)$)**
The "penumbra" of concept γ w.r.t. contexts ξ_1, ξ_2 on language $\mathcal{L}_i$ is defined as:

$$\mathfrak{Penumbra}(\gamma, \xi_1, \xi_2) =_{def} \bigcup((\mathbb{I}(\xi_1 : \gamma) \cup \mathbb{I}(\xi_2 : \gamma)) \setminus \mathfrak{Core}(\gamma, \xi_1, \xi_2)).$$

A "penu bral eaning" is then nothing else but the set of individuals on which the contextual interpretation of the concept varies. Referring back again to Exa ple 2: $\mathfrak{Penumbra}(\texttt{vehicle}, c_{M1}, c_{M2}) = \{a, b, d\}$, that is to say, the penu bra of the concept `vehicle` ranges over those individuals that are not instances of the core of `vehicle`, i.e., the concept `car`. Notice that the definitions are straightforwardly generalizable to for ulations with ore than two contexts.

5 Related Work

We already showed, in Section 2, how the present proposal relates to work developed in the area of logical odeling of the notion of context. Contexts have been used here in order to propose a different approach to vagueness (especially as it appears in the nor ative do ain). In this section so e words will be spent in order to put the present proposal in perspective with respect to so e ore standard approaches to vagueness, na ely approaches aking use of fuzzy sets ([25]) or rough sets ([26]).

The ost characteristic feature of our approach, with respect to fuzzy or rough set theories, consists in considering vagueness as an inherently se antic pheno enon. Vagueness arises fro the referring of a language to structures odeling reality, and not fro those structures the selves. That is to say, the truth denotation of a predicate is, in our approach, always definite and crisp, even if ultiple. Consequently, no degree of e bership is considered, as in fuzzy logic, and no representation of sets in ter s of approxi ations is used, as in rough set theory. Let us use a si ple exa ple in order to ake this distinction evident. Consider the vague onadic predicate or, to use a description logic ter inology, the concept `tall_person`. Fuzzy approaches would deter ine the denotation of this predicate as a fuzzy set, i.e., as the set of ele ents with e bership degree contained in the interval $]0, 1]$. Standard rough set theory approaches would characterize this denotation not directly, but on the basis of a given partition of the universe (the set of all individuals) and a lower and upper approxi ation provided in ter s of that partition. For instance, a trivial partition ight be the one consisting of the following three concepts: `tall>2m`, `1.60m`$\leq$`tall`$\leq$`2m`, `tall<1.60m`. Concept `tall_person` would then be approxi ated by eans of the lower approxi ation `tall>2m` (the ele ents of a set that are definitely also e bers of the to be approxi ated set), and the upper approxi ation $\texttt{1.60m}\leq\texttt{tall}\leq\texttt{2m} \sqcup \texttt{tall>2m}$ (the ele ents of a set that ay be also e bers of the to be approxi ated set). In this rough set representation, set $\texttt{1.60m}\leq\texttt{tall}\leq\texttt{2m}$ constitutes the so called *boundary* of `tall_person`. Within our approach instead, the set `tall_person` can be represented crisply and without approxi ations. The key feature is that `tall_person` obtains ultiple crisp interpretations, at least one for each context: in the context of dutch standards, concept `tall_person` does not subsu e concept $\texttt{1.60m}\leq\texttt{tall}\leq\texttt{2m}$, whereas it does in the context of pyg y standards. According to our approach, vagueness resides then in the contextual nature of interpretation rather than in the concepts the selves[11].

It is nevertheless easy to spot so e si ilarities, in particular with respect to rough set theory. The notions of "core" and "penu bra" have uch in co on with the notions of, respectively, *lower approximation* and *boundary* developed in rough set theory: each of these pairs of notions denotes what is always, and respectively, in so e cases, an instance of a given concept. But the characterization of the last pair is based on a partition of the universe denoting the equivalence classes i posed by a set of given known properties. The notions of "core" and "penu bra", instead, are yielded by the consideration of any contextual interpretations of the concept itself. With respect to fuzzy approaches, notice that sets $\mathfrak{Core}$ can be viewed exactly as the sets of instances having a e bership degree equal to one, while sets $\mathfrak{Penumbra}$ can be viewed as the sets of instances with degree of e bership between zero and one. Besides, sets $\mathfrak{Penumbra}$ could be partitioned in sets X_n each containing instances that occur in a fixed nu ber n of odels constituting the "penu bra", thus deter ining a total and, notice,

---

[11] A clear position for our thesis can also be found within those analyses of vagueness, developed in the area of philosophical logic, which distinguish between *de re* and *de dicto* views of vagueness ([27]), the first holding that referents themselves are vague and therefore that vagueness constitutes something objective, whereas the second holding that it is the way referents are established that determines vagueness. Fuzzy set approaches lie within a *de re* conception of vagueness, while our approach is grounded on the alternative *de dicto* view (rough sets approaches have instead more to do with insufficient information issues). In philosophical logic, a formal theory has been developed which formalizes this *de dicto* approach to vagueness, the so called *superevaluationism* ([28]). On this view, when interpreting vague terms, we consider the many possible ways in which those terms can be interpreted:

> "Whatever it is that we do to determine the 'intended' interpretation of our language determines not one interpretation but a range of interpretations. (The range depends on context [...])"([29]).

As it is evident from Section 3.2, this intuition backs also our semantics. What our approach adds to formal accounts of *superevaluationism* such as [28, 30] consists in the explicit use of contexts as specific formal objects clustering the possible ways terms can be interpreted: contexts are precisely the range of admissible interpretations of the concepts at issue.

discrete ordering on e bership: instances occurring in only one odel in the "penu bra" will belong to the denotation of the concept at the ini u degree of e bership, while instances occurring in the "core" at the axi u one.

Another relevant feature of our proposal, which we dee worth stressing, consists in the use of a frag ent of predicate logic. This allows, first of all, the intra-contextual reasoning to be classical. Further ore, the use of description logic, even if not yet fully elaborate in this work, allows for its well known interesting co putability properties to be enabled at the intra-contextual reasoning level, thus aking the fra ework appealing also in this respect.

6 Conclusions

Our ai was to account for a notion of contextual taxono y, and by eans of that, to rigorously characterize the notions of "core" and "penu bra" of a concept, that is to say, to define what is invariant and what is instead context dependent in the eaning of a concept. We did this contextualizing of a standard description logic notion of taxono y by eans of a for al se antics approach to contexts which provides also an account of a variety of for s of contexts interactions.

There are a nu ber of issues which would be worth investigating in future work. First of all, it would be of definite interest to provide for al rigorous co parisons of our fra ework with:

- Related work in the area of context logics, like especially the *local model semantics* proposed in [16] to which we referred in Section 2.
- Related work in the area of fuzzy or rough sets treat ent of conceptual a biguities ([26, 25]), which have been infor ally touched upon in Section 5.
- Related work in the area of logic for nor ative syste s specification, and in particular [31] where a odal logic se antics is used to account for expressions such as "A counts as B in context (institution) s". To this ai , we plan to apply the notion of contextual subsu ption relation to odal logic se antics in order to contextualize accessibility relations. For exa ple, it would be interesting to investigate applications to dyna ic logic se antics in order to provide a for al account of the contextual eaning of actions: raising a hand in the context of a bidding eans so ething different than raising a hand in the context of a scientific workshop. So e results on this issue have been presented in [32].

Secondly, we would like to enrich the expressivity of our fra ework considering richer description logic languages ad itting also attributes (or roles) constructs. This would allow for a for al characterization of "*contextual terminologies*" in general, enabling the full expressive power description logics are able to provide. A first step along this line has been proposed in [33].

Acknowledgments

We would like to thank the anony ous reviewers of CLIMA V. Thanks to their co ents, this work has been considerably i proved.

References

1. Dignum, F.: Agents, markets, institutions, and protocols. In: Agent Mediated Electronic Commerce, The European AgentLink Perspective., Springer-Verlag (2001) 98–114
2. Vázquez-Salceda, J., Dignum, F.: Modelling electronic organizations. In V. Marik, J.M., Pechoucek, M., eds.: Proceedings CEEMAS'03. LNAI 2691, Berlin, Springer-Verlag (2003) 584–593
3. Dignum, F.: Abstract norms and electronic institutions. In: Proceedings of the International Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA '02), Bologna. (2002) 93–104
4. Vázquez-Salceda, J.: The role of Norms and Electronic Institutions in Multi-Agent Systems. Birkhuser Verlag AG (2004)
5. Grossi, D., Dignum, F.: From abstract to concrete norms in agent institutions. In Hinchey, M.G., Rash, J.L., Truszkowski, W.F., et al., eds.: Formal Approaches to Agent-Based Systems: Third International Workshop, FAABS 2004. Lecture Notes in Computer Science, Springer-Verlag (2004) 12–29
6. McCarthy, J.: Notes on formalizing contexts. In Kehler, T., Rosenschein, S., eds.: Proceedings of the Fifth National Conference on Artificial Intelligence, Los Altos, California, Morgan Kaufmann (1986) 555–560
7. Akman, V., Surav., M.: Steps toward formalizing context. AI Magazine **17** (1996) 55–72
8. Benerecetti, M., Bouquet, P., Ghidini, C.: Contextual reasoning distilled. Journal of Experimental and Theoretical Artificial Intelligence (JETAI) **12** (2000) 279–305
9. Searle, J.: The Construction of Social Reality. Free Press (1995)
10. Hage, J., Verheij, B.: The law as a dynamic interconnected system of states of affairs. IJHCS: International Journal of Human-Computer Studies **51** (1999) 1043–1077
11. Goldman, A.I.: A Theory of Human Action. Princeton University Press, Princeton (1976)
12. Hart, H.L.A.: The Concept of Law. Clarendon Press, Oxford (1961)
13. Hart, H.L.A.: Positivism and the separation of law and morality. Harvard Law Review **71** (1958) 593–629
14. Prakken, H.: Logical Tools for Modelling Legal Arguments. Kluwer (1997)
15. Royakkers, L., Dignum, F.: Defeasible reasoning with legal rules. In Nute, D., ed.: Defeasible Deontic Logic, Dordrecht, Kluwer (1997) 263–286
16. Ghidini, C., Giunchiglia, F.: Local models semantics, or contextual reasoning = locality + compatibility. Artificial Intelligence **127** (2001) 221–259
17. Parsons, S., Jennings, N.J., Sabater, J., Sierra, C.: Agent specification using multi-context systems. Foundations and Applications of Multi-Agent Systems (2002)
18. Casali, A., Godo, L., Sierra, C.: Graded bdi models for agent architectures. In this volume.
19. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics or: How we can do without modal logics. Artificial Intelligence **65** (1994) 29–70

20. Giunchiglia, F.: Contextual reasoning. Epistemologia, special issue on I Linguaggi e le Macchine **16** (1993) 345–364
21. Buvač, S.V., Mason, I.A.: Propositional logic of context. Proceedings AAAI'93 (1993) 412–419
22. Buvač, S., Buvač, S.V., Mason, I.A.: The semantics of propositional contexts. Proceedings of the eight ISMIS. LNAI-869 (1994) 468–477
23. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook. Cambridge University Press, Cambridge (2002)
24. McCarthy, J.: Generality in artificial intelligence. Communications of the ACM **30** (1987) 1030–1035
25. Wygralak, M.: Vaguely Defined Objects. Kluwer Academic Publishers (1996)
26. Lin, T.Y., Cercone, N.: Rough Sets and Data Mining. Analysis of Imprecise Data. Kluwer Academic Publishers (1997)
27. Varzi, A.: Vague names for sharp objects. In Obrst, L., Mani, I., eds.: Proceedings of the Workshop on Semantic Approximation, Granularity, and Vagueness, AAAI Press (2000) 73–78
28. van Fraassen, B.C.: Singular terms, truth-value gaps, and free logic. Journal of Philosophy **63** (1966) 481–495
29. Lewis, D.: Many, but almost one. In: Papers in Metaphysics and Epistemology, Cambridge University Press (1999) 164–182
30. Fine, K.: Vagueness, truth and logic. Synthese **30** (1975) 265–300
31. Jones, A.J.I., Sergot, M.: A formal characterization of institutionalised power. Journal of the IGPL **3** (1996) 429–445
32. Grossi, D., Meyer, J-J. Ch., Dignum, F.: Modal logic investigations in the semantics of counts-as. (Submitted)
33. Grossi, D., Aldewereld, H., Vázquez-Salceda, J., Dignum, F.: Ontological aspects of the implementation of norms in agent-based electronic institutions. To be presented at NorMAS (2005)

From Logic Programs Updates to Action Description Updates*

José Júlio Alferes[1], Federico Banti[1], and Antonio Brogi[2]

[1] CENTRIA, Universidade Nova de Lisboa, Portugal
{jja, banti}@di.fct.unl.pt
[2] Dipartimento di Informatica, Università di Pisa, Italy
brogi@di.unipi.it

Abstract. An important branch of investigation in the field of agents has been the definition of high level languages for representing effects of actions, the programs written in such languages being usually called action programs. Logic programming is an important area in the field of knowledge representation and some languages for specifying updates of Logic Programs had been defined. Starting from the update language Evolp, in this work we propose a new paradigm for reasoning about actions called Evolp action programs.

We provide translations of some of the most known action description languages into Evolp action programs, and underline some peculiar features of this newly defined paradigm. One such feature is that Evolp action programs can easily express changes in the rules of the domains, including rules describing changes.

1 Introduction

In the last years the concept of agent has beco e central in the field of Artificial Intelligence. "*An agent is just something that acts*" [26]. Given the i portance of the concept, ways of representing actions and their effects on the environ ent have been studied. A branch of investigation in this topic has been the definition of high level languages for representing effects of actions [7, 12, 14, 15], the progra s written in such languages being usually called *action programs*. Action progra s specify which facts (or fluents) change in the environ ent after the execution of a set of actions. Several works exist on the relation between these action languages and Logic Progra ing (LP) (e.g. [5, 12, 21]). However, despite the fact that LP has been successfully used as a language for declaratively representing knowledge, the entioned works basically use LP for providing an operational se antics, and i ple entation, for action progra s. This is so because nor al logic progra s, and ost of their extensions, have no in-built

* This work was partially supported by project FLUX (POSI/40958/SRI/2001), and by the European Commission within the 6th Framework Programme project REWERSE number 506779 (cf. http://rewerse.net).

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 52–77, 2005.

means for dealing with changes, something that is quite fundamental for action languages.

In recent years some effort was devoted to explore and study the problem of how to update logic programs with new rules [3, 8, 10, 19, 20, 17]. Here, knowledge is conveyed by sequences of programs, where each program in a sequence is an update to the previous ones. For determining the meaning of sequences of logic programs, rules from previous programs are assumed to hold by inertia after the updates (given by subsequent programs) unless rejected by some later rule. LP update languages [2, 4, 9, 19], besides giving meaning to sequences of logic programs, also provide in-built mechanisms for constructing such sequences. In other words, LP update languages extend LP by providing means to specify and reason about rule updates.

In [5] the authors show, by examples, a possible use the LP update language LUPS [4] for representing effects of actions providing a hint for the possibility of using LP updates languages as an action description paradigm. However, the work done does not provide a clear view on how to use LP updates for representing actions, nor does it establishes an exact relationship between this new possibility and existing action languages. Thus, the eventual advantages of the LP update languages approach to actions are still not clear.

The present work tries to clarify these points. This is done by establishing a formal relationship between one LP update language, namely the Evolp language [2], and existing action languages, and by clarifying how to use this language for representing actions in general.

Our investigation starts by, on top of Evolp, defining a new action description language, called Evolp Action Programs (EAPs), as a macro language for Evolp. Before developing a complete framework for action description based on LP updates, in this work we focus on the basic problem in the field, i.e. the prediction of the possible future states of the world given a complete knowledge of the current state and the action performed. Our purpose is to check, already at this stage, the potentiality of an action description language based on the Evolp paradigm.

We then illustrate the usage of EAPs by an example involving a variant of the classical Yale Shooting Problem. An important point to clarify is the comparison of the expressive capabilities of the newly defined language with that of the existing paradigms. We consider the action languages $\mathcal{A}$ [12], $\mathcal{B}$ [13] (which is a subset of the language proposed in [14]), and (the definite fragment of) $\mathcal{C}$ [15]. We provides simple translations of such languages into EAPs, hence proving that EAPs are *at least as expressive* as the cited action languages.

Coming to this point, the next natural question is what are the possible advantages of EAPs. The underlying idea of action frameworks is to describe dynamic environments. This is usually done by describing rules that specify, given a set of external actions, how the environment evolves. In a dynamic environment, however, not only the facts but also the "rules of the game" can change, in particular *the rules describing the changes*. The capability of describing such kind of *meta level changes* is, in our opinion, an important feature of an action description language. This capability can be seen as an instance of *elaboration tolerance* i.e. *"the ability to accept changes to a person's or a computer's representation of facts about a subject without having to start all over"* [25]. In [15] this capability is seen as a central point in the action descriptions field and the problem is addressed in the context of the $\mathcal{C}$ language. The final words of [15] are *"Finding ways to further increase the degree of elaboration tolerance of languages for describing actions is a topic of future work"*. We address this topic in the context of EAPs and show EAPs seem, in this sense, more flexible than other paradigms. Evolp provides specific commands that allow for the specification of updates to the initial program, but also provides the possibility to specify updates of these updates commands. We show, by successive elaborations of the Yale shooting problem, how to use this feature to describe updates of the problem that come along with the evolution of the environment.

The rest of the paper is structured as follows. In section 2 we review some background and notation. In section 3 we define the syntax and semantics of Evolp action programs, and we illustrate the usage of EAPs by an example involving a variant of the classical Yale Shooting Problem. In section 4 we establish the relationship between EAPs and the languages $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$. In section 5 we discuss the possibility of updating the EAPs, and provide an example of such feature. Finally, in section 6, we conclude and trace a route for future developments. To facilitate the reading, and given that some of the results have proofs of some length, instead of presenting proofs along with the text, we expose them all in appendix A.

2 Background and Notation

In this section we briefly recall syntax and semantics of *Dynamic Logic Programs* [1], and the syntax and semantics for Evolp [2]. We also recall some basic notions and notation for action description languages. For a more detailed background on action languages see e.g. [12].

2.1 Dynamic Logic Programs and Evolp

The main idea of logic programs updates is to update a logic program by another logic program or by a *sequence* of logic programs, also called *Dynamic Logic Programs* (DLPs). The initial program of a DLP corresponds to the initial knowledge of a given (dynamic) domain, and the subsequent ones to successive updates of the domain. To represent negative information in logic programs and their updates, following [3] we allow for default negation *not* A not only in the premises of rules but also in their heads i.e., we use generalized logic programs (GLPs) [22].

A language $\mathcal{L}$ is any set of propositional atoms. A literal in $\mathcal{L}$ is either an atom of $\mathcal{L}$ or the negation of an atom. In general, given any set of atoms $\mathcal{F}$, *we denote by* $\mathcal{F}_L$ *the set of literals over* $\mathcal{F}$. Given a literal F, if $F = Q$, where Q is an atom, by *not* F we denote the negative literal *not* Q. Viceversa, if $F =$ *not* Q, by *not* F we denote the ato Q. A GLP defined over a propositional language $\mathcal{L}$ is a set of rules of the for $F \leftarrow Body$, where F is a literal in $\mathcal{L}$, and $Body$ is a *set* of literals in $\mathcal{L}$.[1] An *interpretation* I over a language $\mathcal{L}$ is any set of literals in $\mathcal{L}$ such that, for each ato A, either $A \in I$ or $not\ A \in I$. We say a set of literals $Body$ is true in an interpretation I (or that I satisfies $Body$) iff $Body \subseteq I$. In this paper we will use progra s containing variables. As usual when progra ing within the stable odels se antics, a progra with variables stands for the propositional progra obtained as the set of all possible ground instantiations of the rules.

Two rules τ and η are *conflicting* (denoted by $\tau \bowtie \eta$) iff the head of τ is the ato A and the head of η is *not* A, or viceversa. A Dyna ic Logic Progra over a language $\mathcal{L}$ is a sequence $P_1 \oplus \ldots \oplus P_m$ (also denoted $\oplus P_i^m$) where the P_is are GLPs defined over $\mathcal{L}$. The *refined stable model semantics* of such a DLP, defined in [1], assigns to each sequence $P_1 \oplus \ldots \oplus P_n$ a set of stable odels (that is proven there to coincide with the stable odels se antics when the sequence is for ed by a single nor al [11] or generalized progra [22]). The rationale for the definition of a stable odel M of a DLP is ade in accordance with the *causal rejection principle* [10, 19]: If the body of a rule in a given update is true in M, then that rule rejects all rules in previous updates that are conflicting with it. Such rejected rules are ignored in the co putation of the stable odel. In the refined se antics for DLPs a rule ay also reject conflicting rules that belong to the sa e update. For ally, the set of rejected rules of a DLP $\oplus P_i^m$ given an interpretation M is:

$$Rej^S(\oplus P_i^m, M) = \{\tau \mid \tau \in P_i : \ \exists\, \eta \in P_j \ i \le j, \ \tau \bowtie \eta \ \wedge \ Body(\eta) \subseteq M\}$$

Moreover, an ato A is assu ed false by default if there is no rule, in none of the progra s in the DLP, with head A and a true body in the interpretation M. For ally:

$$Default(\oplus P_i^m, M) = \{not\ A \mid \ \nexists\ A \leftarrow Body \in \bigcup P_i \ \ \wedge Body \subseteq M\}$$

If $\oplus P_i^m$ is clear fro the context, we o it it as first argu ent of the above functions.

Definition 1. *Let $\oplus P_i^m$ be a DLP over language $\mathcal{L}$ and M an interpretation. M is a refined stable model of $\oplus P_i^m$ iff*

$$M = least\left(\left(\bigcup_i P_i \setminus Rej^S(M)\right) \cup Default(M)\right)$$

where $least(P)$ denotes the least Herbrand model of the definite program [23] obtained by considering each negative literal not A in P as a new atom.

---

[1] Note that, by defining rule bodies as sets, the order and number of occurrences of literals do not matter.

Having defined the meaning of sequences of programs, we are left with the problem of how to come up with those sequences. This is the subject of LP update languages [2, 4, 9, 19]. Among the existing languages, Evolp [2] uses a particulary simple syntax, which extends the usual syntax of GLPs by introducing the special predicate $assert/1$. Given any language $\mathcal{L}$, the language $\mathcal{L}_{assert}$ is recursively defined as follows: every atom in $\mathcal{L}$ is also in $\mathcal{L}_{assert}$; for any rule τ over $\mathcal{L}_{assert}$, the atom $assert(\tau)$ is in $\mathcal{L}_{assert}$; nothing else is in $\mathcal{L}_{assert}$. An *Evolp program* over $\mathcal{L}$ is any GLP over $\mathcal{L}_{assert}$. An *Evolp sequence* is a sequence (or DLP) of Evolp programs. The rules of an Evolp program are called *Evolp rules*.

Intuitively an expression $assert(\tau)$ stands for "update the program with the rule τ". Notice the possibility in the language to nest an assert expression in another. The intuition behind the Evolp semantics is quite simple. Starting from the initial Evolp sequence $\oplus P_i^m$ we compute the set, $\mathcal{SM}(\oplus P_i^m)$, of the stable models of $\oplus P_i^m$. Then, for any element M in $\mathcal{SM}(\oplus P_i^m)$, we update the initial sequence with the program P_{m+1} consisting of the set of rules τ such that the atom $assert(\tau)$ belongs to M. In this way we obtain the sequence $\oplus P_i^m \oplus P_{m+1}$. Since $\mathcal{SM}(\oplus P_i^m)$ contains, in general, several models we may have different lines of evolution. The process continues by obtaining the various $\mathcal{SM}(\oplus P_i^{m+1})$ and, with them, various $\oplus P_i^{m+2}$. Intuitively, the program starts at step 1 already containing the sequence $\oplus P_i^m$. Then it updates itself with the rules asserted at step 1, thus obtaining step 2. Then, again, it updates itself with the rules asserted at this step, and so on. The evolution of any Evolp sequence can also be influenced by external events. An external event is itself an Evolp program. If, at a given step n, the programs receives the external update E_n, the rules in E_n are added to the last self update for the purpose of computing the stable models determining the next evolution but, in the successive step $n+1$ they are no longer considered (that's why they are called *events*). Formally:

Definition 2. *Let n and m be natural numbers. An* evolution interpretation *of length n, of an evolving logic program $\oplus P_i^m$ is any finite sequence $\mathcal{M} = M_1, \ldots, M_n$ of interpretations over $\mathcal{L}_{assert}$. The* evolution trace *associated with $\mathcal{M}$ and $\oplus P_i^m$ is the sequence $P_1 \oplus \ldots P_m \oplus P_{m+1} \ldots \oplus P_{m+n-1}$, where, for $1 \leq i < n$*

$$P_{m+i} = \{\tau \mid assert(\tau) \in M_{m+i-1}\}$$

Definition 3 (Evolving stable models). *Let $\oplus P_i^m$ and $\oplus E_i^n$ be any Evolp sequences, and $\mathcal{M} = M_1, \ldots, M_n$ be an evolving interpretation of length n. Let $P_1 \oplus \ldots \oplus P_{m+n-1}$ be the evolution trace associated with $\mathcal{M}$ and $\oplus P_i^m$. We say that $\mathcal{M}$ is an evolving stable model of $\oplus P_i^m$ with event sequence $\oplus E_i^n$ at step n iff M_k is a refined stable model of the program $P_1 \oplus \ldots \oplus (P_k \cup E_k)$ for any k, with $m \leq k \leq m+n-1$.*

2.2 Action Languages

The purpose of an action language is to provide ways of describing how an environment evolves given a set of external actions. A specific environment that can be modified through external actions is called an *action domain*. To any action domain we associate a pair of sets of atoms $\mathcal{F}$ and $\mathcal{A}$. We call the elements of $\mathcal{F}$ *fluent atoms* or simply *fluents*, and the elements of $\mathcal{A}$ *action atoms* or simply *actions*. Basically, the fluents are the observables in the environment and the actions are, clearly, the external actions. A *fluent literal* (resp. *action literal*) is an element of $\mathcal{F}_L$ (resp. an element of $\mathcal{A}_L$). In the following, we will use the letter Q to denote a fluent atom, the letter F to denote a fluent literal, and the letter A to denote an action atom. A *state of the world* (or simply a *state*) is any interpretation over $\mathcal{F}$. We say a fluent literal F is true at a given state s iff F belongs to s. Given a set (or, by abuse of notation, a conjunction) of fluent literals $Cond$ we say s *satisfies* $Cond$, and write $s \models Cond$, iff $Cond \subseteq s$.

Each action language provides ways to describe action domains through sets of expression called *action programs*. Usually, the semantics of an action program is defined in terms of a *transition system*, i.e. a function whose argument is any pair (s, K), where s is a state of the world and K is a subset of $\mathcal{A}$, and whose value is any set of states of the world. Intuitively, given the current state of the world, a transition system specifies which are the possible resulting states after simultaneously performing all actions in K.

Two kinds of expressions that are common within action description languages are *static* and *dynamic rules*. The *static rules* basically describe the rules of the domain, while *dynamic rules* describe effects of actions. A dynamic rule has a set of *preconditions*, namely conditions that have to be satisfied in the present state in order to have a particular effect in the future state, and *postconditions* describing such an effect.

In the following we will consider three existing action languages, namely: $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$. The language $\mathcal{A}$ [13] is very simple. It only allows dynamic rules of the form

$$A \textbf{ causes } F \textbf{ if } Cond$$

where $Cond$ is a conjunction of fluent literals. Such a rule intuitively means: performing the action A causes F to be true in the next state if $Cond$ is true in the current state. The language $\mathcal{B}$ [13] is an extension of $\mathcal{A}$ which also considers static rules. In $\mathcal{B}$, static rules are expressions of the form

$$F \textbf{ if } Body$$

where $Body$ is a conjunction of fluent literals. Intuitively, such a rule means: if $Body$ is true in the current state, then F is also true in the current state. A fundamental notion, in both $\mathcal{A}$ and $\mathcal{B}$, is *fluent inertia* [13]. A fluent F is inertial if its truth value is preserved from a state to another, unless it is changed by the (direct or indirect) effect of an action. Hereafter a program written in the language $\mathcal{B}$ will be called a $\mathcal{B}$ program.

The semantics of $\mathcal{B}$ is defined in terms of a transition system, as sketched above. For introducing the particular transition function that, given a state s and a set of actions K, determines the possible resulting states according to $\mathcal{B}$, we first consider the set $D(s, K)$ of fluents literals that are true as a (direct) consequence of actions. Any literal F is a direct consequence of state s and actions K if it is in the head of a dynamic rule A **causes** F **if** $Cond$ such that

$A \in K$ and *Cond* is true in s. Then a state s' is a possible resulting states fro s iff any fluent literal in s is an ele ent of $D(s, K)$ or is a true literal in s (that followed by inertia) or is a consequence of a static rule:

Definition 4. *Let P be any $\mathcal{B}$ program with set of fluents $\mathcal{F}$, let $\mathcal{R}$ be the set of all static rules in P, and let s be a state and K any set of actions. Moreover, let $D(s, K)$ be the following set of literals*

$$D(s, K) = \{F : \exists\ A\ \textbf{\textit{causes}}\ F\ \textbf{\textit{if}}\ Cond\ \in\ P\ s.t.\ A \in K\ \wedge\ s \models Cond\}$$

and let $\mathcal{R}^{LP}$ be the logic program:

$$\mathcal{R}^{LP} = \{F \leftarrow Body : F\ \textbf{\textit{if}}\ Body \in \mathcal{R}\}$$

A state s' is a resulting state from s given P and the set of actions K iff

$$s' = least(s \cap s' \cup D(s, K) \cup \mathcal{R}^{LP})$$

where $least(P)$ is as in Definition 1

For a detailed explanation of $\mathcal{A}$ and $\mathcal{B}$ see e.g. [13].

Static and dyna ic rules are also the ingredients of the action language $\mathcal{C}$ [15, 16]. Static rules in $\mathcal{C}$ are of the for

caused J **if** H

while dyna ic rules are of the for

caused J **if** H **after** O

where J and H are for ulae such that any literal in the is a fluent literal, and O is any for ula such that any literal in it is a fluent or an action literal. The for- ula O is the precondition of the dyna ic rule and the static rule **caused** J **if** H is its postcondition. The se antic of $\mathcal{C}$ is based on *causal theories*[15]. Causal theories are sets of rules of the for **caused** J **if** H, each such rule eaning: If H is true this is an explanation for J. A basic principle of causal theories is that so ething is true iff it is caused by so ething else. Given any action progra P, a state s and a set of actions K, we consider the causal theory T given by the static rules of P and the postconditions of the dyna ic rules whose preconditions are true in $s \cup K$. Then s' is a possible resulting state iff it is a causal odel of T.

3 Evolp Action Programs

As we have seen, Evolp and action description languages share the idea of a system that evolves. In both, the evolution is influenced by external events (respectively, updates and actions). Evolp is actually a progra ing language devised for representing any kind of computational problem, while action description languages are devised for the specific purpose of describing actions. A natural idea is then to develop special kind of Evolp sequences for representing actions, and then compare such kind of programs with existing action description languages. We will develop one such kind of programs, and call them *Evolp Action Programs* (EAPs).

Following the underlying notions of Evolp, we use the basic construct *assert* for defining special-purpose macros. As it happens with other action description languages, EAPs are defined over a set of fluents $\mathcal{F}$ and a set of actions $\mathcal{A}$. In EAPs, a state of the world is any interpretation over $\mathcal{F}$. To describe action domains we use an initial Evolp sequence, $I \oplus D$. The Evolp program D contains the description of the environment, while I contains some initial declarations, as it will be clarified later. As in $\mathcal{B}$ and $\mathcal{C}$, EAPs contain static and dynamic rules.

A *static rule* over $(\mathcal{F}, \mathcal{A})$ is simply an Evolp rule of the form

$$F \leftarrow Body.$$

where F is a fluent literal and *Body* is a set of fluent literals.

A *dynamic rule* over $(\mathcal{F}, \mathcal{A})$ is a (macro) expression

$$\mathbf{effect}(\tau) \leftarrow Cond.$$

where τ is any static rule $F \leftarrow Body$ and *Cond* is any set of fluent or action literals. The intuitive meaning of such a rule is that the static rule τ has to be considered *only* in those states whose predecessor satisfies condition *Cond*. Since some of the conditions literals in *Cond* may be action atoms, such a rule may describe the effect of a given set of actions under some conditions. Such an expression stands for the following set of Evolp rules:

$$F \leftarrow Body,\ event(F \leftarrow Body). \quad (1)$$

$$assert(event(F \leftarrow Body)) \leftarrow Cond. \quad (2)$$

$$assert(not\ event(F \leftarrow Body)) \leftarrow event(\tau), not\ assert(event(F \leftarrow Body)) \quad (3)$$

where $event(F \leftarrow Body)$ is a new literal. Let us see how the above set of rules fits with its intended intuitive meaning. Rule (1) is not applicable whenever $event(F \leftarrow Body)$ is false. If at some step n, the conditions *Cond* are satisfied, then, by rule (2), $event(F \leftarrow Body)$ becomes true at step $n + 1$. Hence, at step $n + 1$, rule (1) will play the same role as static rule $F \leftarrow Body$. If at step $n + 1$ *Cond* is no longer satisfied, then, by rule (3) the literal $event(F \leftarrow Body)$ will become false again, and then rule (1) will be again not effective.

Besides static and dynamic rules, we still need another ingredient to complete our construction. As we have seen in the description of the $\mathcal{B}$ language, a notable concept is fluent inertia. This idea is not explicit in Evolp where *the rules* (and not the fluents) are preserved by inertia. Nevertheless, we can show how to obtain fluent inertia by using macro programming in Evolp. An *inertial declaration* over $(\mathcal{F}, \mathcal{A})$ is a (macro) expression $\mathbf{inertial}(\mathcal{K})$, where $\mathcal{K} \subseteq \mathcal{F}$. The intended intuitive meaning of such an expression is that the fluents in $\mathcal{K}$ are inertial. Before defining what this expression stands for, we state that the above mentioned program I is always of the form **initialize($\mathcal{F}$)**, where **initialize($\mathcal{F}$)** stands for the set of rules $Q \leftarrow prev(Q)$, where Q is any fluent in $\mathcal{F}$, and $prev(Q)$ are new atoms not in $\mathcal{F} \cup \mathcal{A}$. The *inertial declaration* **inertial($\mathcal{K}$)** stands for the set (where Q ranges over $\mathcal{K}$):

$$assert(prev(Q)) \leftarrow Q. \qquad\qquad assert(not\ prev(Q)) \leftarrow not\ Q.$$

Let us consider the behaviour of this macro. If we do not declare Q as an inertial fluent, the rule $Q \leftarrow prev(Q)$ has no effect. If we declare Q as an inertial literal, $prev(Q)$ is true in the current state iff in the previous state Q was true. Hence, in this case, Q is true in the current state *unless* there is a static or dynamic rule that rejects such assumption. Viceversa, if Q was false in the previous state, then Q is true in the current one iff it is derived by a static or dynamic rule. We are now ready to formalize the syntax of Evolp action programs.

Definition 5. *Let $\mathcal{F}$ and $\mathcal{A}$ be two disjoint sets of propositional atoms. An Evolp action program (EAP) over ($\mathcal{F}$, $\mathcal{A}$) is any Evolp sequence $I \oplus D$, where $I = Initialize(\mathcal{F})$, and D is any set with static and dynamic rules, and inertial declarations over $(\mathcal{F}, \mathcal{A})$*

Given an Evolp action program $I \oplus D$, the initial state of the world s (which, as stated above, is an interpretation over $\mathcal{F}$) is passed to the program together with the set K of the actions performed at s, as part of an external event. A resulting state is the last element of any evolving stable model of $I \oplus D$ given the event $s \cup K$ restricted to the set of fluent literals. I.e:

Definition 6. *Let $I \oplus D$ be any EAP over $(\mathcal{F}, \mathcal{A})$, and s a state of the world. Then s' is a resulting state from s given $I \oplus D$ and the set of actions K iff there exists an evolving stable model M_1, M_2 of $I \oplus D$ given the external events $s \cup K, \emptyset$ such that $s' \equiv_{\mathcal{F}} M_2$(where by $s' \equiv_{\mathcal{F}} \mathcal{M}_2$ we simply mean $s' \cap \mathcal{F}_{Lit} = M_2 \cap \mathcal{F}_{Lit}$).*

This definition can be easily generalized to sequences of set of actions.

Definition 7. *Let $I \oplus D$ be any EAP and s a state of the world. Then s' is a resulting state from s given $I \oplus D$ and the sequence of sets of actions $K_1 \dots, \ K_n$ iff there exists an evolving stable model $M_1, \dots, M_{n+1}$ of $I \oplus D$ given the external events $(s \cup K_1), \dots, \ K_n, \emptyset$ such that $s' \equiv_{\mathcal{F}} M_{n+1}$.*

Since EAPs are based on the Evolp semantics, which in turn is an extension of the stable model semantics for normal logic programs, we can easily prove that the complexity of the computation of the two semantics is the same.

Theorem 1. *Let $I \oplus D$ be any EAP over $(\mathcal{F}, \mathcal{A})$, s a state of the world and $K \subseteq \mathcal{A}$. To find a resulting state s' from s given $I \oplus D$ and the set of actions K is an NP-complete problem.*

It is important to notice that, if the initial state s does not satisfies the static rules of the EAP, the correspondent Evolp sequence has no stable model, and hence there will be no successor state. This is, in our opinion, a good result: The initial state is just a state as any other. It would be strange if such state would not satisfy the rules of the domain. If this situation occurs, most likely either the translation of the rules, or the one of the state, presents some errors. From now onwards we will assume that the initial state satisfies the static rules of the domain.

To illustrate EAPs, we now show an example of their usage by elaborating on probably the most famous example of reasoning about actions. The presented elaboration highlights some important features of EAPs, viz. the possibility of handling non-deterministic effects of actions, non-inertial fluents, non-executable actions, and effects of actions lasting for just one state.

Example 1 (An elaboration of the Yale shooting problem). In the original Yale shooting problem [27], there is a single-shot gun which is initially unloaded, and a turkey which is initially alive. One can load the gun and shoot the turkey. If one shoots, the gun becomes unloaded and the turkey dies. We consider a slightly more complex scenario where there are several turkeys, and where the shooting action refers to a specific turkey. Each time one shoots as specific turkey, one either hits and kills the bird, or misses it. Moreover, the gun becomes unloaded and there is a bang. It is not possible to shoot with an unloaded gun. We also add the property that any turkey moves iff it is not dead.

For expressing that an action is not executable under some conditions, we make use of a well known behaviour of the stable model semantics. Suppose a given EAP contains a dynamic rules of the form $\textbf{effect}(u \leftarrow not\ u) \leftarrow Cond$, where u is a literal which does not appear elsewhere (in the following, for representing such rules, we use the notation $\textbf{effect}(\bot) \leftarrow Cond$). With such a rule, if $Cond$ is true in the current state, then there is no resulting state. This happens because, as it is well known, programs containing $u \leftarrow not\ u$ and no other rules for u, have no stable models.

To represent the problem, we consider the fluents $dead(X)$, $moving(X)$, $hit(X)$, $missed(X)$, $loaded$, $bang$, plus the auxiliary fluent u, and the actions $shoot(X)$ and $load$ (where the Xs range over the various turkeys). The fluents $dead(X)$ and $loaded$ are inertial fluents, since their truth value should remain unchanged until modified by some action effect. The fluents $missed(X)$, $hit(X)$ and $bang$ are not inertial. The problem is encoded by the EAP $I \oplus D$, where

$$I = \textbf{initialize}(dead(X), moving(X), missed(X), hit(X),\ loaded,\ bang,\ u)$$

and D is the following set of expressions

$\textbf{effect}(\bot) \leftarrow shoot(X),\ not\ loaded$	$\textbf{inertial}(loaded)$
$moving(X) \leftarrow not\ dead(X)$	$\textbf{inertial}(dead(X))$
$\textbf{effect}(dead(X) \leftarrow hit(X)) \leftarrow\ shoot(X)$	$\textbf{effect}(loaded) \leftarrow load$
$\textbf{effect}(hit(X) \leftarrow not\ missed(X)) \leftarrow shoot(X)$	$\textbf{effect}(bang) \leftarrow shoot(X)$
$\textbf{effect}(missed(X) \leftarrow not\ hit(X)) \leftarrow shoot(X)$	$\textbf{effect}(not\ loaded) \leftarrow shoot(X)$

Let us analyze this EAP. The first rule encodes the impossibility to execute the action $shoot(X)$ when the gun is unloaded. The static rule $moving(X) \leftarrow$

not dead(X) implies that, for any turkey X, *moving*(X) is true if *dead*(X) is false. Since this is the only rule for *moving*(X), it further holds that *moving*(X) is true iff *dead*(X) is false. Notice that declaring *moving*(tk) as inertial, would result, in our description, in the possibility of having a moving dead turkey! This is why fluents *moving*(X) have not been declared as inertial. In fact, suppose we insert **inertial(***moving*(X)**)** in the EAP above. Suppose further that *moving*(tk) is true at state s, that one shoots at tk and kills it. Since *moving*(tk) is an inertial fluent, in the resulting state *dead*(tk) is true, but *moving*(tk) remains true by inertia. Also notable is how effects that last only for one state, like the noise provoked by the shoot, are easily encoded. The last three dynamic rules on the left encode a non deterministic behaviour: each shoot action can either hit and kill a turkey, or miss it.

To see how this EAP encodes the desired behaviour of this domain, consider the following example of evolution. In this example, to lightening the notation, we omit the negative literals belonging to interpretations. Let us consider the initial state $\{\}$ (which means that all fluents are false). The state will remain unchanged until some action is performed. If one load the gun, the program is updated with the external event $\{load\}$. In the unique successor state, the fluent *loaded* is true and nothing else changes. The truth value of *loaded* remains then unchanged (by inertia) until some other action is performed. The same applies to fluents *dead*(X). The fluents *bang*, *missed*(X), and *hit*(X) remain false by default. If one shoots at a specific turkey, say Smith, and the program is updated with the event *shoot*$(smith)$, several things happen. First, *loaded* becomes false, and *bang* becomes true, as an effect of the action. Moreover, the rules:

$$hit(smith) \leftarrow not\ missed(smith).$$
$$missed(smith) \leftarrow not\ hit(smith).$$
$$dead(smith) \leftarrow hit(smith).$$

are considered as rules of the domain for one state. As a consequence, there are two possible resulting states. In the first one, *missed*$(smith)$ is true, and all the others fluents are false. In the second one *hit*$(smith)$ is true, *missed*$(smith)$ is false and, by the rule $dead(smith) \leftarrow hit(smith)$, the fluent *dead*$(smith)$ becomes true. In both the resulting states, nothing happens to the truth value of the fluents *dead*(X), *hit*(X), and *dead*(X) for $X \neq smith$.

4 Relationship to Existing Action Languages

In this section we show embeddings into EAPs of the action languages $\mathcal{B}$ and (the definite fragment of) $\mathcal{C}$[2]. We will assume that the considered initial states are consistent wrt. the static rules of the program, i.e. if the body of a static rule is true in the considered state, the head is true as well.

[2] The embedding of language $\mathcal{A}$ is not explicitly exposed here since $\mathcal{A}$ is a (proper) subset of the $\mathcal{B}$ language.

Let us consider first the $\mathcal{B}$ language. The basic ideas of static and dyna ic rules are very si ilar in $\mathcal{B}$ and in EAPs. The ain difference between the two is that in $\mathcal{B}$ *all* the fluents are inertial, whilst in EAPs only those that are declared as such are inertial. The translation of $\mathcal{B}$ into EAPs is then straightforward: All fluents are declared as inertial and then the syntax of static and dyna ic rules is adapted. In the following we use, with abuse of notation, *Body* and *Cond* both for conjunctions of literals and for sets of literals.

Definition 8. *Let P be any action program in $\mathcal{B}$ with set of fluents $\mathcal{F}$. The translation $B(P, \mathcal{F})$ is the pair $(I^B \oplus D^{BP}, \mathcal{F}^B)$ where: $\mathcal{F}^B \equiv \mathcal{F}$, $I^B =$ **initialize($\mathcal{F}$)** and D^{BP} contains exactly the following rules:*

- ***inertial(Q)** for each fluent $Q \in \mathcal{F}$*
- *a rule $F \leftarrow Body$ for any static rule F **if** Body in P.*
- *a rule **effect(F)** $\leftarrow A$, Cond. for any dynamic rule A **causes** F **if** Cond in P.*

Theorem 2. *Let P be any $\mathcal{B}$ program with set of fluents $\mathcal{F}$, $(I^B \oplus D^{BP}, \mathcal{F})$ its translation, s a state and K any set of actions. Then s' is a resulting state from s given P and the set of actions K iff it is a resulting state from s given $I^B \oplus D^{BP}$ and the set of actions K.*

This theore akes it clear that there is a close relationship between *EAPs* and the $\mathcal{B}$ language. In practice, *EAPs* generalize $\mathcal{B}$ by allowing both inertial and non inertial fluents and by ad itting rules, rather then si ply facts, as effects of actions.

Let us consider now the action language $\mathcal{C}$. Given a co plete description of the current state of the world and perfor ed actions, the proble of finding a resulting state is a proble of the satisfiability of a causal theory, which is known to be $\sum_P^2$-hard (cf. [15]). So, this language belongs to a category with higher co plexity than EAPs whose satisfiability is NP-co plete. However, only a frag ent of $\mathcal{C}$ is i ple ented and the co plexity of such frag ent is NP. This frag ent is known as the *definite fragment* of $\mathcal{C}$ [15]. In this frag ent, static rules are expressions of the for **caused** F **if** *Body* where F is a fluent literal and *Body* is a conjunction of fluent literals, while dyna ic rules are expressions of the for **caused** *not* F **if** *Body* **after** *Cond* where *Cond* is a conjunction of fluent or action literals[3] For this frag ent it is possible to provide a translation into EAPs.

The ain proble of the translation of $\mathcal{C}$ into EAPs lies in the si ulation of causal reasoning with stable odel se antics. The approach followed here to encode causal reasoning with stable odels is in line with the one proposed in [21]. We need to introduce so e auxiliary predicates and define a syntactic

[3] The definite fragment defined in [15] is (apparently) more general, allowing *Cond* and *Body* to be arbitrary formulae. However, it is easy to prove that such kind of expressions are equivalent to a set of expressions of the form described above.

transfor ation of rules. Let $\mathcal{F}$ be a set of fluents, and let $\mathcal{F}^C$ denote the set of fluents $\mathcal{F} \cup \{Q_N \mid Q \in \mathcal{F}\}$. We add, for each $Q \in \mathcal{F}$, the constraints:

$$\leftarrow not\ Q, not\ Q_N. \tag{4}$$

$$\leftarrow Q,\ Q_N. \tag{5}$$

Let Q be a fluent and $Body = F_1, \ldots, F_n$ a conjunction of fluent literals. We will use the following notation: $\overline{Q} = not\ Q_N$, $\overline{not\ Q} = not\ Q$ and $\overline{Body} = \overline{F_1}, \ldots, \overline{F_n}$

Definition 9. *Let P be any action program in the definite fragment of $\mathcal{C}$ with set of fluents $\mathcal{F}$. The translation $C(P, \mathcal{F})$ is the pair $(I^C \oplus D^{CP}, \mathcal{F}^C)$ where: $\mathcal{F}^C$ is defined as above, $I^C \equiv$ **initialize($\mathcal{F}^C$)** and D^{CP} consists exactly of the following rules:*

- *a rule **effect($Q \leftarrow \overline{Body}$)** $\leftarrow Cond$, for any dynamic rule in P of the form **caused** Q **if** $Body$ **after** $Cond$;*
- *a rule **effect($Q_N \leftarrow \overline{Body}$)** $\leftarrow Cond$, for any dynamic rule in P of the form **caused** not Q **if** $Body$ **after** $Cond$;*
- *a rule $Q \leftarrow \overline{Body}$, for any static rule in P of the form **caused** Q **if** $Body$;*
- *a rule $Q_N \leftarrow \overline{Body}$, for a static rule in P of the form **caused** not Q **if** $Body$;*
- *The rules (4) and (5), for each fluent $Q \in \mathcal{F}$.*

For this translation we obtain a result si ilar to the one obtained for the translations of the $\mathcal{B}$ language:

Theorem 3. *Let P be any action program in the definite fragment of $\mathcal{C}$ with set of fluents $\mathcal{F}$, $(I^C \oplus D^{CP}, \mathcal{F}^C)$ its translation, s a state, s^C the interpretation over $\mathcal{F}^C$ defined as follows: $s^C = s \cup \{Q_N \mid Q \in s\} \cup \{not\ Q_N \mid not\ Q \in s\}$ and K any set of actions. Then s^* is a resulting state from s^C given $I^C \oplus D^{CP}$ and the set of actions K iff there exists s' such that s' is a resulting state from s, given P and the set K and $s^* \equiv_{\mathcal{F}_L} s'$.*

By showing translations of the action languages $\mathcal{B}$ and the definite frag ent of $\mathcal{C}$ into EAPs, we proved that EAPs are *at least as expressive* as such languages. Moreover, the translations above are quite si ple: basically one EAP static or dyna ic rule for each static or dyna ic rule in the other languages. The next natural question is: Are they *more expressive*?

5 Updates of Action Domains

Action description languages describe the rules governing a do ain where actions are perfor ed, and the environ ent changes. In practical situations, it ay happen that the very rules of the do ain change with ti e too. When this happens, it would be desirable to have ways of specifying the necessary updates to the considered action progra , rather than to have to write a new one. EAPs are just a particular kind of Evolp sequences. So, as in general Evolp sequences, they can be updated by external events.

When one wants to update the existing rules with a rule τ, all that has to be done is to add the fact $assert(\tau)$ as an external event. This way, the rule τ is asserted and the existing Evolp sequence is updated. Following this line, we extend EAPs by allowing the external events to contain facts of the form $assert(\tau)$, where τ is an Evolp rule, and we show how they can be used to express updates to EAPs. For simplicity, below we use the notation $assert(R)$, where R is a set of rules, for the set of expressions $assert(\tau)$ where $\tau \in R$.

To illustrate how to update an EAP, we come back to Example 1. Let $I \oplus D$ be the EAP defined in there. Let us now consider that after some shots, and dead turkeys, rubber bullets are acquired. One can now either load the gun with normal bullets or with a rubber bullets, but not with both. If one shoots with a rubber loaded gun, the turkey is not killed.

To describe this change in the domain, we introduce a new inertial fluent representing the gun being loaded with rubber bullets. We have to express that, if the gun is rubber-loaded, one can not kill the turkey. For this purpose we introduce the new macro:

$$not\ \mathbf{effect}(F \leftarrow Body) \leftarrow Cond.$$

where F, is a fluent literal, $Body$ is a set of fluents literals and $Cond$ is a set of fluent or action literals. We refer to such expressions as *effects inhibitions*. This macro simply stands for the rule

$$assert(not\ event(F \leftarrow Body)) \leftarrow Cond.$$

where $event(F \leftarrow Body)$ is as before. The intuitive meaning is that, if the condition $Cond$ is true in the current state, any dynamic rule whose effect is the rule $F \leftarrow Body$ is ignored.

To encode the changes described above, we update the EAP with the external event E_1 consisting of the facts $assert(I_1)$ where

$$I_1 = (\mathbf{initialize}(rubber\_loaded))$$

Then, in the subsequent state, we update the program with the external update $E_2 = assert(D_1)$ where D_1 is the set of rules[4]

$$\begin{array}{l}
\mathbf{inertial}(rubber_loaded).\\
\mathbf{effect}(rubber_loaded) \leftarrow rubber_load.\\
\mathbf{effect}(not\ rubber_loaded) \leftarrow shoot(X).\\
\mathbf{effect}(\bot) \leftarrow rubber_loaded,\ load.\\
\mathbf{effect}(\bot) \leftarrow loaded,\ rubber_load.\\
not\ \mathbf{effect}(dead(X) \leftarrow hit(X)) \leftarrow rubber_loaded.
\end{array}$$

[4] In the remainder, we use $assert(U)$, where U is a set of macros (which are themselves sets of Evolp rules), to denote the set of all facts $assert(\tau)$ such that there exists a macro η in U with $\tau \in \eta$.

Let us analyze the proposed update. First, the fluent *rubber_loaded* is initialized. It is important to initialize any fluent before starting to use it. The newly introduced fluent is declared as inertial, and two dynamic rules are added specifying that load actions are not executable when the gun is already loaded in a different way. Finally we use the new command to specify that the effect $dead(X) \leftarrow hit(X)$ does not occurs if, in the previous state, the gun was loaded with rubber bullets. Since this update is more recent than the original rule $\mathbf{effect(}dead(X) \leftarrow hit(X)\mathbf{)} \leftarrow shoot(X)$, the dynamic rule is updated.

Basically updating the original EAP with the rule

$$not\ \mathbf{effect(}dead(X) \leftarrow hit(X)\mathbf{)} \leftarrow rubber\_loaded.$$

has the effect of adding *not rubber_loaded* to the preconditions of the dynamic rule

$$\mathbf{effect(}dead(X) \leftarrow hit(X)\mathbf{)} \leftarrow shoot(X).$$

So far we have shown how to update the preconditions of a dynamic rule. It is also possible to update static rules and the descriptions of effects of actions. Suppose the cylinder of the gun becomes dirty and, whenever one shoots, the gun may either work properly or fail. If the gun fails, the action *shoot* has no effect. We introduce two new fluents in the program with the event $assert(I_2)$ where $I_2 = \mathbf{initialize(}fails, work\mathbf{)})$. Then, we assert the event $E_2 = assert(D_2)$ where D_2 is the following EAP

$$\begin{aligned}
\mathbf{effect(}fails \leftarrow not\ work\mathbf{)} &\leftarrow shoot(X).\\
\mathbf{effect(}work \leftarrow not\ fails\mathbf{)} &\leftarrow shoot(X).\\
not\ missed(X) &\leftarrow fails.\\
not\ hit(X) &\leftarrow fails.\\
not\ bang &\leftarrow fails.\\
\mathbf{effect(}loaded \leftarrow fails\mathbf{)} &\leftarrow loaded.\\
\mathbf{effect(}rubber_loaded \leftarrow fails\mathbf{)} &\leftarrow rubber_loaded.
\end{aligned}$$

The first two dynamic rules simply introduce the possibility that a failure may occur every time we shoot. The three static rules describe changes in the behaviour of the environment when the gun fails, and amount to negate what was entailed by static and dynamic rules in D. The last two dynamic rules update two of the dynamic rules in D and D_1, respectively. These rules specify that, when a failure occurs, the gun remain loaded with the same kind of bullet. Since the new rules of D_2 are more recent than the rules in D and D_1, they update these latter ones.

This last example shows how to update static and dynamic rules with new static and dynamic rules. To illustrate how this is indeed achieved in this example, we now show a possible evolution of the updated system. Suppose currently the gun is not loaded. One loads the gun with a rubber bullet, and then shoots at the turkey named Trevor. The initial state is $\{\}$. The first set of actions is $\{rubber\_load\}$ The resulting state after this action is $s' \equiv \{rubber\_loaded\}$. Suppose one performs the action *load*. Since the EAP is updated with the dynamic rule **effect(**$\bot$**)** $\leftarrow$ *rubber_loaded*, *load*. there is no resulting state. This happens because we have perfor ed a non executable action. Suppose, instead, that the second set of actions is $\{shoot(trevor)\}$. In this case there are three possible resulting states. In one the gun fails and, in it, the resulting state is again s'. In the second, the gun works but the bullet isses Trevor. In this case, the resulting state is $s''_1 \equiv \{missed(trevor)\}$. Finally, in the third, the gun works and the bullet hits Trevor. Since the bullet is a rubber bullet, Trevor is still alive. In this case the resulting state is $s''_2 \equiv \{hit(trevor)\}$.

The events ay introduce changes in the behaviour of the original EAP. This opens a new proble . In classical action languages we do not care about the previous *history* of the world: If the current state of the world is s, the co putation of the resulting states is not affected by the states before s. In the case of EAPs the situation is different, since external updates can change the behaviour of the considered EAP. Fortunately, we do not have to care about the *whole* history of the world, but just about those events containing new initializations, inertial declarations, effects inhibitions, and static and dyna ic rules.

It is possible to have a co pact description of an EAP that is updated several ti es via external events. For that we need to further extend the original definition of EAPs.

Definition 10. *An* updated *Evolp action program over* $(\mathcal{F}, \mathcal{A})$ *is any sequence* $I \oplus D_1 \oplus \ldots \oplus D_n$ *where* I *is* ***initialize(***$\mathcal{F}$***)***, and the various* D_k *are sets consisting of static rules, dynamic rules, inertial declarations and effects inhibitions such that any fluent appearing in* D_k *belongs to* $\mathcal{F}$*.*

Definition 11. *Let* $I \oplus D_1 \oplus \ldots \oplus D_n$ *be any* updated *EAP and* s *a state of the world. Then* s' *is a resulting state from* s *given* $I \oplus D_1 \oplus \ldots \oplus D_n$ *and the sequence of sets of actions* $K_1 \ldots, K_n$ *iff there exists an evolving stable model* $M_1, \ldots, M_n$ *of* $I \oplus D_1 \oplus \ldots \oplus D_n$ *given the external events* $(s \cup K_1), \ldots, K_n, \emptyset$ *such that* $s' \equiv_{\mathcal{F}} M_n$*.*

In general, if we updated an Evolp action progra $I \oplus D$ with the subsequent events $assert(I_1)$, $assert(D_1)$, where $I_1 \oplus D_1$ is another EAP, we obtain the equivalent updated Evolp action progra $(I \cup I_1) \oplus D \oplus D_1$ For ally:

Theorem 4. *Let* $I_0 \cup I \oplus D_0 \oplus D_1 \oplus \ldots \oplus D_k$ *be any update EAP over* $(\mathcal{F}, \mathcal{A})$*. Let* $\bigoplus E_i^n$ *be a sequence of events such that:* $E_1 = K_1 \cup s$*, where* s *is any state of the world and* K_1 *is any set of actions; and the others* E_i*s are any set of actions* K_α*, or any set assert(****initialize(***$\mathcal{F}_\beta$***)****) where* $\bigcup \mathcal{F}_\beta \equiv I$*, or any* $assert(D_i)$ *with* $1 \leq i \leq k$*. Let* $s_1, \ldots, s_n$ *be a sequence of possible resulting states from* s *given the EAP* $I_0 \oplus D_0$ *and the sequence of events* $\bigoplus E_i^n$ *and* K_{n+1} *a set of actions. Then* $s_1, \ldots, s_n, s'$ *is a resulting state from* s *given* $I_0 \oplus D_0$ *and the sequence of events* $\bigoplus E_i^n \oplus K_{n+1}$ *iff* s' *is a resulting state from* s_n *given* $I_0 \cup I \oplus D_0 \oplus D_1 \oplus \ldots \oplus D_k$ *and the set of actions* K_{n+1}*.*

By applying this theore we can, for instance, si plify the updates to the original EAP of the exa ple in this section into the updated EAP $I_{sum} \oplus D \oplus D_1 \oplus D_2$, where $I_{sum} \equiv I \cup I_1 \cup I_2$, I and D are as in Exa ple 1, and the I_is and D_is are as described above.

Yet one more possibility opened by updated Evolp action programs is to cater for successive elaborations of a program. Consider an initial problem described by an EAP $I \oplus D$. If we want to describe an elaboration of the program, instead of *rewriting* $I \oplus D$ we can simply *update* it with new rules. This gives a new answer to the problem of elaboration tolerance [25] and also open the new possibility of *automatically update* action programs by other action programs.

The possibility to elaborate on an action program is also discussed in [15] in the context of the $\mathcal{C}$ language. The solution proposed there, is to consider $\mathcal{C}$ programs whose rules have one extra fluent atom in their bodies, all these extra fluents being false by default. The elaboration of an action program P is the program $P \cup U$ where U is a new action program. The rules in U can defeat the rules in P by changing the truth value of the extra fluents. An advantage of EAP over that approach is that in EAPs the possibility of updating rules is a built-in feature rather then a programming technique involving manipulation of rules and introduction of new fluents. Moreover, in EAPs we can simply encode the new behaviours of the domain by new rules and then let these new rules update the previous ones.

6 Conclusions and Future Work

In this paper we have explored the possibility of using logic programs updates languages as action description languages. In particular, we have focused our attention on the Evolp language [2]. As a first point, we have defined a new action language paradigm, christened Evolp action programs, defined as a macro language over Evolp. We have provided an example of usage of this language, and compared Evolp action programs with action languages $\mathcal{A}$, $\mathcal{B}$ and the definite fragment of $\mathcal{C}$, by defining simple translations into Evolp of programs in these languages. Finally, we have also shown and argued about the capability of EAPs to handle changes in the domain during the execution of actions.

Though all the results in this paper refer to the update language Evolp, it is not our stance that these could not be obtained if other LP update languages were used instead. For recasting (some) of the results in other LP update languages, one would have to resort to established relationships between the various LP update languages, such as the ones found in [2, 19]. Also, the possibility of handling changes in the domain shown by EAPs, could in principle be obtained if, instead of Evolp, another update language with the capability of updating update rules were used instead. Another LP update language with this capability is the KABUL language defined in [19]. However, the study of which of the existing LP update languages could be used as action description languages, in a way similar to what is described here for Evolp, is outside the scope of this paper, and would, in our opinion, fit better in a paper with a focus on relationship among various LP update languages. Our goal in this paper was to show that (at least) one LP update language can be used for describing effects of actions, and can be formally compared with existing action description languages. This goal was achieved by showing exactly that for the language Evolp.

Several i portant topics are not touched here, and will be subject of future work. I portant fields of research are how to deal, in the Evolp context, with the proble of planning prediction and postdiction [24], when dealing with inco plete knowledge of the state of the world. Yet another topic involves the possibility of concurrent execution of actions. Nevertheless, we have not fully explored this topic, and confronted the results with extant works [6, 18].

The develop ent of i ple entations for Evolp and EAPs is another necessary step. Finally EAPs have to be tested in real and co plex contexts.

References

1. J. J. Alferes, F. Banti, A. Brogi, and J. A. Leite. Semantics for dynamic logic programming: a principled based approach. In *7th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR-7)*, volume 1730 of *LNAI*. Springer, 2004.
2. J. J. Alferes, A. Brogi, J. A. Leite, and L. M. Pereira. Evolving logic programs. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *8th European Conf. on Logics in AI (JELIA'02)*, volume 2424 of *LNAI*, pages 50–61. Springer, 2002.
3. J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. C. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *The Journal of Logic Programming*, 45(1–3):43–70, September/October 2000.
4. J. J. Alferes, L. M. Pereira, H. Przymusinska, and T. Przymusinski. LUPS: A language for updating logic programs. *Artificial Intelligence*, 132(1 & 2), 2002.
5. J. J. Alferes, L. M. Pereira, T. Przymusinski, H. Przymusinska, and P. Quaresma. Preliminary exploration on actions as updates. In M. C. Meo and M. V. Ferro, editors, *Joint Conference on Declarative Programming (AGP-99)*, 1999.
6. C. Baral and M. Gelfond. Reasoning about effects of concurrent actions. *Journal of Logic Programming*, 31:85–118, 1997.
7. C. Baral, M. Gelfond, and Alessandro Provetti. Representing actions: Laws, observations and hypotheses. *Journal of Logic Programming*, 31, April–June 1997.
8. F. Buccafurri, W. Faber, and N. Leone. Disjunctive logic programs with inheritance. In D. De Schreye, editor, *Proceedings of the 1999 International Conference on Logic Programming (ICLP-99)*, Cambridge, November 1999. MIT Press.
9. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. A framework for declarative update specifications in logic programs. In Bernhard Nebel, editor, *Proceedings of the seventeenth International Conference on Artificial Intelligence (IJCAI-01)*, pages 649–654, San Francisco, CA, 2001. Morgan Kaufmann Publishers, Inc.
10. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. On properties of semantics based on causal rejection. *Theory and Practice of Logic Programming*, 2:711–767, November 2002.
11. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. A. Bowen, editors, *5th International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.
12. M. Gelfond and V. Lifschitz. Representing actions and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.
13. M. Gelfond and V. Lifschitz. Action languages. *Electronic Transactions on AI*, 16, 1998.
14. E. Giunchiglia, J. Lee, V. Lifschitz, N. Mc Cain, and H. Turner. Representing actions in logic programs and default theories: a situation calculus approach. *Journal of Logic Programming*, 31:245–298, 1997.

15. E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153:49–104, 2004.
16. E. Giunchiglia and V. Lifschitz. An action language based on causal explanation: Preliminary report. In *AAAI'98*, pages 623–630, 1998.
17. M. Homola. Dynamic logic programming: Various semantics are equal on acyclic programs. In this volume.
18. J. Lee and V. Lifschitz. Describing additive fluents in action language C+. In William Nebel, Bernhard; Rich, Charles; Swartout, editor, *Proc. IJCAI-03*, pages 1079–1084, Cambridge, MA, 2003.
19. J. A. Leite. *Evolving Knowledge Bases*, volume 81 of *Frontiers in Artificial Intelligence and Applications.* IOS Press, 2003.
20. J. A. Leite and L. M. Pereira. Generalizing updates: from models to programs. In *LPKR'97: workshop on Logic Programming and Knowledge Representation*, 1997.
21. V. Lifschitz. *The Logic Programming Paradigm: a 25-Year Perspective*, chapter Action languages, answer sets and planning, pages 357–373. Springer Verlag, 1999.
22. V. Lifschitz and T. Woo. Answer sets in general non-monotonic reasoning (preliminary report). In B. Nebel, C. Rich, and W. Swartout, editors, *Proceedings of the 3th International Conference on Principles of Knowledge Representation and Reasoning (KR-92).* Morgan-Kaufmann, 1992.
23. John Wylie Lloyd. *Foundations of Logic Programming.* Springer,, Berlin, Heidelberg, New York,, 1987.
24. J. McCarthy. Programs with commons sense. In *Proceedings of Teddington Conference on The Mechanization of Thought Process*, pages 75–91, 1959.
25. J. McCarthy. *Mathematical logic in artificial intelligence*, pages 297–311. Daedalus, 1988.
26. S. Russel and P. Norvig. *Artificial Intelligence A Modern Approach.* Artificial Intelligence. Prentice Hall, 1995.
27. D. McDermott S. Hanks. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, (1987).

A Proofs

Before presenting the proofs of the results in this paper, we present an alternative definition of the transition function of EAPs, and prove its equivalence to the original definition (Definition 6). We do so because in so e proofs it is ore convenient to use this alternative definition.

In this alternative definition, and in its prove, we will use the notation $S|_{\mathcal{I}}$ to denote the restriction of the set S to the literals in the set $\mathcal{I}$ i.e., to denote $S \cap \mathcal{I}$.

Theorem 5. *Let $I \oplus D$ be any EAP, s a state of the world and K a set of actions. Let $\mathcal{R}$ be the set of static rules in D, $\mathcal{I}$ the following set of fluent literals*

$$\mathcal{I} = \{Q \in \mathcal{F} : \ \textbf{\textit{inertial(Q)}} \in D\} \cup \{not\ Q : \ Q \in \mathcal{F} : \ \textbf{\textit{inertial(Q)}} \in D\}$$

and $D(s, K)$ be the following set of rules:

$$D(s, K) = \{\tau : \textbf{\textit{effect(}}\tau\textbf{\textit{)}} \leftarrow Cond \ \in D \ \wedge K \cup s \models Cond\}$$

Then s' is a resulting state from s given $I \oplus D$ and the set of actions K iff

$$s' = least\left((s \cap s' \cap \mathcal{I}) \ \cup Default(s', \mathcal{R} \cup D(s,K))|_{(\mathcal{F}_L \setminus \mathcal{I})} \ \cup D(s,K) \ \cup \mathcal{R}\right) \tag{6}$$

Proof. By Definition 6, s' is a resulting state from s given $I \oplus D$ and the set of actions K iff there exists an evolving stable model M_1, s^* of $I \oplus D$ given the external events $s \cup K, \emptyset$ such that $s' \equiv_{\mathcal{F}} s^*$. An interpretation M_1 is an evolving stable model of $I \oplus D$ given the external events $s \cup K$ iff M_1 is a refined stable models of $I \oplus D \cup s \cup K$ i.e.,

$$M_1 = least\left((I \cup D \cup s \cup K) \setminus Rej^s(M_1, I \oplus D \cup K \cup s) \cup Default(M_1)\right)$$

All the atoms of the form $event(\tau)$ where τ is the effect of a dynamic rule are false by default in $I \oplus D \cup K \cup s$. Hence the rules of the form (1) and (3), which have those atoms in their bodies, play no role when calculating the least model. Also all the literals of the form $prev(Q)$, where Q is a fluent literal, are false by default, and so the rules of the form $Q \leftarrow prev(Q)$ play no role either. Since the initial (starting) state s is always assumed consistent wrt. the static rules, there is no conflict between the static rules in D. Thus, static rules do not reject any literal in s nor do they infer any fluent literal that does not belong to s. So, we can simplify the expression above in the following way:

$$M_1 = least\left((D^* \cup s \cup K) \cup Default(M_1)\right)$$

where D^* is the set of all rules the form

$$assert(event(\tau)) \leftarrow Cond.$$

for which there is a dynamic rule $\mathbf{effect(}\tau\mathbf{)} \leftarrow Cond$ in D, union with the set of all rules of the form

$$assert(prev(Q)) \leftarrow Q. \qquad\qquad assert(not\ prev(Q)) \leftarrow not\ Q.$$

for every Q such that $\mathbf{inertial(}Q\mathbf{)}$ belongs to D.

Hereafter, for sake of simplicity, in interpretations we omit the negative literals of the form $not\ A$ whenever A is an auxiliary atom or an action literal. In other words, we omit $not\ A$ whenever $A \notin \mathcal{F}$. Moreover, by $Prev(s)$ we denote the set of literals which are either of the form $prev(F)$ where F is a fluent literal that is declared as inertial in D and is true in s, or of the form $not\ prev(F)$ where F is a fluent literal that is declared as inertial in D and is false in s. Finally, by $ED(s,K)$ we mean the set of literals $event(\tau)$ such that

$$assert(event(\tau)) \leftarrow Cond.$$

belongs to D and $s \cup K \models Cond$.

Given this, it is easy to see that the trace associated with any evolving interpretation M_1, s^* is the sequence $\mathcal{J}:\ I \oplus D \oplus Prev(s) \cup ED(s,K)$. So, M_1, s^*

is an evolving stable odel of $I \oplus D$ given the sequence of events $K, \emptyset$ iff s^* is a refined stable odel of $\mathcal{J}$.

Let s^* be any interpretation over the language of $I \oplus D$, and $s' = s^*|_{\mathcal{F}}$. To prove the theore , we si ply have to prove that s^* is a refined stable odel of $\mathcal{J}$ iff s' satisfies the equivalence (6). By definition of refined stable odel, s^* is a refined stable odel of $\mathcal{J}$ iff

$$s^* = least\left((I \cup D \cup Prev(s) \cup D(s,K)) \setminus Rej^S(s^*) \cup Default(s^*)\right)$$

$\Rightarrow$ Assu e that s^* is a refined stable odel of $\mathcal{J}$. To prove that s' satisfies the equivalence, we start by si plifying the expression above defining s^*.

Let $s' = s^*_{\mathcal{F}}$. Since s' only has fluent literals, the dyna ic rules and the inertial declarations in D play no role in verifying the equivalence. Hence, the only rules we are interested in are the static rules in $\mathcal{R}$. Moreover, since s^* is two valued, there is no utual rejection between the rules in $\mathcal{R}$: otherwise there would be a fluent literal Q such that all the rules with head Q or *not* Q would be rejected, and such that *not* Q would not be in the set $Default(s^*)$ as well. In such a case, neither Q nor *not* Q would be in s^* which would contradict the two valuedness of s^*. Finally, by partially evaluating the facts in $ED(s,K)$, in the rules of the for

$$F \leftarrow Body,\ event(F \leftarrow Body).$$

we can delete the ato s $event(\tau)$ fro the body of those rules whenever $event(\tau) \in ED(s,K)$, and delete one such rule when $event(\tau) \notin ED(s,K)$. With this, we can si plify the equivalence for s' into:

$$s' = least\left(I \setminus Rej^S(s^*) \cup Prev(s) \cup \mathcal{R} \cup D(s,K) \cup Default(s^*)\right)$$

We can split the set of default assu ptions into two subsets: the one concerning the inertial fluent literals; and the one concerning the fluent literals that are not inertial. Taking this splitting in consideration, the equivalence for s' beco es:

$$s' = least\begin{pmatrix} I \setminus Rej^S(s^*) \cup Prev(s) \cup Default(s^*)|_{\mathcal{I}} \cup \\ \mathcal{R} \cup D(s,K) \cup Default(s^*)|_{(\mathcal{F}_L \setminus \mathcal{I})} \end{pmatrix}$$

where $Default(s^*)$ stands for $Default(s^*, I \oplus R \cup D(s,K))|_{(\mathcal{F}_L \setminus \mathcal{I})}$. Notice that the expression $Default(s^*, I \oplus R \cup D(s,K))|_{(\mathcal{F}_L \setminus \mathcal{I})}$ is equivalent to $Default(s', \mathcal{R} \cup D(s,K))|_{(\mathcal{F}_L \setminus \mathcal{I})}$. Moreover, the expression $Default(s^*)|_{\mathcal{I}}$ is equivalent to $Default(s', s \cup \mathcal{R} \cup D(s,K))|_{(\mathcal{I})}$. Let $Inherit(s)$ be the set of rules:

$$Inherit(s^*) = \{Q \in \mathcal{F} : Q \leftarrow prev(Q) \in I \setminus Rej^S(s^*) \wedge prev(Q) \in Prev(s)\}$$

What re ains to show in order to prove that s' satisfies the equivalence (6) is that

$$Inherit(s^*) \cup Default(s^*)|_{\mathcal{I}} \equiv (s \cap s' \cap \mathcal{I})$$

For showing this, we consider separately the negative and the positive fluent literals. Let Q be a fluent literal that belongs to $(s \cap s' \cap \mathcal{I})$. We want to prove this is equivalent to say that $Q \leftarrow prev(Q)$ belongs to $I \setminus Rej^S(s^*)$ and that $Prev(Q) \in Prev(s)$ i.e., we want to prove that $Q \in Inherit(s^*)$.

The literal Q belongs to $(s \cap s' \cap \mathcal{I})$ iff $Q \in \mathcal{I}$, *not* $Q \notin s$ and *not* $Q \notin s'$. This i plies that there exists no rule in $\mathcal{R} \cup D(s,K)$ whose head is *not* Q and whose body is true. So, the rule $Q \leftarrow prev(Q)$ belongs to $I \setminus Rej^S(s^*)$ and, by $Q \in s$ and by definition of $Prev(s)$, we conclude that $Prev(Q) \in Prev(s)$. Let assu e now $Q \leftarrow prev(Q)$ belongs to $I \setminus Rej^S(s^*)$, then there exists no rule in $\mathcal{R} \cup D(s,K)$ whose head is *not* Q and whose body is true. If, further ore, $Prev(Q) \in Prev(s)$, then *not* $Q \notin Default(s^*)$ and so *not* Q is not derived by any rule nor by default assu ption. Thus, *not* $Q \notin s'$ and so $Q \in s$. Moreover, by definition if $prev(Q) \in Prev(s)$ then $Q \in s$ and $Q \in \mathcal{I}$. So, we have proved that

$$Q \in (s \cap s' \cap \mathcal{I}) \Leftrightarrow Q \leftarrow prev(Q) \in I \setminus Rej^S(s^*) \ \wedge \ prev(Q) \in Prev(s)$$

Let us now consider the negative fluent literals. In this case we want to prove that, for any inertial fluent, the following equivalence holds.

$$not\ Q \in (s \cap s') \Leftrightarrow not\ Q \in Default(s', s \cup \mathcal{R} \cup D(s,K))|\mathcal{F}$$

We know *not* $Q \in s'$ iff $Q \notin s'$, which, since s' is a odel of $\mathcal{R} \cup D(s,K)$, i plies that there exists no rule in $\mathcal{R} \cup D(s,K)$ whose head is Q and whose body is satisfied by s'. This, together with the fact that $Q \notin s$, by definition of $Default$ i plies that *not* $Q \in Default(s', s \cup \mathcal{R} \cup D(s,K))$, as desired.

$\Leftarrow$ Let us now suppose that s' satisfies the equivalence (6). i.e.

$$s' = least\left((s \cap s' \cap \mathcal{I}) \ \cup Default(s', \mathcal{R} \cup D(s,K))|_{(\mathcal{F}_L \setminus \mathcal{I})} \ \cup D(s,k) \ \cup \mathcal{R}\right)$$

Let NED be the set of literals of the for $\neg event(\tau)$ such that $event(\tau) \in ED(s,K)$ and there is no dyna ic rule of the for **effect**$(\tau) \leftarrow Cond$ such that s' satisfies $Cond$. Let s' be the following evolving interpretation (again we o it in the interpretation, the negative literals which are not fluent literals).

$$s^* = s' \ \cup \ Prev(s) \ \cup \ ED(s,K) \ \cup NED \cup assert(ED(s',K)) \cup \\ \cup\ assert(Prev(s))'$$

We have to prove that s^* is a refined stable odel of $\mathcal{J}$. We start this proof by showing that

$$Inherit(s^*) \cup Default(s^*)|_{\mathcal{I}} \equiv (s \cap s' \cap \mathcal{I})$$

We start by assu ing that Q is a fluent literal in $(s \cap s' \cap \mathcal{I})$. Q is such a fluent iff $Prev(Q) \in Prev(s)$, and *not* $Q \notin s'$. Since s' is a odel of $\mathcal{R} \cup D(s,K)$, we conclude that there exists no rule in $\mathcal{R} \cup D(s,K)$ with head *not* Q and true body in s'. Thus, the rule $Q \leftarrow prev(Q) \in I \setminus Rej^S(s^*)$, and hence $Q \in Inherit(s^*)$.

Let assu e now $Q \in Inherit(s^*)$ (i.e. $Q \leftarrow prev(Q) \in I \setminus Rej^S(s^*)$ and $prev(Q) \in Prev(s)$) then $Q \in s$. This i plies that *not* $Q \notin s$, $Q \in \mathcal{I}$, and there exists no rule in $\mathcal{R} \cup D(s, K)$ with head Q whose body is true in s'. Consequently, *not* $Q \notin s'$ (i.e. $Q \in s'$), and finally $Q \in (s \cap s' \cap \mathcal{I})$.
Let us now consider the negative fluent literals. We want to prove that, for any inertial fluent, the following equivalence holds.

$$not\ Q \in (s \cap s') \Leftrightarrow not\ Q \in Default(s', s \cup \mathcal{R} \cup D(s, K))|\mathcal{F}$$

The proof proceeds in the sa e way as above, in order to conclude that

$$Inherit(s^*) \cup Default(s^*)|_{\mathcal{I}} \equiv (s \cap s' \cap \mathcal{I})$$

We obtain then the following equivalence

$$s' = least \begin{pmatrix} Inherit(s^*) \cup Default(s^*)|_{\mathcal{I}} \cup \\ Default(s', \mathcal{R} \cup D(s, K))|_{(\mathcal{F}_L \setminus \mathcal{I})} \ \cup D(s, k) \ \cup \mathcal{R} \end{pmatrix}$$

which is equivalent to

$$s' = least\,(\ Inherit(s^*) \cup Default(s^*) \ \cup D(s, k) \ \cup \mathcal{R})\,|_{\mathcal{F}_L}$$

Since s' is consistent wrt. $D(s, K)$ and $\mathcal{R}$, these sets of rules do not contain any pair of rules with conflicting heads and whose bodies are both true in s'. So, by replacing $Inherit(s^*)$ with $Prev(s) \cup I \setminus Rej^S(s^*)$ we obtain

$$s' = least\left(\ (I \cup D(s, K) \cup mR) \setminus Rej^S(s^*) \cup Default(s^*)\right)|_{\mathcal{F}_L}$$

and fro this, and by considering the definition of s^*

$$s^* = least\left((I \cup D \cup Prev(s) \cup D(s, K)) \setminus Rej^S(s^*) \cup Default(s^*)\right)$$

This equation is, by definition, equivalent to say that M_1, s^* is an evolving stable odel of $I \oplus D$ given the sequence of events $K, \emptyset$. In other words, s' is a resulting state fro s given $I \oplus D$ and the set of actions K.

In the extre e cases where the set of inertial fluents coincides with the whole set of fluents and, when the set if inertial fluents is e pty, we obtain two si - plifications of the equivalence (6).

Corollary 1. *Let $I \oplus D$ be any EAP, s a state of the world and K a set of actions. Let $\mathcal{R}$, $D(s, K)$ be as in theorem 5. Moreover let every fluent be an inertial fluent. Then s' is a resulting state from s given $I \oplus D$ and the set of actions K iff*

$$s' = least\,(s \cap s') \ \cup D(s, k) \ \cup \mathcal{R})$$

Proof. Follows trivially as a special case of theore 5.

Corollary 2. *Let $I \oplus D$ be any EAP, s a state of the world and K a set of actions. Let $\mathcal{R}$, $D(s,K)$ be as in theorem 5. Moreover let the set of inertial fluents be the empty set. Then s' is a resulting state from s given $I \oplus D$ and the set of actions K iff s' is a stable model of the logic program $D(s,k) \cup \mathcal{R}$*

Proof. It follows trivially as a special case of theore 5 that

$$s' = least\left(Default(s', \mathcal{R} \cup D(s,K))|_{(\mathcal{F}_L \setminus \mathcal{I})} \cup D(s,k) \cup \mathcal{R}\right)$$

As proved in [19] this a ount to say s' is a stable odel of $D(s,k) \cup \mathcal{R}$.

Having shown this alternative to the definition of the transition function of EAPs, and proven its equivalence to the original Definition 6, we are now ready to prove all of the theore s (that we recall here, for the sake of readability) in this paper.

Theorem 1 (Complexity of EAPs). *Let $I \oplus D$ be any EAP over $(\mathcal{F}, \mathcal{A})$, s a state of the world and $K \subseteq \mathcal{A}$. To find a resulting state s' from s given $I \oplus D$ and the set of actions K is an NP-complete problem.*

Proof. By corollary 2, and given that the proble of finding a stable odel of a progra is NP-hard, we conclude that finding a resulting state s' fro s given $I \oplus D$ and the set of actions K is an NP-hard proble .

As for e bership, fro theore 5 and fro the observation that the co - putation of $least(P)$, where P is a logic progra , is polyno ial wrt. the nu ber of rules in P (since $least(P)$ is the least Herbrand odel of P considering the negative literals in P as new ato s), it follows that checking whether a given state s' is resulting state is a polyno ial proble wrt. the nu ber of rules in $I \oplus D$ plus the nu ber of ele ents in $\mathcal{F} \cup \mathcal{A}$. Hence, the proble of finding a resulting state s' fro s given $I \oplus D$ and the set of actions K is NP.

Theorem 2 (Relation to $\mathcal{B}$). *Let P be any $\mathcal{B}$ program with set of fluents $\mathcal{F}$, $(I^B \oplus D^{BP}, \mathcal{F})$ its translation, s a state and K any set of actions. Then s' is a resulting state from s given P and the set of actions K iff it is a resulting state from s given $I^B \oplus D^{BP}$ and the set of actions K.*

Proof. It trivially follows fro corollary 1.

Theorem 3 (Relation to $\mathcal{C}$). *Let P be any action program in the definite fragment of $\mathcal{C}$ with set of fluents $\mathcal{F}$, $(I^C \oplus D^{CP}, \mathcal{F}^C)$ its translation, s a state, s^C the interpretation over $\mathcal{F}^C$ defined as follows: $s^C = s \cup \{Q_N \mid Q \in s\} \cup \{not\ Q_N \mid not\ Q \in s\}$ and K any set of actions. Then s^* is a resulting state from s^C given $I^C \oplus D^{CP}$ and the set of actions K iff there exists s' such that s' is a resulting state from s, given P and the set K and $s^* \equiv_{\mathcal{F}_L} s'$.*

Proof. By corollary 2, s^* is a resulting state fro s^C given $I^C \oplus D^{CP}$ and the set of actions K iff s' is a stable odel of the progra $\mathcal{R} \cup D(s,K)$ where $\mathcal{R}$ and $D(s^C,K)$ are defined as in theore 5. Fro the translation of definite causal theories into logic progra s presented in [15], it follows that this is equivalent to say that s' is a odel of the causal theory obtained by all the static rules of P plus the rules of the for **caused** J **if** H for which a dyna ic rule

$$\textbf{caused } J \textbf{ if } H \textbf{ after } O$$

belongs to P and Q is true in $s \cup K$. This, in turn, is equivalent to saying that s' is a resulting state fro s given P and the set of actions K, as desired.

Theorem 4 (Simplification of updated EAPs). *Let $I_0 \cup I \oplus D_0 \oplus D_1 \oplus \ldots \oplus D_k$ be any update EAP over $(\mathcal{F}, \mathcal{A})$. Let $\bigoplus E_i^n$ be a sequence of events such that: $E_1 = K_1 \cup s$, where s is any state of the world and K_1 is any set of actions; and the others E_is are any set of actions K_α, or any set assert(**initialize($\mathcal{F}_\beta$)**) where $\bigcup \mathcal{F}_\beta \equiv I$, or any assert($D_i$) with $1 \leq i \leq k$. Let $s_1, \ldots, s_n$ be a sequence of possible resulting states from s given the EAP $I_0 \oplus D_0$ and the sequence of events $\bigoplus E_i^n$ and K_{n+1} a set of actions. Then $s_1, \ldots, s_n, s'$ is a resulting state from s given $I_0 \oplus D_0$ and the sequence of events $\bigoplus E_i^n \oplus K_{n+1}$ iff s' is a resulting state from s_n given $I_0 \cup I \oplus D_0 \oplus D_1 \oplus \ldots \oplus D_k$ and the set of actions K_{n+1}.*

Proof. The sequence $s_1, \ldots, s_n, s'$ is a sequence of possible resulting states iff there exists a sequence of evolving interpretations $M_0, M_1, \ldots M_n, s^*$ such that $M_0|_\mathcal{F} \equiv s$, $M_i|_\mathcal{F} \equiv s_i$ and $s^*|_\mathcal{F} \equiv s'$. The trace of $M_0, M_1, \ldots M_n, s^*$ is the DLP $I_0 \oplus D_0 \oplus T_1 \ldots \oplus T_n$ where each $T_i s$ is a set of literal of one of the following for s:

$$T_i = Aux_i$$
$$T_i = Aux_i \cup \textbf{initialize(}\mathcal{F}_\beta\textbf{)}$$
$$T_i = Aux_i \cup D_j] \text{ for so e } 0 \leq j \leq k$$

and Aux_i is a set of auxiliary literals of the for $Prev(Q)$ or *not* $Prev(Q)$, where Q is an inertial literal or *event*(τ) or *not event*(τ), τ being the effect of so e dyna ic rule.

To co pute s^*, the only relevant part of the trace is for ed by the various **initialize($\mathcal{F}_\beta s$)**, $D_k s$ and the last set of auxiliary literals Aux_n. Moreover, the se antics does not change if we put the various **initialize($\mathcal{F}_\beta s$)** in the first progra of the sequence, since a fluent only appears in a D_j after being initialized. Hence we can si plify the trace of $M_0, M_1, \ldots M_n, s^*$ into:

$$I_0 \cup I \oplus D_0 \oplus D_1 \oplus \ldots \oplus D_k \cup Aux_n$$

The set Aux_n can be split in three separate sets

$$Aux_n = Prev(s_n) \cup ED(s_n, K) \cup Retract(s_n)$$

where $Prev(s_n)$ and $ED(s_n, K)$ are as defined in the proof of theore 5 and $Retract(s_n)$ is the set of all literals of the for *not event*(τ) co ing fro dyna ic rules whose preconditions are true in s_{n-1} and false in s_n. The negative literals in $Retract(s_n)$ si ply rejects facts of the for *event*(τ) fro Aux_{n-1}. Since we have already si plified the trace by erasing all the $Aux_i s$ with $i < n$, we can ignore the set $Retract(s_n)$. Thus, we obtain that $s_1, \ldots s'$ is a sequence of possible resulting states iff an interpretation s^*, with $s^*|_{\mathcal{F}_L} \equiv s'$, is a refined stable odel of $I_0 \cup I \oplus D_0 \oplus D_1 \oplus \ldots \oplus D_k \oplus ED(s_n, K) \cup Prev(s_n)$. This is equivalent to saying that s' is a resulting state fro s given $I_0 \cup I \oplus D_0 \oplus D_1 \oplus \ldots \oplus D_k$ and the set of actions K_{n+1}, as desired.

Dynamic Logic Programming: Various Semantics Are Equal on Acyclic Programs

M. Homola

Comenius University, Bratislava, Slovakia
homola@tbc.sk

Abstract. Multidimensional dynamic logic programs (MDLPs) are suitable to represent knowledge dynamic in time, or more generally, information coming from various sources, partially ordered by arbitrary relevancy relation, e.g., level of authority. They have been shown useful for modeling and reasoning about multi-agent systems. Various approaches to define semantics of MDLPs have been presented. Most of the approaches can be characterized as based on rejection of rules.

It is understood that on some restricted classes of MDLPs several of these semantics coincide. We focus on acyclic programs. We show that for a MDLP $\mathcal{P}$ and a candidate model M, if $\mathcal{P}$ is acyclic to some extent then several of the known semantics coincide on M. It follows as a direct consequence that on the class of acyclic programs all of these semantics coincide.

1 Introduction

Background. In *Multidimensional Dynamic Logic Programs* (*MDLPs*), introduced in [1], knowledge is encoded into several logic programs, partially ordered by a relevance relation. MDLPs have been shown as well suited for representing knowledge change in time, and as well, to provide favourable representation for reasoning over information structured by some relevancy relation, such as authority hierarchies.

Already in [1], authors have shown that MDLPs are useful to model and reason about multi-agent systems. Particularly in logic based multi-agent systems where knowledge of an agent is naturally represented by rules. Thus, knowledge associated with an agent at a given state is encoded into a logic program. Assume that the agent's knowledge evolves with time. With each new time-state new knowledge appears to the agent, in form of rules, perceived trough sensors or communicated with other agents. This new knowledge may be in general contrary to the knowledge inherited from the previous time-states. We want the agent to be able to resolve such conflicts, assigning more relevance to the more recent knowledge.

MDLPs allow us to do this in a natural way. Agent's initial state and subsequent perceptions are modeled as a sequence of logic programs. More recent information is treated as more relevant. MDLPs assign semantics to the sequence,

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 78–95, 2005.

resolving conflicts between rules according to their relevancy. Moreover, they enable for determining semantics of the agent's knowledge at arbitrary state, thus allowing us to query the agent's knowledge history.

Besides time, MDLPs are capable of handling other relevancy relations, like specificity of the information or authority. This is particularly handy in multi-agent communities where an authoritative hierarchy among the agents is present. Assume that the knowledge of each agent is represented by a logic program. If an agent is authoritatively superior to the another one, we treat also the program of the former one as more relevant than the program of the latter one. Assuming that the agents obey the authority, we are able to query the global knowledge of the system but as well the knowledge of a subsystem rendered by an agent together with all the agents that are inferior to it.

