FAMAS’06

Formal Approaches to Multi-Agent Systems

In recent years, multi-agent systems have come to form one of the key technologies for software development. The Formal Approaches to Multi-Agent Systems (FAMAS) workshop series brings together researchers from the fields of logic, theoretical computer science and multi-agent systems in order to discuss formal techniques for specifying and verifying multi-agent systems. FAMAS addresses the issues of logics for multi-agent systems, formal methods for verification, e.g. model checking, and formal approaches to cooperation, multi-agent planning, communication, coordination, negotiation, games, and reasoning under uncertainty in a distributed environment.

The first FAMAS workshop, FAMAS’03, was a successful satellite event of the European Conference on Theory and Practice of Software (ETAPS’03) in Warsaw. It took place on April 12th 2003, and afterwards a selection of contributed and invited papers was published in Fundamenta Informaticae as volume 63, issue 2,3 of 2004.

The present volume contains seven papers presented at the second FAMAS workshop FAMAS’06, taking place on Monday 28, 2006 in conjunction with the European Conference on Artificial Intelligence (ECAI’06). Again, a selection of FAMAS speakers will be invited to contribute an extended version of their work to a special issue of a well-known international journal.

All research reported here is squarely related to practice, even if the formal approach is taken. Thus, just as in FAMAS’03, authors of FAMAS’06 contributions devote their attention to pressing practical problems such as plan diagnosis, security protocols, and negotiation. In comparison to FAMAS’03, however, there is much more emphasis now on the fact that multi-agent systems are situated in a dynamic environment. Also, quite a few authors take on the challenge to combine different logics (for example, for time and belief) in a methodologically sound manner.

Verification and dynamical aspects of multi-agent systems

In Hindriks and Meyer’s “An agent program logic with declarative goals”, the emphasis is on an agent programming theory consisting of both an agent programming language and a corresponding program logic to verify agent programs. The logic, “dynamic agent logic” is inspired by dynamic epistemic logic, incorporating declarative goals. For the dynamic agent programming language, both an operational and a denotational semantics are provided, and they are proved to be equivalent. Finally, the paper gives an interesting first step into formalizing the process of goal adoption, based on second-order goals.

Roos and Witteveen, in their paper “Models and methods for plan diagnosis”, present a model-based diagnosis approach to plan diagnosis, where they restrict themselves to the case in which failure of actions causes the failure of
the plan as a whole, arguing that if a plan is correctly specified, then errors in
the plan execution process become manifest in the incorrect performance of one
or more actions. They present a formal framework for diagnosis, and show that
their “mini-maxi diagnoses” can be computed efficiently. In future work, the
authors will include more multi-agent elements such as models of the executing
agents.

Communication and security in a multi-agent setting

Orgun, Governatori and Liu make a nice bridge from theory to practice in their
contribution “Modal tableaux for verifying security protocols”. In order to do
this, they first construct a combined logic that adds a temporal dimension to
the logic of beliefs by using the fibring method to combine logics, as introduced
by Gabbay and colleagues. A sound and complete logic is provided, and, as
the cherry on the cake, the system is applied to the authors’ formalization of
the TESLA authentication protocol, that also served as a successful jumping-off
point for Wozna and Lomuscio’s “A complete and decidable security-specialised
logic and its application to the TESLA protocol”, published in the proceedings
of AAMAS’06.

Similarly, Teepe combines a protocol and a security-inspired logic in his
“BAN logic is not ‘sound’: constructing epistemic logics for security is diffi-
cult”. He gives a short introduction to cryptographic hash functions and then
shows, both proof-theoretically and semantically, that the well-known BAN logic
proves a very undesirable conclusion about a simple communication protocol.
The result does not invalidate logical approaches to protocol analysis, but does
provide a welcome warning not to include too strong rules in order to make the
logic complete.

In “More games in belief game model: a preliminary report”, Konieczny
takes a closer look at a variant of negotiation in which a group of agents comes
to a consensus in their beliefs by weakening some of their points of view. His
work is based on cooperative game theory, viewing negotiation as a game of
incomplete information, in which one needs to measure the ‘amount of inconsis-
tency’ brought into the game by each participant.

Modelling mental states

Perreira, Oliveira, and Moreira’s paper “Modelling emotional BDI agents” com-
bines aspects of KARO, such as resources and capabilities, with BDI-logics to
model the emotion of fear. The main emphasis is on time and dynamics: how can
future threats be distinguished into different classes, and how do these impact
the agents’ reactions? Logically, when analyzing satisfiability of a formula, the
question can be reduced to satisfiability of a translated formula in branching-
time BDI, so that it is easy to show decidability of the system. Multi-agent
aspects remain to be treated in future work.

Finally, in “A hybrid representation of knowledge and belief”, Ricardo We-
hbe presents a default-logic based framework for agents working in environments
with incomplete information. A hybrid representation is used, including a mono-
tonic part (the clauses, corresponding to knowledge) as well as a non-monotonic
part, the defaults. Wehbe also presents a constructive manner to obtain invari-
ants and extensions of the default set. In future, the framework will be extended
to include agents’ reasoning about other agents’ knowledge.

Acknowledgments

We would like to thank all the people who helped to bring about FAMAS’06. First of all, we thank all speakers for ensuring a diverse and interesting workshop. Special thanks are due to the members of the program committee for their professionalism and their dedication to select papers of quality and to provide authors with useful, constructive feedback during the in-depth reviewing process.

These workshop proceedings and website would not exist without the help of Marcin Dziubiński and Mateusz Srebrny in Warsaw.

Barbara Dunin-Keplicz
Institute of Informatics, Warsaw University
Institute of Computer Science, Polish Academy of Sciences
keplicz@mimuw.edu.pl
http://www.mimuw.edu.pl/~keplicz

Rineke Verbrugge
Institute of Artificial Intelligence, University of Groningen
rineke@ai.rug.nl
http://www.ai.rug.nl/~rineke

August 2006
Organization

Program Chairs

Barbara Dunin-Kęplicz Warsaw University and Polish Academy of Sciences, Poland
Rineke Verbrugge University of Groningen, The Netherlands

Program Committee

Hans van Ditmarsch University of Otago, New Zealand
Amal El Fallah-Seghrouchni University of Paris 6, France
Andreas Herzig Universite Paul Sabatier, France
Wiebe van der Hoek University of Liverpool, United Kingdom
Wojtek Jamroga Clausthal University of Technology, Germany
Michael Luck University of Southampton, United Kingdom
John-Jules Meyer Utrecht University, The Netherlands
Chris Reed University of Dundee, United Kingdom
David Robertson University of Edinburgh, United Kingdom
Jeff Rosenschein The Hebrew University of Jerusalem, Israel
Onn Shehory IBM - Haifa Research Labs, Israel
Carles Sierra IIIA, Spanish Research Council, Spain
# Table of Contents

- An Agent Program Logic with Declarative Goals *K.V.Hindriks, J.-J.Ch.Meyer* 1

- More Games in Belief Game Model: A Preliminary Report *S.Konieczny* 16

- Modal tableaux for verifying security protocols *M.A.Orgun, G.Governatori, Ch.Liu* ................................................................. 30

- Modelling Emotional BDI Agents *D.Pereira, E.Oliveira, N.Moreira* ........ 46

- Models and Methods for Plan Diagnosis *N.Roos, C.Witteveen* ............. 61

- BAN Logic is Not ‘Sound’, Constructing Epistemic Logics for Security is Difficult *W.Teepe* .......................................................... 77

- A Hybrid Representation of Knowledge and Belief *R.Wehbe* ............... 90
An Agent Program Logic with Declarative Goals

K.V. Hindriks¹ and J.-J. Ch. Meyer²

¹ Nijmegen Institute for Cognition and Information, Radboud University Nijmegen, The Netherlands, k.hindriks@nici.ru.nl
² Computing and Information Science Department, Utrecht University, The Netherlands, jj@cs.uu.nl

Abstract. It has been argued that declarative goals provide for a natural conceptual tool for designing as well as programming agents. This has given rise to various proposals for integrating declarative goals into programming languages. It is not always clear, however, how to establish a precise relation to logical agent theories to reason about such agents. In this paper, we propose an agent programming theory that provides both an agent programming language as well as a corresponding agent program logic to verify agent programs. The agent programming language and agent program logic are developed in parallel to ensure the existence of a mathematically precise relation between the program and logical semantics. To this end, a modal agent logic including the core agent concepts of action, knowledge and goals is introduced. Consecutively, an equivalent state-based semantics is introduced that can be used to relate the logical and operational program semantics. Additionally, it is shown how to integrate goal adoption into the programming theory using the concept of second-order goals.