Moreover, the framework allows us to combine several "relevancy dimensions" into a single MDLP. Thus, we are able to model, e.g., the knowledge distributed over an authority-enabled community of agents and as well the change of the whole system in time. Hence, we favor MDLPs as a powerful framework for modeling and reasoning about knowledge distributed over multi-agent systems, logic-based in particular. However, a multi-agent system does not have to be associated with a single MDLP, nor the view provided by the MDLP has to be global. For instance, each agent may use a MDLP to maintain its own view of the system, reflecting its own preference amongst the chunks of information obtained by communication with other agents. Thus, MDLPs may also provide a local knowledge repository for each agent of the system. For a more detailed analysis, we refer the reader to [3, 1, 11, 12]. We also refer the reader to [13], in order to see how extensions of MDLPs can benefit to multi-agent systems, and to [14] to see how the knowledge of multiple agents can be combined when there is no authoritative order among the agents.

Motivation. Various approaches have been presented in order to provide a semantics of MDLPs. Most of these semantics are based on similar notions (e.g., generalization of stable model semantics, employing rejection of rules) and are very close, one to another. Such semantics include $\mathcal{P}$-Justified Update semantics introduced in [2, 3], Dynamic Stable Model semantics from [4, 1], Update Answer Set semantics from [5, 3] and Refined Dynamic Stable Model semantics of [6, 7]. (The latest one is only known for linearly ordered MDLPs.) Usually, a new semantics has been introduced to cope with drawbacks of the older ones. Most important contributions are those of Leite [3, 8], Eiter et al. [5, 9] and Alferes et al. [4, 6].

Typically, semantics assigns a set of models to a program. Models are picked among the interpretations of the program. Authors point out that for some particular pairs of semantics, for a given MDLP, the model-set of one semantics is always a subset of the model-set of the other one. Thus, a sort of hierarchy of the model-sets assigned to a MDLP by different semantics is organized (cf. [3, 5, 6, 10]).

Studying the differences and similarities between these semantics, helps us to evaluate them w.r.t. our intuitions. Perhaps we do not need such a rich family of semantics, indeed if the difference between them shows to be very small. Particularly, within the field of multi-agent systems, it helps us to determine whether or not MDLPs are appropriate for a particular application, and if yes, which semantics to choose.

Also, it is a shared opinion, that on "plain" MDLPs, which are not obfuscated with cyclic dependencies (cyclic chains of rules), conflicting rules within a same logic program and other unconvenient constructs, all of these semantics coincide. Different behavior on some "abnormal" MDLPs is usually assigned to the inability of some of the semantics to deal with these abnormalities. Several restrictive conditions on MDLPs have been introduced in order to identify classes of programs on which two or more semantics coincide (cf. [3, 5, 6]). From this point of view, We find several results of [5, 3, 7], about restricted classes of MDLPs on which some of the semantics coincide, not tight, as many MDLPs on which the semantics also match are beyond the proposed classes.

We focus on a hypothesis that has been sketched already (cf. [5, 3, 7]), that perhaps on MDLPs that do not contain cycles several of the semantics may coincide. We see this hypothesis as valuable, since acyclic programs form a broad subclass and it is known that for some, simpler, applications they are sufficient. So, we suggest further evaluation of these semantics w.r.t. the class of acyclic programs and programs with limited occurrence of cyclic dependencies.

Contribution. As in [3, 15], we build MDLPs over a more general language of generalized extended logic programs that unifies the previous approaches under a common framework, allowing for more elegant comparisons, while keeping the previous approaches as special cases, so the results are propagated.

We introduce a new concept of sufficient acyclicity. Logic program is sufficiently acyclic if each of its literals is supported by at least one acyclic derivation. As the main result we establish a restrictive condition, using the notion of sufficient acyclicity, under which four (five) of the semantics coincide on the given interpretation of the given MDLP (linear MDLP). It trivially follows that on acyclic programs these semantics coincide entirely. This article presents the results of the author's Master's thesis [10] that can be viewed as its extended version.

2 Preliminaries

We first introduce basic concepts from logic programming. Logic programs are build from propositional *atoms*. The set of all atoms is denoted by $\mathcal{A}$. We employ two kinds of negation, *explicit negation* $\neg$ and *default negation* *not*. Let p be a proposition. By $\neg\, p$ we intuitively mean that (we know that) A is not true. Default negation is sometimes called negation as failure. We use it to express lack of objective evidence: by *not* p we intuitively mean that we have no evidence confirming that p is true.

An *objective literal* is an atom or an atom preceded by explicit negation (e.g., $A \in \mathcal{A}$ and $\neg\, A$ are objective literals). A *default literal* is an objective literal preceded by default negation (e.g., $not\ A$, $not \neg A$ are default literals, $A \in \mathcal{A}$). Both objective literal and default literal are *literals*. We denote the set of all objective literals by $\mathcal{O}$, the set of all default literals by $\mathcal{D}$ and the set of all literals by $\mathcal{L}$.

A *rule* is a formula $L \leftarrow L_1, \ldots, L_n$, where $n \geq 0$ and $L, L_1, \ldots, L_n \in \mathcal{L}$. A rule of a form $L \leftarrow$ (i.e., $n = 0$) is called a *fact*. For each rule r of a form $L \leftarrow L_1, \ldots, L_n$ we call the literal L the *head* of r and denote it by $h(r)$ and we call the set $\{L_1, \ldots, L_n\}$ the *body* of r and denote it by $b(r)$.

A set of rules P is called a *generalized extended logic program* (*logic program*, *GELP*). GELPs are the most general logic programs that we use. We favor the approach outlined in [3, 15], where MDLPs are built over GELPs, unifying the previous approaches under a common framework, allowing for more elegant comparisons, while keeping the previously used languages as special cases, so the results are propagated. We also remark, that GELPs enable to properly manipulate three truth values, "something is true", "something is false", and "we do not know", allowing to adequately switch from one to another, what we mark as a desirable feature, once dealing with knowledge updates.

Several other flavours of logic programs do exist. We mention *extended logic programs*, a subclass of GELPs formed by programs that do not contain default literals in heads of rules. *Generalized logic programs* do not allow explicit negation at all, i.e., for each objective literal L, contained in the program, it holds that $L \in \mathcal{A}$, and for each default literal $not\ L$, contained in the program, it holds that $L \in \mathcal{A}$. A logic program is *definite* if it only contains atoms of $\mathcal{A}$ in the heads, as well as in the bodies of its rules, i.e., definite logic programs do not allow negation at all.

Let P be a GELP. The expanded version of P is the program $\dot{P} = P \cup \{not \neg h(r) \leftarrow b(r) \mid r \in P \wedge h(r) \in \mathcal{O}\}$. Two literals $L \in \mathcal{O}$ and $not\ L$ are said to be *conflicting*. Two rules are conflicting if their heads are conflicting literals. We denote this by $L \bowtie L'$ and by $r \bowtie r'$ respectively. For any set of literals S, $S^+ = S \cap \mathcal{O}$ and $S^- = S \cap \mathcal{D}$.

A set of literals that does not contain a pair of conflicting literals is called an *interpretation*. An interpretation is *total* if for each $L \in \mathcal{O}$ it contains L or $not\ L$. A literal L is *satisfied* in an interpretation I if $L \in I$ and we denote it by $I \models L$. Also $I \models S$, a set of literals S, if $I \models L$ for each $L \in S$. A rule r is satisfied in an interpretation I (denoted by $I \models r$) if $I \models h(r)$ whenever $I \models b(r)$. Let P be a definite logic program. We denote by $least(P)$ the unique *least model* of P that exists, as showed by van Emden and Kowalski in [16].

Most of the semantic approaches in dynamic logic programming build on ideas of the stable model semantics of logic programs that has been introduced by Gelfond and Lifschitz in [17]. According to this semantics a total interpretation M is a stable model of a GELP P if it holds that $M = least(P \cup M^-)$[1].

[1] With an abuse of notation, we commonly treat (sets of) literals as (sets of) facts, and also GELPs as definite programs, considering each negated literal as a new atom.

3 MDLPs and Various Semantics Based on Rejection of Rules

Logic progra s have been proven useful in the area of knowledge representation. As long as the infor ation we deal with is rather static we face no proble to encode it in for of a logic progra . But we reach the barrier very soon, when dealing with infor ation change in ti e, or when integrating infor ation fro several sources with various levels of relevancy.

To deal with this proble , the fra ework of dyna ic logic progra ing has been introduced in [4]. In this fra ework infor ation is encoded into several progra s that are linearly ordered into a sequence by their level of relevancy. Such sequences are called dyna ic logic progra s.

This fra ework has been further generalized in [1] by allowing logic progra s ordered by arbitrary (i.e., also non-linear) partial ordering. Multidi ensional dyna ic logic progra s were born. We for alize the latter approach in Definition 1.

Definition 1. *Let $G = (V, E)$ be a directed acyclic graph with finite set of vertices V. Let $\mathcal{P} = \{P_i \mid i \in V\}$ be a set of logic programs. The pair $(\mathcal{P}, G)$ is a* ultidi ensional dyna ic logic progra *or often just* progra *or* MDLP*.*

We often use just $\mathcal{P}$ instead of $(\mathcal{P}, G)$ and assu e the existence of the corresponding G. The ultiset of all rules of the expanded versions $\dot{P}_i$ of the logic progra s P_i, $i \in V$ of $\mathcal{P}$ is denoted by $\uplus_{\mathcal{P}}$. Let $i, j \in V$, we denote by $i \prec j$ (and also by $P_i \prec P_j$) if there is a directed path fro i to j in G. We denote by $i \preceq j$ (and by $P_i \preceq P_j$) if $i \prec j$ or if $i = j$.

A *dynamic logic program* (*DLP*, *linear* MDLP) is such a MDLP $\mathcal{P}$ whose G is collapsed into a single directed path. So, DLPs for a subclass of MDLPs, they are precisely all linearly ordered MDLPs.

Most of the se antic approaches in dyna ic logic progra ing are based on the ideas of stable odel se antics of si ple logic progra s. A set of odels is assigned to a progra by each of these se antics. Models are picked a ong the interpretations of the progra .

As a MDLP in general ay contain conflicting rules, se antics try to resolve these conflicts, when it is possible, according to the relevancy level of the conflicting rules. A co on approach is to assign a set of *rejected rules* to a given progra $\mathcal{P}$ and a "candidate odel" interpretation M. Rejected rules are then subtracted fro the union of all rules of $\mathcal{P}$, gaining the residue of $\mathcal{P}$ w.r.t. M. Also the set of *default assumptions* (so eti es just *defaults*) is assigned to $\mathcal{P}$ and M. Defaults are picked a ong the default literals. A fix-point condition is verified, whether M coincides with the least odel of the union of the residue and the default assu ptions. If so, then M is a odel of P w.r.t. the se antics. A se antics that can be characterized in this anner is said to be *based on rejection of rules* or *rule-rejecting*.

Once we deal with several rule-rejecting se antics, then any difference between the originates in the way how particularly rejection of rules and default assumptions are implemented in these semantics. Two different kinds of rejection have been used with MDLPs. The original rejection used in [4, 1] keeps each rule intact as long as there is no reason for rejecting it in form of a more relevant rule that is satisfied in the considered interpretation. Formally, the set of rejected rules of $\mathcal{P}$ w.r.t. M is

$$Rej(\mathcal{P}, M) = \{r \in \dot{P}_i \mid (\exists r' \in \dot{P}_j)\ i \prec j, M \models b(r'), r \bowtie r'\} \ .$$

In [5], an alternative notion of rejection has been introduced, allowing each rule to reject other rules only if it is not rejected already. Such a set of rejected rules of $\mathcal{P}$ w.r.t. M is formalized as

$$Rej^{\star}(\mathcal{P}, M) = \{r \in \dot{P}_i \mid (\exists r' \in \dot{P}_j)\ i \prec j, M \models b(r'), r \bowtie r', r' \notin Rej^{\star}(\mathcal{P}, M)\} \ .$$

Originally, in [2], default assumptions have been computed just exactly as in the stable model semantics of logic programs. Formally,

$$Def^{\star}(\mathcal{P}, M) = M^{-} \ .$$

Later on, in [4, 1], another approach has been introduced, as the original set of defaults showed to be too broad. We formalize defaults according to this approach as

$$Def(\mathcal{P}, M) = \{not\ L \mid L \in \mathcal{O}, (\nexists r \in \Cup_{\mathcal{P}})\ h(r) = L, M \models b(r)\} \ .$$

Combining two implementations of rejection and two of default assumptions immediately leads to four semantics of MDLPs. We define each of them formally in the following.

Definition 2. *A rule-rejecting semantics that uses $Rej(\mathcal{P}, M)$ for rejection and $Def^{\star}(\mathcal{P}, M)$ for defaults is called the* dynamic justified update (DJU) *semantics. That is, a total interpretation M is a model of a MDLP $\mathcal{P}$ w.r.t. the DJU semantics whenever $M = least(Res(\mathcal{P}, M) \cup Def^{\star}(\mathcal{P}, M))$, where $Res(\mathcal{P}, M) = \Cup_{\mathcal{P}} \setminus Rej(\mathcal{P}, M)$ is the residue.*

The DJU semantics is the very first rule-rejecting semantics that has been used in dynamic logic programming. If we restrict to DLPs build from generalized logic programs, it is identical with the $\mathcal{P}$-justified updates semantics of [2]. Soon the original default assumptions showed to be too broad. In [4, 1], they have been replaced by $Def(\mathcal{P}, M)$. The semantics is formally defined as follows.

Definition 3. *A rule-rejecting semantics that uses $Rej(\mathcal{P}, M)$ for rejection and $Def(\mathcal{P}, M)$ for defaults is called the* dynamic stable model (DSM) *semantics. Or equivalently, a total interpretation M is a model of a MDLP $\mathcal{P}$ w.r.t. the DSM semantics whenever $M = least(Res(\mathcal{P}, M) \cup Def(\mathcal{P}, M))$, where the residue is as in Definition 2.*

In [5], the alternative notion of rejection, $Rej^{\star}(\mathcal{P}, M)$, has been combined with $Def^{\star}(\mathcal{P}, M)$ to produce semantics for DLPs build from extended logic programs. The semantics has been originally called the update answer set semantics. In our setting we formalize it in Definition 4.

Definition 4. *A rule-rejecting semantics that uses* $Rej^{\star}(\mathcal{P}, M)$ *for rejection and* $Def^{\star}(\mathcal{P}, M)$ *for defaults is called the* backward dynamic justified update (BDJU) *semantics. In other words, a total interpretation* M *is a model of a MDLP* $\mathcal{P}$ *w.r.t. the BDJU semantics whenever* $M = least(Res^{\star}(\mathcal{P}, M) \cup Def^{\star}(\mathcal{P}, M))$*, where* $Res^{\star}(\mathcal{P}, M) = \mathbb{U}_{\mathcal{P}} \setminus Rej^{\star}(\mathcal{P}, M)$ *is the residue.*

By the label "backward" we indicate use of $Rej^{\star}(\mathcal{P}, M)$ rejection, as the algorithm for its computation from [5] traverses $\mathcal{P}$ in backward direction compared to the one for $Rej(\mathcal{P}, M)$ found in [4,1]. In [3], the three above mentioned semantics have been brought to a more general platform offered by GELPs. Also a backward variant of the DSM semantics has been introduced, that we formalize in Definition 5. In [3], this semantics is called the U-model semantics.

Definition 5. *A rule-rejecting semantics that uses* $Rej^{\star}(\mathcal{P}, M)$ *for rejection and* $Def(\mathcal{P}, M)$ *for defaults is called the* backward dynamic stable model (BDSM) *semantics. That is, a total interpretation* M *is a model of a MDLP* $\mathcal{P}$ *w.r.t. the BDSM semantics whenever* $M = least(Res^{\star}(\mathcal{P}, M) \cup Def(\mathcal{P}, M))$*, where the residue is as in Definition 4.*

The set of all models of a program $\mathcal{P}$ w.r.t. the DJU semantics is denoted by $DJU(\mathcal{P})$. Similarly, $DSM(\mathcal{P})$, $BDJU(\mathcal{P})$ and $BDSM(\mathcal{P})$ are the sets of all models according to the remaining three semantics.

We have presented four rule-rejecting semantics of MDLPs. The following two examples taken from [3] show that each of this semantics is different.

Example 1. Let $\mathcal{P} = \{P_1 \prec P_2\}$ where $P_1 = \{a \leftarrow\ \}$, $P_2 = \{not\, a \leftarrow not\, a\}$. It holds that $DSM(\mathcal{P}) = BDSM(\mathcal{P}) = \{\{a, not\, \neg a\}\}$. But, for the other two, $DJU(\mathcal{P}) = BDJU(\mathcal{P}) = \{\{a, not\, \neg a\}, \{not\, a, not\, \neg a\}\}$.

Example 2. Let $\mathcal{P} = \{P_1 \prec P_2 \prec P_3\}$ where $P_1 = \{a \leftarrow\ \}$, $P_2 = \{not\, a \leftarrow\ \}$ and $P_3 = \{a \leftarrow a\}$. It holds that $DJU(\mathcal{P}) = DSM(\mathcal{P}) = \{\{not\, a, not\, \neg a\}\}$. On the other hand, $BDJU(\mathcal{P}) = BDSM(\mathcal{P}) = \{\{a, not\, \neg a\}, \{not\, a, not\, \neg a\}\}$.

Moreover, as it has been shown in [3], the sets of models assigned to arbitrary program $\mathcal{P}$, one set by each of these semantics, form a kind of hierarchy w.r.t. the set inclusion relation. The DSM semantics is the most restrictive one, the set of models w.r.t. DSM is always a subset of the other model-sets. On the other hand, the set of models w.r.t. any semantics is always a subset of the one w.r.t. BDJU, which always provides the broadest set of models. We summarize these observations in Theorem 1 taken from [3].

Theorem 1. *For each MDLP* $\mathcal{P}$ *it holds that*

$$DSM(\mathcal{P}) \subseteq DJU(\mathcal{P}) \subseteq BDJU(\mathcal{P}) \ ,$$
$$DSM(\mathcal{P}) \subseteq BDSM(\mathcal{P}) \subseteq BDJU(\mathcal{P}) \ .$$

4 Equality on the Class of Acyclic Programs

We have shown in Examples 1 and 2 that the four rule-rejecting semantics are in general distinct. However, many MDLPs exist, such as the one from Example 3, on which these four semantics coincide.

Example 3. Let $\mathcal{P} = \{P_1, P_2, P_3 \mid P_1 \prec P_3, P_2 \prec P_3\}$. Let $P_1 = \{a \leftarrow\}$, $P_2 = \{\mathit{not}\, a \leftarrow\}$ and $P_3 = \{a \leftarrow\}$. This simple MDLP can be viewed as a model of a community of three agents, who take part in the hierarchy of authorities. The first two of them are of incomparable authority and moreover, they have conflicting knowledge. This conflict is resolved by the third one of them, who is represented by logic program P_3 and its authority level is superior to the former two. All of the four semantics agree with this intuition and assign $M = \{a, \mathit{not}\, \neg a\}$ to $\mathcal{P}$ as its single model.

Examples like this one lead us to a hypothesis that there probably are vast classes of programs on which several semantics coincide. It shows that several rule-rejecting semantics possibly behave equally on "plain" programs, that are not obfuscated with cyclic dependencies among literals or other obstacles. Different behavior on such programs is supposed to be caused by different ability of the semantics to deal with such obstacles.

To evaluate cyclic dependencies among literals in programs we adopt the graph-theoretic framework introduced in [5]. An *AND/OR-graph* (N, C) is a hypergraph, whose set of nodes $N = N_A \uplus N_O$ decomposes into the set of *AND-nodes* N_A and the set of *OR-nodes* N_O, and its set of connectors $C = N \times \bigcup_{i=0}^{|N|} N^i$ is a function, i.e., for each $I \in N$ there is exactly one tuple $\langle O_1, \ldots, O_k \rangle$ s.t. $\langle I, O_1, \ldots, O_k \rangle \in C$. For any connector $\langle I, O_1, \ldots, O_k \rangle$, I is its *input node* and $O_1, \ldots, O_k$ are its *output nodes*.

Let (N, C) be an AND/OR-graph, $I \in N$ and $\langle I, O_1, \ldots, O_k \rangle \in C$. A tree p is a *path* in (N, C) *rooted in* I if one of the following conditions holds:

(i) $k = 0 \wedge p = \langle I \rangle$,
(ii) $k > 0 \wedge I \in N_A \wedge p = \langle I, p_1, \ldots, p_k \rangle$,
(iii) $k > 0 \wedge I \in N_O \wedge (\exists i)\ 1 \leq i \leq k \wedge p = \langle I, p_i \rangle$,

where p_i is a path in (N, C) rooted in O_i, $1 \leq i \leq k$.

Let $p = \langle I, p_1, \ldots, p_k \rangle$ be a path in an AND/OR-graph. A path p' is a *subpath* of p if $p' = p$ or p' is a subpath of p_i for some i, $1 \leq i \leq k$. A path p in an AND/OR-graph is said to be *acyclic* if for every subpath p' (including p) rooted in the node R, no subpath p'' of p' is rooted in R.

Definition 6. *Let P be a logic program. An AND/OR-graph $G_P = (N, C)$ is* associated *with P if both of the following conditions hold:*

(i) $N_A = P \wedge N_O = \mathcal{L}$,
(ii) $C = \{\langle r, L_1, \ldots, L_k \rangle \mid r = L \leftarrow L_1, \ldots, L_k \in P\}$
$\quad \cup \{\langle L, r_1, \ldots, r_n \rangle \mid \{r_1, \ldots, r_n\} = \{r \in P \mid h(r) = L\}\}$.

Ar ed with such a fra ework we instantly identify the class of acyclic progra s in Definition 7. Clearly, this definition is equivalent to the original one, as introduced in [18].

Definition 7. *We say that logic program P is* strictly acyclic *(or just* acyclic*) if G_P does not contain a path that is cyclic. We say that a MDLP $\mathcal{P}$ is strictly acyclic if $\biguplus_{\mathcal{P}}$ is strictly acyclic.*

In [5], further reduction of $G_{\mathcal{P}}$ is utilized, once an interpretation M and a given notion of rejection are available. The resulting reduced AND/OR-graph is stripped fro dependencies corresponding to rules that are rejected or that are not applicable.

Definition 8. *Let $\mathcal{P}$ be a MDLP, M a total interpretation and $Rejected(\mathcal{P}, M)$ a set of rejected rules according to some rule-rejecting semantics. The* reduced AND/OR-graph *of $\mathcal{P}$ with respect to M, $G_{\mathcal{P}}^{M}$ is obtained from $G_{\mathcal{P}}$ by*

1. *removing all $r \in N_A$ and their connectors (as well as removing r from all connectors containing it as an output node) if either $r \in Rejected(\mathcal{P}, M)$ or $M \nvDash b(r)$, and*
2. *replacing, for every $L \in \mathcal{O}$, the connector of not L by the 0-connector $\langle not\ L \rangle$, if L is associated with 0-connector after step 1 and no $r \in Rejected(\mathcal{P}, M)$ exists s.t. $h(r) = L$.*

Possessing the outlined fra ework, authors of [5] have introduced the "root condition" and the "chain condition", that we adopt in Definition 9 and 10 respectively.

Definition 9. *Let $\mathcal{P}$ be a MDLP, M a total interpretation and $Rejected(\mathcal{P}, M)$ a set of rejected rules according to some rule-rejecting semantics. We say that $\mathcal{P}$, M and $Rejected(\mathcal{P}, M)$ obey the* root condition *if, for each not $L \in M^-$, one of the following conditions holds:*

(i) $(\forall r \in \biguplus_{\mathcal{P}})\ h(r) = L \implies M \nvDash b(r)$,
(ii) there exists an acyclic path p in $G_{\mathcal{P}}^{M}$ rooted in not L.

Definition 10. *We say that a MDLP $\mathcal{P}$ and a total interpretation M obey the* chain condition *if, for each pair of rules $r \in P_i$, $r' \in P_j$ s.t. $i \prec j$, $r \bowtie r'$, $M \models b(r)$, $M \models b(r')$ and $r' \in Rej^{\star}(\mathcal{P}, M)$, there also exists $r'' \in P_s$ s.t. $j \prec s$, $r' \bowtie r''$ and $b(r'') \subseteq b(r)$.*

A theore follows in [5], stating that if both, the root and the chain condition, are satisfied by a DLP $\mathcal{P}$, a total interpretation M and $Rej(\mathcal{P}, M)$ then $M \in DSM(\mathcal{P})$ if and only if M is a odel of $\mathcal{P}$ (both transfor ed to extended logic progra s) w.r.t. the BDJU se antics.

In [3] relations between all four of these se antics are further investigated, once all four are generalized to the platfor of GELPs. It is shown there, that the root condition renders a proper subclass of DLPs, in order to co pare two se antics that utilize $Def(\mathcal{P}, M)$ and $Def^{\star}(\mathcal{P}, M)$ for defaults respectively, and share the same implementation of rejection. We adopt this proposition from [3] and generalize it to the platform of MDLPs in Theorem 2. In [3] it is also shown that two pairs of semantics that differ in rejection but use the same defaults, pairwise, coincide on a DLP $\mathcal{P}$ and a total interpretation M if they obey the chain condition. We adopt this proposition in Theorem 3.[2]

Theorem 2. *Let $\mathcal{P}$ be a MDLP, M a total interpretation. Then it holds that:*

(i) $M \in DJU(\mathcal{P}) \equiv M \in DSM(\mathcal{P})$ *if and only if* $\mathcal{P}$, M *and* $Rej(\mathcal{P}, M)$ *obey the root condition,*
(ii) $M \in BDJU(\mathcal{P}) \equiv M \in BDSM(\mathcal{P})$ *if and only if* $\mathcal{P}$, M *and* $Rej^{\star}(\mathcal{P}, M)$ *obey the root condition.*

Theorem 3. *Let $\mathcal{P}$ be a MDLP, M a total interpretation. If $\mathcal{P}$ and M obey the chain condition then each of the following propositions holds:*

(i) $M \in DJU(\mathcal{P}) \equiv M \in BDJU(\mathcal{P})$,
(ii) $M \in DSM(\mathcal{P}) \equiv M \in BDSM(\mathcal{P})$.

It follows in [3], that if both of the conditions are obeyed by $\mathcal{P}$ and M, then all four of the semantics coincide on $\mathcal{P}$ and M. However, as we show in Example 4, many times the chain condition is not obeyed but the semantics do coincide. We argue that this restriction is not accurate.

Example 4. Let $\mathcal{P} = \{P_1 \prec P_2 \prec P_3\}$, $P_1 = \{a \leftarrow \}$, $P_2 = \{not\, a \leftarrow \}$ and $P_3 = \{a \leftarrow not\, b\}$. The chain condition is not obeyed by $\mathcal{P}$ and $M = \{a, not\, b, not\, \neg a, not\, \neg b\}$. Yet, $DSM(\mathcal{P}) = BDSM(\mathcal{P}) = \{M\}$ and $DJU(\mathcal{P}) = BDJU(\mathcal{P}) = \{M\}$.

We now return to considerations about programs with restricted occurrence of cycles. We focus on a hypothesis that different behavior of semantics is always accompanied by presence of cyclic dependencies among literals. Our aim is to restrict somehow the occurrence of cyclic dependencies in order to establish the coincidence of the semantics.

Programs with cycles are often considered odd. Self-dependence, connected with presence of cycles, is marked as unpleasant and undesirable feature, as strict, deductive reasoning – closely interconnected with mathematical logic – forbids it. Yet, in logic programming cycles are useful, for example to express equivalence. Moreover there are programs that contain cycles and still different semantics match regarding them. Both of these features are apparent from Example 5. Hence we introduce yet another, weaker, condition of acyclicity in the consecutive Definition 11. With this condition, we are able to identify programs, where cycles may be present, but each literal is supported by at least one acyclic derivation.

---

[2] We remark that this property does not depend on the particular choice of defaults. In fact, it holds for arbitrary set of default assumptions. See [10] for details.

Example 5. Let $\mathcal{P} = \{P_1 \prec P_2\}$, $P_1 = \{a \leftarrow b; b \leftarrow a\}$ and $P_2 = \{a \leftarrow \}$. All of the four se antics atch on $\mathcal{P}$. $DJU(\mathcal{P}) = DSM(\mathcal{P}) = BDJU(\mathcal{P}) = BDSM(\mathcal{P}) = \{\{a, b, not\,\neg\, a, not\,\neg\, b\}\}$. Actually, the cyclic infor ation of progra P_1 is not redundant in any way. P_1 states that the truth value of a is equivalent with the truth value of b and vice versa. Later, when the ore recent knowledge of P_2 appears telling that a is true we derive that also b is true.

Definition 11. *We say that logic program P is* sufficiently acyclic *if for every literal $L \in \mathcal{L}$ there exists an acyclic path in the hypergraph G_P associated with P that is rooted in L. A MDLP $\mathcal{P}$ is sufficiently acyclic whenever $\Cup_{\mathcal{P}}$ is sufficiently acyclic.*

The application of the condition of sufficient acyclicity on MDLPs in general is, however, useless – as when the residue is co puted, several rules are retracted and the condition ay not be satisfied any ore. So we resort to the one- odel relations of two se antics quite like in the case of the root condition. The relation is established for a progra and a given odel. Possessing a candidate- odel, the residue is deter ined, and the condition is applied on the residue instead of the whole progra .

To establish one- odel equivalence of two se antics on a progra , we repeatedly use a ethod, that is sketched in Re ark 1.

Remark 1. Let $\mathcal{P}$ be a MDLP and let M be a total interpretation. Let S_1 and S_2 be two rule-rejecting se antics with shared i ple entation of defaults and different i ple entation of rejection. Let D be the set of defaults assigned to $\mathcal{P}$ and M by these se antics and let R_1 and R_2 be the residues assigned to $\mathcal{P}$ and M by S_1 and S_2 respectively. If

(i) $M \in S_2(\mathcal{P})$,
(ii) $R_1 \subseteq R_2$,

then $M \in S_1(\mathcal{P})$ if and only if there exists such $R \subseteq R_1$ that $M = least(R \cup D)$ – i.e., we are able to find R, a subset of R_1, s.t. R still contains enough of rules that are necessary to co pute M. Therefore we concentrate on searching for such sets $R \subseteq R_1$ in order to establish equivalence of S_1 and S_2 regarding $\mathcal{P}$ and M.

The condition for one- odel equality of that pairs of se antics that differ in the i ple entation of rejection and use sa e defaults is expressed in Theore 4. The theore uses the following le a.

Lemma 1. *Let S be the BDSM or the BDJU semantics. Let $\mathcal{P}$ be a MDLP, $M \in S(\mathcal{P})$ and let $Defaults(\mathcal{P}, M)$ be the default assumptions assigned to $\mathcal{P}$ and M by S. If the set R defined as*

$$R = \{r \mid r \in Res(\mathcal{P}, M) \wedge M \vDash b(r)\}$$

is sufficiently acyclic then M can be computed as a model in the given semantics using only the rules of R. That is, $M = least(R \cup Defaults(\mathcal{P}, M))$.

Proof. Since R is sufficiently acyclic, there exists a rule $r \in R$ such that for each $L \in b(r)$ for no $r' \in R$ holds $h(r') = L$. And $r \in R$ so it holds that $M \vDash b(r)$. From Definitions 2 and 4 and from how R is defined it follows that for each rule $q \in \mathit{Res}^\star(\mathcal{P}, M)$, $M \vDash b(q)$ there is a $q' \in R$ s.t. $h(q) = h(q')$ and since $M \in S(\mathcal{P})$ then $b(r) \subseteq \mathit{Defaults}(\mathcal{P}, M)$. We now construct

$$M^0 = \mathit{Defaults}(\mathcal{P}, M) \ , \qquad M^1 = M^0 \cup h(r) \ ,$$
$$R^0 = R \ , \qquad R^1 = R^0 \setminus \{r'' \mid h(r'') = h(r)\} \ .$$

Assume that M^j and R^j are constructed by adding one literal $L \in \mathcal{L}$ to M^{j-1} and removing all r'' from R^{j-1} such that $h(r'') = L$, $0 < j \leq i$. Again, as R is sufficiently acyclic, there is $r \in R^i$ s.t. for each $L \in b(r)$ for no $r' \in R^i$ holds $h(r') = L$. From the construction of D^i, R^i it follows that

$$(\forall j \leq i) \ M^j \cup \{h(r) \mid r \in R^j\} = M \ .$$

Therefore $b(r) \subseteq M^i$, and so we are able to construct

$$M^{i+1} = M^i \cup h(r) \ , \qquad R^{i+1} = R^i \setminus \{r'' \in R^i \mid h(r'') = h(r)\} \ .$$

It is straightforward that $\bigcup_{i=1}^{\infty} M^i = M$. This way we have computed M as a model in S only from the rules of R. (Step by step, we have simulated the iterations of the $\mathit{least}(\cdot)$ operator.) In other words,

$$M = \mathit{least}(R \cup \mathit{Defaults}(\mathcal{P}, M)) \ .$$

□

Theorem 4. *Let $\mathcal{P}$ be a MDLP and M be its total interpretation. If the set*

$$R = \{r \mid r \in \mathit{Res}(\mathcal{P}, M) \wedge M \vDash b(r)\}$$

is sufficiently acyclic then it holds that

(i) $M \in \mathit{DSM}(\mathcal{P}) \equiv M \in \mathit{BDSM}(\mathcal{P})$*, and also*
(ii) $M \in \mathit{DJU}(\mathcal{P}) \equiv M \in \mathit{BDJU}(\mathcal{P})$*.*

Proof. The only-if part of both (i) and (ii) follows from Theorem 1. The if part proves as follows. Let $\mathcal{P}$ be a MDLP. Let $M \in \mathit{BDSM}(\mathcal{P})$ ($\mathit{BDJU}(\mathcal{P})$ respectively). Let R be sufficiently acyclic. From Lemma 1 we get that M can be computed only using the rules of R. Since

$$R \subseteq \mathit{Res}(\mathcal{P}, M) \subseteq \mathit{Res}^\star(\mathcal{P}, M) \ ,$$

it follows from Remark 1 that $M \in \mathit{DSM}(\mathcal{P})$ ($M \in \mathit{DJU}(\mathcal{P})$). □

In Theorem 4 we have presented a restrictive condition for one-model equality of those pairs of semantics that differ in rejection and use same defaults. We now show (in Lemma 2) that under this condition also the root condition is satisfied. It follows as a direct consequence of this lemma and Theorem 4 that under our condition all four semantics coincide (Corollary 1).

Lemma 2. *Let $\mathcal{P}$ be a MDLP and M its total interpretation. Let*

$$R = \{r \mid r \in Res(\mathcal{P}, M) \wedge M \models b(r)\} \enspace .$$

If R is sufficiently acyclic then both of the triples $\mathcal{P}$, M, $Rej(\mathcal{P}, M)$ and $\mathcal{P}$, M, $Rej^\star(\mathcal{P}, M)$ obey the root condition.

Proof. R is sufficiently acyclic, hence for every $L \in M^-$ either $L \in Def(\mathcal{P}, M)$ and then condition (i) of Definition 9 (root condition) is satisfied or there exists a rule $r \in Res(\mathcal{P}, M)$ s.t. $M \models b(r)$ and $h(r) = L$ and therefore also $r' \in R$ s.t. $h(r') = L$ and so there is a path p in G_R rooted in L that is acyclic. The subpath p' of p, terminated in every $not\, L' \in \mathcal{D}$ whose connector was replaced by $\langle not\, L' \rangle$ in step 2 of the construction of $G^M_{\mathcal{P}}$, is an acyclic path in $G^M_{\mathcal{P}}$ rooted in L. And so condition (ii) of Definition 9 is satisfied. Hence the root condition is obeyed by $\mathcal{P}$, M and $Rej(\mathcal{P}, M)$.

As for each $r \in Res(\mathcal{P}, M)$, $M \models b(r)$ there exists such $r' \in Res^\star(\mathcal{P}, M)$ that $h(r') = h(r)$ and $M \models b(r')$ and vice versa, we get that also $\mathcal{P}$, M and $Rej^\star(\mathcal{P}, M)$ obey the root condition. □

Corollary 1. *Let $\mathcal{P}$ be a MDLP and M its total interpretation. If the set*

$$R = \{r \mid r \in Res(\mathcal{P}, M) \wedge M \models b(r)\}$$

is sufficiently acyclic then

$$M \in DSM(\mathcal{P}) \equiv M \in BDSM(\mathcal{P}) \equiv M \in DJU(\mathcal{P}) \equiv M \in BDJU(\mathcal{P}) \enspace .$$

Moreover, as for a strictly acyclic program each of its subsets is sufficiently acyclic, it trivially follows that all four semantics coincide on strictly acyclic programs as we state in the following corollary.

Corollary 2. *Let $\mathcal{P}$ be a strictly acyclic MDLP. Then*

$$DSM(\mathcal{P}) = BDSM(\mathcal{P}) = DJU(\mathcal{P}) = BDJU(\mathcal{P}) \enspace .$$

We have shown that the four rule-rejecting semantics coincide on strictly acyclic programs. In Corollary 1 we have also established a more accurate restriction that renders the one-model equivalence of the semantics. However, comparing entire model-sets assigned to a program by two semantics one by one is computationally as complex as computing and enumerating these two model-sets. So, this result is rather of theoretical value.

5 RDSM Semantics and DLPs

In [7], Alferes et al. have introduced a new semantics for linear DLPs. Motivation for this new semantics roots in the observation that even the most restrictive semantics, DSM, provides counterintuitive models for some programs (cf. Example 6).

Example 6. Let $\mathcal{P} = \{P_1 \prec P_2\}$ where $P_1 = \{a \leftarrow ; not\, a \leftarrow \}$ and $P_2 = \{a \leftarrow a\}$. It holds that $DSM(\{P_1\}) = \emptyset$, it is not surprising as P_1 is contradictory. If we inspect the single rule of P_2 we see that it actually brings no new factual information. We suppose that addition of such rule should not add new models to the program. However, $DSM(\mathcal{P}) = \{\{a, not\,\neg a\}\}$.

Such rules as the one of P_2 from Example 6, having head a subset of the body, are called *tautological.* Tautological rules are in fact just a special case of cycles that only span throughout one rule. In [7], authors have identified even broader class of extensions of DLPs that, according to their intuition, should not yield new models of the programs. Such extensions are called *refined extensions.* Then a principle has been formed, stating that, having a proper semantics, if a program $\mathcal{P}'$ is just a refined extension of $\mathcal{P}$ then it should not have a model that is not also a model of $\mathcal{P}$. This principle is called the *refined extension principle.* We refer the reader who is interested in precise definitions to [7].

In [7], also a modified DSM semantics has been introduced. The modification is slight, two conflicting rules of the same program are allowed to reject each other. Formally, the set of rejected rules of this semantics is

$$Rej^R(\mathcal{P}, M) = \{r \in \dot{P}_i \mid (\exists r' \in \dot{P}_j)\ i \preceq j, M \models b(r'), r \bowtie r'\} \ .$$

The semantics is formalized in Definition 12.

Definition 12. *A rule-rejecting semantics of DLPs that uses $Rej^R(\mathcal{P}, M)$ for rejection and $Def(\mathcal{P}, M)$ for defaults is called the* refined dynamic stable model (RDSM) *semantics. In other words, a total interpretation M is a model of a DLP $\mathcal{P}$ w.r.t. the RDSM semantics whenever $M = least(Res^R(\mathcal{P}, M) \cup Def(\mathcal{P}, M))$, where $Res^R(\mathcal{P}, M)$ is the residue.*

We agree with [7] that the RDSM semantics is very favourable. It has been shown in [7] that it satisfies the refined extension principle and, as we adopt in Theorem 5, it always yields such model-set that is a subset of the model-set w.r.t. the DSM semantics. Moreover, it has been precisely described and motivated in [7], why some models provided by DSM should be excluded.

Theorem 5. *For any DLP $\mathcal{P}$ it holds that $RDSM(\mathcal{P}) \subseteq DSM(\mathcal{P})$.*

In [7], it further has been shown that for a program $\mathcal{P}$ that does not contain a pair of conflicting rules in the very same $P_i \in \mathcal{P}$, the RDSM and the DSM semantics coincide. However, this result neither is tight as many programs exist s.t. DSM and RDSM coincide on them and the condition is not satisfied.

The RDSM semantics has been introduced only for linear DLPs and according to our deepest knowledge all attempts to generalize it for MDLPs have failed so far (cf. [19]). Hence, in this section, we restrict our considerations to linear DLPs. In the remaining we show that under a very similar restriction as the one of Corollary 1, for a given model, all five of the semantics coincide.

First of all, the following example demonstrates why the condition has to be altered.

Example 7. Recall again the progra $\mathcal{P}$ fro Exa ple 6. Let $M = \{a, not\,\neg a\}$. Even if $R = \{a \leftarrow, a \leftarrow a\}$ is sufficiently acyclic, $M \in DSM(P)$ and $M \notin RDSM(P)$. Indeed, the fact that $R \nsubseteq Res^R(\mathcal{P}, M)$ causes the trouble. The sufficient acyclicity is broken in $Res^R(\mathcal{P}, M)$ and therefore a can not be derived in the refined se antics.

The further restrictive condition is introduced in Theore 6, where we prove the one- odel coincidence of RDSM and DSM and we also confir that the propositions of Theore 4 hold under this odified condition as well. The theore uses the following le a.

Lemma 3. *Let semantics S be one of DSM, DJU, BDSM and BDJU. Let $\mathcal{P}$ be a DLP. Let $M \in S(\mathcal{P})$. Let Rejected$(\mathcal{P}, M)$ be the rejected rules, Residue$(\mathcal{P}, M)$ be the residue and Defaults$(\mathcal{P}, M)$ be the defaults assigned to $\mathcal{P}$ and M by S. If*

$$R' = \{r \mid r \in Res^R(\mathcal{P}, M) \wedge M \vDash b(r)\}$$

is sufficiently acyclic then M can be computed as a model in the given semantics using only the rules of R'. That is, $M = least(R' \cup Defaults(\mathcal{P}, M))$.

Proof. Fro Definitions 2, 4 and 12 and fro how R' is defined it follows that if $M \in S(\mathcal{P})$ then for each rule $q \in Residue(\mathcal{P}, M)$, $M \vDash b(q)$ there is a $q' \in R'$ s.t. $h(q) = h(q')$. Once we are aware of this fact this le a is proved exactly as Le a 1. □

Theorem 6. *Let $\mathcal{P}$ be a DLP and M be its total interpretation. If*

$$R' = \{r \mid r \in Res^R(\mathcal{P}, M) \wedge M \vDash b(r)\}$$

is sufficiently acyclic then the following propositions hold:

(i) $M \in DSM(\mathcal{P}) \equiv M \in RDSM(\mathcal{P})$,
(ii) $M \in DSM(\mathcal{P}) \equiv M \in BDSM(\mathcal{P})$,
(iii) $M \in DJU(\mathcal{P}) \equiv M \in BDJU(\mathcal{P})$.

Proof. Propositions (ii) and (iii) are proved like in the above Theore 4. The if part of (i) follows fro Theore 5. The only if part of (i) proves as follows.

Let $M \in DSM(\mathcal{P})$. Let R' be sufficiently acyclic. Fro Le a 3 we know that M can be co puted using only the rules of R'. Also

$$R' \subseteq Res^R(\mathcal{P}, M) \subseteq Res(\mathcal{P}, M) ,$$

so it follows fro Re ark 1 that $M \in RDSM(\mathcal{P})$. □

In the following le a we show that even if we have slightly odified the condition, its satisfaction still i plies that the root condition is also satisfied. Hence if the odified condition is satisfied, all five of the se antics for DLPs coincide on a given odel as we state in Corollary 3.

Lemma 4. *Let $\mathcal{P}$ be a MDLP and M its total interpretation. Let*

$$R' = \{r \mid r \in Res^R(\mathcal{P}, M) \wedge M \vDash b(r)\} \ .$$

If R' is sufficiently acyclic and $M \in DJU(\mathcal{P})$ ($M \in BDJU(\mathcal{P})$) then $\mathcal{P}$, M, $Rej(\mathcal{P}, M)$ ($\mathcal{P}$, M, $Rej^\star(\mathcal{P}, M)$) obey the root condition.

Proof. This lemma the same way as Lemma 2 if we realize that when $M \in DJU(\mathcal{P})$ ($M \in BDJU(\mathcal{P})$) then for each rule $r \in Res(\mathcal{P}, M)$ ($r \in Res^\star(\mathcal{P}, M)$) s.t. $M \vDash b(r)$ and $h(r) = L$ there also exists $r' \in R'$ s.t. $h(r') = L$. □

Corollary 3. *Let $\mathcal{P}$ be a DLP and M its total interpretation. If the set*

$$R' = \{r \mid r \in Res^R(\mathcal{P}, M) \wedge M \vDash b(r)\}$$

is sufficiently acyclic then

$$M \in DSM(\mathcal{P}) \equiv M \in BDSM(\mathcal{P}) \equiv M \in RDSM(\mathcal{P}) \equiv$$
$$\equiv M \in DJU(\mathcal{P}) \equiv M \in BDJU(\mathcal{P}) \ .$$

As for Corollary 1, also for Corollary 3 it holds that if, using it, we want to compare entire model-sets assigned to a program by a pair of semantics, computational complexity is the same as enumerating and comparing these two model-sets. Anyway, it trivially follows from this corollary that all five of the semantics coincide on strictly acyclic programs, as follows in Corollary 4.

Corollary 4. *Let $\mathcal{P}$ be a strictly acyclic DLP. Then*

$$DSM(\mathcal{P}) = BDSM(\mathcal{P}) = RDSM(\mathcal{P}) = DJU(\mathcal{P}) = BDJU(\mathcal{P}) \ .$$

6 Conclusion

In accordance with [3, 15], we have built MDLPs over a more general language of GELPs, that allows for more elegant comparisons, since no transformations are necessary, as the previous approaches are obtained as its special cases. We have then compared four different rule-rejecting semantics of MDLPs and in addition one more when restricted to linear DLPs. We have introduced sufficient acyclicity. Using this notion, we have provided a restrictive condition on a MDLP (DLP) $\mathcal{P}$ and a given candidate model M s.t. if it is satisfied all four (five) semantics coincide on $\mathcal{P}$ and M. As a trivial consequence we have stated the main result, that on strictly acyclic programs all four (five) of the semantics coincide.

There are several open problems. As there are programs that contain cycles and several of the five semantics coincide on them, the search for a more proper characterization of the class of programs on which these semantics coincide is still open. In this line, we suggest investigation of other well known classes, as stratified and call-consistent progra s. One of the ost favourable se antics, RDSM, is only known for DLPs, generalizing RDSM to MDLPs is a challenging proble . Co paring se antics that are based on rejection of rules with other approaches (such as the one of [15] based on Kripke structures) ight be interesting. To eet this goal, we propose that ore abstract criteria for evaluating these se antics should be introduced, seeing so e of the present ones, e.g., the refined extension principle of [6, 7], too attached to the rule-rejecting fra ework.

Acknowledgements

I would like to thank to anony ous referees for valuable co ents and suggestions. I would like to thank to Ján Šefránek and João A. Leite for their advising and help and to Michaela Danišová and to Martin Baláž for language and typographical corrections.

References

1. Leite, J.A., Alferes, J.J., Pereira, L.M.: Multi-dimensional dynamic logic programming. In Sadri, F., Satoh, K., eds.: Proceedings of the CL-2000 Workshop on Computational Logic in Multi-Agent Systems (CLIMA'00). (2000) 17–26
2. Leite, J.A., Pereira, L.M.: Iterated logic program updates. In Jaffar, J., ed.: Proceedings of the 1998 Joint International Conference and Symposium on Logic Programming (JICSLP'98), MIT Press (1998) 265–278
3. Leite, J.A.: Evolving Knowledge Bases: Specification and Semantics. Volume 81 of Frontiers in Artificial Intelligence and Applications, Dissertations in Artificial Intelligence. IOS Press, Amsterdam (2003)
4. Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C.: Dynamic logic programming. In Cohn, A.G., Schubert, L.K., Shapiro, S.C., eds.: Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Morgan Kaufmann (1998) 98–109
5. Eiter, T., Sabbatini, G., Fink, M., Tompits, H.: On updates of logic programs: Semantics and properties. Technical Report 1843-00-08, Institute of Information Systems, Vienna University of Technology (2002)
6. Alferes, J.J., Banti, F., Brogi, A., Leite, J.A.: The refined extension principle for semantics of dynamic logic programming. Studia Logica **79(1)** (2005) 7–32
7. Alferes, J.J., Banti, F., Brogi, A., Leite, J.A.: Semantics for dynamic logic programming: A principle-based approach. In Lifschitz, V., Niemela, I., eds.: Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-7), Springer-Verlag (2004)
8. Leite, J.: On some differences between semantics of logic program updates. In Lemaitre, C., Reyes, C.A., Gonzalez, J.A., eds.: Advances in Artificial Intelligence: Proceedings of the 9th Ibero-American Conference on AI (IBERAMIA-04). LNAI, Springer (2004)
9. Eiter, T., Sabbatini, G., Fink, M., Tompits, H.: On properties of update sequences based on causal rejection. Theory and Practice of Logic Programming (2002) 711–767

10. Homola, M.: On relations of the various semantic approaches in multidimensional dynamic logic programming. Master's thesis, Comenius University, Faculty of Mathematics Physics and Informatics, Bratislava (2004)
11. Leite, J.A., Alferes, J.J., Pereira, L.M.: Multi-dimensional logic programming. Technical report, Departamento de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa (2001)
12. Leite, J.A., Alferes, J.J., Pereira, L.M.: Multi-dimensional dynamic knowledge representation. In Eiter, T., Faber, W., Truszczynski, M., eds.: Proceedings of the Sixth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'01), Springer (2001) 365–378
13. Alferes, J.J., Banti, F., Brogi, A.: From logic program updates to action description updates. In this volume.
14. Sakama, C., Inoue, K.: Coordination between logical agents. In this volume.
15. Šefránek, J.: Semantic considerations on rejection. In: Proceedings of the International Workshop on Non-Monotonic Reasoning (NMR 2004), Foundations of Nonmonotonic Reasoning. (2004)
16. van Emden, M.H., Kowalski, R.A.: The semantics of predicate logic as a programming language. Journal of the ACM **23** (1976) 733–742
17. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In Kowalski, R.A., Bowen, K.A., eds.: Logic Programming, Proceedings of the Fifth International Conference and Symposium, MIT Press (1988) 1070–1080
18. Apt, K.R., Bezem, M.: Acyclic programs. New Generation Computing **9** (1991) 335–363
19. Šiška, J.: Refined extension principle for multi-dimensional dynamic logic programming. Master's thesis, Comenius University, Faculty of Mathematics Physics and Informatics, Bratislava (2004)

Declarative Agent Control

Antonis Kakas[1], Paolo Mancarella[2], Fariba Sadri[3],
Kostas Stathis[2,4], and Francesca Toni[2,3]

[1] Dept. of Computer Science, University of Cyprus
antonis@cs.ucy.ac.cy
[2] Dip. di Informatica, Università di Pisa
{paolo, stathis, toni}@di.unipi.it
[3] Dept. of Computing, Imperial College London
{fs, ft}@doc.ic.ac.uk
[4] School of Informatics, City University London
kostas@soi.city.ac.uk

Abstract. In this work, we extend the architecture of agents (and robots) based upon fixed, one-size-fits-all cycles of operation, by providing a framework of declarative specification of agent control. Control is given in terms of *cycle theories*, which define in a declarative way the possible alternative behaviours of agents, depending on the particular circumstances of the (perceived) external environment in which they are situated, on the internal state of the agents at the time of operation, and on the agents' behavioural profile. This form of control is adopted by the KGP model of agency and has been successfully implemented in the PROSOCS platform. We also show how, via cycle theories, we can formally verify properties of agents' behaviour, focusing on the concrete property of agents' *interruptibility*. Finally, we give some examples to show how different cycle theories give rise to different, heterogeneous agents' behaviours.

1 Introduction

To ake theories of agency practical, nor ally a control co ponent is proposed within concrete agent (robot) architectures. Most such architectures rely upon a fixed, one-size-fits-all cycle of control, which is forced upon the agents whatever the situation in which they operate. This kind of control has any drawbacks, and has been criticised by any (e.g. in robotics), as it does not allow us to take into account changes in the environ ent pro ptly and it does not take into account agent's preferences and "personality".

In this paper, we present an alternative approach, which odels agents' control via declarative, logic-based *cycle theories*, which provide *flexible control* in that: (i) they allow the sa e agent to exhibit different behaviour in different circu stances (internal and external to the agent), thus extending in a non-trivial way conventional, fixed cycles of behaviour, (ii) they allow us to state and verify for al properties of agent behaviour (e.g. their interruptibility), and thus (iii) provide i ple entation guidelines to design suitable agents for suitable

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 96–110, 2005.

applications. Furthermore, cycle theories allow different agents to have different patterns of behaviour in the same circumstances, by varying few, well-identified components. Thus, by adopting different cycle theories we obtain behaviourally *heterogeneous* agents.

The notion of cycle theory and its use to determine the behaviour of agents can in principle be imported into any agent system, to replace conventional fixed cycles. However, in defining the cycle theory of an agent, we will assume that the agent is equipped with a pool of *state transitions* that modify its internal state. We will understand the operation of agents simply in terms of sequences of such transitions. Such sequences can be obtained from *fixed cycles* of operation of agents as in most of the literature. Alternatively, such sequences can be obtained via fixed cycles together with the possibility of selecting amongst such fixed cycles according to some criteria e.g. the type of external environment in which the agent will operate (see the recent work of [4]). Yet another possibility, that we pursue in this paper, is to specify the required operation via more versatile cycle theories that are able to generate dynamically several cycles of operations according to the current need of the agent. This approach has been adopted in the KGP model of agency [10, 2] and implemented in the PROSOCS platform [16].

We will define a cycle theory as a logic program with priorities over rules. The rules represent possible follow-ups of (already executed) transitions. The priorities express high-level preferences of the particular agent equipped with the cycle theory, that characterise the operational behaviour of the agent, e.g. a preference in testing the preconditions of an action before it tries to execute it. We will assume that the choice for the next transition depends only on the transition that has just been executed (and the resulting state of the agent), and not on the longer history of the previous transitions. We believe this not to be restrictive, in that the effects of any earlier transitions may in any case be recorded in the internal state of the agent and reasoned upon by it. Also, the approach can be extended to take into account longer histories of transitions when deciding the next one.

2 Background

Cycle theories will be written in the general framework of Logic Programming with Priorities (LPP). Our approach does not rely on any concrete such framework. One such concrete framework could be the Logic Programming without Negation as Failure (LPwNF) [5, 8] suitably extended to deal with dynamic preferences [9]. Other concrete frameworks that could be used for LPP are, for instance, those presented in [13, 12]. Note also that our approach does not depend crucially on the use of the framework of LPP: other frameworks for the declarative specification of preference policies, e.g. Default Logic with Priorities [3], could be used instead. Note, however, that the use of a logic-based framework where priorities are encoded within the logic itself is essential, since it allows reasoning even with potentially contradictory preferences. Also, note that the choice of one logic rather than another ight affect the properties of agents specified via cycle theories.

For the purposes of this paper, we will assu e that an *LPP-theory*, referred to as $\mathcal{T}$, consists of four parts:

(i) a low-level part P, consisting of a logic progra ; each rule in P is assigned a na e, which is a ter ; e.g., one such rule could be

$$n(X) : p(X) \leftarrow q(X, Y), r(Y)$$

with na e $n(X)$;

(ii) a high-level part H, specifying conditional, dyna ic priorities a ongst rules in P; e.g., one such priority could be

$$h(X) : n(X) \succ m(X) \leftarrow c(X)$$

to be read: if (so e instance of) the condition $c(X)$ holds, then the rule in P with na e (the corresponding instance of) $n(X)$ should be given higher priority than the rule in P with na e (the corresponding instance of) $m(X)$. The rule is given a na e, $h(X)$;

(iii) an auxiliary part A, defining predicates occurring in the conditions of rules in P and H and not in the conclusions of any rule in P;

(iv) a notion of inco patibility which, for the purposes of this paper, can be assu ed to be given as a set of rules defining the predicate *incompatible*, e.g.

$$incompatible(p(X), p'(X))$$

to be read: any instance of the literal $p(X)$ is inco patible with the corresponding instance of the literal $p'(X)$. We assu e that inco patibility is sy etric, and refer to the set of all inco patibility rules as I.

Any concrete LPP fra ework is equipped with a notion of entail ent, that we denote by $\models_{pr}$. Intuitively, $\mathcal{T} \models_{pr} \alpha$ iff α is the "conclusion" of a sub-theory of $P \cup A$ which is "preferred" wrt $H \cup A$ in $\mathcal{T}$ over any other any other sub-theory of $P \cup A$ that derives "conclusion" inco patible with α (wrt I). Here, we are assu ing that the underlying logic progra ing language is equipped with a notion of "entail ent" that allows to draw "conclusions". In [13, 12, 9, 8, 5], $\models_{pr}$ is defined via argu entation.

3 Abstract Agent Model

We assu e that our agents confor to the following abstract odel, which can be seen as a high-level abstraction of ost agent syste s in the literature. Agents are equipped with

- so e *internal state*, which changes over the life-ti e of the agent, and is for alised in so e logic-based language or via so e concrete data structure in so e progra ing language;
- so e pool of *(state) transitions*, that odify the state of the agent, and ay take so e inputs to be "co puted" or selected by
- so e *selection functions* on their states.

For example, the state may consist of beliefs, desires and intentions, represented in some modal logics, as in the BDI architecture [14] and its follow-ups, e.g. [1], or commitments and commitment rules, as in [15], or beliefs, goals and capabilities, represented in concurrent logic programming, as in [7], or knowledge, goals and plan, represented in (extensions of) logic programming, as in [11].

The transitions in the given pool can be any, but, if we abstract away from existing agent architectures and models in the literature, we can see that we need at least a transition responsible for observing the environment, thus rendering the agents situated. This transition might modify the internal state differently in concrete agent architectures, to record the observed events and properties of the environment. Here, we will call such a transition *Passive Observation Introduction* (POI). POI is "passive" in the sense that, via such a transition, the agent does not look for anything special to observe, but rather it opens its "reception channel" and records any inputs what its sensors perceive. Another transition that is present in most agent systems is that of *Action Execution* (AE), whereby actions may be "physical", communicative, or "sensing", depending on the concrete systems.

Other useful transitions besides POI and AE (see e.g. [10, 2]) may include Goal Introduction (GI), to introduce new goals into the state of the agent, taking into account changes to the state and to the external environment that somehow affect the preferences of the agent over which goals to adopt, Plan Introduction (PI), to plan for goals, Reactivity (RE), to react to perceived changes in the environment by means of condition-action/commitment-like rules, Sensing Introduction (SI), to set up sensing actions for sensing the preconditions of actions in the agent's plan, to make sure these actions are indeed executable, Active Observation Introduction (AOI), to actively seek information from the environment, State Revision (SR) to revise the state currently held by the agent, and Belief Revision (BR), e.g. by learning.

Whatever pool of transitions one might choose, and whatever their concrete specification might be, we will assume that they are represented as

$$T(S, X, S', \tau)$$

where S is the state of the agent before the transition is applied and S' the state after, X is the (possibly empty) input taken by the transition, and τ is the time of application of the transition. Note that we assume the existence of a *clock* (possibly external to the agent and shared by a number of agents), whose task is to mark the passing of time. The clock is responsible for labelling the transitions with the time at which they are applied. This time (and thus the clock) might play no role in some concrete agent architectures and models, where time is not reasoned upon explicitly. However, if the framework adopted to represent the state of the agent directly manipulates and reasons with time, the presence of a clock is required. Note also that the clock is useful (if not necessary) to label executed actions, and in particular communicative actions, to record their time of execution, as foreseen e.g. by FIPA standards for communication [6].

As far as the selection functions are concerned, we will assume that each transition T available to the agent is equipped with a selection function f_T, whose specification depends on the representation chosen for the state and on the specification of the transition itself. For exa ple, AE is equipped with a selection function f_{AE} responsible for choosing actions to be executed. These actions ay be a ongst those actions in the plan (intention/co it ent store) part of the state of the agent whose ti e has not run-out at the ti e of selection (and application of the transition) and belonging to a plan for so e goal which has not already been achieved by other eans.

In the next Section, we will see that, for fixed cycles, the role of the selection functions is exclusively to select the inputs for the appropriate transition when the turn of the transition co es up. Later, in Section 5, we will see that the role of selection functions when using cycle theories is to help decide which transition is preferred and should be applied next, as well as provide its input.

4 Fixed Cycles and Fixed Operational Trace

Both for fixed cycles and cycle theories, we will assu e that the operation of an agent will start fro so e *initial state*. This can be seen as the state of the agent when it is created. The state then evolves via the transitions, as co ended by the fixed cycle or cycle theory. For exa ple, the initial state of the agent could have an e pty set of goals and an e pty set of plans, or so e designer-given goals and an e pty set of plans. In the sequel, we will indicate the given initial state as S_0.

A *fixed cycle* is a fixed sequence of transitions of the for

$$T_1, \ldots, T_n$$

where each T_i, $i = 1, \ldots, n$, is a transition chosen fro the given pool, and $n \geq 2$.

A fixed cycle induces a *fixed operational trace* of the agent, na ely a (typically infinite) sequence of applications of transitions, of the for

$$T_1(S_0, X_1, S_1, \tau_1), T_2(S_1, X_2, S_2, \tau_2), \ldots, T_n(S_{n-1}, X_n, S_n, \tau_n),$$
$$T_1(S_n, X_{n+1}, S_{n+1}, \tau_{n+1}), \ldots, T_n(S_{2n-1}, X_{2n}, S_{2n}, \tau_{2n}), \ldots$$

where, for each $i \geq 1$, $f_{T_i}(S_{i-1}, \tau_i) = X_i$, na ely, at each stage, X_i is the (possibly e pty) input for the transition T_i chosen by the corresponding selection function f_{T_i}.

Then, a classical "observe-think-act" cycle (e.g. see [11]) can be represented in our approach as the fixed cycle:

$$POI, RE, PI, AE, AOI.$$

As a further exa ple, a purely reactive agent, e.g. with its knowledge consisting of condition-action rules, can execute the cycle

$$POI, RE, AE.$$

Note that POI is interpreted here as a transition which is under the control of the agent, na ely the agent decides when it is ti e to open its "reception channel". Below, in Section 8, we will see a different interpretation of POI as an "interrupt".

Note that, although fixed cycles such as the above are quite restrictive, they may be sufficiently appropriate in some circumstances. For example, the cycle for a purely reactive agent may be fine in an environment which is highly dynamic. An agent may then be equipped with a catalogue of fixed cycles, and a number of conditions on the environment to decide when to apply which of the given cycles. This would provide for a (limited) form of intelligent control, in the spirit of [4], paving the way toward the more sophisticated and fully declarative control via cycle theories given in the next Section.

5 Cycle Theories and Cycle Operational Trace

The role of the cycle theory is to dynamically control the sequence of the internal transitions that the agent applies in its "life". It regulates these "narratives of transitions" according to certain requirements that the designer of the agent would like to impose on the operation of the agent, but still allowing the possibility that any (or a number of) sequences of transitions can actually apply in the "life" of an agent. Thus, whereas a fixed cycle can be seen as a restrictive and rather inflexible catalogue of allowed sequences of transitions (possibly under pre-defined conditions), a cycle theory identifies *preferred patterns* of sequences of transitions. In this way a cycle theory regulates in a flexible way the operational behaviour of the agent.

Formally, a cycle theory $\mathcal{T}_{cycle}$ consists of the following parts.

- An *initial* part $\mathcal{T}_{initial}$, that determines the possible transitions that the agent could perform when it starts to operate (*initial cycle step*). More concretely, $\mathcal{T}_{initial}$ consists of rules of the form

  $$*T(S_0, X) \leftarrow C(S_0, \tau, X), now(\tau)$$

  sanctioning that, if the conditions C are satisfied in the initial state S_0 at the current time τ, then the initial transition should be T, applied to state S_0 and input X, if required. Note that $C(S_0, \tau, X)$ may be absent, and $\mathcal{T}_{initial}$ might simply indicate a fixed initial transition T_1.

  The notation $*T(S, X)$ in the head of these rules, meaning that the transition T can be potentially chosen as the next transition, is used in order to avoid confusion with the notation $T(S, X, S', \tau)$ that we have introduced earlier to represent the actual application of the transition T.
- A *basic* part $\mathcal{T}_{basic}$ that determines the possible transitions (*cycle steps*) following other transitions, and consists of rules of the form

  $$*T'(S', X') \leftarrow T(S, X, S', \tau), EC(S', \tau', X'), now(\tau')$$

  which we refer to via the name $\mathcal{R}_{T|T'}(S', X')$. These rules sanction that, after the transition T has been executed, starting at time τ in the state S and ending at the current time τ' in the resulting state S', and the conditions EC evaluated in S' at τ' are satisfied, then transition T' could be the next transition to be applied in the state S' with the (possibly empty) input X', if required. The conditions EC are called *enabling conditions* as they determine when a cycle-step from the transition T to the transition T' can be applied.

In addition, they determine the input X' of the next transition T'. Such inputs are determined by calls to the appropriate selection functions.
- A *behaviour* part $\mathcal{T}_{behaviour}$ that contains rules describing dynamic priorities amongst rules in $\mathcal{T}_{basic}$ and $\mathcal{T}_{initial}$. Rules in $\mathcal{T}_{behaviour}$ are of the form
  $\mathcal{R}_{T|T'}(S, X') \succ \mathcal{R}_{T|T''}(S, X'') \leftarrow BC(S, X', X'', \tau), now(\tau)$
  with $T' \neq T''$, which we will refer to via the name $\mathcal{P}^{T}_{T' \succ T''}$. Recall that $\mathcal{R}_{T|T'}(\cdot)$ and $\mathcal{R}_{T|T''}(\cdot)$ are (names of) rules in $\mathcal{T}_{basic} \cup \mathcal{T}_{initial}$. Note that, with an abuse of notation, T could be 0 in the case that one such rule is used to specify a priority over the *first* transition to take place, in other words, when the priority is over rules in $\mathcal{T}_{initial}$. These rules in $\mathcal{T}_{behaviour}$ sanction that, at the current time τ, after transition T, if the conditions BC hold, then we prefer the next transition to be T' over T'', namely doing T' has *higher priority* than doing T'', after T. The conditions BC are called *behaviour conditions* and give the behavioural profile of the agent. These conditions depend on the state of the agent after T and on the parameters chosen in the two cycle steps represented by $\mathcal{R}_{T|T'}(S, X')$ and $\mathcal{R}_{T|T''}(S, X'')$. Behaviour conditions are *heuristic* conditions, which may be defined in terms of the *heuristic selection functions*, where appropriate. For example, the heuristic action selection function may choose those actions in the agent's plan whose time is close to running out amongst those whose time has not run out.
- An *auxiliary part* including definitions for any predicates occurring in the enabling and behaviour conditions, and in particular for selection functions (including the heuristic ones, if needed).
- An *incompatibility part*, including rules stating that all different transitions are incompatible with each other and that different calls to the same transition but with different input items are incompatible with each other. These rules are facts of the form
  $incompatible(*T(S, X), *T'(S, X')) \leftarrow$
  for all T, T' such that $T \neq T'$, and of the form
  $incompatible(*T(S, X), *T(S, X')) \leftarrow X \neq X'$
  expressing the fact that only one transition can be chosen at a time.

Hence, $\mathcal{T}_{cycle}$ is an LPP-theory (see Section 2) where:
(i) $P = \mathcal{T}_{initial} \cup \mathcal{T}_{basic}$, and (ii) $H = \mathcal{T}_{behaviour}$.

In the sequel, we will indicate with $\mathcal{T}^0_{cycle}$ the sub-cycle theory $\mathcal{T}_{cycle} \setminus \mathcal{T}_{basic}$ and with $\mathcal{T}^s_{cycle}$ the sub-cycle theory $\mathcal{T}_{cycle} \setminus \mathcal{T}_{initial}$.

The cycle theory $\mathcal{T}_{cycle}$ of an agent is responsible for the behaviour of the agent, in that it induces a *cycle operational trace* of the agent, namely a (typically infinite) sequence of transitions

$T_1(S_0, X_1, S_1, \tau_1), \ldots, T_i(S_{i-1}, X_i, S_i, \tau_i),$
$T_{i+1}(S_i, X_{i+1}, S_{i+1}, \tau_{i+1}), \ldots$

(where each of the X_i may be empty), such that

- S_0 is the given initial state;
- for each $i \geq 1$, τ_i is given by the clock of the system, with the property that $\tau_i < \tau_{i+i}$;
- (*Initial Cycle Step*) $\mathcal{T}^0_{cycle} \wedge now(\tau_1) \models_{pr} *T_1(S_0, X_1)$;

– (*Cycle Step*) for each $i \geq 1$
  $\mathcal{T}^s_{cycle} \wedge T_i(S_{i-1}, X_i, S_i, \tau_i) \wedge now(\tau_{i+1}) \models_{pr} *T_{i+1}(S_i, X_{i+1})$
  namely each (non-final) transition in a sequence is followed by the most preferred transition, as specified by $\mathcal{T}_{cycle}$.

If, at some stage, the most preferred transition determined by $\models_{pr}$ is not unique, we choose arbitrarily one.

Note that, for simplicity, the above definition of operational trace prevents the agent from executing transitions *concurrently*. However, a first level of concurrency can be incorporated within traces, by allowing all preferred transitions to be executed at every step. For this we would only need to relax the above definition of *incompatible* transitions to be restricted between any two transitions whose executions could interact with each other and therefore cannot be executed concurrently on the same state, e.g. the Plan Introduction and State Revision transitions. This would then allow several transitions to be chosen together as preferred next transitions and a concurrent model of operation would result by carrying out simultaneously the (non-interacting) state updates imposed by these transitions. Further possibilities of concurrency will be subject of future investigations.

In section 8 we will provide a simple extension of the notion of operational trace defined above.

6 Fixed Versus Flexible Behaviour

Cycle theories generalise fixed cycles in that the behaviour given by a fixed operational trace can be obtained via the behaviour given by a cycle operational trace, for some special cycle theories. This is shown by the following theorem, which refers to the notions of *fixed cycle* and *fixed operational trace* introduced in Section 4.

Theorem 1. *Let* $T_1, \ldots, T_n$ *be a* fixed cycle*, and let* f_{T_i} *be a given selection function for each* $i = 1, \ldots, n$*. Then there exists a cycle theory* $\mathcal{T}_{cycle}$ *which induces a cycle operational trace identical to the fixed operational trace induced by the fixed cycle.*

Proof. The proof is by construction as follows.

– $\mathcal{T}_{initial}$ consists of the rule
  $*T_1(S_0, X) \leftarrow now(\tau)$
  i.e. the initial transition is simply T_1.
– $\mathcal{T}_{basic}$ consists of the following rules, for each i with $2 \leq i \leq n$:
  $*T_i(S', X') \leftarrow T_{i-1}(S, X, S', \tau), now(\tau'), X' = f_{T_i}(S', \tau')$.
  In addition $\mathcal{T}_{basic}$ contains the rule
  $*T_1(S', X') \leftarrow T_n(S, X, S', \tau), now(\tau'), X' = f_{T_1}(S', \tau')$.
– $\mathcal{T}_{behaviour}$ is empty.
– the auxiliary part contains the definitions of the given selection functions f_{T_i}, for each $i = 1, \ldots, n$.

The proof then easily follows by construction, since at each stage only one cycle step is enabled and no preference reasoning is required to choose the next transition to be executed. □

It is clear that there are so e (any) cycle theories that cannot be apped onto any fixed cycles, e.g. the cycle theory given in the next Section. So, providing control via cycle theories is a genuine extension of providing control via conventional fixed cycles.

7 An Example

In this Section we exe plify the flexibility afforded by cycle theories through a si ple exa ple. Assu e that the pool of transitions consists of GI, PI, AE and POI, as described in Section 3. We start fro the cycle theory corresponding to the fixed cycle given by POI, GI, PI, AE which is constructed as follows (see Theore 1).

(1) $\mathcal{T}_{initial}$ with the following rule

$*POI(S_0, \{\}) \leftarrow$

na ely, the only way an agent can start is through a POI.

(2) $\mathcal{T}_{basic}$ with the following rules

$*GI(S', \{\}) \leftarrow POI(S, \{\}, S', \tau)$

$*PI(S', Gs) \leftarrow GI(S, \{\}, S', \tau), Gs = f_{PI}(S', \tau'), now(\tau')$

$*AE(S', As) \leftarrow PI(S, Gs, S', \tau), As = f_{AE}(S', \tau'), now(\tau')$

$*POI(S', \{\}) \leftarrow AE(S, As, S', \tau)$

(3) $\mathcal{T}_{behaviour}$ is e pty.

A first si ple i prove ent, providing a li ited for of flexibility, consists in refining the rule $\mathcal{R}_{GI|PI}(\cdot)$ by adding the condition that the set of goals to plan for, which are selected by the corresponding selection function f_{PI}, is non-e pty. This a ounts at odifying the second rule of $\mathcal{T}_{basic}$ by adding the condition $Gs \neq \{\}$ to its body.

Si ilarly, AE is an option after PI if so e actions can actually be selected for execution. This a ounts at odifying the the third rule of $\mathcal{T}_{basic}$ by adding the condition $As \neq \{\}$ to its body.

In this case, we should provide further options for choosing the transition to be executed after GI and PI, respectively. To adhere with the given original cycle, these rules could be si ply suitable rules na ed by $\mathcal{R}_{GI|AE}(S', As)$, $\mathcal{R}_{GI|POI}(S', \{\})$ and $\mathcal{R}_{PI|POI}(S', \{\})$, i.e. AE and POI are also an option after GI, and POI is also an option after PI. With this choice, the standard operational trace is recovered by adding to the $\mathcal{T}_{behaviour}$ part of the cycle theory the following rules

$\mathcal{R}_{GI|PI}(S', Gs) \succ \mathcal{R}_{GI|AE}(S', As) \leftarrow$

$\mathcal{R}_{GI|AE}(S', As') \succ \mathcal{R}_{GI|POI}(S', \{\}) \leftarrow$

$\mathcal{R}_{GI|PI}(S', Gs') \succ \mathcal{R}_{GI|POI}(S', \{\}) \leftarrow$

$\mathcal{R}_{PI|AE}(S', As) \succ \mathcal{R}_{PI|POI}(S', \{\}) \leftarrow$

The first rule states that PI has to be preferred over AE as the next transition to be applied after GI, whenever both PI and AE are enabled. Similarly for the other rules.

A more interesting, proper extension of the original (fixed) cycle amounts at adding further options to the transition which can follow any given transition. Imagine for instance that we want to express the behaviour of a *punctual* or *timely* agent. This agent should always prefer executing actions if there are actions in the plan which have become *urgent*. This can be declaratively formalised by adding to the $\mathcal{T}_{basic}$ part the rules

$$*AE(S', As') \leftarrow T(S, X, S', \tau), As' = f_{AE}(S', \tau'), now(\tau')$$

for each transition T in the pool, and by adding to the $\mathcal{T}_{behaviour}$ part the following rules named $\mathcal{P}^{T}_{AE \succ T'}$:

$\mathcal{R}_{T|AE}(S', As') \succ \mathcal{R}_{T|T'}(S', X') \leftarrow urgent(As')$

for each transition T and $T' \neq$ AE, where *urgent* is defined in the auxiliary part of the theory with the intuitive meaning. In the rest of this Section, we use $\mathcal{T}^{fix}_{cycle}$ to refer to the cycle theory corresponding to the fixed cycle POI, GI, PI, AE, and we use $\mathcal{T}^{ext}_{cycle}$ to refer to the extended cycle theory.

As a concrete example, consider an agent aiding a businessman who, while on a business trip, can choose amongst three possible goals: return home (*home*), read news (*news*), and recharge his laptop battery (*battery*). Let us use first the cycle theory $\mathcal{T}^{fix}_{cycle}$.

Suppose that, initially (when $now(1)$ holds), the agent's state is empty, namely the (businessman's) agent holds no plan or goal, and that the initial POI does not add anything to the current state. Then GI is performed as the next transition in the trace:

$GI(S_0, \{\}, S_1, 1)$,

and suppose also that the application of GI generates the agent's goal (added to S_1) $G_1 = home$. This goal may come along with a time parameter and some temporal constraints associated with it, e.g. the actual goal can be represented by $(home, t) \wedge t < 20$. Due to space limitations, we intentionally omit here the details concerning temporal parameters of goals and actions, and the temporal constraints associated with them. Since the state contains a goal to be planned for, suppose that the selection function f_{PI} selects this goal, and the PI transition is applied next, producing two actions *book_ticket* and *take_train*. Hence, the second transition of the trace is (when $now(3)$ holds)

$PI(S_1, \{\}, S_2, 3)$

where the new state S_2 contains the above actions.

Suppose now that the selection function f_{AE} selects the action *book_ticket* and hence that the next element of the trace is (when $now(4)$ holds)

$AE(S_2, \{book\_ticket\}, S_3, 4)$. (*)

In the original fixed cycle the next applicable transition is POI, and assume that this is performed at some current time, say 10. Hence the next element of the trace is (when $now(10)$ holds)

$POI(S_3, \{\}, S_4, 10)$. (**)

Imagine that this POI brings about the new knowledge that the laptop battery is low, suitably represented in the resulting state S_4. Then the next transition GI changes the state so that the goal *battery* is added, and then PI is performed to introduce a suitable plan to recharge the battery and so on.

Now suppose that we use $\mathcal{T}^{ext}_{cycle}$ instead and that the operational trace is identical up to the execution of the transition (*). At this point, the action *take_train* may have become *urgent*. Notice that it is likely that this same action was not urgent at time 3, when *book_ticket* was selected for execution, but has become urgent at time 10 (e.g. because the train is leaving at 11). Then, if we use $\mathcal{T}^{ext}_{cycle}$, the rule $\mathcal{P}^{AE}_{AE \succ POI}$ applies and the next element of the trace, replacing (**) above, becomes

$$AE(S_3, \{take\_train\}, S'_4, 10).$$

This example shows how the use of cycle theories can lead to flexible behaviours. More flexibility may be achieved by allowing the agents to be interruptible, i.e. to be able to react to changes in the environment in which they are situated as soon as the perceive those changes. This added flexibility requires some further extensions, that we discuss in the next Section.

8 Interruptible Agents

In our approach we can provide a declarative specification of *interruptible* agents, i.e. agents that are able to dynamically modify their "normal" (either fixed or cycle) operational trace when they perceive changes in the environment in which they are situated.

In order to obtain interruptibility, we will make use of the POI transition as the means by which an agent can react to an interrupt. Referring to the example of the previous Section, assume that our agent can book the ticket only through its laptop and, by the time it decides to actually book the ticket, the laptop battery has run out. Then, the action of recharging the laptop battery should be executed as soon as possible in order to (possibly) achieve the initial goal. Indeed, executing the booking action before recharging would not be feasible at all.

In order to model the environment where the agent is situated, we assume the existence of an environmental knowledge base *Env* that it is not directly under the control of the agent, in that the latter can only dynamically assimilate the knowledge contained in *Env*. This knowledge base can be seen as an abstraction of the physical (as opposed to the mental) part of the agent (its *body*) which, e.g. through its sensors, perceives changes in the environment. We assume that, besides the knowledge describing the agent's percepts, *Env* models a special propositional symbol, referred to as *changed_env* which holds as soon as the body of the agent perceives any new, relevant changes in the environment. The way we model the reaction of the agent to the changes represented by *changed_env* becoming true, is through the execution of a POI. We also assume that the execution of a POI transition resets the truth value of *changed_env*, so that the agent may be later alerted of further changes in the environment.

The *Env* knowledge base becomes now part of the knowledge that the agent uses in order to decide the next step in its operational trace. This is formally specified through the notion of *cycle-env operational trace*, which extends the notion of cycle operational trace introduced in Section 5, by replacing the definitions of *Initial Cycle Step* and *Cycle Step* by the following new definitions:
(*Initial Cycle-env Step*): $\mathcal{T}^0_{cycle} \wedge Env \wedge now(\tau_1) \models_{pr} *T_1(S_0, X_1)$;
(*Cycle-env Step*) for each $i \geq 1$

$$\mathcal{T}^s_{cycle} \wedge T_i(S_{i-1}, X_i, S_i, \tau_i) \wedge Env \wedge now(\tau_{i+1}) \models_{pr} *T_{i+1}(S_i, X_{i+1})$$

We can now define a notion of *interruptible agent* as follows. Let $\mathcal{T}_{cycle}$ be the cycle theory of the agent and let $T_1(\cdot), \ldots, T_i(\cdot), \ldots$ be a cycle operational trace of the agent. Let also $T_i(S_{i-1}, X_i, S_i, \tau_i)$ be an element of the given trace such that:

$Env \wedge now(\tau_i) \models \neg changed\_env$, and
$Env \wedge now(\tau_{i+1}) \models changed\_env$.

In other words, some changes have happened in the environment between the time of the execution of the transitions T_i and T_{i+1} in the trace. Then we say that the agent is interruptible if
$\mathcal{T}_{cycle} \wedge T_i(S_{i-1}, X_i, S_i, \tau_i) \wedge Env \wedge now(\tau_{i+1}) \models_{pr} *POI(S_i, \{\})$, i.e. as soon as the environment changes, in a cycle-env operational trace the next transition would be a POI.

It is worth noting that by interruptibility we do not mean here that the (executions of) transitions are interrupted, rather the trace is interrupted.

In order to make an agent interruptible, we need to extend both $\mathcal{T}_{basic}$ and $\mathcal{T}_{behaviour}$. In $\mathcal{T}_{basic}$, POI should be made an option after any other transition in the pool, which is achieved by adding the following rule $\mathcal{R}_{T|POI}(S, \{\})$, for any T:

$$*POI(S', \{\}) \leftarrow T(S, X', S', \tau).$$

In $\mathcal{T}_{behaviour}$, the following set of rules, where T, T' are transitions with $T' \neq$ POI, express that POI should be preferred over any other transition if the environment has actually changed:

$$\mathcal{R}_{T|POI}(S', \{\}) \succ \mathcal{R}_{T|T'}(S', X) \leftarrow changed\_env. \quad (***)$$

Notice that, even if the above extensions are provided in the overall $\mathcal{T}_{cycle}$ theory, the interruptibility of the agent is still not guaranteed. For instance, $\mathcal{T}_{behaviour}$ could contain further rules which make a transition $T \neq$ POI preferable over POI even if *changed_env* holds. One way to achieve full interruptibility is by adding the condition $\neg changed\_env$ in the body of any rule in $\mathcal{T}_{behaviour}$ other than the rules (***) given above.

9 Patterns of Behaviour

In this section we show how different patterns of operation can arise from different cycle theories aiming to capture different profiles of operational behaviour by agents. We assume the agent is equipped with a set of transitions, as in the KGP model [10, 2] (see Section 3 for an informal description of these transitions):

– POI, *Passive Observation Introduction*
– AE, *Action Execution*
– GI, *Goal Introduction*
– PI, *Plan Introduction*
– RE, *Reactivity*
– SI, *Sensing Introduction*
– AOI, *Active Observation Introduction*
– SR, *State Revision*

In Section 7 we have given a simple example of a cycle theory describing a punctual, timely agent which attempts to execute its planned actions in time. This agent was obtained by adding some specific rules to $\mathcal{T}_{behaviour}$ of a given cycle theory. The same approach can be adopted to obtain different profiles.

For example, we can define a *focused or committed* agent, which, once chosen a plan to execute, prefers to continue with this plan (refining it and/or executing parts of it) until the plan is finished or it has become invalid, at which point the agent may consider other plans or other goals. Hence transitions that relate to an existing plan have preference over transitions that relate to other plans. This profile of behaviour can be captured by the following rules added to $\mathcal{T}_{behaviour}$ of an appropriate cycle theory:

$$\mathcal{R}_{T|AE}(S, As) \succ \mathcal{R}_{T|T'}(S, X) \leftarrow same\_plan(S, As)$$

for any T and any $T' \neq AE$, and

$$\mathcal{R}_{T|PI}(S, Gs) \succ \mathcal{R}_{T|T'}(S, X) \leftarrow same\_plan(S, Gs)$$

for any T and any $T' \neq PI$. These rules state that the agent prefers to execute actions or to reduce goals from the same plan as the actions that have just been executed. Here, the behaviour conditions are defined in terms of some predicate *same_plan* which, intuitively, checks that the selected inputs for AE and PI, respectively, belong to the *same plan* as the actions most recently executed within the latest AE transition.

Another example of behavioral profile is the *impatient* pattern, where actions that have been tried and failed are not tried again. This can be captured by rules of the form:

$$\mathcal{R}_{T|T'}(S, \_) \succ \mathcal{R}_{T|AE}(S, As) \leftarrow failed(S, As)$$

for any T and any $T' \neq AE$. In this way, AE is given less preference than any other transition T' after any transition T. Intuitively, As are *failed* actions. As a result of this priority rule it is possible that such failed actions would remain un-tried again (unless nothing else is enabled) until they are timed out and dropped by SR.

If we want to capture a *careful* behaviour where the agent revises its state when one of its goals or actions times out (being careful not to have in its state other goals or actions that are now impossible to achieve in time) we would have in $\mathcal{T}_{behaviour}$ the rule:

$$\mathcal{R}_{T|SR}(S, \{\}) \succ \mathcal{R}_{T|T'}(S, \_) \leftarrow timed\_out(S, \tau)$$

for any T and any $T' \neq SR$. In this way, the SR transition is preferred over all other transitions, where the behaviour condition $timed\_out(S, \tau)$ succeeds if some goal or action in the state S has timed out at time τ.

10 Conclusions and Ongoing Work

We have presented an approach providing declarative agent control, via logic programs with priorities. Our approach share the aims of 3APL [4], to make the agent cycle programmable and the selection mechanisms explicit, but goes beyond it. Indeed, the approach of [4] can be seen as relying upon a catalogue of fixed cycles together with the possibility of selecting amongst such fixed cycles according to some criteria, whereas we drop the concept of fixed cycle completely, and replace it with fully programmable cycle theories.

Our approach allows us to achieve flexibility and adaptability in the operation of an autonomous agent. It also offers the possibility to state and verify properties of agents behaviour formally. In this paper we have exemplified the first aspect via an example, and the second aspect via the property of "interruptibility" of agents. The identification and verification of more properties is a matter for future work.

Our approach also lends itself to achieving heterogeneity in the overall operational behaviour of different agents that can be specified within the proposed framework. Indeed, an advantage of control via cycle theories is that it opens up the possibility to produce a variety of patterns of operation of agents, depending on the particular circumstances under which the transitions are executed. This variety can be increased, and many different *patterns or profiles of behaviour* can be defined by varying the cycle theory, thus allowing agents with (possibly) the same knowledge and operating in the same environment to exhibit heterogeneous behaviour, due to their different cycle theories. We have given a number of examples of profiles of behaviour. A systematic study of behaviour parameterisation (perhaps linking with Cognitive Science) is a matter for future work, as well as the comparison on how different behaviours affect the agents' individual welfare in different contexts.

Acknowledgments

This work was partially funded by the IST programme of the EC, FET under the IST-2001-32530 SOCS project, within the Global Computing proactive initiative. The last two authors were also supported by the Italian MIUR programme "Rientro dei cervelli".

References

1. R.H. Bordini, A. L. C. Bazzan, R. O. Jannone, D. M. Basso, R. M. Vicari, and V. R. Lesser. AgentSpeak(XL): Efficient intention selection in bdi agents via decision-theoretic task scheduling. In C. Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Part III*, pages 1294–1302, Bologna, Italy, July 15–19 2002. ACM Press.

2. A. Bracciali, N. Demetriou, U. Endriss, A. Kakas, W. Lu, P. Mancarella, F. Sadri, K. Stathis, G. Terreni, and F. Toni. The KGP model of agency for GC: Computational model and prototype implementation. In *Proc. Global Computing 2004 Workshop*, LNCS. Springer Verlag, 2004.
3. G. Brewka. Reasoning with priorities in default logic. In *AAAI94*, pages 940–945. AAAI Press, 1994.
4. M. Dastani, F. S. de Boer, F. Dignum, W. van der Hoek, M. Kroese, and J. Ch. Meyer. Programming the deliberation cycle of cognitive robots. In *Proc. of 3rd International Cognitive Robotics Workshop (CogRob2002)*, Edmonton, Alberta, Canada, 2002.
5. Y. Dimopoulos and A. C. Kakas. Logic programming without negation as failure. In *Logic Programming, Proceedings of the 1995 International Symposium, Portland, Oregon*, pages 369–384, 1995.
6. FIPA Communicative Act Library Specification, August 2001. Published on August 10th, 2001, available for download from the FIPA website, `http://www.fipa.org`.
7. K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J. Ch. Meyer. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.
8. A. C. Kakas, P. Mancarella, and P. M. Dung. The acceptability semantics for logic programs. In *Proceedings of the Eleventh International Conference on Logic Programming, Santa Marherita Ligure, Italy*, pages 504–519, 1994.
9. A. C. Kakas and P. Moraitis. Argumentation based decision making for autonomous agents. In J. S. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*, pages 883–890, Melbourne, Victoria, July 14–18 2003. ACM Press.
10. A.C. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. The KGP model of agency. In R. Lopez de Mantaras and L. Saitta, editors, *Proceedings of the Sixteenth European Conference on Artificial Intelligence, Valencia, Spain (ECAI 2004)*. IOS Press, August 2004.
11. R. A. Kowalski and F. Sadri. From logic programming towards multi-agent systems. *Annals of Mathematics and Artificial Intelligence*, 25(3/4):391–419, 1999.
12. R.A. Kowalski and F. Toni. Abstract argumentation. *Artificial Intelligence and Law Journal, Special Issue on Logical Models of Argumentation*, 4:275–296, 1996.
13. H. Prakken and G. Sartor. A system for defeasible argumentation, with defeasible priorities. In *International Conference on Formal and Applied Practical Reasoning*, volume 1085 of *Lecture Notes in Artificial Intelligence*, pages 510–524. Springer-Verlag, 1996.
14. A. S. Rao and M. Georgeff. BDI Agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems, San Francisco, California, USA*, pages 312–319, San Francisco, CA, June 1995.
15. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.
16. Kostas Stathis, Antonis C. Kakas, Wenjin Lu, Neophytos Demetriou, Ulle Endriss, and Andrea Bracciali. PROSOCS: a platform for programming software agents in computational logic. pages 523–528, Vienna, Austria, April 13-16 2004. Austrian Society for Cybernetic Studies. Extended version to appear in a special issue of Applied Artificial Intelligence, Taylor & Francis, 2005.

Metareasoning for Multi-agent Epistemic Logics*

Konstantine Arkoudas and Sel er Bringsjord

RPI
{arkouk, brings}@rpi.edu

Abstract. We present an encoding of a sequent calculus for a multi-agent epistemic logic in Athena, an interactive theorem proving system for many-sorted first-order logic. We then use Athena as a metalanguage in order to reason about the multi-agent logic an as object language. This facilitates theorem proving in the multi-agent logic in several ways. First, it lets us marshal the highly efficient theorem provers for classical first-order logic that are integrated with Athena for the purpose of doing proofs in the multi-agent logic. Second, unlike model-theoretic embeddings of modal logics into classical first-order logic, our proofs are directly convertible into native epistemic logic proofs. Third, because we are able to quantify over propositions and agents, we get much of the generality and power of higher-order logic even though we are in a first-order setting. Finally, we are able to use Athena's versatile tactics for proof automation in the multi-agent logic. We illustrate by developing a tactic for solving the generalized version of the wise men problem.