1 Introduction

In this paper, an agent programming theory is introduced for rational agents. A programming theory in our sense consists of two components: a programming language and an associated program logic that can be used to axiomatically verify programs written in the programming language. Intuitively, rational agents derive their choice of action from their knowledge and goals and these core agent concepts need to be incorporated into an agent programming theory. We show how these concepts can be operationalized in an agent programming language and how the operational semantics can be formally related to the semantics of an associated agent logic. In our approach to agent programming, knowledge and goals are explicit computational resources. The associated operational semantics defines how agents compute with these resources. The associated logical semantics provides for a declarative formalism to specify and reason about agents.

The paper is organized as follows. As a first step, the core of our programming theory is introduced. This core includes a program logic called DEL based on a modal dynamic logic extended with an epistemic operator. A state-based semantics equivalent with the modal semantics is constructed that can be used
to define an operational semantics for the associated programming language. Second, the core logic is extended with a motivational operator. We construct an equivalent state-based semantics that extends the state-based semantics for DEL in a modular way and incorporates the state-based semantics for DEL without modifications. Some constraints are imposed on the modal semantics to achieve this goal of modular design which are discussed and motivated. Third, the programming semantics is extended to incorporate goal adoption, and we present a mechanism to adopt goals using second-order goals.

2 Dynamic Epistemic Logic
As the core of our agent programming theory, we briefly introduce dynamic epistemic logic (DEL) and discuss a corresponding state-based semantics. The logic DEL is a combination of standard propositional dynamic logic and an S5 operator to model epistemic attitudes of an agent (cf. [1, 2]). The language of the logic is defined relative to an infinite set \( \mathcal{A} \) of propositional atoms \( p_1, p_2, \ldots \) and a finite set \( \mathcal{A} \) of atomic actions \( a_1, \ldots, a_n \). The reasons for insisting on an infinite set \( \mathcal{A} \) and a finite set \( \mathcal{A} \) will become clear below.

**Definition 1.** (Syntax of Dynamic Epistemic Logic)
The syntax of propositional dynamic epistemic formulas \( \varphi \in \mathcal{L}_{DEL} \) and programs \( \pi \in \Pi_{DEL} \) is given by the following inductive BNF definitions:

\[
\varphi ::= \text{any element of } \mathcal{A} \mid \neg \varphi \mid \varphi \land \varphi \mid [\pi] \varphi \mid K \varphi
\]

\[
\pi ::= \text{any element of } \mathcal{A} \mid \varphi? \mid \pi; \pi \mid \pi + \pi \mid \pi^*
\]

The set of classical propositional formulae \( \varphi \in \mathcal{L}_{DEL} \) without any occurrences of modal operators \( [\pi] \) or \( K \) is denoted by \( \mathcal{L}_0 \) and \( \varphi \in \mathcal{L}_0 \) are called objective propositions. The set of propositions \( \varphi \in \mathcal{L}_{DEL} \) without any occurrences of operators \( [\pi] \) is denoted by \( \mathcal{L}_k \) and \( \varphi \in \mathcal{L}_k \) are called knowledge propositions. The operator \( \Box \) expressing that \( \varphi \) is true at the next point in time can be defined using the fact that DEL is built from a finite set of actions. \( \Box \varphi \) is defined as \([a_1 + \ldots + a_n] \varphi\). This definition illustrates the assumption built into dynamic logic that these actions are the only way to change the world. Similarly, a temporal operator \( \square \varphi \) expressing that \( \varphi \) is always true can be defined as \([a_1 + \ldots + a_n]^\ast \varphi\).