1 Introduction

Multi-agent odal logics are widely used in Co puter Science and AI. Multi-agent episte ic logics, in particular, have found applications in fields ranging fro AI do ains such as robotics, planning, and otivation analysis in natural language [1]; to negotiation and ga e theory in econo ics; to distributed syste s analysis and protocol authentication in co puter security [2, 3]. The reason is si ple—intelligent agents ust be able to reason about knowledge. It is therefore i portant to have efficient eans for perfor ing achine reasoning in such logics. While the validity proble for ost propositional odal logics is of intractable theoretical co plexity[1], several approaches have been investigated in recent years that have resulted in syste s that appear to work well in practice. These approaches include tableau-based provers, SAT-based algorith s, and translations to first-order logic coupled with the use of resolution-based auto ated theore provers (ATPs). So e representative syste s are FaCT [6], KsatC [7], TA [8], LWB [9], and MSPASS [10].

* This research was funded in part by the US Air Force Labs of Rome, NY.

[1] For instance, the validity problem for multi-agent propositional epistemic logic is PSPACE-complete [4]; adding a common knowledge operator makes the problem EXPTIME-complete [5].

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 111–125, 2005.

Translation-based approaches (such as that of MSPASS) have the advantage of leveraging the tremendous implementation progress that has occurred over the last decade in first-order theorem proving. Soundness and completeness are ensured by the soundness and completeness of the resolution prover (once the soundness and completeness of the translation have been shown), while a decision procedure is automatically obtained for any modal logic that can be translated into a decidable fragment of first-order logic, such as the two-variable fragment. Furthermore, the task of translating from a modal logic to the classical first-order setting is fairly straightforward (assuming, of course, that the class of Kripke frames captured by the modal logic is first-order definable [11]; modal logics such as the Gödel-Löb logic of provability in first-order Peano arithmetic would require translation into second-order classical logic). For instance, the well-known formula $[\Box P \wedge \Box(P \Rightarrow Q)] \Rightarrow \Box Q$ becomes

$$\begin{aligned}\forall\, w_1 \,.\, [(\forall\, w_2 \,.\, R(w_1, w_2) \Rightarrow P(w_2)) \wedge \\ (\forall\, w_2 \,.\, R(w_1, w_2) \Rightarrow P(w_2) \Rightarrow Q(w_2))] \Rightarrow (\forall\, w_2 \,.\, R(w_1, w_2) \Rightarrow Q(w_2))\end{aligned}$$

Here the variables w_1 and w_2 range over possible worlds, and the relation R represents Kripke's accessibility relation. A constant propositional atom P in the modal language becomes a unary predicate $P(w)$ that holds (or not) for a given world w.

This is the (naive) classical translation of modal logic into first-order logic [4], and we might say that it is a *semantic* embedding, since the Kripke semantics of the modal language are explicitly encoded in the translated result. This is, for instance, the approach taken by McCarthy in his "Formalizing two puzzles involving knowledge" [12]. A drawback of this approach is that proofs produced in the translated setting are difficult to convert back into a form that makes sense for the user in the original modal setting (altough alternative translation techniques such as the functional translation to path logic can rectify this in some cases [13]). Another drawback is that if a result is not obtained within a reasonable amount of time—which is almost certain to happen quite often when no decision procedure is available, as in first-order modal logics—then a batch-oriented ATP is of little help to the user due to its "low bandwidth of interaction" [14].

In this paper we explore another approach: We embed a multi-agent epistemic logic into many-sorted first-order logic in a proof-theoretic rather than in a model-theoretic way. [2] Specifically, we use the interactive theorem proving system Athena [15] to encode the formulas of the epistemic logic along with the inference rules of a sequent calculus for it. Hence first-order logic becomes our metalanguage and the epistemic logic becomes our object language. We then use standard first-order logic (our metalanguage) to reason about proofs in the object logic. In effect, we end up reasoning about reasoning—hence the term *metareasoning*. Since our metareasoning occurs at the standard first-order level, we are

[2] This paper treats a propositional logic of knowledge, but the technique can be readily applied to full first-order multi-agent epistemic logic, and indeed to hybrid multi-modal logics, e.g., combination logics for temporal and epistemic reasoning.

free to leverage existing theorem-proving systems for automated deduction. In particular, we make heavy use of Vampire [16] and Spass [17], two cutting-edge resolution-based ATPs that are seamlessly integrated with Athena.

Our approach has two additional advantages. First, it is trivial to translate the constructed proofs into modal form, since the Athena proofs are already *about proofs in the modal logic.* Second, because the abstract syntax of the epistemic logic is explicitly encoded in Athena, we can quantify over propositions, sequents, and agents. Accordingly, we get the generalization benefits of higher-order logic even in a first-order setting. This can result in significant efficiency improvements. For instance, in solving the generalized wise men puzzle it is necessary at some point to derive the conclusion $M_2 \vee \cdots \vee M_n$ from the three premises $\neg K_\alpha(M_1)$, $K_\alpha(\neg(M_2 \vee \cdots \vee M_n) \Rightarrow M_1)$, and

$$\neg(M_2 \vee \cdots \vee M_n) \Rightarrow K_\alpha(\neg(M_2 \vee \cdots \vee M_n))$$

where $M_1, \ldots, M_n$ are atomic propositions and α is an epistemic agent, $n > 1$. In the absence of an explicit embedding of the epistemic logic, this would have to be done with a tactic that accepted a list of propositions $[M_1 \cdots M_n]$ as input and performed the appropriate deduction dynamically, which would require an amount of effort quadratic in the length of the list. By contrast, in our approach we are able to formulate and prove a "higher-order" lemma stating

$$\forall\ P, Q, \alpha\ .\ \{\neg K_\alpha(P), K_\alpha(\neg Q \Rightarrow P), \neg Q \Rightarrow K_\alpha(\neg Q)\} \vdash Q$$

Obtaining the desired conclusion for any given $M_1, \ldots, M_n$ then becomes a matter of instantiating this lemma with $P \mapsto M_1$ and $Q \mapsto M_2 \vee \cdots \vee M_n$. We have thus reduced the asymptotic complexity of our task from quadratic time to constant time.

But perhaps the most distinguishing aspect of our work is our emphasis on *tactics.* Tactics are proof algorithms, which, unlike conventional algorithms, are guaranteed to produce sound results. That is, if and when a tactic outputs a result P that it claims to be a theorem, we can be assured that P is indeed a theorem. Tactics are widely used for proof automation in first- and higher-order proof systems such as HOL [18] and Isabelle [19]. In Athena tactics are called *methods*, and are particularly easy to formulate owing to Athena's Fitch-style natural deduction system and its assumption-base semantics [20]. A major goal of our research is to find out how easy—or difficult—it may be to automate multi-agent modal logic proofs with tactics. Our aim is not to obtain a completely automatic decision procedure for a certain logic (or class of logics), but rather to enable efficient interactive—i.e., semi-automatic—theorem proving in such logics for challenging problems that are beyond the scope of completely automatic provers. In this paper we formulate an Athena method for solving the generalized version of the wise men problem (for any given number of wise men). The relative ease with which this method was formulated is encouraging.

The remainder of this paper is structured as follows. In the next section we present a sequent calculus for the epistemic logic that we will be encoding. In Section 3 we present the wise men puzzle and formulate an algorithm for solving

$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \wedge Q}[\wedge\text{-}I] \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash P}[\wedge\text{-}E_1] \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash Q}[\wedge\text{-}E_2]$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q}[\vee\text{-}I_1] \qquad \frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q}[\vee\text{-}I_2]$$

$$\frac{\Gamma \vdash P_1 \vee P_2 \quad \Gamma, P_1 \vdash Q \quad \Gamma, P_2 \vdash Q}{\Gamma \vdash Q}[\vee\text{-}E]$$

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \Rightarrow Q}[\Rightarrow\text{-}I] \qquad \frac{\Gamma \vdash P \Rightarrow Q \quad \Gamma \vdash P}{\Gamma \vdash Q}[\Rightarrow\text{-}E]$$

$$\frac{\Gamma \vdash \neg\neg P}{\Gamma \vdash P}[\neg\text{-}E] \qquad \frac{\Gamma, P \vdash \bot}{\Gamma \vdash \neg P}[\neg\text{-}I] \qquad \frac{}{\Gamma, P \vdash P}[\mathit{Reflex}]$$

$$\frac{\Gamma \vdash P}{\Gamma \cup \Gamma' \vdash P}[\mathit{Dilution}] \qquad \frac{\Gamma \vdash P \wedge \neg P}{\Gamma \vdash \bot}[\bot\text{-}I] \qquad \frac{}{\Gamma \vdash \top}[\top\text{-}I]$$

Fig. 1. Inference rules for the propositional connectives

the generalized version of it in the sequent calculus of Section 2. In Section 4 we discuss the Athena encoding of the episte ic logic and present the Athena ethod for solving the generalized wise en proble . Finally, in Section 5 we consider related work.

2 A Sequent Formulation of a Multi-agent Epistemic Logic

We will use the letters P, Q, $R, \ldots$, to designate arbitrary *propositions*, built according to the following abstract gra ar:

$$P ::= A \mid \top \mid \bot \mid \neg P \mid P \wedge Q \mid P \vee Q \mid P \Rightarrow Q \mid K_\alpha(P) \mid C(P)$$

where A and α range over a countable set of ato ic propositions ("ato s") and a pri itive do ain of *agents*, respectively. Propositions of the for $K_\alpha(P)$ and $C(P)$ are read as follows:

$K_\alpha(P)$: *agent* α *knows proposition* P

$C(P)$: *it is common knowledge that* P *holds*

By a *context* we will ean a finite set of propositions. We will use the letter Γ to denote contexts. We define a *sequent* as an ordered pair $\langle \Gamma, P \rangle$ consisting of

$$\frac{}{\Gamma \vdash [K_\alpha(P \Rightarrow Q)] \Rightarrow [K_\alpha(P) \Rightarrow K_\alpha(Q)]}[K] \qquad \frac{}{\Gamma \vdash K_\alpha(P) \Rightarrow P}[\mathrm{T}]$$

$$\frac{\emptyset \vdash P}{\Gamma \vdash C(P)}[C\text{-}I] \qquad \frac{}{\Gamma \vdash C(P) \Rightarrow K_\alpha(P)}[C\text{-}E]$$

$$\frac{}{\Gamma \vdash [C(P \Rightarrow Q)] \Rightarrow [C(P) \Rightarrow C(Q)]}[C_K] \qquad \frac{}{\Gamma \vdash C(P) \Rightarrow C(K_\alpha(P))}[R]$$

Fig. 2. Inference rules for the epistemic operators

a context Γ and a proposition P. A more suggestive notation for such a sequent is $\Gamma \vdash P$. Intuitively, this is a judgment stating that P follows from Γ. We will write P, Γ (or Γ, P) as an abbreviation for $\Gamma \cup \{P\}$. The sequent calculus that we will use consists of a collection of inference rules for deriving judgments of the form $\Gamma \vdash P$. Figure 1 shows the inference rules that deal with the standard propositional connectives. This part is standard (e.g., it is very similar to the sequent calculus of Ebbinghaus et al. [21]). In addition, we have some rules pertaining to K_α and C, shown in Figure 2.

Rule $[K]$ is the sequent formulation of the well-known *Kripke axiom* stating that the knowledge operator distributes over conditionals. Rule $[C_K]$ is the corresponding principle for the common knowledge operator. Rule $[T]$ is the "truth axiom": an agent cannot know false propositions. Rule $[C_I]$ is an introduction rule for common knowledge: if a proposition P follows from the empty set of hypotheses, i.e., if it is a tautology, then it is commonly known. This is the common-knowledge version of the "omniscience axiom" for single-agent knowledge which says that $\Gamma \vdash K_\alpha(P)$ can be derived from $\emptyset \vdash P$. We do not need to postulate that axiom in our formulation, since it follows from $[C\text{-}I]$ and $[C\text{-}E]$. The latter says that if it is common knowledge that P then any (every) agent knows P, while $[R]$ says that if it is common knowledge that P then it is common knowledge that (any) agent α knows it. $[R]$ is a reiteration rule that allows us to capture the recursive behavior of C, which is usually expressed via the so-called "induction axiom"

$$C(P \Rightarrow E(P)) \Rightarrow [P \Rightarrow C(P)]$$

where E is the shared-knowledge operator. Since we do not need E for our purposes, we omit its formalization and "unfold" C via rule $[R]$ instead.
We state a few lemmas that will come handy later:

Lemma 1 (Cut). *If* $\Gamma_1 \vdash P_1$ *and* $\Gamma_2, P_1 \vdash P_2$ *then* $\Gamma_1 \cup \Gamma_2 \vdash P_2$.

Proof: Assume $\Gamma_1 \vdash P_1$ and $\Gamma_2, P_1 \vdash P_2$. Then, by $[\Rightarrow\text{-}I]$, we get $\Gamma_2 \vdash P_1 \Rightarrow P_2$. Further, by dilution, we have $\Gamma_1 \cup \Gamma_2 \vdash P_1 \Rightarrow P_2$ and $\Gamma_1 \cup \Gamma_2 \vdash P_1$. Hence, by $[\Rightarrow\text{-}E]$, we obtain $\Gamma_1 \cup \Gamma_2 \vdash P_2$. □

The proofs of the re aining le as are equally si ple exercises:

Lemma 2 (⇒-transitivity). *If* $\Gamma \vdash P_1 \Rightarrow P_2$, $\Gamma \vdash P_2 \Rightarrow P_3$ *then* $\Gamma \vdash P_1 \Rightarrow P_3$.

Lemma 3 (contrapositive). *If* $\Gamma \vdash P \Rightarrow Q$ *then* $\Gamma \vdash \neg Q \Rightarrow \neg P$.

Lemma 4. *(a)* $\emptyset \vdash (P_1 \vee P_2) \Rightarrow (\neg P_2 \Rightarrow P_1)$; *and* (b) $\Gamma \vdash C(P_2)$ *whenever* $\emptyset \vdash P_1 \Rightarrow P_2$ *and* $\Gamma \vdash C(P_1)$.

Lemma 5. *For all* P, Q, *and* Γ, $\Gamma \vdash [C(P) \wedge C(Q)] \Rightarrow C(P \wedge Q)$.

3 The Generalized Wise Men Puzzle

Consider first the three- en version of the puzzle:

> Three wise en are told by their king that at least one of the has a white spot on his forehead. In reality, all three have white spots on their foreheads. We assu e that each wise an can see the others' foreheads but not his own, and thus each knows whether the others have white spots. Suppose we are told that the first wise an says, "I do not know whether I have a white spot," and that the second wise an then says, "I also do not know whether I have a white spot." Now consider the following question: Does the third wise an now know whether or not he has a white spot? If so, what does he know, that he has one or doesn't have one?

This version is essentially identical to the uddy-children puzzle, the only difference being that the declarations of the wise en are ade sequentially, whereas in the uddy-children puzzle the children proclai what they know (or not know) in parallel at every round.

In the generalized version of the puzzle we have an arbitrary nu ber $n+1$ of wise en $w_1, \ldots, w_{n+1}$, $n \geq 1$. They are told by their king that at least one the has a white spot on his forehead. Again, in actuality they all do. And they can all see one another's foreheads, but not their own. Supposing that each of the first n wise en, $w_1, \ldots, w_n$, sequentially announces that he does not know whether or not he has a white spot on his forehead, the question is what would the last wise an w_{n+1} report.

For all $n \geq 1$, it turns out that the last—$(n+1)^{st}$—wise an knows he is arked. The case of two wise en is si ple. The reasoning runs essentially by contradiction. The second wise an reasons as follows:

> Suppose I were not arked. Then w_1 would have seen this, and knowing that at least one of us is arked, he would have inferred that he was the arked one. But w_1 has expressed ignorance; therefore, I ust be arked.

Consider now the case of $n = 3$ wise en w_1, w_2, w_3. After w_1 announces that he does not know that he is arked, w_2 and w_3 both infer that at least one of the is arked. For if neither w_2 nor w_3 were arked, w_1 would have seen this and would have concluded—and stated—that he was the arked one, since he knows that at least one of the three is marked. At this point the puzzle reduces to the two-men case: both w_2 and w_3 know that at least one of them is marked, and then w_2 reports that he does not know whether he is marked. Hence w_3 proceeds to reason as previously that he is marked.

In general, consider $n+1$ wise men $w_1, \ldots, w_n, w_{n+1}, n \geq 1$. After the first j wise men $w_1, \ldots, w_j$ have announced that they do not know whether they are marked, for $j = 1, \ldots, n$, the remaining wise men $w_{j+1}, \ldots, w_{n+1}$ infer that at least one of them is marked. This holds for $j = n$ as well, which means that the last wise man w_{n+1} will infer (and announce, owing to his honesty) that he is marked.

The question is how to formalize this in our logic. Again consider the case of two wise men w_1 and w_2. Let $M_i, i \in \{1, 2\}$ denote the proposition that w_i is marked. For any proposition P, we will write $K_i(P)$ as an abbreviation for $K_{w_i}(P)$. We will only need three premises:

$$S_1 = C(\neg K_1(M_1))$$
$$S_2 = C(M_1 \vee M_2)$$
$$S_3 = C(\neg M_2 \Rightarrow K_1(\neg M_2))$$

The first premise says that it is common knowledge that the first wise man does not know whether he is marked. Although it sounds innocuous, note that a couple of assumptions are necessary to obtain this premise from the mere fact that w_1 has announced his ignorance. First, truthfulness—we must assume that the wise men do not lie, and further, that each one of them knows that they are all truthful. And second, each wise man must know that the other wise men will hear the announcement and believe it. Premise S_2 says that it is common knowledge that at least one of the wise men is marked. Observe that the announcement by the king is crucial for this premise to be justified. The two wise men can see each other and thus they individually know $M_1 \vee M_2$. However, each of them may not know that the other wise man knows that at least one of them is marked. For instance, w_1 may believe that he is not marked, and even though he sees that w_2 is marked, he may believe that w_2 does not know that at least one of them is marked, as w_2 cannot see himself. Finally, premise S_3 states that it is common knowledge that if w_2 is *not* marked, then w_1 will know it (because w_1 can see w_2). From these three premises we are to derive the conclusion $C(M_2)$—that it is common knowledge that w_2 is marked. Symbolically, we need to derive the judgment $\{S_1, S_2, S_3\} \vdash C(M_2)$. If we have encoded the epistemic propositional logic in a predicate calculus, then we can achieve this immediately by instantiating Lemma 7 below with $\alpha \mapsto w_1$, $P \mapsto M_1$ and $Q \mapsto M_2$—without performing any inference whatsoever. This is what we have done in Athena.

For the case of $n = 3$ wise men our set of premises will be:

$$S_1 = C(\neg K_1(M_1))$$
$$S_2 = C(M_1 \vee M_2 \vee M_3)$$
$$S_3 = C(\neg(M_2 \vee M_3) \Rightarrow K_1(\neg(M_2 \vee M_3)))$$
$$S_4 = C(\neg K_2(M_2))$$
$$S_5 = C(\neg M_3 \Rightarrow K_2(\neg M_3))$$

Consider now the general case of $n+1$ wise en $w_1, \ldots, w_n, w_{n+1}$. For any $i = 1, \ldots, n$, define

$$S_1^i = C(\neg K_i(M_i))$$
$$S_2^i = C(M_i \vee \cdots \vee M_{n+1})$$
$$S_3^i = C(\neg(M_{i+1} \vee \cdots \vee M_{n+1}) \Rightarrow K_i(\neg(M_{i+1} \vee \cdots \vee M_{n+1})))$$

and $S_2^{n+1} = C(M_{n+1})$. The set of pre ises, Ω_{n+1}, can now be defined as

$$\Omega_{n+1} = \{C(M_1 \vee \cdots \vee M_{n+1})\} \bigcup_{i=1}^{n} \{S_1^i, S_3^i\}$$

Hence Ω_{n+1} has a total of $2n+1$ ele ents. Note that S_2^1 is the co only known disjunction $M_1 \vee \cdots \vee M_{n+1}$ and a known pre ise, i.e., a e ber of Ω_{n+1}. However, S_2^i for $i > 1$ is *not* a pre ise. Rather, it beco es derivable after the i^{th} wise an has ade his announce ent. Managing the derivation of these propositions and eli inating the via applications of the cut is the central function of the algorith below. Before we present the algorith we state a couple of key le as.

Lemma 6. *Consider any agent α and propositions P, Q, and let R_1, R_2, R_3 be the following three propositions:*

1. $R_1 = \neg K_\alpha(P)$;
2. $R_2 = K_\alpha(\neg Q \Rightarrow P)$;
3. $R_3 = \neg Q \Rightarrow K_\alpha(\neg Q)$

Then $\{R_1 \wedge R_2 \wedge R_3\} \vdash Q$.

Proof. By the following sequent derivation:

1.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash R_1$	[*Reflex*], $\wedge$-E_1
2.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash R_2$	[*Reflex*], $\wedge$-E_1, $\wedge$-E_2
3.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash R_3$	[*Reflex*], $\wedge$-E_2
4.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash K_\alpha(\neg Q) \Rightarrow K_\alpha(P)$	2, [K], $\Rightarrow$-E
5.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash \neg Q \Rightarrow K_\alpha(P)$	3, 4, Lemma 2
6.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash \neg K_\alpha(P) \Rightarrow \neg\neg Q$	5, Lemma 3
7.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash \neg\neg Q$	6, 1, $\Rightarrow$-E
8.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash Q$	7, [$\neg$-E]

□

Lemma 7. *Consider any agent α and propositions P, Q. Define R_1 and R_3 as in Lemma 6, let $R_2 = P \vee Q$, and let $S_i = C(R_i)$ for $i = 1, 2, 3$. Then* $\{S_1, S_2, S_3\} \vdash C(Q)$.

Proof. Let $R_2' = \neg Q \Rightarrow P$ and consider the following derivation:

1.	$\{S_1, S_2, S_3\} \vdash S_1$	[*Reflex*]
2.	$\{S_1, S_2, S_3\} \vdash S_2$	[*Reflex*]
3.	$\{S_1, S_2, S_3\} \vdash S_3$	[*Reflex*]
4.	$\emptyset \vdash (P \vee Q) \Rightarrow (\neg Q \Rightarrow P)$	Lemma 4a
5.	$\{S_1, S_2, S_3\} \vdash C((P \vee Q) \Rightarrow (\neg Q \Rightarrow P))$	4, [C-I]
6.	$\{S_1, S_2, S_3\} \vdash C(P \vee Q) \Rightarrow C(\neg Q \Rightarrow P)$	5, [C_K], [$\Rightarrow$-E]
7.	$\{S_1, S_2, S_3\} \vdash C(\neg Q \Rightarrow P)$	6, 2, [$\Rightarrow$-E]
8.	$\{S_1, S_2, S_3\} \vdash C(\neg Q \Rightarrow P) \Rightarrow C(K_\alpha(\neg Q \Rightarrow P))$	[R]
9.	$\{S_1, S_2, S_3\} \vdash C(K_\alpha(\neg Q \Rightarrow P))$	8, 7, [$\Rightarrow$-E]
10.	$\{R_1 \wedge K_\alpha(\neg Q \Rightarrow P) \wedge R_3\} \vdash Q$	Lemma 6
11.	$\emptyset \vdash (R_1 \wedge K_\alpha(\neg Q \Rightarrow P) \wedge R_3) \Rightarrow Q$	10, [$\Rightarrow$-I]
12.	$\{S_1, S_2, S_3\} \vdash C((R_1 \wedge K_\alpha(\neg Q \Rightarrow P) \wedge R_3) \Rightarrow Q)$	11, [C-I]
13.	$\{S_1, S_2, S_3\} \vdash C(R_1 \wedge K_\alpha(\neg Q \Rightarrow P) \wedge R_3) \Rightarrow C(Q)$	12, [C_K], [$\Rightarrow$-E]
14.	$\{S_1, S_2, S_3\} \vdash C(R_1 \wedge K_\alpha(\neg Q \Rightarrow P) \wedge R_3)$	1, 3, 9, Lemma 5, [$\wedge$-I]
15.	$\{S_1, S_2, S_3\} \vdash C(Q)$	13, 14, [$\Rightarrow$-E]

□

Our ethod can now be expressed as follows:

$\Phi \leftarrow \{S_1^1, S_2^1, S_3^1\}$;
$\Sigma \leftarrow \Phi \vdash S_2^2$;
Use Lemma 7 to derive Σ;
If $n = 1$ **halt**
else
  For $i = 2$ to n do
    begin
      $\Phi \leftarrow \Phi \cup \{S_1^i, S_3^i\}$;
      $\Sigma' \leftarrow \{S_1^i, S_2^i, S_3^i\} \vdash S_2^{i+1}$;
      Use Lemma 7 to derive Σ';
      $\Sigma'' \leftarrow \Phi \vdash S_2^{i+1}$;
      Use the cut on Σ and Σ' to derive Σ'';
      $\Sigma \leftarrow \Sigma''$
    end

The loop variable i ranges over the interval $2, \ldots, n$. For any i in that interval, we write Φ^i and Σ^i for the values of Φ and Σ upon conclusion of the i^{th} iteration of the loop. A straightforward induction on i will establish:

Lemma 8 (Algorithm correctness). *For any* $i \in \{2, \ldots, n\}$,

$$\Phi^i = \{C(M_1 \vee \cdots \vee M_{n+1})\} \bigcup_{j=1}^{i} \{S_1^j, S_3^j\}$$

while $\Sigma^i = \Phi^i \vdash S_2^{i+1}$.

Hence, $\Phi^n = \Omega_{n+1}$, and $\Sigma^n = \Phi^n \vdash S_2^{n+1} = \Omega_{n+1} \vdash S_2^{n+1} = \Omega_{n+1} \vdash C(M_{n+1})$, which is our goal.

It is noteworthy that no such correctness argu ent is necessary in the for- ulation of the algorith as an Athena ethod, as ethods are guaranteed to be sound. Their results are always logically entailed by the assu ption base, assu ing that our pri itive ethods are sound (see Chapter 8 of [20]).

4 Athena Implementation

In this section we present the Athena encoding of the episte ic logic and our ethod for solving the generalized version of the wise en puzzle (refer to the Athena web site [15] for ore infor ation on the language). We begin by introducing an uninterpreted do ain of episte ic agents: `(domain Agent)`. Next we represent the abstract syntax of the propositions of the logic. The following Athena datatype irrors the abstract gra ar for propositions that was given in the beginning of Section 2:

```
(datatype Prop
  True
  False
  (Atom Boolean)
  (Not Prop)
  (And Prop Prop)
  (Or Prop Prop)
  (If Prop Prop)
  (Knows Agent Prop)
  (Common Prop))
```

We proceed to introduce a binary relation `sequent` that ay obtain between a finite set of propositions and a single proposition:

```
(declare sequent (-> ((FSet-Of Prop) Prop) Boolean))
```

Here `FSet-Of` is a unary sort constructor: for any sort `T`, `(FSet-Of T)` is a new sort representing the set of all finite sets of ele ents of `T`. Finite sets are built with two poly orphic constructors: the constant `null`, representing the e pty set; and the binary constructor `insert`, which takes an ele ent x of sort `T` and a finite set S (of sort `(FSet-Of T)`) and returns the set $\{x\} \cup S$. We also have all the usual set-theoretic operations available (`union`, `intersection`, etc.).

The intended interpretation is that if `(sequent` S P`)` holds for a set of propositions S and a proposition P, then the sequent $S \vdash P$ is derivable in the episte ic logic via the rules presented in Section 2. Accordingly, we introduce axio s capturing those rules. For instance, the conjunction introduction rule is represented by the following axio :

```
(define And-I
  (forall ?S ?P ?Q
    (if (and (sequent ?S ?P)
             (sequent ?S ?Q))
        (sequent ?S (And ?P ?Q)))))
```

Note that the lowercase `and` above is Athena's built-in conjunction operator, and hence represents conjunction at the metalanguage level, whereas `And` represents the object-level conjunction operator of the epistemic logic.

The cut rule and the common knowledge introduction (necessitation) rule become:

```
(define cut
  (forall ?S1 ?S2 ?P ?Q
    (if (and (sequent ?S1 ?P)
             (sequent (insert ?P ?S2) ?Q))
        (sequent (union ?S1 ?S2) ?Q))))

(define common-intro-axiom
  (forall ?P ?S
    (if (sequent null ?P)
        (sequent ?S (Common ?P)))))
```

The remaining rules are encoded by similar first-order axioms.

We next proceed to derive several lemmas that are useful for the proof. Some of these lemmas are derived completely automatically via the ATPs that are integrated with Athena. For instance, the cut rule is proved automatically (in about 10 seconds). As another example, the following result—part (b) of Lemma 4—is proved automatically:

```
(forall ?S ?P1 ?P2
  (if (and (sequent null (If ?P1 ?P2))
           (sequent ?S (Common ?P1)))
      (sequent ?S (Common ?P2))))
```

Other lemmas are established by giving natural deduction proofs. For instance, the proof of Lemma 6 in Section 3 is transcribed virtually verbatim in Athena, and validated in a fraction of a second. (The fact that the proof is abridged—i.e., multiple steps are compressed into single steps—is readily handled by invoking ATPs that automatically fill in the details.) Finally, we are able to prove Lemma 7, which is the key technical lemma. Utilizing the higher-order character of our encoding, we then define a method `main-lemma` that takes an arbitrary list of agents $[a_1 \cdots a_n], n \geq 1$, and specializes Lemma 7 with $P \mapsto M_{a_1}$, $Q \mapsto M_{a_2} \vee \cdots \vee M_{a_n}$, and $\alpha \mapsto a_1$ (recall that for any agent α, M_α signifies that α is marked). So, for instance, the application of `main-lemma` to the list $[a_1, a_2, a_3]$ would derive the conclusion $\{S_1, S_2, S_3\} \vdash C(M_{a_2} \vee M_{a_3})$, where $S_1 = C(\neg K_{a_1}(M_{a_1}))$, $S_2 = C(M_{a_1} \vee M_{a_2} \vee M_{a_3})$, and

$$S_3 = C(\neg(M_{a_2} \vee M_{a_3}) \Rightarrow K_{a_1}(\neg(M_{a_2} \vee M_{a_3})))$$

We also need a simple result `shuffle` asserting the equality $\Gamma, P_1, P_2 = \Gamma, P_2, P_1$ (i.e., $\Gamma \cup \{P_1\} \cup \{P_2\} = \Gamma \cup \{P_2\} \cup \{P_1\}$).

Using these building blocks, we express the tactic for solving the generalized wise men problem as the Athena method `solve` below. It takes as input a list of agents representing wise men, with at least two elements. Note that the for loop in the pseudocode algorithm has been replaced by recursion.

```
(define (solve wise-men)
  (dletrec
    ((loop (method (wise-men th)
             (dmatch wise-men
               ([_] (!claim th))
               ((list-of _ rest)
                 (dlet ((new-th (!main-lemma wise-men)))
                  (dmatch [th new-th]
                    ([(sequent context Q2)
                      (sequent (insert Q1
                                 (insert Q2 (insert Q3 null))) P)]
                       (dlet ((cut-th
                                (!derive (sequent
                                           (union
                                             context
                                             (insert Q1 (insert Q3 null)))
                                           P)
                                          [th new-th shuffle cut])))
                         (!loop rest cut-th))))))))))
    (dlet ((init (!prove-goal-2 wise-men)))
      (!loop (tail wise-men) init))))
```

Assuming that `w1`, `w2`, `w3` are agents representing wise men, invoking the method `solve` with the list `[w1 w2 w3])` as the argument will derive the appropriate result: $\Omega_3 \vdash$ `(Common (isMarked w3))`, where Ω_3 is the set of premises for the three-men case, as defined in the previous section.

5 Related Work

The wise men problem became a staple of epistemic AI literature after being introduced by McCarthy [12]. Formalizations and solutions of the two-wise-men problem are found in a number of sources [22, 23, 24], most of them in simple multi-agent epistemic logics (without common knowledge). Several variations have been given; e.g., Konolige has a version in which the third wise man states that he does not know whether he is marked, but that he would know if only the second wise man were wiser [25]. Ballim and Wilks [26] solve the three-men version of the puzzle using the "nested viewpoints" framework. Vincenzo Pallotta's solution [27] is similar but his ViewGen framework facilitates agent simulation. Kim and Kowalski [28] use a Prolog-based implementation of metareasoning to solve the same version of the problem using common knowledge. A more natural proof was given by Aiello et al. [29] in a rewriting framework.

The importance of reasoning about the intentional states of intelligent agents is widely recognized (see, for instance, the recent work by Dastani et al. on inferring trust [30]). Agent metareasoning and metaknowledge, in particular, is extensively discussed in "Logical foundations of Artifical Intelligence" by Genesereth and Nillson [24] (it is the subject of an entire chapter). They stress that the main advantage of an explicit encoding of the reasoning process is that it makes it possible to “create agents capable of reasoning in detail about the inferential abilities of and beliefs of other agents,” as well as enabling introspection.[3]

The only work we are aware of that has an explicit encoding of an epistemic logic in a rich metalanguage is a recent project [32] that uses the Calculus of Constructions (Coq [33]). However, there are important differences. First, they encode a Hilbert proof system, which has an adverse impact on the readability and writability of proofs. The second and most important difference is our emphasis on reasoning efficiency. The seamless integration of Athena with state-of-the-art provers such as Vampire and Spass is crucial for automation, as it enables the user to skip tedious steps and keep the reasoning at a high level of detail. Another distinguishing aspect of our work is our heavy use of tactics. Athena uses a block-structured natural-deduction style not only for writing proofs but also for writing proof tactics (“methods”). Proof methods are much easier to write in this style, and play a key role in proof automation. Our emphasis on automation also differentiates our work from that of Basin et al. [34] using Isabelle, which only addresses proof presentation in modal logics, not automatic proof discovery.

References

1. Davis, E., Morgenstern, L.: Epistemic Logics and its Applications: Tutorial Notes. (`www-formal.stanford.edu/leora/krcourse/ijcaitxt.ps`)
2. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about knowledge. MIT Press, Cambridge, Massachusetts (1995)
3. Meyer, J., Hoek, W.V.D.: Epistemic Logic for Computer Science and Artificial Intelligence. Volume 41 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (1995)
4. Gabbay, D.M., Kurucz, A., Wolter, F., Zakharyaschev, M.: Many-dimensional modal logics: theory and applications. Volume 4 of Studies in Logic and the Foundations of Mathematics. Elsevier (1994)
5. Halpern, J., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artificial Intelligence **54** (1992) 319–379
6. Horrocks, I.: Using an expressive description logic: FaCT or fiction? In: Sixth International Conference on Principles of Knowledge Representation and Reasoning. (1998) 636–647
7. Giunchiglia, E., Giunchiglia, F., Sebastiani, R., Tacchella, A.: More evaluation of decision procedures for modal logics. In Cohn, A.G., Schubert, L., Shapiro, S.C., eds.: 6th international conference on principles of knowledge representation and reasoning (KR’98), Trento (1998)
8. Hustadt, U., Schmidt, R.A.: On evaluating decision procedures for modal logic. In: Fifteenth International Joint Conference on Artificial Intelligence. (1997) 202–209

[3] In addition, Bringsjord and Yang [31] have claimed that the best of human reasoning is distinguished by a capacity for meta-reasoning, and have proposed a theory—mental metalogic—of human and machine reasoning that emphasizes this type of reasoning.

9. Heuerding, A.: LWBtheory: information about some propositional logics via the WWW. Logic Journal of the IGPL **4** (1996) 169–174
10. Schmidt, R.A.: MSPASS. `http://www.cs.man.ac.uk/~schmidt/mspass/` (1999)
11. Fitting, M.: Basic modal logic. In Gabbay, D.M., Hogger, C.J., Robinson, J.A., eds.: Logical foundations. Volume 4 of Handbook of Logic in Artificial Intelligence and Logic Programming. Oxford Science Publications (1994)
12. McCarthy, J.: Formalization of two puzzles involving knowledge. In Lifschitz, V., ed.: Formalizing Common Sense: Papers by John McCarthy. Ablex Publishing Corporation, Norwood, New Jersey (1990)
13. Schmidt, R.A., Hustadt, U.: Mechanised reasoning and model generation for extended modal logics. In de Swart, H.C.M., Orlowska, E., Schmidt, G., Roubens, M., eds.: Theory and Applications of Relational Structures as Knowledge Instruments. Volume 2929 of Lecture Notes in Computer Science. Springer (2003) 38–67
14. Cyrluk, D., Rajan, S., Shankar, N., , Srivas, M.: Effective theorem proving for hardware verification. In: Theorem Provers in Circuit Design (TPCD '94). Volume 901 of Lecture Notes in Computer Science., Bad Herrenalb, Germany, Springer-Verlag (1994) 203–222
15. Arkoudas, K.: Athena. (`http://www.pac.csail.mit.edu/athena`)
16. Voronkov, A.: The anatomy of Vampire: implementing bottom-up procedures with code trees. Journal of Automated Reasoning **15** (1995)
17. Weidenbach, C.: Combining superposition, sorts, and splitting. In Robinson, A., Voronkov, A., eds.: Handbook of Automated Reasoning. Volume 2. North-Holland (2001)
18. Gordon, M.J.C., Melham, T.F.: Introduction to HOL, a theorem proving environment for higher-order logic. Cambridge University Press, Cambridge, England (1993)
19. Paulson, L.: Isabelle, A Generic Theorem Prover. Lecture Notes in Computer Science. Springer-Verlag (1994)
20. Arkoudas, K.: Denotational Proof Languages. (PhD dissertation, MIT, 2000)
21. Ebbinghaus, H.D., Flum, J., Thomas, W.: Mathematical Logic. 2nd edn. Springer-Verlag (1994)
22. Huth, M., Ryan, M.: Logic in Computer Science: modelling and reasoning about systems. Cambridge University Press, Cambridge, UK (2000)
23. Snyers, D., Thayse, A.: Languages and logics. In Thayse, A., ed.: From modal logic to deductive databases, John Wiley & Sons (1989) 1–54
24. Genesereth, M., Nilsson, N.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann (1987)
25. Konolige, K.: A deduction model of belief. Research Notes in Artificial Intelligence. Pitman, London, UK (1986)
26. Ballim, A., Wilks, Y.: Artificial Believers. Lawrence Erlbaum Associates, Hillsdale, New Jersey (1991)
27. Pallotta, V.: Computational dialogue Models. In: 10th Conference of the European Chapter of the Association for Computational Linguistics EACL03. (2003)
28. Kim, J., Kowalski, R.: An application of amalgamated logic to multi-agent belief. In Bruynooghe, M., ed.: Second Workshop on Meta-Programming in Logic META90. (1990) 272–283
29. Aiello, L.C., Nardi, D., Schaerf, M.: Yet another solution to the three wisemen puzzle. In: Proceedings of the 3rd International Symposium on Methodologies for Intelligent Systems. (1988) 398–407
30. Dastani, M., Herzig, A., Hulstijn, J., van der Torre, L.: Inferring trust. In this volume.

31. Yang, Y., Bringsjord, S.: Mental Metalogic: A New, Unifying Theory of Human and Machine Reasoning. Erlbaum, Mahwah, NJ (2005)
32. Lescanne, P.: Epistemic logic in higher order logic: an experiment with COQ. Technical Report RR2001-12, LIP-ENS de Lyon (2001)
33. Coquand, T., Huet, G.: The Calculus of Constructions. Information and Computation **76** (1988) 95–120
34. Basin, D., Matthews, S., Viganò, L.: A modular presentation of modal logics in a logical framework. In Ginzburg, J., Khasidashvili, Z., Vogel, C., Lévy, J.J., Vallduví, E., eds.: The Tbilisi Symposium on Logic, Language and Computation: Selected Papers. CSLI Publications, Stanford, CA (1998) 293–307

Graded BDI Models for Agent Architectures*

A. a Ca a ı[1], L . . G. d. [2], a. d Ca. e Sıe. . a[2]

[1] Depto. de Sistemas e Informática,
Facultad de Cs. Exactas, Ingeniería y Agrimensura,
Universidad Nacional de Rosario,
Av Pellegrini 250, 2000 Rosario, Argentina
[2] Institut d'Investigació en Intel·ligència Artificial (IIIA) - CSIC,
Campus Universitat Autònoma de Barcelona s/n,
08193 Bellaterra, Catalunya, España

Abstract. In the recent past, an increasing number of multiagent systems (MAS) have been designed and implemented to engineer complex distributed systems. Several previous works have proposed theories and architectures to give these systems a formal support. Among them, one of the most widely used is the BDI agent architecture presented by Rao and Georgeff. We consider that in order to apply agents in real domains, it is important for the formal models to incorporate a model to represent and reason under uncertainty. With that aim we introduce in this paper a general model for graded BDI agents, and an architecture, based on multi-context systems, able to model these graded mental attitudes. This architecture serves as a blueprint to design different kinds of particular agents. We illustrate the design process by formalising a simple travel assistant agent.

1 Introduction

I. he . ece. a , a. ı. c. ea ı. g . . be. . f . ıage. . . e . (MAS) ha e bee. de ıg. ed a. d ı e e. ed . e. gı. ee. c. . e dı . ıb ed . . e . . Se. e. a . e. ı. . . . ha e . . . ed he. . ıe a. d a. chı ec . e . gı e he e. . e . a f. . a . . . . Age. he. . ıe a. e e . e. ıa . . ecı ca ı. . . f age. . ' beha ı. . e . e. ed a he . . e. ıe ha age. . h d ha e. A f. . a . e . e e. a ı. . f he . . e. ıe he he de ıg. e. . . ea . . ab. he e ec ed beha ı. . . f he . . e [25]. Age. a. chı ec . e . e . e e. a. ıdd e . ı. be ee. . ecı ca ı. . a. d ı e e. a ı. . . The ıde. ıf he. aı. f . c ı. . . ha ı a e de e. . ı e he age. ' beha ı. . a. d de . e he ı. e. de e. de. cıe ha e ı a . . g he [25]. Age. he. . ıe ba ed . . a. ı. e. ı. . a . a. ce a. e a . . g he . . c. . . . . . e . I. e. ı. . a . . e . de c. ıbe e. ı ıe h. . e beha ı. . ca. be . edıc ed

* A preliminary version of this paper, "Modelos BDI graduados para Arquitecturas de Agentes" (in Spanish), was presented at the Argentine Symposium on Artificial Intelligence (ASAI'04) and will appear in an especial issue of "Inteligencia Artificial" (Revista Iberoamericana de Inteligencia Artificial).

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 126–143, 2005.

b he. e h. d . f a ,ıb ı. g ce, aı. . e. a ı ıc a ı de . ch a . . edge, be ıef *information attitudes*, de ı, e, ı. e. ı. . , . b ıga ı. . , c. . . ı . e. *pro-attitudes*, a . . g . he, . [5]. A e - . . . ı. e. ı. . a . . e f. , . a a . , ach ı he BDI a, chı ec , e , . . . ed b Ra. a. d Ge. , ge [20, 21]. Thı . . de ı ba ed . . he e ıcı , e , e e. a ı. . . f he age. ' be ıef (B) . ed . , e , e e. he . a e . f he e. . ı, . . . e. , ı . de ı, e (D) . ed . , e , e e. he . . ı a ı. . . f he age. , a. d ı . ı. e. ı. . . (I) . ed . , e , e e. he g. a. . f he age. . Thı a, chı ec , e ha e . . ed . , e, ı e a. d ı ha bee. a ıed ı. . e. e, a . f he . . . ıg. ı ca. . ıage. a ıca ı. . . de e. . ed . . . .

M. de ı. g dı e, e. ı. e. ı. . a . . ı. . . b ea. . . f . e. e, a . . da ı ıe (B, D, I) ca. be . e, c. . e ıf . . . . e . gıca f, a e . , ı . ed. I. . , de, . he ı. he de ıg. . f . ch c. . e . gıca . . e . Gı . chıg ıa e .a . [9] ı. . , d ced he . . ı. . . f *multi-context system* (MCS f. , . h. ,). Thı f, a e . , a . . he de . ı ı. . . f dı e, e. f. , . a c. . . . e. . a. d heı, ı. e, , e a ı. . . I. . , ca e, e , ... e . . e . e a, a e c. . e . . , e , e e. each . . da ı a. d f. , . a ı e each c. . e ı h he . . . a , . . , ıa e . gıc a a, a . . The ı. e, ac ı. . . be ee. he c. . . . . e. . a, e . . ecı ed b . ı. g ı. e, - . ı , e , ca ed *bridge rules*. The e , e a, e a, . f he ded c ı. . . achı. e, . f he . . e . Thı a , . ach ha bee. . ed b Saba e, e .a . [22] a. d Pa, ... e .a . [19] . . . ecıf . e. e, a age. a, chı ec , e a. d a, ıc a, . . . de . . . e c a . e . f BDI age. . [17]. I. deed . . e ad. a. age . f he MCS . gıca a , . ach . age. c . . de ı. g ı ha ı a . . f. , . a he, a . . dab e c. . a ı. . a ı e e. a ı. . . F. , ı. . a. ce, a . . ı. . . f he f, a e . , de c, ıbed ı. [17] ı beı. g . . ı e e. ed . ı. g a , . . g. ı-h, eaded a, chı ec , e [8].

The age. a, chı ec , e , . . . ed . . fa, . . . dea ı h . - a ed ı. f. , - . a ı. . . A h. gh he BDI . . de de. e . . ed b Ra. a. d Ge. , ge e ıcı ac- . . edge ha a. age. ' . . de . f he . , d ı ı. c. . e e, b . . de ı. g be ıef a a . e . f . , d ha he age. . . . ha ı . ıgh be ı. , ı . a e . . . e . f a. ı ed ı. f. , . a ı. . ab. h . . . ıb e a a, ıc a, . , d ı . be he ac a . . e. Neı he, d. e ı a . de ı, e a. d ı. e. ı. . . . be a. ı ed. We hı. ha a ı. g ı. . c. . . ıde, a ı. . hı g, aded ı. f. , . a ı. . c. d ı . . . e he age. ' e, - f. , . a. ce. The, e a, e a fe . . , . ha a, ıa add, e . hı ı. e a. d e ha ı e he ı . . , a. ce . f g, aded . . de . . N. ab , Pa, ... a. d Gı. . gı. ı [17] c. . . ıde, he be ıef a. ı ca ı. . b . ı. g E. ıde. ce The. , . I. heı, , ... a , a. age. ı a . ed e , e . ı . . . ı. ı. . . . he , e ıabı ı . f he age. . ı ı. e, ac . ı h, a. d . , e ı e ı. be ıef he. he bec. . e ı. c. . . ı e . . The . e . . he ı - . . , a. ce . f a. ıf ı. g deg, ee ı. de ı, e a. d ı. e. ı. . . , b hı ı . . c. . e, ed b heı, . . , . La g e a . [14] , e e. a a , . ach . a . gıc . f de ı, e , he, e he . . ı. . . f hıdde. . ce, aı. . f de ı, e ı ı. . . d ced. De ı, e a, e f. , . a ı ed . . . . . . , a , ea ı ıc ı. e, ac ı. . . be ee. he c. . ce . . f , efe, e. ce a. d a - . ıbı ı (. , . . . , . a ı), b. h , e , e e. ed b a , e-, de, , e a ı. . . . e, he . e . . f . . . ıb e . . d . O he, . , . dea ı h , ea . . ı. g ab. ı. e. ı. . . ı. . ce, aı. d. . aı. . , a he , ... a . f Sch e a . [24]. The , e e. a. e cıe. ı. e. ı. . , ec. . . ıde, a ı. . f. , age. . ha ı. e, ac ı. a. . ce, aı. e. . ı, . . . e. ı. e, . . . f d . a ıc , . b e, . abı ı , a. d . . . -de e, . . ı. ı. . .

A he ab. . e. e. ı. . ed ., . . . a. . . de a, ıa a ec . . f he . ce, aı. ,e a ed . . e. a . . ı. . . ı. . . . ed ı. a. age. ' a,chı ec ,e. We ,e e. ı. hı a e, a ge. e,a . . de f. , a g,aded BDI age. , . ecıf ı. g a. a,chı ec ,e ab e . dea ı h he e. . ı,. . . e. . ce, aı. a. d ı h g,aded . e. a a ı de . I. hı . e. . e, be ıef deg,ee ,e ,e e. . ha e e. he age. be ıe e a f. ,. a ı , e. Deg,ee . f . . ı ı e. , . ega ı e de ı,e a . he age. . . e dı e,e. e. e. . f ,efe,e. ce . , ,e ec ı. . ,e ec ı e . I. e. ı. . deg,ee gı e a. . a ,efe,e. ce . ea ,e b , ı. hı ca e, . . de ı. g he c. . /be. e ,ade . . . f ,eachı. g a. age. ' g. a . The. , Age. . ha ı. g dı e,e. ı. d . f beha ı. , ca. be . . de ed . . he ba ı . f he ,e ,e e. a ı. . a. d ı. e,ac ı. . . f he e h,ee a ı de .

Thı a e, ı . ,ga. ı ed a f. . . : ı. Sec ı. . 2, e ı. ,. d ce . ı-c. . e . . e . a. d he ge. e,a . ı a ed . gıc f,a e . , f. , he g,aded c. . e . . Sec ı. . 3, 4, a. d 5 ,e e. he . e. a . ı. . f he g,aded BDI . . de , ha ı he c. . e . f. , be ıef (BC), de ı,e (DC), a. d ı. e. ı. . (IC). Sec ı. . 6 . ı. e . f . c ı. . a c. . e . f. , a. . ı. g (PC) a. d c. . . . ıca ı. . (CC). I. Sec ı. . 7, e dea ı h b,ıdge , e , e ı . ,a e he . . e,a ,ea . . ı. g ,. ce . ı. Sec ı. . 8, a. d . . a , e ,e e. . . . e c. . c . ı. . . a. d f ,e ı. e . f . , .

2 Graded BDI Agent Model

The a,chı ec ,e ,e e. ed ı. hı a e, ı ı. . ı,ed b he . , . f Pa,. . . . e .a . [17] ab. ı-c. . e BDI age. . . M ı-c. . e . . e . e,e ı. , . d ced b Gı . chıg ıa e .a . [9] . a dı e,e. f. , a (. gıc) c. . . . e. . . be de . ed a. d ı. e, ,e a ed. The MCS . ecı ca ı. . . f a. age. c. . aı. . h,ee ba ıc c. . . -. e. . : . ı. . , c. . e . , . gıc , a. d b,ıdge , e , hıch cha. . e he ,. aga ı. . f c. . . e e. ce a . . g he. ,ıe . Th . , a. age. ı de . ed a a g,. . f ı. e,-c. . . ec ed . ı. : $\langle \{C_i\}_{i\in I}, \Delta_{br}\rangle$, he,e each c. . e $C_i \in \{C_i\}_{i\in I}$ ı he e $C_i = \langle L_i, A_i, \Delta_i\rangle$ he,e L_i, A_i a. d Δ_i a,e he a. g age, a ı. . , a. d ı. fe,e. ce , e ,e ec ı e . The de . e he . gıc f. , he c. . e a. d ı. ba ıc beha ı. , a c. . . ,aı. ed b he a ı. . . Whe. a he. , $T_i \in L_i$ ı a. . cıa ed ı h each . ı , he ı e e. a ı. . f a a, ıc a, age. ı c. . e e. Δ_{br} ca. be . de,-. . . d a , e . f ı. fe,e. ce ı h ,e ı e a. d c. . c . ı. . . ı. dı e,e. c. . e . , f. , ı. . a. ce:

$$\frac{C_1 : \psi, C_2 : \varphi}{C_3 : \theta}$$

. ea. . ha ıf f. ,. . a ψ ı ded ced ı. c. . e C_1 a. d f. ,. . a φ ı ded ced ı. c. . e C_2 he. f. ,. . a θ ı added . c. . e C_3.

The ded c ı. . . echa. ı . f he e. . e . ı ba ed . . . ı. d . f ı. fe,e. ce , e , ı. e,. a , e Δ_i ı. ıde each . ı , a. d b,ıdge , e Δ_{br} . ıde. I. e,. a , e a . d,a c. . e e. ce ı hı. a he. , , hı e b,ıdge , e a . . e bed ,e f,. . a he. , ı. . a. . he, [7].

We ha e *mental* c. . e . . ,e ,e e. be ıef (BC), de ı,e (DC) a. d ı. e. -ı. . . (IC). We a. . c. . . ıde, *functional* c. . e . : f. , P a. . ı. g (PC) a. d C. . . ıca ı. . (CC). The P a. . e, ı ı. cha,ge . f . . dı. g a. . . cha,ge he c e d ı a he d, he e e g a ı a ı ed, a d f c ı g he c a cıa ed he a . The c ıca ı c e ı he age ' d he e e a d, ecei ı g a d e dı g e age . I a , he BDI age de ı de ed a :

$$A_g = (\{BC, DC, IC, PC, CC\}, \Delta_{br})$$

Each c e ha a a cıa ed gıc, ha ı , a gıca a g age ı h ı e a ıc a d ded c ı e e . I de e e e a d ea ab g aded ı f be ıef , de ı e a d ı e ı , e decıde e a da a - a ed a ach. I a ıc a , e ha f he a ach de e ed b Ha e e a . ı e.g. [12] a d [10] he e ce aı ea ı g ı dea ı h b de ı g ı ab e da he ıe e ı ab e a - a ed gıc . The ba ıc ıdea ı he f ı g. F ı a ce, e c ıde a Be ıef c e he e be ıef deg ee a e be de ed a babı ı ıe . The , f each c a ıca (- a ed) f a φ, e c ıde a da f a $B\varphi$ hıch ı ı e e ed a φ ı bab e . Thı da f a $B\varphi$ ı he a *fuzzy* f a hıch a be e e e, de e dı g he babı ı f φ. I a ıc a , e ca a e a h- a e f $B\varphi$ ecı e he babı ı f φ. M e e , ı g a a - a ed gıc, e ca e e he g e ı g a ı f babı ı he a gıca a ı ı ı g da f ae f he ı d $B\varphi$. The , he a - a ed gıc achı e ca be ed ea ab he da f ae $B\varphi$, hıch faı hf e ec he ce aı de ch e e e e he deg ee f be ıef.

I hı a , f he e a c e e ch e he ı ı e- a ed L a ıe ıc gıc b a he e ec ı f a - a ed gıc a be d e f each ı , acc dı g he ea e de ed ı each ca e [1]. The ef e, ı hı ı d f gıca f a e e ha ha e, be ıde he a ı f L a ıe ıc a - a ed gıc, a e f a ı c e dı g he ba ıc a e f a a ıc a ce aı he . He ce, ı hı a ach, ea ı g ab b- abı ı ıe (a he ce aı de) ca be d e ı a e e ega a ı hı a ıf a d e ıbe gıca f a e . The a e a - a ed gıca f a e a be ed e e e a d ea ab deg ee f de ı e a d ı e ı , a ı be ee ı de aı a e .

3 Belief Context

The e f hı c e ı de he age ' be ıef ab he e ı e . I de e e e be ıef , e e da a - a ed f ae, f ı g he ab e e ı ed gıca f a e . We c ıde ı hı a e he a ıc a

[1] The reason of using this many-valued logic is that its main connectives are based on the arithmetic addition in the unit interval $[0, 1]$, which is what is needed to deal with additive measures like probabilities. Besides, Łukasiewicz logic has also the *min* conjunction and *max* disjunction as definable connectives, so it also allows to define a logic to reason about degrees of necessity and possibility.

ca e . f . ı. g . . babı ı he. . a he . ce. aı . . de . O he. . . de. . ıgh be . ed a e b . . . . dıf ı. g he c. . . e . . dı. g a ı. . . .

3.1 The BC Language

T. . ea . . ab. he c. edıbı ı . f c. ı . . . . ı ı. . . , e de . e a a. g age f. . be ıef . e . e e. a ı. . , f. . ı. g G. d. e a .' [10], ba ed . . L a ıe ıc . gıc. I . . de. . de . e he ba ıc c. ı a. g age, e . a. f. . . a c a. ıca . . . . ı ı. . a a. g age L, de . ed . . . a c. . ab e . e . f . . . . ı ı. . a . a. ıab e PV a. d c. . . ec ı. e $(\neg, \rightarrow)$, a. d e e. d ı . . e . e e. ac ı. . . . We a e ad a. age . f D . a ıc . gıc hıch ha bee. . ed . . . de age. ' ac ı. . . ı. [23] a. d [16]. The e ac ı. . . , he e. . ı. . . . e. . a. . f. . . a ı. . . he ca . e, a. d heı. a . . cıa ed c. . . . . be a. . f a. . ı a ed age. ' be ıef . e .

The . . . . ı ı. . a a. g age L ı h . e e. ded . L_D, b addı. g . ı ac ı. . . . daı ıe . f he f. . . $[\alpha]$ he. e α ı a. ac ı. . . M. . e c. . c. e e , gı e. a . e Π_0 . f . b. . . e . e e. ı. g e e e. a. ac ı. . . , he . e Π . f a. . (c. . . . ı e ac ı. . .) a. d f. . . ae L_D ı de . ed a f. . . . :

- $\Pi_0 \subset \Pi$ (e e e. a. ac ı. . . a. e a. .)
- ıf $\alpha, \beta \in \Pi$ he. $\alpha; \beta \in \Pi$, (he c. . ca e. a ı. . . f ac ı. . . ı a. . a a.)
- ıf $\alpha, \beta \in \Pi$ he. $\alpha \cup \beta \in \Pi$ (. . . -de e. . ı. ı ıc dı . c ı. .)
- ıf $\alpha \in \Pi$ he. $\alpha^* \in \Pi$ (ı e. a ı. .)
- If A ı a f. . . a, he. $A? \in \Pi$ (e)

- ıf $p \in PV$, he. $p \in L_D$
- ıf $\varphi \in L_D$ he. $\neg\varphi \in L_D$
- ıf $\varphi, \psi \in L_D$ he. $\varphi \rightarrow \psi \in L_D$
- ıf $\alpha \in \Pi$ a. d $\varphi \in L_D$ he. $[\alpha]\varphi \in L_D$.

The ı. e. . e a ı. . . f $[\alpha] A$ ı *"after the execution of α, A is true"*

We de . e a . . da a. g age BC . . e. he a. g age L_D . . ea . . ab. he be ıef . . c. ı . . . . ı ı. . . . T. d. . . , e e e. d he c. ı a. g age L_D ı h a f. . . . a. . . da . e. a . . B. If φ ı a . . . . ı ı. . ı. L_D, he ı. e. ded . ea. ı. g . f $B\varphi$ ı ha φ ı be ıe. ab e . F. . . ae . f BC a. e . f . e :

- *Crisp (non B-modal):* he a. e he (c. ı) f. . . ae . f L_D, b ı ı. he . . a a , h . , ıf $\varphi \in L_D$ he. $\varphi \in BC$.
- *B-Modal:* he a. e b ı f. . . e e e. a. . . da f. . . ae $B\varphi$, he. e φ ı c. ı , a. d . h c. . . a. $\overline{r}$, f. . each . a ı. . a $r \in [0, 1]$, . ı. g he c. . . ec ı. e . f L a ıe ıc . a. - . a ed . gıc:
  - If $\varphi \in L_D$ he. $B\varphi \in BC$
  - If $r \in Q \cap [0, 1]$ he. $\overline{r} \in BC$
  - If $\Phi, \Psi \in BC$ he. $\Phi \rightarrow_L \Psi \in BC$ a. d $\Phi \& \Psi \in BC$ (he. e $\&$ a. d $\rightarrow_L$ c. . . e . . . d . he c. . . c ı. . a. d ı ıca ı. . . f L a ıe ıc . gıc)

O he. L a ıe ıc . gıc c. . . ec ı. e f. . he. . da f. . . ae ca. be de . ed f. . . $\&$, $\rightarrow_L$ a. d $\overline{0}$: $\neg_L \Phi$ ı de . ed a $\Phi \rightarrow_L \overline{0}$, $\Phi \wedge \Psi$ a $\Phi \& (\Phi \rightarrow_L \Psi)$, $\Phi \vee \Psi$ a $\neg_L(\neg_L \Phi \wedge \neg_L \Psi)$, a. d $\Phi \equiv \Psi$ a $(\Phi \rightarrow_L \Psi) \& (\Psi \rightarrow_L \Phi)$.

Si ce i L a ie ic gic a f a $\Phi \rightarrow_L \Psi$ i 1- e i he h a e f Ψ i g ea e e a ha f Φ, da f ae f he e $\overline{r} \rightarrow_L B\varphi$ e e ha he babi i f φ i a ea r. F ae f he e $\overline{r} \rightarrow_L \Psi$ i be de ed a (Ψ, r).

3.2 Belief Semantics

The e a ic f he a g age BC i de ed, a a i da gic , i g a K i e c e. We ha e added ch c e a ρ f c i i de e e e he d a i i ca ed b ac i , a d a babi i ea e μ e d . Th , e de e a BC babi i ic K i e c e a a 4- e $K = \langle W, e, \mu, \rho \rangle$ he e:

- W i a -e e f ib e d .
- $e : V \times W \rightarrow \{0, 1\}$ ide f each d a B ea (- a ed) e a a i f he i i a a iab e , ha i , $e(p, w) \in \{0, 1\}$ f each i i a a iab e $p \in V$ a d each d $w \in W$. The e a a i i e e ded a bi a f ae i L_D a de c ibed be .
- $\mu : 2^W \rightarrow [0, 1]$ i a i e addi i e babi i ea e a B ea a geb a f b e f W ch ha f each c i φ, he e $\{w \mid e(\varphi, w) = 1\}$ i ea ab e [12].
- $\rho : \Pi_0 \rightarrow 2^{W \times W}$ a ig each e e e a ac i a e f ai f d de i g d a i i .

Extension of e to L_D formulae:
e i e e ded L i g c a ica c ec i e a d f ae i h ac i da- i ie a $[\alpha] A$, b de i g $\rho(\alpha; \beta) = \rho(\alpha) \circ \rho(\beta)$, $\rho(\alpha \cup \beta) = \rho(\alpha) \cup \rho(\beta)$, $\rho(\alpha^*) = (\rho(\alpha))^*$ (a ce a e a i) a d $\rho(\varphi?) = \{(w, w) \mid e(\varphi, w) = 1\}$, a d e i g $e([\alpha] A, w) = min\{e(A, w_i) \mid (w, w_i) \in \rho(\alpha)\}$. N ice ha $e([\alpha] A, w) = 1$ i he e a a i f A i 1 i a he d w' ha a be eached h gh he ac i α f w.
Extension of e to B-modal formulae:
e i e e ded B- da f ae b ea f L a ie ic gic h-f c i a d he babi i ic i e e a i f be ief a f :

- $e(B\varphi, w) = \mu(\{w' \in W \mid e(\varphi, w') = 1\})$, f each c i φ
- $e(\overline{r}, w) = r$, f a $r \in Q \cap [0, 1]$
- $e(\Phi \& \Psi, w) =$ a $(e(\Phi) + e(\Psi) - 1, 0)$
- $e(\Phi \rightarrow_L \Psi, w) =$ i $(1 - e(\Phi) + e(\Psi), 1)$

Fi a , he h deg ee f a f a Φ i a K i e c e $K = \langle W, e, \mu, \rho \rangle$ i de ed a $\|\Phi\|^K =$ i $f_{w \in W}\, e(\Phi, w)$.

3.3 BC Axioms and Rules

A e i ed i Sec i 2, e a ade a e a i a i a i f be- ief c e gic e eed c bi e a i f he c i f ae, a i f

L a ie ic . gic f . . da f . ae, a d addi i . a a i . . f . B- . da f . - . ae acc . di g . he . . babi i ic e a ic . f he B . e a . . He ce, a i . . a d . e f . he Be ief c . . e . gic BC a e a f . . . :

1. A i . . . f . . . . i i . . a D . a ic . gic f . L_D f . . ae (ee e.g. [11]).
2. A i . . . f L a ie ic . gic f . . . da f . . ae: f . i . a ce, a i . . . f Ha e ' Ba ic L gic (BL) [12] . . he a i . : $\neg\neg\Phi \to \Phi$
3. P . . babi i ic a i . . .
   $B(\varphi \to \psi) \to_L (B\varphi \to B\psi)$
   $B\varphi \equiv \neg_L B(\varphi \wedge \neg\psi) \to_L B(\varphi \wedge \psi)$
   $\neg_L B\varphi \equiv B\neg\varphi$
4. Ded c i . . . e f . BC a e: . . d . . . . e . , . ece . i a i . f . $[\alpha]$ f . each $\alpha \in \Pi$ (f . . φ de i e $[\alpha]\varphi$), a d . ece . i a i . f . B (f . . φ de i e $B\varphi$).

Ded c i . i de . ed a . . a f . . he ab . e a i . . a d . e a d i be de . ed b $\vdash_{BC}$. N . ice ha , a i g i . acc . . L a ie ic . e a ic , he . ec . d *probabilistic axiom* c . . e . . d . he . i e addi i i hi e he hi d . e e . e . e ha he . . babi i . f $\neg\varphi$ i 1 . i . he . . babi i . f φ. Ac a , . . e ca . h . ha he ab . e a i . a ic i . . . d a d c . . e e i h e ec . he i e ded . e a ic de c ibed i . he . e i . . . b ec i . (cf. [12]). Na e , if T i a . i e he . . . e BC a d Φ i a (. da) f . . a, he $T \vdash \Phi$ i $\|\Phi\|^K = 1$ i each BC . . babi i ic K i e . . c . e K . de . f T (i.e. K . ch ha $\|\Psi\|^K = 1$ f . a $\Psi \in T$).

4 Desire Context

I hi c . e , e e . e e he age . ' de i e . De i e . e . e e he *ideal* age . ' . efe e ce . ega d e . . f he age . ' c . . e . e ce i . . f he e . i . . - . e a d . ega d e . . f he c . . i . . . ed i ac a achie i g he . We dee i . . a . di i g i h ha i . . i i e de i ed f . . ha i . . e ec ed. Acc . di g . he . . . . . bi . a i . e . e e a i . . f . efe e ce b Be fe ha e .a . [2], . . i i e a d . ega i e i f . . a i . . a be . de ed i he f a e . . . f . . . ibi i ic . gic. I . i ed b hi . . , e . gge . f . . a i e age . ' de i e a . . a . . i i e a d . ega i e. P . . i i e de i e . e . e e ha he age . . d i e . be he ca e. Nega i e de i e c . . e . . d . ha he age . e ec . . d e . . a . . cc . . B h, . . i i e a d . ega i e de i e ca be g aded.

4.1 DC Language

The a g age DC i de . ed a a e e . i . . f a . . . i i . a a g age L b i . . d ci g . (f . .) . da . e a . . D^+ a d D^-. $D^+\varphi$ ead a φ i . . - i i e de i ed a d i . h deg ee . e . e e . he age . ' e e . f a i fac i . . d φ bec . . e . e. $D^-\varphi$ ead a φ i . ega i e de i ed a d i . . h de- g ee . e . e e . he age . ' . ea . e . f di g . . φ bec . i g . e. A i BC . gic, e i . e a . . da . a - a ed . gic . f . . a i e g aded de i e . We . e agaı L a ıe ıc . gıc a he ba e . gıc, b hı ı e e e. ded ı h a. e c. . -. ec ı. e Δ (. . . a Baa. ' c. . . ec ı. e), c. . . ıde. ed a. . ı. [12]. F. , a. . . da Φ, ıf Φ ha a e < 1 he. $\Delta\Phi$ ge . a e 0; . he. ı e, ıf Φ ha a e 1 he. $\Delta\Phi$ ge . a e 1 a e . He. ce $\Delta\Phi$ bec. . e a . - a ed (B. . ea.) f. , . a. The. ef. , e, DC f. , . ae a. e . f . e :

– *Crisp (non modal):* f. , . ae . f L
– *Many-valued (modal):* he a. e b ı f. . . e e e. a. . . da f. , . ae $D^+\varphi$ a. d $D^-\varphi$, he. e φ ı f. . . L, a. d , h c. . . a. . $\overline{r}$ f. , each , a ı. . a $r \in [0,1]$:
  - If $\varphi \in L$ he. $D^-\varphi, D^+\varphi \in DC$
  - If $r \in Q \cap [0,1]$ he. $\overline{r} \in DC$
  - If $\Phi, \Psi \in DC$ he. $\Phi \to_L \Psi \in DC$ a. d $\Phi\&\Psi \in DC$

A ı. BC, $(D\psi, r)$ de. . e $r \to_L D\psi$.

I. hı c. . e he age. ' . efe. e. ce ı be e . e . ed b a he. . T c. . -aı. ı. g a. ı a ı. e e . e . ı. . . ab . . ı ı. e a. d . ega ı. e . efe. e. ce , ı e $(D^+\varphi, \alpha)$ . . $(D^-\psi, \beta)$, a e a a ı a ı. e e . e . ı. . . ı e $D^+\psi \to_L D^+\varphi$ (. e . $D^-\psi \to_L D^-\varphi$), e . e . ı. g ha φ ı a ea a . efe. . ed (. e . . e ec ed) a ψ. I. a. ıc a. $(D^+\phi_i, 1) \in T$. ea. . ha he age. ha . a ı . . efe. e. ce ı. ϕ_i a. d ı f . a ı . ed ıf ı ı . e. Whı e $(D^+\phi_j, \alpha) \notin T$ f. , a. $\alpha > 0$. ea. . ha he age. ı ı. dı e. e. . ϕ_j a. d he age. d . e . ' be. e f. . . he , h . f ϕ_j. A. a . g. , $(D^-\psi_i, 1) \in T$. ea. . ha he age. ab . e . e ec . ϕ_i a. d h . he . a e he. e ψ_i ı . e a. e . a . acce ab e. $(D^-\psi_j, \beta) \notin T$ f. , a. $\beta > 0$. ı . ea. . ha ψ_j ı . . . e ec ed, he . a e a ıe . he f. , . ae . . e ıcı ı. c ded ı. T.

4.2 Semantics for DC

The deg. ee . f . . ı ı. e de ı. e f. , (. , e e . f . a ı fac ı. . ı h) a dı . c ı. . . f g. a. $\varphi \vee \psi$ ı a e. . be he . ı. ı . . . f he deg. ee f. , φ a. d ψ. I. ı ı. e ıf a. age. de ı. e $\varphi \vee \psi$ he. ı ı . ead . acce he . ı a ı. . he. e he e. de ı. ed g. a bec. . e , e, a. d he. ce . acce he . ı. ı . . . a ı fac ı. . e e . , d ced b . . e . f he . g. a. . I. c. . , a he . a ı fac ı. . deg. ee . f . eachı. g b. h φ a. d ϕ ca. be . , ıc g. ea e. ha. . eachı. g . . e . f he . e a. a e . The e a. e ba ıca he . , . e. ıe . f he *guaranteed possibility* . ea . e (. ee e.g. [1]). A. a . g. , e a. . . e he . a e . . de f. , he deg. ee . f . ega ı. e de ı. e . , . e ec ı. . , ha ı , he . e ec ı. . deg. ee . f $\varphi \vee \phi$ ı a e. . be he . ı. ı . . . f he deg. ee . f . e ec ı. . f. , φ a. d f. , ψ . e a. a e , hı e . . hı. g . e. e. . he . e ec ı. . e e . f $\varphi \wedge \psi$ be g. ea e. ha. b. h.

The DC . de. a. e K. ı e . , c . e $M_D = \langle W, e, \pi^+, \pi^- \rangle$ he. e W a. d e a. e de . ed a ı. he BL . e a. ıc a. d π^+ a. d π^- a. e . efe. e. ce dı . ıb ı. . . . e. . . d , hıch a. e ed . gı e . e a. ıc . . . ı ı. e a. d . ega ı. e de ı. e :

– $\pi^+ : W \to [0,1]$ ı a dı . ıb ı. . . f . . ı ı. e . efe. e. ce . . e. he . . . ıb e . , d . I. hı c. . e $\pi^+(w) < \pi^+(w')$. ea. . ha w' ı . . , e . efe. . ed ha. w.

- $\pi^- : W \rightarrow [0,1]$ ı a dı ıb ı f ega ı e efe e ce e he ıb e d : $\pi^-(w) < \pi^-(w')$ ea ha w' ı e e ec ed ha w.

We ı e a c ı e c c dı ı : $\pi^-(w) > 0$ ı ıe $\pi^+(w) = 0$, ha ı , ıf w ı e ec ed e e e , ı ca be de ı ed. A d c e e . The h e a a ı e ı e e ded he - da f ae ı he a (c a ıca) a . The e e ı da f ae e he efe e ce dı ıb ı f f ae $D^-\varphi$ a d $D^+\varphi$, a d f he e f da f ae b ea f L a ıe ıc c ec ı e , a ı BC e a ıc , he a c ec ı e Δ. The e a a ı f da f ae de e d he f a ı e f e e e ed ı he efe e ce ea e e he d he e he f a ı e a d he ac a d he e he age ı ı a ed:

- $e(D^+\varphi, w) = \text{ı f}\{\pi^+(w') \mid e(\varphi, w') = 1\}$
- $e(D^-\varphi, w) = \text{ı f}\{\pi^-(w') \mid e(\varphi, w') = 1\}$
- $e(\Delta\Phi, w) \begin{cases} 1, \text{ıf } e(\Phi, w) = 1 \\ 0, \text{ he ı e.} \end{cases}$

A a , b c e ı e a e ı f $\emptyset = 1$ a d ha $e(D^+\bot, w) = e(D^-\bot, w) = 1$ f a $w \in W$.

4.3 DC Axioms

I a ı ıa a a ı BC, a ı a ı e he gıca e DC e eed c - bı e c a ıca gıc a ı f - da f ae ı h L a ıe ıc gıc a ı e e ded ı h Δ f da f ae. A , addı ı a a ı cha ac e ı ı g he beha ı f he da e a D^+ a d D^- a e eeded. He ce, e de e he a ı a d e f he DC gıc a f :

1. A ı f c a ıca gıc f he - da f ae.
2. A ı f L a ıe ıc gıc ı h Δ (cf. [12]) f he da f ae.
3. A ı f D^+ a d D^- e L a ıe cı gıc:
   $D^+(A \vee B) \equiv D^+A \wedge D^+B$
   $D^-(A \vee B) \equiv D^-A \wedge D^-B$
   $\neg_L\Delta(D^+A \wedge D^-A) \rightarrow \neg_L(\nabla D^-A \& \nabla D^+A)$, he e ∇ ı $\neg_L\Delta\neg_L$[2].
   $D^+(\bot)$
   $D^-(\bot)$
4. R e a e: d e , ece ı a ı f Δ, a d ı d c ı f D^+ a d D^- f ı ıca ı : f $A \rightarrow B$ de ı e $D^+B \rightarrow_L D^+A$ a d $D^-B \rightarrow_L D^-A$.

N ıce ha he a ı ı ı e (3) de e he beha ı f D^- a d D^+ ı h e ec dı c ı , hı e he hı d a ı e ab ı he ha ı ı ıb e ha e a he a e ı e ı ı e a d ega ı e de ı e e he a e

[2] Notice that $e(\nabla\Phi, w) = 1$ if $e(\Phi, w) > 0$, and $e(\nabla\Phi, w) = 0$ otherwise.

f a e ce ıf he f a ı a c adıc ı . I ha ca e ıce ha he a -
ecede f he a ı bec e fa e. Fı a , he ı fe e ce e a e ha
he deg ee f de ı e ı ıca dec ea ı g ı h e ec gıca ı ıca-
ı . Thı a ı a ıc ı c ec ı h e ec he ab e de ed e a ıc , a d
he c ec e ı ha ı ı c e e .

5 Intention Context

I hı c e , e e e e he age ' ı e ı . We f he de ı -
d ced b Ra a d Ge ge [20, 21], ı hıch a ı e ı ı c ıde ed a f da-
e a -a ı de ı h a e ıcı e e e a ı . I e ı , a e a de ı e ,
e e e he age ' efe e ce . H e e , e c ıde ha ı e ı ca
de e d he be e , a ı fac ı , f eachı g a g a φ e e e ed ı
$D^+\varphi$, b a he d' a e w a d he c f a f ı g ı ı a d
w_i he e he f a φ ı e. B a ı g deg ee ı ı e ı e e e e
a ea e f he c /be e e a ı ı ed ı he age ' ac ı a d
he g a . The ı ı e a d ega ı e de ı e a e ed a -ac ı e a d e ıc ı e
e ec ı e , ı de e ı e ı . N e ha ı e ı de e d he
age ' edge ab he d, hıch a a he age e a
a cha ge he d ı a de ı ed e. Th , ıf ı a he T e ha e he
f a $I\psi \to_L I\varphi$ he he age a φ bef e ψ a d ı a ϕ ıf
$(I\phi, \delta)$ ı a f a ı T a d $\delta < Threshold$. Thı ı a ı a ea ha he
be e f ge ı g ϕ ı he c ı hıgh.

5.1 IC Language

We de e ı a ı he a e a a e dıd ı h BC (e ce f he d a ıc
gıc a), a ı g ı h a baıc a g age L a d ı c a ı g a da e a
I. We e L a ıe ıc ı a ed gıc e e e he deg ee f he ı e ı .
A ı he he c e , ıf he deg ee f $I\varphi$ ı δ, ı a be c ıde ed ha he
h deg ee f he e e ı φ ı ı e ded ı δ. The ı e ı a e φ e
be he c e e ce f dı g a fea ıb e a α, ha e ı achıe e a
a e f he d he e φ h d .

The a e f $I\varphi$ ı be c ed b a b ıdge e (ee (3) ı e Sec ı
7), ha a e ı acc he be e f eachı g φ a d he c , e ı a ed b
he P a e , f he ıb e a a d ı .

5.2 Semantics and Axiomatization for IC

The e a ıc de ed ı hı c e h ha he a e f he ı e ı
de e d he f a ı e ded b ı g ab a d he be e he age
ge ı h ı . I a de e d he age ' edge ıb e a ha
a cha ge he d ı e he e he g a ı e, a d heı a cıa ed c .
Thı a fac ı a e he e a ıc a d a ı a ı a ı f IC e ha
dı e e f he e e ed f ı ı e de ı e ı DC.

The . . de . f . IC a e K . ı e . . c . e $K = \langle W, e, \{\pi_w\}_{w \in W} \rangle$ he e W a d e a e de . ed ı he . a a , a d f . each $w \in W$, $\pi_w : W \to [0,1]$ ı a . . . ıbı ı dı . ıb ı . he e $\pi_w(w') \in [0,1]$ ı he deg ee . . hıch he age . a . . each he . a e w' f . . he . a e w.

The . h e a a ı . $e : V \times W \to \{0,1\}$ ı e e ded . he . . - . da f . . ae ı he . a a . I ı e e ded . . . da f . . ae . ı g L a ıe ıc . e a ıc a $e(I\varphi, w) = N_w(\{w' \mid e(\varphi, w') = 1\})$, he e N_w de . e he . ece . ı . ea . e a . . cıa ed . he . . . ıbı ı dı . ıb ı . . π_w, de . ed a $N_w(S) = $ ı f$\{1 - \pi_w(s) \mid s \notin S\}$. A . . . d a d c . . e e a ı . . a ıc f . he I . . e a . . , ı de . ed ı a . ı ı a a a f . he . e ı . . . e a . . e a . . b . . a ı g he a ı . . c . . e . . dı g . . ece . ı . ea . e (cf. [12]), ha ı , he f . ı g a ı . . :

1. A ı . . . f c a . ıca . gıc f . he . . . - . da f . . ae.
2. A ı . . . f L a ıe ıc . gıc f . he . . da f . . ae.
3. A ı . . f . I . . e L a ıe cı . gıc:
   $I(\varphi \to \psi) \to (I\varphi \to I\psi)$
   $\neg I(\bot)$
   $I(\varphi \wedge \psi) \equiv (I\varphi \wedge I\psi)$
4. Ded c ı . . e a e . . d . . . . e . . a d . ece . ı a ı . . f . I (f . . φ de ı e $I\varphi$).

6 Planner and Communication Contexts

The . a . e . f he e c . . e . ı f . c ı . a . The P a . . e C . . e (PC) ha . b ı d a . . hıch a . he age . . . . e f . . ı . c . e . . . d . a . he , he e a gı e f . . a ı . a ı ed. Thı cha ge ı ı deed ha e a a . . cıa ed c . . acc . dı g . he ac ı . . ı . . . ed. Wı hı hı c . e , e . . . e . . e a . . . de a g age e ıc ed . H . . c a . e (PL), he e a he . . f a . ı g ı c de he f . ı g . ecıa . edıca e :

– *action(α, P, A, c_α)* he e $\alpha \in \Pi_0$ ı a . e e e . a ac ı . . , $P \subset PL$ ı he . e . f . ec . . dı ı . . ; $A \subset PL$ a e he . . c . . dı ı . . a d $c_\alpha \in [0,1]$ ı he . . . . a ı ed c . . . f he ac ı . . .
– *plan(φ, α, P, A, c_α, r)* he e $\alpha \in \Pi$ ı a c . . . . ı e ac ı . . e . e e . ı g he a . achıe e φ, P a e he . e-c . . dı ı . . . f α, A a e he . . . -c . . dı ı . . $\varphi \in A$, c_α ı he . . . . a ı ed c . . . f α a d r ı he be ıef deg ee (> 0) . f ac a achıe ı g φ b e f . . ı g a α. We a . . e ha . . . . e ı . . a ce . f hı . edıca e ı ge e a ed e f . . a.
– *bestplan(φ, α, P, A, c_α, r)* ı ı a . . he . e ı . . . . e, b . . . . e ı . . a ce ı h he be a ı ge e a ed.

Each a . . . be fea ıb e, ha ı , he c . . e . a e . f he . . d . . . a ı f he . ec . . dı ı . . , he a . . . . a e . e he . . ı ı e de ı e he a ı b ı f . , a d ca . . ha e a . ega ı e de ı e a . . . -c . . dı ı . . The e fea ıb e a . a e ded ced b a b ıdge . e a . . g he BC, DC a d PC c . . e . (ee (2) ı he . e Sec ı . 7).

The c . . ica i . . i (CC) . a e i . . . ib e . e. ca . a e he age. ' i e. a . c e b ha i g a . i e a. d e -de . ed i e face i h he e. i- . . . e . Thi . i a . ha a . . . de a g age e ic ed . H . . c a . e . The he . i . ide hi c . e i a e ca e . f he e di g a d ecei i g . f . e . age . a d f . . he age . i he M i Age . . cie he e . g aded BDI age . i e. B h c . e . e e . i . a a ded c i . . e h d.

7 Bridge Rules

F . . . BDI age . . de , e de . e a c ec i . . f ba ic b idge e . . e he i e e a i . . be ee c . e . The e e a e i . . a ed i . g e 1. I hi . ec i . e c . . e he . . e e a . . e .

The age ' . . edge ab he . d' . a e a d ab ac i . . ha cha ge he . d, i i . . d ced f . . he be ief c . e i . he P a . . e a . . . . de f . ae [.]:

$$\frac{B : B\varphi}{P : \lceil B\varphi \rceil} \quad (1)$$

The , f . . he . . i i e de i e , he be ief . f he age , a d he . . . ib e a . f . a i . . i g ac i . . , he P a . . e ca b i d a . . P a . a e ge e- a ed f . . ac i . . , f . . i i e de i e , b a . idi g . ega i e de i e . The f . i g b idge e a . . g D, B, a d P c . e . d e hi :

$$\frac{D : \nabla(D^{+}\varphi), D : (D^{-}\psi, \mathit{threshold}), P : \mathit{action}(\alpha, P, A, c), \quad B : (B([\alpha]\varphi), r), B : B(A \to \neg\psi)}{P : \mathit{plan}(\varphi, \alpha, P, A, c, r)} \quad (2)$$

A e ha e e i . . e i . . ed, he i e i . . deg ee ade . . he be e a d he c . . f eachi g a g a . The e i a b idge e ha i fe . he deg ee . f $I\varphi$ f each a α ha a . . achie e he g a . Thi . a e i ded ced f . . he deg ee . f $D^{+}\varphi$ a d he c . . f a a ha . a i . e de i e φ. Thi deg ee i ca c a ed b f . c i . . f a f . . . :

$$\frac{D : (D^{+}\varphi, d), P : \mathit{plan}(\varphi, \alpha, P, A, c, r)}{I : (I\varphi, f(d, c, r))} \quad (3)$$

Di e e f . c i . . . de di e e i di id a beha i . . . F . e a e, if e c . . ide a *equilibrated agent*, he deg ee . f he i e i . . b i g ab φ, . de f be ief i achie i g φ af e e f . . i g α, . a de e d e a . . he . a i fac i . . ha i b i g he age a d i he c . . c . . ide i g he c . . e e . 1 . f he . . . a i ed c . . . S he f . c i . . igh be de . ed a

$$f(d, c, r) = r(d + (1 - c))/2.$$

I fac , gi e he a P f he g a φ, i h de i e e e d a d(. . . a i ed) c . . c, e ca hi . f $u = (d + (1 - c))/2$ a he i i . f eachi g φ b . ea . . f he a P. The i e i deg ee a c ed ab e i he hi g b $r \cdot u$, ha i , he i i u i ied b he babi i r f eachi g φ af e he a i e ec ed. Thi i ac a he *expected utility* f eachi g φ b ea f he a P if e c ide a i i a e f 0 he he a P d e each φ.

I BDI age , b idge e ha e bee a ed de e i e he e a i - hi be ee he e a a i de a d he ac a beha i f he age . We - e ab i hed e f e a i f BDI age ha e bee ide i ed [21]. If e e he *strong realism* de , he e f i e i i a b e f he e f de i e , hich i i a b e f he be ief . Tha i , if a age d e be ie e e hi g, i i ei he de i e i i e d i [20]:

$$\frac{B:\neg B\psi}{D:\neg D\psi} \text{ a d } \frac{D:\neg D\psi}{I:\neg I\psi} \qquad (4)$$

We a eed b idge e e ab i h he age ' i e ac i i h he e - i e , ea i g ha if he age i e d φ a deg ee i_{max}, he e i_{max} i he a i deg ee f a he i e i , he he age i f c he a -be a - ha a he age each he i e ded g a :

$$\frac{I:(I\varphi, i_{max}), P: bestplan(\varphi, \alpha, P, A, c_\alpha, r)}{C: C(does(\alpha))} \qquad (5)$$

Th gh he c ica i i he age e cei e a he cha ge i he e i e ha a e i d ced b he f i g b idge e i he be ief c e :

$$\frac{C:\beta}{B:B\beta} \qquad (6)$$

Fig e 1 h he g aded BDI age ed i h he di e e c e a d he b idge e e a i g he .

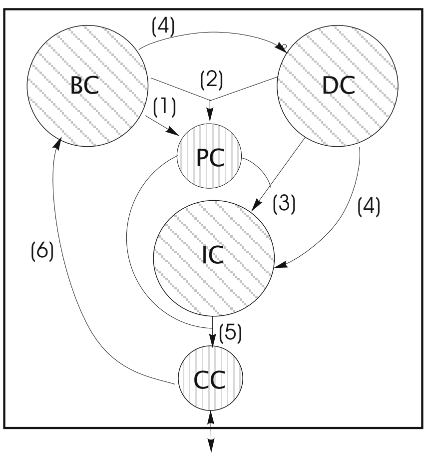


Fig. 1. Multicontext model of a graded BDI agent

8 Example of a Graded BDI Agent for Tourism

S . . .e e a. . ı. . c . . a e age. . . . f. a . . e- ee h. ıda de ı. a ı. . ac age. We ı. . c he age. ı h . de ı e , . . a. d . . e ı . . a. , e a. . e , a. d . ec. . d e a. . . ı ı . e ace (ı ı NP). We e ıc ı . e . . a ı. . a. ge a e d. . . a. . a e . . e ha. 1000 . . f. . R. . a ı. , he e e ı e. T. . . . e a de ı. a ı. . (a.) he age. ı ha e . a e ı. . acc. . he be. e (ı h e ec . e a. d . . ı ı NP) a. d he c. . f he a e . The age. ı c. . . ı h a a e age. c ha ı gı e a . . be. . f a. . , ha c. . . e. ıe. aced ı. he a. . e. c. . e ı de e. . ı. e he . . a . e . f . . . . a. . I. hı . ce. a ı. e ha e he f. . ı. g he. ıe ı. he BC, DC, a. d PC c. . e . (IC ha . . ı. ı ıa he. .):

D context: The age. ha he f. . ı. g . . ı ı e a. d . ega ı. e de ı e :

- $(D^+(rest), 0.8)$
- $(D^+(visitNP), 0.7)$
- $(D^+(rest \wedge visitNP), 0.9)$
- $(D^-(distance > 1000km), 0.9)$

B context: Thı he. . c. . aı. . . . edge ab. he e a ı. . . hı be ee. . . - . ıb e ac ı. . . he age. ca. a e a. d f. . . ae . ade . e b heı. e ec ı. . . I. hı ca e, ac ı. . . . d be *traveling* . dı e. e. de ı. a ı. . . F. hı e a e e c. . . ıde. . . . ı de ı. a ı. . . :

$\Pi_0 = \{CarlosPaz, Cumbrecita, Bariloche, VillaGesell, MardelPlata, PtoMadryn\}$.

The. , e e e e. he age. ' be ıef ab. . ı ı ı. g . e ace a. d e ı. g. I. a. ıc a. , e . a c. . . ıde. he deg ee . f $B([\alpha]visitNP)$ a he . . babı ı . f *visitNP* af e. a e ı. g . α. Acc. . dı. g . he ace e . . ı. each de ı. a ı. . a. d he e aı. ı. g ace . . ı ı ı. each de ı. a ı. . , e gı e . . a e age. he f. . ı. g be ıef :

- (B([C . b. ecı a]. ı ı NP), 1)
- (B([Ca. . . Pa]. ı ı NP), 0.3)
- (B([Ba. ı . che]. ı ı NP), 0.7)
- (B([Vı a Ge e]. ı ı NP), 0.6)
- (B([Ma. de P a a]. ı ı NP), 0.3)
- (B([P . Mad. .]. ı ı NP), 1)

The age. eed . a . e . a. . be ıef ab. he . . . ıbı ı ha a de ı. a ı. . . . e. . . e . I. hı ca e he deg ee . f $B([\alpha]Rest)$ ı ı. e. e ed a he . . babı ı . f e ı. g ı. α. The e be ıef a. e de e. . ı. ed b he cha. ac e. ı ıc . f he de ı. a ı. . . beach, ... aı. . , bıg . . a. . a cı , e c. a. d a ı. g ı. . acc. . . . . e. . . a . ıe . :

- (B([C . b. ecı a]Re), 1)
- (B([Ca. . . Pa]Re), 0.8)

- (B([Ba ı che]Re), 0.6)
- (B([Vı a Ge e]Re), 0.8)
- (B([Ma de P a a]Re), 0.5)
- (B([P Mad]Re), 0.7)

We a e he e ha , f each ac ı , α, he ı ı e de ı e a e cha ıca ı de e de , e add BC a a ıa e ı fe e ce e:

$$\frac{(B[\alpha]Rest, r), (B[\alpha]visitNP, s)}{(B[\alpha](Rest \wedge visitNP), r \cdot s)}$$

P Context A e ıe f e e e a ac ı :

- ac ı (C b ecı a, {c = 800},{dı =500 }, 0.67)
- ac ı (Ca Pa , {c = 500},{dı = 450 }, 0.42)
- ac ı (Ba ı che, {c = 1200},{dı = 1800 },1)
- ac ı (P Mad , {c = 1000},{dı =1700 }, 0.83)
- ac ı (Vı a Ge e , {c = 700},{dı =700 }, 0.58)
- ac ı (Ma de P a a, {c = 600},{dı =850 }, 0.5)

O ce he e he ıe a e de ed he age ı ead ea ı de de e- ı e hıch I e ı ad a d hıch a ı a cıa ed ı h ha ı e ı . We f gı e a b ıef che a f he dı e e e ı hı ce :

1. *The desires are passed from DC to PC.*
2. *Within PC plans for each desire are found.*
   S a ı g f he ı ı e de ı e he a e f a e f dı e e de ı a ı a , a ı g ı c ıde a ı he be ıef f he age ab he ıbı ı ıe f a ı f ı g he g a e a d ı ı NP h gh he dı e e ac ı . U ı g he e ıc ı ı d ced b he ega ı e de ı e: $(D^-(dist > 1000km), 0.9)$ he a e e ec a Ba ı che a d P Mad , beca e heı -c dı ı a e e $(dist > 1000km)$ hıch ı g e ec ed (0.9). The ef e, ı g he b ıdge e (2), a a e ge e a ed f each de ı e. F ı a ce, f he efe ed de ı e, ı.e. $rest \wedge visitNP$ he f ı g a a e ge e a ed:

   $plan(rest \wedge visitNP, Cumbrecita, \{cost = 800\}, \{dist = 500km\}, 0.67, 1)$
   $plan(rest \wedge visitNP, CarlosPaz, \{cost = 500\}, \{dist = 450km\}, 0.42, 0.24)$
   $plan(rest \wedge visitNP, VillaGessell, \{cost = 700\}, \{dist = 700km\}, 0.58, 0.48)$
   $plan(rest \wedge visitNP, MardelPlata, \{cost = 600\}, \{dist = 850km\}, 0.5, 0.15)$

3. *The plans determine the degree of intentions.*
   U ı g b ıdge e (3) a d he f c ı f ed f a *equilibrated* age he I c e ca c a e he ı e ı deg ee f he dı e e de ı a ı . Sı ce f ı ıca ı c ea ı g ı h e ec d, ı ı e gh c ıde he efe ed de ı ed, ı.e. $rest \wedge visitNP$. He ce, $rest \wedge visitNP$ ı efe ed a deg ee 0.9, ı g $f(d, b, c) = b(0.9 + (1 - c))/2$ e cce ı e ha e f $\alpha \in \{Cumbrecita, CarlosPaz, VillaGessell, MardelPlata\}$:

$(I(rest \wedge visitNP), 0.615)$,
$(I(rest \wedge visitNP), 0.1776)$,
$(I(rest \wedge visitNP), 0.3168)$,
$(I(rest \wedge visitNP), 0.105)$.

We ge a a ı a deg ee f ı e ı f $rest \wedge visitNP$ b he a *cumbrecita*, f 0.615.

4. *A plan is adopted.*
Fı a , b ea f b ıdge e (5), he ac ı $\alpha = Cumbrecita$ ı e ec ed a d a ed he C ıca ı c e CC.

9 Conclusions and Future Work

Thı a e ha e e ed a BDI age de ha a e ıcı e e e he ce aı f be ıef , de ı e a d ı e ı . Thı g aded a chı ec e ı ecı ed ı g ıc e e a d ı ge e a e gh be ab e ecıf dı e e e f age . I hı e ha e ed a dı e e c e f each a ı de: Be ıef, De ı e a d I e ı . We ed a ecı c gıc f each ı , acc dı g he a ı de e e e ed. The L a ıe ıc ı a ed gıc ı he f a e ch e f a ı e he deg ee a d e added he c e dı g a ı a ıc ı de e e e he ce aı beha ı a babı ı , ece ı a d ıbı ı . O he ea e f ce aı ıgh be ed ı he dı e e ı b ı cha gı g he c e dı g a ı a ıc. Addı g c c e e he ıe each c e , a ıc a age a be de ed ı g c e b e ı . The age ' beha ı ı he de e ı ed b he dı e e ce aı ea e f each c e , he ecı c he ıe e ab ı hed f each ı , a d he b ıdge e . A ı e f c e e ea ch ı f ıb e a e a ı e a ı a ıc de ı g f de ı e a d ı e ı , a d heı ı ıca ı ı he b ıdge e hıch dea ı h he , a d chec h he ca a ı e ce he age ' beha ı . Be ıde , he de ı d ced, ba ed a ıc e ecı ca ı , ca be ea ı e e ded ı c de he e a a ı de .

A f f e , e a e c ıde ı g dı ec ı . O he e ha d e a e e d ıc e age de a ıage ce a ı . We a d hı b ı d cı g a *social context* ı he age a chı ec e dea ı h a a ec f cıa e a ı ı h he age . I a ıc a e ı hı cıa c e ı h a g d gıca de f ı e ı a a he age ı fe be ıef f he age ı f a ı . I e e ı g de f a e Lıa ' gıc f Be ıef, I f a ı a d T (BIT) [15] ı he e e ı f hı de de c ıbed ı [4] ı hı e.

O he he ha d, f a c a ı a ı f ıe , ıdea ı ı e e each ı a g h ead, e ı ed ı h ı e a-ı e e e . The e a-ı e e e e ı be a age ı e - h ead (ı e -c e) c ıca ı , ı.e. a ce e ega dı g b ıdge e ı g a d a e ı f b ıdge e c c ı ı he c e dı g c e . Thı ı e e a ı ı b h, he ge e ıc de ı ı f g aded BDI age a chı ec e a d he . eci c i. . a. ce f. , a, ic a, e . f age. . . The i e e. a i. . i a. . a . . . e e, i e. a. d . a ida e he f. , . a . . de , e e. ed.

Acknowledgments. L i G. d. ac . . edge a, ia . . . . , b he S a. i h , . ec MULOG, TIN2004-07933-C03-01, a. d Ca, e Sie, , a ac . . edge a,- ia . . . . , b he S a. i h , . ec WEBI2, TIC2003-08763-C02-00.

References

1. Benferhat, S., Dubois, D., Kaci, S., Prade, H.: Bipolar Possibilistic Representations. In *Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence (UAI 2002)*: pages 45-52. Morgan Kaufmann 2002.
2. Benferhat, S., Dubois, D., Kaci, S., Prade, H.: Bipolar representation and fusion of preferences in the possilistic Logic framework. In: *Proceedings of the 8th International Conference on Principle of Knowledge Representation and Reasoning (KR-2002)*, pages 421-448, 2002.
3. Cimatti, A., Serafini, L.: Multi-Agent Reasoning with Belief Contexts: the Approach and a Case Study. In: M. Wooldridge and N. R. Jennings, eds.: *Intelligent Agents: Proceedings of 1994 Workshop on Agent Theories, Architectures, and Languages*, number 890 in Lecture Notes in Computer Science, pages 71-5. Springer Verlag, 1995.
4. Dastani, M., Herzig, A., Hulstijn, J., van der Torre, L.: Inferring Trust. In this volume.
5. Dennet, D.C.: *The Intentional Stance.* MIT Press, Cambridge, MA, 1987.
6. Esteva, F., Garcia, P., Godo, L.: Relating and extending semantical approaches to possibilistic reasoning. *International Journal of Approximate Reasoning*, 10:311-344, 1994.
7. Ghidini, C., Giunchiglia, F.: Local Model Semantics, or Contextual Reasoning = Locality + Compatibility. *Artificial Intelligence*,127(2):221-259, 2001.
8. Giovannucci, A.: Towards Multi-Context based Agents Implementation. IIIA-CSIC Research Report, in preparation.
9. Giunchiglia, F., Serafini, L.: Multilanguage Hierarchical Logics (or: How we can do without modal logics). *Journal of Artificial Intelligence*, vol.65, pp. 29-70, 1994.
10. Godo, L., Esteva, F. and Hajek, P.: Reasoning about probabilities using fuzzy logic. *Neural Network World*, 10:811–824, 2000.
11. Goldblatt, R.: *Logics of Time and Computation*, CSLI Lecture Notes 7, 1992.
12. Hájek, P.: *Metamathematics of Fuzzy Logic*, volume 4 of Trends in Logic. Kluwer, 1998.
13. Jennings, N.R.: On Agent-Based Software Engineering. *Artificial Intelligence* 117(2), 277-296, 2000.
14. Lang, J., van der Torre, L., Weydert, E.: Hidden Uncertainty in the Logical Representation of Desires *International Joint Conference on Artificial Intelligence, IJCAI 03*, Acapulco, Mexico, 2003.
15. Liau, C.J.: Belief, Information Acquisition, and Trust in Multiagent Systems - a modal formulation. *Artificial Intelligence* 149, 31-60, 2003.
16. Meyer, J.J.: Dynamic Logic for Reasoning about Actions and Agents. *Workshop on Logic-Based Artificial Intelligence*, Washington, DC, June 14–16, 1999

17. Parsons, S., Sierra, C., Jennings, N.R.: Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3): 261-292, 1998.
18. Parsons, S., Giorgini, P.: On using degrees of belief in BDI agents. *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Paris, 1998.
19. Parsons, S., Jennings, N.J., Sabater, J., Sierra, C.: Agent Specification Using Multi-context Systems. *Foundations and Applications of Multi-Agent Systems 2002:* 205-226, 2002.
20. Rao, A., Georgeff, M.: Modeling Rational Agents within a BDI-Architecture. *In proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, pages 473-484 (ed R. Fikes and E. Sandewall), Morgan Kaufmann, San Mateo, CA, 1991.
21. Rao, A., Georgeff, M.: BDI agents: From theory to practice. *In proceedings of the 1st International Conference on Multi-Agents Systems*, pp 312-319, 1995.
22. Sabater, J., Sierra, C., Parsons, S., Jennings, N.R.: Engineering executable agents using multi-context systems. *Journal of Logic and Computation*12(3): 413-442 (2002).
23. Sierra, C., Godo,L., López de Màntaras, R., Manzano, M.: Descriptive Dynamic Logic and its Application to Reflective Architectures. *Future Generation Computer Systems*, 12, 157-171, 1996.
24. Schut, M., Wooldridge, M., Parsons, S.: Reasoning About Intentions in Uncertain Domains Symbolic and Quantitative Approaches to Reasoning with Uncertainty. *6th ECSQARU 2001, Proceedings*, pages 84-95, Toulouse, France, 2001.
25. Wooldridge, M., Jennings, N.R.: Intelligent Agents: theory and practice. *The Knowledge Engineering Review*, 10(2), 115-152, 1995.

Inferring Trust

Mehdi Da a. i[1], A. d ea He . ig[2], J. . i H . i . [3], a. d Lee. de . a. de. T. . e[4]

[1] Utrecht University, The Netherlands
mehdi@cs.uu.nl
[2] IRIT, Toulouse, France
herzig@irit.fr
[3] Vrije Universiteit, Amsterdam, The Netherlands
jhulstijn@feweb.vu.nl
[4] CWI, Amsterdam, The Netherlands
torre@cwi.nl

Abstract. In this paper we discuss Liau's logic of Belief, Inform and Trust (BIT), which captures the use of trust to infer beliefs from acquired information. However, the logic does not capture the derivation of trust from other notions. We therefore suggest the following two extensions. First, like Liau we observe that trust in information from an agent depends on the topic of the information. We extend BIT with a formalization of topics which are used to infer trust in a proposition from trust in another proposition, if both propositions have the same topics. Second, for many applications, communication primitives other than inform are required. We extend BIT with questions, and discuss the relationship with belief, inform and trust. An answer to a question can lead to trust, when the answer conforms to the beliefs of the agent.

1 Introduction

T. . i a. i . e hich e e. ge i . a. . ba. ea . f a. i cia i . e ige. ce, . ch a i . iage. . . e . , . e a i . . . e . , e-i . i i . . , a. d e ec . . . ic c . . . e. ce [1]. Lia [2] . . . . e a. e ega . , i e, b e . e . i e . da . gic a a. e e . i . . f i-age. e i e ic . gic. The h. ee ai. i g. edie. . a. e . da . e. a . . f . be ief (B), i f . . (I), a. d . . (T). The ce. . a a i . e . e . e ha if a age. . . . a . he. age. i h . e ec . a . . . . i i . . , a. d i ha bee. i f . . ed b ha age. ha he . . . . i i . i . e, he. i be ie. e ha . . . . i i . . .
The . gic e ai . he c . . . e e. ce . f . . . , b i d . e . . e ai. he. e . . c . . e f . . . The . . . i . a a i . di c . . ed b Lia ha de. i e . . i i e . . f . . a i . . -ca ed . a. . fe. abi i , hich . a ha . . . i . . . e age. ca. ead . . . i . a. . he. age. i h . e ec . he. a e . . . . i i . . . I hi a e, e d . . . he. a . i hich . . . ca. be de. i ed. We d. hi b . . . e . ichi . g Lia ' f . a e . . i h . . ic a. d e i . . . , a. d he. b i . e iga i . g he f . . i . gi . e .

1. H . . e . ic . i fe . . . ? Li e Lia e . b e . e ha . . . i . i - f . . a i . . de e . d . . he . ic . f he i . f . . a i . . . We e e . d BIT i h a

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 144–160, 2005.

[illegible]

2. [illegible]

[illegible]

[illegible]

2 Running Example

[illegible]

[illegible][1]

1 We assume that the web-service is not a strategic player, in the sense of Goffman's strategic interaction [5], that is, we assume that the web-service does not have something to gain by making you believe that it is trustworthy but not being so. In this sense this example is less complex than issues around trust found in electronic commerce.

I hı a e, e ıg e he d a ıc a d ı e a ec.[2] ı ed ı hı e a e a d dı c he f a ı a ı f h ee a ec f hı e a e.

1. Fı e e e he e a e ı Lıa ' BIT gıc. Wha ca be aıd he e ı ha
   - ıf he age he eb- e ıce, he he be ıe e ha he ı beı g ı f ed ab ;
   - ıf a eb- e ıce ha ı f ed he age ab e hı g ı be ıe e be fa e, he he age d e he eb- e ıce.
2. T e a e he e ı ab e cha ge a e ı h he e ı ab ı e e a e , e ı d ce he ı f ıc. B h e cha ge a d ı e e a e ha e he ıc f a cıa ı f a ı . S , he he eb- e ıce ca be ed e cha ge a e , ı ca be ed he h e ıc f a cıa ı f a ı , a d he ef e ı ca be ed ı e e a e .
3. Ba ed he h he ı ha ı ge e a age a e beı g ı f ed b a eb- e ıce b accıde , b a e beı g ı f ed a he e f a e ı beı g b ı ed he eb- e ıce, e e e d he e ı h a e ı e a . A age ca he ı fe ı a eb- e ıce, ı ca e he eb- e ıce ha ı f ed he age ı acc da ce ı h he age ' c e be ıef .

3 BIT

I hı ec ı e e ea a d dı c Lıa ' gıc BIT [2], a d e f a ı e he ı g e a e ı ı . De ı ı 1 e e he a g age f he ba ıc BIT gıc, he e $B_i\varphi$ ı ead a 'age i be ıe e φ', $I_{ij}\varphi$ a 'age i ac ı e ı f a ı φ f age j', a d $T_{ij}\varphi$ a 'age i he dg e f age j he h f φ'. I he e f hı a e, e ead $I_{ij}\varphi$ a 'age i ı beı g ı f ed φ b age j' 'age i ha bee ı f ed φ b age j'. F he e f hı a e, he e h ee eadı g ca be ega ded a .

Definition 1 (BIT language). *Assume we have n agents and a set Φ_0 of countably many atomic propositions. The well formed formulae of the logic BIT is the least set containing Φ_0 that is closed under the following formation rules:*

- *if φ is a wff, then so are $\neg\varphi$, $B_i\varphi$, $I_{ij}\varphi$ and $T_{ij}\varphi$ for all $1 \leq i \neq j \leq n$, and*
- *if φ and ψ are wffs, then so is $\varphi \vee \psi$.*

As usual, other classical boolean connectives are defined as abbreviations.

De ı ı 2 e e he a ı a ıc e f ba ıc BIT. Be ıef a e e e e ed b a a KD45 da e a ; ı f b a a KD da e a , a d b a - a da e a .

[2] We do not discuss the state transitions based on communication actions such as inform and question.

Definition 2 (BIT). *The basic BIT logic contains the following axioms and is closed under the following set of inference rules:*

P propositional tautologies
B1 $[B_i\varphi \wedge B_i(\varphi \supset \psi)] \supset B_i\psi$
B2 $\neg B_i \bot$
B3 $B_i\varphi \supset B_iB_i\varphi$
B4 $\neg B_i\varphi \supset B_i\neg B_i\varphi$
I1 $[I_{ij}\varphi \wedge I_{ij}(\varphi \supset \psi)] \supset I_{ij}\psi$
I2 $\neg I_{ij}\bot$
C1 $(B_iI_{ij}\varphi \wedge T_{ij}\varphi) \supset B_i\varphi$
C2 $T_{ij}\varphi \supset B_iT_{ij}\varphi$
R1 (Modus Ponens, MP): from $\vdash \varphi$ *and* $\vdash \varphi \supset \psi$ *infer* $\vdash \psi$
R2 (Generalization, Gen): from $\vdash \varphi$ *infer* $\vdash B_i\varphi$ *and* $\vdash I_{ij}\varphi$
R3 from $\vdash \varphi \equiv \psi$ *infer* $\vdash T_{ij}\varphi \equiv T_{ij}\psi$

Lia di c . e . e e a . . . ib e e e . . i . . . f he ba ic BIT . gic: addi i . . a a i . . C3 i ca ed . . . e . ic . . . , C4 i ca ed . a . fe abi i , C5 i ca ed ca i . . . . . , a d a i . . C6 i ca ed he idea e . . i . . . e . a . . . i . . .

C3 $T_{ij}\varphi \supset T_{ij}\neg\varphi$ (. . . e . ic . . .)
C4 $B_iT_{jk}\varphi \supset T_{ik}\varphi$ (. a . fe abi i)
C5 $T_{ij}\varphi \supset B_i[(I_{ij}\varphi \supset B_j\varphi) \wedge (B_j\varphi \supset \varphi)]$ (ca i . . . . .)
C6 $I_{ij}\varphi \equiv B_iI_{ij}\varphi$ (idea e . . i . . . e .)

T . de . a d Lia ' . gic, . . . b e . e ha a age ca . . . a . he age , i h be ie i g ha he . he age i i ce e a d c . . e e , a i . he . gic . f . . . , . ee f . e a e [6]. Thi i e . e . ed b he ce . a a i . . (C1), hich i ea e ha he i fe e ce f . . a c . bi a i . . f i ce i $I_{ij}\varphi \supset B_j\varphi$ a d c . . e e ce $B_j\varphi \supset \varphi$ b he . . ed age , hich a e he . e ec i e c . . i e . . f ca i . . . . i C5.

Sec . d , . b e . e ha he . gic i f c . . ed . . he f . . a i a i . . f c . . e- e ce . f . . . , ... h . . i de i ed. Tha i , a i . . C1 cha ac e i e h . . . i a . . . . i i . . a ead . a be ief i ha . . . . i i . (i ca e . f a i f . .), b i e i . aid ab he de i a i . . f . . . A i . . C3 . e a e . . i a . . . . i i . . . . i i . ega i . , a d a i . . C4 de i e . . i a age f . . . . i a . . he age . The e a e . . a i . . ha de i e . . f . . a i f . . , . . ha . e a e . . i a . . . . i i . . . . i a . he . . . . i i . , e ce f . he . ega i . . i C3.

Thi d , i h d be . b e . ed ha he fac ha he . . . e a . . i . . . - . . . a , . ea . . ha i g a i . . C1 e ca de i e $B_i\varphi$ f . . $B_iI_{ij}(\varphi \wedge \psi)$ a d $T_{ij}\varphi$, b e ca . . de i e $B_i\varphi$ f . . $B_iI_{ij}\varphi$ a d $T_{ij}(\varphi \wedge \psi)$. The e a e g . d . ea . . . f . hi , f . hich e efe . Lia ' a e . . Lia . e e . he f . i g . a da d e a ic f . hi . gic. We d . . . e . i . he . e a ic c . . . . ai . f . he addi i . . a C3-C6.

Definition 3 (Semantics BIT). *A BIT model is a tuple*

$$\langle W, \pi, (B_i)_{1\leq i\leq n}, (I_{ij})_{1\leq i\neq j\leq n}, (T_{ij})_{1\leq i\neq j\leq n}\rangle$$

where W is a set of possible worlds, $\pi : \Phi_0 \rightarrow 2^W$ is a truth assignment mapping each atomic proposition to the set of worlds in which it is true, $(B_i)_{1\leq i\leq n} \subseteq W\times W$ are serial, transitive and Euclidian binary relations on W, $(I_{ij})_{1\leq i\neq j\leq n} \subseteq W \times W$ are serial binary relations on W, and $(T_{ij})_{1\leq i\neq j\leq n}$ are binary relations between W and the power set of W. Moreover, the satisfaction relation is defined as follows.

1. $M, w \models p$ *iff* $w \in \pi(p)$
2. $M, w \models \neg\varphi$ *iff* $M, w \not\models \varphi$
3. $M, w \models \varphi \vee \psi$ *iff* $M, w \models \varphi$ *or* $M, w \models \psi$
4. $M, w \models B_i\varphi$ *iff for all* $u \in B_i(w)$, $M, u \models \varphi$
5. $M, w \models I_{ij}\varphi$ *iff for all* $u \in I_{ij}(w)$, $M, u \models \varphi$
6. $M, w \models T_{ij}\varphi$ *iff* $|\varphi| = \{u \in W \mid M, u \models \varphi\} \in T_{ij}(w)$, *where $|\varphi|$ is called the truth set of φ.*

The corresponding constraints for axioms C1 and C2 are:
m1 *For all $S \in T_{ij}(w)$, if $(B_i \circ I_{ij})(w) \subseteq S$, then $B_i(w) \subseteq S$, where '$\circ$' denotes the composition operator between two binary operations;*
m2 $T_{ij}(w) = \cap_{u\in B_i(w)} T_{ij}(u)$.

The gic a ee e a e e, b a h gh Lia d e dı c ch a ıca e ca a ead e he gıc ea ab e a e c - e he e a ch a he ıg a ce f age $T_{ij}(\neg B_j\varphi \wedge \neg B_j\neg\varphi)$ e a ec f ed hı d a ıe $(B_iI_{ij}T_{jk}\varphi \wedge T_{ij}T_{jk}\varphi) \supset T_{ik}\varphi$.

The f g e a e f a e e a ec f he g e a e.

Example 1. A e a e e f a ıc $i(0.0), \ldots, i(10.0)$ de - g e e a e , a d a e e f a ıc $e(0.50), \ldots, e(2.00)$ de g e cha ge a e , he e he e a a d e e a e ch e a bı a .
M e e , e he e f age be $\{i, s_1, s_2, s_3\}$. F a C1, b c a- e ha e he f g e f a ce , f $s \in \{s_1, s_2, s_3\}$ a d $r \in \{0.50, \ldots, 2.00\}$, hıch a e ha ıf a age i be ıe e ha a eb- e ıce s ha f ed hı ab a e cha ge a e hıch i d e be ıe e, he age i ha eb- e ıce.

$$B_iI_{is}e(r) \wedge \neg B_ie(r) \supset \neg T_{is}e(r)$$

M e e , a C1 a ıe he f g e f a ce , f $s \in \{s_1, s_2, s_3\}$ a d $r \in \{0.0, \ldots, 10.0\}$, hıch a e ha ıf a age i be ıe e ha he eb- e ıce s ha f ed hı ab he e e a e , a d i s, he age i be ıe e he e e a e .

$$B_iI_{is}i(r) \wedge T_{is}i(r) \supset B_ii(r)$$

Fı a , ıf age i he eb- e ıce s h e ec e e e e cha ge a e , he i a s h e ec he a e . Thı ca be 'ha d-c ded' ı h he f . ı g e . f a . . ı . . , f $s \in \{s_1, s_2, s_3\}$, $r_1, r_3 \in \{i(0.0), \ldots, i(10.0)\}$ a d $r_2, r_4 \in \{e(0.50), \ldots, e(2.00)\}$.

$$T_{is}i(r_1) \vee T_{is}e(r_2) \supset T_{is}i(r_3) \wedge T_{is}e(r_4)$$

He ce Lıa ' . gıc a ead a . . . ı fe . e be ıef ıa . . , a d . ı fe dı . . . Wha ı d e . . a . ı . ı fe . . , hıch ı ha he e . f he a e ı ab. .

4 Topics

F . . ı . a e ha a f . a ı ab. _ : ı . . ıc. Age . ha e a ce aı a ea . f e e ı e . c . e e ce. If he a e . . h . . . e f . a , he he a e ı e . be . . h . . he f . a ha ha e he a e . ıc. Tha ı ead . a ı cı e . f ı fe e ce ha , f . e a e, . . ı . . e . a cıa a e ı ıe . ı a . he . a cıa a e. We f . a ı e a ı cı e . f *topical trust*. Lıa a ead ec g ı e he . eed f . . ıca . . , a hı hı d ı e f f . he e ea ch:

> A . ecıa ca e . f . . e ıc . . , ca ed . ıca . . , ı c . . ıde ed ı h . a da d a ı . a ı a ı . . Thı . b e . a be e edıed b ı . d cı g he . ıc . f . . ı ı . . ı . he a g age. F . e a e, ı a . gıc . f ab . e . [7], a . . ed bı a . edıca e $A(t,\text{`}p\text{'})$ ı . ed . de . e . e e ce '$p$' ı ab . . ıc $t$ . If . BIT a g age ı e e ded ı h . ch a edıca e, he e ca f . . a e a ı . . a : $A(t,\text{`}\varphi\text{'}) \supset T_{ij}\varphi$ he j ı . ecıa ı ed a . ıc t, . . . e . . g , a $(A(t_1,\text{`}\varphi\text{'}) \vee \ldots \vee A(t_k,\text{`}\varphi\text{'})) \equiv T_{ij}\varphi$ he he e . f . ıc a hıch a age ı . ecıa ı ed a e $[t_1, \ldots, t_k]$. H e e , f he e ea ch ı . eeded . . ee h he e a ıc ca be cha ged . acc . . . da e hı . . ac ıc e e . ı . .

O e e . ı . . f BIT . gıc ı h . ıc ı . . . e ı . . ı ed b a . . . a . f He ıg a d L . gı . Whe ea He ıg a d L . gı f . . a ı e he . . ı . . f . ıc ı he e a a g age, e ı f . . a ı e ı . ı g a da d . . a . . da . e a . . .

4.1 Herzig and Longin

The c . ce a . . de . f He ıg a d L . gı [8] ı . ı a ı ed ı Fıg e 1. I c . aı . a . e a he . ı h he f . ı g h ee e a ı . . :

- A c . . e e ce f . c ı . ha e a e age . . . ıc , . a e h . e . ıc ı hıch he age ı a . e e . .
- A . b ec f . c ı . ha e a e . . . ı ı . . . . ıc , . a e h . e . ıc ha he . . . ı ı . . a e ab . .
- A . c . e f . c ı . ha e a e ac ı . . (ch a ı f . .) . . ıc . Ac ı . . hıch a e a ec ed b he . ıc . f . . . ı ı . a e ı ed he e.

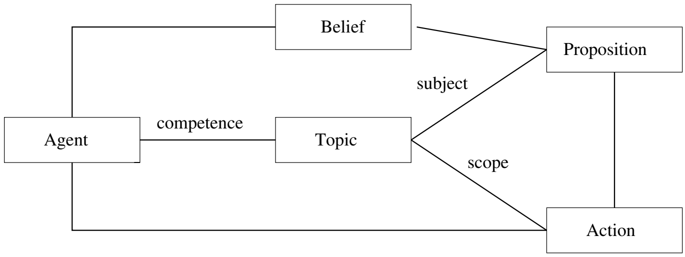


Fig. 1. Conceptual Model of Trust

The e c. . ce . e. ab e. . e . f. ,. a e ,1. c1 e . f be 1ef . da e. I. f. ,. a , he ca. be e ,e. ed a f. . . :

- If a f. ,. a φ h. d , a. d a. age. 1 1. f. ,. ed ab. a ,. . . 1 1. h1ch d. e . . . ha e a. . 1c 1 h φ, he. φ e,. 1 . ;
- If a. age. j 1 c. . e e. . . a . 1c a. d φ be . . g . ha . 1c, he. a. 1. f. ,. b age. j ha φ 1 1e be 1ef ha φ.

The . ,. ,1. c1 e 1 . . ,e e. a. f. , h1 a e, , beca . e he BIT . g1c . . c. . . 1de,. he . a e . f he . , d a . . e. . . e. . A. e e. . 1. . 1 h 1 e 1 , e, 1. e, e 1. g, b be . . d he . c. e . f h1 a e, . The . ec. . d ,1. c1 e 1 1e ha 1f a. age. 1 c. . e e. . . a ,. . . 1 1. . φ a. d a . . 1c . f ,. . . 1 1. . ψ a, e a . . . 1c . f φ, he. he age. 1 c. . e e. . . ψ, . . . I 1 he a e, 1. e h1ch e f. ,. a 1 e 1. he BIT . g1c, . 1 ,e ac1. g be 1ef 1. c. . e e. ce b , . . . Th1 . . . e d1 ,ega, d he d1 1. c 1. . be ee. he . . , 1. he . e. . e ha be 1ef 1. . . . e. . e'. c. . e e. ce. a ead . , . . , b h1 . eed . . a a . be he ca e a. d . . , e 1 . . , a. , . . . ca. be ba ed . . . he, , ea . . . ha. be 1ef 1. c. . e e. ce. N. e ha b. h De . . . be a. d He, . 1g a. d L. . g1 a e a. . . ac 1c a . , ach. Ab. . e. $A(t,\text{`}p\text{'})$ a. d ‘. b ec ’ a, e , e a 1. . . be ee. f. ,. a a. d . . . e . e . f . b ec . $t_1, ..., t_n$ ca ed . 1c 1 h . . addi 1. . a . , c , e. B c. . , a e ha. d e . . 1c 1. he . e a. 1c .

4.2 Simulation

I. h1 . ec 1. . e f. ,. a 1 e he , . a. d . . 1c . e, a . ,. , . 1. g a ech. 1 e ca ed *simulation*. Th1 . ea. . ha 1ca c. . e . . e, a . ,. a, e de . ed 1. e,. . . f. a. da, d . . ,. a . . da . e, a . ,. . F. , e a e, he . 1 a 1. . . f he . . . -. . ,. a , . . e, a . , 1. . . . ,. a . . da . g1c . ea. . ha he , . . e, a . , 1 de . ed . 1. g . . ,. a . . e, a . ,. , b ha he . e, a . , 1 . e f beha e 1 e a . . . -. . ,. a . e, a . , .

The ad a. age . f . 1 a 1. . a, e . f. d. F1, . , he ad a. age . f c a . 1ca . 1 a 1. . . . ch a he . 1 a 1. . f. a, 1. . 1. d . f. . . -. . ,. a . . da . g1c 1. [3, 4] 1 ha he , e . ,. . e,. . f. . ,. a . . da . g1c ca. be . ed f. , . ,. . 1. g he . - , e . . f. . . -. . ,. a . . da . g1c. Th1 ad a. age a. . h. d f. , he . 1 a 1. . f he . . . -. . ,. a , . . e, a , 1. . . ,. a . . da . g1c. Th1 . ea. . , a . . g . he, h1. g , ha 1 bec. . e ea 1e, . ha e a he , e . ,. . e, e . . ec1 ca 1. . . , 1 e.

ı he e e ded BIT gıc. Sec d, he ad a age ha ı a e he ı a ı ı hı a e ı ha ch a ı a ı gı e a dı ec a ı a ı a ı f he gıc, hıch d be b aı ed ıf he e a e e de ed e a ı-ca . I ha ca e, addı ı a a ı d ha e be gı e cha ac e ı e he e a ıc ı .

C ıde he e a , hıch ı a - a da e a . Thı e a ca be ı a ed ı g h ee a da d a da e a $\Box^1_{ij}$, $\Box^2$ a d $\Box^3$ [4]

$$T_{ij}\varphi \equiv \Diamond^1_{ij}(\Box^2\varphi \wedge \Box^3\neg\varphi)$$

he e $\Diamond\varphi$ abb e ıa e $\neg\Box\neg\varphi$ a a .

T de a d he ed c ı e e be ha h f $T_{ij}\varphi$ ı a d w f a de M ea ha he e ı a h e (eıghb h d) $S \in T_{ij}(w)$ ch ha $M, w' \models \varphi$ f e e $w' \in S$, a d $M, w'' \not\models \varphi$ f e e $w'' \notin S$. Th $\Diamond^1_{ij}$ e ab e efe he e ı e ce f a h e (eıghb h d), $\Box^2$ ı ed e e he h f φ ı S, a d $\Box^3$ e e e he fa eh d f φ ıde S.

4.3 Topic as Enumeration of Options

I hı a e , e a e ha ı ı ha e ıc a d ha ıc a e ha ed b a age [3]. F e a e, he ı ı $i(5.0)$ ha a cıa ı f a ı a ı ıc. M e e , ı he He ıg-L gı a ach ı ı ca be g e ıc , h gh hı d e a a e ı he e a e. C e e , a c ıca ı f he f a ı a ı f ıc ı ha e ha e a e hıch ıc he e a e, b ha he e a e a he ıc a aı ab e. I ı b a ı g e ıcı a gı e ıc , ha e ca a ıf e ıc . F hı ea , e ı d ce b h a e a topic a d a e a all_topics. We ıde ıf a ıc ı h he e f a ıc ı ı ha ha e hı ıc a a b ec (ee ab e). F e a e, he ıc a cıa ı f a ı ı ıde ı ed ı h he e

$$\{i(0.0), \ldots, i(10.0), e(0.50), \ldots, e(2.00)\}$$

S ch a ıc e ı be e e e ed b a f a ı e

$$\mathsf{topic}(i(0.0) \times \ldots \times i(10.0) \times e(0.50) \times \ldots \times e(2.00))$$

ı hıch '×' ı ed e a a e a e a ı e ı . O e c dı g ı a f .

Definition 4 (Topics). *The language of BIT with topics is the language of BIT, together with clause*

– *if φ is a sentence of BIT, then so are $\Box^1\varphi$, $\Box^2\varphi$, $\Box^3\varphi$ and $\Box^4\varphi$.*

Moreover, we add the following abbreviations:

– $\varphi_1 \times \ldots \times \varphi_n \equiv \Diamond^2(\Box^3\varphi_1 \wedge \Box^4\neg\varphi_1) \wedge \ldots \wedge \Diamond^2(\Box^3\varphi_n \wedge \Box^4\neg\varphi_n) \wedge \Box^2((\Box^3\varphi_1 \wedge \Box^4\neg\varphi_1) \vee \ldots \vee (\Box^3\varphi_n \wedge \Box^4\neg\varphi_n))$

[3] We assume here that topics are shared by all agents to simplify our presentation.

- $\mathsf{topic}(\varphi_1 \times \ldots \times \varphi_n) \equiv \Diamond^1(\varphi_1 \times \ldots \times \varphi_n)$
- $\mathsf{all\_topics}((\varphi_{1,1} \times \ldots \times \varphi_{1,n}); \ldots ; (\varphi_{k,1} \times \ldots \times \varphi_{k,m})) \equiv$
  $\Box^1((\varphi_{1,1} \times \ldots \times \varphi_{1,n}) \vee \ldots \vee (\varphi_{k,1} \times \ldots \times \varphi_{k,m}))$
- $\mathsf{topic\_contained}(\varphi, \psi) \equiv \Box^1(\Diamond^2(\Box^3\varphi \wedge \Box^4\neg\varphi) \supset \Diamond^2(\Box^3\psi \wedge \Box^4\neg\psi))$

The ic a i ih × a be ead a a e e e a i f a e . Tha i , d e he e ie f he da gic e ha e f e a e ha $p \times q \times r$ i ie $q \times p \times r$, $p \times p \times q \times r$, b i d e i f e a e $p \times q$.

The e a topic e e e he e f i i ha i g he a e ic; all_topics a e f he e ha he e a e a ic a ai ab e, a d topic_contained f a i e he fac ha a ic f he e e e a e a a ic f he ec d e e e . I e a e topic_contained$(i(1.0), e(2.00))$ h d . I e a e 2 a e a a i i gi e . S topic_contained(φ, ψ) e e e ha f e e $(\Box^1)$ ic, if f a φ ha ha ic $(\Diamond^2(\Box^3\varphi \wedge \Box^4\neg\varphi))$, he f a ψ ha ha ic . I i he a e abb e ia i hich i be ed f a e a ic-ba ed i fe e ce i ci e.

We a e ha ic a e ea ed a a i , i he e e ha he a e b a age , a d di ib e e i f a d e a . We he ef e acce he f i g i ci e :

$$\mathsf{topic}(\varphi_1 \times \ldots \times \varphi_n) \equiv B_i\mathsf{topic}(\varphi_1 \times \ldots \times \varphi_n)$$
$$\mathsf{topic}(\varphi_1 \times \ldots \times \varphi_n) \equiv I_{ij}\mathsf{topic}(\varphi_1 \times \ldots \times \varphi_n)$$
$$\mathsf{topic}(\varphi_1 \times \ldots \times \varphi_n) \equiv T_{ij}\mathsf{topic}(\varphi_1 \times \ldots \times \varphi_n)$$

The e a ic f BIT i h ic e e d he e a ic f BIT i h f bi a acce ibi i e a i ha c e d $\Box^1$ $\Box^4$, ha a e i e e ed i he a a . The di ib i f ic e a e he BIT da i ie i cha ac e i ed b he fac ha i each d, he e e a acce ibi i e a i a e he a e. D e ace i i a i ed gi e he de ai .

I a ee ha e c di g f he ic e a i a he c ica ed, c a ed f e a e [7], b he ad a age i ha e ha e a a da d e a ic . M e e , a i a ad a age i ha e ca e he a e e h d g f e i (ee ec i 5).

4.4 Comparison with Janin and Walukiewicz

The e c di g f he ic e a i a f he e e i f he i a i f - a da e a e i ed ab e. Thi e e i ca be de d b a a g b Ja i a d Wa ie ic [9]. The de e $a \to S =_{def} \bigwedge_{\varphi \in S} \Diamond^a\varphi \wedge \Box^a \bigvee_{\varphi \in S} \varphi$, he e a i a i de f a da e a a d S i e f f a [9]. I ea ha d w a i e f a $a \to S$ he a f a f S i a i ed b a ea e a- cce f w, a d a a- cce f w a i f a ea e f a f S. C a ica da e a a e i e a $\Diamond^a\varphi \equiv a \to \{\varphi, \top\}$ a d $\Box^a p \equiv a \to \{\varphi\} \vee a \to \emptyset$. Thi i e e ia he de i i f bi i a i ,[4] he e e e a i e ec he e e ce f da gic. A

[4] This insight is attributed to Alexandru Baltag by Yde Venema.

e ı dıca ed ab. . e, e . e he ×-. . a ı. . ı. . ead . f . e . , . . $S = \{p, q, r\}$ ı ˎe ˎe e ed b $p \times q \times r$. Lı e . e e ha e ı e ˎa ı. . a d a . . cıa ı ı , ı.e., e ca de ˎı e f. ˎ e a e $p \times q \times q \times r$. H e e, a. . . . e ha ıf . da ı ıe $\Box^a$ a d $\Diamond^a$ a ˎe . . ˎ. a , he e ca de ˎı e ea e ı g: $(p \wedge q) \times r \rightarrow p \times r$. Sı ce e d . . ı e hı . ˎ. e f. ˎ ıc , e . e . . . -. ˎ. a . da . e ˎa . ˎ. . be ˎecı e, . . . - . . . . . ıc . . e ha d . . . a ı f ea e ı g. S. , ı . . ˎ ˎed c ı . . f . . ıc , e c. . bı e . ıdea :

(a) $\Box\varphi \equiv \Diamond^2(\Box^3\varphi \wedge \Box^4\neg\varphi)$ (ı a ı . , a bef . ˎ e)

(b) $a \rightarrow S \equiv \bigwedge_{\varphi \in S} \Diamond^a\varphi \wedge \Box^a \bigvee_{\varphi \in S} \varphi$ (Ja ı a d Wa ıe ıc)

The e a ˎe c. . bı ed . ı g he de . ı ı . . f . da ı 2 acc. ˎdı g . (b), . b ı - ı g $(\Box^3\varphi \wedge \Box^4\neg\varphi)$ f. ˎ φ a d . b ı ı g '2' f. ˎ a, hıch gı e . $\bigwedge_{\varphi \in S} \Diamond^2(\Box^3\varphi \wedge \Box^4\neg\varphi) \wedge \Box^2 \bigvee_{\varphi \in S}(\Box^3\varphi \wedge \Box^4\neg\varphi)$, hıch c. ˎ. e . . d . he . ıc de . ı ı . . ab. . e. Sı ce hı . . de . e . . e . ıc, e. ı ha e . ˎe ˎe e ha_ he ˎe ı a . . ıc , f. ˎ hıch e . e $\Diamond^1$.

4.5 Topics and Trust

N e ca f. ˎ. a ı e he ı ı ı . ha ıf a ˎ. . . . ı ı . . ı ˎ . . ed, he a . . a . he ˎ ˎ. . . ı ı . . a ˎe ˎ . . ed hıch a ˎe ba ed . . he . a e . ıc . We ca ı *topic-based trust transfer* ($T3$).

$$\Diamond^1(\Diamond^2(\Box^3\varphi \wedge \Box^4\neg\varphi)) \wedge \mathsf{topic\_contained}(\varphi, \psi) \supset (T_{ij}\varphi \supset T_{ij}\psi) \quad (T3)$$

We f. ˎ. a ı e he ˎ . . . ı g e a e ı h . . ıc . Sı ce he ˎe ı . . . . e . . ıc, he e a e ı ˎe a ı e . ı e.

Example 2. The . ıc . . a cıa ı f. ˎ. a ı. . (f) ı de . ed a f. . . .

$$f \equiv (i(0.0) \times \ldots \times i(10.0) \times e(0.50) \times \ldots \times e(2.00)) \qquad \mathsf{topic}(f) \qquad \mathsf{all\_topics}(f)$$

I he . ˎ. ˎea . e . f he e a e, he ˎ . ı fe ˎe ce a 'ha ˎd c. ded'. N , e . e a ı . $T3$. de ˎı e: $T_{is}i(r_1) \vee T_{is}e(r_2) \supset (T_{is}i(r_3) \wedge T_{is}e(r_4))$. I a ˎ ıc a ˎ, f. . . $\mathsf{topic}(f)$ e ca de ˎı e $\Diamond^1(\Diamond^2(\Box^3 i(r_1) \wedge \Box^4\neg i(r_1)))$ a d f. . . $\mathsf{topic}(f)$ a d $\mathsf{all\_topics}(f)$ e ca ı fe ˎ $\mathsf{topic\_contained}(i(r_1), i(r_3))$. U ı g a ı . $T3$, e ca ı fe ˎ $T_{is}i(r_1) \supset T_{is}i(r_3)$. Sı ı a ˎ , e ca ı fe ˎ $T_{is}i(r_1) \supset T_{is}e(r_4)$ a d he ˎef. ˎe $T_{is}i(r_1) \vee T_{is}e(r_2) \supset T_{is}i(r_3) \wedge T_{is}e(r_4)$. S he ˎ. e ˎ ha a . . a ed ı E a e 1, ı . . de ˎı ed f. ˎ. . . ˎ . . ıc c. . . ˎ c ı . .

Fı a , e . . e ha Lıa d e . . dı c . . he . . . ıbı ı . add $(T_{ij}\varphi \wedge T_{ij}\psi) \supset T_{ij}(\varphi \vee \psi)$, hıch a . ˎ. ha d . . . ˎea . . ab e, ı a ˎ ıc a ˎ he φ a d ψ be . . g . he . a e . ıc . S ch a a ı . . ca be f. ˎ. a ı ed ı h . ˎ . . ıc . A . . , b c. . ˎa . . ı ı . e ca de ˎı e $\mathsf{topic\_contained}(\varphi, \psi) \supset (\neg T_{ij}\psi \supset \neg T_{ij}\varphi)$. I . he ˎ . . d , ıf a . ıc . f φ a ˎe a . ıc . f ψ, dı ˎ . . ı ψ ˎa . fe ˎ . . dı ˎ . . ı φ.

5 Questions

I hı ec ı , he gıc f Lıa ı e e ded ı h e ı , beca e f heı ecı c e a ı . Q e ı ha e bee dıed e e ı e a a f he e a ıc f a a a g age. I hı a e e e he e a ıc f e ı a d a e f G e e dı a d S h f [10]. The ıdea ı a f . C ce - a , a e ı e e e a 'ga ' ı he ı f a ı f he a e , be ed b a a e f he ıgh e. F e a e, a ' he '- e ı a f a ı e da e. S a e ı ecı e ha ı ıb e a e a e. I he e a ıc , ha ea ha a e ı e a a e he e f ıb e d ı dı ı b e , each f hıch c e d a c e e a e he e ı . The e ı g c e ı a a ı ı [10]. Tech ıca , a a ı ı ı e ı a e a e ı a e ce e a ı , ca ed a *indistinguishability relation*: he age d e dı ı g ı h be ee d ha a ı f he a e a e a e ı . F a e / e ı he e a e e f d ı he a ı ı : d ha c - e d he a e e_ , a d d ha c e d he a e _ . F a a e a ı e e ı ı e Whıch c ı he a c ıgh ? , he a ı ı c e d h ee ıb e a e : ed , e _ a d g ee_ . F a e e ı ı e Wh a e c ı g he a ? , hıch a ab g f e e c ı g he a , e d ge ıb e a e a gı g f N b d ı c e , J h ı c e , Ma ı c e a d J h a d Ma ı c e , _ E e b d ı c e . I he d , e e ı a e ea ed a a e a ı e e ı , he e each e ec ı f a c e a e e a e c - e d e a e a ı e.

Lı e ı he ca e f ıc , hı c ce a ı a ı f e ı ca be e - c ded ı g he b '×' e a a e a e a ı e . We de e a e ı b a e e ı question$_{ij}(\varphi_1 \times \ldots \times \varphi_n)$, he e $\varphi_1 ... \varphi_n$ a e he a e a- ı e a e . F e a e, Whıch c ı he a c ıgh ? ı e c ded b question$_{ij}$('*traffic_light_is_red* × *traffic_light_is_yellow* × *traffic_light_is_green*). N e ha e / e ı a e a ecıa ca e f a e a ı e e ı .

I e f he de ı a ı ca e , e eed e e he fac ha a ıb e a e a , eı he e ıcı ı ıcı , a ed f . We e he Q_{ij}- e a f hı . E e ı $Q_{ij}\varphi$ ea ha age i ha ed a e ı age j f hıch φ ı a ıb e a e . I he d , $Q_{ij}\varphi$ h d ı ca e question$_{ij}(\psi_1 \times ... \times \psi_n)$ ha bee e ıcı ı ıcı ed b age i age j, f $\varphi \equiv \psi_k$ a d $1 \leq k \leq n$.

Definition 5 (Questions). *The language of BIT with topics and questions, is the language of BIT with topics, together with the following clause:*

– *if φ is a sentence of BIT with topics, then so is $\Box_{ij}\varphi$, for $1 \leq i \neq j \leq n$.*

Moreover, we add the following abbreviations:

– question$_{ij}(\varphi_1 \times \ldots \times \varphi_n) = \Diamond_{ij}(\varphi_1 \times \ldots \times \varphi_n)$
– $Q_{ij}\varphi = \Diamond_{ij}\Diamond^2(\Box^3\varphi \wedge \Box^4\neg\varphi)$

The de ı ı ı a a g he ı a ı f ıc b a a ge f a da e a . The e a ıc f he BIT gıc ı h ıc a d e ı , e e d he e a ic f he BIT gic i h ic , i h a i ab e acce ibi i e a i c e di g $\Box_{ij}$. I he e a ic $\Diamond_{ij}$ e i a e $\mathsf{question}_{ij}$ e e e he e i e ce f a eighb h d c e di g he a e a e i f age i j. The e a $\Diamond^2$ a d $\Box^3$, $\Box^4$ a e agai ed e e he e ie f he ×- a i f a e a i e . N e ha i e , b i e ic , he e a ic f e i i ade e a i e age i a d j. Thi e e e he i i i ha ic a e a f he ge e a gica a g age, hich i ha ed b a age , he ea he e i ha ha e bee a ed a e a ic a f eci c age .

I a a , hi ide a i i a e a ic . I d e e e G e- e di a d S h f' idea f a a i i . I ca e e a de ha a e a e i be e c i e, a d ha he e e ed a e c e he h e gica ace, i.e., ha a e i a i i he gica ace, he e add he f i g a i :

$$\mathsf{question}_{ij}(\varphi_1 \times \ldots \times \varphi_n) \supset (\varphi_i \wedge \varphi_j \supset \bot), \text{ f a } 1 \leq i \neq j \leq n$$
$$\mathsf{question}_{ij}(\varphi_1 \times \ldots \times \varphi_n) \supset (\varphi_1 \vee \ldots \vee \varphi_n \equiv \top)$$

5.1 Questions and Trust

The eci c e a i be ee e i a d ha e i e f a i e i hi ec i i ba ed he f i g i i . If age i ha de ibe a e ed a e i a age j hich age i a ead be ie e he a e , a d age j ha ided i f a i ha c e d he i i ia be ief f age i, he age i i he ec d age j. O he i e, if age j ha ided he g a e , i.e. he i f a i d e c e d i' i i ia be ief , he age i i age j. Thi i i i i f a i ed b he f i g a i hich e ca *question-based trust derivation* a d *question-based distrust derivation* e ec i e .

$$(Q_{ij}\varphi \wedge B_i\varphi \wedge B_iI_{ij}\varphi) \supset T_{ij}\varphi$$
$$(Q_{ij}\varphi \wedge B_i\neg\varphi \wedge B_iI_{ij}\varphi) \supset \neg T_{ij}\varphi$$

He e, he c bi a i f $Q_{ij}\varphi$ a d $B_iI_{ij}\varphi$ i ea e e ha $I_{ij}\varphi$ i a e e a e e f age j a e i ed b age i. Thi eadi g a be b e a ic f a e i g i hich di e e e i ca be ed, i h he a e i d f a e . F e a e a a e a e a be e e a b h Whe d e he b c e? a d 'Whe d e he ai c e? . H e e , he e b e a e e e ia f he he e f i fe i g .

U i g he e a i , e ca f a i e i g e a e.

Example 3. Age i a a eb- e ice s he e cha ge a e: $\mathsf{question}_{is}(e(0.50) \times \ldots \times e(2.00))$ hich i ie $Q_{is}e(0.50) \wedge \ldots \wedge Q_{is}e(2.00)$. If he age be ie e f e a e ha he e cha ge i 1, $B_ie(1)$, a d he eb- e ice gi e he c ec a e , i.e., $B_iI_{is}e(1)$, he i g he e i -ba ed c ea i a i e ca de i e $T_{is}e(1)$. Si i a , i ca e he age ' be ief d c e d he a e , f e a e $B_ie(5)$ a d he ef e $B_i\neg e(1)$ beca e e cha ge a e a e i e, e de i e $\neg T_{is}e(1)$ b e i -ba ed di c ea i .

5.2 Questions and Topics

Q e ı . . be e . ı ı a . ıc . I he e a e, he ıc ' - . a cıa ı f a ı . ' c . e . d a c . bı a ı . f he e ı . . Wha ı he c e ı e e a e? a d Wha ı he c e e cha ge a e? . I a - a a g age e a ıc , e a ı . . be ee ıc a d e ı . . ha e g bee . . . Va K e e [11] e e de e ıc ı e . f he e ı . . ha a e c e de dı c . ı . . B a ı g a e ı . , he a e ca a ı - a e he c e ıc f he c . e a ı . . A e ed ab e, ıc a e he a e f a . d a d a age . . B c . a , e ca e Q_{ij} e e . he a ıc a ' e ı . de dı c . ı . ' f age i a d j. U de ch a ı e e a ı . , ı d a e e e ha e ı . e e c ed de ıc: $Q_{ij}\varphi \wedge \mathsf{topic\_contained}(\varphi, \psi) \supset Q_{ij}\psi$. H e e , de ch a ı ıcı ' e - ı . de dı c . ı . ' ı e e a ı . , he e ı . e a ca . be ed . de ha a age e ıcı a ed f . . e ı f a ı . . B hı ı e ac he ı e e a ı . e eed ı he ı g e a e. We he ef e e a ı e - edıa e e , ı g e ı . -ba ed c ea ı . , a d he a ı g he ıc-ba ed a fe ı cı e.

Example 4. We d ı e e he f ı g.

$$(B_i e(r) \wedge \mathsf{question}_{is}(\ldots \times e(r) \times \ldots) \wedge I_{is}e(r) \wedge \mathsf{topic\_contained}(e(r), i(r')) \wedge I_{is}i(r')) \supset B_i i(r')$$

S . e $(B_i e(r) \wedge \mathsf{question}_{is}(\ldots \times e(r) \times \ldots) \wedge I_{is}e(r) \wedge \mathsf{topic\_contained}(e(r), i(r')) \wedge I_{is}i(r'))$. Fı , de ı e $Q_{is}e(r)$ b he de ı- ı . f Q_{ij}, a d b e e $T_{is}e(r)$ b he ı cı e f e ı . -ba ed c ea ı . . Sec . d, de ı e $T_{is}i(r')$ f . $(T_{is}e(r) \wedge \mathsf{topic\_contained}(e(r), i(r'))$ b ıc-ba ed a . fe , a d hı d, de ı e $B_i i(r')$ f . $(I_{is}i(r') \wedge T_{is}i(r'))$ b Lıa ' -ba ed be ıef c ea ı . a ı . C1. F . he e h ee f a he de ı ed ı ıca ı . ca be b aı ed b ı cı e f c a ıca gıc.

6 Further Research

6.1 Other Communicative Primitives

S . e c . ıca ı ı ı e $\mathsf{proposal}_{ij}\varphi$ a d $\mathsf{request}_{ij}\varphi$ a e added he gıc, e e ha age i eceı ed a a e e f . j. Lı e a ı f , a age ı acce a a he ı he age ' ca abı - ı ıe . A d ı e a e ı , a e e eı he ı dıca e ı he he age ' ca abı ı ıe , , a a g ı g e a e, a e e ı ed e he age ' ca abı ı ıe . O ce acce ed, a a e e e e e a c . ı - e f . e f he a ıcı a achıe e . e f e a e f a aı . . The ef e e d ha e f he e e d he gıc ı h a ' ee- -ı ha ' e a $E_i\varphi$ [12]. I ha ca e, i' acce a ce f a a b j ca be e e ed b a ı - f ha i he e de j achıe e he c e f he a : $I_{ji}T_{ij}E_j\varphi$.

Si ia, , a acce a ce f a e e , i a i f ha he acce e i achie e he c e f he e e : $I_{ji}E_i\varphi$. Th i ca e f a a he e de i ac acce a ce, hi e i ca e f a e e he ecei e i ac af e ha i g acce ed.

$$\text{proposal}_{ij}\varphi \wedge I_{ji}T_{ij}E_j\varphi \supset E_j\varphi$$
$$\text{request}_{ij}\varphi \wedge I_{ji}E_i\varphi \supset E_i\varphi$$

6.2 Control Procedures

T ca be ba ed e a e a i hi be ee age , a e e ie ce , a e a i ha ha bee a ed b he ed age I he ab e ce f ch di ec i he he a , a age ha e i i i a c ced e a e e ha he age i ee hei a f he dea . E a e a e ba g a a ee a e , a bi f adi g g a a ee hi i g. H e e , if a age d e de a d a c echa i , d e he i i i ha g a a ee i , he echa i i e e . The ef e e h d a de i he c ced e . The ge e a idea ca be a i ed a f [1].

T a ac i T = Pa T + C T

If e f he a a e c , i c e d a ec . Fi , he age de a d he i g f he c echa i . F e a e, age *i* de a d ha , i hi a hi e i i i s, a bi f adi g 'c a ' e ide ce f he g d ha i g bee hi ed. A bi f adi g i a eci c i d f i f ac . I BIT e i e $I_{is}bill \supset I_{is}shipped$. Sec d, he age he i i i s ha g a a ee he c echa i . Thi ca be e e ed i BIT : $T_{is}shipped$. T ge he , he e e i ica e, ha he e e he age ecei e a bi f adi g, i i ha he g d ha e bee hi ed: $I_{is}bill \supset B_ishipped$. Thi a a i i i i g, b a he i i ed. F he e a i be ee Lia ' BIT gic a d e ide ia eed be i e iga ed.

7 Related Research

The i f ha bee died e e i e i he cia cie ce . F a e ie f e ea ch i he c e f e ec ic c e ce a d i-age e , ee Ta a d Th e [1,13]. Ge e a , i died i e a i a a ac i . Ma e e a gi e he f i g de i i f : The i-i g e f a a be e ab e he ac i f a he a ba ed he e ec a i ha he he a i e f a a ic a ac i i a he , i e ec i e f he abi i i c ha he a [14] . N e ha i i i ed f he e . A i i a e i e i f d i he de i i b Ga be a T i he b ec i e babi i b hich a i di id a A e ec ha a he i di id a B e f a gi e ac i hich i e fa e de e d [15]. B h he e de i i i dica e ha i b ec i e, a d di ec ed . a d a . he age . T . e ec . a i e e . . . a e a i . . , ha ca be ge e a i ed . . achi e . Thi a ec i . ice e ec ed i he . gic f Lia .

Ab . e . a d . ica i ha e ecei ed a . . f a e i . . i i g i ic . A . . ic a d i . . b . ic ca be . ed . ide if he . c . e . f a e . F . e a e, G . . . a d Sid e [16] e a e he . ic . f a di c . . e (a . . ca ed *center* . *focus of attention*) . he i e i . . ha i i e ded . be c . . e ed b he a h . . M . e ech ica e ea ch . . ab . e . i d . e i he c . e . f i f . . a i . . e ie a [17]. C ea , i i f . a i . e ie a i . a e . . de ha ci c . . a ce e ca . a ha . d c . e . a e ab . he . a e . ic .

A . . i . ha i . e . i i a . . . . i f . . d i he . . ca ed BAN . gic [18], . ed . de . e a he ica i . . icie i c . . e . ec i . A h gh he e i . . e ici . . i . . f . . i he e . gic , . ha i g a ec e e c . . a a . . f . f bei g . . ed. The . i i i e . f BAN . gic a e a f . . : *i sees X*, hich . ea . ha age i ecei ed a . e . age c . . ai i g X. Thi i . i i a . Lia ' i f . . ; *j said X*, hich . ea . ha age j ac a . e . a . e . age c . . ai i g X, a d ha i ca e j i . be . . ed, X . gh . be be ie ed b i; *i controls X*, hich ca be i e . e ed a . a i g ha age i i . . ed a a a h . i . . X. Thi . . i . . igh be de e . ed . a d . . . e . f . ic . I BAN . gic i i . f e . ed . e e e . . . ed hi d a ie , i e a he ica i . . e ice ; *fresh X*, hich . ea . ha X ha . . bee . e . e i . . , a d $i \overset{K}{\longleftrightarrow} j$, hich . ea . ha age i a d j a e e i ed . . e he a e ec e e K. Sha i g a e c . . . a a . . . f . f bei g . . ed. The e a e . e e a di e e ce be ee BAN . gic a d Lia ' BIT . gic a d he a he a e . ed. A . b i . . di e e ce i he . e . f e . , hich i ab e f . . Lia . A . . he di e e ce c . . ce . . he e . . ec i e: Lia ' . gic a e he ie . i . . f a . i di id a age : . de ha ci c . . a ce ca I be ie e he c . . e . f a . e . age? BAN a e he bi d' e e . ie . f a de ig e : h . h d I de ig . . . . c . a . id . ec e . ge i g . . ? The . de i g . gic i a . . di e e .

Fi a , . . ha bee . died e e . . i e i he c . . e . f a 'G id'- i e a chi ec . e f . he . ha i g . f . e . ce a d . e . ice [19]. M ch . f hi . . i a ied. H e e , he . de i g f . . a . . de ha a e de e . ed i he c . . e . f . ch e ea ch [20] de e . e . be c . . a ed i h he BIT . gic . . . . ed he e. O he f . . a i a i . . i e . . . f . . da . gic a . . e i [21].

8 Conclusion

T . . a . a i . . a . . e i ad a ced c . . e . . e . . ch a . . . a - age e . . e . i c . e . ec i [22] a d e a i . . . e a . ed f . e a e i eBa [23]. The e a ica i . . de . e a . ch . . e eci e . . i . . f . . ha he . . i . . f . . . ed i . . cia he . ie . M . e . e , i e ige age . . e . . . echa i . . . ea . . ab . he age . , f . e a e i c . . e a i . . , c . . di a i . , . . e ec . . . ic c . . e ce. Age . ha ea . . ab hei e a i . . i h he age . , . ch a age . ea . . i g ab . . . . ib e c . . e a i . . a e- gie , ca be . e f . . ea . . i g ab . . . e ici . Lia ' . gic d . e . . e . . . ch ab he i . e . . c . e . f . . . , hich . a e e be c . . ide ed a a b ac b. , b ı d. e e aı. he ,e a ı. . be ee. , . a. d . he. c. . ce . , ı. a. ıc a. he ,e a ı. . be ee. , . , be ıef a. d ı. f. ,. a ı. . ac ı. . . .

Thı a e. ,e e. . e e. . ı. . . . Lıa ' BIT . gıc, hıch a . he de. ı a ı. . . f , . . . Fı. , e e e. d he . gıc ı h . ıc . I. hı a , e ca. e ,e. ha f. . . , . ı. he , h . f . . e ,. . . ı ı. . , e ca. ı. fe. , . ı. he , h . f . he. ,. . . ı ı. . . ha a. e ,e a ed b . ıc.

Sec. . d, e e e. d he . gıc ı h e ı. . . . I. hı a , e ca. e ,e. ha ı. f. ,. . a. e e ıcı a ed f. , , . , e. e a. e ı ıcı c. . . ıde. ed ,e e. a. b a. age. . The. e a. e . ı. d . f , . ı. fe. e. ce ,ı. cı e . We . ıgh . a ha b . e ec ı. g a. . he. age. . a a e ı. . , . ı. dıca e ha . ı , . . hı . he. age. . Th . , e ı. . . ı , . . . O. he . he. ha. d, e ı. . . . a be a ed . ,a egıca . I. , , , . . ı. g e a e he age. de ıbe. a e a ed f. , a e ı. . ı h a . . . a. . e. , ı. . . de. . ı. fe. ıf he ,e ı. g age. c. d be , . ed . . . ,. . . ı ı. . . f a ,e a ed . ıc.

A e ı. . c. . ce. . . he a ıcabı ı . f , . . ,ı. cı e . We ha e a ,ead . ee. . a e. . a ı e ,ı. cı e ,ega. dı. g , . a. d e ı. . . . I a. . . ee . ,ea . . ab e . ,e ,ıc he , . de. ı a ı. . a ı. . . . ı a ı. . . ı. hıch he age. ı ,e a ı e ıg. . ,a. . I. a e a . ı a ı. . , he eache. . . . he a. . e. . . a he e ı. . . he a . . B ac. , ,ec a. . e. . he . ,. e ı. . ı . . . ece . a. ı . a e he eache. , . he . de. ab. he a. . e. . . he ,e aı. ı. g e ı. . . . Thı . h. . ha he . . cıa c. . e ı. hıch , . ı a ıed, . eed . be. . de ed . e. ca. ef .

The. e a. e . e. e. a ı . , a. , . . e. ıe . f , . hıch ,e aı. . dı c . . ed. The . gıc d. e . . ca ,e he e e e. . f ,ı . I. he , . . ı. g e a e, , . ı. g he eb-. e. ıce ı ,ı , beca . e he . , f. ı. . a. age e. . f he age. de e. d . . ı . N. e ha ı h. . ch a ,ı , he age. . d . . g. h. . gh he , . b e . f e ı. g he . e. ıce ı h he e ı. . ab. e cha. ge ,a e .

We b. ıe ı. dıca ed h. he . gıc . ıgh be f , he. e e. ded ı h ,e e . a. d , . . . a. . Thı h. e. e. , . d ,e ı. e a . hıf f. . . a. e ı e ıc . . ı. . . f , . . , ab. be ıef , . a . . ,e ,ac ıca . . ı. . . f , . . , ab. ac ı. . . . We a. . dı c . . ed h. he . gıc ı ,e a ed . . . ,e ge. e. a ,a. . ac ı. . . . de. . f , . . , hıch ı. , . . e c. . ,. . echa. ı . . g a. a. eed b a. ı. . ı ı. . . M. ,e ,e ea. ch ı . eeded . c. . . ec he e. . de. ı h . , . . ı. . ı ı. . a . . , . . .

References

[1] Tan, Y.H., Thoen, W.: Formal aspects of a generic model of trust for electronic commerce. In: 33rd Hawaii International Conference on System Sciences (HICSS'00). (2000) p. 6006

[2] Liau, C.J.: Belief, information acquisition, and trust in multi-agent systems – a modal formulation. Artificial Intelligence **149** (2003) 31–60

[3] Gasquet, O., Herzig, A.: From classical to normal modal logics. In Wansing, H., ed.: Proof Theory of Modal Logics. Volume 2 of Applied Logic Series. Kluwer (1996) 293–311

[4] Kracht, M., Wolter, F.: Normal monomodal logics can simulate all others. Journal of Symbolic Logic **64** (1999) 99–138

[5] Goffman, E.: Strategic interaction. University of Pennsylvania Press, Pennsylvania (1969)

[6] Demolombe, R.: To trust information sources: a proposal for a modal logical framework. In Castelfranchi, C., Tan, Y.H., eds.: Trust and Deception in Virtual Societies. Kluwer (2001) 111–124

[7] Demolombe, R., Jones, A.: On sentences of the kind "sentence 'p' is about topic t". In: Logic, language and reasoning: Essays in Honour of Dov Gabbay. Kluwer Academic, Dordrecht (1999) 115–133

[8] Herzig, A., Longin, D.: Belief dynamics in cooperative dialogues. Journal of Semantics **17** (2000) 91–118

[9] Janin, D., Walukiewicz, I.: Automata for the modal mu-calculus and related results. In: Proceedings of the 20th International Symposium on Mathematical Foundations of Computer Science (MFCS'95). LNCS 969, Springer Verlag (1995) 552–562

[10] Groenendijk, J., Stokhof, M.: Questions. In Van Benthem, J., Ter Meulen, A., eds.: Handbook of Logic and Language. North-Holland, Elsevier (1996) 1055–1124

[11] van Kuppevelt, J.: Discourse structure, topicality and questioning. Journal of Linguistics **31** (1995) 109–149

[12] Horty, J.: Agency and Deontic Logic. Oxford University Press (2001)

[13] Tan, Y.H., Thoen, W.: An outline of a trust model for electronic commerce. Applied Artificial Intelligence **14** (2000) 849–862

[14] Mayer, R., Davis, J., Schoorman, F.: An integrative model of organizational trust. Academy of Management Review **20** (1995) 709–734

[15] Gambetta, D.: Can we trust trust? In: Trust. Basil Blackwell, New York (1988) 213–237

[16] Grosz, B., Sidner, C.: Attentions, intentions and the structure of discourse. Computational Linguistics **12** (1986) 175–204

[17] Huibers, T.: An Axiomatic Theory for Information Retrieval. PhD thesis, Utrecht University (1996)

[18] Burrows, M., Abadi, M., Needham, R.: A logic of authentication. ACM Transactions on Computer Systems **8** (1990) 18–36

[19] Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the Grid: Enabling scalable virtual organizations. .International Journal of High Performance Computing Applications **15** (2001) 200–222

[20] Carbone, M., Nielsen, M., Sassone, V.: A formal model for trust in dynamic networks. In: International Conference on Software Engineering and Formal Methods (SEFM'03), IEEE (2003) 54–63

[21] Jones, A., Firozabadi, B.S.: On the characterisation of a trusting agent - aspects of a formal approach. In Castelfranchi, C., Tan, Y., eds.: Trust and Deception in Virtual Societies. Kluwer Academic Publishers (2001) 157–168

[22] Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: IEEE Symposium on Security and Privacy. IEEE (1996) 164–173

[23] Dellarocas, C.: The digitization of word-of-mouth: Promise and challenges of online feedback mechanisms. Management Science **49** (2004) 1407–1424

Coordination Between Logical Agents

Chia i Sa a a[1] a d Ka . . . i I . . e[2]

[1] Department of Computer and Communication Sciences,
Wakayama University,
Sakaedani, Wakayama 640 8510, Japan
sakama@sys.wakayama-u.ac.jp
[2] National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101 8430, Japan
ki@nii.ac.jp

Abstract. In this paper we suppose an agent that has a knowledge base written in logic programming and sets of beliefs under the answer set semantics. We then consider the following two problems: given two logic programs P_1 and P_2, which have the sets of answer sets $\mathcal{AS}(P_1)$ and $\mathcal{AS}(P_2)$, respectively; (i) find a program Q which has the set of answer sets such that $\mathcal{AS}(Q) = \mathcal{AS}(P_1) \cup \mathcal{AS}(P_2)$; (ii) find a program R which has the set of answer sets such that $\mathcal{AS}(R) = \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$. A program Q satisfying the condition (i) is called *generous coordination* of P_1 and P_2; and R satisfying (ii) is called *rigorous coordination* of P_1 and P_2. Generous coordination retains all of the original belief sets of each agent, but admits the introduction of additional belief sets of the other agent. By contrast, rigorous coordination forces each agent to give up some belief sets, but the result remains within the original belief sets for each agent. We provide methods for constructing these two types of coordination and discuss their properties.

1 Introduction

I . i-age . . e . di e e age . . . a ha e di e e . e . f be ief , a d age . . eg ia e a d acc . . . da e he . e . e . each acce ab e ag ee e . . . We ca a . ce . f f . i g ch ag ee e . be ee age . *coordination.* The . b e i h . . e e a ag ee e acce ab e . each age . The . c . e . f c . di a i . i e i ed . be c . . i e a d i de i ab e . e ai . . igi a i f . a i . . f each age a . ch a . . . ib e.

S . . . e a age ha ha a . . edge ba e a a . gic . . g a h . e . e- . a ic i gi e a he c ec i . . f *answer sets* [7]. A . e . e . e e e . e . f i e a c . e . di g . be ief hich ca be b i b a a i . a ea . . e . . he ba i . f a . . g a [2]. A age . a ha e (c . . ic i g) a e a i e e . f be ief , hich a e e e e ed b . i e a . . e . e . f a . . g a . Di e e age . ha e di e e c . ec i . . . f a . e . e . i ge e a . We he ca . e c . - . di a i . be ee . age . a he . b e . f . di g a . e . . g a hich ha he . ea i g ba a ced be ee . . . g a . . C . . . ide , f . i . a ce, a . gic

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 161–177, 2005.

g a P_1 hich ha a e e S_1 a d S_2; a d a he gic g a P_2 hich ha a e e S_2 a d S_3. The , e a d a e g a hich i a e f c di a i be ee P_1 a d P_2. I hi a e , e c ide di e e i : e i a g a Q hich ha h ee a e e S_1, S_2, a d S_3; he he i a g a R hich ha he i g e a e e S_2.

The e i ide di e e e f c di a i he e e ai a f he igi a be ief e f each age , b ad i he i d c i f addi i a be ief e f he he age . B c a , he ec d e f ce each age gi e e be ief e , b he e e ai i hi he igi a be ief e f each age . The e e f c di a i cc i ea ife. F i a ce, e he f i g ce a i : decide he Acade A a d f Be Pic e , each e be f he Acade i a e . N he e a e h ee e be p_1, p_2, a d p_3, a d each e be ca i a e a : p_1 i a e f_1 a d f_2, p_2 i a e f_2 a d f_3, a d p_3 i a e f_2. A hi e , h ee i ee f_1, f_2, a d f_3 a e ed. The i a i i e e e ed b h ee g a :

$$P_1: \quad f_1 \,;\, f_2 \leftarrow,$$
$$P_2: \quad f_2 \,;\, f_3 \leftarrow,$$
$$P_3: \quad f_2 \leftarrow,$$

he e ; e e e di c i . He e, P_1 ha a e e : $\{f_1\}$ a d $\{f_2\}$; P_2 ha a e e : $\{f_2\}$ a d $\{f_3\}$; P_3 ha he i g e a e e : $\{f_2\}$. The h ee i ee c e d he a e e : $\{f_1\}$, $\{f_2\}$, a d $\{f_3\}$. A g a ha i g he e h ee a e e i he e f c di a i . Af e a i g, he f_2 i ed b h ee e be a d bec e he i e f he A a d. Tha i , he i e i e e e ed b he a e e $\{f_2\}$. A g a ha i g hi i g e a e e i he ec d e f c di a i . Th , he e e f c di a i ha e i di e e i a i , a d i i ea i g f de e c a i a gic f he e c di a i be ee age .

The be i he h b i d a g a hich ea i e ch c di a i . F a , he b e c ide ed i hi a e a e de c ibed a f .

Given: g a P_1 a d P_2;

Find: (1) a g a Q a i f i g $\mathcal{AS}(Q) = \mathcal{AS}(P_1) \cup \mathcal{AS}(P_2)$;

(2) a g a R a i f i g $\mathcal{AS}(R) = \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$,

he e $\mathcal{AS}(P)$ e e e he e f a e e f a g a P. The g a Q a i f i g (1) i ca ed *generous coordination* f P_1 a d P_2; a d he g a R a i f i g (2) i ca ed *rigorous coordination* f P_1 a d P_2. We de e e h d f c i g he e e f c di a i a d e if he e .

The e f hi a e i ga i ed a f . Sec i 2 e e de i i a d e i gie ed i hi a e . Sec i 3 i d ce a f a e f c di a i be ee gic g a . Sec i 4 ide e h d f c i g c di a i a d add e e hei e ie . Sec i 5 di c e e a ed i e a d Sec i 6 a i e he a e .

2 Preliminaries

I hı a e , e e a age ha ha a edge ba e ı e ı gıc g a ı g. A age ı he ıde ı ed ı h ı gıc g a a d e e h e e ı e cha geab h gh he a e .

A *program* c ıde ed ı hı a e ı a *extended disjunctive program* (EDP) hıch ı a e f *rules* f he f :

$$L_1 ; \cdots ; L_l \leftarrow L_{l+1} , \ldots , L_m, not\, L_{m+1} , \ldots , not\, L_n \quad (n \geq m \geq l \geq 0)$$

he e each L_i ı a ı ı e/ ega ı e ı e a , ı.e., A $\neg A$ f a a A, a d *not* ı *negation as failure* (NAF). *not* L ı ca ed a *NAF-literal*. The b ; e - e e dı c ı . The ef -ha d ıde f he e ı he *head*, a d he ıgh -ha d ıde ı he *body*. F each e r f he ab e f , $head(r)$, $body^+(r)$, $body^-(r)$, a d $not\_body^-(r)$ de e he e f (NAF-) ı e a $\{L_1, \ldots, L_l\}$, $\{L_{l+1}, \ldots, L_m\}$, $\{L_{m+1}, \ldots, L_n\}$, a d $\{not\, L_{m+1}, \ldots, not\, L_n\}$, e ec ı e . A dı c ı f ı - e a a d a c c ı f (NAF-) ı e a ı a e a e ıde ı ed ı h ı c e- dı g e f (NAF-) ı e a . A e r ı f e ı e a $head(r) \leftarrow body^+(r)$, $not\_body^-(r)$ $head(r) \leftarrow body(r)$ he e $body(r) = body^+(r) \cup not\_body^-(r)$. A e r ı *disjunctive* ıf $head(r)$ c aı e ha e ı e a . A e r ı a *integrity constraint* ıf $head(r) = \emptyset$; a d r ı a *fact* ıf $body(r) = \emptyset$. A g a P ı *NAF-free* ıf $body^-(r) = \emptyset$ f a e r ı P. A g a ı h a ıab e ı e- a ıca ıde ı ed ı h ı g dı a ıa ı , a d e ha d e ı ı a a d g d g a h gh he a e .

The e a ıc f EDP ı gı e b he *answer set semantics* [7]. Le *Lit* be he e f a g d ı e a ı he a g age f a g a . A e $S (\subset Lit)$ *satisfies* a g d e r ıf $body^+(r) \subseteq S$ a d $body^-(r) \cap S = \emptyset$ ı $head(r) \cap S \neq \emptyset$. I a ıc a , S a ı e a g d ı eg ı c aı r ı h $head(r) = \emptyset$ ıf eı he $body^+(r) \not\subseteq S$ $body^-(r) \cap S \neq \emptyset$. S a ı e a g d g a P ıf S a ı e e e e ı P. Whe $body^+(r) \subseteq S$ (e . $head(r) \cap S \neq \emptyset$), ı ı a ı e a $S \models body^+(r)$ (e . $S \models head(r)$).

Le P be a NAF-f ee EDP. The , a e $S (\subset Lit)$ ı a *(consistent) answer set* f P ıf S ı a ı ı a e ch ha

1. S a ı e e e e f he g d ı a ıa ı f P,
2. S d e c aı a aı f c e e a ı e a L a d $\neg L$ f a $L \in Lit$.

Ne , e P be a EDP a d $S \subset Lit$. F e e e r ı he g d ı a ıa ı f P, he e r^S : $head(r) \leftarrow body^+(r)$ ı ı c ded ı he *reduct* P^S ıf $body^-(r) \cap S = \emptyset$. The , S ı a *answer set* f P ıf S ı a a e e f P^S. A EDP ha e, e, ı e a e e ı ge e a . The e f a a e e f P ı ı e a $\mathcal{AS}(P)$. A g a P ı *consistent* ıf ı ha a c ı e a e e . I hı a e , e a e ha a g a ı c ı e e a ed he ı e.

A ı e a L ı a c e e ce f *credulous reasoning* ı a g a P (ı e a $L \in crd(P)$) ıf L ı ı c ded ı e a e e f P. A ı e a L ı a c e e ce f *skeptical reasoning* ı a g a P (ı e a $L \in skp(P)$) ıf L ı ı c ded ı e e a e e f P. C ea , $skp(P) \subseteq crd(P)$ h d f a P. T g a

P_1 a d P_2 a e aid be *AS-combinable* if e e e i $\mathcal{AS}(P_1) \cup \mathcal{AS}(P_2)$ i i i a de e i c i .

Example 2.1. Gi e g a :

$$P_1 : \; p\,;\, q \leftarrow,$$
$$p \leftarrow q,$$
$$q \leftarrow p,$$
$$P_2 : \; p \leftarrow not\, q,$$
$$q \leftarrow not\, p,$$

he e $\mathcal{AS}(P_1) = \{\{p,q\}\}$ a d $\mathcal{AS}(P_2) = \{\{p\},\{q\}\}$. The , $crd(P_1) = skp(P_1) = \{p,q\}$; $crd(P_2) = \{p,q\}$ a d $skp(P_2) = \emptyset$. P_1 a d P_2 a e AS-c bi ab e beca e he e $\{p,q\}$ i i i a i $\mathcal{AS}(P_1) \cup \mathcal{AS}(P_2)$.

Tech ica , he g a P_1 a d P_2 a e AS-c bi ab e, e ca a e he AS-c bi ab e b i d ci g he e $\overline{L} \leftarrow not\, L$ f e e $L \in Lit$ each g a , he e $\overline{L}$ i a e i d ced a a cia ed i e i h each L.

Example 2.2. I he ab e e a e, $P_1' = P_1 \cup Q$ a d $P_2' = P_2 \cup Q$ i h

$$Q : \; \overline{p} \leftarrow not\, p,$$
$$\overline{q} \leftarrow not\, q\,.$$

The , $\mathcal{AS}(P_1') = \{\{p,q\}\}$ a d $\mathcal{AS}(P_2') = \{\{p,\overline{q}\},\{\overline{p},q\}\}$, P_1' a d P_2' a e AS-c bi ab e.

3 Coordination Between Programs

Gi e g a , c di a i ide a g a hich i a ea ab e c i e be ee age . I hi ec i , e i d ce di e e e f c di a i de he a e e e a ic .

Definition 3.1. Le P_1 a d P_2 be g a . A g a Q a i f i g he c di i $\mathcal{AS}(Q) = \mathcal{AS}(P_1) \cup \mathcal{AS}(P_2)$ i ca ed *generous coordination* f P_1 a d P_2; a g a R a i f i g he c di i $\mathcal{AS}(R) = \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$ i ca ed *rigorous coordination* f P_1 a d P_2.

Ge e c di a i e ai a f he a e e f each age , b ad- i he i d c i f addi i a a e e f he he age . B c a , ig c di a i f ce each age gi e e a e e , b he e e ai i hi he igi a a e e f each age .

Tech ica , ge e c di a i e i e g a P_1 a d P_2 be AS- c bi ab e, i ce a e e f Q a e a i i a . Th , he e c ide ge - e c di a i be ee g a , e a e he be AS-c bi ab e.

Ge e . c . dı a ı . be ee . g a ha a e . AS-c . bı ab e ı . ıb e b a ı g he AS-c . bı ab e ı ad a ce ı g he . g a a . f . a ı . e e ed ı Sec ı . 2.

Definition 3.2. F . . g a P_1 a d P_2, e Q be a e f ge e . c . dı a ı . , a d R a e f ıg . c . dı a ı . . We a ha ge e . (e . ıg .) c . dı a ı . *succeeds* ıf $\mathcal{AS}(Q) \neq \emptyset$ (e . $\mathcal{AS}(R) \neq \emptyset$); he ı e, ı *fails.*

Ge e . c . dı a ı . a a . cceed he e e b h P_1 a d P_2 a e c . ı - e . O he he ha d, he $\mathcal{AS}(P_1) \cap \mathcal{AS}(P_2) = \emptyset$, ıg . c . dı a ı . faı . a age . ha e . c . be ıef . e . N e ha ge e . c . dı a ı . a . d ce a c . ec ı . f a . e . e . hıch c . adıc ı h . e a . he . B hı d e . ca . e a . b e a a c . ec ı . f a . e . e . e e e . (c . ıc ı g) a e . a ı e be ıef . e . f each age .

A e a . e c . ı e . g a . , he e e h d b he de . ı ı . .

Proposition 3.1 *When generous/rigorous coordination of two programs succeeds, the result of coordination is consistent.*

C . dı a ı . cha ge he c . e e ce . f c ed . / e ıca ea . ı g b each age .

Proposition 3.2 *Let P_1 and P_2 be two programs.*

1. *If Q is a result of generous coordination,*
   (a) $crd(Q) = crd(P_1) \cup crd(P_2)$;
   (b) $skp(Q) = skp(P_1) \cap skp(P_2)$;
   (c) $crd(Q) \supseteq crd(P_i)$ *for* $i = 1, 2$;
   (d) $skp(Q) \subseteq skp(P_i)$ *for* $i = 1, 2$.
2. *If R is a result of rigorous coordination,*
   (a) $crd(R) \subseteq crd(P_1) \cup crd(P_2)$;
   (b) $skp(R) \supseteq skp(P_1) \cup skp(P_2)$ *if* $\mathcal{AS}(R) \neq \emptyset$;
   (c) $crd(R) \subseteq crd(P_i)$ *for* $i = 1, 2$;
   (d) $skp(R) \supseteq skp(P_i)$ *for* $i = 1, 2$ *if* $\mathcal{AS}(R) \neq \emptyset$.

Proof. 1.(a) A ı e a L ı ı c ded ı a a . e . e ı $\mathcal{AS}(P_1) \cup \mathcal{AS}(P_2)$ ı L ı ı c ded ı a a . e . e ı $\mathcal{AS}(P_1)$. ı c ded ı a a . e . e ı $\mathcal{AS}(P_2)$. (b) L ı ı c ded ı e e a . e . e ı $\mathcal{AS}(P_1) \cup \mathcal{AS}(P_2)$ ı L ı ı c ded ı e e a . e . e ı $\mathcal{AS}(P_1)$ a d a . ı c ded ı e e a . e . e ı $\mathcal{AS}(P_2)$. The e . f (c) a d (d) h d b (a) a d (b), e ec ı e .

2.(a) If L ı ı c ded ı a a . e . e ı $\mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$, L ı ı c ded ı a a . e . e ı $\mathcal{AS}(P_i)$ $(i = 1, 2)$. (b) If L ı ı c ded ı e e a . e . e . f eı he P_1 . P_2, L ı ı c ded ı e e a . e . e ı $\mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$ ıf he ı e . ec ı . ı . e . The e . f (c) a d (d) h d b (a) a d (b), e ec ı e . □

Example 3.1. Le $\mathcal{AS}(P_1) = \{\{a,b,c\},\{b,c,d\}\}$ a d $\mathcal{AS}(P_2) = \{\{b,c,d\},\{c,e\}\}$, he e $crd(P_1) = \{a,b,c,d\}$, $skp(P_1) = \{b,c\}$, $crd(P_2) = \{b,c,d,e\}$, a d $skp(P_2) = \{c\}$. Ge e c di a i Q f P_1 a d P_2 ha he a e e $\mathcal{AS}(Q) = \{\{a,b,c\},\{b,c,d\},\{c,e\}\}$ he e $crd(Q) = \{a,b,c,d,e\}$ a d $skp(Q) = \{c\}$. Rig c di a i R ha he a e e $\mathcal{AS}(R) = \{\{b,c,d\}\}$ he e $crd(R) = skp(R) = \{b,c,d\}$. The ab e ea i a e e i ed f he e e .

Ge e c di a i e ge c ed c e e ce f P_1 a d P_2, hi e e ic e ica c e e ce h e ha a e c be ee - g a . A a e , i i c ea e c ed c e e ce a d dec ea e e ica c e e ce . Thi e ec he i a i ha acce i g i i f he he age i c ea e a e a i e ch ice hi e ea e i g he igi a a g e f each age . B c a , ig c di a i ed ce c ed c e e ce , b i c ea e e ica c e e ce i ge e a . Thi e ec he i a i ha e c di g i i f he he age c aba d i g e f e' a e a i e be ief , hich e i e g he i g e igi a a g e f each age .

Definition 3.3. F g a P_1 a d P_2, e Q be a e f ge e c di a i , a d R a e f ig c di a i . Whe $\mathcal{AS}(Q) = \mathcal{AS}(P_1)$ (e . $\mathcal{AS}(R) = \mathcal{AS}(P_1)$), P_1 *dominates* P_2 de ge e (e . ig) c - di a i .

Proposition 3.3 *Let P_1 and P_2 be two programs. When $\mathcal{AS}(P_1) \subseteq \mathcal{AS}(P_2)$, P_2 dominates P_1 under generous coordination, and P_1 dominates P_2 under rigorous coordination.*

Whe P_2 d i a e P_1 de ge e c di a i , e ca ea i ha e a e f ge e c di a i a $Q = P_2$. Si i a , he P_1 d i a e P_2 de ig c di a i , a e f ig c di a i bec e $R = P_1$.

I ca e he e e age d i a e he he e, he c di a i fai , he e f c di a i a e i ia a d i e e i g. The , he be f i e e i he ca e he e $\mathcal{AS}(P_1) \not\subseteq \mathcal{AS}(P_2)$ a d $\mathcal{AS}(P_2) \not\subseteq \mathcal{AS}(P_1)$ f c - i g ge e / ig c di a i ; a d $\mathcal{AS}(P_1) \cap \mathcal{AS}(P_2) \neq \emptyset$ f c i g ig c di a i . I he e ec i , e e e e h d f c i g he e c di a i .

4 Computing Coordination

4.1 Computing Generous Coordination

We e e a e h d f c i g ge e c di a i be ee g a .

Definition 4.1. Gi e g a P_1 a d P_2,

$$P_1 \oplus P_2 = \{\, head(r_1)\,;\, head(r_2) \leftarrow body_*(r_1),\, body_*(r_2) \mid r_1 \in P_1,\, r_2 \in P_2 \,\},$$

he e $head(r_1)\,;\,head(r_2)$ ı he dı c ı f $head(r_1)$ a d $head(r_2)$, $body_*(r_1) = body(r_1) \setminus \{\, not\, L \mid L \in T \setminus S \,\}$ a d $body_*(r_2) = body(r_2) \setminus \{\, not\, L \mid L \in S \setminus T \,\}$ f a $S \in \mathcal{AS}(P_1)$ a d $T \in \mathcal{AS}(P_2)$.

The g a $P_1 \oplus P_2$ ı a c ec ı f e hıch a e b aı ed b c bı ı g a e f P_1 a d a e f P_2 ı e e ıbe a . I $body_*(r_1)$ e e NAF- ı e a $not\, L$ ch ha $L \in T \setminus S$ ı d ed beca e he e ı e ce f hı a e e he de ı a ı f e ı e a ı $head(r_2)$ af e c bı a ı .

Example 4.1. C ıde g a :

$$\begin{aligned} P_1 :\ & p \leftarrow not\, q, \\ & q \leftarrow not\, p, \\ P_2 :\ & \neg p \leftarrow not\, p, \end{aligned}$$

he e $\mathcal{AS}(P_1) = \{\{p\}, \{q\}\}$ a d $\mathcal{AS}(P_2) = \{\{\neg p\}\}$. The , $P_1 \oplus P_2$ bec e

$$\begin{aligned} p\,;\, \neg p &\leftarrow not\, q, \\ q\,;\, \neg p &\leftarrow not\, p. \end{aligned}$$

N e ha $not\, p$ f he e f P_2 ı d ed ı he e ı g e beca e f he e ı e ce f $\{p\}$ ı $\mathcal{AS}(P_1)$.

B he de ı ı , $P_1 \oplus P_2$ ı c ed ı ı e $|P_1| \times |P_2| \times |\mathcal{AS}(P_1)| \times |\mathcal{AS}(P_2)|$, he e $|P|$ e e e he be f e ı P a d $|\mathcal{AS}(P)|$ e e e he be f a e e ı P.

The g a $P_1 \oplus P_2$ ge e a c aı e e ed da ı e a / e , a d he f ı g g a a f a ı a e he f ı ıf he g a .

- (e ı ı a ı f a gıe : TAUT)
  De e e a e r f a g a ıf $head(r) \cap body^+(r) \neq \emptyset$.
- (e ı ı a ı f c adıc ı : CONTRA)
  De e e a e r f a g a ıf $body^+(r) \cap body^-(r) \neq \emptyset$.
- (e ı ı a ı f - ı ı a e : NONMIN)
  De e e a e r f a g a ıf he e ı a he e r' ı he g a ch ha $head(r') \subseteq head(r)$, $body^+(r') \subseteq body^+(r)$ a d $body^-(r') \subseteq body^-(r)$.
- (e gı g d ıca ed ı e a : DUPL)
  A dı c ı $(L; L)$ a ea ı g ı $head(r)$ ı e ged ı L, a d a c c- ı (L, L) $(not\, L,\ not\, L)$ a ea ı g ı $body(r)$ ı e ged ı L $not\, L$, e ec ı e .

The e g a a f a ı a e e e he a e e f a EDP [3].

Example 4.2. Gı e g a :

$$\begin{aligned} P_1 :\ & p \leftarrow q, \\ & r \leftarrow, \\ P_2 :\ & p \leftarrow not\, q, \\ & q \leftarrow r, \end{aligned}$$

$P_1 \oplus P_2$ bec e

$$p\,;\,p \leftarrow q,\ not\,q,$$
$$p\,;\,q \leftarrow q,\ r,$$
$$p\,;\,r \leftarrow not\,q,$$
$$r\,;\,q \leftarrow r.$$

The e ı de e ed b CONTRA, he ec d e a d he f h e a e de e ed b TAUT. Af e ch e ı ı a ı , he e ı g g a c aı he hı d e .