The semantics for the dynamic epistemic language \( \mathcal{L}_{DEL} \) is defined as usual in terms of frames and models.

**Definition 2.** (DEL Frames and Models)
A DEL frame \( F \) is a triple \( \langle W, R, K \rangle \) with

- \( W \) a non-empty set of worlds typically denoted by \( w, w_1, w_2, \ldots, \)
- \( R : \mathcal{A} \rightarrow W \times W \) a function mapping each atomic action onto an accessibility relation on \( W \), and
- \( K \subseteq W \times W \) an equivalence relation modeling the knowledge of an agent.

A DEL model \( M = \langle W, R, K, V \rangle \) is an extension of a DEL frame \( \langle W, R, K \rangle \) with an assignment function \( V : \mathcal{A} \times W \rightarrow \{1, 0\} \) mapping propositional atoms and worlds onto the truth values \( \{1, 0\} \).
Definition 3. (Semantics of Dynamic Epistemic Logic)
Let \( M = (W, R, K, V) \) be a DEL model. Then the semantics of \( \varphi \in \mathcal{L}_{\text{DEL}} \) and the extension of the relation \( R \) to arbitrary DEL programs \( \pi \in \Pi_{\text{DEL}} \) is defined by simultaneous induction as usual:

- The truth conditions for \( \varphi \in \mathcal{L}_{\text{DEL}} \) are defined by:
  - \( M, w \models p \) iff \( V(p, w) = 1 \),
  - \( M, w \models \neg \varphi \) iff \( M, w \not\models \varphi \),
  - \( M, w \models \varphi \land \psi \) iff \( M, w \models \varphi \) and \( M, w \models \psi \),
  - \( M, w \models [\pi] \varphi \) iff \( \forall w' (w R_{\pi} w' \Rightarrow M, w' \models \varphi) \),
  - \( M, w \models K \varphi \) iff \( \forall w' (w K w' \Rightarrow M, w' \models \varphi) \).

- The semantics of \( \pi \in \Pi_{\text{DEL}} \) is defined by:
  - \( R_{\varphi} = \{ (w, w') \mid M, w \models \varphi \} \),
  - \( R_{\pi_1 \circ \pi_2} = R_{\pi_1} \circ R_{\pi_2} = \{ (w, w') \mid \exists w'' (w R_{\pi_1} w'' \text{ and } w'' R_{\pi_2} w') \} \),
  - \( R_{\pi_1 + \pi_2} = R_{\pi_1} \cup R_{\pi_2} \),
  - \( R_{\pi^*} = (R_{\pi})^* = \{ (w, w') \mid \exists n \in \mathbb{N} (w R_{\pi}^n w') \} \).

Note that the modal program semantics in definition 3 implies that performing an action or executing a program in a world may result in changes to that world as well as to the knowledge of the agent. That is, both objective propositions as well as knowledge propositions may have changed truth value as a consequence of performing an action.

Also note that since a standard S5 modal logic is used to model knowledge of an agent, multiple occurrences of the epistemic operator \( K \) in a knowledge proposition \( \varphi \) can be eliminated to obtain a logically equivalent knowledge proposition \( \psi \) with at most a single operator \( K \) (cf. [2]). This property of knowledge propositions also holds for formulae in the weaker system \( KD45 \) modeling belief instead of knowledge where the T-scheme has been replaced by the D-scheme.

We now proceed with the definition of a state-based semantics that is equivalent to the modal semantics, using these observations. States have to incorporate the “local” information available in any world. Each state thus defined will contain enough information to evaluate both objective as well as knowledge propositions. Formally, a DEL state, typically denoted by \( s, s_1, s_2, \ldots \) is defined as a pair \( (v, k) \) with \( v \subseteq At \) and \( k \subseteq \mathcal{L}_0 \), such that \( k \) is consistent and \( v \models k \) to ensure that knowledge is veridical. The first component \( v \) is called a world state and the second component \( k \) a knowledge base. It is clear that with any world \( w \) in a modal DEL model \( M \) a unique DEL state \( s = (v, k) \) can be associated, with \( v = \{ p \in At \mid M, w \models p \} \) and \( k = \{ \varphi \in \mathcal{L}_0 \mid M, w \models K \varphi \} \). Vice versa, things are slightly more complicated because there may be more than one world in a DEL model that corresponds with a particular DEL state \( s \) (if \( w, w' \) both correspond to a state \( s \), \( w \) is called a copy of world \( w' \); note that the worlds \( w, w' \) may still differ with respect to the actions that can be executed).