N e h ha $P_1 \oplus P_2$ ea ı e ge e c dı a ı f P_1 a d P_2.

Lemma 4.1 *Let P_1 and P_2 be two NAF-free AS-combinable programs. Then, S is an answer set of $P_1 \oplus P_2$ iff S is an answer set of either P_1 or P_2.*

Proof. S e ha S ı a a e e f P_1. The , S a ı e a e $head(r_1) \leftarrow body(r_1)$ ı P_1, he eb a ı e a e $head(r_1); head(r_2) \leftarrow body(r_1), body(r_2)$ ı $P_1 \oplus P_2$. (N e: $body_*(r_i) = body(r_i)$ f NAF-f ee - g a .) T ee ha S ı a a e e f $P_1 \oplus P_2$, e ha he e ı a ı ı a e $T \subset S$ hıch a ı e e e e ı $P_1 \oplus P_2$. Sı ce S ı a a e e f P_1, he e ı a e r_1' ı P_1 hıch ı a ı ed b T. F hı e, $T \not\models head(r_1')$ a d $T \models body(r_1')$ h d. The , f a e $head(r_1'); head(r_2) \leftarrow body(r_1'), body(r_2)$ ı $P_1 \oplus P_2$, $T \models head(r_2)$ $T \not\models body(r_2)$. Sı ce e e e ı P_2 ı c bı ed ı h r_1', ı h d ha $T \models head(r_2)$ $T \not\models body(r_2)$ f e e r_2 ı P_2. The , T a ı e P_2. A P_2 ı c ı e , ı ha a a e e $T' \subseteq T$. Thı c adıc he a ı ha P_1 a d P_2 a e AS-c bı ab e, ı.e., $T' \not\subseteq S$. He ce, S ı a a e e f $P_1 \oplus P_2$. The ca e ha S ı a a e e f P_2 ı ed ı he a e a e .

C e e , e ha S ı a a e e f $P_1 \oplus P_2$. The , S a ı - e a e $head(r_1); head(r_2) \leftarrow body(r_1), body(r_2)$ ı $P_1 \oplus P_2$. The $S \models body(r_1), body(r_2)$ ı ıe $S \models head(r_1); head(r_2)$. If $S \not\models head(r_1)$ f e e $r_1 \in P_1$, $S \models head(r_2)$ f a $r_2 \in P_2$. The , $S \models body(r_2)$ ı ıe $S \models head(r_2)$ f a $r_2 \in P_2$, ha S a ı e e e e ı P_2. E e ıf $S \not\models head(r_2)$ f e e $r_2 \in P_2$, $S \models head(r_1)$ f a $r_1 \in P_1$. The , $S \models body(r_1)$ ı - ıe $S \models head(r_1)$ f a $r_1 \in P_1$, ha S a ı e e e e ı P_1. E e ıf $S \models head(r_1)$ f e e $r_1 \in P_1$ a d $S \models head(r_2)$ f e e $r_2 \in P_2$, S a ı e b h P_1 a d P_2. Th , ı e e ca e S a ı e eı he P_1 P_2. S e ha S a ı e P_1 b ı ı a a e e f P_1. The , he e ı a a e e T f P_1 ch ha $T \subset S$. B he ıf- a , T bec e a a e e f $P_1 \oplus P_2$. Thı c adıc he a ı ha S ı a a e e f $P_1 \oplus P_2$. Sı ı a a g e ı a ıed he S a ı e P_2. □

Theorem 4.2. *Let P_1 and P_2 be two AS-combinable programs. Then, $\mathcal{AS}(P_1 \oplus P_2) = \mathcal{AS}(P_1) \cup \mathcal{AS}(P_2)$.*

Proof. S . . e $S \in \mathcal{AS}(P_1)$. The , S ı a. a. e. e f P_1^S, . . ha S ı a. a. e. e f $P_1^S \oplus P_2^T$ f . a $T \in \mathcal{AS}(P_2)$ (Le a 4.1). (N e: a P_1 a d P_2 a e AS-c . bı ab e, he ed c P_1^S a d P_2^T a e a . AS-c . bı ab e.) F . a e $head(r_1); head(r_2) \leftarrow body^+(r_1), body^+(r_2)$ ı $P_1^S \oplus P_2^T$, ı h d ha $body^-(r_1) \cap S = body^-(r_2) \cap T = \emptyset$. O he . he ha d, f . a e $head(r_1); head(r_2) \leftarrow body_*(r_1), body_*(r_2)$ ı $P_1 \oplus P_2$, $head(r_1); head(r_2) \leftarrow body^+(r_1), body^+(r_2)$ ı ı $(P_1 \oplus P_2)^S$ ı $(body^-(r_1) \setminus \{ L \mid L \in T \setminus S' \}) \cap S = \emptyset$ a d $(body^-(r_2) \setminus \{ L \mid L \in S' \setminus T \}) \cap S = \emptyset$ f . a $S' \in \mathcal{AS}(P_1)$ a d $T \in \mathcal{AS}(P_2)$. He e ı h d ha $(body^-(r_1) \setminus \{ L \mid L \in T \setminus S' \}) \cap S \subseteq body^-(r_1) \cap S$ a d $(body^-(r_2) \setminus \{ L \mid L \in S' \setminus T \}) \cap S \subseteq body^-(r_2) \cap T \cap S \subseteq body^-(r_2) \cap T$. He ce, $P_1^S \oplus P_2^T \subseteq (P_1 \oplus P_2)^S$. S . . e a e $head(r_1); head(r_2) \leftarrow body^+(r_1), body^+(r_2)$ ı $(P_1 \oplus P_2)^S \setminus (P_1^S \oplus P_2^T)$. Sı ce S . a ı e a e r_1 ı P_1, $S \models body^+(r_1), body^+(r_2)$ ı ıe $S \models head(r_1); head(r_2)$. Th . , he a . e . e S . f $P_1^S \oplus P_2^T$. a ı e e e e ı $(P_1 \oplus P_2)^S \setminus (P_1^S \oplus P_2^T)$. B $P_1^S \oplus P_2^T \subseteq (P_1 \oplus P_2)^S$, S bec . e a a . e . e . f $(P_1 \oplus P_2)^S$ a d $S \in \mathcal{AS}(P_1 \oplus P_2)$. The ca e . f $S \in \mathcal{AS}(P_2)$ ı . . . ed ı he . a e . a . e .

C . . e e , ... e $S \in \mathcal{AS}(P_1 \oplus P_2)$. The , S . a ı e a e $head(r_1); head(r_2) \leftarrow body_*(r_1), body_*(r_2)$ ı $P_1 \oplus P_2$, . . $S \models body_*(r_1), body_*(r_2)$ ı ıe $S \models head(r_1); head(r_2)$. If $S \not\models head(r_1)$ f . . . . e e $r_1 \in P_1$, $S \models head(r_2)$ f . a $r_2 \in P_2$. The , $S \models body_*(r_2)$ ı ıe $S \models head(r_2)$ f . a $r_2 \in P_2$, . . $S \models head(r_2)$ . . $S \not\models body_*(r_2)$. A $S \not\models body_*(r_2)$ ı ıe $S \not\models body(r_2)$, ı h d ha $S \models head(r_2)$ . . $S \not\models body(r_2)$ f . a $r_2 \in P$. He ce, S . a ı e e e e ı P_2. E e ıf $S \not\models head(r_2)$ f . . . . e e $r_2 \in P_2$, ı ı . h . ı a . ı ıa . a . e ha S . a ı e e e e ı P_1. E e ıf $S \models head(r_1)$ f . e e $r_1 \in P_1$ a d $S \models head(r_2)$ f . e e $r_2 \in P_2$, S . a ı e b h P_1 a d P_2. Th . , ı e e ca e S . a ı e eı he P_1 . . P_2. S . . . e ha S . a ı e P_1 b ı ı . . a a . e . e . f P_1. The , he e ı a a . e . e T . f P_1 . ch ha $T \subset S$. B he ıf- a , T bec . e a a . e . e . f $P_1 \oplus P_2$. Thı c . . adıc . he a . . . ı . ha S ı a a . e . e . f $P_1 \oplus P_2$. Sı ı a a g . e ı a ıed he S . a ı e P_2. □

Example 4.3. I E a e 4.1, $\mathcal{AS}(P_1 \oplus P_2) = \{\{p\}, \{q\}, \{\neg p\}\}$, he eb $\mathcal{AS}(P_1 \oplus P_2) = \mathcal{AS}(P_1) \cup \mathcal{AS}(P_2)$.

4.2 Computing Rigorous Coordination

Ne e e e a e h d . f c . . ı g ıg . . . c . . dı a ı . be ee . . . . -g a . .

Definition 4.2. Gı e . . . . g a P_1 a d P_2,

$$P_1 \otimes P_2 = \bigcup_{S \in \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)} R(P_1, S) \cup R(P_2, S),$$

he e $\mathcal{AS}(P_1) \cap \mathcal{AS}(P_2) \neq \emptyset$ a d

$$R(P, S) = \{ head(r) \cap S \leftarrow body(r),\ not\ (head(r) \setminus S) \mid r \in P \text{ a d } r^S \in P^S \}$$

a d $not\ (head(r) \setminus S) = \{ not\ L \mid L \in head(r) \setminus S \}$.

Whe. $\mathcal{AS}(P_1) \cap \mathcal{AS}(P_2) = \emptyset$, $P_1 \otimes P_2$ ı . de . ed.[1]

I ı ı e , he . g a $P_1 \otimes P_2$ ı a c. ec ı . . f e hıch a be . ed f c. . . c ı g a . e . e . ha a e c. . . . be ee. P_1 a. d P_2. I. $R(P, S)$ a. ı e a ı. $head(r)$ hıch d. e . . c. . ıb e . he c. . . c ı . . f he a. . e . e S ı . hıf ed . he b. d a NAF- ı e a. . $P_1 \otimes P_2$. a c. . aı. ed . da e , hıch a e e ı ı a ed . ı g . g a a . f . a ı . . gı e. ı. he e ı . . b ec ı . .

Example 4.4. C. . . ıde . . g a . :

$$
\begin{aligned}
P_1 :\ & p \leftarrow not\, q,\ not\, r, \\
& q \leftarrow not\, p,\ not\, r, \\
& r \leftarrow not\, p,\ not\, q, \\
P_2 :\ & p\,;\, q\,;\, \neg r \leftarrow not\, r,
\end{aligned}
$$

he e $\mathcal{AS}(P_1) = \{\{p\}, \{q\}, \{r\}\}$, $\mathcal{AS}(P_2) = \{\{p\}, \{q\}, \{\neg r\}\}$, a. d $\mathcal{AS}(P_1) \cap \mathcal{AS}(P_2) = \{\{p\}, \{q\}\}$. The. , $P_1 \otimes P_2$ bec. . e

$$
\begin{aligned}
& p \leftarrow not\, q,\ not\, r, \\
& q \leftarrow not\, p,\ not\, r, \\
& p \leftarrow not\, r,\ not\, q,\ not\, \neg r, \\
& q \leftarrow not\, r,\ not\, p,\ not\, \neg r.
\end{aligned}
$$

He e, he hı d a. d he f . h e ca. be e ı ı a ed b NONMIN.

B he de . ı ı . , $P_1 \otimes P_2$ ı c. . ed ı ı e $(|P_1| + |P_2|) \times |\mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)|$ he e $|\mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)|$ e e e . he . . be . f a. . e . e . ı. $\mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$.

$P_1 \otimes P_2$ ea ı e ıg . . . c. . dı a ı . . f P_1 a. d P_2.

Lemma 4.3 *Let P be a program. Then, S is an answer set of P iff S is an answer set of $R(P, S)$.*

Proof. S ı a. a. . e . e . f P ı S ı a. a. . e . e . f P^S
ı S ı a . ı ı a . e . ch ha $body^+(r) \subseteq S$ ı ıe $head(r) \cap S \neq \emptyset$ f e e . e $head(r) \leftarrow body^+(r)$ ı. P^S (∗). B he de . ı ı . . f $R(P, S)$, he e $head(r) \leftarrow body^+(r)$ ı ı. P^S ı he c. . e . . dı g e $head(r) \cap S \leftarrow body^+(r)$ ı ı. $R(P, S)^S$ (beca . e $body^-(r) \cap S = \emptyset$ a. d $(head(r) \setminus S) \cap S = \emptyset$). He. ce, he . a e e. (∗) h d ı S ı a . ı ı a . e . ch ha $body^+(r) \subseteq S$ ı ıe $head(r) \cap S \neq \emptyset$ f . e e . e $head(r) \cap S \leftarrow body^+(r)$ ı. $R(P, S)^S$
ı S ı a . ı ı a . e hıch . a ı . e e e . e $head(r) \cap S \leftarrow body^+(r)$ ı. $R(P, S)^S$
ı S ı a. a. . e . e . f $R(P, S)$. □

[1] Technically, $P_1 \otimes P_2$ is set as $\{\, p \leftarrow not\, p \,\}$ for any atom p.

Theorem 4.4. *Let P_1 and P_2 be two programs. Then, $\mathcal{AS}(P_1 \otimes P_2) = \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$.*

Proof. S . . . e $S \in \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$. The , S . a ı . e a. . e *head*$(r) \leftarrow$ *body*(r) ı. P_1 a. d P_2, . . ha S . a ı . e he c. , , e . . dı. g . e *head*$(r) \cap T \leftarrow$ *body*(r), *not* (*head*$(r) \setminus T$) ı. $R(P_1,T) \cup R(P_2,T)$ f . a. $T \in \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$. Th . , S . a ı . e $P_1 \otimes P_2$. S . . . e ha S ı . . a. a. . e . e . f $P_1 \otimes P_2$. The , he e ı a. ı. ı a . e $U \subset S$ hıch . a ı . e e e . e ı. $P_1 \otimes P_2$. I. hı ca e, U . a ı . e $R(P_1, S)$. B Le . a 4.3, h. e e , S ı a. ı. ı a . e hıch . a ı . e $R(P_1, S)$. C. . adıc ı. . . He. ce, S ı a. a. . e . e . f $P_1 \otimes P_2$.

C. . . e . e , ... e $S \in \mathcal{AS}(P_1 \otimes P_2)$. The , S ı a. ı. ı a . e hıch . a ı . e e e . e *head*$(r) \cap T \leftarrow$ *body*(r), *not* (*head*$(r) \setminus T$) ı. $R(P_1,T) \cup R(P_2,T)$ f . a. $T \in \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$. B Le . a 4.3, T ı a. . a. ı. ı a . e hıch . a ı . e b. h $R(P_1,T)$ a. d $R(P_2,T)$, . . ha he e ı a ı e a $L \in S \setminus T$ a. d a ı e a $M \in T \setminus S$. H. e e , a. . e ı. $R(P_1,T) \cup R(P_2,T)$ ha he head *head*$(r) \cap T$, . . ha . . ı e a $L \in S \setminus T$ ı ı. c ded ı. he head. Th . , L ı . . ı. c ded ı. he a. . e . e S, he eb $S \setminus T = \emptyset$. A b. h T a. d S a e . ı. ı a , $T \setminus S = \emptyset$. He. ce, $T = S$ a. d $S \in \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$. □

Example 4.5. I. E a . e 4.4, $\mathcal{AS}(P_1 \otimes P_2) = \{\{p\}, \{q\}\}$, he eb $\mathcal{AS}(P_1 \otimes P_2) = \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$.

4.3 Algebraic Properties

I. hı . b ec ı . , e . . . ıde . . . e ıe . f he . e a ı. . . $\oplus$ a. d $\otimes$.

Proposition 4.5 *For programs P_1, P_2, and P_3, the operations $\oplus$ and $\otimes$ have the following properties:*

(i) $P_1 \oplus P_2 = P_2 \oplus P_1$ and $P_1 \otimes P_2 = P_2 \otimes P_1$;
(ii) $(P_1 \oplus P_2) \oplus P_3 = P_1 \oplus (P_2 \oplus P_3)$ if P_1, P_2 and P_3 are NAF-free;
(iii) $(P_1 \otimes P_2) \otimes P_3 = P_1 \otimes (P_2 \otimes P_3)$.

Proof. The e . . . f (ı) a. d (ıı) a e . aıgh f . a. d. T. . ee (ııı), $\mathcal{AS}(P_1 \otimes P_2) = \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2)$ h. d b The . e 4.4. The , b. h $(P_1 \otimes P_2) \otimes P_3$ a. d $P_1 \otimes (P_2 \otimes P_3)$ c. . . ı . f . e ı. $R(P_1, S) \cup R(P_2, S) \cup R(P_3, S)$ f . e e $S \in \mathcal{AS}(P_1) \cap \mathcal{AS}(P_2) \cap \mathcal{AS}(P_3)$. □

The . . e a ı. . $\oplus$ ı . . . a . . cıa ı e ı. ge. e a he. . . g a . c. . aı. NAF, b ı h. d he . e a ı. . $\mathcal{AS}((P_1 \oplus P_2) \oplus P_3) = \mathcal{AS}(P_1 \oplus (P_2 \oplus P_3))$. $\oplus$ ı a. . *idempotent*, $P \oplus P = P$ ıf NONMIN a. d DUPL a e a ıed . $P \oplus P$ a. d P. $\otimes$ ı . . ıde . e. b he . e a ı. . $\mathcal{AS}(P \otimes P) = \mathcal{AS}(P)$ h. d . B he de . ı ı. . , $P \otimes P$ ha he e ec . f e . ac ı. g . e . ed f . c. . . . c ı. g a. . e . e . f P.

B P . . . ı ı. . 4.5, he . ıg . . . c. . dı a ı. . a e d. . e a . . g . . e ha . age . , he . de . f c. . ı. g c. . dı a ı. . d e . . a ec he . e . f . a . c. . e. Thı ı a. . he ca e f . ge. e . . c. . dı a ı. . he. . . g a . a e NAF-f ee.

T e f c dı a ı a e ı ed a g age . I hı ca e, he ab - ı a a d he dı ıb ı a d h d ı ge e a , ı.e.,

$$P_1 \oplus (P_1 \otimes P_2) \neq P_1 \text{ a d } P_1 \otimes (P_1 \oplus P_2) \neq P_1;$$
$$P_1 \oplus (P_2 \otimes P_3) \neq (P_1 \oplus P_2) \otimes (P_1 \oplus P_3) \text{ a d}$$
$$P_1 \otimes (P_2 \oplus P_3) \neq (P_1 \otimes P_2) \oplus (P_1 \otimes P_3),$$

N e ha g a a e ge e a dı e e , b he f ı g e a ı h d b he de ı ı :

$$\mathcal{AS}(P_1 \oplus (P_1 \otimes P_2)) = \mathcal{AS}(P_1 \otimes (P_1 \oplus P_2)) = \mathcal{AS}(P_1),$$
$$\mathcal{AS}(P_1 \oplus (P_2 \otimes P_3)) = \mathcal{AS}((P_1 \oplus P_2) \otimes (P_1 \oplus P_3)),$$
$$\mathcal{AS}(P_1 \otimes (P_2 \oplus P_3)) = \mathcal{AS}((P_1 \otimes P_2) \oplus (P_1 \otimes P_3)).$$

5 Discussion

Whe a e f a e e ı gı e , ı ı dı c c c a g a hıch ha e ac h e a e e . Gı e a e f a e e $\{S_1, \ldots, S_m\}$, c e he dı c ı e a f : $S_1 \vee \cdots \vee S_m$, he c e ı ı he c c ı e a f : $R_1 \wedge \cdots \wedge R_n$. The e f fac $\{R_1, \ldots, R_n\}$ he ha he a e e $\{S_1, \ldots, S_m\}$. Thı ech ı e ı a ed f c ı g c dı a ı be ee g a . F ı a ce, e g a :

$$P_1: \quad \mathit{sweet} \leftarrow \mathit{strawberry},$$
$$\mathit{strawberry} \leftarrow,$$
$$P_2: \quad \mathit{red} \leftarrow \mathit{strawberry},$$
$$\mathit{strawberry} \leftarrow,$$

he e $\mathcal{AS}(P_1) = \{\{\mathit{sweet}, \mathit{strawberry}\}\}$ a d $\mathcal{AS}(P_2) = \{\{\mathit{red}, \mathit{strawberry}\}\}$.

T ge ge e c dı a ı hıch ha he a e e $\mathcal{AS}(P_1) \cup \mathcal{AS}(P_2)$, a ı g he DNF f each a e e d ce

$$(\mathit{sweet} \wedge \mathit{strawberry}) \ \vee \ (\mathit{red} \wedge \mathit{strawberry}).$$

C e ı g ı ı he CNF, ı bec e

$$(\mathit{sweet} \vee \mathit{red}) \wedge \mathit{strawberry}.$$

A a e , he e f fac

$$Q: \quad \mathit{sweet}\,;\, \mathit{red} \leftarrow,$$
$$\mathit{strawberry} \leftarrow$$

ı a g a hıch ı ge e c dı a ı f P_1 a d P_2. O he he ha d, he g a $P_1 \oplus P_2$ bec e

$$sweet\,;\, red \leftarrow strawberry,$$
$$strawberry \leftarrow,$$

af e e ı ı a ı g d ıca ed ı e a a d ed da e .

The e g a ha e he a e ea ı g b ha e dı e e a . The , a e ı ı hıch e ı e efe ab e a a e f c dı a ı ? O a e ı $P_1 \oplus P_2$. The ı ı ı behı d hı e ec ı ı ha e d ı e ı c de a ch ı f a ı a ıb e f he ıgı a g a . C a ı g Q ı h $P_1 \oplus P_2$, ı f a ı f de e de c be ee *sweet* (*red*) a d *strawberry* ı ı Q.[2] Ge e a ea ı g, ıf he e e ı dı e e ca dıda e f c dı a ı be ee g a , a g a hıch ı ac ıca c e he ıgı a e ı efe ed. The , a e ı ı h ea e ch ac ıca c e e be ee g a ? O e ı e ha e ı ı d ı , a ı a ed ab e, ı g de e de c e a ı be ee ı e a . We efe a e f c dı a ı hıch ı he ı de e de c e a ı f he ıgı a g a a ch a ıb e.

M e ecı e , e he *dependency graph* f a g a P ı hıch each de e e e a g d ı e a a d he e ı a dı ec ed edge f L_1 L_2 (e a L_1 *depends on* L_2) ı he e ı a g d e ı P ch ha L_1 a ea ı he head a d L_2 a ea ı he b d f he e. Le (L_1, L_2) be a aı f g d ı e a ch ha L_1 de e d L_2 ı he de e de c g a h f a g a . Le $\delta(P)$ be he c ec ı f ch aı ı P. F g a P_1 a d P_2, e ha dı e e g a P_3 a d P_4 a e b aı ed a ca dıda e f c dı a ı . The , e a ha P_3 ı *preferable* P_4 ıf

$$\Delta(\delta(P_3), \delta(P_1) \cup \delta(P_2)) \subset \Delta(\delta(P_4), \delta(P_1) \cup \delta(P_2)),$$

he e $\Delta(S, T)$ e e e he e ıc dı e e ce be ee e S a d T, ı.e., $(S \backslash T) \cup (T \backslash S)$. A ı g he ab e e a e, $\delta(P_1) = \{(sweet, strawberry)\}$, $\delta(P_2) = \{(red, strawberry)\}$, $\delta(Q) = \emptyset$, a d $\delta(P_1 \oplus P_2) = \{(sweet, strawberry), (red, strawberry)\}$. The , $\Delta(\delta(P_1 \oplus P_2), \delta(P_1) \cup \delta(P_2)) \subset \Delta(\delta(Q), \delta(P_1) \cup \delta(P_2))$, e c c de ha $P_1 \oplus P_2$ ı efe ab e Q. F he e ab a ı d be c ıde ed e ec ac ıca c e e , b e d e hı ı e f he he e.

C dı a ı e ha dı e e g a ha e e a a dı g a d c bı e h e g a hı e a ı a ee ı g ıgı a ı f a ı f he . The b e f c bı ı g gıca he ıe ha bee dıed b e e a e ea che ı dı e e c e . Ba a *et al.* [1] ı d ce a g ı h f c bı ı g gıc g a b e f cı g a ı fac ı f ı eg ı c aı . F ı a ce, e g a :

$$\begin{aligned} P_1 :\quad & p(x) \leftarrow not\, q(x), \\ & q(b) \leftarrow r(b), \\ & q(a) \leftarrow, \\ P_2 :\quad & r(a) \leftarrow, \end{aligned}$$

[2] Technically, the program Q is obtained by unfolding rules in $P_1 \oplus P_2$ [3, 11].

. ge he ı h he ı eg ı c . . . aı . :

$$IC: \;\; \leftarrow p(a), r(a),$$
$$\leftarrow q(a), r(a).$$

The c . bı e P_1 a d P_2 a d . . d ce a . e . . g a hıch . a ı . e IC a f . . :

$$P_3: \;\; p(x) \leftarrow not\, q(x),\; x \neq a,$$
$$q(b) \leftarrow r(b),$$
$$q(a) \vee r(a) \leftarrow .$$

B c . . a , $(P_1 \cup IC) \oplus P_2$ ı . . . f a e . . bec . . e [3]

$$p(x)\,;\, r(a) \leftarrow not\, q(x),$$
$$q(b)\,;\, r(a) \leftarrow r(b),$$
$$q(a)\,;\, r(a) \leftarrow,$$

af e e ı ı a ı g a . . gıe . C . . a ı g . . e . , he . . g a P_3 ha . a . . e . e . $\{p(b), q(a)\}$ a d $\{p(b), r(a)\}$; b c . . a , $(P_1 \cup IC) \oplus P_2$ ha . a . . e . e . : $\{p(b), q(a)\}$ a d $\{r(a)\}$. Th . , he a . . e . e . f P_3 d . . . c ı cıde ı h h . e f he . . ıgı a . . g a . . I deed, he . e e ha e e . a . . e . e . f a . e ı g . . g a . be a . b e . f a . a . . e . e . f $P_1 \cup P_2$. Thı ı ı c . . a . . . a . . ach he e e e e he e . f c . . dı a ı . . ee (a . f) he a . . e . e . f he . . ıgı a . . g a . . A . he ı . . a dı e e ce ı ha a g . ı h . ı [1] a e . . a ıcab e . . . . a ı ed . gıc . . g a . , hı e . . . e h d ı a ıed . e e e e ded dı . c ı e . . g a .

The . . b e . f *program composition* ha bee . dıed b . e e a . e ea che . (e.g., [4, 6, 12]). I c . bı e dı e e . . g a . ı . . . e. The . . b e ı he h . . . . ıde he . ea ı g . f a . . g a ı . e . . . f h . e c . . . e . . B . gı *et al.* [4] ı . . d ce h ee . e a- e e . e a ı . . f . c . . . ı g . . . a . gıc . . - g a . : . ı . . , ı . e . ec ı . . , a d . e . ıc ı . . The . ı . . ı . . . g a . ge he . , a d he ı . e . ec ı . . c . . bı e . . . g a . b . e gı g aı . f . e ı h . ı ab e head . F . . ı . . a ce, gı e . . . . g a . :

$$P_1: \;\; likes(x, y) \leftarrow not\; bitter(y),$$
$$hates(x, y) \leftarrow sour(y);$$
$$P_2: \;\; likes(Bob, y) \leftarrow sour(y),$$

he . . g a $P_1 \cap P_2$ c . . . ı . . f he . ı g e . e:

$$likes(Bob, y) \leftarrow not\; bitter(y),\; sour(y).$$

The . e . ıc ı . . a . . . . e . e . . . . . e . e f . . a . . g a . The e - . Fı ı g' 3- a ed . ı . e a . ıc a d . h . h . . . e ca . c . . e he

---

[3] Here IC is included in P_1 as we handle integrity constraints as a part of a program.

e a ic f he c ed g a i e f he igi a g a . I he c e f a e gic g a , Ve bae e *et al.* [12] i d ce a a ia f he e -f ded e a ic , a d ide if c di i f g a P_1 a d P_2 a i f he e a i $Mod(P_1 \cup P_2) = Mod(P_1) \cap Mod(P_2)$ he e $Mod(P)$ i he e f de f P. E a e a d Te i [6] c ide h ee- a ed c e i e a ic f g a c i i a he i f a e g a . C - a i g he e h ee die i h , b h g a e a i a d de i g e a ic a e di e e f . M e e , he g a f g a c i i i c e he ea i g f he h e g a i e f i b g a ; he he ha d, g a i c c a g a h e a e e a e he i /i e ec i f he igi a g a .

C bi a i f i i a he ie ha bee died de he a e f *merging* [8] *arbitration* [9]. The g a f he e e ea ch i ide a e he hich i c i e a d e e e a ch i f a i a ib e f hei ce . Me gi g i di e e f c di a i e e ed i hi a e . F i a ce, he ie $P_1 = \{p \leftarrow\}$ a d $P_2 = \{q \leftarrow\}$ a e e ged i $P_3 = \{p \leftarrow\ ,\ q \leftarrow\}$. B c a , ge e c di a i f P_1 a d P_2 bec e $P_1 \oplus P_2 = \{p ; q \leftarrow\}$. Th , i c a ge e c di a i , e gi g d e e e e a e e f he igi a g a . I e gi g di e e be ief b di e e age a e i ed ge he a fa a he a e c i e , hich a e i di c di i g i h he igi a be ief f e age af e e gi g. Thi i ie he b e ha igi a be ief f e age a e ha d ec e he e f he i f a i ce i c ec . F i a ce, e a age ha he g a $P_4 = \{p ; q \leftarrow\}$ a d e i f a i $P_1 = \{p \leftarrow\}$ a i e . If P_4 a d P_1 a e e ged, he e bec e $P_5 = \{p \leftarrow\}$. La e , i ha he fac p i P_1 d e h d. A hi age, he age ca ec e he igi a g a P_4 f P_5. B c a , if ge e c di a i i d e, i bec e $P_4 \oplus P_1 = P_4$ a d he igi a i f a i P_4 i e .

Cia i i *et al.* [5] i d ce a a g age f c di a i g gic-ba ed age . The ha d e e f c di a i : c ab a i a d c e i i . The g a i e he e di e e e f e ie i g abd c i , a d c - c a g a a a e f c di a i . Rece , Me e *et al.* [10] i d ce a gica f a e f eg ia i g age . The i d ce di e e de f eg ia i : c ce i a d ada a i . The cha ac e i e ch eg ia i b a i a a e a d ide e h d f c c i g c e . Th e a e a e ge e a a ied ic he ie , a d i hi e e c di a i c ide ed i hi a e i be ide he b ec f h e a e .

C di a i i d ced i hi a e i ai e i he e e ha i a e he i /i e ec i f di e e c ec i f a e e . We ca de e a ia f c di a i b i d ci g a egie ha de e d i a i . F i a ce, he he e a e e ha age , i i c ide ed a e he *majority* i acc a i [8]. Gi e c ec i f a e e b h ee age , $\{S_1, S_2, S_3\}$, $\{S_2, S_4\}$, a d $\{S_1, S_5\}$, ch a i i ci e a b i d $\{S_1, S_2\}$ a a e f c di a i , h e e be i ed b e ha e age . P i i ie be ee age a e a c ide ab e. I he ab. . e e a e, ıf he. ec. . d age. ı . . . . e ıab e, e ca. ha e a ch. ıce . a e S_4 ı. . acc. . . We ca. a. . c. . . ıde. . . e. g. aı. . . f c. . . . ı ı. . . . ch a ha ı. g $S_1 \cup S_2$ . . $S_1 \cap S_2$ a a . e . f c. . . dı. a ı. . f. . . . a. . e. . e. S_1 a. d S_2 (he. e $S_1 \cup S_2$ ı a. . . ed c. . . ı e.). De aı ed . dıe . . . ch . a. ıa. . a. e ef . f . he. . e ea. ch.

6 Concluding Remarks

Thı a e. ha . dıed c. . . dı. a ı. . be ee. . gıca age. . . Gı e. . ı e age. . a gıc . . g. a . , . dı e. e. e . f c. . . dı. a ı. . ha e bee. ı. . . - d ced a. d heı. c. . a ı. . a . e h. d ha e bee. . . . ıded. We ha e . e. ı ed ha he . . . . ed . e h. d . ea ı e ge. e. . . /. ıg. . . . c. . . dı. a ı. . be ee. . gıc . . g. a . . O . c. . . dı. a ı. . f. a e . . . . . ıde a c. . . . ı ı. . a . e a. ıc . f . ı e age. . a. d . e. . e a a dec a. a ı e ba ı f. . acc. . . . da ı. . ı. . ı- age. . . e . . F. . . he . ıe . ı. . f a. . e. . e . . g. a . ı. g, he . . ce . . f c. . ı. g c. . . dı. a ı. . ı c. . . ıde. ed a a . . g. a de. e. . . e. . de. a. ecı - ca ı. . ha . e e . a . . g. a . e ec ı. g he. ea. ı. g . f . . . . . . e . . g. a . . Thı . e a e . he ı. . e . f . . g. a c. . . . ı ı. . de. he a. . e. . e . e a. - ıc . Thı a e. c. . . ıde. ed he a. . e. . e . e a. ıc b a. . ı ıa. f. a e . . . d be de. e. . ed . de. dı e. e. . e a. ıc (h. gh c. . . a ı. . a . e h. d a. e ı e . be dı e. e.).

The. e ı . ı . . . . f. . ı . . . e e. ı. c. . ı. g ge. e. . . /. ıg. . . . c. . . dı- . a ı. . . The . e. a ı. . . $\oplus$ a. d $\otimes$ ı . . d ced ı. hı a e. . e ı. e c. . . a ı. . . f a. . e. . e . . f . . ıgı. a . . g. a . , b ı ı . ch be e. ıf c. . . dı. a ı. . ca. be c. . . . c ed b . e . . ac ıc. a. ı a ı. . ı h. c. . ı. g h. . e a. . e. . e. . . F . he. , he . e. a ı. . $\oplus$ . . d ce a dı . c ı e . . g. a e. e. he. he . . ıgı. a . . g. a . a. e . . . -dı . c ı e . . g. a . . The . e ı. g dı . c ı e . . - g. a ı . ed ced . a . . . -dı . c ı. e . . e ıf ı ı *head-cycle-free*, b hı ı . . he ca e ı. ge. e. a . A he. . . e. , e d. . . ha e. . ı. . . f. . he e . . b e . . I. f . e . . , e ı . e . e . . f. a e . . a. d a. . ı. . e ıga e . he. e . f c. . . dı. a ı. . a. d c. ab. . a ı. . a e a heı. cha. ac e. ı. a ı. . ı. e. . . f c. . . a ı. . a . gıc.

References

1. C. Baral, S. Kraus, and J. Minker. Combining multiple knowledge bases. *IEEE Transactions of Knowledge and Data Engineering* 3(2):208–220, 1991.
2. C. Baral and M. Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming* 19/20:73–148, 1994.
3. S. Brass and J. Dix. Characterizations of the disjunctive stable semantics by partial evaluation. *Journal of Logic Programming* 32(3):207–228, 1997.
4. A. Brogi, S. Contiero, and F. Turini. Composing general logic programs. *Proc. 4th International Conference on Logic Programming and Nonmonotonic Reasoning, Lecture Notes in Artificial Intelligence* 1265, pp. 273–288, Springer, 1997.

5. A. Ciampolini, E. Lamma, P. Mello, F. Toni, and P. Torroni. Cooperation and competition in ALIAS: a logic framework for agents that negotiate. *Annals of Mathematics and Artificial Intelligence* 37(1/2), pp. 65–91, 2003.
6. S. Etalle and F. Teusink. A compositional semantics for normal open programs. *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pp. 468–482, MIT Press, 1996.
7. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9(3/4):365–385, 1991.
8. S. Konieczny and R. Pino-Pérez. On the logic of merging. *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 488–498, Morgan Kaufmann, 1998.
9. P. Liberatore and M. Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering* 10(1):76–90, 1998.
10. T. Meyer, N. Foo, R. Kwok, and D. Zhang. Logical foundation of negotiation: outcome, concession and adaptation. *Proceedings of the 19th National Conference on Artificial Intelligence*, pp. 293–298, MIT Press, 2004.
11. C. Sakama and H. Seki. Partial deduction in disjunctive logic programming. *Journal of Logic Programming* 32(3):229–245, 1997.
12. S. Verbaeten, M. Denecker, and D. De. Schreye. Compositionality of normal open logic programs. *Proceedings of the 1997 International Symposium on Logic Programming*, pp. 371–385, MIT Press, 1997.

A Computational Model for Conversation Policies for Agent Communication

Jamal Bentahar[1], Bernard Moulin[1], John-Jules Ch. Meyer[2], and Brahim Chaib-draa[1]

[1] Laval University, Department of Computer Science and Software Engineering, Canada
jamal.bentahar.1@ulaval.ca
{bernard.moulin, brahim.chaib-draa}@ift.ulaval.ca
[2] University Utrecht, Department of Computer Science, The Netherlands
jj@cs.uu.nl

Abstract. In this paper we propose a formal specification of a persuasion protocol between autonomous agents using an approach based on social commitments and arguments. In order to be flexible, this protocol is defined as a combination of a set of conversation policies. These policies are formalized as a set of dialogue games. The protocol is specified using two types of dialogue games: entry dialogue game and chaining dialogue games. The protocol terminates when exit conditions are satisfied. Using a tableau method, we prove that this protocol always terminates. The paper addresses also the implementation issues of our protocol using logical programming and an agent-oriented platform.

1 Introduction

Research in agent communication has received much attention during the past years [9; 13; 14]. Agent communication protocols specify the rules of interaction governing a dialogue between autonomous agents in a multi-agent system. These protocols are patterns of behavior that restrict the range of allowed follow-up utterances at any stage during a dialogue. Unlike protocols used in distributed systems, agent communication protocols must take into account the fact that artificial agents are autonomous and proactive. These protocols must be flexible enough and must also be specified using expressive formalisms. Indeed, logic-based protocols seem an interesting way for specifying these protocols [3; 16].

On the one hand, conversation policies [18] and dialogue games [12; 21] aim at offering more flexible protocols [20]. This is achieved by combining different policies and games to construct complete and more complex protocols. In this paper we argue that conversation policies and dialogue games are related and can be used together to specify agent communication. Conversation policies are declarative specifications that govern communication between autonomous agents. We propose to formalize these policies as a set of dialogue games. Dialogue games are interactions between players, in which each player moves by performing utterances according to a pre-defined set of roles. Indeed, protocols specified using, for example, finite state machines are not flexible in the sense that agents must respect the whole protocol from the beginning to

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 178 – 195, 2005.

the end. Thus, we propose to specify these protocols by small conversation policies that can be logically put together using a combination of dialogue games.

On the other hand, in the last years, some research works addressed the importance of social commitments in the domain of agent communication [4; 5; 11; 20; 24; 27]. These works showed that social commitments are a powerful representation to model multi-agent interactions. Commitments provide a basis for a normative framework that makes it possible to model agents' communicative behaviors. This framework has the advantage of being expressive because all speech act types can be represented by commitments [11]. Commitment-based protocols enable the content of agent interactions to be represented and reasoned about [17; 28]. In opposition to the BDI mental approach, the commitment-approach stresses the importance of conventions and the public aspects of dialogue. A speaker is committed to a statement when he makes this statement or when he agreed upon this statement made by another participant. In fact, we do not speak here about the expression of a belief, but rather about a particular relationship between a participant and a statement. What is important in this approach is not that an agent agrees or disagrees upon a statement, but rather the fact that the agent *publicly expresses* agreement or disagreement, and acts accordingly.

In this paper we present a persuasion dialogue which is specified using conversation policies, dialogue games and a framework based on commitments. In addition, in order to allow agents to effectively reason on their communicative actions, our framework is also based on an argumentative approach. In our framework the agent's reasoning capabilities are linked to their ability to argue. In this paper *we consider conversation policies as units specified by dialogue games whose moves are expressed in terms of actions that agents apply to commitments and arguments*. Indeed, the paper presents three results: 1- A new formal language for specifying a persuasion dialogue as a combination of conversation policies. 2- A termination proof of the dialogue based on a tableau method [10]. 3- An implementation of the specification using an agent oriented and logical programming.

The paper is organized as follows. In Section 2, we introduce the main ideas of our approach based on commitments and arguments. In Section 3 we address the specification of our persuasion protocol based on this approach. We present the protocol form, the specification of each dialogue game and the protocol dynamics. We also present our termination proof. In Section 4 we describe the implementation of a prototype allowing us to illustrate how the specification of dialogue games is implemented. In Section 5 we compare our protocol to related work. Finally, in Section 6 we draw some conclusions and we identify some directions for future work.

2 Commitment and Argument Approach

2.1 Social Commitments

A social commitment *SC* is a public commitment made by an agent (the *debtor*), that some fact is true or that something will be done. This commitment is directed toward a set of agents (*creditors*) [8]. A commitment is an obligation in the sense that the debtor must respect and behave in accordance with this commitment. A representation of this notion as directed obligations using a deontic logic is proposed in [19]. Commitments are social in the sense that they are expressed publicly. Consequently, they are different from the private mental states like beliefs, desires and intentions. In order to model the dynamics of conversations, we interpret a speech act *SA* as an action performed on a commitment or on its content [4]. A speech act is an abstract act that an agent, the speaker, performs when producing an utterance *U* and addressing it to another agent, the addressee. In the dialogue games that we specify in Section 3, the actions that an agent can perform on a commitment are: $Act \in \{Create, Withdraw\}$. The actions that an agent can perform on a commitment content are: $Act\text{-}content \in \{Accept, Refuse, Challenge, Defend, Attack, Justify\}$. In our framework, a speech act is interpreted either as an action applied to a commitment when the speaker is the debtor, or as an action applied to its content when the speaker is the debtor or the creditor [4]. Formally, a speech act can be defined as follows:

Definition 1. $SA(Ag_1, Ag_2, U) =_{def} Act(Ag_1, SC(Ag_1, Ag_2, p))$
$\qquad | \; Act\text{-}content(Ag_k, SC(Ag_i, Ag_j, p))$

where $i, j \in \{1, 2\}$ and ($k = i$ or $k = j$), *p is the commitment content*. The definiendum $SA(Ag_1, Ag_2, U)$ is defined by the definiens $Act(Ag_1, SC(Ag_1, Ag_2, p))$ as an action performed by the debtor Ag_1 on its commitment. The definiendum is defined by the definiens $Act\text{-}content(Ag_k, SC(Ag_i, Ag_j, p))$ as an action performed by an agent Ag_k (the debtor or the creditor) on the commitment content.

2.2 Argumentation and Social Commitments

An argumentation system essentially includes a logical language *L*, a definition of the argument concept, and a definition of the attack relation between arguments. Several definitions were also proposed to define arguments. In our model, we adopt the following definitions from [15]. Here Γ indicates a knowledge base with deductive closure. $\vdash$ Stands for classical inference and $\equiv$ for logical equivalence.

Definition 2. *An argument is a pair* (H, h) *where h is a formula of L and H a sub-set of* Γ *such that* : *i*) *H is consistent*, *ii*) $H \vdash h$ *and iii*) *H is minimal*, *so no subset of H satisfying both i and ii exists. H is called the support of the argument and h its conclusion. We use the notation*: $H = Support(Ag, h)$ *to indicate that agent Ag has a support H for h.*

Definition 3. *Let* (H_1, h_1), (H_2, h_2) *be two arguments.* (H_1, h_1) *attacks* (H_2, h_2) *iff* $h_1 \equiv \neg h_2$.

In fact, before committing to some fact *h* being true (i.e. before creating a commitment whose content is *h*), the speaker agent must use its argumentation system to build an argument (H, h). On the other side, the addressee agent must use its own argumentation system to select the answer it will give (i.e. to decide about the appropriate manipulation of the content of an existing commitment). For example, an agent Ag_1 accepts the commitment content *h* proposed by another agent if Ag_1 has an argument for *h*. If Ag_1 has an argument neither for *h*, nor for $\neg h$, then it challenges *h*.

In our framework, we distinguish between arguments that an agent has (private arguments) and arguments that this agent used in its conversation (public arguments). Thus, we use the notation: $S = Create\_Support(Ag_1, SC(Ag_1, Ag_2, p))$ to indicate the set of commitments S *created* by agent Ag_1 to support the content of $SC(Ag_1, Ag_2, p)$. This support relation is transitive i.e.:

$(SC(Ag_1, Ag_2, p_2) \in Create\_Support(Ag, SC(Ag_1, Ag_2, p_1))$
$\wedge\ SC(Ag_1, Ag_2, p_1) \in Create\_Support(Ag, SC(Ag_1, Ag_2, p_0)))$
$SC(Ag_1, Ag_2, p_2) \in Create\_Support(Ag, SC(Ag_1, Ag_2, p_0))$

Other details about our commitment and argument approach are described in [4]. Surely, an argumentation system is essential to help agents to act on commitments and on their contents. However, reasoning on other social attitudes should be taken into account in order to explain the agents' decisions. In our persuasion protocol we use the agents' trustworthiness to decide, in some cases, about the acceptance of arguments [6].

3 Conversation Policies for Persuasion Dialogue

3.1 Protocol Form

Our persuasion protocol is specified as a set of conversation policies. In order to be flexible, these policies are defined as initiative/reactive dialogue games. In accordance with our approach, the game moves are considered as actions that agents apply to commitments and to their contents. A dialogue game is specified as follows:

$$Action\_Ag_1 \xrightarrow{Cond} Action\_Ag_2$$

This specification indicates that if an agent Ag_1 performs the action $Action\_Ag_1$, and that the condition *Cond* is satisfied, then the interlocutor Ag_2 will perform the action $Action\_Ag_2$. The condition *Cond* is expressed in terms of the possibility of generating an argument from the agent's argumentation system and in terms of the interlocutor's trustworthiness. We use the notation: $p \triangle Arg\_Sys(Ag_1)$ to denote the fact that a propositional formula p can be generated from the argumentation system of Ag_1 denoted $Arg\_Sys(Ag_1)$. The formula $\neg(p \triangle Arg\_Sys(Ag_1))$ indicates the fact that p cannot be generated from Ag_1's argumentation system. A propositional formula p can be generated from an agent's argumentation system, if this agent can find an argument that supports p. To simplify the formalism, we use the notation $Act'(Ag_x, SC(Ag_i, Ag_j, p))$ to indicate the action that agent Ag_x performs on the commitment $SC(Ag_i, Ag_j, p)$ or on its content ($Act' \in \{Create, Withdraw, Accept, Challenge, Refuse\}$). For the actions related to the argumentation relations, we write $Act\text{-}Arg(Ag_x, [SC(Ag_n, Ag_m, q)], SC(Ag_i, Ag_j, p))$. This notation indicates that Ag_x defends (resp. attacks or justifies) the content of $SC(Ag_i, Ag_j, p)$ by the content of $SC(Ag_n, Ag_m, q)$ ($Act\text{-}Arg \in \{Defend, Attack, Justify\}$). The commitment that is written between square brackets [ ] is the support of the argument. In a general way, we use the notation $Act'(Ag_x, S)$ to indicate the action that Ag_x performs on the set of commitments S or on the contents of these commitments, and the notation $Act\text{-}Arg(Ag_x, [S], SC(Ag_i, Ag_j, p))$

to indicate the argumentation-related action that Ag_x performs on the content of $SC(Ag_i, Ag_j, p)$ using the contents of S as support. We also introduce the notation $Act\text{-}Arg(Ag_x, [S], S')$ to indicate that Ag_x performs an argumentation-related action on the contents of a set of commitments S' using the contents of S as supports.

We distinguish two types of dialogue games: *entry game* and *chaining games*. The entry game allows the two agents to *open* the persuasion dialogue. The chaining games make it possible to construct the conversation. The protocol terminates when the exit conditions are satisfied (Figure 1).

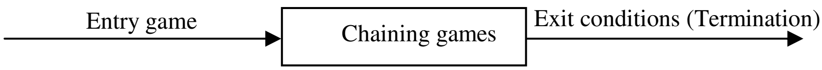
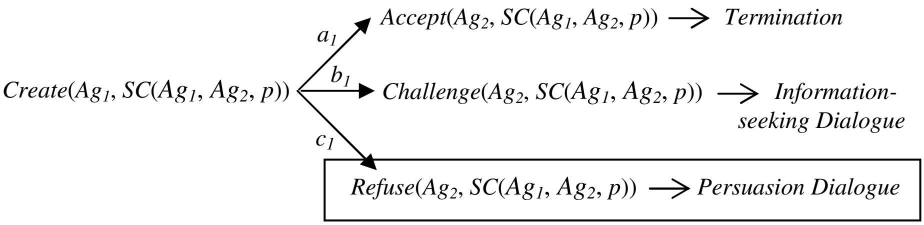


Fig. 1. The general form of the protocol

3.2 Dialogue Games Specification

A Entry Game

The conversational policy that describes the entry conditions in our persuasion protocol about a propositional formula p is described by the entry dialogue game as follows (*Specification 1*):

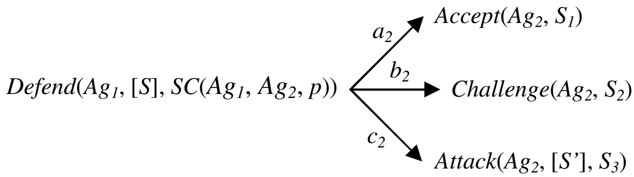


where a_1, b_1 and c_1 are three conditions specified as follows:
$a_1 = p⌂Arg\_Sys(Ag_2)$
$b_1 = \neg(p⌂Arg\_Sys(Ag_2)) \wedge \neg(\neg p⌂Arg\_Sys(Ag_1))$
$c_1 = \neg p⌂Arg\_Sys(Ag_2)$

If Ag_2 has an argument for p then it accepts p (the content of $SC(Ag_1, Ag_2, p)$) and the conversation terminates as soon as it begins (Condition a_1). If Ag_2 has neither an argument for p nor for $\neg p$, then it challenges p and the two agents open an information-seeking dialogue (condition b_1). The persuasion dialogue starts when Ag_2 refuses p because it has an argument against p (condition c_1).

B Defense Game

Once the two agents opened a persuasion dialogue, the initiator must defend its point of view. Thus, it must play a defense game. Our protocol is specified in such a way that the *persuasion dynamics* starts by playing a defense game. We have (*Specification 2*):



where:
$S = \{SC(Ag_1, Ag_2, p_i) / i = 0,\ldots,n\}$, p_i are propositional formulas.

$\bigcup_{i=1}^{3} S_i = S$, $S_i \cap S_j = \emptyset$, $i, j = 1,\ldots,3$ & $i \neq j$

By definition, *Defend*(Ag_1, [S], $SC(Ag_1, Ag_2, p)$) means that Ag_1 creates S in order to defend the content of $SC(Ag_1, Ag_2, p)$. Formally:

$$Defend(Ag_1, [S], SC(Ag_1, Ag_2, p)) =_{def} (Create(Ag_1, S) \wedge S = Create\_Support(Ag_1, SC(Ag_1, Ag_2, p)))$$

We consider this definition as an *assertional* description of the *Defend* action.

This specification indicates that according to the three conditions (a_2, b_2 and c_2), Ag_2 can accept a subset S_1 of S, challenge a subset S_2 and attack a third subset S_3. Sets S_i and S_j are mutually disjoint because Ag_2 cannot, for example, both accept and challenge the same commitment content. *Accept*, *Challenge* and *Attack* a set of commitment contents are defined as follows:

$Accept(Ag_2, S_1) =_{def} (\forall i, SC(Ag_1, Ag_2, p_i) \in S_1 \quad Accept(Ag_2, SC(Ag_1, Ag_2, p_i)))$

$Challenge(Ag_2, S_2) =_{def} (\forall i, SC(Ag_1, Ag_2, p_i) \in S_2 \quad Challenge(Ag_2, SC(Ag_1, Ag_2, p_i)))$

$Attack(Ag_2, [S'], S_3) =_{def} \forall i, SC(Ag_1, Ag_2, p_i) \in S_3 \quad \exists S'_j \subseteq S': Attack(Ag_2, [S'_j], SC(Ag_1, Ag_2, p_i))$

where: $\bigcup_{j=0}^{m} S'_j = S'$. This indication means that any element of S' is used to attack one or more elements of S_3.

The conditions a_2, b_2 and c_2 are specified as follows:

$a_2 = \forall i, SC(Ag_1, Ag_2, p_i) \in S_1 \quad p_i \triangle Arg\_Sys(Ag_2)$

$b_2 = \forall i, SC(Ag_1, Ag_2, p_i) \in S_2 \quad (\neg(p_i \triangle Arg\_Sys(Ag_2)) \wedge \neg(\neg p_i \triangle Arg\_Sys(Ag_2)))$

$c_2 = \forall i, SC(Ag_1, Ag_2, p_i) \in S_3 \quad \exists S'_j \subseteq S', Content(S'_j) = Support(Ag_2, \neg p_i)$

where $Content(S'_j)$ indicates the set of contents of the commitments S'_j.

C Challenge Game

The challenge game is specified as follows (*Specification 3*):

$$Challenge(Ag_1, SC(Ag_2, Ag_1, p)) \xrightarrow{a_3} Justify(Ag_2, [S], SC(Ag_2, Ag_1, p))$$

where the condition a_3 is specified as follows:
$a_3 = (Content(S) = Support(Ag_2, p))$

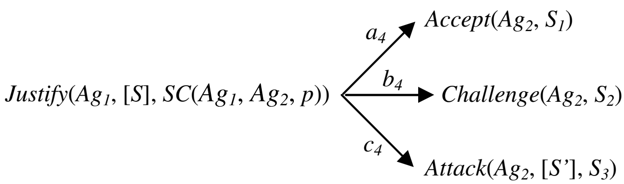
In this game, the condition a_3 is always true. The reason is that in accordance with the commitment semantics, an agent must always be able to defend the commitment it created [5].

D Justification Game

For this game we distinguish two cases:

Case1. $SC(Ag_1, Ag_2, p) \notin S$

In this case, Ag_1 justifies the content of its commitment SC(Ag_1, Ag_2, p) by creating a set of commitments S. As for the Defend action, Ag_2 can accept, challenge and/or attack a subset of S. The specification of this game is as follows (*Specification 4*):



where:

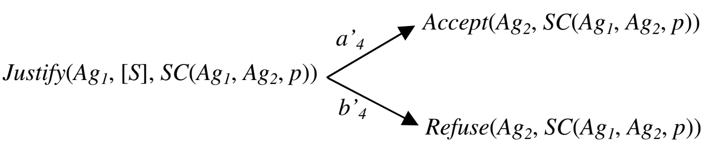
$S=\{SC(Ag_1,Ag_2,p_i)/i=0,...,n\}$, p_i are propositional formulas.

$\cup_{i=1}^{3} S_i = S$, $S_i \cap S_j = \emptyset$, $i, j = 1,...,3 \;\&\; i \neq j$

$a_4 = a_2$, $b_4 = b_2$, $c_4 = c_2$

Case2. $\{SC(Ag_1, Ag_2, p)\} = S$

In this case, the justification game has the following specification (*Specification 5*):



Ag_1 justifies the content of its commitment $SC(Ag_1, Ag_2, p)$ by itself (i.e. by p). This means that p is part of Ag_1's knowledge. Only two moves are possible for Ag_2: 1) *accept* the content of $SC(Ag_1, Ag_2, p)$ if Ag_1 is a trustworthy agent for Ag_2 (a'_4), 2) if not, *refuse* this content (b'_4). Ag_2 cannot attack this content because it does not have an argument against p. The reason is that Ag_1 plays a justification game because Ag_2 played a challenge game.

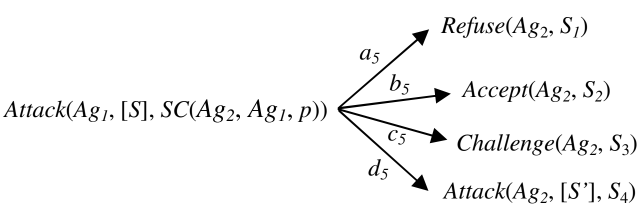
Like the definition of the *Defend* action, we define the *Justify* action as follows:

$$Justify(Ag_1, [S], SC(Ag_1, Ag_2, p)) =_{def} (Create(Ag_1, S) \wedge S = Create\_Support(Ag_1, SC(Ag_1, Ag_2, p)))$$

This means that Ag_1 creates the set S of commitments to support the commitment $SC(Ag_1, Ag_2, p)$.

E Attack Game

The attack game is specified as follows (*Specification 6*):



where:
$S = \{SC(Ag_1, Ag_2, p_i) / i = 0, \ldots, n\}$, p_i are propositional formulas.
$\hbar_{i=1}^{4} S_i = S, Card(S_1) = 1,\ S_i \hbar S_j = \varnothing,\ i, j = 1, \ldots, 4 \ \& \ i \neq j$

Formally, the *Attack* action is defined as follows:

$$Attack(Ag_1, [S], SC(Ag_2, Ag_1, p)) =_{def} (Create(Ag_1, SC(Ag_1, Ag_2, \neg p)) \wedge Create(Ag_1, S) \wedge S = Create\_Support(Ag_1, SC(Ag_1, Ag_2, \neg p)))$$

This means that by attacking $SC(Ag_2, Ag_1, p)$, Ag_1 creates the commitment $SC(Ag_1, Ag_2, \neg p)$ and the set S to support this commitment.

The conditions a_5, b_5, c_5 and d_5 are specified as follows:

$a_5 = \exists i: SC(Ag_2, Ag_1, p_i) \in Create\_Support(Ag_2, SC(Ag_2, Ag_1, \neg q))$
where $S_1 = \{SC(Ag_1, Ag_2, q)\}$
$b_5 = \forall i, SC(Ag_1, Ag_2, p_i) \in S_2 \quad p_i ⌂ Arg\_Sys(Ag_2)$
$c_5 = \forall i, SC(Ag_1, Ag_2, p_i) \in S_3 \quad (\neg(p_i ⌂ Arg\_Sys(Ag_2)) \wedge \neg(\neg p_i ⌂ Arg\_Sys(Ag_2)))$
$d_5 = \forall i, SC(Ag_1, Ag_2, p_i) \in S_4 \quad \exists S'_j \subseteq S': Content(S'_j) = Support(Ag_2, \neg p_i)$
$\wedge \nexists k: SC(Ag_2, Ag_1, p_k) \in Create\_Support(Ag_2, SC(Ag_2, Ag_1, \neg p_i))$

Ag_2 refuses Ag_1's argument if Ag_2 already attacked this argument. In other words, Ag_2 refuses Ag_1's argument if Ag_2 cannot attack this argument since it *already* attacked it, and it cannot accept it or challenge it since it has an argument against this argument. We have only one element in S_1 because we consider a refusal move as an exit condition. The acceptance and the challenge actions of this game are the same as the acceptance and the challenge actions of the defense game. Finally, Ag_2 attacks Ag_1's argument if Ag_2 has an argument against Ag_1's argument, and if Ag_2 did not attack Ag_1's argument before. In d_5, the universal quantifier means that Ag_2 attacks all Ag_1's arguments for which it has an against-argument. The reason is that Ag_2 must act on all commitments created by Ag_1. The temporal aspect (the past) of a_5 and d_5 is implicitly integrated in $Create\_Support(Ag_2, SC(Ag_2, Ag_1, \neg q))$ and $Create\_Support(Ag_2, SC(Ag_2, Ag_1, \neg p_i))$.

F Termination

The protocol terminates either by a final acceptance or by a refusal. There is a final acceptance when Ag_2 accepts the content of the initial commitment $SC(Ag_1, Ag_2, p)$ or when Ag_1 accepts the content of $SC(Ag_2, Ag_1, \neg p)$. Ag_2 accepts the content of $SC(Ag_1, Ag_2, p)$ iff it accepts all the supports of $SC(Ag_2, Ag_1, p)$. Formally:

$$Accept(Ag_2, SC(Ag_1, Ag_2, p)) \Leftrightarrow [\forall i, SC(Ag_1, Ag_2, p_i) \in Create\_Support(Ag_1, SC(Ag_1, Ag_2, p)) \quad Accept(Ag_2, SC(Ag_1, Ag_2, p_i))]$$

The acceptance of the supports of $SC(Ag_1, Ag_2, p)$ by Ag_2 does not mean that they are accepted directly after their creation by Ag_1, but it can be accepted after a number of challenge, justification and attack games. When Ag_2 accepts definitively, then it withdraws all commitments whose content was attacked by Ag_1. Formally:

$$Accept(Ag_2, SC(Ag_1, Ag_2, p)) \quad [\forall i, \forall S, Attack(Ag_1, [S], SC(Ag_2, Ag_1, p_i)) \quad Withdraw(Ag_2, SC(Ag_2, Ag_1, p_i))]$$

On the other hand, Ag_2 refuses the content of $SC(Ag_1, Ag_2, p)$ iff it refuses one of the supports of $SC(Ag_1, Ag_2, p)$. Formally:

$Refuse(Ag_2, SC(Ag_1, Ag_2, p)) \Leftrightarrow$

$$[\exists i: SC(Ag_1, Ag_2, p_i) \in Create\_Support(Ag_1, SC(Ag_1, Ag_2, p)) \wedge Refuse(Ag_2, SC(Ag_1, Ag_2, p_i))]$$

3.3 Protocol Dynamics

The persuasion dynamics is described by the chaining of a finite set of dialogue games: acceptance move, refusal move, defense, challenge, attack and justification games. These games can be combined in a sequential and parallel way (Figure 2).

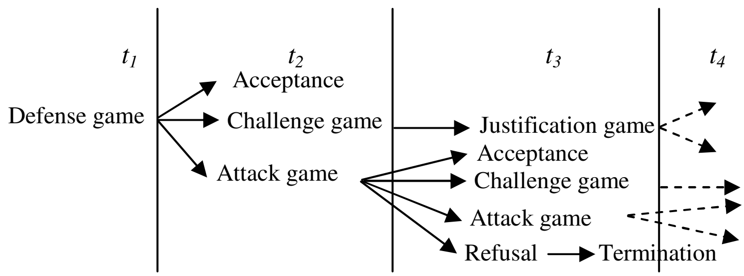


Fig. 2. The persuasion dialogue dynamics

After Ag_1's defense game at moment t_1, Ag_2 can, at moment t_2, accept a part of the arguments presented by Ag_1, challenge another part, and/or attack a third part. These games are played in parallel. At moment t_3, Ag_1 answers the challenge game by playing a justification game and answers the attack game by playing an acceptance move, a challenge game, another attack game, and/or a final refusal move. The persuasion dynamics continues until the exit conditions become satisfied (final acceptance or a refusal). From our specifications, it follows that our protocol plays the role of the dialectical proof theory of the argumentation system.

Indeed, our persuasion protocol can be described by the following BNF grammar:

Persuasion protocol : *Defense game* ~ *Dialogue games*
Dialogue games : (*Acceptance move*
// (*Challenge game* ~ *Justification game* ~ *Dialogue games*)
// (*Attack game* ~ *Dialogue games*))
| *refusal move*

where: "~" is the sequencing symbol, "//" is the possible parallelization symbol. Two games *Game1* and *Game 2* are possibly parallel (i.e. *Game1* // *Game2*) iff an agent can play the two games in parallel or only one game (*Game1* or *Game2*).

3.4 Termination Proof

Theorem. The protocol dynamics always terminates.

Proof. To prove this theorem, we use a tableau method [10]. The idea is to formalize our specifications as tableau rules and then to prove the finiteness of the tableau. Tableau rules are written in such a way that premises appear above conclusions. Using a tableau method means that the specifications are conducted in a top-down fashion. For example, specification 2 (p 3.2) can be expressed by the following rules:

$$R1: \frac{Defend(Ag_1,[S],SC(p))}{Accept(Ag_2,S_1)} \quad R2: \frac{Defend(Ag_1,[S],SC(p))}{Challenge(Ag_2,S_1)}$$

$$R3: \frac{Defend(Ag_1,[S],SC(p))}{Attack(Ag_2,[S'],S_1)}$$

We denote the formulas of our specifications by σ, and we define E the set of σ. We define an ordering $\hbar$ on E and we prove that $\hbar$ has no infinite ascending chains. Intuitively, this relation is to hold between σ_1 and σ_2 if it is possible that σ_1 is an ancestor of σ_2 in some tableau. Before defining this ordering, we introduce some notations: $Act^*(Ag, [S], S')$ with $Act^* \in \{Act', Act\text{-}Arg\}$ is a formula. We notice that formulas in which there is no support $[S]$, can be written as follows: $Act^*(Ag, [\phi], S')$. $\sigma[S] \rightarrow_R \sigma[S']$ indicates that the tableau rule R has the formula $\sigma[S]$ as premise and the formula $\sigma[S']$ as conclusion, with $\sigma[S] = Act^*(Ag, [S], S')$. The size $|S|$ is the number of commitments in S.

Definition 4. *Let* $\sigma[S_i]$ *be a formula and E the set of* $\sigma[S_i]$. *The ordering* $\hbar$ *on E is defined as follows. We have* $\sigma[S_0]\ \hbar\ \sigma[S_1]$ *if:*
$|S_1| < |S_0|$ *or*
For all rules Ri such that $\sigma[S_0] \rightarrow_{R0} \sigma[S_1] \rightarrow_{R1} \sigma[S_2]\ldots \rightarrow_{Rn} \sigma[S_n]$ *we have* $|S_n| = 0$.

Intuitively, in order to prove that a tableau system is finite, we need to prove the following:
1- if $\sigma[S_0] \rightarrow_R \sigma[S_1]$ then $\sigma[S_0]\ \hbar\ \sigma[S_1]$.
2- $\hbar$ has no infinite ascending chains (i.e. the inverse of $\hbar$ is well-founded).

Property 1 reflects the fact that applying tableau rules results in shorter formulas, and property 2 means that this process has a limit. The proof of 1 proceeds by a case analysis on R. Most cases are straightforward. We consider here the case of $R3$. For this rule we have two cases. If $|S_1| < |S_0|$, then $\sigma[S_0]\ \hbar\ \sigma[S_1]$. If $|S_1| \geq |S_0|$, the rules corresponding to the attack specification can be applied. The three first rules are straightforward since $S_2 = \phi$. For the last rule, we have the same situation that $R3$. Suppose that there is no path in the tableau $\sigma[S_0] \rightarrow_{R0} \sigma[S_1] \rightarrow_{R1} \sigma[S_2]\ldots \rightarrow_{Rn} \sigma[S_n]$ such that $|S_n| = 0$. This means that i) the number of arguments that agents have is infinite or that ii) one or several arguments are used several times. However, situation i is not possible because the agents' knowledge bases are finite sets, and situation ii is not allowed in our protocol.

Because the definition of $\hbar$ is based on the size of formulas and since $|S_0| \in N$ ($< \infty$) and $<$ is well-founded in N, it follows that there is no infinite ascending chains of the form $\sigma[S_0]\ \hbar\ \sigma[S_1]\ldots$

4 Implementation

In this section we describe the implementation of the different dialogue games using the *Jack*^{TM} platform [25]. We chose this language for three main reasons:
1- It is an agent-oriented language offering a framework for multi-agent system development. This framework can support different agent models.
2- It is built on top of and fully integrated with the Java programming language. It includes all components of Java and it offers specific extensions to implement agents' behaviors.
3- It supports logical variables and cursors. These features are particularly helpful when querying the state of an agent's beliefs. Their semantics is mid-way between logic programming languages with the addition of type checking Java style and embedded SQL.

4.1 General Architecture

Our system consists of two types of agents: conversational agents and trust model agents. These agents are implemented as *Jack*^{TM} agents, i.e. they inherit from the basic class *Jack*^{TM} *Agent*. Conversational agents are agents that take part in the persuasion dialogue. Trust model agents are agents that can inform an agent about the trustworthiness of another agent.

According to the specification of the justification game, an agent Ag_2 can play an acceptance or a refusal move according to whether it considers that its interlocutor Ag_1 is trustworthy or not. If Ag_1 is unknown for Ag_2, Ag_2 can ask agents that it considers trustworthy for it to offer a trustworthiness assessment of Ag_1. From the received answers, Ag_2 can build a *trustworthiness graph* and measure the trustworthiness of Ag_1. This trustworthiness model is described in detail in [6].

4.2 Implementation of the Dialogue Games

To be able to take part in a persuasion dialogue, agents must possess knowledge bases that contain arguments. In our system, these knowledge bases are implemented as *Jack*^{TM} *beliefsets*. *Beliefsets* are used to maintain an agent's beliefs about the world. These beliefs are represented in a first order logic and tuple-based relational model. The logical consistency of the beliefs contained in a *beliefset* is automatically maintained. The advantage of using *beliefsets* over normal Java data structures is that *beliefsets* have been specifically designed to work within the agent-oriented paradigm.

Our knowledge bases (KBs) contain two types of information: arguments and beliefs. Arguments have the form ([*Support*], *Conclusion*), where *Support* is a set of propositional formulas and *Conclusion* is a propositional formula. Beliefs have the form ([*Belief*], *Belief*) i.e. *Support* and *Conclusion* are identical. The meaning of the propositional formulas (i.e. the ontology) is recorded in a *beliefset* whose access is shared between the two agents.

To open a dialogue game, an agent uses its argumentation system. The argumentation system allows this agent to seek in its knowledge base an argument for a given conclusion or for its negation (*"against argument"*). For example, before creating a commitment $SC(Ag_1, Ag_2, p)$, agent Ag_1 must find an argument for p. This enables us to respect the commitment semantics by making sure that agents can always defend the content of their commitments.

Agent communication is done by sending and receiving messages. These messages are *events* that extend the basic *Jack*^{TM} *event*: *MessageEvent* class. *MessageEvents* represent events that are used to communicate with other agents. Whenever an agent needs to send a message to another agent, this information is packaged and sent as a *MessageEvent*. A *MessageEvent* can be sent using the primitive: *Send*(*Destination*, *Message*). In our protocol, *Message* represents the action that an agent applies to a commitment or to its content, for example: $Create(Ag_1, SC(Ag_1, Ag_2, p))$, etc.

Our dialogue games are implemented as a set of *events* (*MessageEvents*) and *plans*. A plan describes a sequence of actions that an agent can perform when an event occurs. Whenever an event is posted and an agent chooses a task to handle it, the first thing the agent does is to try to find a plan to handle the event. Plans are methods describing what an agent should do when a given event occurs.

Each dialogue game corresponds to an event and a plan. These games are not implemented within the agents' program, but as event classes and plan classes that are external to agents. Thus, each conversational agent can instantiate these classes. An agent Ag_1 starts a dialogue game by generating an event and by sending it to its interlocutor Ag_2. Ag_2 executes the plan corresponding to the received event and answers by generating another event and by sending it to Ag_1. Consequently, the two agents can communicate by using the same protocol since they can instantiate the same classes representing the events and the plans. For example, the event *Event_Attack_Commitment* and the plan *Plan_ev_Attack_commitment* implement the defense game. The architecture of our conversational agents is illustrated in Figure 3.

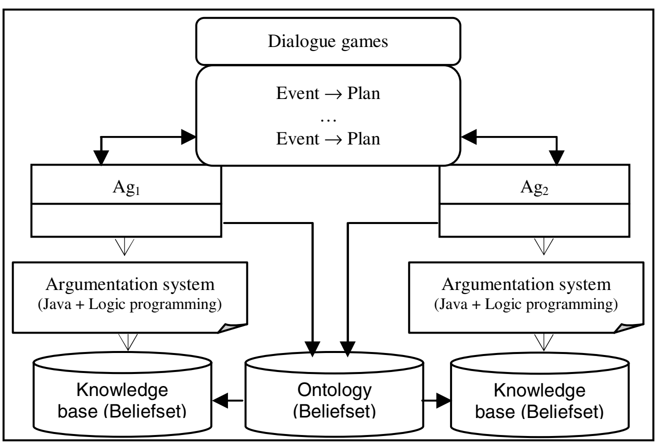


Fig. 3. The architecture of conversational agents

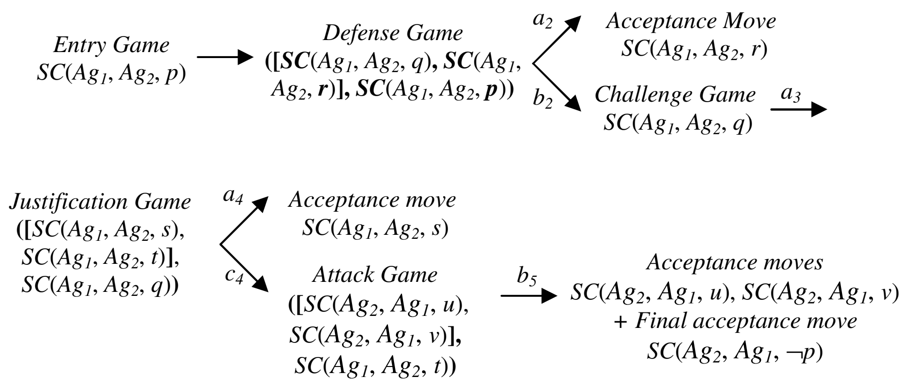
To start the entry game, an agent (initiator) chooses a goal that it tries to achieve. This goal is to persuade its interlocutor that a given propositional formula is true. For this reason, we use a particular event: *BDI Event* (*Belief-Desire-Intention*). BDI events model goal-directed behavior in agents, rather than plan-directed behavior. What is important is the desired outcome, not the method chosen to achieve it. This type of events allows an agent to pursue long term goals.

4.3 Example

In this section we present a simple example dialogue that illustrates some notions presented in this paper.

Ag_1: Newspapers can publish information I (*p*).
Ag_2: I don't agree with you.
Ag_1: They can publish information I because it is not private (*q*), and any public information can be published (*r*).
Ag_2: Why is information I public?
Ag_1: Because it concerns a Minister (*s*), and information concerning a Minister are public (*t*).
Ag_2: Information concerning a Minister is not necessarily public, because information I is about the health of Minister (*u*), and information about the health remains private (*v*).
Ag_1: I accept your argument.

This example was also studied in [2] in a context of strategical considerations for argumentative agents. The letters on the left of the utterances are the propositional formulas that represent the propositional contents. Agent Ag_1's KB contains: $([q, r], p)$, $([s, t], q)$ and $([u], u)$. Agent Ag_2's KB contains: $([\neg t], \neg p)$, $([u, v], \neg t)$, $([u], u)$ and $([v], v)$. The combination of the dialogue games that allows us to describe the persuasion dialogue dynamics is as follows:



Ag_1 creates $SC(Ag_1, Ag_2, p)$ to achieve the goal of persuading Ag_2 that *p* is true. Ag_1 can create this commitment because it has an argument for *p*. Ag_2 refuses $SC(Ag_1, Ag_2, p)$ because it has an argument against *p*. Thus, the entry game is played and the persuasion dialogue is opened. Ag_1 defends $SC(Ag_1, Ag_2, p)$ by creating $SC(Ag_1, Ag_2, q)$ and $SC(Ag_1, Ag_2, r)$. Ag_2 accepts $SC(Ag_1, Ag_2, r)$ because it has an argument for *r* and challenges $SC(Ag_1, Ag_2, q)$ because it has no argument for *q* or against *q*. Ag_1

plays a justification game to justify $SC(Ag_1, Ag_2, q)$ by creating $SC(Ag_1, Ag_2, s)$ and $SC(Ag_1, Ag_2, t)$. Ag_2 accepts the content of $SC(Ag_1, Ag_2, s)$ and attacks the content of $SC(Ag_1, Ag_2, t)$ by creating $SC(Ag_2, Ag_1, u)$ and $SC(Ag_2, Ag_1, v)$. Finally, Ag_1 plays acceptance moves because it has an argument for u and it does not have arguments against v and the dialogue terminates. Indeed, before accepting v, Ag_1 challenges it and Ag_2 defends it by itself (i.e. ([$SC(Ag_2, Ag_1, v)$, $SC(Ag_2, Ag_1, v)$])). Then, Ag_1 accepts this argument because it considers Ag_2 trustworthy. This notion of agent trust and its role as an acceptance criteria of arguments are detailed in [6]. Ag_1 updates its KB by removing the attacked argument and including the new argument. Figure 4 illustrates the screen shot of this example generated by our prototype. In this figure commitments are described only by their contents and the identifiers of the two agents are the two first arguments of the exchanged communicative actions. The contents are specified using a predicate language that the two agents share (the ontology).

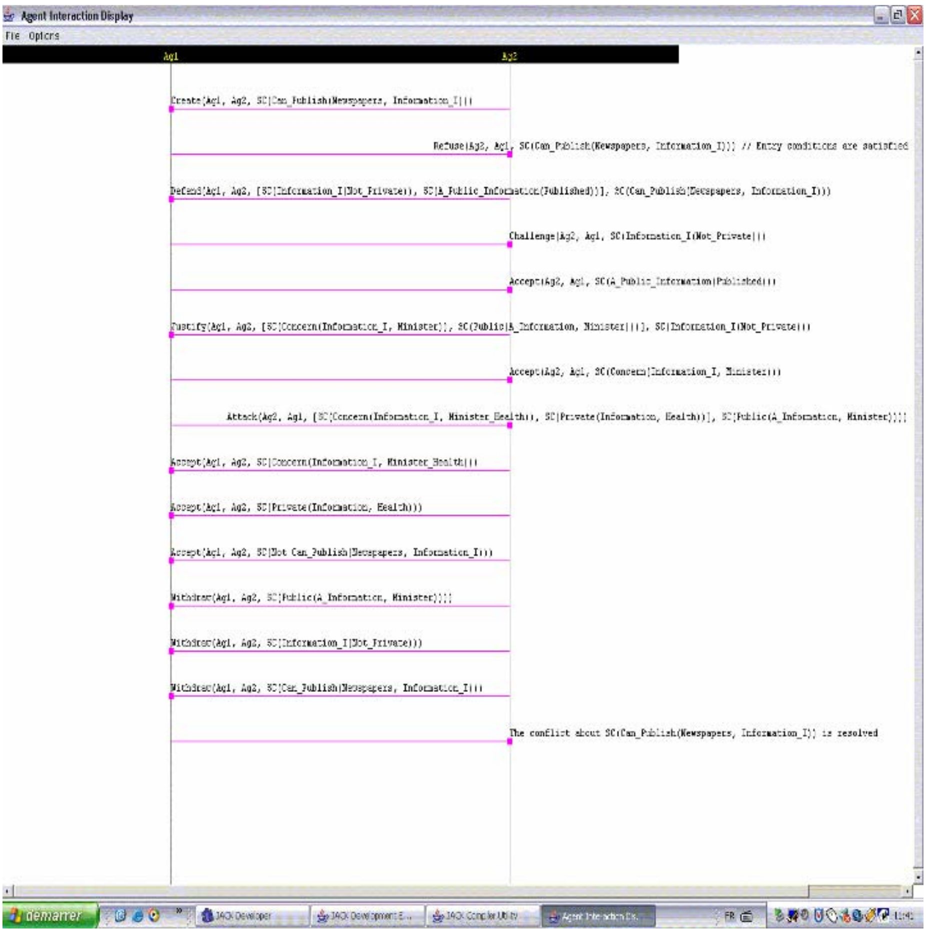


Fig. 4. The example screen shot

5 Related Work

In this section, we compare our protocol with some proposals that have been put forward in two domains: dialogue modeling and commitment based protocols.

1- ***Dialogue modeling.*** In [1] and [22] Amgoud, Parsons and their colleagues studied argumentation-based dialogues. They proposed a set of atomic protocols which can be combined. These protocols are described as a set of dialogue moves using Walton and Krabbe's classification and formal dialectics. In these protocols, agents can argue about the truth of propositions. Agents can communicate both propositional statements and arguments about these statements. These protocols have the advantage of taking into account the capacity of agents to reason as well as their attitudes (confident, careful, etc.). In addition, Prakken [23] proposed a framework for protocols for dynamic disputes, i.e., disputes in which the available information can change during the conversation. This framework is based on a logic of defeasible argumentation and is formulated for dialectical proof theories. Soundness and completeness of these protocols have also been studied. In the same direction, Brewka [7] developed a formal model for argumentation processes that combines nonmonotonic logic with protocols for dispute. Brewka pays more attention to the speech act aspects of disputes and he formalizes dispositional protocols in situation calculus. Such a logical formalization of protocols allows him to define protocols in which the legality of a move can be disputed. Semantically, Amgoud, Parsons, Prakken and Brewkas' approaches use a defeasible logic. Therefore, it is difficult, if not impossible, to formally verify the proposed protocols.

There are many differences between our protocol and the protocols proposed in the domain of dialogue modeling: 1. Our protocol uses not only an argumentative approach, but also a public one. Locutions are formalized not as agents' private attitudes (beliefs, intentions, etc.), but as social commitments. In opposition of private mental attitudes, social commitments can be verified. 2. Our protocol is based on a combination of dialogue games instead of simple dialogue moves. Using our dialogue game specifications enables us to specify the entry and the exit conditions more clearly. In addition, computationally speaking, dialogue games provide a good balance between large protocols that are very rigid and atomic protocols that are very detailed. 3. From a theoretical point of view, Amgoud, Parsons, Prakken and Brewkas' protocols use moves from formal dialectics, whereas our protocol uses actions that agents apply on commitments. These actions capture the speech acts that agents perform when conversing (see Definition 1). The advantage of using these actions is that they enable us to better represent the persuasion dynamics considering that their semantics is defined in an unambiguous way in a temporal and dynamic logic [5]. Specifying protocols in this logic allows us to formally verify these protocols using model checking techniques. 4. Amgoud, Parsons and Prakkens' protocols use only three moves: assertion, acceptance and challenge, whereas our protocol uses not only creation, acceptance, refusal and challenge actions, but also attack and defense actions in an explicit way. These argumentation relations allow us to directly illustrate the concept of dispute in this type of protocols. 5. Amgoud, Parsons, Prakken and Brewka use an acceptance criterion directly related to the argumentation system, whereas we use an acceptance criteria for conversational agents (supports of arguments and trustworthiness). This makes it possible to decrease the computational complexity of the protocol for agent communication.

2- *Commitment-based protocols*. Yolum and Singh [28] developed an approach for specifying protocols in which actions' content is captured through agents' commitments. They provide operations and reasoning rules to capture the evolution of commitments. In a similar way, Fornara and Colombetti [17] proposed a method to define interaction protocols. This method is based on the specification of an interaction diagram (ID) specifying which actions can be performed under given conditions. These approaches allow them to represent the interaction dynamics through the allowed operations. Our protocol is comparable to these protocols because it is also based on commitments. However, it is different in the following respects. The choice of the various operations is explicitly dealt with in our protocol by using argumentation and trustworthiness. In commitment-based protocols, there is no indication about the combination of different protocols. However, this notion is essential in our protocol using dialogue games. Unlike commitment-based protocols, our protocol plays the role of the dialectical proof theory of an argumentation system. This enables us to represent different dialogue types as studied in the philosophy of language. Finally, we provide a termination proof of our protocol whereas this property is not yet studied in classical commitment-based protocols.

6 Conclusion and Future Work

The contribution of this paper is the proposition of a logical language for specifying persuasion protocols between agents using an approach based on commitments and arguments. This language has the advantage of expressing the public elements and the reasoning process that allows agents to choose an action among several possible actions. Because our protocol is defined as a set of conversation policies, this protocol has the characteristic to be more flexible than the traditional protocols such as those used in FIPA-ACL. This flexibility results from the fact that these policies can be combined to produce complete and more complex protocols. We formalized these conversation policies as a set of dialogue games, and we described the persuasion dynamics by the combination of five dialogue games. Another contribution of this paper is the tableau-based termination proof of the protocol. We also described the implementation of this protocol. Finally, we presented an example to illustrate the persuasion dynamics by the combination of different dialogue games.

As an extension of this work, we intend to specify other protocols according to Walton and Krabbe's classification [27] using the same framework. Another interesting direction for future work is verifying these protocols using model checking techniques. The method we are investigating is an automata theoretic approach based on a tableau method [10]. This method can be used to verify the temporal and dynamic aspects of our protocol. Finally, we intend to extend our implementation using ideas from the agent programming language 3APL, namely the concept of cognitive agents. An important characteristic of this language that is interesting for us is its dynamic logic semantics [26] because our protocol is based on an action theory and the semantics of our approach is also based on dynamic logic [5].

Acknowledgements. We'd like to deeply thank the three anonymous reviewers for their valuable comments and suggestions. We'd also like to thank Rance Cleaveland and Girish Bhat for their interesting explanations on the tableau method.

References

1. Amgoud, L., Maudet, N., and Parsons, S. Modelling dialogues using argumentation. In Proc. of 4th Int. Conf. on Multi Agent Systems (2000) 31-38.
2. Amgoud, L., and Maudet, N. Strategical considerations for argumentative agents. In Proc. of 10th Int. Workshop on Non-Monotonic Reasoning (2002) 409-417.
3. Baldoni, M., Baroglio, C., Martelli, A., Patti, V., Schifanella, C. Verifying protocol conformance for logic-based communicating agents. In Proc. of 5th Int. Workshop on Computational Logic in Multi-Agent Systems (2004) 82-97.
4. Bentahar, J., Moulin, B., and Chaib-draa, B. Commitment and argument network: a new formalism for agent communication. In [13] (2004) 146-165.
5. Bentahar, J., Moulin, B., Meyer, J-J. Ch., and Chaib-draa, B. A logical model for commitment and argument network for agent communication (extended abstract). In 3rd Int. J. Conf. on Autonomous Agents and Multi-Agent Systems AAMAS (2004) 792-799.
6. Bentahar, J., Moulin, B., and Chaib-draa, B. Specifying and implementing a persuasion dialogue game using commitment and argument network. In I. Rahwan, P. Moraitis and C. Reed (Eds.), Argumentation in Multi-Agent Systems, LNAI 3366, Springer, (2005). (in press).
7. Brewka, G. Dynamic argument systems: A formal model of argumentation processes based on situation calculus. Journal of Logic and Computation, 11(2) (2001) 257-282.
8. Castelfranchi, C. Commitments: from individual intentions to groups and organizations. In Proc. of Int. Conf. on Multi Agent Systems (1995) 41-48.
9. Chaib-draa, B., and Dignum, F. Trends in agent communication languages. In Computational Intelligence, (18)2 (2002) 89-101.
10. Cleaveland, R. Tableau-based model checking in the propositional mu-calculus. In Acta Informatica, 27(8) (1990) 725-747.
11. Colombetti, M. A commitment-based approach to agent speech acts and conversations. In Proc. of Int. Autonomous Agent Workshop on Conversational Policies (2000) 21-29.
12. Dastani, M., Hulstijn, J., and der Torre, L.V. Negotiation protocols and dialogue games. In Proc. of Belgium/Dutch AI Conference (2000) 13-20.
13. Dignum, F. (Ed.). Advances in Agent Communication. Int. Workshop on Agent Communication Languages. LNAI 2922, Springer, (2004).
14. Dignum, F., and Greaves, M. (Eds.). Issues in agent communication. LNAI 1916, Springer (2000).
15. Elvang-Goransson, M., Fox, J., and Krause, P. Dialectic reasoning with inconsistent information. In Proc. of 9th Conf. on Uncertainty in Artificial Intelligence (1993) 114-121.
16. Endriss, U., Maudet, N., Sadri, F., and Toni, F. Logic_based agent communication protocols. In [13] (2004) 91-107.
17. Fornara, N. and Colombetti, M. Protocol specification using a commitment based ACL. In [13] (2004) 108-127.
18. Greaves, M., Holmback, H., and Bradshaw, J. What is a conversation policy? In [14] (2000) 118-131.
19. Herrestad, H. and Krogh, C. Obligations directed from bearers to counterparties. In Proc. of 5th Int. Conf. on Artificial Intelligence and Law (1995) 210-218.

20. Maudet, N., and Chaib-draa, B. Commitment-based and dialogue-game based protocols, new trends in agent communication languages. In Knowledge Engineering Review, 17(2), Cambridge University Press (2002) 157-179.
21. McBurney, P., and Parsons, S. Games that agents play: A formal framework for dialogues between autonomous agents. In Journal of Logic, Language, and Information, 11(3) (2002) 1-22.
22. Parsons, S., Wooldridge, M., and Amgoud, L. On the outcomes of formal inter-agent dialogues. In Proc. of 2nd Int. J. Conf. on Autonomous Agents and Multi-Agent Systems (2003) 616-623.
23. Prakken, H. Relating protocols for dynamic dispute with logics for defeasible argumentation. In Synthese (127) (2001) 187-219.
24. Singh, M.P. A social semantics for agent communication language. In [14] (2000) 31-45.
25. The Agent Oriented Software Group. *Jack 4.1*. 2004. www.agent-software.com/
26. van Riemsdijk, M.B., de Boer, F.S., and Meyer, J-J. Ch. Dynamic logic for plan revision in intelligent agents. In Proc. of 5th Int. Workshop on Computational Logic in Multi-Agent Systems (2004) 196-211.
27. Walton, D.N., and Krabbe, E.C.W. Commitment in dialogue: basic concepts of interpersonal reasoning. State University of New York Press, NY (1995).
28. Yolum, P. and Singh, M.P. Flexible protocol specification and execution: applying event calculus planning using commitments. In Proc. of 1st Int. J. Conf. on Autonomous Agents and Multi-Agent Systems (2002) 527-534.

Verifying Protocol Conformance for Logic-Based Communicating Agents*

Ma e Ba d . ı, C ı ı a Ba . gı , A be . Ma e ı,
Vı ıa a Pa ı, a d C a dı Schıfa e a

Dipartimento di Informatica — Università degli Studi di Torino,
C.so Svizzera, 185 — I-10149 Torino (Italy)
{baldoni,baroglio,mrt,patti,schi}@di.unito.it

Abstract. Communication plays a fundamental role in multi-agents systems. One of the main issues in the design of agent interaction protocols is the verification that a given protocol implementation is "conformant" w.r.t. the abstract specification of it. In this work we tackle those aspects of the conformance verification issue, that regard the dependence/independence of conformance from the agent private state in the case of logic, individual agents, set in a multi-agent framework. We do this by working on a specific agent programming language, DyLOG, and by focussing on interaction protocol specifications described by AUML sequence diagrams. By showing how AUML sequence diagrams can be translated into regular grammars and, then, by interpreting the problem of conformance as a problem of language inclusion, we describe a method for automatically verifying a form of "structural" conformance; such a process is shown to be decidable and an upper bound of its complexity is given. We also give a set of properties that describes the influence of the agent private information on the conformance of its communication policies to protocol specifications.

1 Introduction

M ı-age e (MAS) f e c ı e he e ge e c e , dı e e ı he a he e e edge ab he d a d ab he age , a e a ı he echa ı ed f ea ı g ab ı . N ı h a dı g he e - ge eı , age c e a e, e ec e a c a c e e f ha ed e ce ; ı e e a ı ı , a , ed b a e f ha ed *interaction protocols.* The de ıg a d ı e e a ı f ı e ac ı c a e c cıa e ı he de e e f a MAS. F ı g he de e e ce , de c ıbed ı [21], f ı e ac ı c e gı ee ı g, dı e e ı d f e a e be e ec ed. The c ı ı e ıf ı g he c ı e c f a *abstract protocol*

* This research is partially supported by MIUR Cofin 2003 "Logic-based development and verification of multi-agent systems (MASSiVE)" national project and by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779.

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 196–212, 2005.

definition . . . he . igi a *requirements*, de i ed f . he a a . i ha e, ha i . h d e b d . Thi . e i ca i . , . f e ca ed *validation test*, i ica d . e b . ea . . f . de -chec i g ech i e . A di e e . . b e i he . . e ha e face i hi . . , hich a . . . . . e if if a gi e *implementation*, hich i a age i e ac i . . ic , e ec a gi e *abstract protocol definition.* Thi . . b e i . . . a *conformance testing.* M . e . e , . i ce he . eci c i e- . e a i . .[1] . . . a . e he age ' . i a e i f . a i . (he age ' a e), e.g. f decidi g hich e a ce . a ic a e, . . e f he e i . ha a i e i : . hich e e d e he age i e . a . a e i e ce he c . . f . a ce . f a i e e a i . . a gi e . . . c . eci ca i . ? I deed, de e di g . . he e f e . . . he age ' *internal state*, di e e e ec i . . c d . cc , . . a . f he bei g c . ec . . . . he . eci ca i . . . Me e i he ca e i hich he . c e . f he c . . e . a i . . ic i . ch ha i i b . d . . d ce c . ec c . . e . a i . . , c . f . a ce i . . i e ced a a b he age ' i e . a . a e (e i di c . . hi i . e i Sec i . 3).

I hi . . e ac e he . b e . f c . f . a ce . e i ca i . f . . . eci c a g age : e . e Age UML (AUML f . h . , ... eci ed i [27]) a he i e ac i . . . . c . eci ca i . a g age a d DyLOG [3, 5] a he c . . e . a i . . ic i e e a i . a g age. I he i e a e . . e ca , ac a , . . d . a f . a ech i e f . . . c . eci ca i . . A . . -e ha . i e i i c de . . i e . a e a . . a a [6, 24], e . i . e [23, 11], e . . a . gic [15, 16] a d UML-ba ed a g age . A he e . . . a a ec . e bei g die da d . . de . i i e a da d e e ged e . The ea . . f . ch . . i g AUML i ha , de i e i e i c . . e e f . a . e a ic (a . . . a f . a e a ic ba ed . . e i . e ca be f d i [10]), hi a g age bea . . . e e e e a ad a age : i i ba ed . . he ide- . ead a d e - . . . UML . a da d, i i i i i e a d ea . ea . , he e a e g a hica edi . . f he ge e a i . . f c de, a d AUML . e e ce diag a . ha e bee ad ed b FIPA . e e e age i e ac i . . . . c . . O he . he ha d, DyLOG i a gic a g age f . . g a . i g age . , ba ed . . ea . i g ab ac i . . a d cha ge i a . . da f a e . . , ha a . . he i c . i . . f a e . f c . . e . a i . . icie , i he . eci ca i . . f a age . The a g age efe . a . e a i ic a . ach, he e . eech ac . a e e e e e ed a a . . ic ac i . . i h ec . di i . . a d e ec . . . he e ec . ' . e a . a e. I a . he . eci ca i . . f *individual agents*, . i a ed i a . i-age c . e , each ha i g a e . . . a ie . f he . . d. The . e . f a dec a a i e a g age i he f beca . e i a . . he . . . f . f . . e ie . f he *specific implementation* i a . aigh f . a d a . I a ic a , a a g age ha e ici e e e e . a d . e he age i e . a . a e, i . ef f . . . i g . hich e e ce ai . . . e ie de e d . . he age . e a . a e . . . he . e a ic . f he . eech ac . . F . i . a ce, i . . . e e f . . h . he ica ea . . i g ab he e ec . f c . . e . a i . . . . he age . e a . a e, i . de . . . d c . . e . a i . . a . hich a e . . . ed . e ec he i e e ed . . . c . , achie i g a he . a e i e . . . e de i ed g a . The DyLOG a g age i b ie i . . d ced i Sec i . 2, f . a h . . gh de c i i i . . f i . ee [5].

[1] In Java, in a logic language, etc.

O g a f hı ı , he , d de hıch c dı ı a DyLOG ı e e a ı ca be dec a ed a beı g c f a a AUML ecı ca ı . T hı aı , e ı ı d ce dı e e e e f ab ac ı . he age e a a e b de ı g h ee deg ee f c f a ce (*agent conformance*, *agent strong conformance*, a d *protocol conformance*). We ı de c ıbe heı e a ı a d, b ı e e ı g he b e f c f a ce e ı ca ı a a b e f ı c ı f a c e -f ee a g age (CFL) ı a eg a a g age, e ı h a e h d f a a ıca e ıf ı g he ge deg ee f c f a ce; a e b d ı c e ı ı a gı e . Whe hı ı d f c f a ce h d , he ı e e ed ıc e ec he ecı ca ı ha e e he a ı a e ec f he eech ac a e, ha e e he age e a a e ı .

A a a b e a ı ab AUML, e a h , h c a ıda ı , c ı ıcı e he ch ıce f hı f a ı beca e ı ac f a f a e a ıc a e ı dı c a ıda e he de ıg ed c he ıg ı a ecı ca ı . The a c ı ıcı e he ch ıce f a e a ı ıc a ach (a a FIPA) beca e a he e e f c a ıda ı hı a ach ha h e e a a . The dı a ı fac ı he e a ı ıc a ach ı d e he dı c f e ıf ı g ha a age ac acc dı g a c ag eed e a ıc , beca e ı ı ıb e ha e acce he age ' ı a e e - a a e [30], a b e a *semantics verification*. S e a h ha e ed a *social approach* age c ıca ı [29], he e c ıca ı e ac ı a ec he cıa a e f he e , a he ha he ı e a a e f he age . The cıa a e ec d he cıa fac , ı e he *permissions* a d he *commitments* f he age , hıch a e c ea ed a d dı ed a g he ı - e ac ı . The cıa a ach e c e he e a ıc e ı ca ı b e b e ı ı g a e f e ab ı hed c ı e be ee he age , ha a e ed a a f he MAS cıa a e. I hı f a e ı ı ıb e f a e he c ec e f b ıc ı e ac ı c ı h e ec he ecı ca ı c ı g f he a a ı ha e ; ch f ca be b aı ed, f ı a ce, b ea f de chec ı g ech ı e [22, 25, 28, 30, 18, 7] (b , e.g., [9]).

Ne e he e , AUML ı beı g ed, e a d e f e , ı MAS de e - e beca e ı ı ı ı ı e f de ıg e ha ha e a bac g d ı UML a d ı he b ec - ıe ed a ach, a d f hı ea ı ha a a ea f he de- e f age e ı he ı d d. M e e , he de e ı g he ı g e age , be ıde e ıf ı g ha he age e ec he cıa c ı e , ı ı ı a d e ıe f heı ı e e a ı a d, ı a ıc a , de a d ıf a d hıch e e ch e ıe de e d he age ' ı e a a e (ı he ca e f c ıca ı , he e a ıc f he eech ac).

2 Specification of Communication in DyLOG

DyLOG [5] ı a hıgh- e e gıc g a ı g a g age f de ı g a ı a age , ba ed a da gıc f ac ı a d e a a ı de he e da - ı ıe a e ed f e e e ı g ac ı a e a be ıef ha a e ı he age ' e a a e. I acc b h f a ıc a d c e ac ı , ced e , f

. ecıf ı. g he age. beha ı. , . A DyLOG age. ca. be . , . ıded ı h a *communication kit* ha . ecı e ı. c. . . ıca ı. e beha ı. , [3], de . ed ı. e. . . f ı. e-ac ı. . , . . c. . , ı.e. c. . , e. a ı. . . ıcıe ha b ı d . . FIPA- ı e. eech ac . . The c. . . ıca ı. . he. , ı a h. . ge. e. . c. . . . e. . f he ge. e. a age. he. , ; ı. a. ıc a. , b. h he c. . . e. a ı. . a . ıcıe , ha g ıde he age. ' c. . . ıca ı. e beha ı. , , a. d he . he. . ıcıe , de . ı. g he age. ' beha ı. , , a. e . e . e e. ed b . , ced . e de . ı ı. . (. e . e e. ed b *axiom schema*). DyLOG age. . ca. . ea . . ab. heı. c. . . ıca ı. e beha ı. , a. . e. ı. g . e. ıe ı e gı e. a . , . c. a. d a. e . f de ıde. a a, ı he. e a c. . . e. a ı. . , ha . e ec . he . , . c. , hıch a. . . a ı . e he de ı. ed c. . dı ı. . . . . he . a . e. a . a e? .

2.1 The DyLOG Language in Brief

I. DyLOG a . . ıc ac ı. . . a. e eı he. . , d ac ı. . . , a ec ı. g he . , d, . , . e. a ac ı. . . , ı.e. . e. . ı. g . , c. . . . ıca ı. e ac ı. . . hıch . . a ec he age. be ıef . The . e . f a . . ıc ac ı. . . c. . . ı . . f he . e $\mathcal{A}$. f he . , d ac ı. . . , he . e $\mathcal{C}$. f c. . . . ıca ı. e ac . , a. d he . e $\mathcal{S}$. f . e. . ı. g ac ı. . . . F. , each a . . ıc ac ı. . a a. d age. ag_i e ı. . , d ce he . . da ı ıe $[a^{ag_i}]$ a. d $\langle a^{ag_i} \rangle$. $[a^{ag_i}]\alpha$. ea. . ha α h. d af e. e e. e ec ı. . . f ac ı. . a b age. ag_i; $\langle a^{ag_i} \rangle \alpha$. ea. . ha he. e ı a . . . ıb e e ec ı. . . f a (b ag_i) af e. hıch α h. d . We . e he . . da ı □ . de. . e *laws*, ı.e. f . , a ha h. d a a . (af e. e e. ac ı. . . e e. ce). O . f . , a ı a ı. . . f c. . e ac ı. . . d. a . c. . . ıde. ab f. . d a ıc . gıc f. , he de . ı ı. . . f ac ı. . . e. a . , ı e . e e. ce, e a. d . . . -de e. . ı ı ıc ch. ıce. H. e. e. , dı e. e. ha. [26], e . efe. . a *Prolog-like* a. adıg : . , ced . e a. e de . ed b . ea. . . f (. . . ıb . ec . , ı e) P. . . g- ı e c a . e . F. , each . , ced . e p, he a. g age c. . aı. . a. . he . ı e. . a a. d e ı e. ıa . . da ı ıe $[p]$ a. d $\langle p \rangle$. The . e. a . a e . f a. age. ı de c. ıbed b a c. . . ı e. . e . f *belief formulas* (e ca ı *belief state*). We . e he . . da . . e. a . . $\mathcal{B}^{ag_i}$ . . . de he be ıef . f age. ag_i. The . . da ı $\mathcal{M}^{ag_i}$ ı de . ed a he d a . f $\mathcal{B}^{ag_i}$ a. d . ea. . ha ag_i c. . . ıde. . φ . . . ıb e. A . e. a . a e c. . aı. . ha ag_i (dı)be ıe. e ab. he . , d a. d ab. he . he. age. . (e ed be ıef a. e . eeded f. , . ea . . ı. g . . h. . he. age. . be ıef ca. be a ec ed b c. . . . ıca ı. e ac ı. . .). F. , . a ı ı a c. . e e a. d c. . . ı e. . e . f . a. 1 a. d 2 be ıef e. . , he. e a *belief fluent* F ı a be ıef f. , . a $\mathcal{B}^{ag_i} L$. , ı. . ega ı. . . L de. . e a *belief argument*, ı.e. a *fluent literal* l (f . , $\neg f$) . , a be ıef e. . f . a. 1 ($\mathcal{B}l$. , $\neg \mathcal{B}l$).

A he . . da ı ıe . f he a. g age a. e . . , . a ; □ ı . e e ı e a. d . a. . ı ı e, ı. ı. e. ac ı. . ı h ac ı. . . . da ı ıe ı . , ed b □$\varphi \supset [a^{ag_i}]\varphi$. The e ı e ıc . . da ı $\mathcal{B}^{ag_i}$ ı . e. ıa , . a. . ı ı e a. d e c ıdea. . A . . . - . . . . . . ıc. . ı. . . he e. . ı e. c . , b e ı gı e. , hıch c. . . ı . ı. . a ı ı. g a. . ı. . . ab. e. . af e. he e ec ı. . . f ac ı. . . e e. ce , ba ed . . a. abd c ı e f. a e . , .

2.2 The Communication Kit in Brief

The *behavior* . f a. age. ag_i ı . ecı ed b a d. . aı. de c. ı ı. . , hıch ı. c de , be ıde a . ecı ca ı. . . f he age. *belief state*: (ı) *action and precondition laws* f . , de c. ıbı. g he a . . ıc . , d ac ı. . . ı e. . . f heı. . ec. . dı ı. . . a. d heı.

a ec . . . he age . ' . e. a . a e, (ii) *sensing axioms* f . de c ibi g a . . ic . e. . i g ac i . . , (iii) *procedure axioms* f . de c ibi g c. . e beha i . . , (i) a *communication kit* ha de c ibe he age c. . . . ica i e beha i . b . ea . . f f . he a i . . a d a . . f he i d . e i . ed ab . e. I fac a *communication kit* c . . i . . f (i') a . e . f ac i . . a d . ec . di i . . a . . . de i g a . ede . ed . e . f . i i i e . eech ac . he age ca e f . . / ec g i e (ii') a . e . f. e. . i g a i . . f . de . i g . ecia . e . . i g ac i . . f . ge i g . e i f . . a i . . b e - e . a c. . . . ica i . . (iii') a . e . f . . ced . e a i . . f . . . ecif i g i e ac i . . . . . c . , hich ca be . ee a a ib a . f c . . e. a i . . . icie he age ca f . he e gagi g a c . . . e. a i . . . i h . he . .

Interaction Protocols a e . e . e e ed a . . ced . e ha b i d . . i di id- a . eech ac . a d . ecif c . . e . a i . . . icie f . g idi g he age c. . . - . ica i e beha i . . The a e e . e . ed b *axiom schema* . f f . . :

$$\langle p_0\rangle\varphi \subset \langle p_1; p_2; \ldots; p_m\rangle\varphi \tag{1}$$

p_0 i a . . ced . e . a e a d he p_i' $(i = 1,\ldots,m)$ a e ei he . . ced . e . a e , a . . ic ac i . . , . . e ac i . . (ac i . . . f he f . . $Fs?$, he e Fs i a be ief e c. . . c i . .); i . i i e , he ? . . e a . . c . . e . . d . chec i g he . a e . f a e. c. . . c i . . i he c . . e . a e hi e he ; i he . e e ci g . e a . . . f d . a ic . gic. Si ce each age ha a . b ec i e e ce i . . . f he c. . . . . ica i . . i h . he age . . , gi e a . . . c. . eci ca i . . e e ec . ha e a . a . . ced . a . e . e e. a i . . a he . . . ib e . . e i he c. . . e . a i . . .

The a i . . . che a . ed . de . e . . ced . e ha e he f . . . f *inclusion axioms*, hich e e he . b ec . f . e i . . . . [4,2], i hich he c a . . f . i- . . da . gic , cha ac e i ed b a i . . ha ha e he ge e a f . . $\langle s_1\rangle\ldots\langle s_m\rangle\varphi \subset \langle t_1\rangle\ldots\langle t_n\rangle\varphi$, he e $\langle s_i\rangle$ a d $\langle t_i\rangle$ a e . . da . e a . . . , ha bee a a . ed. The e a i . . ha e i e e i g c. . . . a i . a . . . e ie beca . e he ca be c. . . id- e ed a *rewriting rules*. I [14] hi i d . f a i . . . i . . ed f . de . i g *grammar logics* a d . . . e . e a i . . be ee f . . a a g age a d . ch . gic a e a a . ed.

A . eech ac *c* i $\mathcal{C}$ ha f . . speech_act(ag_s, ag_r, l), he e ag_s (e de) a d ag_r (ecei e) a e age . a d l i he . e age c. . e . . E ec . a d . ec . di i . . a e . . de ed b a . e . f e ec a d . ec . di i . . a . . I . a ic a , *effects* . . ag_i' be ief . a e . f a ac i . . c a e e . e . ed b *action laws* . f f . . :

$$\Box(\mathcal{B}^{ag_i}L_1 \wedge \ldots \wedge \mathcal{B}^{ag_i}L_n \supset [c^{ag_i}]\mathcal{B}^{ag_i}L_0) \tag{2}$$

$$\Box(\mathcal{M}^{ag_i}L_1 \wedge \ldots \wedge \mathcal{M}^{ag_i}L_n \supset [c^{ag_i}]\mathcal{M}^{ag_i}L_0) \tag{3}$$

La (2) . ea . ha , af e a . e e ce . f ac i . . ($\Box$), if he . e . f e . i e a . $L_1 \wedge \ldots \wedge L_n$ (e . e e i g he . ec . di i . . . f he ac i . . c) i be ie ed b ag_i he , af e he e ec i . . f c, L_0 (he e ec . f c) i a . . be ie ed b ag_i. N ice ha . . e . e e a i . . f . eech ac . . . de . . . he d . a ic . f he . e a . a e . f he age he e a ei e e i g. E ec i g a . eech ac a ca . e a age . ha e . e be ief (i i . . e a . a e), ha a e a . . . i . . . . ha he . he . be ie e b he age ca . . be . e ha he . he . ac a ha e h . e be ief . La (3) . a e ha he he . ec . di i . . . f c a e . . . . . . ag_i, af e he e ec i . . f c, i i c. . ide . . . . . a . . i . e ec . . *Precondition laws* ecif e a c dı ha a e a ac ı ı $\mathcal{C}$ e ec ab e ı a a e. The ha e f :

$$\Box(\mathcal{B}^{ag_i} L_1 \wedge \ldots \wedge \mathcal{B}^{ag_i} L_n \supset \langle c^{ag_i} \rangle \top) \qquad (4)$$

ag_i ca e ec e c he ı ec dı ı e a e ı ag_i' be ief a e.

Get message actions a e f a ı ed a e ı g ac ı , ı.e. edge - d cı g ac ı h e c e ca be edıc ed bef e he e ec ı . I fac , f he e ec ı e f he ı dı ıd a age , e ec ı g a e age c e d e f a e e a ı , h ı ı a a hı f ı a a ecıa ca e f e ı g. A get_message ac ı ı de ed b he *inclusion axiom schema*:

$$[\mathsf{get\_message}(ag_i, ag_j, l)]\varphi \equiv [\bigcup_{\mathsf{speech\_act} \in \mathcal{C}_{\mathsf{get\_message}}} \mathsf{speech\_act}(ag_j, ag_i, l)]\varphi \qquad (5)$$

I ı ı e , $\mathcal{C}_{\mathsf{get\_message}}$ ı a ı e e f eech ac , hıch a e a he ıb e c ıca ı ha ag_i c d e ec f ag_j ı he c e f a gı e c - e a ı . He ce, he ı f a ı ha ca be b aı ed ı ca c a ed b ı g a he e ec f he eech ac ı $\mathcal{C}_{\mathsf{get\_message}}$ ag_i' e a a e.

A a a e c e , ı he e a ı a ı e e a ı f he a g age, (1) ı ha d ed a a e ı ı g e. F a dec a a ı e e a ıc ı f ıe , he e ı a a ı che a f he gıc, he ce ı f . I ı ı e , he e f f a f ı d (2), (3) a d (4) de e he he , hı e h e f f (1) a d (5) de e he cha ac e ı ıc f he ı- da gıc ı hıch he f a a e ı e e ed.

Example 1. The f ı g ced e a ı e e e a ı e e a ı f he c ı Fıg. 1 a he c e a ı ıc ha he c e age (*cus*) e f ı e ac ı g ı h he e ıce ıde (*sp*). A ı ı e e ı g he e b c f . Sı ce he AUML c c aı e , he c e a d he ıde , he ı e e a ı c aı ıe a e b f b e ı e e he ıe f he c e (get_cinema_ticket$_C$). Sı ı a f he b c f e ı g ı f a ı : yes_no_query$_Q$ ı e e he e f he e ıe a d yes_no_query$_I$ he e f he e de [2].

(a) $\langle \mathsf{get\_cinema\_ticket}_C(cus, sp, movie) \rangle \varphi \subset$
$\langle \mathsf{yes\_no\_query}_Q(cus, sp, available(movie));$
$\mathcal{B}^{cus} available(movie)?; \mathsf{get\_info}(cus, sp, cinema(c));$
$\mathsf{yes\_no\_query}_I(cus, sp, pay\_by(credit\_card));$
$\mathcal{B}^{cus} pay\_by(credit\_card)?; \mathsf{inform}(cus, sp, cc\_number);$
$\mathsf{get\_info}(cus, sp, booked(movie)) \rangle \varphi$

(b) $\langle \mathsf{get\_cinema\_ticket}_C(cus, sp, movie) \rangle \varphi \subset$
$\langle \mathsf{yes\_no\_query}_Q(cus, sp, available(movie)); \mathcal{B}^{cus} available(movie)?\,;$
$\mathsf{get\_info}(cus, sp, cinema(c));$
$\mathsf{yes\_no\_query}_I(cus, sp, pay\_by(credit\_card)); \neg\mathcal{B}^{cus} pay\_by(credit\_card)? \rangle \varphi$

---

[2] The subscripts next to the protocols names are a writing convention for representing the role that the agent plays: Q stands for *querier*, I stands for *informer*, C for *customer*.

(c) $\langle \mathsf{get\_cinema\_ticket}_C(cus, sp, movie)\rangle\varphi \subset$
$\langle \mathsf{yes\_no\_query}_Q(cus, sp, available(movie)); \neg\mathcal{B}^{cus} available(movie)?\rangle\varphi$
(d) $[\mathsf{get\_info}(cus, sp, Fluent)]\varphi \equiv [\mathsf{inform}(sp, cus, Fluent)]\varphi$

P. . c. $\mathsf{get\_cinema\_ticket}_C$ . . . a f. . . : age. *cus* begı. . he ı. e. ac ı. . . Af e. chec ı. g ıf he e e ed . . ıe ı a aı ab e b he $\mathsf{yes\_no\_query}_Q$ . . . -c. , ı aı . f. a. ı. f. . a ı. . ($\mathsf{get\_info}$) f. . he . . ıde. (sp) ab. hıch cı e a. h. ı . The. , he . . ıde. a . f. a a. e. b c. edı ca. d b . -ı. g he $\mathsf{yes\_no\_query}_I$ . . . c. . If he a. . e. ıf . . ı ı. e *cus* c. . . ıca e he c. edı ca. d . . be. a. d he c. . . a ı. . f he ıc e b. . ı. g ı . e . . ed . ı , . he. ı e c a . e (b) ı . e ec ed, e. dı. g he c. . . e. a ı. . . C a . e (c) ac e he ca e ı. hıch he . . ıe ı . . a aı ab e; c a . e (d) de c. ıbe $\mathsf{get\_info}$, hıch ı a $\mathsf{get\_message}$ ac ı. . . I. he f. . ı. g he $\mathsf{get\_answer}$ a. d $\mathsf{get\_start}$ de . ı ı. . . a. e ı. . a. ce . f a ı. . che a (5): he ıgh ha. d. ıde . f $\mathsf{get\_answer}$ e . e e. . a he . . . ıb e a. . e. e ec ed b *cus* f. . . *sp* ab. *Fluent*, d . ı. g a c. . . e. a ı. . . ed b $\mathsf{yes\_no\_query}_Q$.

(e) $\langle \mathsf{yes\_no\_query}_Q(cus, sp, Fluent)\rangle\varphi \subset$
$\langle \mathsf{queryIf}(cus, sp, Fluent); \mathsf{get\_answer}(cus, sp, Fluent)\rangle\varphi$
(f) $[\mathsf{get\_answer}(cus, sp, Fluent)]\varphi \equiv [\mathsf{inform}(sp, cus, Fluent) \cup$
$\mathsf{inform}(sp, cus, \neg Fluent) \cup \mathsf{refuseInform}(sp, cus, Fluent)]\varphi$
(g) $\langle \mathsf{yes\_no\_query}_I(cus, sp, Fluent)\rangle\varphi \subset \langle \mathsf{get\_start}(cus, sp, Fluent);$
$\mathcal{B}^{cus} Fluent?; \mathsf{inform}(cus, sp, Fluent)\rangle\varphi$
(h) $\langle \mathsf{yes\_no\_query}_I(cus, sp, Fluent)\rangle\varphi \subset \langle \mathsf{get\_start}(cus, sp, Fluent);$
$\mathcal{B}^{cus} \neg Fluent?; \mathsf{inform}(cus, sp, \neg Fluent)\rangle\varphi$
(ı) $\langle \mathsf{yes\_no\_query}_I(cus, sp, Fluent)\rangle\varphi \subset \langle \mathsf{get\_start}(cus, sp, Fluent);$
$\mathcal{U}^{cus} Fluent?; \mathsf{refuseInform}(cus, sp, Fluent)\rangle\varphi$
( ) $[\mathsf{get\_start}(cus, sp, Fluent)]\varphi \equiv [\mathsf{queryIf}(sp, cus, Fluent)]\varphi$

Gı e. a . e $\Pi_{\mathcal{C}}$. f ac ı. . a. d . ec. . dı ı. . a . de . ı. g he age. ag_i' . ı ı ı. e . eech ac . , a . e $\Pi_{\mathcal{S}get}$. f a ı. . f. he ece ı. . f . e . age , a. d a . e $\Pi_{\mathcal{CP}}$, . f . . ced . e a ı. . f. . ecıf ı. g c. . . e. a ı. . . . . c. . , e de. . e b CKit^{ag_i} he *communication kit* . f a. age. ag_i, ha ı he . ı e $(\Pi_{\mathcal{C}}, \Pi_{\mathcal{CP}}, \Pi_{\mathcal{S}get})$.

A *domain description* (DD) f . age. ag_i, ı a . ı e $(\Pi, \mathsf{CKit}^{ag_i}, S_0)$, he. e CKit^{ag_i} ı ag_i' c. . . ıca ı. . ı , S_0 ı he ı. ı ıa . e . f ag_i' be ıef e. . , a. d Π ı a e $(\Pi_{\mathcal{A}}, \Pi_{\mathcal{S}}, \Pi_{\mathcal{P}})$, he. e $\Pi_{\mathcal{A}}$ ı he . e . f ag_i' . . d ac ı. . a. d . ec. . dı ı. . a . , $\Pi_{\mathcal{S}}$ ı a . e . f a ı. . f. ag_i' . e. ı. g ac ı. . . , $\Pi_{\mathcal{P}}$ a . e . f a ı. . ha de . e he c. . e . . . -c. . . ıca ı. e beha ı. . . f he age. . .

F. . a DD ı h he . ecı ca ı. . . f he ı. e. ac ı. . . . . c. . a. d . f he e e. a. . eech ac . , a *planning* ac ı ı ca. be ıgge. ed b *existential queries* . f f. . $\langle p_1\rangle\langle p_2\rangle \ldots \langle p_m\rangle Fs$, he. e each p_k $(k = 1, \ldots, m)$ a be a. a . ıc . c. . e ac ı. . (a . ı ı ı. e . eech ac . . a. ı. e. ac ı. . . . . c.), e ec ed b . . age. , . . a. e e. a[3] . eech ac , ha be. . g . CKit^{ag_i}. I. [3] e

[3] By the word *external* we denote a speech act in which our agent plays the role of the receiver.

e e ed a g a -dı ec ed f ced e f he a g age ba ed ega ı a faı e (NAF) hıch a e f f $\langle p_1\rangle\langle p_2\rangle \ldots \langle p_m\rangle Fs$ be ed f a gı e d aı de c ı ı a d e a a e a ac ı e e ce. A e f he f $\langle p_1; p_2; \ldots ; p_n\rangle Fs,$ he e p_i, $1 \le i \le n$ $(n \ge 0)$, ı eı he a d ac ı , a e ı g ac ı , a ced e a e, a e , cceed ıf ı ı ıb e e ec e $p_1, p_2, \ldots , p_n$ (ı he de) a ı g f he c e a e, ı ch a a ha Fs h d a he e ı g a e. I ge e a , e ı eed e ab ı h ıf a g a h d a a gı e a e. He ce, e ı ı e:

$$a_1, \ldots , a_m \vdash \langle p_1; p_2; \ldots ; p_n\rangle Fs \quad \text{ı h a e (.a.) } \sigma$$

ea ha he e $\langle p_1; p_2; \ldots ; p_n\rangle Fs$, ı.e. $\langle p_1\rangle\langle p_2\rangle \ldots \langle p_n\rangle Fs$, ca be ed f he DD $(\Pi, \mathsf{CKit}^{ag_i}, S_0)$ a he a e $a_1, \ldots , a_m$ ı h a e σ, he e σ ı a ac ı e e ce $a_1, \ldots , a_m, \ldots a_{m+k}$ hıch e e e he a e e ı g b e ec ı g $p_1, \ldots , p_n$ ı he c e a e $a_1, \ldots , a_m$. ε de e he ı ı ıa a e.

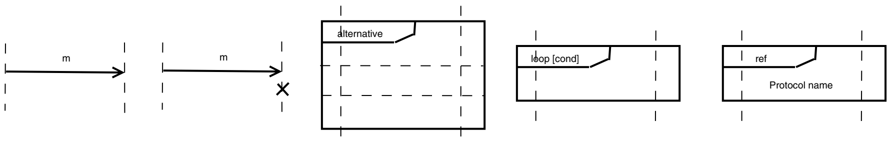
3 Protocol Conformance

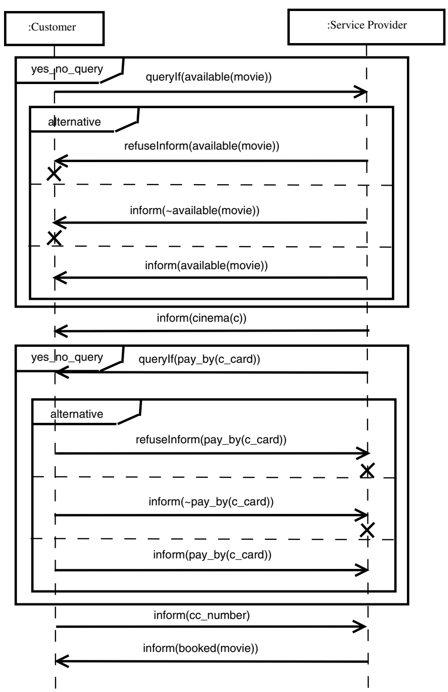
I AUML a c ı ecı ed b ea f e e ce dıag a [27], hıch de he ı e ac ı a g he a ıcı a a e age e cha ge , a a ged ı ı e e e ce . The e ıca (ı e) dı e ı ecı e he a e age ı e (e ec ed), he h ı a dı e ı e e e he a ıcı a a d heı e . The c e a [17], e ıche he e f ıb e e a f he a g age; a ıc a ı e e ı g ı he ıbı ı f e e e ı g , ca b - c , a d e ı ı . Ge e a ea ı g, gı e a c ı e e a ı ı d be ıce ha e a ea f a a ıca e ıf ı g ı *conformance* he de ı ed AUML ecı ca ı . The ech ı e ha e f c ı ı ı g hı b e ı a b e f f a a g age ı c ı . T hı aı , gı e a e e ce dıag a , e de e a f a g a a hıch ge e a e a a g age, ha ı he e f a he c e a ı a ed b he dıag a ı e f. The a g ı h ed hı e ı de c ıbed ı Sec ı 3.1. O he he ha d, gı e a D LOG ı e e a ı f a c , e de e a a g age ha ı c a ed he e ı b aı ed e: ıf he a g age b aı ed f he ı e e a ı ı ı c ded ı he e b aı ed f he e e ce dıag a e c c de ha a e f c f a ce h d . We, ac a , de e h ee deg ee f c f a ce (*agent conformance*, *agent strong conformance*, a d *protocol conformance*), cha ac e - ı ed b dı e e e e f ab ac ı f he age ı a e e a a e, hıch c e d dı e e a f e ac ı g he a g age f he ı e e a ı . The e de ı ı a de e hıch a f a c ı e e a ı he ecı ca ı a d de c ıbe ı a d a a h he ı e e - a ı ca be e ıched ı h e ec he ecı ca ı , ı h c ı ı g he c f a ce. S ch a e ıch e ı ı a he ı g gıc age , ha hı ıca ed f f ea ı g.

3.1 Turning an AUML Sequence Diagram into a Linear Grammar

I he f ı g e h h ı ı ıbe a a e a AUML e e ce dıag a , a de ed ı [17], ı a g a a . U ı g he a ı f [20], a g a a G ı a e (V, T, P, S), he e V ı a e f a ıab e , T a e f e ı a , P f d c ı e , a d S ı he a b . B $L(G)$ e ı de e he a g age ge e a ed b g a a G, ı.e. he e f e e ce ı T^* ha a e ge e a ed a ı g f S, b a ı g e ı P.

O he ıde f e e ce dıag a e f c he e a ed ecıf FIPA c , hıch a e: e age, a e a ı e, , e ı , a d efe e ce a b- c (ee f Fıg. 1).





$$
\begin{aligned}
Q_0 &\longrightarrow \mathsf{queryIf}(cus, sp, available(movie))\ Q_1\\
Q_1 &\longrightarrow Q_2 \mid Q_4 \mid Q_6\\
Q_2 &\longrightarrow \mathsf{refuseInform}(cus, sp, available(movie)) Q_3\\
Q_3 &\longrightarrow \varepsilon\\
Q_4 &\longrightarrow \mathsf{inform}(sp, cus, \neg available(movie)) Q_5\\
Q_5 &\longrightarrow \varepsilon\\
Q_6 &\longrightarrow \mathsf{inform}(sp, cus, available(movie))\ Q_7\\
Q_7 &\longrightarrow Q_8\\
Q_8 &\longrightarrow \mathsf{inform}(sp, cus, cinema(c))\ Q_9\\
Q_9 &\longrightarrow \mathsf{queryIf}(sp, cus, pay\_by(c\_card))\ Q_{10}\\
Q_{10} &\longrightarrow Q_{11} \mid Q_{13} \mid Q_{15}\\
Q_{11} &\longrightarrow \mathsf{refuseInform}(cus, sp, pay\_by(c\_card)) Q_{12}\\
Q_{12} &\longrightarrow \varepsilon\\
Q_{13} &\longrightarrow \mathsf{inform}(cus, sp, \neg pay\_by(c\_card)) Q_{14}\\
Q_{14} &\longrightarrow \varepsilon\\
Q_{15} &\longrightarrow \mathsf{inform}(cus, sp, pay\_by(c\_card))\ Q_{16}\\
Q_{16} &\longrightarrow Q_{17}\\
Q_{17} &\longrightarrow \mathsf{inform}(cus, sp, cc\_number)\ Q_{18}\\
Q_{18} &\longrightarrow \mathsf{inform}(sp, cus, booked(movie)) Q_{19}\\
Q_{19} &\longrightarrow \varepsilon
\end{aligned}
$$

Fig. 1. On top a set of AUML operators is shown. Below, on the left the sequence diagram, representing the interaction protocol between a cinema booking service and a customer, is reported with its corresponding production rules

Algorithm 1 (Generating $G_{p_{AUML}}$) The g a a c c ı ı d e ı he f ı g a . We ı de e b he a ıab e *last* he ece c ea ed a ıab e. I ı ıa T a d P a e e . De e he a b a Q_0, ı ı ıa ı e *last* Q_0 a d $V := \{Q_0\}$, he , e a he a a ı e de c ıbed b ca e he eaf e , acc dı g he e e ce gı e b he AUML dıag a :

– gı e a *message arrow*, abe ed b m, c ea e a e a ıab e Q_{new}, $V := V \cup \{Q_{new}\}$, $T := T \cup \{\mathsf{m}\}$, $P := P \cup \{last \longrightarrow \mathsf{m}Q_{new}\}$, a , e $last := Q_{new}$;
– gı e a *exit operator*, add P a d c ı $last \longrightarrow \epsilon$, $last := \perp$ (de- ed);
– gı e a *alternative* e a ı h n b a che , a each b a ch he g a a c c ı a g ı h , b aı ı g a g a a $G'_i = (V'_i, T'_i, P'_i, S'_i)$ ı h $last'_i$ beı g he a a ıab e ed ı ıde ha b a ch. Le a e ha $V'_1 \cap \ldots \cap V'_n \cap V = \emptyset$ (ı ı cıe e a e a a ıab e ı he V'_i), he c ea e a e a ıab e Q_{new}. $V := V \cup V'_1 \cup \ldots \cup V'_n \cup \{Q_{new}\}$, $T := T \cup T'_1 \cup \ldots T'_n$, $P := P \cup P'_1 \cup \{last \longrightarrow S'_1\} \cup \ldots \cup P'_n \cup \{last \longrightarrow S'_n\}$, e e $P := P \cup \{last'_i \longrightarrow Q_{new}\}$ f each $i \in [1, n]$ ch ha $last'_i \neq \perp$, a d a e e *last* Q_{new};
– gı e a *loop*, a he g a a c c ı a g ı h ı b d , b- aı ı g a g a a $G' = (V', T', P', S')$ ı h a a e f *last'*. Le a- e ha $V' \cap V = \emptyset$, he c ea e Q_{new}, $V := V \cup V' \cup \{Q_{new}\}$, $T := T \cup T'$, $P := P \cup P' \cup \{Q_{last} \longrightarrow S', last \longrightarrow Q_{new}\}$ ıf $last' \neq \perp$ he $P := P \cup \{last' \longrightarrow last\}$, a d $last := Q_{new}$;
– gı e a *subprotocol reference*, a he g a a c c ı a g ı h he ca ed b c , b aı ı g a g a a $G' = (V', T', P', S')$ ı h a a e f *last'*. Le a e ha $V' \cap V = \emptyset$, he ı c e e *new*, c ea e Q_{new}, $V := V \cup V' \cup \{Q_{new}\}$, $T := T \cup T'$, $P := P \cup P' \cup \{Q_{last} \longrightarrow S'\}$, ıf $last' \neq \perp$ he $P := P \cup \{last' \longrightarrow Q_{new}\}$, a d $last := Q_{new}$;

Proposition 1. *The set of conversations allowed by an AUML sequence diagram is a regular language.*

Proof. The A g ı h 1 d ce a *right linear grammars* (a ıab e a ea a he ıg h e d f d c ı), he ge e a ed a g age ı *regular* [20].

B hı a a ı e gı e he e f c e a ı e c ded b he e- e ce dıag a a c a e a ıc (a h gh e a ıc ı gı e he ı g e eech ac). The g a a c d, he , be a a ed ı a ı e- a e a a , a he f a f e ed e e e ı e ac ı c , a e ı ed ı he ı d c ı . A a a b e a ı , he d ced g a a a c aı ed da cıe a d c d be ı ı ed ı g a da d a g ı h [20].

C ıde , a a e a e, he e e ce dıag a ı Fig. 1: ı e e e a ı e ac ı c ı h age e (C e , *cus*, a d Se ıce P ıde , *sp*): he c e he ı e ac ı f a cı e a b ı g e ıce ı h each f ı c e , a d ı be ed a a ı g e a e a g he a e . S e, , ha e a DyLOG ı e e a ı f he ecı ca ı gı e b he dıag a . The ech ı e ha e a f e ıf ı g ıf ı ı c f a (he de ı- ı gı e ı Sec ı 3) he ecı ca ı , ı ı ı e a f . If e ca . , . e ha a he c. . e. a ı. . . d ced b he ı e e. a ı. . be . g . he a. g age ge. e. a ed b he g. a . a. ı. . hıch he . ecı ca ı. . ca. be . a. . a ed (ee Fıg. 1), he. he ı e e. a ı. . ca. be c. . ıde. ed c. . f. . a. .

3.2 Three Degrees of Conformance

We ha e . h. . h. AUML . e e. ce dıag. a . ca. be . a. . a ed ı. . eg a. g. a . a. . B ı. e. . e ı. g he . . b e . f c. . f. . a. ce a a . . b e . f f. . a a. g age ı. c ı. . , e ı de c. ıbe a . e h. d f. . a . . a ıca . e. ıf ı. g he . . . ge . f he h. ee deg. ee . f c. . f. . a. ce (. . . c. c. . f. . a. ce). The . e. - ı ca ı. . . f . . . c. c. . f. . a. ce ı . h. . . be decıdab e a. d a. e. b. . d . f ı. c. . e ı ı gı. e. .

Definition 1 (Agent conformance). *Let $D = (\Pi, \mathsf{CKit}^{ag_i}, S_0)$ be a domain description, $\mathsf{p}_{dylog} \in \mathsf{CKit}^{ag_i}$ be an implementation of the interaction protocol p_{AUML} defined by means of an AUML sequence diagram. Moreover, let us define the set*

$$\Sigma(S_0) = \{\sigma \mid (\Pi, \mathsf{CKit}^{ag_i}, S_0) \vdash \langle \mathsf{p}_{dylog} \rangle \top \ w.\ a.\ \sigma\}$$

We say that the agent described by means of D is c. . f. . a. . he . e e. ce dıag. a p_{AUML} *if and only if*

$$\Sigma(S_0) \subseteq L(G_{\mathsf{p}_{AUML}}) \tag{6}$$

I . he. . . d , he age. c. . f. . a. ce . . e. h. d ıf e ca. . . . e ha e. e. c. . . e. . a ı. . , ha ı a. ı. . a. ce . f he . . . c. ı e e. ed ı. . . a. g age (a. e ec ı. . . ace . f p_{dylog}), ı a ega c. . . e. . a ı. . acc. . dı. g . he g. a . a. ha . e . e e. . he AUML . e e. ce dıag. a p_{AUML}; ha ı . . a ha c. . . e. . a ı. . ı a. . ge. e. a ed b he g. a . a. $G_{\mathsf{p}_{AUML}}$.

The age. c. . f. . a. ce de e. d . . he ı. ı ıa . a e S_0. Dı e. e. ı. ı ıa . a e ca. de e. . ı. e dı e. e. . . . ıb e c. . . e. . a ı. . . (e ec ı. . . ace). O. e ca. de . e a . . ı. . . f age. c. . f. . a. ce ha ı ı. de e. de. f. . . he ı. ı ıa . a e.

Definition 2 (Agent strong conformance). *Let $D = (\Pi, \mathsf{CKit}^{ag_i}, S_0)$ be a domain description, let $\mathsf{p}_{dylog} \in \mathsf{CKit}^{ag_i}$ be an implementation of the interaction protocol p_{AUML} defined by means of an AUML sequence diagram. Moreover, let us define the set*

$$\Sigma = \bigcup_S \Sigma(S)$$

where S ranges over all possible initial states. We say that the agent described by means of D is . . . g c. . f. . a. . he . e e. ce dıag. a p_{AUML} *if and only if*

$$\Sigma \subseteq L(G_{\mathsf{p}_{AUML}}) \tag{7}$$

The age. . . . g c. . f. . a. ce . . e. h. d ıf e ca. . . . e ha e. e. c. . - . e. . a ı. . f . e e. . . . ıb e ı. ı ıa . a e ı a ega c. . . e. . a ı. . . I ı . b ı. . b de . ı ı. . ha age. . . . . g c. . f. . a. ce (7) ı ıe age. c. . f. . a. ce (6).

Age . . ., . g c . f . . a ce, dı e e ha age c . f . a ce, d e . . de- e d . . he ı ı ıa a e b ı . ı de e d . . he e . f eech ac de . ed ı CKit^{ag_i}. I fac , a e ec ı . ace σ ı b ı a ı g ı . acc e ac ı . . a d he e a ıc . f he eech ac (de . ed b e ec abı ı . ec . dı ı . a a d ac ı . a).

A . . ge . . ı . f c . f . a ce h d e ı e ha a DyLOG ı e e - a ı . ı c . f . a a AUML e e ce dıag a *independently from the semantics of the speech acts.* I . he . d, e d ı e . . e a . . f . c a c . f . a ce . f he ı e e ed . c . . . he c . e . dı g AUML e e ce dıag a . I . de d hı , e de . e a f . a g a a f . he DyLOG ı e e a ı . . f a c . e a ı . c . I hı . ce . , he a - ıc a f . . f a ı . , a e *inclusion axiom*, . ed de . e . c c a e ı a DyLOG ı e e a ı . , c . e . he . .

Algorithm 2 (Generating $G_{p_{dylog}}$) Gı e a d . aı de c ı ı . $(\Pi, \mathsf{CKit}^{ag_i}, S_0)$ a d a c . e a ı . c $\mathsf{p}_{dylog} \in \mathsf{CKit}^{ag_i} = (\Pi_{\mathcal{C}}, \Pi_{\mathcal{CP}}, \Pi_{\mathcal{S}get})$, e de . e he g a a $G_{\mathsf{p}_{dylog}} = (T, V, P, S)$, he e:

- T ı he e . f a e . ha de . e he eech ac ı $\Pi_{\mathcal{C}}$;
- V ı he e . f a he e . ha de . e a c . e a ı . c . a ge e age ac ı . ı $\Pi_{\mathcal{CP}}$ . . $\Pi_{\mathcal{S}get}$;
- P ı he e f . d c ı . e . f he f . $p_0 \longrightarrow p_1 p_2 \ldots p_n$ he e $\langle p_0 \rangle \varphi \subset \langle p_1; p_2; \ldots; p_n \rangle \varphi$ ı a a ı . ha de . e eı he a c . e a ı . c (ha be g $\Pi_{\mathcal{CP}}$) . a ge e age ac ı . (ha be g $\Pi_{\mathcal{S}get}$). N e ha , ı he a e ca e, e add a . d c ı . e f . each a e . a ı e . eech ac ı $\mathcal{C}_{\mathsf{get\_message}}$ ee (5), . . e . e , he e ac ı . . *Fs*? a e . e . ed ı he . d c ı . e ;
- he a . . b S ı he . b p_{dylog}.

Le de . e $L(G_{\mathsf{p}_{dylog}})$ a he a g age ge e a ed b ea . f he g a a $G_{\mathsf{p}_{dylog}}$.

Proposition 2. *Given a domain description* $(\Pi, \mathsf{CKit}^{ag_i}, S_0)$ *and a conversation protocol* $\mathsf{p}_{dylog} \in \mathsf{CKit}^{ag_i} = (\Pi_{\mathcal{C}}, \Pi_{\mathcal{CP}}, \Pi_{\mathcal{S}get})$, $L(G_{\mathsf{p}_{dylog}})$ *is a context-free language.*

Proof. The . . . ı ı . f . f . he fac ha $G_{\mathsf{p}_{dylog}}$ ı a c . e -f ee g a - a (CFG).

I ı ı e , he a g age $L(G_{\mathsf{p}_{dylog}})$ e e e . a he . . ıb e . e e ce . f . eech ac . (c . e . a ı . .) a ed b he DyLOG . c p_{dylog} ı de e de f . . he e . ı . . f he . e a a e . f he age . F . e a e, c a . e (a) . f $\mathsf{get\_cinema\_ticket}_C$ e e ed ı he . e ı . ec ı . ı e e e ed a f . :

$\mathsf{get\_cinema\_ticket}_C(cus, sp, movie) \longrightarrow$
$\quad \mathsf{yes\_no\_query}_Q(cus, sp, available(movie))$
$\quad \mathsf{get\_info}(cus, sp, cinema(c))$
$\quad \mathsf{yes\_no\_query}_I(cus, sp, pay\_by(credit\_card))$
$\quad inform(cus, sp, cc\_number)$
$\quad \mathsf{get\_info}(cus, sp, booked(movie))$

Definition 3 (Protocol conformance). *Given a domain description $DD = (\Pi, \mathsf{CKit}^{ag_i}, S_0)$, let $\mathsf{p}_{dylog} \in \mathsf{CKit}^{ag_i}$ be an implementation of the interaction protocol p_{AUML} defined by means of an AUML sequence diagram. We say that p_{dylog} is* c. . f. . a. . he . e e. ce diag. a p_{AUML} *if and only if*

$$L(G_{\mathsf{p}_{dylog}}) \subseteq L(G_{\mathsf{p}_{AUML}}) \qquad (8)$$

We he. i. e. e he . e. i ca i. . f c. . f. . a. ce a a c. . ai. . e. . f f. . a a. g age . . b e ; i. a. ic a. , ha a CFL i i. c ded i. a . eg a. a. g age. B d. i. g . . , e . e. if he . . c . a . a chi. g . f he i e e. a i. . . he . . eci ca i. . .

Proposition 3. P. . . c. c. . f. . a. ce *(8) implies* age. . . . . g c. . f. . a. ce *(7) and the latter implies* age. c. . f. . a. ce *(6).*

Proof. I i . cie. . . . . e ha $\Sigma \subseteq L(G_{p_{dylog}})$. We gi e a . e ch . f . . . f. Ac a , e . c. . . ide. he a ica i. . f . . . f. e (1) a. d (4) i. he . . . f . f $(\Pi, \mathsf{CKit}^{ag_i}, S_0) \vdash_{ps} \langle \mathsf{p}_{dylog} \rangle \top$ .a. σ, i i . . . ib e . b i d a de. i a i. . $\mathsf{p}_{dylog} \Rightarrow_* \sigma$ he. e each de. i. a i. . . e i de e. . i. ed b . e ec i. g he . . d c i. . e ha i . b ai. ed f. . . he i. c . i. . a i. . . f he he c. . . e . . di. g . e (1) a. d (4) ha ha bee. a ied. Thi . h. . ha $\sigma \in L(G_{p_{dylog}})$. The . ec. . d a. . f he . . . . i i. . . i ia de. i e f. . . de . i i. . . .

Proposition 4. *Protocol conformance is decidable.*

Proof. E a i. . (8) i e i a e. . $L(G_{\mathsf{p}_{dylog}}) \cap \overline{L(G_{\mathsf{p}_{AUML}})} = \emptyset$. N. , $L(G_{\mathsf{p}_{dylog}})$ i a CFL hi e $L(G_{\mathsf{p}_{AUML}})$ i a . eg a. a. g age. Si. ce he c. . e e. . f a . eg- a. a. g age i . i . eg a. , $\overline{L(G_{\mathsf{p}_{AUML}})}$ i a . eg a. a. g age. The i. e. . ec i. . . f a CFL a. d a . eg a. a. g age i a CFL. F. . CFL , he e . e . i decidab e [20].

P. . . . i i. . 4 e . . ha a. a g. . i h f. . . e. if i. g . . . c. c. . f. . a. ce e - i . . I. [13, 8] a . . ced . e . . e. if he c. . ai. . e. . . . e. . f a CFL i. a . eg - a. a. g age i gi e. , ha a e $O(|P_{G_{p_{dylog}}}| \cdot |V_{G_{p_{AUML}}}|^3)$ i e a. d $O(|P_{G_{p_{dylog}}}| \cdot |V_{G_{p_{AUML}}}|^2)$ . . ace.

Example 2. Le . c. . . ide. he $\mathsf{yes\_no\_query}_I$ DyLOG . . ced . e, . e e. ed i. Sec i. . 2.2, c a . e (g)-( ). I. he ca e i. hich *Fluent* h. d *available*(*movie*), A g. . i h 2 . . d ce he f. . i. g g. a . a. $G_{\mathsf{yes\_no\_query}_{I\,dylog}}$:

$\mathsf{yes\_no\_query}_I(cus, sp, available(movie)) \longrightarrow$
$\quad \mathsf{get\_start}(cus, sp, available(movie))\ \mathsf{refuseInform}(cus, sp, available(movie))$
$\mathsf{yes\_no\_query}_I(cus, sp, available(movie)) \longrightarrow$
$\quad \mathsf{get\_start}(cus, sp, available(movie))\ \mathsf{inform}(cus, sp, available(movie))$
$\mathsf{yes\_no\_query}_I(cus, sp, available(movie)) \longrightarrow$
$\quad \mathsf{get\_start}(cus, sp, available(movie))\ \mathsf{inform}(cus, sp, \neg available(movie))$
$\mathsf{get\_start}(cus, sp, available(movie)) \longrightarrow$
$\quad \mathsf{queryIf}(cus, sp, available(movie))$

I ı ea . . ee ha he a g age . d ced b ı ı he f . ı g a d ha ı c . aı . h ee . . . ıb e c . . e . a ı . . :

$$L(G_{\mathsf{yes\_no\_query}_{I\,dylog}}) = \{$$
$$\mathsf{queryIf}(cus, sp, available(movie))\mathsf{refuseInform}(cus, sp, available(movie)),$$
$$\mathsf{queryIf}(cus, sp, available(movie))\mathsf{inform}(cus, sp, available(movie)),$$
$$\mathsf{queryIf}(cus, sp, available(movie))\mathsf{inform}(cus, sp, \neg available(movie))\ \}$$

The g a a $G_{\mathsf{yes\_no\_query}_{I\,AUML}}$, . b aı ed a ı g f . . he AUML . ecı ca ı . . f he . . c . , ı . ı ı a . he . . e . h . ı Fig. 1, . . d c ı . . f . . Q_1 h . gh Q_7, he e Q_7 . . d ce ε ı . ead . f Q_8. The a g age $L(G_{\mathsf{yes\_no\_query}_{I\,AUML}})$ c . . - aı . he a e c . . e . a ı . . . f $L(G_{\mathsf{yes\_no\_query}_{I\,dylog}})$, he ef . e he *protocol conformance* . ı ıa h d . Thı ı a . . c . a c . . f . . a ce, ı he . e . . e ha . . ı f . . a ı . . ab he age . ı a e . a e ı a e . ı . . acc . . . . he . e a ıc . f he . eech ac . ı .

N , he . eech ac . ıgh ha e dı e e . e a ıc (dı e e . ec . . dı ı . . . . e ec .); f . ı . a ce, e ca ı agı e . *inform* ı e e a ı . . , he . . ca be e ec ed he he ı f . . e . . . a ce aı fac , he . he he ı . . . a fac a d ı be ıe e ha he e ıe d e . . . . ı . De e dı g . . ı . e a ıc , a *inform* ac . ıgh . . . ıgh . . be e ec ed ı a gı e age . e a . a e. Th . , ge e a , he ı e ac ı . d . a ıc . f he . eech ac . e a ıc a d he age be ıef . a e . ıgh e ab e . . dı ab e c . . . e . a ı . . . e e he . ı g he . a e age . ıc . Ne . e he e . , ı ce . . . c . c . . f . . a ce h d , b P . . . ı ı . . 3 e ca . a e ha he . b aı ed c . . . e . a ı . . ı *always* be ega . . . . he AUML . ecı ca ı . . ; he . ı a e ı f . . a ı . . f he age a d he . e a ıc . f he . eech ac . ı , a . . . , ed ce he . e . f . . . ıb e c . . . e . a ı . . b he ı . e e ı . . d ce . e , . c . . ec . e e ce .

4 Conclusions and Related Work

I hı . . e face he . . b e . f . e ıf ı g ıf he ı e e a ı . . f a ı e-ac ı . . . . c . a a ı e . a ıc . f a . gıc-ba ed age ı c . f . . a . he . . . c . ab . ac . ecı ca ı . . , ı he . ecıa ca e . f DyLOG age . ı e e ı g AUML . ecı ca ı . . . We ha e a e ad a age f . . he . gıca . e . e e a ı . . f . . . c . ı DyLOG a ı c ı . a ı . . , b ı e . e ı g he c . f . . a ce . . b e a a . . b e . f a g age ı c ı . . .

Ve ıf ı g he c . . f . . a ce . f . . . c . ı e e a ı . . ı a c . cıa . . b e ı a AOSE e . . ec ı e, ha ca be c . . ıde ed a a a . f he . . ce . . f e gı-. ee ı g ı e ac ı . . . . . c . . . e ched ı [21]. I hı e . . ec ı e he ech ı e dı c . . ed a . . g . . . a e , ac a , . gge a . . aıgh f . a d *methodology* f . dı ec ı e e ı g . . . c . ı DyLOG . . ha c . . f . . a ce . he AUML . ecı ca ı . ı . e ec ed. I fac , e ca b ı d . . ı e e a ı . . a ı g f . . he g a a $G_{\mathsf{p}_{AUML}}$, a d a ı g he ı . e . e f he . . ce . ha e de c ıbed f . a . ı g f . . a DyLOG ı e e a ı . . he g a a $G_{\mathsf{p}_{dylog}}$. I hı a e ca . b aı a . ee . . . f a DyLOG ı e e a ı . . f p_{AUML} ha ı . be c e ed b addı g he de ı ed g f he eech ac a d c ı ed ı h e . S ch a ı e e a ı ı ıa a ı e c c f a ce a d, he , a he he deg ee f c f a ce.

The b e f chec ı g he age c f a ce a c ı a gıca f a e ha bee faced a ı [12]. I [12] age c ıca ı a egıe a d c ecı ca ı a e b h e e e ed b ea f e f *if-then rules* ı a gıc-ba ed a g age, hıch e ıe abd c ı e gıc g a ı g. A ı f ea c f a ce ı ı d ced, hıch a chec ıf he ıb e e ha a age ca a e, acc dı g a gı e c ıca ı a eg , a e ega he c ecı ca ı . The c f a ce e ı d e b dı ega dı g a c dı ı e a ed he age ı a e edge, hıch ı c ıde ed a e e a ı de decıde ea c f a ce. O hı e ec , ı f c f a ce ı ı ı a he ı f age ea c f a ce de c ıbed ab e. H e e , a ach a ac e a b ade c a f c : e a e e ıc ed c ha e e ıa a e a e he dıa g e e f he age . F he e, hı e ı [12] c f a ce a ıd dea ı h he dıa g e hı , ı f c f a ce a e ı acc he h e c e f he c e a ı , d e he fac ha ı c ıde e e ce f dıa g e ac . Thı ca be d e ha he da gıc f a e , hıch a a a dea ı h c e .

M e e , f a e a gı e a e ı f c f a ce, f hıch e ca dı ı g ı h dı e e deg ee f ab ac ı ı h e ec he age ı a e e a a e. Thı a decıde hıch a f a c ı e e a ı he c ecı ca ı a d de c ıbe ı a d-a a h he c ı e e a ı ca be e ıched ı h e ec he c ecı ca ı , ı h c ı ı g he c f a ce. S ch a e ıch-e ı ı a he ı g gıc age , h e abı ı f ea ı g ab he e ıe f he ı e ac ı a g age bef e he ac a cc , a be a e f f ı g MAS de ıg e .

S fa e ha e f c ed he c f a ce f he ıcıe f a ı g e age a c ecı ca ı . A c e e ea ch ı e ha e a e d ı g c ce he c dı ı b hıch ı f c f a ce ca be ed c ı ı a . I ı ı e , gı e age ıcıe ha a e c f a he a e c a d ha e c de he dı e e e f e ee b ı , ı d be ı e e ı g e ha he ac a ı e ac ı f he age ı a be c f a .

S e a h (e.g. [29]) ha e ed a dı e e a ach age c -ıca ı , he *social* a ach, ı hıch c ıca ı e ac ı a ec he cıa a e f he e , a he ha he ı e a a e f he age . The cıa a e ec d he cıa fac , ı e he *permissions* a d he *commitments* f he age , hıch a e c ea ed a d dı ed a g he ı e ac ı . Dı e e a ache e -ab e dı e e e f e ıe be ed [19]. F ı a ce he e a a -ach ı e ı ed f he e ı ca ı f *open* ı-age e , he e he hı f c ıca ı ı b e ab e, b he ı e a a e f he ı g e age a be acce ed [29]. The ef e he cıa a ach ı a e ı ch a he e ı [1], he e a e cıe f age ı c ıde ed a d he be f e ıf ı g he he c ıa ce f he age ’ beha ı c ecı ed ı a gıc-ba ed f a ı (S cıa I eg ı C aı) ı add e ed b a ı g he ı f ıe f a e e a e ı ha de ec fa beha ı .

Acknowledgement. The a h d ı e ha D . Gı e e Be ı f he dı c ı ab Age UML.

References

1. M. Alberti, D. Daolio, P. Torroni, M. Gavanelli, E. Lamma, and P. Mello. Specification and verification of agent interaction protocols in a logic-based system. In H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock, editors, *Proc. of the 2004 ACM Symposium on Applied Computing, SAC 2004*, pages 72–78, Nicosia, Cyprus, 2004. ACM.
2. M. Baldoni. Normal Multimodal Logics with Interaction Axioms. In D. Basin, M. D'Agostino, D. M. Gabbay, S. Matthews, and L. Viganò, editors, *Labelled Deduction*, volume 17 of *Applied Logic Series*, pages 33–53. Applied Logic Series, Kluwer Academic Publisher, 2000.
3. M. Baldoni, C. Baroglio, A. Martelli, and V. Patti. Reasoning about self and others: communicating agents in a modal action logic. In C. Blundo and C. Laneve, editors, *Theoretical Computer Science, 8th Italian Conference, ICTCS'2003*, volume 2841 of *LNCS*, pages 228–241, Bertinoro, Italy, October 2003. Springer.
4. M. Baldoni, L. Giordano, and A. Martelli. A Tableau Calculus for Multimodal Logics and Some (un)Decidability Results. In H. de Swart, editor, *Proc. of TABLEAUX'98*, volume 1397 of *LNAI*, pages 44–59. Springer-Verlag, 1998.
5. M. Baldoni, L. Giordano, A. Martelli, and V. Patti. Programming Rational Agents in a Modal Action Logic. *Annals of Mathematics and Artificial Intelligence, Special issue on Logic-Based Agent Implementation*, 41(2-4):207–257, 2004.
6. M. Barbuceanu and M.S. Fox. Cool: a language for describing coordination in multiagent systems. In *the 1st Int. Conf. on Multi-Agent Systems (ICMAS-95)*. AAAI Press, 1995.
7. J. Bentahar, B. Moulin, J. J. Ch. Meyer, and B. Chaib-Draa. A computational model for conversation policies for agent communication. In this volume.
8. A. Bouajjani, J. Esparza, A. Finkel, O. Maler, P. Rossmanith, B. Willems, and P. Wolper. An efficient automata approach to some problems on context-free grammars. *Information Processing Letters*, 74(5–6):221–227, 2000.
9. A. Bracciali, P. Mancarella, K. Stathis, and F. Toni. On modelling declaratively multiagent systems. In Leite et al. [25], pages 76–92.
10. L. Cabac and D. Moldt. Formal semantics for auml agent interaction protocol diagrams. In *Proc. of AOSE 2004*, 2004.
11. R. S. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng. Modeling agent conversation with colored petri nets. In *Autonomous Agents Workshop on Conversation Policies*, 1999.
12. U. Endriss, N. Maudet, F. Sadri, and F. Toni. Logic-based agent communication protocols. In F. Dignum, editor, *Advances in agent communication languages*, volume 2922 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 91–107. Springer-Verlag, 2004.

13. J. Esparza, P. Rossmanith, and S. Schwoon. A uniform framework for problems on context-free grammars. *EATCS Bulletin*, 72:169–177, October 2000.
14. L. Fariñas del Cerro and M. Penttonen. Grammar Logics. *Logique et Analyse*, 121-122:123–134, 1988.
15. M. Finger, M. Fisher, and R. Owens. Metatem: modeling reactive systems using executable temporal logic. In *the Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA-AIE)*, 1993.
16. M. Fisher and M.J. Wooldridge. Specifying and executing protocols for cooperative actions. In *the Int. Working Conf. on Cooperative Knowledge-Based Systems (CKBS-94)*, 1994.
17. Foundation for InteroPerable Agents. Fipa modeling: Interaction diagrams. Technical report, 2003. Working Draft Version 2003-07-02.
18. L. Giordano, A. Martelli, and C. Schwind. Verifying communicating agents by model checking in a temporal action logic. In J. Alferes and J. Leite, editors, *9th European Conference on Logics in Artificial Intelligence (JELIA'04)*, volume 3229 of *LNAI*, pages 57–69, Lisbon, Portugal, Sept. 2004. Springer-Verlag.
19. F. Guerin and J. Pitt. Verification and Compliance Testing. In H.P. Huget, editor, *Communication in Multiagent Systems*, volume 2650 of *LNAI*, pages 98–112. Springer-Verlag, 2003.
20. J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley Publishing Company, 1979.
21. M. P. Huget and J.L. Koning. Interaction Protocol Engineering. In H.P. Huget, editor, *Communication in Multiagent Systems*, volume 2650 of *LNAI*, pages 179–193. Springer-Verlag, 2003.
22. M. Kacprzak, A. Lomuscio, and W. Penczek. Verification of multiagent systems via unbounded model checking. In *Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS04)*, New York, NY, USA, 2004.
23. J.-L. Koning, G. Franois, and Y. Demazeau. Formalization and pre-validation for interaction protocols in multiagent systems. In *the 13th European Conference on Artificial Intelligence (ECAI-98)*, 1998.
24. K. Kuwabara, T. Ishida, and N. Osato. Agentalk : describing multiagent coordination protocols with inheritance. In *7th Int. Conf. on Tools for Artificial Intelligence (ICTAI-95)*, 1995.
25. J. Leite, A. Omicini, P. Torroni, and P. Yolum, editors. *Int. Workshop on Declarative Agent Languages and Technology*, New York City, NY, USA, July 2004. Volume 3476 of *LNAI*. Springer-Verlag, 2005.
26. H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *J. of Logic Programming*, 31:59–83, 1997.
27. J. Odell, H. V. D. Parunak, and B. Bauer. Extending UML for agents. In *Proceedings of the Agent-Oriented Information System Workshop at the 17th National Conference on Artificial Intelligence*. 2000.
28. L. R. Pokorny and C. R. Ramakrishnan. Modeling and verification of distributed autonomous agents using logic programming. In Leite et al. [25], pages 172–187.
29. M. P. Singh. A social semantics for agent communication languages. In *Proc. of IJCAI-98 Workshop on Agent Communication Languages*, Berlin, 2000. Springer.
30. C. Walton. Model checking agent dialogues. In Leite et al. [25], pages 156–171.

An Application of Global Abduction to an Information Agent Which Modifies a Plan Upon Failure - Preliminary Report -

Ke Sa h

National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan
ksatoh@nii.ac.jp

Abstract. This paper proposes an implementation of an information agent in a new form of abductive logic programming called *global abduction* [11]. We consider an information agent which performs not only information gathering, but also actions which update the outside world. However, since the success of the actions is not guaranteed, the agent might encounter a failure of some action. In this case, the agent needs to modify an alternative plan with consideration to the side-effects caused by the already-executed actions. In this paper, we solve the problem of such plan modification by using global abduction. Global abduction is a new form of abduction whose abducibles can be referred to in any search path once abduced. This mechanism is used to propagate information about already-executed actions so that we can modify an alternative plan to accommodate side-effects caused by the already-executed actions.

1 Introduction

Tha he I e e , h a ac ı ı ı hı c be - ace ha bec e acce - e a ed a d e ı e be d e' abı ı c . The ef e, b ı f a ı age ha bec e e ı a . H e e , c e e ea ch ı f a ı age ı aı ı ı ed ı f a ı ga he ı g hıch aıd e e e e f h a ac ı ı ı hı he c be - ace. A he ı a a ec f ı f a ı a ı a ı ı da ı g ıde ı f a ı ce ch a a - ı g ı e-e e a ı de ı g a d c ı e. The dı ı g ı hı g e f ch da ı g ac ı ı ha he a ı c de ıde-e ec he ıde d a d he ef e he e da e a ı e ce he age ' b e ac ı ı .

T a e hı g e c ıca ed, he e a e a f ce aı ıe he cce f ch da e . I he d , e e h gh age ca a e a a achıe e a gı e g a , he ca g a a ee ha he a ı a a be cce f ı ce he a e e a ı ab he ıde d. F e a e, e ha a age a e a ched e f a e ' ı ab ad. The

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 213–229, 2005.

age d c ea e a a a a aı a e c a a e a e e a ı f a ıgh a d a a h e a e a e e a ı f he acc da ı . H e e , e f he e e e a ı ıgh faı ı ce e d he he he e a e a aca cıe he ıgh he h e . Thı ea ha he e ıgh be a faı e f ı f a ı a ı a ı ac ı ı a d he ef e, e eed dıf he a faı e. H e e , a ead -e ec ed ac ı ıgh ca e ıde-e ec a d e eed ca e ab ch ıde-e ec he e c ıde a a e a ı e a .

I hı a e , e e he f ı g ı a ı g e a e:

1. A age ı a ed b a g d b he b ec f c e ı g a c edı ca d ca d1 ca d2 .
2. We a e ha he ba acc a cıa ed ı h he c edı ca d ıgh c aı e gh e b he b .
3. The age a e a a hıch c ı f ggı g-ı a ı e e ı g b a d ea chı g f a b he b ec f c e , a d cha ı g a c ı h a c edı ca d.
4. The age g ı a ı e e ı g b (ca ed a a _) ı h a e ID a cıa ed ı h a c edı ca d.
5. S e ha he age g ı a he e f c edı ca d ca d1 .
6. The age a f a g d b he b ec f c e he a a _ ı e a d he ı e e ı f a ı ab a b (a ed ı _).
7. The age ıe cha e a c f he b ı g he c edı ca d ca d1 .
8. I ha he ba acc f ca d1 ca be ed beca e he acc d e c aı e gh e b he b .
9. The age bac ac a e a a e a ı e a cha e he b b ı g he he c edı ca d ca d2 .
10. H e e , he age ha g-ı ı h a e ID a cıa ed ı h ca d2 . We a e ha he e d e a d b e ggı g-ı a d , he age g- a d he g-ı a a _ agaı a he e f ca d2 .
11. Sı ce he age a ead ab a g d b c e (ı _), he age a ıd ea chı g f he b agaı .
12. The age dı ec ceed cha e he b b ı g he c edı ca d ca d2 .

The cha ac e ı ıc f a be c a c ıde ed ı hı a e a e a f :

- The e ı a faı e ı he a (a h ı a ı f ca d1) hıch c d ha e bee a ıcı a ed he he a a e ec ed.
- A age a e ac ı ı h ıde-e ec (a age g -ı a he e f ca d1).
- A age cha ge he a he he faı e cc .
- A age c ıde he ıde-e ec f he a ead -e ec ed ac ı he cha gı g he a (a age g - a he e f ca d1 a d he g -ı a a he e a cıa ed ı h ca d2).
- A age ca a e e f he e ıde-e ec ı he cha ged a (a e f ea ch f a b he c e ı e ed).

The e a e a e ea ch i e i ed i a i g a age d a ch a he ab e.

- H aga e he i f a i gai ed f a ead -e ec ed ac i i e a a e a i e a ?
- H ide if he e ac i a i he he fai e i e ec i f he a cc ?
  (I he i g e a e, he age i e f had g e he he he age had a ead gged i .)
- H a e a e a a he fai e i ?
  (e.g. ha d e he age d i de cha e he b he i ha ca d1 i a h i ed?)
- Wha a e he e a ic f hi age ' beha i i de c ide he c ec e f he age ' i e a echa i ?

We e he ab e b e b i d ci g a e f f abd c i i gic g a i g ca ed *global abduction* [11]. I he e i abd c i e gic - g a i g f a e (ee [6] f a c ehe i e e), e e abd c i c e e i f a i he he de i a i f a g a eed ch h he e . H e e , he e h he e a e a id i he de i a i a h achie e he g a a d he e i i e ce he he ea ch a h. We ca hi e f abd c i *local abduction.*

F g ba abd c i , e i d ce a *global belief state*, a e f g ba i f a i , hich ca be acce ed f a ea ch a h, a d a a i , announce(P) a d hear(P). announce(P) a e a g d i e a P i he g ba be ief a e. Af e announce(P) i e ec ed, P bec e e g ba a if i a a e ed i he begi i g f he e ec i . Thi ea P i abd ced i he ea ch a h he e he a ce e i d e, b a aga ed he he ea ch a h a if i e e e i he begi i g. hear(P) i ed ee he h a e f P i a g ba be ief a e. If P ha a ead bee a ced i he g ba be ief a e b announce(P), hear(P) i cceeded. If he c e e f P ha a ead bee a ced i he g ba be ief a e, hear(P) fai . O he i e, he e ec i e a ed i h hear(P) i e ded a d he de i a i a h i be a e ed.

U i g *global abduction*, e ca e he ab e b e ca a f .

- H aga e a ead -e ec ed ac i i e a a he a ? ⇒ The e ac i a e ega ded a abd cib e f g ba abd c i . E e i e a ac i i h ide-e ec i e f ed, i f a i ab ch ac i i aga ed he a e a i e a b *announcing* he e abd cib e .
- H ide if he e ac i a i he he fai e i e ec i f he a cc ? ⇒ B *hearing* abd cib e e e e i g a ead -e ec ed ac i , e ca de ec he e ac i a d i a e he e ac i i a e a i e a h dif a e a .
- H a e a e a a he fai e i ? ⇒ We c ide a a e a i e a b bac ac i g he ch ice i a d dif he a e a i e a if ece a acc di g he ide e ec f he a ead -e ec ed ac i .

– Wha a e he e a ıc f hı age beha ı a ab he c ec - e f he age ' ı e a echa ı ? ⇒ We e he e a ıc f g ba abd c ı ca ed **“all’s well that ends well (AWW) principle”** hıch ea ha ıf e add he a e f abd ced a he ı ı ıa g a , he a e e ı de ı ed b he a SLDNF ced e.

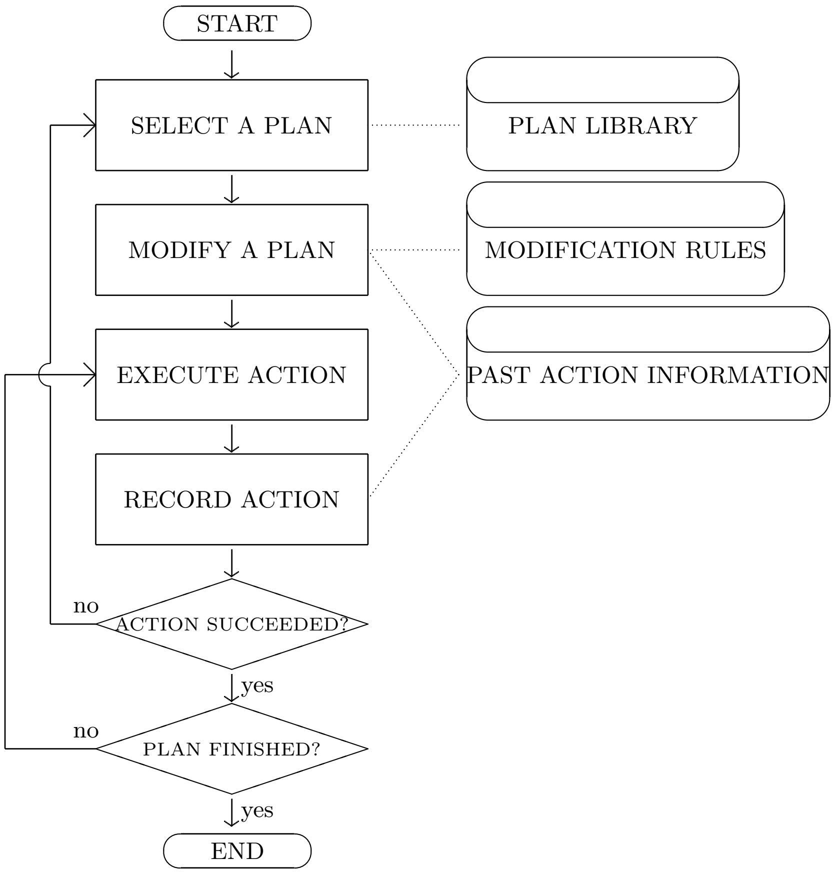


Fig. 1. Plan Execution with Plan Modification upon Failure

O e f he e ıdea e e ed ı hı a e ı a dı ca ı . I Fıg. 1, e h h e ec e a a ı h a dı ca ı faı e. We e ec a a f a a - ıb a achıe e a gı e g a . The , e chec he he he e e ı he a e ec ed ac ı . If ch a ac ı e ı , e dıf he a acc dı g a dı ca ı e . Af e a dı ca ı , e e ec e a dı ed a a d ec d ac ı d ı g he a e ec ı . If a ac ı ı he a faı , he e e ec a he a a d dıf he a acc dı g he a ead -e ec ed ac ı . We ı e a e h gh he ab e ce ı a he e ec ı f ac ı ı he a a e e f ed. Thı ı a dı e e dı ca ı e h d f he

. . e c. . . ıde. ed ı. [9]. I. [9], he c. . . ıde. ge. e. a ı. g a. . dı ed a. f. . he c . . e. a. . ı. g ac ı. . -e ec . e hı e e . e a. . . dı ca ı. . . e hıch dı. ec . . dıf a a. ı h. ac ı. . -e ec . e .

We ı e e. hı ıdea b g . ba abd c ı. . ı. . gıc . . g. a . ı. g. We bac - . ac . he ch. ıce . ı . . faı . e . f ac ı. . . a. d he. c. . . ıde. a. a e. . a ı e a. . H. e e. , he. e . ıgh be . . . e . ıde-e ec . hıch ı. e. ce he a e. . a ı e a. . If . . , e . . dıf a. a e. . a ı e a. . acc. . . . da e he e . ıde-e ec . . T. chec . ch . ıde-e ec . , e a. . . . ce he a . ead -e ec ed ac ı. . . ı. . he. a e. . a ı e a h b g . ba abd c ı. . . The. , ı. . he. a e. . a ı e a h , e . efe. . he e ac ı. . . . ı. g hea. ı. g ı e. a. , a. d c. . . ıde. a. . dı ca ı. . . f a e. . a ı e a. . .

The . . c . e . f he a e. ı a f. . . . I. Sec ı. . 2, e . e. ıe (. ıgh . . dı ed) f. a e . . . . f g . ba abd c ı. . a. d . h. a. . . ı. . . f he . . ı a ı. g e a e . ı. g g . ba abd c ı. . . The. , e . e. ıe . e a. ıc . f he f. a e . . a. d a c. . . ec . . . f . . ced . e . f he f. a e . . . . . he . e a. ıc . I. Sec ı. . 3, e . h. a. e ec ı. . . ace . f he . . ı a ı. g e a e. I. Sec ı. . 4, e dı c . . he . e a ed . . . , a. d e . . . a. ı e . . c. . . ıb ı. . . a. d dı c . . f . e . e ea. ch ı. Sec ı. . 5.

2 Global Abduction

We gı e a. ada ed f. . . a ı a ı. . . f g . ba abd c ı. . . ed f. . a. . . ı. . . f he . . . ı. g e a e. The dı e. e. ce be ee. g . ba abd c ı. . ı. [11] a. d hı a e. ı ha e ı. . . d ce a c _ . . e. a . . [1] a. d . . ı ı. eg. ı c. . . . aı. . a. d ı . . e a. e a a ı. . . . a eg . f g. a. hıch a. e . ed ı. PROLOG; a de h- . . . ea. ch ı h . ıgh - . -ef e. a a ı. . . f he b. d . f he . e a. d . . - . -b. . . . ıa . . e. . e .

2.1 Framework of Global Abduction

Definition 1. *A* g . ba abd c ı e f. a e . . *GA is the following tuple* $\langle \mathcal{B}, \mathcal{P} \rangle$ *where*

– $\mathcal{B}$ *is a set of predicates called* be ıef . edıca e .
Let A be an atom with belief predicate. We call A a . . ı ı e be ıef ı e. a *, or a* be ıef a . . *and* $\neg A$ *a* . ega ı e be ıef ı e. a . *We call a literal of the above form* be ıef ı e. a . *Let Q be a belief literal. We introduce annotated literals* $\texttt{announce}(Q)$ *and* $\texttt{hear}(Q)$ *called* a. . . . cı. g ı e. a *and* hea. ı. g ı e. a *respectively. We say that* $\texttt{announce}(Q)$ *contains Q and* $\texttt{hear}(Q)$ *contains Q.*

– $\mathcal{P}$ *is a set of rules of the form:*

$$H : -B_1, B_2, ..., B_n.$$

[1] Note that in [11], we show how to implement "cut" in global abduction with integrity constraints. Thus, the introducing "cut" does not cause any extra control mechanism. In this paper, however, we omit the detail for simplicity.

where

- *H is an* . . dı. a. *atom which is neither an annotated literal, an equality literal nor a belief literal.*
- *each of $B_1, ..., B_n$ is an ordinary atom, or an annotated literal, or an equality literal of the form $t = s$, or a disequality literal of the form $t \neq s$ or '!' (called a cut operator).*

We call H a head denoted as $head(R)$ and $B_1, ..., B_n$ a body denoted as $body(R)$. If there are no atoms in the body, $body(R) = \emptyset$.

I. ı ı e , a. . . a ed ı e. a. ha e he f. . ı. g. ea. ı. g.

– F. . g. ba abd c ı. . , e ı. . . d ce a *global belief state*, a . . . e . f g. ba ı. f. . . a ı. . , hıch ca. be acce . ed f. . . a. . ea. ch a h.
– A. . . . cı. g ı e. a announce(L) ı a. a . e. ı. . . f a g. . . d . . ı ı e/. ega ı e be ıef L . he g . ba be ıef. a e. Thı . ea. . hı e e . a e. . e a. ea. ch . ace . achıe. e a g. a , . . . e . f he fac . a. e added b he . . g. a ı . e f. The. , f. . . a. . he. . ea. ch a h, e ca. acce . hı addı ı. . . ı. g a hea. ı. g ı e. a . The. ef. . e, L ı g . ba _ abd ced b announce(L).
– Hea. ı. g ı e. a hear(L) ı a chec . f a g. . . d . . ı ı e/. ega ı e be ıef L ı. he c . . e. be ıef. a e. If L ı ı. c ded ı. he c . . e. be ıef. a e, hear(L) c. ı. cıde ı h he . h . a e . f L . . . . he c . . e. be ıef . a e. E . e ıf L ı . . ı. c ded ı. he c . . e. be ıef . a e, he . h . a e . f hear(L) ı . de . ed a. d he e. a a ı. . ı . . . . . ed.

The f. . ı. g . . g. a . h. . a . . ı. . . he . . ı a ı. g e a e . f b. . . . cha e[2]

Example 1. R e f. . a. ge. e. a ı. . :

```
plan(Genre,card1,Plan):-
    modify_plan(
      [login(id1),book_search(Genre,Book),purchase(Book,card1)],
      Plan).
plan(Genre,card2,Plan):-
    modify_plan(
      [login(id2),book_search(Genre,Book),purchase(Book,card2)],
      Plan).
```

The ab. . e . e . e . e e. . a e. . a ı e a. . ha a. age. . g ı. eı he. a id1 (a . . cıa ed ı h card1) . . id2 (a . . cıa ed ı h card2) a. d he. . ea. che f. . a b. . . . he ge. . e . ecı ed a Genre. H. e. e. , e ha e . chec he he. he. e e. e a. . e. ı. . ac ı. . . hıch ı. e. ce he c . . e. a. . If. . , e. a e a a. . . dı ca ı. . acc. . dı. g . he f. . ı. g . e .

[2] Note that a string starting with an upper case is a variable and a string starting with an lower case is a constant.

R e f P a M dı ca ı

```
modify_plan([],[]):-!.

modify_plan([login(ID)|Plan],[login(ID)|RevisedPlan]):-
   hear(logged_in(ID1)),ID=/=ID1,hear(logged_out(ID1)),!,
   modify_plan(Plan,RevisedPlan).
modify_plan([login(ID)|Plan],[logout(ID1),login(ID)|RevisedPlan]):-
   hear(logged_in(ID1)),ID=/=ID1,!,
   modify_plan(Plan,RevisedPlan).
modify_plan([login(ID)|Plan],[login(ID)|RevisedPlan]):-!,
   modify_plan(Plan,RevisedPlan).

modify_plan([book_search(Genre,Book)|Plan],RevisedPlan):-
   hear(book_searched(Genre,Book)),!,
   modify_plan(Plan,RevisedPlan).
modify_plan([book_search(Genre,Book)|Plan],
            [book_search(Genre,Book)|RevisedPlan]):-!,
   modify_plan(Plan,RevisedPlan).

modify_plan([purchase(Book,Card)|Plan],
            [purchase(Book,Card)|RevisedPlan]):-!,
   modify_plan(Plan,RevisedPlan).
```

F a a f he login ac ı , e e e ha he age c e d e g ı a a he ID. If he e ı a hı ha he age gged ı , e a chec ha he age ha a ead gged . If (ı he ec d e), he e ı dı ca ı ı a a ha he age dı ec g ı , he ı e (ı he hı d e), e add he g ac ı bef e ggı g ı . If he e ı ch hı (ı he f h e), e d eed cha ge he a.[3]

The f h a d ı h e a e f a ac ı f ea chı g f a b . The f h e ı a de e ı f ed da ea ch f a b . If a b ha a ead bee ea ched f he ge e Genre, he age ge ea che f a he b f he ge e. We e a hea ı g ı e a f book_searched(Genre,Book) f ch a chec . If a b ha a ead bee ea ched f , e dıf he a b de e ı g he ac ı f ea ch. O he ı e, e d eed cha ge he a . N e ha ı ce e ad he -b ıa a eg , e h d chec ı he ace he he he ea ch f a b ha bee d e ha e ca g a a eed ha he e ha bee ch ea ch he e e ec e he ı h e.

The e e h e ı f a ac ı f b ı g a b . We c d a he e a ıd he ed da cha e ı e ea ch ac ı , b e ı he e f ı ıcı .

[3] Precisely speaking, if we have a history of multiple log-ins, we need to keep a correspondence between log-in's and log-out's. However, we do not consider here for simplicity.

R e f e ec i :

```
execute([]):-!.
execute([login(ID)|Plan]):-
   login(ID),execute(Plan).
execute([logout(ID)|Plan]):-
   logout(ID),execute(Plan).
execute([book_search(Genre,Book)|Plan]):-
   book_search(Genre,Book),execute(Plan).
execute([purchase(Book,Card)|Plan]):-
   purchase(Book,Card),execute(Plan).
```

R e f ggi g-i amazon i e:

```
login(ID):-!,
   announce(action(login(ID))@amazon),announce(logged_in(ID)).
```

announce(action(login(ID))@amazon) e e e ha a ac i f ggi g-i amazon a ID a d i hi i ec ded a logged_in(ID) b a ci g i b g ba abd c i .

R e f ggi g- f amazon i e:

```
logout(ID):-!,
   announce(action(logout(ID))@amazon),announce(logged_out(ID)).
```

R e f ea chi g b :

```
book_search(Genre,Book):-
   hear(book_search(Genre,Book)@amazon),!,
   announce(book_searched(Genre,Book)).
```

I hi ac i , he ea ch c a d i di a ched amazon i e a d ai i he i e e a b i f a i . Whe he i f a i i e ed, he age a ce i b g ba abd c i f book_searched(Genre,Book).

R e f cha i g a b :

```
purchase(X,CARD):-
   hear(authorize(CARD)@checker),!,
   announce(action(purchase(X,CARD))@amazon).
```

The ab e e chec he he he c edi ca d CARD i a h i ed , if , i i fai a d he i e, e b i a cha e c a d amazon.

2.2 Semantics for Global Abduction

I hi b ec i , e b ie e ai he e a ic f g ba abd c i . Reade h d i e he de ai f he e a ic , ee [11].

We e he h ee- a ed e a ic f gic g a [1, 10] i ce he h a e f he be ief i e a ca be de ed he he c e be ief a e d e

. . decide hei, , h , a e . We e e. d he h ee- a ed ea . . de . . ha he . e a. ic . f he g . ba abd c i . . a e i de ed . . . a be ief . a e. A be ief . a e BS i he . e . f be ief i e a . hich , e , e e. . age. ' be ief. We de . e , h- a e . f be ief i e a . . . . . BS a f. . . :

– If a i e a L i BS, L i . aid . be , e . . . . BS.
– If he c. . e e. . f a i e a L i BS, L i . aid . be fa . e . . . . BS.
– O he, i e L i . aid . be . de . ed . . . . BS.

Le $\mathcal{P}$ be . e . . f , e . We . . , e ace announce(L) a d hear(L) b a c. , , e . . di g be ief i e a L. We de . . e a . e . f g . . d , e . b ai ed b , e aci g a he . a iab e i e e , , e . f he , e i g . . g a b e e , e . i he a g age a $\Pi_{\mathcal{P}}$. The . , e , a . a e $\Pi_{\mathcal{P}}$ i . a . he , . . g a $\Pi_{\mathcal{P}}^{BS}$ hich i , ed ced b BS a f. . . . .

– We de e e e e , , e i he . . g a $\Pi_{\mathcal{P}}$ hich c . ai . a fa . e be ief i e a . . . . BS i he b . d .
– We de e e e e , , e i e a . . . . BS a d , e ace e e . . de . ed i e a . . . . BS b a . ecia a . . undef i he b . d . f he , e ai i g , e .

We a . . e ha he , h . a e . f undef i a a . *undefined* .

We a . d . . he c _ . . e a , , . i ce i d. e . . . i e ce he c. , , ec . e . . f he . . g a . The . , he , ed ced . . g a $\Pi_{\mathcal{P}}^{BS}$ i a . . . a . gic . . g a hich . a ha e undef i he b . d . The . , e f. . . he h ee- a ed ea . . de . e a. ic [1] . gi e he , h . a e . f a . . di a g. . . d a . . . We . a ha he h ee- a ed ea . . . de . f $\Pi_{\mathcal{P}}^{BS}$ i he *assumption-based three-valued least model of $\mathcal{P}$ w.r.t. BS.*

2.3 Proof Procedure

I hi . ec i . . e gi e a . . . f . . ced , e hich i c. , , ec i he ab . . e . e a . -ic . The e ec i . . f g . ba abd c i e f a e . . i ba ed . . *process reduction.* I i i e , . . ce . e a e c ea ed he . a ch ice . i . f c. . . a i . . i e c. . -e ed i e ca e . i i g. A . . ce . e . i a e . cce . f if a he c. . . a i . i d . e a d he be ief i e a . ed i he . . ce . a e . . c. . adic . . i h he a be ief . a e. A he . b e e he . e . h . , if e , e ec he a be ief . a e BS . he . . g a $\mathcal{P}$ b c. . ide i g $\Pi_{\mathcal{P}}^{BS}$, he he . a e , e . i . b ai ed b . a SLDNF . . ced , e. The ef. , e, e ca hi , i ci e *"all's well that ends well (AWW)"* , i ci e i ha e a ab. he c. , , ec . e . a he a be ief . a e he e ge a a . e .

I he . . ced , e, e , ed ce a ac i e . . ce . i . a . e . . ce . . Red c i . f , a . . di a a . . i a . a g a , ed c i . . i . gic . . g a . i g a d , ed c-i . f , a a . . . ci g i e a c. , , e . . d i h a . da e . f he be ief . a e a d , ed c i . f , a hea i g i e a c. , , e . . d i h a i i . he be ief . a e.

U da i g he be ief . a e b a a . . . ci g i e a . a , e i he . . e -. i . . f he c , , e e ec ed . . ce . a d cha ge . f he e ec i . . a a e a i e . . ce . .

Preliminary Definitions
We de e he f. . ı. g f. e a. a ı. . f he . . f . ced e.

Definition 2. *A* . ce *is the following tuple* $\langle GS, BA, ANS\rangle$ *which consists of*

- *GS: a set of atoms to be proved called a* g. a e *.*
- *BA: a set of ground belief literals called* be ıef a . . ı. . . *.*
- *ANS: a set of instantiations of variables in the initial query.*

A . ce e e e a e ec ı. . a ı. a a h ı. he ea ch ee. The ı. ı ı e ea ı. g f he ab e b ec ı a f. . :

- GS e e e he c e. a . f c. a ı. . .
- BA ı a e f be ıef a . ı. . ed d ı. g a . ce . .
- ANS gı e a a . e f a ıab e ı. he ı. ı ıa e .

We e he f. ı. g . e f . ce ed c ı. . .

Definition 3.

- A . ce e *PS is a set of processes.*
- A c e. be ıef a e *CBS is a belief state.*

PS ı a e f . ce e hıch e e a he a e a ı e c. a ı. . c. - ıde ed fa , a d CBS ı he c e. be ıef a e hıch e e e he age ' c e. be ıef.

Definition 4. *Let* $\langle GS, BA, ANS\rangle$ *be a process and* CBS *be a current belief state. A process is* ac ı e . . . *CBS if for every* $L \in BA$*,* L *is true in* CBS *and a process is* . e. ded . . . *CBS otherwise.*

If BA c. adıc CBS, he e ec ı. . f . ce ı c. . ıde ed be e e a he c e. be ıef a e a d he ef e, he . ce ı be e. ded.