Observe that a state \( s \) contains the information that is required to evaluate any objective or knowledge proposition \( \varphi \). This is obvious for objective propositions. Since knowledge propositions can be assumed to consist of at most a single epistemic operator by the property mentioned above, a knowledge base \( k \) that consists of objective formulae can be used to evaluate propositions of the form \( K \varphi \) as well as \( \neg K \varphi \) with \( \varphi \in \mathcal{L}_0 \); check whether \( k \models \varphi \) or \( k \not\models \varphi \).
Definition 4. (State-based Structure for Dynamic Epistemic Logic)
A state-based structure $M^c$ for DEL is a pair $(W^c, R^c)$ with $W^c$ a non-empty set of DEL states and $R^c : Act \rightarrow W^c \times W^c$ a function mapping each action onto a binary relation on states.

The state-based semantics for $L_{DEL}$ can be defined in terms of state-based structures. We use a subscript $c$ in $|=c$ to distinguish it from the standard modal entailment relation $|=$. For any component $c$ in a state $s$, we write $s[c'/c]$ to denote that $c$ is replaced with $c'$, e.g., $s[c'/v]$ is $\langle v', k \rangle$ for $s = \langle v, k \rangle$.

Definition 5. (State-based Semantics for Dynamic Epistemic Logic)
Let $M^c = \langle W^c, R^c \rangle$ be a state-based DEL structure and $s = \langle v, k \rangle \in W^c$ be a state. Then the semantics of $\varphi \in L_{DEL}$ and the extension of $R^c$ to arbitrary DEL programs $\pi \in \Pi_{DEL}$ is defined by simultaneous induction as follows:

- The truth conditions for $\varphi \in L_{DEL}$ are defined by:
  - $M^c, s \models p$ iff $p \in v$.
  - $M^c, s \models \neg \varphi$ iff $M^c, s \not\models \varphi$.
  - $M^c, s \models \varphi \land \psi$ iff $M^c, s \models \varphi$ and $M^c, s \models \psi$.
  - $M^c, s \models [\pi] \varphi$ iff $\forall s'(s R^c_k s' \Rightarrow M^c, s' \models \varphi)$.
  - $M^c, s \models K_{\pi} \varphi$ iff $\forall s'(s' \models k \Rightarrow M, s[c'/v] \models \varphi)$.

- The semantics of $\pi \in \Pi_{DEL}$ is defined by:
  - $R_{\pi^?} = \{ (s, s) | M^c, s \models \varphi \}$.
  - $R_{\pi_1 \circ \pi_2} = R_{\pi_1} \circ R_{\pi_2}$.
  - $R_{\pi_1 + \pi_2} = R_{\pi_1} \cup R_{\pi_2}$.
  - $R_{\pi^*} = (R_{\pi})^*$.

Theorem 1. (Equivalence of Modal and State-Based Semantics)
The modal and the state-based semantics are equivalent:
for any $\varphi \in L_{DEL}$: $\models \varphi$ iff $|= \varphi$

Proof. The proof of the equivalence of the modal and state-based semantics essentially uses the fact that $At$ is infinite. We give a sketch of the proof, the full proof is available in the full paper. The right to left implication is proved by a straightforward mapping from state-based structures to standard models. For the left to right implication, use the finite model property for the Kripke semantics to show the equivalence with the state-based semantics. We need to prove that if $M^c, s \not\models \varphi$, then also $M, w \not\models \varphi$. Since the truth of $\varphi$ can only depend on a finite number of propositional atoms, an infinite number of atoms remains that can be used as names for possible worlds in the standard model to keep track of these worlds in a state-based structure. Using this observation, then setup a correspondence between possible worlds and states and show their equivalence. □

Remark 1. The results presented in this section may remind the reader of knowledge structures studied in [3]. Knowledge structures provide a model-theoretic alternative for Kripke semantics. Our objectives, however, are different: We are interested in designing a programming language for agents that compute with databases (syntactic sets of sentences) and not in the structure of knowledge states. The “flat” syntactic approach to knowledge bases suits our purposes, whereas it “begs the question of what a model of a state of knowledge is” [3].
The state-based semantics can be used to define an operational semantics for DEL programs. To avoid the need for undecidable look-ahead facilities, tests \( \varphi \) in DEL programs need to be restricted to knowledge propositions. It can be shown that the modal DEL logic can be used to prove partial correctness properties of DEL programs, and, consequently, that DEL can be used to axiomatically verify the correctness of DEL programs. The details will be postponed and presented later in section 3.2 for an extension of DEL with a motivational operator.

3 Dynamic Agent Logic

In the previous section, the preliminaries of a programming theory for agents which use knowledge to guide their choice of action has been presented. Rational agents, however, base their choice of action on their knowledge as well as their goals. We now proceed with incorporating declarative goals into the DEL programming theory. The extension of the modal logic with a goal operator follows standard practice in the literature (cf. [4, 5]) and adds a \( KD \) operator to the language. The associated program logic is called dynamic agent logic (DAL).

The strategy for incorporation of declarative goals into the programming theory is, first, to show how to incorporate the notion of (static) goals, and, second, to show how to incorporate goal dynamics into the semantics of agent programs. The first objective is explored in this section and is accomplished by adding a modal goal operator to the DEL language and by introducing an equivalent state-based semantics. The second objective is explored in section 4.

Dynamic agent logic is an extension of DEL with a modal \( KD \) operator \( G \varphi \), where \( \varphi \) is informally interpreted as a(n achievement) goal of the agent.

**Definition 6.** (Syntax of Dynamic Agent Logic)

The syntax of propositional dynamic agent formulas \( \varphi \in L_{DAL} \) is defined as that for DEL except for one additional clause in the definition of formulae:

\[
\varphi ::= \ldots \mid G\varphi
\]

Compared with a DEL frame a DAL frame additionally includes a binary relation \( G \) to model the goals of an agent. A DAL frame is a tuple \((W, R, K, G)\) such that \((W, R, K)\) is a DEL frame and \( G \subseteq W \times W \) is a serial relation on \( W \). ADAL model \((W, R, K, G, V)\) extends a DAL frame with an assignment function \( V \). One additional semantic clause is needed to specify the truth conditions for formulae in \( L_{DAL} \) of the form \( G\varphi \).

**Definition 7.** (Semantics of Dynamic Agent Logic)

Let \( M = (W, R, K, G, V) \) be a DAL model and \( w \in W \). The truth conditions for propositions \( \varphi \in L_{DAL} \) are those of \( L_{DEL} \) extended with one additional clause:

\[
\bullet M, w \models G\varphi \quad \text{iff} \quad \forall w' (wGu' \Rightarrow M, w' \models \varphi).
\]

3.1 A State-Based Semantics for DAL

One of our additional objectives is to extend the DEL programming theory in a modular way. Several considerations motivate the choice for a modular ap-
approach. One reason for a modular approach is that it allows for a relatively independent design (and consecutive verification) of one program component from other components which eases the development process of agent programs. More in particular, the DEL programming theory is elegant and simple, and, ideally, this component (or module) of the programming theory would not have to be modified. A modular approach can be implemented by using several advanced techniques for combining logics that are available. However, for reasons of simplicity, here a pragmatic approach is taken and it is shown that by relatively simple means a modular approach can be supported as well. The investigation of the use of more sophisticated techniques based on, for example, fibering, we think, would be interesting in relation to the approach presented here (cf. [6]).

The more pragmatic approach used here is discussed first informally by considering the action and knowledge components in the DEL programming theory. The knowledge base defined as a set of objective propositions is elegant and there do not seem to be good reasons to extend such knowledge bases and to complicate their definition by the incorporation of goals. Such an extension would allow for the (partial) introspection of goals, but we prefer to introduce this as a standard capability of an agent (cf. also e.g. [7]). This introspective capability can be axiomatized in the agent logic by the (positive) introspection axiom \( G\varphi \rightarrow KG\varphi \).