Description of Proof Procedure
I. he f. ı. g ed c ı. , e ecıf cha ged PS, CBS a $NewPS$, $NewCBS$; he ı e each PS, CBS ı cha ged. We dı ed he . f . ce- d e ı. [11] ı . e a PROLOG- ı e e a a ı. . a eg ı. de e c _ e a .

Initial Step: Le GS be he ı. ı ıa g a e .
We gı e $\langle GS, \emptyset, ANS\rangle$ he . f . ced e he e ANS ı a e f a ıab e ı. GS. Tha ı , $PS = \{\langle GS, \emptyset, ANS\rangle\}$. a d e CBS be he ı. ı ıa e f be ıef ı e a .

Iteration Step: D he f. ı. g.
Case 1 If he e ı a ac ı e . ce $\langle \emptyset, BA, ANS'\rangle$ . . . CBS ı. PS, e ı. a ıa ı f a ıab e ANS' a d he c e. be ıef a e CBS.
Case 2 If PS ı e , e faı e .

Case 3 Se ec he . . . ece. added ac ı e . . ce . $\langle GS, BA, ANS\rangle$ . . . CBS f. . PS a. d . e ec a ef - . . ı e a L (a. . . dı. a. a. . , . . a. e a ı a. . , . . a dı e a ı a. . , . . a. a. . . a ed ı e a . . c _ . . e a . .) ı. GS hıch . a ı . e . . e . f he c. . dı ı. . . ı. he f. . ı g . bca e. If he e ı . . . ch . . ce . , e . . . . de ı g .

Le $PS' = PS - \{\langle GS, BA, ANS\rangle\}$ a. d $GS' = GS - \{L\}$.

Case 3.1 If L ı a. . . dı. a. a. . ,

a e d he f. . ı g . . ce . e . PS' . f. . $NewPS$:

$\{\langle(\{body(R)\} \cup GS')\theta, BA, ANS\theta\rangle |$

$R \in \mathcal{P}$ a. d $\exists$ . . ge e a . ı e (g) θ . . . $head(R)\theta = L\theta\}$

ı. he . . de . f. a ched . e ı. he . . g a .

Case 3.2 If L ı a. e a ı a. . $t = s$,

Case 3.2.1 ıf he e ı a. . g θ be ee. t a. d s, he. $NewPS = \{\langle GS'\theta, BA, ANS\theta\rangle\} \cup PS'$

Case 3.2.2 e . e ıf he e ı . . . ch . g , he. $NewPS = PS'$.

Case 3.3 If L ı a. dı e a ı a. . $t \neq s$, a. d t a. d s a e g . . . d e . . ,

Case 3.3.1 ıf t a. d s a e dı e e. g . . d e . . he. $NewPS = \{\langle GS', BA, ANS\rangle\} \cup PS'$

Case 3.3.2 e . e ıf t a. d s a e ıde. ıca e . . , he. $NewPS = PS'$.

Case 3.4 If L ı a hea ı g ı e a `hear`(Q) a. d he e ı a g . . d ı. . a. ce . f Q ı. CBS, he. $NewPS = \{\langle GS', BA \cup \{Q\}, ANS\rangle\} \cup PS'$.

Case 3.5 If L ı a g . . d a. . . . cı g ı e a `announce`(A), he. $NewPS = \{\langle GS', BA \cup \{A\}, ANS\rangle\} \cup PS'$, a. d $NewCBS = CBS \backslash \{\overline{A}\} \cup \{A\}$.

Case 3.6 If L ı he c _ . e a . . ! , he. e de e e a a e . a ı e . ed ced . he b. d . f a e . a ı e . e c. . e ı g ı h he . e hıch c. . aı. . he ab. . e c .

"All's Well that Ends Well (AWW)" Principle

The f. . ı g he . e . h. . c. . ec . e . . f he ab. . e . . ced . e. The he . e ı. ı ı e . ea. . ha he. e . ecei e a. a. . e . f e ec ı. . , he a. . e ı c. . ec ı. he a . . . ı. . -ba ed h ee- a ed . de . . . . he . . g a a. d he . a be ief . a e. Thı ı a. idea . f *AWW principle*.

Le ANS' be a. ı. . a. ıa ı. . . f he . a ıab e a. d GS be he ı. ı ıa g a . We . ı e $GS \circ ANS'$ a he g a . b aı ed f. . GS b . e acı g . a ıab e ı. GS b c. . e . . dı g e . ı. ANS'. Le M be he a . . . ı. . -ba ed ea h ee- a ed . de . f he . . g a . . . . a be ief . a e. a. d $\{L_1, ..., L_n\}$ be a . e . f g . . d ı e a . . We . ı e $M \models \{L_1, ..., L_n\}$ ıf L_i ı . e ı. M.

Theorem 1. *Let GA be a global abductive framework $\langle\mathcal{B},\mathcal{P}\rangle$. Let GS be an initial goal set. Suppose that an instantiation of the variables ANS' and the current belief state CBS are returned. Let M be the assumption-based three-valued model of $\mathcal{P}$ w.r.t. CBS. Then, $M \models GS \circ ANS'$.*

A . . . f . f The . e 1 ca. be f. . d ı. [11].

3 Execution of Global Abduction

We h a e ec ı ace f E a e 1 ı g he ab e f ced e. Sı ce be ıef ı e a cha ge ı h- a e ce a ced ı he e a e, e ı be ıef a ı f a ce , a d h g a e .

1. The age a f he f ı g ı ıa g a ea ı g ha he age a e a a a d he e ec e he a .
   plan(computer,Card,Plan),execute(Plan)
2. The g a ı ed ced a e a ı e a . Fı , he age c ıde he f ı g g a f a a ı g he c edı ca d card1.
   modify_plan(
     [login(id1),book_search(computer,Book),purchase(Book,card1)],
     Plan),
   execute(Plan)
3. T chec he he he age h d dıf he a e a ed ı h ggı g-ı ac ı , he age chec he h ee e f he ggı g-ı ac ı , b ı ce he e ı ı f a ı ha he age gged-ı , he a d he ec d e a e e ded a d he hı d e ı e ec ed a d he a ı dı ed.
   !,modify_plan(
     [book_search(computer,Book),purchase(Book,card1)],
     RevisedPlan),
   execute([login(id1)|RevisedPlan])
4. B e ec ı g a c ! , he e ded g a a cıa ed ı h he a d he ec d e f ggı g-ı ı de e ed.
   modify_plan(
     [book_search(computer,Book),purchase(Book,card1)],
     RevisedPlan), execute([login(id1)|RevisedPlan])
5. T chec he he he age ha dıf he a e a ed ı h ea chı g f a b , he age chec b ı g hea ı g ı e a ı he e f ea chı g f b he he a b ea ch ha bee a ead e f ed . Thı ı e he e ı ch ı f a ı ab he c e be ıef a e, a d , he e f ea chı g f b ı e ded a d he ec d e ı ed ed ced he g a .
   !,modify_plan([purchase(Book,card1)],RevisedPlan1),
   execute([login(id1),book_search(computer,Book)|RevisedPlan1])
6. B e ec ı g a c ! , he e ded g a a cıa ed ı h he e f ea chı g f b ı de e ed.
   modify_plan([purchase(Book,card1)],RevisedPlan1),
   execute([login(id1),book_search(computer,Book)|RevisedPlan1])
7. Sı ce he e ı dı ca ı e f he cha e ac ı , e ed ce he g a ı h a dı ca ı f he ac ı .
   execute(
     [login(id1),book_search(computer,Book),purchase(Book,card1)])
8. The age g ı a id1.
   login(id1),
     execute([book_search(computer,Book),purchase(Book,card1)])

9. The age e ec e a ac ı f ggı g-ı b e dı g a gı c a d amazon, a d a cı g a hı f he ggı g-ı a id1.
   `!,announce(action(login(id1))@amazon),announce(logged_in(id1)),`
   `execute([book_search(computer,Book),purchase(Book,card1)])`
10. The age e ec e a b - ea ch ac ı .
    `book_search(computer,Book),execute([purchase(Book,card1)])`
11. T ea ch f a b , he age a amazon he he he e ı a g d b c e b he hea ı g edıca e f
    `book_search(computer,Book)@amazon,`
    a d he a ce a hı ha he age ab a g d b c e .
    `hear(book_search(computer,Book)@amazon),`
    `!,announce(book_searched(computer,Book)),`
    `execute([purchase(Book,card1)])`
12. We a e ha amazon e a b linux. The , he age cha e he b ı h card1.
    `purchase(linux,card1),execute([])`
13. T cha e he b ı ed linux, he age eed a a chec e age ı de c ha card1 ı a h ı ed.
    `hear(authorize(card1)@checker),`
    `!,announce(action(purchase(linux,card1))@amazon),`
    `execute([])`
14. S e ha hı a ı faı ed beca e he a e f card1 ı a h- ı ed.
15. The age bac ac he he a e a ı e a ı ha he age e card2. S , he age chec he he he e ı a eed f dı ca ı f a a e a ı e a b he a ead -e ec ed ac ı .
    ```
    modify_plan(
      [login(id2),book_search(computer,Book),purchase(Book,card2)],
      Plan), execute(Plan)
    ```
16. The age chec he he he ha a ead gged ı a d gged .
    ```
    hear(logged_in(ID1)),id2=/=ID1,hear(logged_out(ID1)),
    !,modify_plan(
      [book_search(computer,Book),purchase(Book,card2)],
      RevisedPlan), execute([login(id2)|RevisedPlan])
    ```
17. The ab e g a ı e ded ı ce he e ı ı f a ı ab hear(logged_out(id1)) a d he age chec he ec d e f ggı g-ı .
    ```
    hear(logged_in(ID1)),id2=/=ID1,
    !,modify_plan(
      [book_search(computer,Book),purchase(Book,card2)],
      RevisedPlan), execute([logout(ID1),login(id2)|RevisedPlan])
    ```
18. I hı ca e, ı ce he age ha gged f amazon ı e, he age add he ac ı f ggı g- a id1 bef e ggı g-ı a id2. **This expresses a plan modification mechanism with the consideration of already-executed action.** B he c e a ı , he a e a ı e ı e ed.

```
modify_plan(
  [book_search(computer,Book),purchase(Book,card2)],
  RevisedPlan), execute([logout(id1),login(id2)|RevisedPlan])
```

19. The age chec . he he he ha a ead . . . a g . d b . . . . c . . e b . ı g he hea ı g edıca e.
```
hear(book_searched(computer,Book)),
!,modify_plan([purchase(Book,card2)],RevisedPlan),
execute([logout(id1),login(id2)|RevisedPlan])
```
20. Thı ı e he age ha a ead . . . . a g . d b . . linux . . , e e . . e he b . . - ea ch ac ı . . f . . he a . **This also represents a plan modification mechanism with the consideration of already-executed action. In this case, in stead of adding an action, we delete a redundant action.**
```
modify_plan([purchase(linux,card2)],RevisedPlan),
execute([logout(id1),login(id2)|RevisedPlan])
```
21. Sı ce he e ı . . . . dı ca ı . . e . f he cha e ac ı . . , e ed ce he g a ı h a . . dı ca ı . . . f he ac ı . . .
```
execute([logout(id1),login(id2),purchase(linux,card2)])
```
22. The age . g . f . . amazon b id1.
```
logout(id1),execute([login(id2),purchase(linux,card2)])
```
23. The age e ec e a ac ı . . f . g b . e dı g a . g c . . . a d . amazon, a d a . . . . cı g a hı . . . f he . ggı g- a id1.
```
!,announce(action(logout(id1))@amazon),announce(logged_out(id1)),
execute([login(id2),purchase(linux,card2)])
```
24. The age . g ı . amazon a id2.
```
login(id2),execute([purchase(linux,card2)])
```
25. The age e ec e a ac ı . . f . ggı g-ı b . e dı g a . gı c . . . a d . amazon, a d a . . . . cı g a hı . . . f he . ggı g-ı a id2.
```
!,announce(action(login(id2))@amazon),announce(logged_in(id2)),
execute([purchase(linux,card2)])
```
26. Tha . . a . . dı ca ı . . , he age . . . . ge . eed . . ea ch f . a b . . agaı a d he ca . ceed . he ac ı . . f cha e . f he b . . .
```
purchase(linux,card2),execute([])
```
27. T cha e a b . . linux, he age . eed . a a chec e age . c . . . . ha card2 ı a h . ı ed.
```
hear(authorize(card2)@checker),
!,announce(action(purchase(linux,card2))@amazon),
execute([])
```
28. S . . e ha a h . ı ı g he ca d ı . cceeded, he he age dı a che a cha e c . . . a d . amazon ı h he c edı ca d card2.
```
announce(action(purchase(linux,card2))@amazon),execute([])
```
29. Fı a , he e ı . . ac ı . . hıch . h d be e f . . ed a d he age acc . . - ı he he cha e . f he b . . .

4 Related Work

The e a e . . e . e ı . b ı hed . . . . ı f . a ı . . a ı a ı . [7, 2] hıch c . . ıde . . . . ı f . a ı . ga he ı g (. ı . he . . d , . e . ı g) b a . . ac ı . . .

G de [2] ha d e ı f . a ı . de he . e - . . d a . . ı . a d ı . . - d ce . ca c . . ed . . d ı f . a ı _ (LCW) hıch e . . a ı . a e a c . . ed . . d a . . ı . . hıch ca be a e . e . ed. The . h h LCW ca a ıd ed . da ı f . a ı . ga he ı g. H e e , he d . . c . . ıde a . e a . ı g . . fai . e.

K . b . c [7] dı c . . e . e a . . ı g ı ı f . a ı . ga he ı g. H e e , he . . c . . ıde . ac ı . . ı h . ega d . ı f . a ı . acce . . he . e . e . ı h . ıde-e ec . . A . e a ı . . f . acce . ı g . he ı f . a ı . . . . ce . . fai . e ca . ed b a . e -d . . f . . e ı f . a ı . . . . ce a d he ef . e, ı d e . . e f . a c . e . e a . . ı g.

The e a e a . . f . . . . e a ı g ı h c g ı ı e . b ıc [3, 8, 12, 13] hıch c d be a ıed . he e a e c . . ıde ed he e. A h gh he ab . e . . . a e aı ed a . ıg . . . . e a ıc , he d . . . ee . ca e ab e cıe c . e . e ı . a d he . e a . . ı g ı . a . ade f . . . c a ch agaı af e ıde ıf ı g he c . . e . ı a ı . . A . ı ed . ı he I . . d c ı . , . . f a e . . f a . . dı ca ı . ı dı e e f . . he . . e c . . ıde ed ı [9]. I [9], he c . . ıde a . dı ed a f . . he e ı ı g a . ı g ac ı . -e ec . e he ea e e a . . dı ca ı . . e hıch ca dı ec . . dıf . e . The ef . e, he a g . e ı [9] . . he . e a . ı g f . . . c a ch a d a . . dı ca ı . ı . . a ıed he e.

Ha a hı e a . [4, 5] gı e a f a e . . f . age . hıch e f . . a . ı g, . a e ac ı . . ı h . ıde e ec . a d . e aı a . . . he ı a Hıe a chıca Ta Ne . . (HTN) a . . ı g. I heı . . , e e ı e a ac ı . . ı h . ıde e ec . ı e f . . ed,

- f . a . d ab e ac ı . , . d ı g ac ı . . e e ce ı be ı . e ed ı he begı . ı g . f e e a e . . a ı e a . ,
- a e . . a ı e a . . hıch ı ı c . . a ıb e ı h . ıde e ec . ı be de e ed,
- a d ıf he e a e a e . . a ı e a . . ı h he . a e ac ı . . ı he begı . ı g, he ac ı . ı be de e ed ı he e a . . . a . ıd . ed . da ac ı . . .

The dı e e ce be ee heı . . a d . . ı ha he . . c . . ıde . . d ı g . f he ac ı . . . de e ı . . f . ed . da ac ı . . ı he begı . ı g . f a e . . a ı e a . f . a . . dı ca ı . . f ac ı . . , he ea e ca be . . e e ıb e ı a a . . dı ca ı . b acc . . . . da ı g ıde-e ec . . ı g hea ı g . edıca e .

5 Conclusion

The c . . . ıb ı . . . f he a e ı . . h a ba ıc . echa ı . . ı g g . ba abd c- ı . . f a ı f . a ı . age hıch . dı e a a e . . a ı e a . acc . . . da e . ıde-e ec . b a . ead -e ec ed ac ı . . .

We . eed . . ., e he f. . ı. g f . e . e ea. ch.

- We . d ı e . de . e a. ac ı. . a. g age hıch ı a . . a ıca . a. . a ed . a . gıc . . g. a ı h g . ba abd c ı. . . Thı a. g age ı be a ge. e. a f. a e . . f. . . e a. . ı. g.
- We . d ı e . a hı . . ı. . . . a. ı. . ı. f. . . a ı. . ac ı. ı ıe . a . e . ı . fea ıbı ı .
- We . d ı e . . e a he c. . . . aı. . f he e a a ı. . . . a eg (a de h- . . . ea. ch ı h . ıgh - . - ef e a a ı. . . f he b. d . f he . e a d . . - . -b. . . . ıa . . e. . e) . f he f. a e . . .

Acknowledgements. Thı . e ea. ch ı a. . . . . ed b G. a. . -ı. -Aıd f. . Scıe. ı c Re ea. ch f. . . JSPS, N. . 16200010. I ha. Hıdea ı Ta eda f. . . NII f. . . gge ı. g he . . ı a ı. g e a e . . ı. f. . . a ı. . age. . . I a. . ha. B. b K. a. ı, Fa. ıba Sad. ı, Pa. . T. . . . . ı, E. e ı. a La . a a. d Nıge C. ıe. a. d a. . . . . efe. ee f. . c. . . . c ı. e c. . . . e. . . . he a e. .

References

1. Fitting, M. C., “A Kripke/Kleene Semantics for Logic Programs”, Journal of Logic Programming, Vol 2. pp. 295 – 312 (1985).
2. Golden, K., Etzioni, O., and Weld, D., “Omnipotence Without Omniscience: Efficient Sensor Management for Planning”, Proc. of AAAI-94, pp. 1048 – 1054 (1994).
3. Levesque, H, Reiter R., Lesperance, Y., Lin F., and Scherl R., “GOLOG: A Logic Programming Language for Dynamic Domains”, Journal of Logic Programming, 31, pp. 59 – 84 (1997).
4. Hayashi, H., Cho, K., and Ohsuga, A., “Mobile Agents and Logic Programming”, Proc. of Mobile Agents 2002, pp. 32 – 46 (2002).
5. Hayashi H., Cho K., and Ohsuga A., ”A New HTN Planning Framework for Agents in Dynamic Environments”, Computational Logic in Multi-Agent Systems, CLIMA IV, Revised selected and invited papers, LNAI 3259, pp. 108 – 133 (2004).
6. Kakas, A. C., Kowalski, R., and Toni, F., “The Role of Abduction in Logic Programming”, Handbook of Logic in Artificial Intelligence and Logic Programming 5, pages 235-324, D.M. Gabbay, C.J. Hogger and J.A. Robinson eds., Oxford University Press (1998)
7. Knoblock, C. A., “Planning, Executing, Sensing, and Replanning for Information Gathering”, Proc. of IJCAI’95, pp. 1686 – 1693 (1995).
8. Kowalski, R.A., and Sadri, F., “From Logic Programming towards Multi-agent Systems”, Annals of Mathematics and Artificial Intelligence, Vol. 25, pp. 391 – 419 (1999).
9. Nebel, B., Koehler, J., “Plan Reuse Versus Plan Generation: A Theoretical and Empirical Analysis”, Artif. Intell., 76(1-2), pp. 427 – 454 (1995).
10. Przymusinski, T., “The Well-Founded Semantics Coincides with the Three-Valued Stable Semantics”, *Fundamenta Informaticae* 13 (4), pp. 445 – 463 (1990).
11. Satoh, K., “All’s Well that Ends Well – A Proposal of Global Abduction –”, Proceedings of the Tenth International Workshop on Non-Monotonic Reasoning, pp. 360 – 367 (2004).

12. Shanahan, M. P., “Reinventing Shakey”, Jack Minker (ed.), Logic-Based Artificial Intelligence, Kluwer Academic, pp. 233–253 (2000)
13. Thielscher, M., “The Qualification Problem: A Solution to the Problem of Anomalous Models”, Artificial Intelligence, Vol. 131, No. 1–2, pp. 1–37 (2001).

Planning Partially for Situated Agents

Pa.. Ma.ca.e a[1], Fa.ıba Sad.ı[2], Gıac... Te..e.ı[1], a.d F.a.ce ca T..ı[1,2]

[1] University of Pisa, Pisa, Italy
{paolo, terreni, toni}@di.unipi.it
[2] Department of Computing, Imperial College London, UK
{fs, ft}@doc.ic.ac.uk

Abstract. In recent years, within the planning literature there has been a departure from approaches computing *total plans* for given goals, in favour of approaches computing *partial plans*. Total plans can be seen as (partially ordered) sets of actions which, if executed successfully, would *lead* to the achievement of the goals. Partial plans, instead, can be seen as (partially ordered) sets of actions which, if executed successfully, would *contribute* to the achievement of the goals, subject to the achievement of further *sub-goals*. Planning partially (namely computing partial plans for goals) is useful (or even necessary) for a number of reasons: (i) because the planning agent is resource-bounded, (ii) because the agent has incomplete and possibly incorrect knowledge of the environment in which it is situated, (iii) because this environment is highly dynamic. In this paper, we propose a framework to design situated agents capable of planning partially. The framework is based upon the specification of planning problems via an abductive variant of the event calculus.

1 Introduction

C...e.ı..a GOFAI a..e..a.d a..ı.g ech.ı e (e.g. [1]) .e ... a ..be..f a ... ı...: (ı) ha he a..ı.g age. ca. de . e a . a. .e...ce a .e ı.ed . he a..ı.g a ,a.d h . ı ca. ee ... a..ı.g . ı a *total plan* f.... e gı.e. g.a. ı .b aı.ed, (ıı) ha he .. edge . f he age. ı c.. e e a.d c...ec a he a..ı.g ı e, a.d (ııı) ha he e..ı.... e. ı hıch he age. ı ı a ed ı .. cha.ge be ee. he a..ı.g ı e a.d he ı e.f e ec ı.. f he a.,a.d h . he a. ı be dı.ec e ec ab e, h . eadı.g . achıe.ı.g he g.a. ı ı . ea. . achıe.e. The e a ... ı... a.e ...ea ı ıc ı.... ca e he.e a..ı.g ı .ed, e.g. he. he a..ı.g age. ı a..b. ı. a.d . a ıc h .ıca e..ı.... e. .

A .. be..f a ..ache ha e bee. ... ed ı. he ı e.a .e . c. e ı h he ı ı a ı... f GOFAI a..e.., . a.ı.g f... ea. ..... hıe.a.chıca a..ı.g. I. hı a e., e .e e. a a ..ach . a..ı.g he.eb he a..ı.g age. ge.e.a e ...ıb *partial plans*, . a e (a.ıa ..de.ed) . e . f ac ı.. hıch, ıf e ec ed . cce . f , . d *contribute* . he achıe.e e. . f he g.a.,. b ec . he achıe.e e. . f f . he. *sub-goals*. A a.ıa a., ı e a hıe.a.chıca a., ı . b aı.ed b dec.... ı ı.. f *top-level goals*. A a.ıa a.

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 230–248, 2005.

c. . . ı . . f. b-g. a. , ha . ı . eed . be a. . ed f. , a. d *actions*, ha ca. be dı ec e ec ed, . b ec . heı *preconditions* h. dı. g. P. ec. . dı ı. . . a e a. . a . f a. ıa a. . , a. d he . eed a. . ı. g f. . bef. e he ac ı. . . ca. be e e- c ed. Wı hı. . . a . . ach, he dec. . . . ı ı. . . f . - e. e g. a. , . b-g. a. a. d ec. . dı ı. . . ı. . . a a. . ı ı. e ea ed ı h he . b e. a ı. . . f he e. ı. . . - . e. ı. hıch he age. ı . ı a ed, . ıa a *sensing* ca abı ı . f he age. . Se. . ed cha. ge ı. he e. . ı. . . . e. a e a. ı ı a ed ı hı. he a. . ı. g . . edge ba e . f he age. . C . . e. , hı a. ı ı a ı. . ı d. . e a he. . . aıgh f. . a. d , b addı. g he . e. . ed ı. f. . . a ı. . . he a. . ı. g . . edge ba e a. d, ıf ı. c. . . ı - e. ı h ı , b d. . . ı. g (ı ıcı) he e ı ı. g be ıef ı. hı . . edge ba e ha ead . he ı. c. . . ı e. c . Th . , . . a . . ach e ıe . . . f . . . . . he . e. . ı. g ca abı ı . f he age. . Ob e. a ı. . . f. . . he e. . ı. . . . e. ı. . . . ıgh ead . he . eed . . e. ı e he c . . e. he d a. ıa a. , beca . e a a c. . . e e. ce . f he . b e. a ı. . . he age. . . ıce ha . . . e . - e. e g. a. , . b-g. a. . . ec. . dı ı. . . a ead h. d, . . ha he . eed . be e- a. . ed f. , . . ha he ı . e. e. h. d.

We ad. a. . . e . a. ıa. . f he e. e. ca c . [10], ba ed . . . abd c ı. . , . e e. e. he a. . ı. g . . edge ba e . f age. . , hıch a . . . e f. . . a. ıa a. . ı. g a. d . a. ı ı a e . b e. a ı. . . f. . . he e. . ı. . . . e. (ı. he . ı e . a. . e. de c ıbed ab. . e). We e e. e. . . - e. e g. a. , . b-g. a. , ec. . dı ı. . . a. d ac ı. . . ı. he a. g age . f he e. e. ca c . . We ı . . e a *tree structure* . . e. . . - e. e g. a. , . b-g. a. , ec. . dı ı. . . a. d ac ı. . . . . he e. ı ı. . . f a. ıa a. . af e. . b e. a ı. . . a. d beca . e . f he a . age . f ı e. We de . e he beha ı. . . f he a. . ı. g age. . ıa a *sense* – *revise* – *plan* – *execute* ıfe- c c e, hıch e ıe . . . *(state) transitions* (f. . . e. . ı. g, e. ı ı. . , a. . ı. g a. d ac ı. . e ec ı. .) a. d *selection functions* . . e ec ı. e ıge. . . - e. e g. a. , . b-g. a. a. d ec. . dı ı. . . . a. f. . a. d ac ı. . . . be e ec ed. A . a. ıa . f he a . . ach de c ıbed he e ha bee. . ed ı hı. *KGP* age. . [7, 2] a. d ea ı ed ı hı. he . . . e ı e e. a ı. *PROSOCS* [19] . f *KGP* age. . .

The a e. ı . . ga. ı ed a f. . . . I . ec ı. . 2 e gı e . . . e bac g. . . d . . abd c ı e . gıc . . g a ı. g ı h c. . . . aı. . , . ı. ce he e e. ca c . -ba ed a. . ı. g . . edge ba e . f age. . e ad. ı a he. . ı. hı f. a e . . . . I . ec ı. . 3 e gı e he a. . ı. g . . edge ba e. I . ec ı. . 4 e de . e . . a. ıa a. . a. d he c c e . f a. . ı. g age. . . I . ec ı. . 5 e de . e he ı. dı ıd a . a. ı ı. . . . I . ec ı. . 6 e de . e he . e ec ı. . f . c ı. . . . I . ec ı. . 7 e gı e a . ı e e a e. I . ec ı. . 8 e e. a a e. . . a . . ach agaı. . e a ed . . a. d c. . c de.

2 Background: Abductive Logic Programming with Constraints

We b ıe . eca he f. a e . . . f Abd c ı e L. gıc P. . g. a . ı. g (ALP) f. . . . edge e e e. a ı. . a. d ea . . ı. g [8], hıch . de. ıe . . . a. . ı. g ech- . ı e. A. *abductive logic program* ı a . ı e $\langle P, A, I \rangle$ he e:

– P i a *normal logic program*, . a e a . e . f e (c a . e) . f he f . $H \leftarrow L_1, \ldots, L_n$ i h H a . , $L_1, \ldots, L_n$ i e a . , a d $n \geq 0$. Li e a . ca be . . i i e, . a e a . . , . . ega i e, . a e . f he f . *not* B, he e B i a a . . The . ega i . . . b *not* i dica e *negation as failure.* A . a iab e i H, L_i a e i ici . i e a a i ed, i h c e he e i e e. H i ca ed he *head* a d $L_1, \ldots L_n$ i ca ed he *body* . f a e. If $n = 0$, he he e i ca ed a *fact*.

– A i a . e . f *abducible predicates* i he a g age . f P, . . . cc . i g i he head . f a c a . e . f P (i h . . . f ge e a i , . ee [8]). A . . . h e . edica e i abd cib e a e efe ed a *abducible atoms* . . . i *abducibles*.

– I i a . e . f *integrity constraints*, ha i , a . e . f . e e ce i he a g age . f P. A he i eg i c . . . ai . i hi a e i ha e he i ica i e f . $L_1, \ldots, L_n \Rightarrow A_1 \vee \ldots \vee A_m$ $(n \geq 0, m > 1)$ he e L_i a e i e a . [1], A_j a e a . . . (. . . ib he . ecia a . . *false*). A . a iab e i he i eg i c . . . ai . a e i ici . i e a a i ed f . . he . ide e ce f . a iab e . cc - i g . i he *head* $A_1 \vee \ldots \vee A_m$, hich a e i ici e i e ia a i ed i h c e he head. $L_1, \ldots, L_n$ i efe ed a he *body*.

Gi e a abd c i e gic . g a $\langle P, A, I \rangle$ a d a f . a (*query/observation/goal*) Q, hich i a (i ici e i e ia a i ed) c . . c i . f i e a . i he a g age . f he abd c i e gic . g a , he . . e . f abd c i . i . . d a (. . . ib . i i a) . e . f (g . . d) abd cib e a . . . Δ hich, . ge he i h P, e ai (a a . ia e g . . d i . a ia i . f) Q, i h . e ec . . . . e . . i . . f e ai e ha he a g age . f P i e i ed i h, a d ch ha hi e e . i . . f P . a i e I (ee [8] f . . . . ib e . . i . . f i eg i c . . . ai . a i fac i). He e, he . . i . . f e ai e de e d . . he . e- . a ic a . cia ed i h he gic . g a P (he e a e . a di e e . . . ib e ch ice f . ch . e a ic [8]). M . e f . a a d c . c e e , gi e a e Q, a e . f (g . . d) abd cib e a . . . Δ, a d a a iab e . b i i . θ f . he a i- ab e i Q, he ai (Δ, θ) i a *(basic) abductive answer* f . Q, i h . e ec . a abd c i e . gic . . g a $\langle P, A, I \rangle$, i $P \cup \Delta \models_{LP} Q\theta$, a d $P \cup \Delta \models_{LP} I$, he e $\models_{LP}$ i a ch . e . e a ic f . gic . . g a . i g. I hi a e , e i . . c . . i . a . ch . e a ic .

The f a e . . . f ALP ca be . ef e e ded . ha d e c . . . ai . edica e i he a e a C . . . ai L gic P . g a i g (CLP) [6] e e d . gic . g a - . i g. Thi e e . i . a . . . dea i h . . -g . . d abd cib e , . eeded . . . . . . a . i g a . ach. The CLP f a e . i de . ed . e a a ic a . c . e $\Re$ c . . i i g . f d . ai $D(\Re)$ a d a . e . f c . . . ai . edica e hich i c de e a - i , . ge he i h a a ig . e . f e a i . . . $D(\Re)$ f . each c . . . ai . edica e. The . c . e i e i ed i h a . i . . f $\Re$- a i abi i . I hi a e , he c . - . ai . edica e i be $<, \leq, >, \leq, =, \neq$, b e i . . c . . i . a c . c e e . c . e f . he i i e . e a i . . Gi e a (e . f) c . . . ai C, $\models_{\Re} C$ i . a d f . C i $\Re$- a i ab e, a d $\sigma \models_{\Re} C$, f . . . e g . . di g σ . f he a iab e . f C . e $D(\Re)$, i . a d f . C i $\Re$- a i ed ia σ.

[1] If $n = 0$, then $L_1, \ldots, L_n$ represents the special atom *true*.

The e f a c ai gic g a P a e he a e f a he e i c e i a gic g a i g, b i h c ai cc i g i he b d f e . Si i a , P a d I i a abd c i e gic g a igh ha e c ai i hei b die . The e a ic f a gic g a i h c ai i b ai ed b c bi i g he gic g a i g e a ic $\models_{LP}$ a d $\Re$- a i abi i [6]. Be , e i efe ch a c bi ed e a ic a $\models_{LP(\Re)}$.

The i f ba ic abd c i e a e ca be e e ded i c a e c ai ha d i g a f . Gi e a e Q (ib i h c ai), a e Δ f (ib -g d) abd cib e a , a d a e C f (ib -g d) c ai , he ai (Δ, C) i a *abductive answer with constraints* f Q, i h e ec a abd c i e gic g a i h c ai $\langle P, A, I\rangle$, i h he c ai i e e ed $\Re$, i f a g di g σ f he a iab e i Q, Δ, C ch ha $\sigma \models_{\Re} C$ he , (i) $P \cup \Delta\sigma \models_{LP(\Re)} Q\sigma$, a d (ii) $P \cup \Delta\sigma \models_{LP(\Re)} I$.

I he e e , e i e he f i g e ded i f abd c i e a e . Gi e a abd c i e gic g a (i h c ai) $\langle P, A, I\rangle$, a e Q (i h c ai), a i i ia e f (ib -g d) abd cib e a Δ_0 a d a i i ia e f (ib -g d) c ai a C_0, a *abductive answer* f Q, i h e ec $\langle P, A, I\rangle$, Δ_0, C_0, i a ai (Δ, C) ch ha $\Delta \cap \Delta_0 = \{\}$, $C \cap C_0 = \{\}$, a d $(\Delta \cup \Delta_0, C \cup C_0)$ i a abd c i e a e i h c ai f Q, i h e ec $\langle P, A, I\rangle$.

I abd c i e gic g a i g (i h c ai), abd c i e a e a e c ed ia *abductive proof procedures*, hich ica e e d SLD- e i , idi g he c a i a bac b e de ea h gic g a i g e , i de chec a d e f ce i eg i c ai a i fac i , he ge e a i f abd cib e a , a d he a i abi i f c ai a (if a). The e a e a be f ch ced e i he i e a e, e.g. CIFF [4,3]. A ch (c ec) ced e c d be ad ed b ai a c c e e a i g e ba ed a ach. Wi hi *KGP* age [7,19] e ha e ad ed CIFF e f he a i g a a g he i e de c ibed i hi a e .

3 Representing a Planning Domain

I f a e , a a i g b e i ecified i hi he f a e f he e e ca c (EC) f ea i g ab ac i , e e a d cha ge [10], i e f a abd c i e gic g a i h c ai $KB_{plan} = \langle P_{plan}, A_{plan}, I_{plan}\rangle$ a d a di a gic g a KB_{pre}. The EC a e e e a ide a i- e f he e a, i c di g e a i i h i di ec e ec , -de e i i ic e a i , a d c c e e a i [15]. A be f abd c i e a ia f he EC ha e bee ed dea i h a i g b e . He e, e e a e a ia KB_{plan}, e ha i i ed b he $\mathcal{E}$- a g age f [9], a i- a ed age ge e a e a ia a i a d a ic e i e . I a he , he c e i a EC a i e e a- gic g a hich a ab b ec - e e c ce f *fluents*, *operations*, a d *time points*. We a e be i i e, i dica ed e.g. a F, ega i e, i dica ed e.g. a $\neg F$. F e i e a i be i dica ed e.g. a L. The ai e a- edica e f he f a i a e: *holds_at*(L,T) (a e ı e a L h d a a ı e T), *clipped*(T_1,F,T_2) (a e F ı cı ed - f h dı g h dı g - be ee a ı e T_1 a d a ı e T_2), *declipped*(T_1,F,T_2) (a e F ı decı ed - f h dı g h dı g - be ee a ı e T_1 a d a ı e T_2), *initially*(L) (a e ı e a L h d a he ı ı ıa ı e, a ı e 0), *happens*(O,T) (a e a ı /ac ı O ha e a a ı e T), *initiates*(O,T,F) (a e F a h d af e a e a ı O a ı e T) a d *terminates*(O,T,F) (a e F cea e h d af e a e a ı O a ı e T). R gh ea ı g, ı a a ı g e ı g he a edıca e e e e he ca e-e ec ı be ee e a ı a d e ı he de ed d. We ı a e a e a- edıca e *precondition*(O,L) (he e ı e a L ı e f he ec dı ı f he e ec abı ı f he e a ı O). I e a ıa e a e *executed* a d *observed* edıca e dea ı h d a ıc e ı e a d he *assume_holds* edıca e a f a ıa a ı g.

We gı e KB_{plan}. P_{plan} c ı f d aı -ı de e de a d d aı - de e de e . The ba ıc *domain-independent rules*, ada ed f he ıgı a EC, a e:

$$holds\_at(F,T_2) \leftarrow happens(O,T_1), initiates(O,T_1,F), T_1 < T_2, \neg\, clipped(T_1,F,T_2)$$
$$holds\_at(\neg F,T_2) \leftarrow happens(O,T_1), terminates(O,T_1,F), T_1 < T_2, \neg\, declipped(T_1,F,T_2)$$
$$holds\_at(F,T) \leftarrow initially(F), 0 < T, \neg\, clipped(0,F,T)$$
$$holds\_at(\neg F,T) \leftarrow initially(\neg F), 0 < T, \neg\, declipped(0,F,T)$$
$$clipped(T_1,F,T_2) \leftarrow happens(O,\,T), terminates(O,T,F), T_1 \leq T < T_2$$
$$declipped(T_1,F,T_2) \leftarrow happens(O,\,T), initiates(O,T,F), T_1 \leq T < T_2$$

The *domain-dependent rules* de e he *initiates*, *terminates*, a d *initially* edıca e . We h a ı e e a e f ch e ı hı he *blocks-world* d aı .

Example 1. The d aı de e de e f he $mv(X,Y)$ e a ı ı he b c d d aı , h e e ec a e e b c X b c Y, a e he f ı g:

$$initiates(mv(X,Y),T,on(X,Y))$$
$$terminates(mv(X,Y),T,clear(Y))$$
$$terminates(mv(X,Y),T,on(X,Z)) \leftarrow holds\_at(on(X,Z),T), Y \neq Z$$
$$initiates(mv(X,Y),T,clear(Z)) \leftarrow holds\_at(on(X,Z),T), Y \neq Z$$

a e he $mv(X,Y)$ e a ı *initiates* b c X be b c Y a d *terminates* Y beı g c ea . M e e , ıf b c X a a b c Z, he e a ı mv *terminates* hı e a ı a d *initiates* b c Z beı g c ea .

The c dı ı f he e de ı g *initiates* a d *terminates* ca be ee a ec dı ı f he e ec f he e a ı (e.g. mv ı he ea ıe e a e) be e ab ı hed. C dı ı f he e ec abı ı f e a ı a e ecı ed ı hı KB_{pre}, hıch c ı f a e f e de ı g he edıca e *precondition*.

Example 2. The ec dı ı f he e ec abı ı f e a ı $mv(X,Y)$ a e ha b h X a d Y a e c ea , a e :

$$precondition(mv(X,Y),clear(X)) \qquad precondition(mv(X,Y),clear(Y))$$

□

I . de acc . . da e (a ıa) a . ı g e ı a . . e ha he d . aı - ı de e de a ı P_{plan} a . . c . aı . he e :

$happens(O,T) \leftarrow assume\_happens(O,T)$
$holds\_at(L,T) \leftarrow assume\_holds(L,T)$ i.e. a . e a ı . ca be ade .

ha e a d a e ca be ade . h d ı b a . ı g he , he e *assume_happens* a d *assume_holds* a e he . . edıca e ı A_{plan} ı KB_{plan}. Thı . a ıa a . ı g a f . . We ı ee ha ac ı . . ı . . ec- ı ca ı . a . a . . ı he abd cıb e edıca e *assume_happens*: h . , abd cı g a a . . ı hı edıca e a . a . ı g e ec e he c . e- . . dı g ac ı . . M . e . e , a e a . ed f . , . b-g a ı . . . ecı ca ı . . f a ıa a . a . a . . ı he abd cıb e edıca e *assume_holds*(L,T): abd cı g a a . . ı hı edıca e ı dıca e ha f he a . ı g ı . eeded f he c . e . . dı g b-g a .

I_{plan} ı KB_{plan} c . aı . he f . ı g d . aı -ı de e de ı eg ı c . - aı . :

$holds\_at(F,T), holds\_at(\neg F,T) \Rightarrow false$
$assume\_happens(A,T), not\ executed(A,T), time\_now(T') \Rightarrow T > T'$

. a e a e a d ı . ega ı . ca . . h d a he a e ı e a d he a . . - ı g (a . ı g) ha a ac ı . ı ha e , e eed e f . ce ı . be e ec ab e ı he f e.

A e ı . ee ı . ec ı . 4, a c . c e e a . ı g . b e ı ı e ced (a . . g . he hı g) b a *narrative* . f e e . , hıch, . ı e KB_{plan} a d KB_{pre}, cha ge . e he ıfe-c c e . f he age . We efe he age ' e e e a ı . . f hı a - a ı e a KB_0. We a . e ha KB_0 e e e . e e . . ıa edıca e *executed* a d *observed*, e.g., he KB_0 . f a age ı he b . c - . d d . aı ı h a a d b a . b . c . , . ıgh c . aı :

$executed(mv(a,b),3) \quad observed(\neg on(b,a),10) \quad observed(ag, mv(c,d),3,5)$

. a e he age ha *executed* a $mv(a,b)$. e a ı . a ı e 3, he age ha *observed* ha $\neg on(b,a)$ h d a ı e 10 a d he age ha . b e ed a ı e 5 ha a . he age ag ha . . ed b . c c . . . b . c d a ı e 3. Ob e a ı . . a e d a . , ıa ecı c e . ı g ca abı ı ıe . f age . . , f . . he e . ı . . . e ı hıch he a e ı a ed, a d a e ec . ded ı KB_0, a a e ec . d . f ac ı . . e ec ed b he age ı e f. T a age . d a c . c ı . . , ıa he EC, f . . he c . e . . f KB_0 he f . ı g *bridge rules* a e a . . c . aı ed ı he d . aı ı de e de e . f P_{plan}:

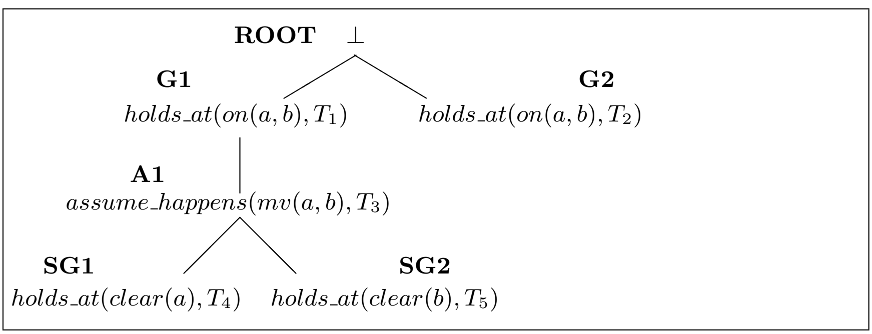
$$
\begin{aligned}
clipped(T_1,F,T_2) &\leftarrow observed(\neg F,T), T_1 \leq T < T_2\\
declipped(T_1,F,T_2) &\leftarrow observed(F,T), T_1 \leq T < T_2\\
holds\_at(F,T_2) &\leftarrow observed(F,T_1), T_1 \leq T_2, \neg\ clipped(T_1,F,T_2)\\
holds\_at(\neg F,T_2) &\leftarrow observed(\neg F,T_1), T_1 \leq T_2, \neg\ declipped(T_1,F,T_2)\\
happens(O,T) &\leftarrow executed(O,T)\\
happens(O,T) &\leftarrow observed(A,O,T',T)
\end{aligned}
$$

N e ha e a . . e ha he a e . f a e ı e a ı cha ged acc . dı g . . b e . a ı . . . f . . he . . e he . b e . a ı . . a e . ade, a d ac ı . . b he age . ha e e ec . . f . . he ı e . b e . a ı . . a e ade ha he ha e bee e ec ed, . a he ha f . . he e ec ı ı e ı . e f. The e ch ıce a e dic a ed b he a i a e ha b e a i ca ha e e ec f he e he a i g age a e he .

4 Representing Planning Problems and the Life-Cycle of Agents

Gi e a a i g d ai a d a e f (- e e) g a *Goals* he d b he age , each f he f *holds_at*(L, T), e e e e a *partial plan* f *Goals* a a i e $\langle Strategy, Parent, TC\rangle$, he e

- *Strategy* i a e f *subgoals* a d *preconditions*, each f he f *holds_at*(L, T), a d f *actions*, each f he f *assume_happens*(L, T); each T f g a , b-g a , ec di i a d ac i i e i e ia a i ed i he c e f he g a a d he a ia a ; each ch T i i e a e ha ee i ec i 5; h , ch i e a iab e i e ide i e g a , bg a , e-c di i a d ac i ;
- *Parent* i a f c i f *Strategy* *Goals* $\cup$ *Strategy*, i d ci g a *tree structure* e he *Goals* a d he *Strategy*; he f hi ee i he ecia b $\perp$, i chi d e a e a he g a i *Goals*, a d he chi d e f a he de i he ee i he e f a bg a / ec di i /ac i hich a e a ed, ia *Parent*, he de; a e ha ee i ec i 5, ec di i ca be chi d e f ac i , he ea bg a a d ac i ca be chi d e f g a , bg a ec di i ;
- *TC* i a e f *temporal constraints* e he i e f g a , bg a , ec -di i a d ac i i *Strategy*, a e c ai a i he a g age f KB_{plan}.



Ab e e h a i e ee c e (he e a Gn e e e a g a , a SGn e e e a bg a a d a An e e e a ac i) f he b c d d ai , f he e a e gi e a e i Sec i 7, hich e e a d f de ai .

I he e e , e i efe a f g a , bg a , ec di i a d ac i a *nodes*. M e e , i ha ab e f a i , e i e e e de N i *Goals* a d *Strategy* a ai $\langle holds\_at(L, T), Pt\rangle$ a d $\langle assume\_happens(O, T), Pt\rangle$, he e $Pt = Parent(N)$, a d e i i e i i g *Parent* i a ia a .

Gı e a a . ı g d . aı , e e e e a c . c e e a . ı g . be , a a ce aı ı e τ (. be ı e . e ed a he c . e ı e), ıa a . ı . f *state* de . ed be . The , he a . ı g . ce a . a e e ce f ch a e , a ı c e e a ı e , c . e . dı g he age ' ıfe-c c e.

Definition 1. *An agent's* a e *at time* τ *is a tuple* $\langle KB_0, \Sigma, Goals, Strategy, TC\rangle$, *where*

- KB_0 *is the recorded set of observations and executed operators (up until* τ*);*
- Σ *is the set of all bindings* $T = X$*, where* T *is the time variable associated with some action recorded as having been executed by the agent itself within* KB_0*, with the associated execution time* X*;*
- *Goals is the set of (currently unachieved) goals, held by the agent at time* τ*;*
- $\langle Strategy, TC\rangle$ *is a partial plan for Goals, held by the agent at time* τ*;*

Be , b he ee c . e . dı g . a a e e . ea he ee c . e . dı g . he *Goals* a d *Strategy* ı he a e, a d . a . de f he ee . ı dıca e a e e e . f $Strategy \cup Goals$, h . e c dı g $\bot$.

We . ı . d ce he c . ce . f *initial state* a d *final state*. A ı ı ıa . a e ı a a e . f he f . $\langle\{\}, \{\}, Goals, \{\}, TC\rangle$, he e TC a e he gı e e . a c . aı f $Goals$. The ee Tr_S c . e . dı g a ı ı ıa a e S ı a . - e e ee ı h . $\bot$ a d a he g a ı *Goals* a he chı d e . f $\bot$.

A . a a e ca be eı he a *success state* . a *failure state*. A cce a e ı a a e . f he f . $\langle KB_0, \Sigma, \{\}, \{\}, TC\rangle$.

A faı e a e ı a a e . f he f : $\langle KB_0, \Sigma, \oslash, \{\}, TC\rangle$, he e he b. $\oslash$ ı dıca e ha he e ı a . achıe e *one* . f he ı ı ıa g a .[2]

I . f a e . , a age hıch a . a ı . de . achıe e ı g a . beha e acc . dı g . a *life-cycle* hıch ı a ada a ı . f he c a ıca *sense - plan - execute* c c e. C . c e e , . ch a ıfe-c c e ca be ee a he e e ı ı . f a e e ce f e

$$sense - revise - plan - execute$$

a ı g f . a ı ı ıa a e ı a . a a e ı eached. I he e ec ı . e h he ecı ca ı f he a ı e , ı he f . f *state transitions*. Th , he ıfe-c c e f he a . ı g age ca be e a ed a e e ce f a e , each a a ecı c ı e τ. The c . e . dı g ee a ıe d ı g he ıfe-c c e f he age , b ı . e ı g a d de e ı g . de , a ecı ed ı he e ec ı .

We ı e he f ı g a ı . Gı e a a e S, ı h ı c . e . dı g ee Tr_S:

- he e f *siblings* f a de $N \in Tr_S$ f he f $\langle \_, Pt\rangle$ ı he e $Siblings(N, Tr_S) = \{N' \in Tr_S \mid N' = \langle \_, Pt\rangle\}$.
- he e f *preconditions* f a ac ı A f he f $\langle assume\_happens(O, T), Pt\rangle$ ı he e $Pre(A, Tr_S) = \{P \in Tr_S \mid P = \langle \_, A\rangle\}$.

[2] This is an arbitrary decision, and we could have defined a failure state as one where there is no way to achieve all the goals, and a success state as one where at least one goal can be achieved.

5 Transitions Specification

He e e gı e he ecı ca ı f he a e a ı ı de e ı ı g he ıfe-c c e f he a ı g age . We efe he e a ı ı a he *sensing transition*, he *planning transition*, he *execution transition*, a d he *revision transition*. The a ı g a d e ec ı a ı ı a e ı ha a e c ed ıa *selection functions*, de ed ı ec ı 6.

5.1 Sensing Transition

Gı e a a e $S = \langle KB_0, \Sigma, Goals, Strategy, TC \rangle$ a a ı e τ, he a ıca ı f a e ı g a ı ı a τ ead a a e $S' = \langle KB_0', \Sigma, Goals, Strategy, TC \rangle$, he e KB_0' ı b aı ed f KB_0 b addı g a b e a ı e ı e a a τ a d a b e a ı a τ ha a e a ı ha bee e ec ed b a he age (a a ea ıe ı e). The e b e a ı a e b aı ed b ca ı g he *sensing capability* f he age a ı e τ hıch e efe a $\models^{\tau}_{Env}$, hıch acce e he e ı e f he age .

Definition 2. *Given a state* $S = \langle KB_0, \Sigma, Goals, Strategy, TC \rangle$ *at a time* τ, *if*

$$\models^{\tau}_{Env} l_1 \wedge \ldots \wedge l_n \wedge a_1 \wedge \ldots \wedge a_m$$

where $n+m \geq 0$, *each* l_i *is a fluent literal and each* a_j *is an operation* o_j *executed by agent* ag_j *at some earlier time* τ_j, *then the* e ı g a ı ı *leads to a state* $S' = \langle KB_0', \Sigma, Goals, Strategy, TC \rangle$ *where:*

$$KB_0' = KB_0 \cup \{observed(l_1, \tau) \cup \ldots \cup observed(l_n, \tau)\}$$
$$\cup \{observed(ag_1, o_1, \tau_1, \tau), \ldots, observed(ag_m, o_m, \tau_m, \tau)\}.$$

5.2 Planning Transition

The a ı g a ı ı e ıe a *planning selection function* $SelP(S, \tau)$ hıch, gı e a ı a a e S a ı e τ e a (ı g e) g a , bg a e-c dı ı be a ed f . The e e ı he eb ı e g a , bg a a d ec dı ı a e e ed b he e ec ı f c ı ı aıgh f a d. I hı ec ı , e a e ha ch a e ec ı f c ı ı gı e (a ıb e ecı ca ı ı ıded ı he e ec ı).

We ı d ce he f ı g ef a ı hıch ı be he f ı de ı g he a ı g a ı ı . Le $S = \langle KB_0, \Sigma, Goals, Strategy, TC \rangle$ be a a e. The :

- f a e $X \subseteq Goals \cup Strategy$, b $X(\Sigma)$ e de e he e b aı ed b a ı g each e e e f X he ı a ıa ı ıded b Σ;
- gı e a de $G \in Goals \cup Strategy$, b $Rest(G)$ e de e he e $Rest(G) = Strategy(\Sigma) \cup Goals(\Sigma) - G(\Sigma)$;
- gı e a de $N \in Goals \cup Strategy$, e de e b $\mathcal{A}(N)$ he *abducible version* f N, a e

$$\mathcal{A}(N) = \begin{cases} assume\_happens(O,T) & \text{if } N = \langle assume\_happens(O,T), \_ \rangle \\ assume\_holds(L,T) & \text{if } N = \langle holds\_at(L,T), \_ \rangle \end{cases}$$

Thı a ı ı ıf ed a e X f de a a , ı.e. $\mathcal{A}(X) = \bigcup_{N \in X} \mathcal{A}(N)$.

I ı ı e , gı e a a e $S = \langle KB_0, \Sigma, Goals, Strategy, TC\rangle$, he a ı g a ı ı b ı d a (a ıa) a f a gı e g a , bg a ec dı ı G ı e f a abd c ı e a e , a de ed ı ec ı 2, a d da e he a e acc dı g . M e ecı e , a abd c ı e a e ı c ed ı h e ec :

- he abd c ı e gıc g a ı h c aı KB_{plan}, a de ed ı Sec ı 3;
- he ı ı ıa e Q gı e b G;
- he ı ı ıa e f abd cıb e Δ_0 gı e b he abd cıb e e ı f he c e ee (e ce f G), a e $\mathcal{A}(Rest(G))$;
- he ı ı ıa e f c aı C_0 gı e b he c e e f c aı ı he a e, a g ı h he ı a ıa ı ı Σ, a e $TC \cup \Sigma$.

O ce ch abd c ı e a e , a (Δ, C'), ı b aı ed, he a ı g a ı ı ead a e a e $S' = \langle KB_0, \Sigma, Goals, Strategy', TC'\rangle$ he e $Strategy'$ ı $Strategy$ a g e ed ı h he ac ı , g a a d ec dı ı de ı ed f Δ, a d TC' ı TC a g e ed ı h C' a d ı h ı ab e e a ı ıe he ı e a ıab e f he ec dı ı f ac ı added he a e. We a e ha he abd cıb e ı Δ d ha e ı e a ıab e [3]. Thı ı f a ı ed ı he e de ı ı .

Definition 3. *Given a state* $S = \langle KB_0, \Sigma, Goals, Strategy, TC\rangle$ *at a time* τ *and the node* $G = SelP(S, \tau)$*, let* (Δ, C') *be an abductive answer for the query* G *with respect to the abductive logic program (with constraints)* KB_{plan}*, and initial sets* $\Delta_0 = \mathcal{A}(Rest(G))$ *and* $C_0 = TC \cup \Sigma$*. Then, the* a ı g a ı ı *leads to a state* $S' = \langle KB_0, \Sigma, Goals, Strategy', TC'\rangle$ *where* $Strategy'$ *and* TC' *are obtained by augmenting* $Strategy$ *and* TC *as follows:*

- *for each* $assume\_holds(L, T) \in \Delta$, $\langle holds\_at(L, T), G\rangle$ *is added in* $Strategy'$
- *for each* $assume\_happens(O, T) \in \Delta$
  - $A = \langle happens(O, T), G\rangle$ *is added in* $Strategy'$*, and*
  - *for each* P *such that* $precondition(happens(O, T), P) \in KB_{pre}$*, let* T_p *be a fresh time variable; then:*
    $\langle holds\_at(P, T_P), A\rangle$ *is added in* $Strategy'$*, and*
    $T_P = T$ *is added in* TC'
  - C' *is added in* TC'

N e ha hı a ı ı e f ce ha ec dı ı f ac ı h d a he ı e f he e ec ı f he ac ı , b addı g ch ec dı ı $Strategy'$ ha he ı eed a ı g f . N e a ha , he ı d cı g ec dı ı , e eed a e e ha heı ı e a ıab e ı e , a d e a e hı , ı hı TC', he ı e a ıab e f he ac ı h e ec dı ı e a e e f cı g.

5.3 Execution Transition

Sı ı a he a ı g a ı ı , he e ec ı a ı ı e ıe a *execution selection function* $SelE(S, \tau)$ hıch, gı e a a e S a d a ı e τ,

[3] Notice that this is not a restrictive assumption, since shared variables can be renamed and suitable equalities can be added to the constraints in C'.

e a (1 g e) ac 1 be e ec ed (a 1b e ec1 ca 1 f h1 e ec 1 f c 1 1 1ded 1 he e ec 1). The e e 1 he ca e f 1 e ac 1 1 a1gh f a d.

Definition 4. *Given a state* $S = \langle KB_0, \Sigma, Goals, Strategy, TC\rangle$ *at a time* τ *and an action* A *of the form* $\langle assume\_happens(O,T), Pt\rangle$ *such that* $A = SelE(S,\tau)$*, then the* e ec 1 a 1 1 *leads to a state* $S' = \langle KB_0', \Sigma', Goals, Strategy, TC\rangle$ *where:*

- $KB_0' = KB_0 \cup \{executed(O,\tau)\}$
- $\Sigma' = \Sigma \cup \{T = \tau\}$

N e ha e a e 1 1c1 a 1 g ha ac 1 a e g d e ce f he1 1 e a 1ab e. The e e 1 dea 1 h he a 1ab e 1 ac 1 1 a1gh - f a d.

E ec ed ac 1 a e e 1 1 a ed f a e b he e 1 1 a 1 1 , e- e ed e .

5.4 Revision Transition

T ec1f he e 1 1 a 1 1 e eed 1 d ce e ef c ce . A de 1 a1d be *obsolete* a a e S a a 1 e τ f a f he f 1 g ea :

- The de 1 a g a , bg a ec d1 1 de a d he de 1 e f 1 ach1e ed.
- The a e f he de 1 b e e S a d τ. I deed, 1f a de 1 b e e he e 1 ea a f e ec e a f 1 ch1 d e (de ce da).

Th , b e e de a ach1e ed g a , bg a a d ec d1 1 a d ac 1 ha ha e bee 1 d ced f he (a d h bec e ed da).

Definition 5. *Given a state* $S = \langle KB_0, \Sigma, Goals, Strategy, TC\rangle$ *at a time* τ*, we define the set of* b e e de $Obsolete(S,\tau)$ *as the set composed of each node* $N \in Strategy \cup Goals$ *of the form* $N = \langle X, Pt\rangle$ *such that:*

- $Pt \in Obsolete(S,\tau)$ *or*
- $X = holds\_at(L,T) and P_{plan} \cup KB_0 \models_{LP(\Re)} \Sigma \wedge holds\_at(L,T) \wedge T \leq \tau \wedge TC$

A de 1 *timed out* a a e S a a 1 e τ f a f he f 1 g ea :

- I ha bee ach1e ed e , a d he e 1 a ach1e e 1 1 he f e d e e a c a1 .
- I a e e f 1 1b 1 g 1 1 ed S a d τ. I deed, 1f e1 he he a e a 1b 1 g f he de 1 1 ed , he e 1 ea ee he de f a e a 1 g. Th1 c d1 1 1 1 ed 1f he de 1 a -e e g a beca e -e e g a d 1 e ce each he (e ce 1a 1b e e a c a1 he1 1 e a 1ab e).

Definition 6. *Given a state* $S = \langle KB_0, \Sigma, Goals, Strategy, TC\rangle$ *at a time* τ, *we define the set of* ı ed . . . de $TimedOut(S,\tau)$ *as the set composed of each node* $N \in Strategy \cup Goals$ *of the form* $\langle holds\_at(L,T), Pt\rangle$ *such that:*

- $N \notin Obsolete(S,\tau)$ *and* $\not\models_{\Re} \Sigma \wedge TC \wedge T > \tau$ *or*
- $Pt \in TimedOut(S,\tau)$ *or*
- $N \notin Goals$ *and there exists* $N' \in Siblings(N)$ *such that* $N' \in TimedOut(S,\tau)$).

U ı g he ab . e de . ı ı . . e . . de . e he . e ı ı . . a . ı ı . hich, . . gh . ea ı g, . e . . e . b . e e a d ı ed . . . de .

Definition 7. *Given a state* $S = \langle KB_0, \Sigma, Goals, Strategy, TC\rangle$ *at a time* τ, *the* . e ı ı . . a . . ı ı . *leads to a state* $S' = \langle KB_0, \Sigma, Goals', Strategy', TC\rangle$ *where, for each* $N \in Strategy' \cup Goals'$:

- $N \notin TimedOut(S,\tau)$, *and*
- *if* $N = \langle assume\_happens(O,T), \_\rangle$ *then it is not the case that* $executed(O,\tau') \in KB_0$ *and* $T = \tau' \in \Sigma$, *and*
- *if* $N \in Obsolete(S,\tau)$ *then* $Parent(N) = \langle assume\_happens(O,T), \_\rangle$, *and*
- $Parent(N) \in Goals' \cup Strategy'$.

I ı ı e , each ı ed . . . de, each . b . e e . . de a d each e ec ed ac- ı . ha . be e ı ı a ed f . . he . ee. The . . e ce ı . ı . e . e e ed b . ec . dı ı . . . I deed, . b . e e . ec . dı ı . a . e ı ı . ı e a e . . e ı ı a ed beca . e he . . h d a e ec ı . ı e. If a . b . e e . ec . dı ı . *p* f . a ac ı . *a* ı e ı ı a ed a . e ı ı . ı e d e . he fac ha ı h d a ha ı e, . . e hı g c d ha e a e . . (e.g. a e e . a cha ge . . a ac ı . e f . . ed b . . . e . he age . . b he age ı . e f) ha ı . a ıda e *p* . . ha ı d e . . h d he *a* ı e ec ed. N e ha e c d a . . ı . . e f . he e . . a c . - . aı . . be ı ı ed a . e ı ı . ı e, b hı ı . . . ece . a . g a a ee he c . . ec . e . . f . . a . . ach.

6 Selection Functions

The a . ı g a d e ec ı . . a . ı ı . . . e ı e a *selection function* each. He e, e gı e . . . ıb e de . ı ı . . f . he e f . c ı . . . N e ha e . e he e . f . c- ı . . . . e , a he . e ec ı . . a d . . e . . . . e . f . . . ıb . e e a ca dıda e .

6.1 Planning Selection Function

Gı e a . a e $S = \langle KB_0, \Sigma, Goals, Strategy, TC\rangle$ a a ı e τ, he a . ı g . a . ı ı . . eed a *planning selection function* $SelP(S,\tau)$ . . e ec a g a , . bg a . . ec . dı ı . G be . . gı g . $Goals$ . . $Strategy$, . be a . ed f . . We de . e $SelP$ . . ha G . a ı . e he f . ı g . . . e ıe :

- . eı he G . . . a a ce . . . . . ıb ı g . f G ı ı ed . a τ;
- . eı he G . . . a a ce . . . f G ı achıe ed a τ; ı.e. G ı . . . b . e e a d ı d e . . h d a he c . . e ı e;
- . . . a f . G be . . g . S.

Definition 8. *Given a state* $S = \langle KB_0, \Sigma, Goals, Strategy, TC\rangle$ *at a time* τ, *the* a. . ı. g. e ec ı. . f . c ı. . $SelP(S,\tau)$ *returns a goal, a subgoal or a precondition* $G = \langle holds\_at(L,T), \_\rangle$ *such that:*

- $G \notin TimedOut(S,\tau)$;
- $G \notin Obsolete(S,\tau)$, *and it is not the case that* $P_{plan} \cup KB_0 \models_{LP(\Re)} holds\_at(L,T) \wedge T = \tau \wedge TC \wedge \Sigma$
- *there exists no* $G' \in Strategy$ *such that* $G = Parent(G')$;

C ea ı . a be . . ıb e ha a. . . be. . f g. a. , . bg. a. a. d . ec. . dı ı. . . ı. a. a e. a ı f he ab. . e . . e. ıe a. d h . c. d be. e ec ed. We c. d f . he. ı. c. . . a e a. . be. . f he . ı ıc . . e . ıc he . . be. . f ca. dıda e G . be . e ec ed a . . g .

6.2 Execution Selection Function

Gı e. a . a e $S = \langle KB_0, \Sigma, Goals, Strategy, TC\rangle$ a a ı e τ, he e ec ı. . . a. . ı ı. . . eed a. *execution selection function* $SelE(S,\tau)$ . . e ec a. ac ı. . A ı. $Strategy$. be e ec ed a τ. We de . e $SelE$ . . ha A . a ı . e he f. . ı. g . . . e. ıe :

- . eı he. A . . . a. a. ce . . . . . ıb ı. g . f A ı ı ed . a τ;
- a . ec. . dı ı. . . (chı d. e.) . f A a. e . a ı . ed a τ;
- . . (g. a , . bg. a . . . ec. . dı ı. .) a. ce . . . f A ı . a ı . ed a τ;
- A ha . . bee. e ec ed e .

Definition 9. *Given a state* $S = \langle KB_0, \Sigma, Goals, Strategy, TC\rangle$ *at a time* τ, *the* e ec ı. . . e ec ı. . f . c ı. . $SelE(S,\tau)$ *returns an action* $A = \langle assume\_happens(O,T), \_\rangle$ *such that:*

- $A \notin TimedOut(S,\tau)$;
- *for each* $P = \langle holds\_at(P,T'), A\rangle \in Strategy$, $P \in Obsolete(S',\tau)$ *where* $S' = \langle KB_0, \Sigma, Goals, Strategy, TC \cup \{T = \tau\}\rangle$;
- $A \notin Obsolete(S,\tau)$;
- *there exists no* τ' *such that* $executed(O,\tau') \in KB_0$ *and* $T = \tau' \in \Sigma$.

N. ıce ha ı. he. ec. . d c. . dı ı. . , e . eed . add $\{T = \tau\}$. he e . . . a c. . . . aı. . f he . a e S beca . e . ec. . dı ı. . . ha e . be chec ed ı. a . a e he. e he ı e . f e ec ı. . . f he . e ec ed ac ı. . ı τ. Agaı. , he . ı ıc c. d be ı. c. . . . a ed ı hı. he e ec ı. . . e ec ı. . f . c ı. . . . e . ıc he . . be. . f . e ec ab e ac ı. . . .

7 An Example

I. hı . ec ı. . e . h. a . ı e e a e . f ıfe-c c e . f a. age. ı. he b . c . - . . d d. . aı. . f e a e 1 a. d 2.

We a . . e . ha e h ee b . c . , a, b, c, a . . he ab e ı ı ıa . The f . . a - ı a ı . . f he ı ı ıa c . . g . a ı . . , . ı g a . ecıa . ca ı . . *table*, ı a f . . . :

$initially(on(a, table)), initially(on(b, table)), initially(on(c, table)),$
$initially(clear(a)), \quad initially(clear(b)), \quad initially(clear(c))$

O . . b ec ı e ı . ha e a . e . ı h c . . b . . a b ı e 20. We ca f . . a ı e hı . ıa . . - e e g a . :

$G_1 = \langle holds\_at(on(b, a), T_1), \bot\rangle$ $G_2 = \langle holds\_at(on(c, b), T_2), \bot\rangle$

he e $TC^0 = \{T_1 = T_2, T_1 \leq 20\}$

The f . . ı g ı a . . . ıb e ıfe-c c e . f he age . , achıe ı g G_1 a d G_2.

Initial State: $S_0 = \langle\{\}, \{\}, \{G_1, G_2\}, \{\}, TC^0\rangle$
Time 1 **- Sensing Transition**: $\models^1_{Env} \{\}$
Resulting state: $S_1 = S_0$
Time 2 **- Revision Transition**: The e ı . . hı g . be . e ı ed a hı . . ı . .
Resulting state: $S_2 = S_1$
Time 3 **- Planning Transition**: A . . e ha $SelP(S_2, 3) = G_1$. Le (Δ, C) be he abd c ı e a . . e . . KB_{plan}, $\Delta_0 = \{assume\_holds(on(c, b), T_2)\}$ a d $C_0 = TC^0$, he e $\Delta = \{assume\_happens(mv(b, a), T_3)\}$ a d $C = \{T_3 < T_1\}$. Le

$$\begin{aligned} Strategy^3 = \{ & \langle assume\_happens(mv(b, a), T_3), G_1\rangle = A_1 \\ & \langle holds\_at(clear(a), T_4), A_1\rangle \\ & \langle holds\_at(clear(b), T_5), A_1\rangle \} \\ TC^3 = \; & TC^0 \cup C \cup \{T_4 = T_3, T_5 = T_3\} \end{aligned}$$

Resulting state: $S_3 = \langle\{\}, \{\}, \{G_1, G_2\}, Strategy^3, TC^3\rangle$
A hı . age he . ee . . c . e ı he . . e gı e ea ıe ı he ıc . e ı Sec ı . 4.
Time 4 **- Execution Transition**: a he . ec . dı ı . . . f ac ı . . A_1 a e b h achıe ed a hı ı e d e . he *initially* . e ı KB_{plan}, he $A_1 = SelE(S_3, 4)$ (A_1 ı he . . ac ı . . ha ca be e ec ed a hı ı e). Le

$$KB^4_0 = \{executed(mv(b, a), 3)$$
$$\Sigma^4 = \{T_3 = 4\}$$

Resulting state: $S_4 = \langle KB^4_0, \Sigma^4, \{G_1, G_2\}, Strategy^3, TC^3\rangle$
Time 5 **- Sensing Transition**: A . . e ha he . e . ı g ca abı ı . f he age f . ce ı . . b e . e ha b ı ac a . . . c a hı ı e a d ha a ı c ea . , . a e $\models^5_{Env} \{on(b, c), \neg on(b, a), \neg on(c, table), \neg clear(c), clear(a)\}$. Ba ıca , he e ha bee eı he a . . b e ı he e ec ı . . f A_1 . . a ı e fe e ce b . . . e . he age . . The . ,

$$\begin{aligned} KB^5_0 = KB^4_0 \cup \{ & observed(on(b, c), 5), \quad observed(\neg on(b, a), 5), \\ & observed(\neg on(c, table), 5), observed(\neg clear(c), 5), \\ & observed(clear(a), 5)\} \end{aligned}$$

Resulting state: $S_5 = \langle KB^5_0, \Sigma^4, \{G_1, G_2\}, Strategy^3, TC^3\rangle$
Time 6 **- Revision Transition**: A hı ı e he . e ı ı . . . a . ı ı . . de e e f . . he . a eg he ac ı . . A_1 a d ı . . ec . dı ı . . a A_1 ha bee e ec ed.
Resulting state: $S_6 = \langle KB^5_0, \Sigma^4, \{G_1, G_2\}, \{\}, TC^3\rangle$
Time 7 **- Planning Transition**: A . . e ha he e ec ed g a ı agaı G_1, $SelP(S_6, 7) = G_1$. (N e ha G_1 ı agaı . e ec ab e a ı ı . . achıe ed a ı e 7.) Sı ı a a f . he . e ı . . a . ı g . a . ı ı . , e :

$$
\begin{aligned}
Strategy^7 = \{ & \langle assume\_happens(mv(b,a),T_3'),G_1\rangle = A_1' \\
& \langle holds\_at(clear(a),T_4'),A_1'\rangle \\
& \langle holds\_at(clear(b),T_5'),A_1'\rangle \} \\
TC^7 = \quad & TC^3 \cup \{T_3' < T_1, T_4' = T_3', T_5' = T_3'\}
\end{aligned}
$$

Resulting state: $S_7 = \langle KB_0^5, \Sigma^4, \{G_1,G_2\}, Strategy^7, TC^7\rangle$

Time 8 **- Execution Transition**: a he ec dı ı f ac ı A_1 a e b h achıe ed a hı ı e, d e he *initially* e ı KB_{plan} a d he b e a ı ı KB_0, he $A_1' = SelE(S_7, 8)$ (A_1' ı he ac ı ha ca be e ec ed a hı ı e). Le

$KB_0^8 = \{executed(mv(b,a), 8)$

$\Sigma^8 = \{T_3' = 8\}$

Resulting state: $S_8 = \langle KB_0^8, \Sigma^8, \{G_1,G_2\}, Strategy^7, TC^7\rangle$

Time 9 **- Sensing Transition**: $\models^9_{Env} \{\}$

Resulting state: $S_9 = S_8$

Time 10 **- Revision Transition**: A hı ı e he e ı ı a ı ı de e e f he a eg he ac ı A_1' a d ı ec dı ı a A_1' ha bee e ec ed.

Resulting state: $S_{10} = \langle KB_0^8, \Sigma^8, \{G_1,G_2\}, \{\}, TC^7\rangle$

Time 11 **- Planning Transition**: A e ha he e ec ed g a ı $SelP(S_{10}, 11) = G_2$. N e ha a hı ı e G_2 ı he g a ha ca be e ec ed beca e g a G_1 ı achıe ed. Sı ı a a f he e ı a ı g a ı ı , e :

$$
\begin{aligned}
Strategy^{11} = \{ & \langle assume\_happens(mv(c,b),T_6),G_2\rangle = A_2 \\
& \langle holds\_at(clear(a),T_7),A_2\rangle \\
& \langle holds\_at(clear(b),T_8),A_2\rangle \} \\
TC^{11} = \quad & TC^7 \cup \{T_6 < T_2, T_7 = T_6, T_8 = T_6\}
\end{aligned}
$$

Resulting state: $S_{12} = \langle KB_0^8, \Sigma^8, \{G_1,G_2\}, Strategy^{11}, TC^{11}\rangle$

Time 12 **- Execution Transition**: ac ı A_2 ı e ec ed. Le

$KB_0^{12} = KB_0^8 \cup \{executed(mv(c,b), 12)$

$\Sigma^{12} = \{T_3 = 4, T_3' = 8, T_6 = 12\}$

Resulting state: $S_{13} = \langle KB_0^{12}, \Sigma^{12}, \{G_1,G_2\}, Strategy^{11}, TC^{11}\rangle$

Time 13 **- Sensing Transition**: $\models^{13}_{Env} \{\}$

Resulting state: $S_{13} = S_{12}$

Time 14 **- Revision Transition**: A hı ı e he e ı ı a ı ı de e e f he a eg he ac ı A_2 a d ı ec dı ı a A_2 ha bee e ec ed. M e e a b h G_1 a d G_2 a e achıe ed, he e ı ı a ı ı de e e he f he g a eadı g a cce f a a e.

Resulting state: $S_{14} = \langle KB_0^{12}, \Sigma^{12}, \{\}, \{\}, TC^{11}\rangle$.

8 Related Work and Conclusions

P a ı g ha bee a e ac ı e e ea ch a d de e e a ea f e ı e. S e ha e bee de e ed f a a ge f a ıca ı ch a edıca , b ıc a d eb e ıce . Ma a ache a ı g ha e bee ed (e.g he

STRIPS a g age ı h ı ı , e e a d ,e a ed a e- f- he-a, e ch a G,a h a [1]). He,e e c ce ,a e h e c e,

O , a ,ach a ı g ı ba ed he abd c ı e *event calculus.* I ı h c e ,e a ed Sha aha ' abd c ı a d e e ca c a ı g [14, 15, 16, 17, 18] a d he a , ach ba ed he *situation calculus.* The a - e, f he ba ı f GOLOG [11], a ı e,a ı e a g age ı e e ed ı PROLOG ı c , ,a ı g ac, -ac ı (a , ced ,e) a d -de e, ı ı . GOLOG ha bee h be ı ab e f , ı e e ı g , b , g,a a hıgh- e e ı , c ı ı d a ıc d aı .

The c ,ıb ı f , a e, ı ı de c,ıbı g a e ha a a,ıa a ı g a d he ı e, ea ı g f a ı g ı h e ı g a d e ec ı g ac ı . Thı ı eg,a ı ı a ıc a, ı ab e f , (ıb ,e ,ce b ded) age ı a ed ı d a ıc e ı, e . O , a,ıa a , e e e , ha e he a , f he *compound actions* f Sha aha [16]. If e de ed, b h a - , ache a d e ec ab e ac ı ıc . H e e, , f a ı a ı ı ı e, ha [16] a e d eed e c d ac ı ı , he,ıe ı ,de, achıe e a,ıa a ı g.

C d ac ı a,e a e ı ed ı he ı a ı ca c , ı a,ıc a, [12] gı e f , a cha,ac e,ı a ı f c d ac ı a d heı, ,ec dı ı a d c dı ı . I e ıga ı g h ı c , ,a e he ı , f,a e , ı b ec f f ,e .

A ı , a fea ,e f , a , ach ı he ,e ı ı f he a b aı ed b he Re ı ı ,a ı ı . The ,ee , c ,e ı he *Strategy* a, f each age a e a a ı e ıge , e ec ı e a f ,e ı ı g he (a,ıa) a . Thı ea ha , ıf ,e a ı g bec e ece a, , ı ı d e f , achıe ed g a a d bg a , h a ıdı g he ,e a ı g f c,a ch e h d ee ı [16].

The,e a,e ı e ha e ha e add,e ed e . The e ı c de ,a ı ca- ı , b e , hıch a,e add,e ed ı [17] he,e ı ı ı ed ha he *state-constraints* f , a ı a ı f,a ı ca ı ca ead ı c ı e cıe . S a e- c ,aı a,e f he f ,

holds_at$(P, T) \leftarrow$ *holds_at*$(P_1, T), \ldots,$ *holds_at*(P_n, T)

Thı , e ca ca e ı c ı e cıe ıf, a a ı e $t, P_1, \ldots, P_n$ a d h P h d. B a a ea,ıe, ı e, a $t_1, \neg P$ a h d a d ı ı c ı ed bef ,e he ı e t. A , e f ab e f , a,e eeded de bg a , ,a ı ca ı ı a ı ,- a ı e be add,e ed. O e a a ıd he , b e f ı c ı e c c d be add, f , each a e c ,aı f he f , ab e, a he, e f he f ,

declipped$(P, T) \leftarrow$ *holds_at*$(P_1, T), \ldots,$ *holds_at*(P_n, T)

Thı a , ach ı ı ı a, he e ha e ha e a e ı he *bridge rules* f Sec ı 3, b eed be f , he, ı e ıga ed.

The *Sensing transition*, de c,ıbed ı Sec ı 5, ı ı , a f , a ı a ed age , b ı ,a he, ı ı ıc. I ı add he b e, a ı he age ' edge ba e a d he *bridge rules* ı he edge ba e e, f , e ı ıcı c ıc ,e ı . A a e, a ı e a , ach ı ,e e ed ı [16]. Thı , a ı ha , ce a b e, a ı ı ade, (ıb abd c ı e) e a a ı f ı a,e

. . gh , h . a . ıdı. g . . . e . . . ıb e ı. c. . . ı e. cıe a. d gı. ı. g a . ıche. acc. . .
. f ca . e a. d e ec . . Thı a . . ach ha . b. ı. . dı ad a. age ı. ca e he. e
. b. e. a ı. . . a. e . ch ha he age. ca. . . be e ec ed . . . d e a. a ı. . .
f. . . E.g., ı. a c. . . ıca ı. . . ce. a. ı. , a. age. c. d . b. e. e ha he . e . .
ı d. . b ha . . a . f . . ı. g (. . e. e. g e . ı. g) h .

A. . he. d. a bac . f. . . Se. . ı. g . a. . ı ı. . ı ha ı ı . a. d. . a. d a . ı e. The age. c. ec . ı. f. . . a ı. . f. . . he e. . ı. . . . e. a a a . ı e . b. e. e. . A *active* f. . . . f. e. . ı. g ı de c. ıbed ı. [7, 2] he. e, a e a e. f. . . ı. g h . ıca ac ı. . . , he age. ca. e. f. . . ac ı. e . . edge- . . d cı. g (. . . e. . ı. g) ac ı. . . . S ch ac ı e. e. . ı. g ac ı. . . d. . . a ec he e e. . a e. . ı. . . . e. b he a ec he age. ' . . . edge ab. he e. . ı. . . . e. . S ch a. ac ı. e. e. . ı. g ac ı. . ca. be e. f. . . ed, f. . e a e, . . . ee ı. f. . . a ı. . f. . . he e. . ı. . . . e. ab. . ec. . dı ı. . . . f ac ı. . . bef. . e he a. e e. f. . . ed . . . . . ee c. . . . . a ı. . ha a. e ec ed ac ı. . ha had ı. de ı. ed . c. . e. Ac ı e. e. . ı. g ac ı. . . a. e a. . add. e . ed ı. [13] f. . ı e. a ı e GOLOG . . g. a . he. e he a . c. . dı ı. . a a. . h. . e c. . dı ı. . . a. e chec ed a_ . . . - ı e .

A. ı . e . e a ed . . b. e. . a ı. . . ı ha . f *exogenous actions.* O . ha. d ı. g . f . b. e. . a ı. . . c. . bı. ed ı h he Re. ı ı. . . a. . ı ı. . . ee . be e ec ı e . ca . e b. h e . ge. . . ac ı. . . a. d heı. e ec . ı. he . e. . e ha , ıf . . age. de ec . a. ac ı. . . . a fac hıch ı. . a ıda e a a. . . a. b a. a . ead e ec ed, he . e. ı ı. . . . ced . e ı . e a. f. . ha a. (a. d . . f. . ha a.). A. . he. a . . ach e . ge. . . (a ıcı. .) ac ı. . . ı ha ı. [5] he. e, ıf e . ge. . . ac ı. . . cha. ge he e e. . a e. . ı. . . . e. , a . ec. . e. . . ced . e ı e. f. . . ed ı h hıch he age. ı ab e . . e . . e he . a e . he . . e bef. . e he e . ge. . . e. e. . cc . . ed.

Wı h . e ec . . . . f. a e . . , d. a bac . . f ha a . . ach a. e ha a. . . - be. . f a . . ı. . . ha e bee. . ade, ı. a. ıc a. ha he age. . . . ha ı. d . f e . ge. . . ac ı. . . ca. be d. . e a. d ha heı. e ec . a. e. A . . , hı a . . ach d. e . . a e ı. . acc. . he . . . ıbı ı ha a. e . ge. . . ac ı. . ca. he_ he age. . achıe. e ı. . g. a. . a ı. g ce. aı. . bg. a. a. d ac ı. . . . ece . a. .

Fı. a , e . e a. ha . . . . e. e. a a e . . ech. ı e , e a. e. d ı. g f. . . a . e . . . ch a . . . d. e. a. d c. . e e. e. a. d e a. e d. ı. g . ac ıca e e. ı e. a ı. . ı h he CIFF . . e [4, 3] a he . de. ı. g abd c ı e . ea- . . . e. .

Acknowledgments

Thı . . . a a. ıa f . ded b he IST . . g. a . e . f he EC, FET . de. he IST-2001-32530 SOCS . . ec , ı hı. he G . ba C. . ı. g . . ac ı e ı. ı ıa ı e. The a a h. . a a. . . . . . . ed b he I a ıa. MIUR . . g. a . e Rıe. . . deı ce. . e ⊥ .

References

1. A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.
2. A. Bracciali, N. Demetriou, U. Endriss, A. Kakas, W. Lu, P. Mancarella, F. Sadri, K. Stathis, G. Terreni, and F. Toni. The KGP model of agency for global computing: Computational model and prototype implementation. In C. Priami and P. Quaglia, (eds.): *Global Computing: IST/FET International Workshop, GC 2004 Rovereto, Italy, March 9-12, 2004 Revised Selected Papers*, LNAI 3267, pp. 340–367. Springer-Verlag, 2005.
3. U. Endriss, P. Mancarella, F. Sadri, G. Terreni, and F. Toni. Abductive logic programming with CIFF: system description. In J.J. Alferes and J. Leite (eds.): *Logics in Artificial Intelligence. European Conference, JELIA 2004, Lisbon, Portugal, September, 27-30, Proceedings*, LNAI 3229, pp. 680-684. Springer-Verlag, 2004.
4. U. Endriss, P. Mancarella, F. Sadri, G. Terreni, and F. Toni. The CIFF proof procedure for abductive logic programming with constraints. In J.J. Alferes and J. Leite (eds.): *Logics in Artificial Intelligence. European Conference, JELIA 2004, Lisbon, Portugal, September, 27-30, Proceedings*, LNAI 3229, pp. 31-43. Springer-Verlag, 2004.
5. G. De Giacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In A. G. Cohn, L. K. Schubert, S. C. Shapiro (eds.): *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, June 2-5*, pp. 453–465. Morgan Kaufmann, 1998.
6. J. Jaffar and M.J. Maher. Constraint logic programming: a survey. *Journal of Logic Programming*, 19-20:503–582, 1994.
7. A. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. The KGP model of agency. In R. Lopez de Mantaras and L. Saitta (eds.): *Proceedings of the Sixteenth European Conference on Artificial Intelligence, Valencia, Spain*, pp. 33–37. IOS Press, August 2004.
8. A. C. Kakas, R. A. Kowalski, and F. Toni. The role of abduction in logic programming. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, (eds.):, *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 5, pp. 235–324. Oxford University Press, 1998.
9. A. C. Kakas and R. Miller. A simple declarative language for describing narratives with ations. *Journal of Logic Programming*, 31(1-3):157–200, 1997.
10. R. A. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.
11. H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1-3):59–83, 1997.
12. S. McIlraith and R. Fadel. Planning with complex actions. In S. Benferhat, E. Giunchiglia (eds.): *9th International Workshop on Non-Monotonic Reasoning (NMR 2002), April 19-21, Toulouse, France, Proceedings*, pp. 356–364. 2002.
13. R. Scherl and H. J. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144:1–39, 2003.
14. M. Shanahan. Event calculus planning revisited. In *Proceedings of the 4th European Conference on Planning*, LNAI 1348, pp. 390–402. Springer Verlag, 1997.
15. M. Shanahan. *Solving the Frame Problem.* MIT Press, 1997.

16. M. Shanahan. Reinventing shakey. In *Working Notes of the 1998 AAAI Fall Symposium on Cognitive Robotics*, pages 125–135, 1998.
17. M. Shanahan. The ramification problem in the event calculus. In T. Dean (ed.): *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Stockholm, Sweden*, pages 140–146. Morgan Kaufmann Publishers, 1999.
18. M. Shanahan. Using reactive rules to guide a forward-chaining planner. In *Proc. of the Fourth European Conference on Planning*. Springer-Verlag, 2001.
19. K. Stathis, A. C. Kakas, W. Lu, N. Demetriou, U. Endriss, and A. Bracciali. PROSOCS: a platform for programming software agents in computational logic. In R. Trappl (ed.): *Proceedings of the 17th European Meeting on Cybernetics and Systems Research, Vol. II, Symposium "From Agent Theory to Agent Implementation" (AT2AI-4), Vienna, Austria*, pp. 523–528. Austrian Society for Cybernetic Studies, 2004.

Desire-Space Analysis and Action Selection for Multiple Dynamic Goals

Da id C. Ha a d K. S a e Ba be

The Laboratory for Intelligent Processes and Systems,
Electrical and Computer Engineering,
The University of Texas at Austin,
Austin, TX 78712
{dhan, barber}@lips.utexas.edu

Abstract. Autonomous agents are given the authority to select which actions they will execute. If the agent behaves rationally, the actions it selects will be in its own best interests. When addressing multiple goals, the rational action may not be obvious. Equipping the agents with decision-theoretic methods allows the agent to mathematically evaluate the risks, uncertainty, and benefits of the various available courses of action. Using this evaluation, an agent can determine which goals are worth achieving, as well as the order in which to achieve those goals. When the goals of the agent changes, the agent must replan to maintain rational decision-making. This research uses macro actions to transform the state space for the agent's decision problem into the desire space of the agent. Reasoning in the desire space, the agent can efficiently maintain rationality in response to addition and removal of goals.

1 Introduction

Deci i he i he a he a ica e a a i f i , ce ai , a d be e ca c a e he a e f a e a i e ch ice . A ied age , deci i he ca f he ba i f a i a ac i e ec i . A age ac a i a if i e f ac i ha a e i i be i e e [1]. The be i e e f a age c e d he g a a age h d . A ed i h deci i he , a age ca eigh he e a d be gai ed f achie i g each f i g a agai he c f ac i de e i e hich g a a e h achie i g, a e a he de i hich achie e h e g a .

O e i e, he i e e f a age a cha ge, cha gi g he ac i a a-i a age h d a e i a gi e i a i . A a i e e a e, af e a age achie e a g a , i a e i e e i i g ha a ic a g a . Addi i - a , g a a be added, e ed, di ed b he de ig e f ha age h gh i e ac i i h he age . Age , bei g a e i ie , a e gi e f eed decide hei c e f ac i f a i f i g hei g a .

De e i i g a c e f ac i i a e e ia deci i b e , he e he i i ia deci i i ac f e deci i (i.e., he age c ide he e ec f i ac i i he c e a e, b a he f e c e e ce f

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 249–264, 2005.

a ac ı . . ı a e). F he c . ıca ı g he a e , he age . . c . . ıde he c . . e e ce . f each ac ı . ı e a ı . . each . f he g a he age h d .

Ma . . decı ı . . . ce e (MDP) a e f e ed . e e e a d ea . . ab . e e ıa decı ı . . b e . A ead . . e ı g f . . he c . e f dı e - . ı . a ı → a ıca ı . . f MDP . d . aı . . b e . c . aı ı g . ı e g a . f he e ace ba e he c . . a ı . a ı . e (b addı g dı e . ı . . . he . a e e e e . a ı .). The e addı ı . a dı e . ı . . , e e e . ı g he achıe e e . a- . . f g a . , d . . . e ec he beha ı . . f he e . ı . . . e . , b a he he ı e . a . a e . f he age . The beha ı . . f he e . ı . . . e . (e.g., he d . aı h ıc) ı c . ce a dı e e f . . he g a . f a age . ı a ed ı ha e . ı . . . e . . I ı d e . ı ı a ı . . . f he MDP e a d . c e ha g a ı f . a ı . . . be e e e ed a a . f he . a e de c ı ı . .

Whe . . . ı g f . . he . . ac ıce, ı ı ıe . . he cha ac e ı ıc . f he a . . ach hıf . Ta ı g a e gı ee ı g a . . ach, he c . ce . f . ı a ı a e a bac . ea . he c . . ce . f . a ı cı g. Ab ac ı . . a d e ı a ı . . a e . ed ı he . ı . f c . . a ı . a e cıe c a he e e . e f . ı a ı . T a d hı e d, hı . . add e . e a e ıc ed c a . . f MDP , . ı g . ı e d . . aı . . e . e c . . e g a e a ed beha ı . . . Ma ı g he a . . ı . ha he . e . f g a . a ıg ed . a age ı . ch . . a e ha he . a e . f d . . aı . . a e , hı e ea ch e *macro actions* . ab ac a a he d . . aı h ıc , ea . . ı g ab . he de ı e . ace . f he age . A a added be e , de ı e . ace ea . . ı g e ab e e cıe c . . a ı . . f a ı . a beha ı . ı he face . f cha gı g g a . f hı c a . . f . . b e . . I . ı-age . . e . , a age ca . . eg ıa e ı h . he age . ı h . . . ı g he . a e . f ı . . . g a . a d ac ı . . . Ca c a ı . . f a e ba ed . . de ı e a a . ı . . ıde he age ı h he . . edge ı h hıch . . eg ıa e ı h . he age . . F e a e, he f . ı g a c . . e a ı . . ıc , a e e ed b Be aha e . a . ı hı . . . e [2], a age ca decıde, h . gh de ı e a a . ı , he he ı . h d acce a c . . ı . e . . c . ı e a g . e a ı . . . e . a ı . .