Additionally, the actions in the DEL programming theory only change the world and knowledge of an agent. In the extended agent logic, a choice can be made to allow arbitrary actions to modify the goals of an agent as well. This would, of course, require modifications to the action component. However, it seems to us more natural to think of agent capabilities as only changing the world and the knowledge of an agent representing these changes but not the agent’s goals. Separate mechanisms then will (need to) be introduced to support dynamic changes to the agent’s goals (cf. section 4).

Formally, the discussion in the previous paragraph can be implemented by imposing additional restrictions on the semantics of the agent logic. The (positive and negative) introspective capabilities with respect to goals and the fact that goals are not changed by performing actions impose the following restrictions.

**Constraint 1.** (Introspection of Goals and Goal Persistence)

Let \( \langle W, R, K, G \rangle \) be a DAL frame and let \( G(w) \) be a shorthand notation for the set \( \{ w' \in W | wGw' \} \). The following restrictions are imposed on DAL frames:

- \( \forall w, w', w'' ((wKw' \text{ and } w'Gw'') \Rightarrow wGw'') \),
- \( \forall w, w', w'' ((wGw'' \text{ and } w'Kw') \Rightarrow w'Gw'') \),
- \( \forall w, w' (G(w) = G(w')) \).

It is easy to verify that the first two constraints imply respectively positive and negative introspection of goals and validate the axiom schemas \( G\varphi \rightarrow KG\varphi \) and \( \neg G\varphi \rightarrow K\neg G\varphi \). Because of the validity of \( K\varphi \rightarrow \varphi \) it follows that \( K(\neg)G\varphi \leftrightarrow (\neg)G\varphi \) is valid. It is this fact that allows us to continue to use the definition of a knowledge base as a set of objective propositions without modifications (in this context, it is important to remark that if the epistemic operator \( K \) would be replaced by a doxastic operator \( B \), we have to require that at least the beliefs
of an agent about its own goals are veridical). The third constraint implies that goals do not change and validates the axiom schema $G\varphi \rightarrow \Box G\varphi$ which express that goals (always) persist. Such persistence, of course, is not what we finally aim for, but will do for now to incorporate (static) goals into the semantics. These constraints are instrumental in defining a state-based semantics that extends the DEL state-based semantics in a modular way.

The additional component that needs to be accounted for in a state-based semantics for the agent logic is, of course, the goal base of an agent, to which we now turn. Recall that a DEL state is a pair $⟨v, k⟩$ with $v$ a world state and $k$ a knowledge base. We want to extend these states with a goal base $g$ that can be used to provide the semantics of formulas of the form $G\varphi$. Goal bases that correspond with the properties of the $G$ operator obviously cannot be defined similarly as the “flat” knowledge bases due to the different properties of the $G$ and $K$ operator. First, we cannot “erase” multiple occurrences of $G$ in formulas like $GG\varphi$; second-order goals differ logically in meaning from first-order goals. This difference in meaning, moreover, can usefully be exploited as a mechanism to adopt goals. Second, we want to allow for goals to obtain information, i.e. query goals of the form $GK\varphi$.

The main property of goals that we need to account for thus is the existence of higher-order goals different from simple first-order goals, i.e. the existence of a hierarchy of goals at various levels. This is not simply a technical issue. Interestingly, Sloman et al. in [8] have pointed to the role of such higher-order goals as motive generators, an idea we will follow up on in section 4. An agent that has a first-order goal $\varphi$ (expressed by $G\varphi$ with $\varphi \in \mathcal{L}_k$ in DAL) signifies that the agent wants to change the world to satisfy $\varphi$. A second-order goal of the same agent (expressed by $GG\varphi$ with $\varphi \in \mathcal{L}_k$ in DAL), however, signifies that the agent in some sense wants to change its current set of (first-order) goals. The importance of the distinction between first-order and higher-order goals is also illustrated by the fact that it has been used to provide an account of autonomy and to argue that it differentiates humans from animals and artificial machines (cf. [9]; see also [10], who state that it is the self-generation of goals that makes an agent autonomous).