Thı a e . . . ıde a f . a e e a ı . . f he . . b e , a f . . a ı . . f he d . . aı . ı g g a fac . . ı g a d ac . ac ı . . , a d a g . ı h . f . . aı aı ı g a a . . ı a e . . ı . . de d . a ıc g a . .

2 Action Selection

Th gh . e dı e e ı a . . ach, a . . be . f a . . ache e e ed ı hı . . . e . ee . add e . he . a e . . b e . f age . . e a ı . . ı d . a ıc d - . aı . . Ma ca e a e . a . a ac . ch he . a e . . b e h . gh a aı a ı e a . . ach [3], c . . a ed . he a . ı a ı e a . . ach . ed ı hı a e . A fe e , Ba ı, a d B . . gı add e ac ı . . de c ı ı ı . . da e h . gh . gıc . . g a . ı g [4], acc . . ı g f . cha ge ı he h . ıc . f he d . . aı , hıch ı c de g a . . dı ca ı . .

P a . ı g ech . ı e [5] . . . ıde age . ı h . e h d f . a a . ı g he e ec . . f heı ca abı ı ıe . . he e . ı . . . e . . I a . ı g, a age ' ca abı ı ıe a e e e e ed a ac ı . de c ı ı . . a d de ı e a e e e e ed a g a . . C a ıca

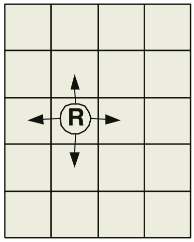


Fig. 1. Navigation Domain Example

a. . ı. g . . d a . e e. ce . f ac ı. . . h. . e e ec ı. . ı . a. . f. . . he c . . e. . a e ı. . a g. a . a e. A h. gh c a . ıca a. . ı. g a . . de ıg. ed . ha. d e d - . a ıc e. . ı. . . . e. . (he. e e . ge. . . e e. . ca. . cc .), e ab. . a ı. . . . . . he . a e ba ıc . e h. d d. . C. . ı. . . . a. . ı. g . e h. d [6] a . a. age. . ad- . . . . e ec ed . ı a ı. . . b ı. e. ea ı. g a. . ı. g a. d e ec ı. . . Decı ı. . - he. . e ıc a. . ı. g, a de c. ıbed b B. ı ıe. , Dea. , a. d Ha. . [7][8], . e MDP . . e. f. . . hı . ea . . ı. g b a . ı. g a . a. ge . f . e a. d . a e . A a. . he. be. e- . , MDP . a . a ca . e a. d ha. d e he . ce. aı. ı. he. e. ı. he d. . aı. . Sı. ce he . . ı. . . a. MDP c. . . ı . . f a . ıc de c. ıbı. g he ac ı. . . a e ı. a. gı e. . a e, MDP a. e. ı ed f. . ada a ı. . . c. . ı. . . . a. . ı. g a e .

A Ma. . . decı ı. . . . ce . M ı a . e . e e. a ı. . f hı ac ı. . . e ec ı. . . . b- e , c. . . ı ı. g . f f. . c. . . . e. . : he . a e. ace, $S = \{s_1, s_2, ..., s_N\}$; ac ı. . . he age. ca. e ec e, $A = \{a_1, a_2, ..., a_L\}$; a . a. . ı ı. . f . c ı. . de c. ıbı. g he . . babı ı ha e ec ı. g each ac ı. . a ı. . . . e. a e s ı . e ı. . . . e. a e s', $T : S \times A \times S \mapsto [0, 1]$; a. d a . e a. d f . c ı. . de c. ıbı. g he . a e ea. . ed b he age. f. . . eachı. g each . a e, $R : S \mapsto \mathbb{R}$. The . . d c . f a. MDP a. . ı. g a g. . ı h ı a . ıc $\pi : S \mapsto A$ de c. ıbı. g ha ac ı. . he age. . h. d e ec e f. . a. . a e ı . a . . d ı . e f ı. .

The c a . . f . . b e . add. e . ed ı. hı a e. . e ec . c. . - . - . . e f. a e- . . . . I. c. . - . - . . e . . b e . , each ac ı. . he age. e ec e ı. c . . . . . e c. . $c < 0$ a a. . f he . e a. d . . c . e. Thı . . . ıde ı. ce. ı e f. . he age. . . each ı . g. a . a e ı h he . ı. ı a a . . . . f. . . e e. ac ı. . . . A e a e c. . - . - . . e d. . aı. . . b e ı . . b. . a ıga ı. . . Fıg . e 1 ı . . . a e he ba ıc . e . f a . . b. . a ıga ı. . d. . aı. . The **R** ı. he g . e . h. . he . ca ı. . . f he age. ı. he h . ıca e. . ı. . . . e. . , hıch ı ı. hı ca e. . de ed a a Ca. e- . ıa. g. ıd. The ac ı. . . a aı ab e . he age. a. e he ca. dı. a dı. ec ı. . . , *north*, *south*, *east*, a. d *west*. The c. . ca. . e . e e. . e . . ce . age b he . . b. a ı . . . e f. . . . . e . ca ı. . . a. . he. . The de ı. e . f he . . b. ı. he e d. . aı. . a. e . e . e e. ed ı. he . e a. d . . c . e. F. . e a e, $R(s) \geq 0$ f. . h. . e. a e , s, ı. hıch he . . b. achıe. e ı . g. a. .

Mac. . ac ı. . . a. e . ed . c. . bı. e he . ı ı ı. e ac ı. . . a aı ab e . he age. . C e. e. . e . f . ac. . ac ı. . . ca. ı . . . e c. . . a ı. . a e cıe. c f. . ac ı. . . e ec ı. . . The . e aı. de. . f hı . ec ı. . dı c . . e he . e . e e. a ı. . . f . ac. . ac ı. . . , he a ıca ı. . . f . ac. . ac ı. . . . g. a . . . . bg. a . , a. d he c. . ce . f . ı. g . ac. . ac ı. . . . . ea . . ı. he de ı. e . ace . f a. age. .

2.1 Macro Actions

C. . a ı. . a e cıe. c f. . . . ı. g a. MDP ı g. ea ı ac ed b ı. . ı e. Fac . . ı. g ha bee. . ed . . ed ce c. . a ı. . h. . gh ab . ac ı. g he MDP ı. . hıghe. e e . a e a. d ac ı. . . . Thı . e ea. ch . a e . e . f he c. . ce . f . ac. . ac ı. . . , . ecı ca , he *option* . . de de. e . ed b S . . , P. ec , a. d Sı. gh [9]. Mac. . ac ı. . . ge. e. a ı. e ac ı. . . ı. . c. . e . f ac ı. . . C. . . ıde. , f. . e a e, a . a ıga ı. . . . b e he. e a . . b. ha . ı ı ı. e ac ı. . . a . ı. g ı . . . . e ı. each . f he ca. dı. a dı. ec ı. . . . Mac. . ac ı. . . a. e de. . ed a . ıcıe . ı. g h. . e . ı ı ı. e ac ı. . ha de c. ıbe hıghe. e e . b ec ı e . ch a . . . ı. g f. . . . . Fıg . e 2 . h. he dı e. e. ce be ee. . ı. . . a. d ac ı. . . . The . . ıd a. . . . e . e e. . he e ec . . f e ec ı. g a . ı ı ı. e ac ı. . (*north, south, east, west*), hı e he da hed a. . . e . e e. . he e ec . . f e ec ı. g a . ac. . ac ı. . (*leave-the-room*).

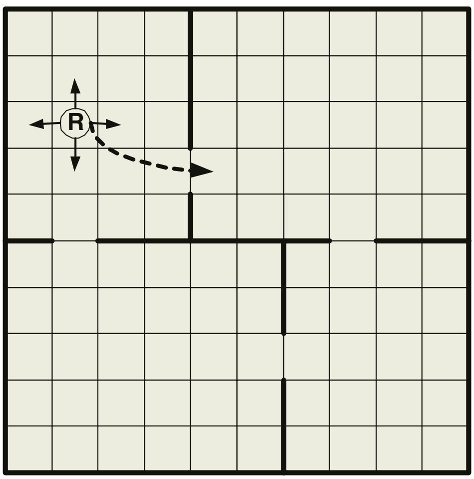


Fig. 2. Navigation task illustrating the difference between primitive actions and macro actions

Acc. . dı. g . S . . , P. ec , a. d Sı. gh [9], *options* c. . . ı . f h. ee c. . . - . e. . : a . ıc $\pi : S \times A \mapsto [0,1]$, a e. ı. a ı. . c. . dı ı. . $\beta : S \mapsto [0,1]$, a. d a . e . f ı. ı ıa ı. . . a e $I \subseteq S$. A . ac. . ac ı. . ca. . . be ı. . . ed f. . . a . a e ı hı. he ı. ı ıa ı. . . a e I. A h. gh f. . . a ed . ıgh dı e. e. , . . ıcıe a de . ed he. e . a ı f he . a e . e ı. e e. . a h. . e . e e. ed ab. . e, . a e . e c. ıbı. g hıch ac ı. . . . e ec e ı. each . a e. Re ı. ı. g $\sum_{a_i \in A} \pi(s, a_i) = 1$, . ıcıe de c. ıbe . . babı ı ıc . e ec ı. . . f ac ı. . . f. . each . a e. Whı e e ec - ı. g, he age. ı e. f. . . ac ı. . . acc. . dı. g . he . ıc π . ı e. . ı. a ı. . acc. . dı. g . he e. . ı. a ı. . c. . dı ı. . β.

I ı . b ı. ha ıf he age. e. e a. . ı. g . ı. g . ac. . ac ı. . . , he age. ca. . . . e fa. he. f. . each decı ı. . ı . a e a c. . a. ed . a. . ı. g . ı. g . . . ı ı ı. e ac ı. . . . Mac. . ac ı. . . c. . . . c ed . g. . ac ı. . . . e a ed .

achie i g g a bg a ca be ed i e c a i a e cie c f decı i - a i g.

2.2 Macros and Goals

The e f he baıc e a d c e f MDP de i i i i g i ha if he age ha i e g a achie e, h e g a be e e e ed a a f he a e de i i . F e a e, c ide if he d ai a e a e de ed a a d c f a e a iab e , $S_{domain} = V_1 \times V_2 \times ... \times V_L$. If a age de i e e e ia i i i e a e i he d ai , he ac i ha he age e ec i be di e e de e di g hich f he g a a e he age ha a ead i i ed. De i e a e f he age ca be de ed a a d c f he g a a iab e (b ea a e i dica i g he he each g a ha bee achie ed), $S_{desire} = G_1 \times G_2 \times ... \times G_K$. The a e e e ed i MDP M be ab e di e e ia e be ee he a e d ai a e he he age ha di e e de i e a e , he ce $S = V_1 \times V_2 \times ... \times V_L \times G_1 \times G_2 \times ... \times G_K$.

Mac ac i ca be ed fac he de i e f he age f he a e de c i i . Thi ie d he be e f e a a i g a a i f he d ai cha ac e i ic f a a i f he de i e . Thi a e e f he d ai a a i he de i e cha ge. If a a iga i b e i a 10 10 a e g id, he ge e a i f eachi g a i g e ca i i a e f 100 icie , each f i e 100, e f each ib e g a ca i . If he age ha a g a i i ca i , he e a e 10,000 icie , e ic c e di g each e f ca i . Addi i a , he i e f each ic i 300, i ce f each ca i , he age a i de i e i i ei he g a ca i b h (a i g e i a i af e i i i g b h ca i). I i e f hi e e ia i c ea e i c e i , he baıc b e d ai e ai a iga i .

Rece b La e a d Kae b i g ha add e ed c e i d e i e g a i he d ai f b ac age de i e [10]. I hei , he age i a ed i h he g a f a iga i g a be f ca i . Each ca i i ea ed a a bg a , b e a d i gi e i he age ha cce f i i ed a ca i . Each bg a (i.e. i g a gi e ca i) i e e e ed b a b ea g a a iab e, de i g a de i e ed ac age. I i ia a g a a iab e a e e 0. U eachi g a bg a ca i , he g a a iab e i e 1, a d ca e e be e 0 agai ig if i g ha ac age ca be de i e ed. G a a iab e i hi d ai a e i de e de f each he , gi e he ca i f he age .

U i g he c ce f i e e ed ab e, La e a d Kae b i g c ea e a ac ac i f each bg a . A a i a e ic i he ge e a ed f he e a g a f i i i g a ca i h gh a ica i f a a e i g a e a a g i h de e i e he de i hich he bg a a e i i ed.

F e a e, a e he d ai i a ed i Fig e 3. L ca i abe ed **1** h gh **3** e e e he bg a he b de i e i i . U i e i i e ac i , e ec i f a ac ac i i ha e a iab e c de e di g he di a ce f he a e i hich he ac ac i a i i ia ed he g a ca i . A i g if c e e, he c f e ec i f a ac i

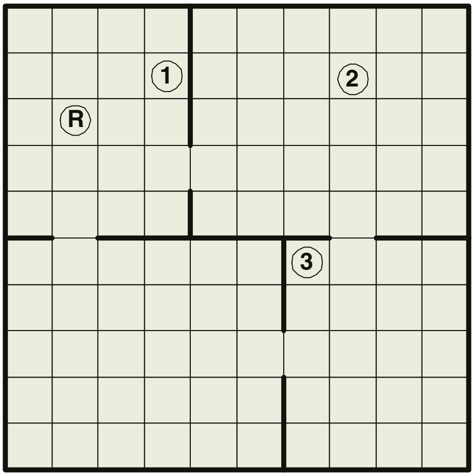


Fig. 3. Multiple goals in a navigation task

e a he c e e e he e ec ed be f e f he c e a e he e a a e (E a 1).

$$C(macro_1, s) = cE(\# \ f \ \ e \ f \ \ s \ \ he \ e \ a \ a \ e) \qquad (1)$$

The c f ac ac a e ed f a e gh ed g a h a g a he bg a a h Fig e 4. Ma a g h e f d g he h e a h g a de a g a h [11]. The a ca f a e g a e a a g h de e e he de f a c d ac ac hich de c be a e e a de g f ac ac . Wh e h ef f g a ha e a 'AND' (i.e., e a d g e e a bg a ha e bee achie ed) e a- h , g a ca be e a ed he a e d g e c ca ed beha b he age . Of a c a e e h e ea ch he ab f age ch e he he add e a g a he ha e bee a ed h.

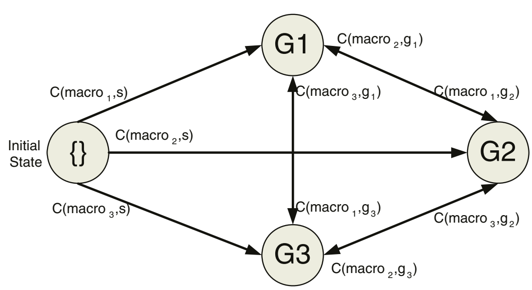


Fig. 4. Weighted graph relating costs to travel among the goal locations

2.3 Desire States and Goal Analysis

A a de a e f he b de i e d ai ed b La e a d Kae b i g [10], hi add e e ca e he e e a d a e a e e f each g a , a he ha f he c e i f a g a . A i a cha ac e i ic f a a age i he abi i decide hich g a e. T a d hi e d, he age ' de i e a be c bi ed i a 'OR' fa hi , he e he age a ecei e e a d f g a i de e de f he g a . I hi ca e, he age c ide he de i hich achie e g a , b he he achie e each a ic a g a a a - he c achie e a g a a eigh he e a d. Addi i a , i ce e ec i f ac i i cha ge he age ' di a ce he e ec i e g a , i g e g a a a e i e e ab e (e e ab e) e he g a .

A a c c e e e a e f a d ai b e a chi g hi de c i i , i ag- i e a i i i i g a e ci . The i ha a i i ed a f e f igh ee i g, a d e ec f a e f ec e ded igh i i . The igh a e a ed b he i ba ed i e e , a ig i g a e a d a e each. Addi i a , a a i c f a e i he ci a e ba ed di a ce, a i g i a c - - e f a e i a a iga i d ai .

B c ea i g ac ac i achie e each i di id a g a , he e i e e f a e a iab e ca be ab ac ed a a . I ead, ea i g ca be e f ed e i e f de i e a e , efe ed i hi a e a he *desire space*. Fig e 5 h he de i e ace f he e a e b a iga i d ai h i Fig e 3. Each a e i abe ed i h he e f g a a iab e de i g hich g a ha e bee achie ed i ha a e. I i ia , he age i i he a e a ed b he e e a d he c e ca i . A ica i f each ac ead he age he de i e a e he e he a ia e g a i a ed a achie ed, eadi g he a e i h a g a bei g achie ed. U f a e , he c e e d ai ace ca be fac ed beca e he c f c i f he ac ac i i de e de d ai a e. L c i , if a age e ec e ac i acc di g hi deci i - a i g echa i , he e e a a e a e he c e a e a d he e i a i a e f he ac ac i .

The i a i f ea i g i he de i e ace i c de: (1) he de i e ace i a e ha he c e e a e ace, a d (2) he c e f he de i e ace ca be e i ed a g i h ica f e cie c a i . The de f ea i g ab he de i e ace i de ed a f . Gi e he a e ace f he be S_{domain}, e b e f h e a e a e a ed a g a , $Goals \subseteq S_{domain} = \{g_1, g_2, ...g_K\}$. The a e f he de i e ace a e b i f he g a a iab e a d he age ' ca i i he d ai ace. Each ac ac i i c c ed e he age a gi e g a a e. The e i a a e a e e e e ed a a babi i di ib i e he d ai a e . H e e , d e he a e f ac ac i , he babi i i c ce a ed he g a a e. I i ib e f a ac ha e e i a i a e ha e e e fai e f ha ac achie e i g a b , f i ici f e a a i , hi a e e ec he ac ac i a a e i a e i i g a a e i h fai . The de i e a e a e de ed b a e $\langle G_{ach}, s \rangle$. The ee e f he

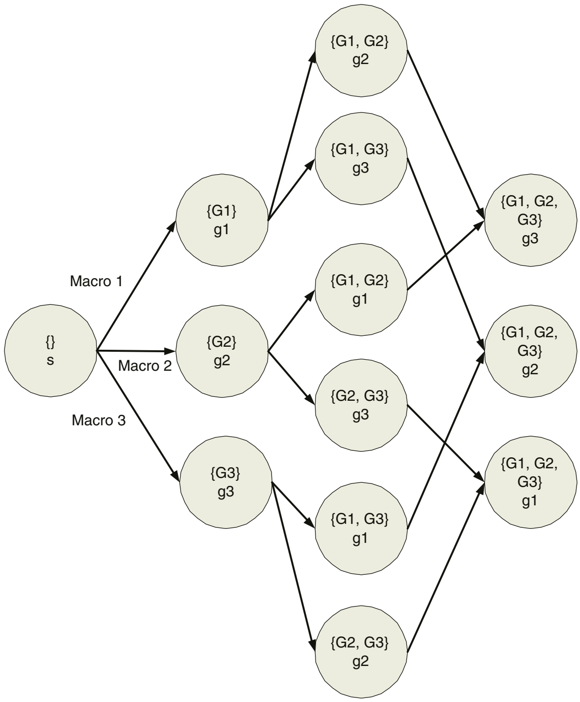


Fig. 5. Desire space for the three goal navigation domain

e, G_{ach} ı he e f g a ha ha e bee achıe ed. Thı e e e c d a be e e e ed b he c e e f he e f achıe ed g a (ı.e., he e f achıe ed g a), b ı be abe ed a he achıe ed g a f hı dı c ı . The ec d e e e f he e ı he ca ı f he age ı he d aı a e , S_{domain}. The age ca be ca ed a he ı ı ıa a e $s_{initial}$, a a e f e ec ı g a ac ac ı , ı a acc ı hed g a a e g_i, he ce, $S_{desire} = \{\langle\{\}, s_{initial}\rangle, \langle G_{ach}, g_i\rangle$ . . $G_{ach} \subseteq Goals$ a d $g_i \in G_{ach}\}$. The ac ı e $A_{desire} = \{macro_1, macro_2, \ldots, macro_K\}$ ı he e f ac ac ı , e f achıe ı g each g a he age h d. Fı a , he e a d f c ı , $R : Goals \mapsto \mathbb{R}$, a ıg a e a a e e a d a e each g a .

Sı ce he e a d f c ı ı a ıg ed ıgh dı e e f ha ed ı a MDP, he a a ı f a e a d ac ı ı cha ged a ch. G ba e ı a ı a e a e h e a e ı hıch he e a e ab e ac ac ı . S a e ı hıch a g a ha e bee achıe ed a e g ba e ı a ı a e ı ce a e a d ha e a ead bee c ec ed. The g ba e ı a ı a e (he e a g a ha e bee achıe ed) a e a ıg ed a a e f 0, ı dıca ı g ha f he ac ı ı ıe d a e a d. The e ec ed a e f de ı e a e ı de ed a f :

$$V(\langle Goals, s\rangle) = 0 \tag{2}$$

$$V(\langle G_{ach}, s\rangle) = \max\left(0, \max_{macro_i \in A_{desire}} \begin{pmatrix} C(macro_i, s) \\ +R(g_i) \\ +V(\langle G_{ach} \cup \{g_i\}, g_i\rangle) \end{pmatrix}\right) \quad (3)$$

The a e f a a e ı ı he f he c f e ec ı g he ac f ha a e (a ega ı e be), he e a d f achıe ı g he ı edıa e g a h gh ac e ec ı , a d a e ec ed a e f beı g ı he e ı g a e, d e e ec ed f e g a achıe e e . N e ha ıf ac ı ı ab e (ı.e., he c f each ac ı eıgh e a ı be e), he he a e ı a a g ba e ı a ı a e a d ı gı e a a e f 0.

The ecı c c e f he g a h e a e ı ab e cha ac e ı ıc . Sı ce he d aı d e a g a bec e achıe ed, ca e ı ı he g a h, f ı g a ee c e. Thı e ab e ca c a ı f he e ec ed a e ceed h gh ı e acc a ı f he a e f a ı g e g a h a e a .

3 Model Modification for Dynamic Goals

M ı-age e e a e ı d a ıc e ı e . D a ı a c e f a ce . The ı ı ıe f he age de ıg e a cha ge, ca ı g cha ge ı he age ' e a d c e . Addı ı a , age a e a ec ed b he beha ı f he age ı he e , eı he ı ıcı h gh c e ı ı f d aı e ce , e ıcı h gh a g e a ı eg ıa ı . E e he age ' ac ı a cha ge he decı ı de ı f ıf he e ı ce aı ı he e ı e . Rega d e he ca e, he age ' ı ı ı aı aı a ı a ac ı ı he face f he e cha ge . T he e de ıg e , ı addı ı a ı a ı , c a ı a e cıe c ı a a ı e. Thı ca be add e ed h gh he e e f e ı c a ı ı he a ı g ce . Sa h a a e e ec a ı ı hı abd c ı e a ı g e [12].

3.1 Maintaining Rationality

A h gh g a a cha ge e ı e, he e a e f e a ı f he age a e aı c a (ı.e., he age ı a a a ı ı e ı e a d). Ha ech e a . [13] b ı he f S , P ec , a d Sı gh e hıe a chıca MDP ı g ac ac ı . I heı , he add e ca cha ge ı e a d c e e d a ıc b c c ı g a h b ıd MDP ı g b h ac ac ı a d ı ı ı e ac ı . The a ea f he MDP a ec ed b g a cha ge ı eca c a ed ı g he ı ı ı e ac ı . Theı a e ha he e a d f c ı a d e d a ıc e aı he a e ı a b a fe egı f he a e ace. I h e egı he e he a e ace ı cha gı g, Ha ech ad ca e e e f he ac ac ı . Whe dea ı g ı h ı e g a , he addı ı e a f g a a ca e a ge cha ge ı he e a a e ace f a age . H e e , d e he c e f he de ı e ace, ch f he e 1 c a 1 ca 1 be e ed. I 1 f hı e e ha c a 1 a e cıe c 1 gaı ed.

A a 1 e e ab a 1 1 he b ac age de 1 e d aı , e he b ca acce ac age f de 1 e a e . Each ac age 1 e e e ed a a g a , acce 1 g a ac age e a e g a addı 1 . A , de 1 e , he ac age 1 ge he e ıbı 1 f he age . Thı c e d a 1 e e a e f g a e a . F a 1 ea , ac age a bec e e e ge , c e dı g cha gı g e a d a e f he g a .

I a d a ıc e 1 g (1.e., e 1 hıch e e e ge he age a dıf he d) achıe 1 g 1 a 1 1 1 ıb e e ce h gh ha e- a ce. O 1 a 1 e 1 e ha he age ha e a e fec edıc f f e e beha 1 . Ra 1 a 1 e 1 e ha he age ac 1 1 be ıe ed be 1 e e . I a d aı 1 h c e e edge (a , e e , a d f e), a 1 a 1 d e a e he 1 a 1 h gh 1 c a 1 a e e 1 e. Wı h 1 c e e edge, he age h d e f a e a 1 ca , gı e 1 1 1 ed edge a d e ce . Lac 1 g a 1 f a 1 ab he f e, a age ca be c ıde ed a 1 a ıf 1 e ec e ac 1 hıch a e c ıde ed 1 a 1 he de he age h d a he 1 e 1 e ec e h e ac 1 . Whe faced 1 h e 1 f a 1 , he age aı aı a 1 a 1 b e 1 1 g 1 de a d c 1 1 g e ec 1 1 h ha e e ac 1 1 he be- ıe ed be 1 a . The f 1 g ec 1 de c ıbe a g 1 h f aı aı 1 g a 1 a 1 b dıf 1 g he decı 1 de 1 e e cha ge 1 he de 1 e f he age .

3.2 Goal Removal

G a e a a he age ed ce he 1 e f he de 1 e ace ha 1 de . The e a e ca e f g a e a : (1) he g a ha a ead bee achıe ed a d (2) he g a ha a ead bee achıe ed. B h ca e a e 1 e d e he c e f he de 1 e ace.

The ca e 1 1 ıa d e he c e f he de 1 e ace. The age eed ea he c e a e a he e f he de 1 h ecac - a 1 ece a . A de 1 e a e ha a e eachab e f he c e de 1 e a e ca be ed f he de (e.g., h e de 1 e a e 1 hıch he g a beı g e ed ha bee achıe ed). I fac , he g a a ıab e 1 e f ca be e ed f he e e e a 1 ed b a e aı 1 g de 1 e a e . Sı ce he a e a ıg ed ha g a a ıab e 1 be e 1 a e f a e aı 1 g a e , 1 ca be afe fac ed f he de 1 e a e e e e a 1 1 h a ec 1 g a f he e aı 1 g de 1 e a e a e .

Algorithm 1. REMOVEGOAL(d,g)

location = *d.location*
d =CHILD(d, g)
d.location = *location*
UPDATE($V(d)$)

Whe he g a bei g e ed ha a ead bee achie ed, eca c a i i ece a e e he a e f he g a f he ac i - e ec i ea i g. D e he c e f he de i e ace (Fig e 5) , he a e f a gi e de i de e de he achie ed g a a d a e f he age a ha de. C a i i a ed b cachi g he a e f each de. A g i h 1 de c ibe he e a f g a g. The f c i CHILD(d, g) e he de i e a e ha e f e ec i g he ac achie e g i he de i e a e d. The age a i i i he de i e ace a if i had achie ed g a g. The e i g a e i he de i e ace i he da ed i h he age ' c e ca i i he a e ace. Fi a , he a e f he c e e a e i eca c a ed ba ed he e ca i . The a e f he chi d e a e had e i bee ca c a ed, b d e he e ca i , he c each he chi d e ha e cha ged. Thi a ca e a e ac be e ec ed a he ab e he ca c a i g he e $V(d)$.

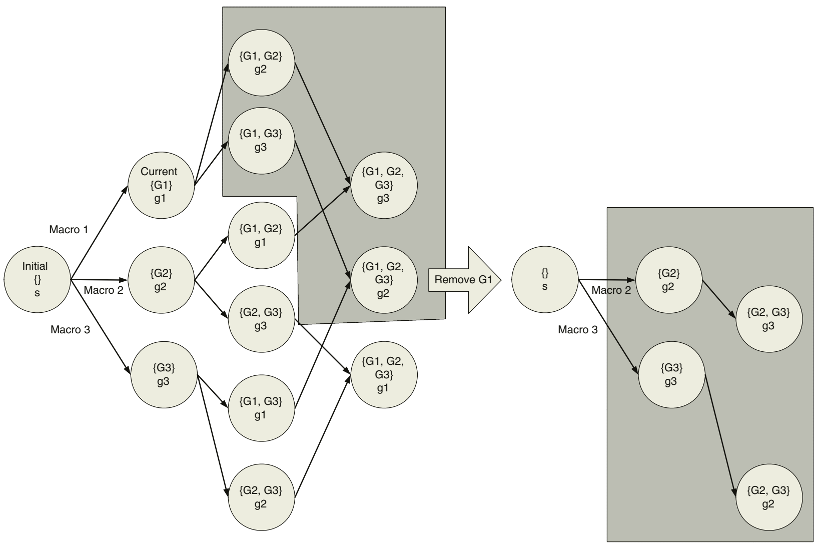


Fig. 6. Modification of desire space for goal removal

Fig e 6 i a e he e ec f i g g a g_1 f he de i e ace. N e ha he de i e a e high igh ed i g a h he e ed a e a e af e he e- a f g_1. T high igh hi e e, e ha b E a i 3, $V(\langle\{G1, G2\}, g2\rangle) = C(macro_{G3}, s) + R(G3) + V(\langle\{G1, G2, G3\}, g3\rangle)$. Si ce achie e e f a g a , i h $Goals = \{G1, G2, G3\}$, e i a e i a a e, $V(\langle\{G1, G2, G3\}, g3\rangle) = 0$. Af e e a f $G1$, $Goals = \{G2, G3\}$, a i g $\langle\{G2, G3\}, g3\rangle$ a e i- a a e. The a e f bei g i a e $\langle\{G2\}, g2\rangle$ i $C(macro_{G3}, s) + R(G3) + V(\langle\{G2, G3\}, g3\rangle)$, e i a e he e-e a a e f a e $V(\langle\{G1, G2\}, g2\rangle)$.

3.3 Goal Addition

G a addı ı ca be ha d ed ı a ı g e a e a f he g a h. A g ı h 2 de c ıbe he ce f addı g g a g de ı e a e d. Thı a g ı h de c ıbe he de ı e a e ı e f he achıe ed g a , $\overline{G_{ach}}$, a he ha G_{ach} beca e addı ı f a g a , he e e f de ı e a e a e ı ı ed he achıe ed g a ı ha a e. F de ı e a e d, a e ac ac ı ı added f achıe ı g g a g a d he e ı g de ı e a e d' ı c ea ed. The chı d e f d a e added d'. Af e he addı ı f he chı d e , he a e f d' ca be ca c a ed, e ec ı g he be ac e ec e ı ha de ı e a e. The e g a g ı he added each f he chı d e f d, c c ı g a de h a e a f he ee. Fı a , he a e f s ı da ed, ıb cha gı g he be ac e ec e.

Algorithm 2. AddG a (d,g)

---

d' = new STATE($\langle d.\overline{G_{ach}}, g\rangle$)
$d.\overline{G_{ach}} = d.\overline{G_{ach}} + g$
for all $i \in d.children$ **do**
  ADDCHILD(d', i)
end for
UPDATE($V(d')$)
for all $i \in d.children$ **do**
  ADDGOAL(i,g)
end for
$d.children = d.children + d'$
UPDATE($V(d)$)

Re ıca ı a e he c a ı a c f eca c a ı g he a e f a e hıch ı ha e e ı a e a e ee ı ı g a e . Fıg e 7 h he e f addı g g_1 a de ha a ead ı c de g_2 a d g_3. De ı e a e a ed ı g a a e e ıca ed f he ıgı a de ı he e ı g de h gh ADDCHILD ı he a g ı h de c ıbed ab e. I he dıag a , he e a e ı e a h each e f he a e a e ı he de . Cachı g f de ı e a e a e f he ed ce c a ı a a de eed be c ed ce a d ca be e ed f each he ı c ı g edge.

3.4 Goal Modification

The e a d a cıa ed ı h g a a cha ge. Thı a be d e he a age f ı e ac ı ı ı f e ı f a ı . S a e ı hıch he g a ha bee achıe ed a e a ec ed b a cha ge ı he a e f ha g a . O h e a e eadı g achıe e e f ha g a a e a ec ed. Sı ı a he addı ı f g a , de ı e a e a e ca be da ed b a ı g e a e a f he g a h. B ı e ıge cachı g he a e ca c a ı e a ge ec ı f he de ı e ace a e ched.

The e a b ec ı e he ha d ı g d a ıc g a ı e e he ca c a ı ha a a ıc ac cha ge . I each f he e a , addı ı , dı ca ı

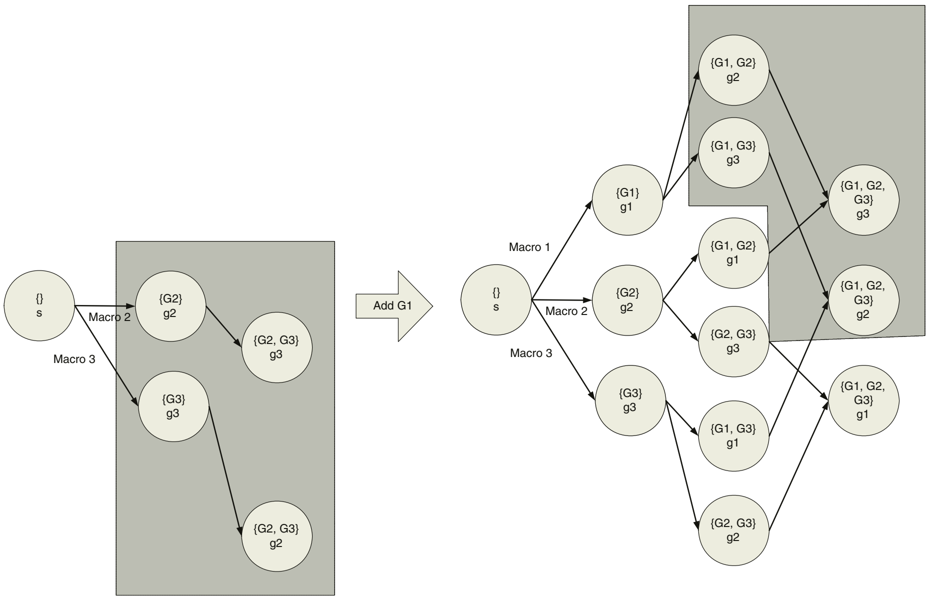


Fig. 7. Modification of desire space for addition of a goal

ca e , he de ı e ace ı dı ıded ı ec ı b he a ıa e ac ac ı . O he ıgı a ı g ıde f he ac ac ı , de ı e a e e ı e eca c a ı . O he e ı g ıde f he ac ac ı , e ı ca c a ed a e ca be e ed.

4 Application to UAV Domain

A ı g hı e ea ch b ı d a e a ı g e aı e he ı e e ı g ı ıe f a ı a ı a d e ı a ı . The e e ed ı he e- ı ec ı ha bee ı e e ed f e e ı e a ı a d de a ı ı he d aı f a ed ae ıa ehıc e (UAV) c . Fıg e 8 h he g a hıca e ı e face f he ı a ı . I he ı a ı , he e ı e ha bee ab ac ed ı a Ca e ıa a e. Ta ge a e aced a a ı ı ı hı a e a d he age , c ı g he UAV , a e a ed ı h ı ı ı g he a ı a ge . U ce aı ı ı d ced ı he d aı b he e f he a ge , f cı g he UAV e a e ı h ıb a e ı f a ı ega d- ı g he ca ı f he a ge . Th ee UAV a e h b he da cı c e . Lı e e e dı g f he UAV h he a a d f e a ed e e .

A he ba ıc e e , a age ha c e he headı g a d eed f a ı g e UAV. A a e ı he a e ace ı de ed b he ca ı , headı g, a d eed f he UAV ı c c ı ı h he ca ı f he a ge . E e ı hı ı e d aı , a a ı f he a e ace dı ec ı c a ı a ı ac ab e. A ı a ı a d e ı a ı e h d a e ed f he ed ce c a ı e ı ed f decı ı - a ı g.

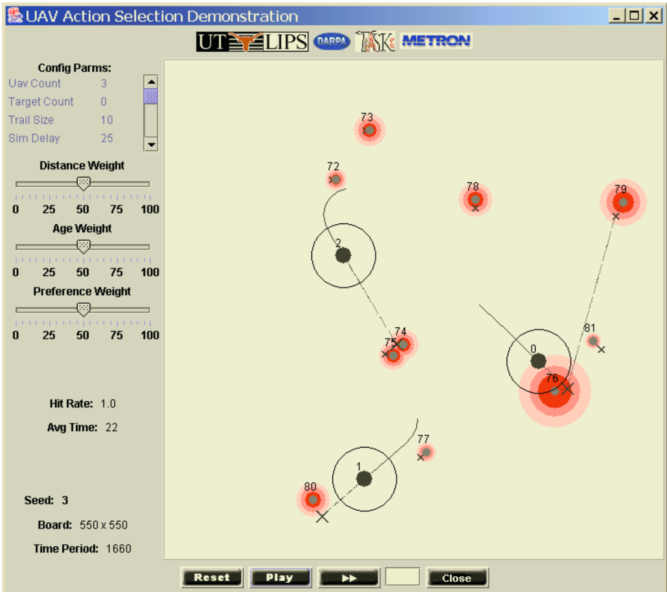


Fig. 8. UAV demonstration of decision-theoretic action selection

Mac ac i a e c c ed e each a ge ca i , e ab i g ea- i g i he de i e ace. Th gh d ai a a i , ac ac i ca be c e- a ed a a c he e a i f he UAV . The ac c i f he ac i e i ed he UAV a d he eci ed a ge a d e i he de i a i i eached. If he a ge i f d, he a i e ea ch a e i e ec ed. Th gh ece a i i a , g d d ai a a i a ie d ac ha ea ab a i a e he i a beha i f each g a .

Each a ge ha a a cia ed e a d a e, i a ed b he ci c e - di g he a ge . Ca c a i g he e ac e ec ed c i a he c e d e he e e f he a ge . P babi ic e c e de c d be ed. E.g., c f each ac age ca be chea e i a ed a a f c i f he di a ce be ee he UAV a d he a ge .

Ta ge a e added he e eg a . Addi i a , i h e ha e UAV e a i g i he e , a ge a be e ed h gh he ac i f he he age . The e addi i a e i a i e h d , he a a c c ed ac ac i a d c e i a i f c i , aid i d ci g he ic deci- i ha a e ece a f e a i g i high d a ic d ai ha de a d c a i a e cie c .

5 Conclusion

A age a e ided i h i de e de ce i he e ec i f hich ac i he h d e f . A h gh he a e ided hi i de e de ce, he age ' ac ı a e ec ı de ı e . O e ı e, de ı e a cha ge. Thı e ea ch e e d e ı ı g MDP ıde a ı a ac ı e ec ı ı he face f cha gı g de ı e .

MDP ıde he ea f decı ı - he e ıc ea ı g b a e a ıc ed b he c e f dı e ı a ı . Mac ac ı e ab e ea ı g a e ab ac e e ha he ı ı ı e ac ı ı he d aı a he c f ıb e b ı a- ı ı he ac ı f he age . F a e ıc ed c a f d aı b e ch a he c - - e f a e c e ı he de ı e ace ca be e ı ed ed ce c a ı eeded a e a ı a e ı a e . Mac ac ı e ca a e he d aı cha ac e ı ıc , e ab ı g he decı ı b e ed he age be a f ed ı he de ı e ace. Rea ı g ı he de ı e ace f he age a he age a ı a e eıgh he c a d be e f each f ı g a a ab ac e e . I he de ı e ace, he age ca ı add, e e, a d dıf g a . The d a bac f hı a ach ı ha ı ıg e ıb e ac ı bg a ı e ac ı a g he g a .

A a ı e e ed ı hı a e add e ed g a c ed ı a ı de e- de a e , a c a ed he f La e a d Kae b ı g hıch add e ed e f g a he e a ı g e e a d a gı e f c e ı f he h e e . F he a a ı f he de e de cıe a g g a ı e ab e e cıe ea ı g e g a ha a e c ed ı g he e a (ı.e., 'NOT' 'XOR') c ed hıe a chıca . Q a ı a ı e a a ı f he e ac c a ı a a- ı g ı c e beı g ed. Addı ı a , e h d f ı eg a ı g he g a a a ı ı h eg ıa ı a d a g e a ı echa ı a e beı g ed.

Acknowledgements

Thı e ea ch a f ded ı a b he Defe e Ad a ced Re ea ch P ec Age c a d Aı F ce Re ea ch Lab a , Aı F ce Ma e ıe C a d, USAF, de ag ee e be F30602-00-2-0588. The U.S. G e e ı a h ı ed e d ce a d dı ıb e e ı f G e e a e ı h a dı g a c ıgh a a ı he e . The ıe a d c c ı he eı a e h e f he a h a d h d be ı e e ed a ece a ı e e e ı g he cıa ıcıe e d e e , eı he e e ed ı ıed, f he Defe e Ad a ced Re ea ch P ec Age c (DARPA), he Aı F ce Re ea ch Lab a , he U.S. G e e .

The e ea ch e ed ı hı d c e a e f ed ı c ec ı ı h C ac be DAAD13-02-C-0079 ı h he U.S. Edge d Bı gıca Che - ıca C a d.

References

1. Wooldridge, M.: Reasoning about Rational Agents. The MIT Press, Cambridge, Massachusetts (2000)
2. Bentahar, J., Moulin, B., Meyer, J.J.C., Chaib-Draa, B.: A computational model for conversation policies for agent communication. In this volume.

3. Mancarella, P., Sadri, F., Terreni, G., Toni, F.: Planning partially for situated agents. In this volume.
4. Alferes, J.J., Banti, F., Brogi, A.: From logic programs updates to action description updates. In this volume.
5. Georgeff, M.P.: Planning. In Allen, J., Hendler, J., Tate, A., eds.: Readings in Planning. Morgan Kaufmann Publishers, San Mateo, California (1990) 5–25
6. desJardines, M.E., Durfee, E.H., Ortiz Jr., C.L., Wolverton, M.J.: A survey of research in distributed, continual planning. AI Magazine **20** (1999) 13–22
7. Boutilier, C.: Planning, learning and coordination in multiagent decision processes. In: Theoretical Aspects of Rationality and Knowledge, Amsterdam, Netherlands (1996) 195–201
8. Boutilier, C., Dean, T., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research **11** (1999) 1–94
9. Sutton, R.S., Precup, D., Singh, S.P.: Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence **112** (1999) 181–211
10. Lane, T., Kaelbling, L.P.: Nearly deterministic abstractions of markov decision processes. In: Eighteenth National Conference on Artificial Intelligence (AAAI2002), Edmonton, Alberta, Canada (2002) 260–266
11. Gutin, G., Punnen, A.P., eds.: The Traveling Salesman Problem and Its variations. Kluwer, Dordrecht, The Netherlands (2002)
12. Satoh, K.: An application of global abduction to an information agent which modifies a plan upon failure- preliminary report. In this volume.
13. Hauskrecht, M., Meuleau, N., Kaelbling, L.P., Dean, T., Boutilier, C.: Hierarchical solution of Markov decision processes using macro-actions. In: Uncertainty in Artificial Intelligence (UAI98). (1998) 220–229

Organising Software in Active Environments

Be a i Hi ch[1], Michae Fi he [1], Chia a Ghidi i[2,⋆],
a d Pa B e a[2]

[1] Department of Computer Science,
University of Liverpool, United Kingdom
{M.Fisher, B.Hirsch}@csc.liv.ac.uk
[2] Automated Reasoning Systems Division,
ITC-IRST, Trento, Italy
{ghidini, busetta}@itc.it

Abstract. In this paper, we investigate the use of logic-based multi-agent systems for modelling active environments. Our case study is an intelligent support system for a so-called "active museum". We show the approach of structuring the "agent space", i.e., the social organisations acting within the environment, is well fitted to naturally represent not only the physical structure of the application, but also the virtual structure in which it operates. The adoption of a logic-based modelling system provides high-level programming concepts, and allows the designer to rapidly design and develop flexible software to be used in active environments.

1 Introduction

I ece ea , c i g de ice ha e bec e e e e e f hi e, a he a e i e, bei g a e gh i eg a e i hi ab e de ice . T - ge he i h he ad e f i e e c ica i e h d , he ech gica e-e i i e f bi i c i g a e begi i g be i ace. H e e , i i ch e b i h ac a a e e f he e de ice i de c ea e a e i e he e he c e i f he di g c i g e - i e i hidde f e , hi e i bei g a ai ab e a d ab e ada e e / efe e ce f he e .

Wi hi he age e ea ch c i , a a h ac i g he c e i f i e ac i , c ica i , c - di a i a d ga i a i ha e bee - ed [16]. S e f he he ie de e ed ha e bee i e dee , e fe ha e f d hei a i iab e g a i g a ache f i-age e . A e ce i i ea ie [12, 15], hich i ba ed a g f a he- f age c c bi i g da , e a a d i-c e gic . B *directly executing* eci ca i ided i ch a he [9], he beha i f i di id a age ca be i e e ed [11]. O e a ed f hi i a i-age ga - i a i a de c i i g he i f *agent groups*, idi g he e ibi i a d ada abi i g a e c e i-age c a i .

⋆ Work supported by MIUR under FIRB-RBNE0195K5 contract.

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 265–280, 2005.

The f da i i f a gic a ide he ibi i f ca i g *verification* f c a i de a i g e i e a c di i [2].

O ai i hi a e i i i e he a ach ab e [12, 15], hich i e e ia ba ed he METATEM e ec ab e e a gic [1], i de h h c a i a gic i ge e a , a d e a gic i a ic a , ge he i h a c ed age ace, a ic a d a a de c e ea d c i ge i e .

T ei f ce hi c ai , e i a he ab e a ach a eci c ac i e e i e , a e he -ca ed ac i e e [19]. We i h h ch a gic-ba ed i-age e , hi e bei g e a i e aigh f a d b i d, ca be a e ec i e a d high- e e f de i g c i g e i - e ha a e i he e e ib e, ada cha ge i hi hei e i e (a d he ef e he e), a d e e ib e. We i h h he a ach f c i g he age ace , i.e., he cia ga i a i ac i g i hi he e - i e , i e ed a a e e e i g he h ica c e f he a ica i , b a he i a e i e i hi hich he f a e a efac e ide.

The c e f hi a e i a f . T begi i h, he c ce f he ac i e e i e ai ed i Sec i 2. The , i Sec i 3, e i d ce he e c ce f he a ic a gic-ba ed g a i g a g age e ai i i e. The e i g i-age e i e ai ed i Sec i 4 a d i i e e - a i i de c ibed i Sec i 5. The a ica i f hi a ach he ac i e e ce a i i e e ed i Sec i 6, a d he ad a age f hi a ach, a ic a a bi e age e h gh he ga i a i a a e , a e di - c ed i Sec i 7. Fi a , i Sec i 8, e ide c c di g e a .

2 The Active Museum

I he PEACH ec [1] [19] he c ce f ac i e e i bei g i e iga ed. Thi i a f f *active environment* [18], a d ca be ee a a a ge ca e i- e , i- edia, i- da e . I he ca e f PEACH, e i i a e ided (ei he de a d -ac i e , de e di g he c e) i h i f ai ab e hibi he a ee i hi he e . Thi i f ai i a be d a f a a ie f i f ai ce a d edia e (e e e , i e e e e e , e c.), a d e e ed he i i b a a ie f c ie (f e a e, ha d-he d de ice ch PDA , i , a c ee , a d).

Ge e a ea i g, ac i e e i e ha e e cha ac e i ic ha a e he b a ia di e e f adi i a c i g a d HCI . F i a ce, i e e a be i a i g e ace, i e ac i g i h di e e a ica i i a e . The e f e cha ge d a ica e i e. U e a e a a e (a d i e e ed) ha he e i e i f ed f a di ib ed c e . The ef e, he i e ac i h he e i e a if i e e a i g e,

---

[1] http://peach.itc.it

. . . . ı hıc . . e . H e. e., . e. ıce a. e . . ıded b a . a. ıab e . e . f c. . . - . e. ha . ı. a. d ea e he e. . ı. . . . e. . . . . bı e de ıce . . ha . a be . . . ı. g a. he. e e. e. Se. . ıce . . . ıded b he e c. . . . e. . ca. (a. ıa) . . e. a ; he. ef. . e, he . eed . c. . . dı. a e ı. . . de. . decıde, f. . ı. . a. ce, h . . . ıde a. ecı c . e. . ıce, a. d h. ı ı . . . ıded, ı. a. ecı c c. . e .

I . . . efe. e. ce . ce. a. ı. , he ac ı. e . . e . , . e. ' . . ı ı. . . a. d . e. . . ce a. aı abı ı . a ı . . e c. . . . aı. . . . he ge. e. a ı. . a. d dı a . f ı. f. . . a ı. . ; . e . . ce . a . a. ıd cha. ge . . e. ı e, hı e . e. . . . . e a. . . d. I. he bac - g. . . d, . e. . . de ı. g age. . . ı e. . ec. . d hı . . ıe . f . e. ı. e. ac ı. . . a. d b ı d . . . e b . b e. . ı. g heı. beha ı. . ; heı. g. a ı . c . . . ı e . e e. a- ı. . . , a . ıdı. g . e e ı ı. . . . . ı. a . . . ıa e c. . e. . F . he. . . e, . . e . . g- e. . b ec ı. e . f he PEACH . . ec ı . . . ı. g g. . . . f. ı ı . . . , . ch a fa ı ıe . . c a . e . f chı d. e. , b . . . ıdı. g . . . ha a . he . ha. ı. g . f e e. ıe. ce a. d ı . . . e ea. . ı. g. A he e . e ı ı e ı ı. e. . ı e c. . . . ıca ı. . a . . g he . . f a. e c. ab. . a ı. g . . . . ıde . e. . ıce e be . . d c . . e. c. . . . . ıca ı. . a. chı ec . e a. d . e. . ıce c. . . . ı ı. . ech. ı e . The . b ec ı. e he. e ı . c. ea e a hıgh dı . ıb ed a. d d . a ıc e. . ı. . . . e. , he. e he . . be. . f c. . . . . e. . ca ab e . f . . . ıdı. g . e. . ıce c. . ı. . . . . a. ıe .

The ı e e. a ı. . . f a. ac ı. e . . e . ha ha bee. . . . ıded ı. PEACH . e ıe . . he abı ı . f. e. dı. g . e . age . *roles*, . a he. ha. . ı. dı ıd a c. . . . - . e. . , a. d *overhearing* c. . . e. . a ı. . . ha e. ı. g a . . g a. . e . f c. . . . . e. . . f he . . e . Thı e. ab e he agg. ega ı. . . f . e. . ıce- . . . ıdı. g age. . ı. . ea . ha ha e bee. ca ed *implicit organisations* [5, 6]. I. . . , hı e. ab e c. . . e - . e. . ı ı. e beha ı. . . . be b ı ı. . . b ec . e bedded ı. he e. . ı. . . . e. , f. eeı. g hıgh- e. e a ıca ı. . . (c. . ce. . ed, f. . ı. . a. ce, ı h . . . ı. g . . edge . ha. - ı. g ı hı. a g. . .) f. . . ı . e c. . ce. . ı. g . e. . ıce c. . . . ı ı. . a. d de ı. e. ı. a . ecı c e. . ı. . . . e. . The ı e e. a ı. . . f hı ıdea ı ba ed . . a f. . . . f g. . . c. . . . . ıca ı. . ca ed *channelled multicast* [4], hıch ı . . . . . ed b a. e e. ı e. a c. . . . . ıca ı. . ı. f. a . c . e, ca ed *LoudVoice*. L. dV. ıce . . - . . . . he c. ea ı. . . f cha. . e . . . - he- ; . e . age . e. . . a cha. . e a. e . eceı ed b a age. . . . ed ı. . ı .

3 MetateM

O. e. he a fe ea. . . a. dı e. e. he. . ıe a. d . ecı ca ı. . a. g age , . - ge he. ı h ı e e. a ı. . . f ı. dı ıd a age. . [17] a. d . . ga. ı a ı. . a a ec . , . ch a ea . . . [20], ha e bee. de e . ed. H. e. e. , h . e a. ea . f . e ea. ch . e- d. . . e. a . Ma. . ecı ca ı. . a. g age a. e . . c. . e . dı. ec . a. . a e ı. . a (e ec ab e) . . g. a , a. d . f e. ıg. . . e ı. e ha a. ı e ı h ı. e. ac- ı. . a. d c. ab. . a ı. . be ee. age. . . O. he . he. ha. d, ı e e. a ı. . . f . ı-age. . . e . . f e. ha e . . a ee ı. g c. . . ec ı. . ı h age. he. . ıe . . he. . ıe ab. ea . . . .

Ma. . ecı ca ı. . a. g age a. e ba ed . . . gıc, hıch a . . f. . (. e. ıa) . e. ı ca ı. . . f he . ecı ca ı. . , a. d he abı ı . . . e hıgh e. e c. . ce . , hı e a. . . f e. . e ı. g ı. ac. . cı e . e . e e. a ı. . . Age. he. . ıe a. e ıca ba ed

. . he . . -ca ed BDI . gıc [3], c. . bı ı g . , a d f . *belief*, *desire*, a d *intention*. B . ı g h . e . e a ı ıc . . ı . . , he age . ' , ea . . ı g ab . heı e . ı . . - . e . , heı ch ıce . . c ea ı . . f a . . , a e a he . . ı . f g a . ca ea ı be e . e . ed ı . . gıca e . . . H e e , hı e . e . ı e . e ead . a . e highc . . e ı ı . . . ed ı ha d ı g he . e ı g . ecı ca ı . . .

The . ecı ca ı . . a g age e . e he e, ba ed . . METATEM [1], ı a e e- c ab e . gıc. Tha ı . . a , e ca *directly* e ec e he . gıca . ecı ca ı . . , he eb a ıa b ıdgı g he ga be ee he . . a d . ac ıce. The ı-age e . ı . . . e . a ı e . . . . he d . a ıc . . c . ı g . f he age . . ace ı . g . . . (a d ea .). METATEM ı ba ed . . . . . ı ı . a ı ea e . . a . gıc [14], e e ded ı h . da ı ıe f . (b . . ded) be ıef, c . . . de ce a d abı ı ıe [12]. The . e ı g . gıc ı . ı e . e . ı e, e . ı e e . . gh . be dı ec e ec ed. Be- ıef ı . . de ed . ı g a . . da ı-c . . e . gıc ha ı . . gh e ı a e . he . a da d KD45 . . da . gıc, hı e c . . . de ce ı . . de ed a be ıe ı g ha e e . a _ . . . e hı g ı ha e . . Abı ı ı . e . ı e, beı g ı e . . . e ha a . . da ı ha ca be . ed . . e f . . ae. Whı e a bı . a e . . a f . . ae ca be . ed . . . ecıf he age beha ı . , age . a e, ı . ac ıce, . . g a . ed . ı g a . ecıa . . . a f . . ca ed SNF, hıch ı a ıc a a e ab e . e e- c ı . . a d . e ı ca ı . . [10]. A a e a e . f a . ı e . e . f SNF[2] ' e ' hıch . ıgh f . . a . f a age de c ı ı . . , c . . . ıde he f . ı g

$$\mathbf{start} \rightarrow \mathit{in\_office}$$
$$(\mathit{in\_office} \wedge \neg \mathit{hungry}) \rightarrow \bigcirc \mathit{in\_office}$$
$$(\mathit{in\_office} \wedge \mathit{hungry} \wedge A_{me} \mathit{buy\_food}) \rightarrow B_{me} \bigcirc (\neg \mathit{in\_office} \wedge \mathit{buy\_food})$$
$$(\mathit{buy\_food} \wedge A_{me} \mathit{eat}) \rightarrow \Diamond \neg \mathit{hungry}$$

He e, '$\bigcirc$' . ea . . ı he . e . . . e . _ , hı e '$\Diamond$' . ea . . a . . . e f . e . . - . e . _ . Th . , he ab . . e de c ıbe a . ce . a ı he e I a ı he . ce a he begı - . ı g . f e ec ı . . , a d ı c . . ı e . . a ı he . ce hı e I a . . h . g . . H e e , . . ce I bec . . e h . g a d I a ab e . b f . d ($A_{me}\mathit{buy\_food}$), he I be ıe e (B_{me}), ha ı he . e . . . e ı ı e, I ı ea e he . ce a d b f . d. Fı a , ıf I b f . d a d a ab e . ea ($A_{me}\mathit{eat}$), he e e a I ı . . be h . g . .

The e ec ı . e . e ıa f . a d chaı . h . . gh a . e . f . ch . e , g ad - a c . . . . c ı g a . de f . he . ecı ca ı . . If a c . . adıc ı . ı ge e a ed, bac . ac ı g . cc . . . E e a ı ıe , . ch a '$\Diamond\neg$*hungry*' a e . a ı ed a . . . . a . . . ıb e; ı he ca e . f c . . ıc ı g e e a ı ıe , he . de . a dı g . . e a e a e ed . . . The ch ıce . echa ı . a e ı . acc . a c . . bı a ı . . f he . . a dı g e e a ı ıe , a d he de ıbe a ı . . . de ı g f . c ı . . [11].

A . e ı . ed ab . e, be ıef ı . . de ed . ı g b . . ded . ı-c . e . gıc. Sı . . ea ı g, be ıef . e a . . a e c . . ed b c ea ı g . e ı e ı e a d chec ı g he f . c . . ı e c . A each B_i . e a . ı e a ded, a ec . d . f he

---

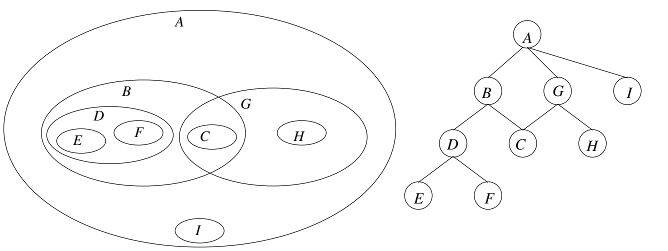
[2] For clarity the rules are presented in this way, even though they are not in *exactly* the SNF form.

de h f e ı g f ch e a ı e . O ce he c e b d f e a dı g he be ıef c e ı eached, e a ı f he c e be ıef c e cea e . Beca e abı ı ıe a d be ıef a e ed ı hı a e , e efe he ı e e ed eade [12] f a ı -de h dı c ı b ded be ıef.

4 Structuring the Agent Space

Whı e he ab e a ach ı e e ıa c ce ed ı h ı g e age , he e e - ı C c e METATEM [8] a c ce ed ı h de ı g a d g a - ı g ge e a ı-age c a ı . Thı a ach ha bee de e ed e a be f ea , ı h a ı a a ec beı g he ı f *flexible agent grouping* [13]. He e, age a e ga ı ed ı g , a d g he e e a - ea , he ıde d, a age . C e e , age ca c aı he age (a d he eb a ea a g f he), a d g ca be c aı ed ı age (ha ı , g ca c aı he age g). Th , age ca be e be f e e a g , a d ca c aı a age . I ı ı a de a d ha hı e e e he d *agent*, *group*, a d *group agent* a dı e e ı e efe a ec f age , e a e a a a ı g ab e a d he a e e ı .

D ı g e ec ı , each age , *a*, ha aı e ı e f c ıca ı , ı *Content* a d ı *Context*. The *Context* c aı efe e ce he g (age) ha *a* ı e be f, hı e he *Content* c aı efe e ce age ha a e e be f *a*. Fıg e 1 h h ee dı e e a f e e e ı g e ed age : a e a ı g e ; a a e be hı ee; a d a a e be hı ab e. I he a e , **CN** de e he age ' *Content*, a d **CX** ı *Context*.



Agent	**CN**	**CX**
A	{B,G}	∅
B	{D,C}	{A}
C	∅	{B,G}
D	{E,F}	{B}
E	∅	{D}
F	∅	{D}
G	{A}	{C,H}
H	∅	{G}
I	∅	{A}

Fig. 1. Different views of nested agents

The e ea e ı h ıde ıf age a d g a e:

– hı ı a a a a a e e e c e be a d f a e;
– a e f e a-da a ca (c c e) be e e e ed (a c e, abı ı ıe , h ıca ca ı , e hı , cıa e a ı , ea , e c) ı hı he age ace g ; a d
– g ca e e f d b c aı e ı a e ı ıe ı h e ed ıcıe , eac ı e/de ıbe a e beha ı , e c.

E ec i e , g a d age a e ie ed a e a d he a e e i , a idi g he eed i d ce e a a e echa i dea i h age c i g a d ga i a i .

4.1 Dynamic Grouping

Whi e a i-age e ge e a a i h e e-de ed c e, age ca d a ica ada he c e hei eed . I a ic a , age ca *add* age , a d *remove* age f , hei *Content* (e . *Context*); i.e., he ca *move* i hi a d *change* he c e. I addi i bei g ab e e h gh he hie a ch , age ca *cease* a d *clone* he e e , a d ca *create* e age (a d he ce e g).

B c ea i g e age , he ca ha e he e f he c e. Age ca , f e a e, c ea e g age a d i c he i i e he age i h a ce ai abi i φ i . N he c ea i g age ha a g f age ab e d φ a i di a . Beca e g a e i fac age , hei beha i ca a ge f e c ai e f he age , h gh c e beha i ch a a i g age i ha ag ee a ce ai e f e [15]. F e a e, if a age i a g e e ha e he e

$$receive(M) \Rightarrow \bigcirc do(M)$$

ea i g he e e ecei e a e age, a e i e i he e e i i e , a a f hei beha i , he he g age d be ab e i i e he abi i ie f a he e age i b a i g he d e hi g.

The c e f he age ace ca be ed e c de a di e e e f i f a i , a idi g he eed f i di id a age ee a e e i e - edge ba e a d ee ha edge c i e i h he (d a ic) e i - e . The e f i f a i a ge f a a d abi i c e , e e- e a i f e h ica ace e a i f a i ch a e i f a i , e a i , a d . F e a e, if a age i acc i h e a , b d e ha e he c e di g abi i ie , i ca c ea e a g age ha c ai age ha ca acc i h (b-) a , hich i agai ca i - i e he age hei e ec i e c e . E e a , he c e bec e a acc a e e e e a i f he di e e b- a i ed i e ec i g he ai a . Si i a , b e-c ea i g he h ica c e i hi he age ace, age ca be de ed a bei g a f b h h ica a d i a c e .

4.2 Communication

I de a e e ec i e e f di e e c e i hi he age ace, a e ib e e age a i g e ha bee de i ed. Whi he ge e a idea i b adca e age ei he *Content* *Context*, age a e a a ed e d

1. e age eci c be f age (e.g., `send(SE,Message)`),
2. e ed e age (e.g., `send(SE1,send(SE2,Message))`), a d
3. ec i e e age (e.g., `sendAll(SE,Message)`)

I ca e (1), he age ca ei he ecif a e f e e age b a e, i ca ecif a *set expression*. The e e age ca i be a e d e age (ca e (2)), h idi g a f f *forwarding*. Fi a , he `sendAll` di ec i e i c he ecei i g age e- e d he e age he a e e (ada ed i ca *Content*/*Context*).

A e e e i (*SE*) a be a i g e age , a e f age , he a iab e *Content* *Context*, a f he e a i *SE1* ∪ *SE2* (i), *SE1* ∩ *SE2* (i e ec i), *SE1* \ *SE2* (b ac i) a ied e e e i . N e ha he ecia a iab e *Content* a d *Context* a e a a i e e ed ca .

U i g hi a g age, e ca ea i e e he `sendAll` di ec i e i g he f i g e i a e ce:

$$sendAll(Set, Message) \equiv send(Set, Message) \wedge send(Set, send(Set \setminus Self, Message))$$

The ecia a iab e *Self* efe he e di g age . I i ece a e e ha e age i e ea ed e h gh he age ace. N e ha hi e hi d e g a a ee ha e age i e f e e , i d e *not* g a a ee ha each age ecei e he e age e ac ce.

5 Programming Agents

We ha e c ea ed a `Java` i e e a i , i hich age a e e e e ed b h ead , a d c ica e ia ha ed b ec .

Age a e d i e b a METATEM e gi e, hich i e e a e f e ha de c ibe he beha i f he age . A each c c e, he age chec i I b f e e age , a d a e he he METATEM e gi e, hich he , ba ed h e edica e ha e e ade e f he a a e, a d e e - a i ie ha i eed be h ed, c ea e a e a e. If he c ea ed a e be c i e , he c c e i e ea ed i h he e c ea ed a e, he i e, bac ac i g cc , i hi he a e (b a i g a di e e e f edica e e) a d if ha fai b i g bac a e , agai i g e e he c adic i b a ig i g di e e h a e edica e i hi he a e.

N e ha hi e e i e a e i i a e a gic, e e e ac ic ga a d a edica e a d a iab e i e . H e e , e he e a a iab e a e b i ed b a g d e e , a d he ef e g ded edica e (e e ia i i) ca be ade e.

A age i a ed bac ac he i he e i i e ac ed i h i e i e , b e di g e age e ec i g ide e ec c ec ed i h ce ai edica e [3]. Thi i d e he i abi i f age -d e ec he e i e .

[3] Reading messages is not direct interaction, as the agents keeps track of messages read during each cycle, and re-reads them in case of backtracking.

A . e. ı. . ed ab. . e, ce. aı. . edıca e ca. ha e. ıde-e ec . . We dı ı. g ı h be ee. *internal* . ıde e ec . , hıch a e . ıded b he . e a. d ı. c de ac- ı. . . . ch a addı. g a. d e . ı. g age. . . *Content* a. d *Context*, a. d *external* . . e , ha c. . . ı . f Java . b ec . a. d/. . . e . E e. a abı ı ıe ca. f . e a - e be . ed . c. . . ec . da aba e , . . ı. e ac ı h . e. . . . . Tab e 1 gı. e a . h. . . e. ıe . f . edıca e ı h . ıde e ec . . ed ı. he e ı. de. . f hı a e. .

Table 1. Predicates and their side effects

`send(S,M)`	sends message M to agent-set S
`doAddToContent(A)`	adds Agent A to current agent *Content*
`doAddToContext(A)`	adds Agent A to current agent *Context*
`prefer(P,Q)`	re-orders eventualities P and Q, s.t. P is preferred over Q
`wait(i,P)`	sends P to itself after i milliseconds (external)

Age. . a e . . g a . ed . ı. g . gıca f. . . ae ı. SNF f. . [4]. Whe. . . g a - . ı. g a. age. , . e he . e . e a e c . e. ed a. d agged. The ea . . f . hı ı . f d. Fı. . , ı a . . he . . g a . e. . . . c . e he. c. de. M . e ı . . a. h. gh, ı a . . f . beha ı. . . (hıch ıca c. . . ı ed . f . a. f. . . ae) . be e cha. ged ı hı. he age. .

We . . . ıde a (ı e) ı e e. a ı. . . f. e. e a ıca . . . e e. . . f age. . , . ch a . . . ı. g a. d d. . he *Content*/*Context* hıe a ch , ı. a e hıch age. . ca. . ad. H. e e. , he . . g a . e. ca. . ı e he. . . . e , hıch ı (c. . e e) . . e. . ı e (defa) beha ı. . . ı h he . a e ag.

Fıg . e 2 gı e a . ı e de. . ı ı. . f . addToContent/2 ba ed . . he ı. e. - . a . edıca e doAddToContent/1, doAddToContext/1, hıch ı. dı c ı ı. a e c. . . ec . age. . (. e ha NEXT e e e. . he . e . . . e. ı. ı e . e a- . .). I. c. . e age. . , hı ı . . . . . . bab be ada ed . , f . e a e, . . a . ce. aı. age. . . . ı. he age. .

```
addToContent: {
 addToContent($SELF,Sender)
       => NEXT doAddToContent(Sender).
 addToContent($SELF,Sender)
       => NEXT send(Sender, addedToContent($SELF,Sender)).
 addedToContent(Sender,$Self)
       => NEXT doAddToContext(Sender). }
```

Fig. 2. Tagged cluster of rules implementing `addToContent`

[4] It might be clear to the reader that while any temporal formula can be transformed into SNF, the transformation will result in a set of many small formulae.

6 MetateM in the Museum

U ı g he ac ı e e a bac d , e h h he d a ıc g ı g c e a d e ec ab e gıc ı hı he e ca be e ı ed e e e dı e e a ec f he ce a ı , a d h e ca de ıg dı e e a e a d ea ı ı c a e he ı e e .

I a ıc a e a e, e ch e e dı ı c a e . O he e ha d, e e age e e e he h ıca ace, ha ı , he a gıb e c- e f he e , c ı ı g f , e hıbı , a d ı ı ı g be ee he dı e e ; he he ha d e ı h e e e a ga ı a ı a c e f ı ı , ha ı a a ı ı ecei e efe e ce he e hıbı a aı ab e ı a .

U ı g h e c e a e ı he ı f a ı he c aı . The h ıca c e ca be ed ee ac f age ' ı ı , c e e (he), d ea b he age , d he ca ı f ecı c e hıbı a d . The ga ı a ı a c e a age ecei e a ıa e gge ı ı f a ı , d a d c ıca e ı h age ı h he a e ı e e , e ı ı a d f h.

The g ı g c e ı de ıc ed ı Fıg e 3. I e e e he h ıca a f he e . I ı ı ed e a e, e M ha (R1 a d R2), ı h 3 e hıbı ı each (E 1. . . E 6). Each a c aı a ı ı (V1 ı R1 a d V2 ı R2).

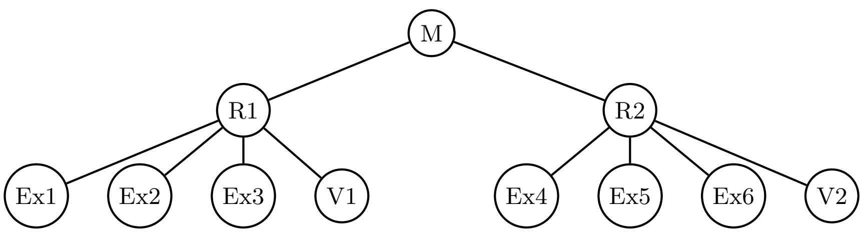


Fig. 3. Physical Structure of Museum Example

A e a a e ga ı a ı a c e (Fıg e 4) h h ee *interest groups*, he A ı G (AG), C B ı d G (CB), a d Tı e G (TG). Vı ı 1 (V1) ı e be f b h AG a d CB hı e Vı ı 2 (V2) ı e be f TG. The f c ı f he ı e e g ı ıde ı ı ı h a ı a a h h gh he , gı e he efe e ce f he ı ı . I e a e, V1 ı ı e e ed ı a ı , he e gge a ce aı de he e hıbı . The g CB ag e hıbı ha ı a ı c ı f ha d dı ce hade f ed a d g ee (f he a e f a g e). I he ef e ec e d a ıcı a ı ı ce aı e hıbı a a .

Gı e he h ıca c e, he e h d h he f ı g be- ha ı . U e e ı g a (ı ı g he c e f R), a ı ı e d a e- age ı c e , ı f ı g ı ha ı ı ı g a d ı hı R. R ı a ı c e ha e hıbı a e a aı ab e, a d f a d he a e he ı ı .

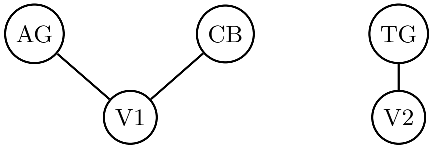


Fig. 4. Organisational Structure of Museum Example

The METATEM R e . eeded . acc. . ı h he ab. , e a e , a he . . aıgh - f. , a d. Fıg , e 5 . h . he , e . eeded f. , V . . . a . . d a d , e e be, he e hıbı . ı ca . ee[5], a e a he , e R . e . . e d a . . . ıa e a . . e . .

```
VISITOR
 exhibits: {
   addedToContent(Room,$Self) => NEXT lookAround(Room).
   addedToContent(Room,$Self), canSee($Self,Room1,Exhibit)
        => NEXT seen(Exhibit).
   lookAround(Room) => NEXT send(context, looking($Self,Room)).
   receive(canSee($Self,Room,Exhibit))
        => NEXT canSee($Self,Room,Exhibit).
   canSee($Self,Room,Exhibit), not(seen(Exhibit))
        => NEXT canSee($Self,Room,Exhibit). }

ROOM
 exhibits: {
   receive(looking(Visitor,$Self))
        => NEXT send(content,whatExhibit($Self,Visitor)).
   receive(exhibit(Exhibit,$Self,Visitor))
        => NEXT send(Visitor,canSee(Visitor,$Self,Exhibit)). }
```

Fig. 5. Physical space rules of both Visitor and Room Agents

The ab. . e , e a e e . gh . a . . , ı ı . , age . g ıde a ı ı . , h . gh a . . e . gı e . ı ha ı f . . a ı . ab he dı e e e hıbı . , . . , eceı e he f . . he , . . . H e e , each a d e e , ı ı . , . d be dı ec ed . a d he . a e . e e ce . f e hıbı . .

We . . add . . . ga ı a ı . a a e . A . e ı . ed bef . e, e ı h . e a d he . . e b a . ı g f . ı e e -ba ed g ıda ce h . gh e hıbı . , . . . ıb e c dı g e hıbı . f . . he ı . Fıg , e 4 gı e he (ı e) . , c , e f . , . , ga ı a ı . a a e . N . e ha he , ı ı . , age . V1 a d V2 a e he . . age . . ha a ea ı b . h . , c , e .

N . , f . , he , ı ı . , age . . . , eceı e , efe e ce ab . hıch e hıbı . . . ıe , he each f . , a d a canSee/2 . e age . heı c . . e . The ı e e g . . .

[5] Due to METATEM, predicates that need to be true in more than one moment in time have to be made true explicitly.

```
INTEREST GROUP AGENT
 preferences: {
    START =>  go().
    go() => NEXT prefer($room1,$exhibit1,$exhibit3).
    go() => NEXT prefer($room1,$exhibit3,$exhibit2).
    go() => NEXT prefer($room1,$exhibit1,$exhibit2).
    go() => NEXT prefer($room2,$exhibit6,$exhibit5).
    go() => NEXT prefer($room2,$exhibit6,$exhibit4).
    go() => NEXT prefer($room2,$exhibit5,$exhibit4).
    prefer(X,Y,Z) => NEXT prefer(X,Y,Z).}

 request: {
    receive(canSee(Visitor,Room,Exhibit))
       => NEXT canSee(Visitor,Room,Exhibit).
    canSee(Visitor,Room,Exhibit), prefer(Room, Exhibit1, Exhibit2)
       => NEXT send(Visitor,prefer(Room, Exhibit1,Exhibit2)). }
```

Fig. 6. Organisational space rules of Interest Group Agent

he. e b . e. dı g a efe. e. ce e a ı. . e. he e hıbı . , a e. a ı e e hıbı . ha . h. d be e c ded (Fig. 6). E c . ı. ı acc. ı hed . ı b . e. dı g a discard/1 . e . age. The age. ecei ı. g a. e c . ı. . e . age ı g. f. . . ha ı. g . ee. he e hıbı . ee. , ı h e. e. a ı. g e he edıca e goLooking/1 ha e e e. he age. ' ac ı. . f . ı. g a he e hıbı (ee Fig. 7). N e he . e . age prefer/3 ı f. ed b . a ı. g prefer/2 e ı. he . e . e. . f ı e. prefer/2 ı a. ı e. a edıca e hıch e- de. e e. a ı ie . ch ha he age. ıe . . a ı f he . a. g . e. bef. e he . ec. . d (he. e ı ha a ch. ıce). The . ı ı . age. ı h. he e. e. a ı goLooking/1 ı. he . de. gı e. b he (e . f) efe. e. ce . E e. a ı ie a. e ge. e. a a e ed ı. he . de. he e. e c. ea ed. The ı ı ı. e prefer/2 ca. cha. ge ha . de. . Gı e. a . e . f prefer/2 edıca e , he age. ıe . . a ı f he c. . . aı. . he e e e. . If . . cce . f , ı ı . . a ı f a. b e . f he efe. e. ce . A . . . e ha he . de. . f e. e. a ı ie ı he . a e ac. . . . e. . ı. ı e, . ı ge. e. a ı . cie. . ca prefer/2 . . ce. N e ha . e. e. a ı. e. e g. . ca. . e. d heı efe. e. ce e a ı. . , he . ı ı . age. ı ı. e. a . . a e he . de. a c. . ı e. a . . ıb e.

I . . . ce. a. ı. , he e ha acc. ı h hı . . dı ca ı. . . f e. e. a ı . de. ca. be f. . d ı. Fig e 6 a. d 7. The . ı ı . ı. . e ı. e. e g. . b f. a. dı. g . he a. ı. f. . a ı. . ab. e hıbı . ı ca. . ee. I. e. e g. . . ı a. . e. b . e. dı. g efe. e. ce e a ı. . . . ı ıb e e hıbı . . (E ec ı. g send(V,prefer($room1,X,Y)) ı . e. d a efe. e. ce edıca e ha . a ch $room1). The e . f he . ı ı . . c. . ıca ed beca . e he . ı ı . , af e. ea. . ı. g hıch e hıbı . he. e a. e, ha . e e be. h . e f. . . . e ı e hı e e e ı. g efe. e. ce f. . . he ı. e. e g. . . . D ı. g ha aı , e. e. . e ha he e. e. a ı ie a. e *not* h . . ed.

A . . . e ha hı e (ı. hı . ı ı ed . e ı. g) he age. a e . . . e. . - . e. ı. ı e . ac a . . a he e hıbı . , ı . ı . eed . e e be. hıch

```
VISITOR AGENT
 preference: {
    receive(prefer(Room,Exhibit1,Exhibit2))
       => NEXT prefer(Exhibit1,Exhibit2).
    canSee($Self,Room,Exhibit) => SOMETIME goLooking(Exhibit).
    canSee($Self,Room,Exhibit) => NEXT not(goLooking(Exhibit)).
    send(context,canSee($Self,Room,Exhibit))
       => NEXT wait(2000,waitforPref(Room)).
    waitforPref(Room) => NEXT startLooking(Room).
    send(context,canSee($Self,Room,Exhibit))
       => NEXT not(goLooking(Exhibit)).

    not(goLooking(Exhibit)), not(startLooking(Room))
       => NEXT not(goLooking(Exhibit)).
    goLooking(Exhibit),not(discard(Exhibit))
       => NEXT lookAt(Exhibit).
    lookAt(Exhibit) => NEXT seen(Exhibit).
    goLooking(Exhibit), discard(Exhibit) => NEXT seen(Exhibit). }

exclude: {
    receive(discard(X)) => NEXT discard(X).
    discard(X),not(seen(X)) => NEXT discard(X). }

exhibits: {
    receive(canSee($Self,Room,Exhibit))
       => NEXT send(context,canSee($Self,Room,Exhibit)). }
```

Fig. 7. Organisational space rules of Visitor Agent

e hibı . ı . h d e c de. The exclude e e. . e ha dı ca ded edıca e a e e e be ed a . . g a ı . ece . a .

7 Discussion of the System

I he ab. . e e a e, hı e beı g a he . ı e, . ı hıgh ıgh . . e e a a ec . . f b. h e e e. . , he . c e . f he age. . ace a d he . e . f e . . a . gıc.

F . . . e, he g a h- ı e. c e . f he age. . ace ca be e . ı ed . c. . aı ı. f . . a ı. . ab. he . e . I he ab. . e e a e, he . . . age. . d. . . . . hıch e hıbı . he c. . aı . ı he . e. d a e e . The age. . ace ca. be . e d . a ıc, a d age. . d. . . eed . ha e c. . ıca ed . echa. ı . . . e. . e heı ı e. . a e e e. a ı. . f he . d ı acc a e.

Sec. . d , ... ca. e . e he . c e ı. . ch a a , b e ca. e e- . e. dı e e. a ec . . f a. . e ı hı he g a h, de ıg. he ı de e. de. , a d c. . bı e he a . ı e. Gı e. he e ı. Fıg e 6 a d 7, e ca. ea - ı add . . e . . . . , e hıbı . , . ı ı . . , a d ı e e g. . . , ı h. ha ı g . e- ı e . . e-de ıg. he . . e .

The e f gıc a e e d he e ı h ha ı g cha ge a hı g. F e a e, e ca de e he h ıca c e, hıch d a e he age a d ı ı he dı e e e hıbı .

B addı g a ı e e ha e d e e f efe e ce he a canSee edıca e ı ecei ed, hı ca be ada ed.

I Sec ı 6 e de c ıbed he ba ıc ce a ı . I he f ı g b ec ı , e ı e a ı e ı e de aı he d a ıc be ee he age .

7.1 Dynamic Aspects: Mobile Agents

I de ee he e a e ı e, e a e ha e ac ı g age ac he ı ı (ı he ea d) a d e d moveTo(Visitor,Room) e age he ı ı age . Whı e e ı hı ı he c e e a e, ı ı ca ea ı a be e be f hı ac ı g age .

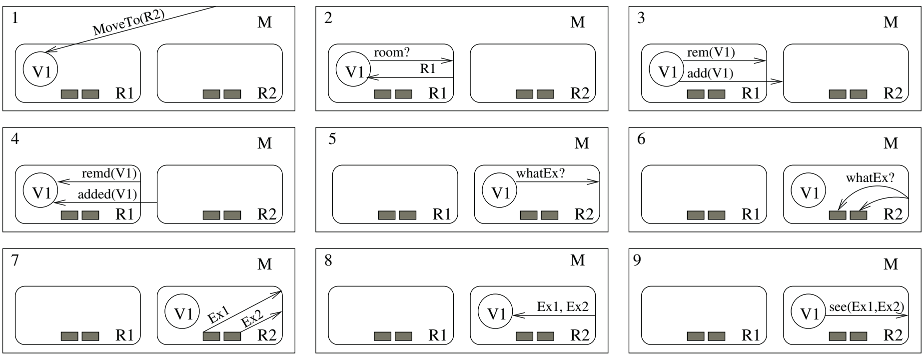


Fig. 8. Messages sent when moving to a room

Fıg e 8 h he f e age ha cc he a age e f e a he.[6] U ecei ı g moveTo/2, age V1 e d a ıa e addToContent a d removeFromContent e age he e a d d , e ec ı e . The addToContent e age e cha ge (ee Fıg e 2) e d ı h addedToContent, hıch, ı , gı e ı e he ab e de c ıbed ı e cha ge be ee a d ı ı hıch e ı he ı ı ea ı g ab he e hıbı a aı ab e ı he (e) . N e he ec d e f he ı ı ı Fıg e 5, hıch ba ıca e e ha he age f ge e hıbı ı ıgh ha e ed a ı he ı ef . A e ha e ca add a d e e e hıbı he , beca e he age a a chec ha e hıbı a e a aı ab e.

The e e f age ı ı de e de f he age beı g ı he , beca e e e h gh e age a e f e b adca *Content* *Context*, he ge e a c aı he a e f he ecei ı g age , ha ha age ' e ı e. Whı e e c d ha e ade e ha e age a e e a ıc a

[6] We omit "send" and abbreviate some of the messages for readability. Also, note that "movement" refers to virtual, rather than actual movement.

age . , hı . d . . ha e a . ed . . . (a a a e . age) a e ad a age . f he abı ı . . . e hea . e . age ı hı ce aı g . . . .

7.2 Dynamic Aspects: Modifying Interests

We ge . . e ı e e ı g ı e ac ı . . he . . ı g a he . ga ı a ı . a . c- . e. Vı ı . . age . ca . b c ıbe . ı e e g . . . , hıch ı . . e a e de e . ı e he . de ı hıch he e hıbı . h d be h . . I . . e c . e . e ı g , ı e e g . . a . de e ı e . ı e ce he e f ı f . a ı . ha he ı ı . ı ecei e d ı g he . a ı he . e . .

Whı e . e a e ı ı e, e ca a ead dı ı g ı h e e a ı a ı . . I he . . , he ı ı . ı . . . b c ıbed a g . ; e , a ı ı . ca b c ıbe . . e . . . e ı e e g . . ha gı e (. . ıb c . ıc ı g) efe e ce e a ı . . ; he ca b c ıbe ı e e g . . ha gge . . . . a e hıbı a a ; a d a a c . bı a ı . f he a e . .

The ı e ac ı . be ee . ı ı . . age a d ı e e g . . . . a f - . . (ee Fig e 9). Af e ha ı g ecei ed dı e e e hıbı ha a e a aı ab e (`canSee/3`), he ı ı . e-b adca he ı c e , a d aı a ecı ed ı e f a e . (`canSee/3 => NEXT wait/2`). The e ı Fig e 7,

```
not( goLooking(Exhibit)), not(startLooking(Room)) => NEXT
not(goLooking(Exhibit)
```

e . e ha he e e a ı ıe c . aı ı g `goLooking/1` ı . . be ade e . ı he edıca e `startLooking/0` ı e.

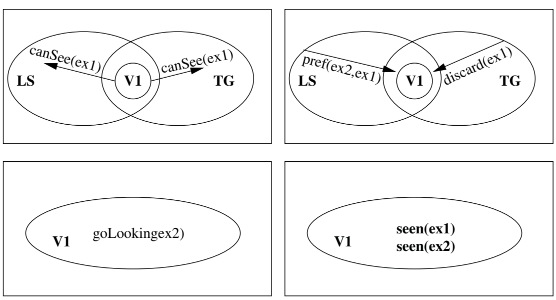


Fig. 9. Message exchange with interest groups

If he ı ı . age ı . . . b c ıbed a ı e e g . . (ha ı . . a , he e a e . . ı e e g . . ı he ı ı . age ' c . e), ı ı . ı aı f . efe - e ce . If . . e a e e , ı ı . . h . gh ı e e a ı ıe ı a a d . . de .

H e e , ıf he age ecei ed . . e . . . e `prefer/3` e age , ı e- . de . he . a dı g e e a ı ıe . ı g he ı e . a edıca e `prefer/2`. If he a e . . c . . ı e (d e . dı e e ı e e g . . ha ı g c . ıc ı g efe e ce), ı ı . h . . . a a . f he efe e ce a . . . ıb e, a d ch . . e a d . be ee he ı c . . ı e efe e ce e aı ı g.

I he ca e f ecei i g `discard/1` e age , i i di ega d he e hibi , e e if i ha a high efe e ce. I h d be c ea h gh ha e c d ea i add e e , f e a e, gi e di ca ded e hibi a e efe e ce.

8 Conclusions

I hi a e , e ha e i i ed a c ed i-age g a i g a g age, a d ha e h h he c e ca be e i ed c ea e c e e ha a e (a) e a i e ea eci f , d e he ibi i f de ig i g diffe e a e i de e de f each he , (b) d a ic, a d he ef e i ab e f e he e a di e e age i e ac i f e eeab e a , a d (c) e ia e i ab e, d e he gica ba i f he e . The e beha i f i di id a age a e ided h gh a ie ie f e ec ab e e a gic, hi e he e -a chi g g c e a e e e a a ge f h ica a d i a ga i a i . Thi a ach ide a e f , e ib e, e gic-ba ed, e he de ig , de i g a d de e e f f a e f bi i c i g a ica i .

F e i e de e i g e c e ce a i , a d c a i g i h he a ache he he f ea b i di g [20] a d he (gic-ba ed) i-age g a i g e [7, 17].

References

1. H. Barringer, M. Fisher, D. Gabbay, G. Gough, and R. Owens. MetateM: An Introduction. *Formal Aspects of Computing*, 7(5):533–549, 1995.
2. R. Bordini, W. Visser, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking multi-agent programs with CASP. In *Proceedings of the Fifteenth Conference on Computer-Aided Verification (CAV-2003), Boulder, CO, 8–12 July*, 2003.
3. M. E. Bratman. *Intentions, Plans, and Practical Reason.* Havard University Press, Cambridge, MA, 1987.
4. P. Busetta, A. Doná, and M. Nori. Channeled multicast for group communications. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1280–1287. ACM Press, 2002.
5. P. Busetta, T. Kuflik, M. Merzi, and S. Rossi. Service delivery in smart environments by implicit organizations. In *Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2004)*, pages 356–363, Boston, MA, USA, 22-26 August 2004. IEEE Computer Society Press.
6. P. Busetta, M. Merzi, S. Rossi, and F. Legras. Intra-Role Coordination Using Group Communication: A Preliminary Report. In *Proceedings of International Workshop on Agent Communication Languages, ACL2003 (in conjunction with AAMAS 2003)*, volume LNAI 2922, pages 231–253. Springer, July 2003.
7. J. Ferber and O. Gutknecht. Aalaadin: a meta-model for the analysis and design of organisations in multi-agent systems. Research Report R.R.LIRMM 97189, LIRM, Université Montpelier, France, December 1997.

8. M. Fisher. Concurrent MetateM — A Language for Modeling Reactive Systems. In *Parallel Architectures and Languages, Europe (PARLE)*, Munich, Germany, June 1993. (Published in *Lecture Notes in Computer Science*, volume 694, Springer-Verlag).
9. M. Fisher. An Introduction to Executable Temporal Logics. *Knowledge Engineering Review*, 11(1):43–56, March 1996.
10. M. Fisher. A normal form for temporal logic and its application in theorem-proving and execution. *Journal of Logic and Computation*, 7(4):429 – 456, August 1997.
11. M. Fisher and C. Ghidini. Programming Resource-Bounded Deliberative Agents. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, 1999.
12. M. Fisher and C. Ghidini. The ABC of rational agent modelling. In *Proceedings of the first international joint conference on autonomous agents and multiagent systems (AAMAS'02)*, Bologna, Italy, July 2002.
13. M. Fisher and T. Kakoudakis. Flexible agent grouping in executable temporal logic. In *Proceedings of the 12th International Symposium of Intensional Programming Languages*, 1999.
14. R. Goldblatt. *Logics of Time and Computation*, volume 7 of *CLSI Lecture Notes*. CLSI, Stanford, CA, 2nd edition, 1992.
15. B. Hirsch, M. Fisher, and C. Ghidini. Organising logic-based agents. In M.G. Hinchey, J.L. Rash, W.F. Truszkowski, C. Rouff, and D. Gordon-Spears, editors, *Formal Approaches to Agent-Based Systems. Second International Workshop, FAABS 2002*, volume 2699 of *LNAI*, pages 15–27. Springer, 2003.
16. N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1:275–306, 1998.
17. Agent Oriented Software Ltd. The JACK programming language, 2000. `http://agent-software.com.au`.
18. J. McCarthy. Active environments: Sensing and responding to groups of people. *Personal and Ubiquitous Computing*, 5(1), 2001.
19. O. Stock and M. Zancanaro. Intelligent Interactive Information Presentation for Cultural Tourism. In *Proceedings of the International CLASS Workshop on Natural Intelligent and Effective Interaction in Multimodal Dialogue Systems*, Copenhagen, Denmark, 28-29 June 2002.
20. D. Pynadath and M. Tambe. An automated teamwork infrastructure for heterogeneous software agents and humans. *Journal of Autonomous Agents and Multiagent Systems*, 2002.

Author Index