A proposal for a goal base structure is to define it is a linear hierarchy of goals, where goals $\varphi$ at each level are knowledge propositions (i.e. $\varphi \in \mathcal{L}_k$). Unfortunately, this does not quite work due to the branching structure of the modal semantics for goals (we may have $G(G\varphi \lor G\psi)$ without either $GG\varphi$ or $GG\psi$ being the case in the modal semantics). To remain faithful to the modal semantics we thus would have to introduce a branching structure in goal bases as well. It seems to us, however, that there is not much cognitive plausibility in agents having branching goal bases and even less use for such structures from an agent programming perspective; it only further complicates the already quite sophisticated reasoning required for executing agent programs. Therefore, to exclude such counterintuitive structures and to avoid unnecessary complications in the agent programming theory, we introduce the following constraint.
Constraint 2. (Definite Stance towards Goal Adoption)
Let \((W, R, K, G)\) be a DAL frame. The following restriction is imposed:

\[- \forall w_1, w_2, w_3, w_4 ((w_1 G w_2 \land w_1 G w_3 \land w_2 G w_4) \Rightarrow w_3 G w_4).\]

The constraint defines a confluence property of the \(G\)-relation. It constrains the branching structure of the \(G\)-relation and can be viewed as a “linearization” of that structure. The class of frames \(F\) that satisfy constraint 2 validate the schema \(G^i \varphi \lor G^{i+1} \varphi\) for \(i \geq 0\), with \(G^0 \varphi = \varphi\) and \(G^{i+1} \varphi = G (G^i \varphi)\).

Besides the technical advantage that the validity of \(G^i \varphi \lor G^{i+1} \varphi\) for \(i \geq 0\) yields in order to keep our goal structure simple, it also expresses a clear attitude of the agent regarding its higher-order goals. The second-order version of the axiom schema \(GG \varphi \lor G \neg G \varphi\) can be used to illustrate this attitude. It expresses that the agent has a definite stance towards goal adoption. The agent is either inclined to adopt the goal or it is inclined not to adopt it (which is not the same as being opposed to the realization of \(\varphi\); we do not have \(GG \varphi \lor G \neg \varphi\)!) Note that disjunctive first-order goals \(G (\varphi \lor \psi)\) are also not excluded. An agent thus cannot be “indifferent” to having a goal \(\varphi\), in the sense that it does not care whether to have \(\varphi\) as a goal or not.

Summarizing, goals are organized in a linear hierarchy, or as we will also say goals are stratified. A goal base can be defined as a linear structure where goals at each level consist of knowledge propositions.

Definition 8. (Stratified Goal Base)
A stratified goal base \(g\) is a function mapping the natural numbers to consistent sets of knowledge propositions closed under logical consequence. I.e., for all natural numbers \(i\), \(g(i) \subseteq L_k\). The set \(g(0)\) consist of the first-order goals of the agent, \(g(1)\) consist of second-order goals, etc.

It is useful to introduce a “single step” operation on goal bases that removes the first-order goals from the hierarchy. This operation is the state-based variant of taking a single step in the tree of possible worlds induced by \(G\) in a frame. For a goal base \(g\), we write \(g^{i+1}\) for the goal base \(g^{i+1}(i) = g(i + 1)\). State-based DAL structures are defined as DEL structures before, with DEL states \(\langle v, k, g \rangle\) replaced by DAL states \(\langle v, k, g \rangle\) which include an additional goal base \(g\).

Definition 9. (State-Based Structure for Dynamic Agent Logic)
A state-based DAL structure \(M^c\) for DAL is a pair \((W^c, R^c)\) with \(W^c\) a non-empty set of DAL states of the form \(\langle v, k, g \rangle\) and \(R^c : Act \rightarrow W^c \times W^c\) a function mapping each action onto a binary relation on states.

The state-based truth definition for DAL requires us to add one additional clause to the semantic clauses for DEL.

Definition 10. (State-Based Semantics for Dynamic Agent Logic)
Let \(M^c = (W^c, R^c)\) be a state-based DAL structure, and \(s = \langle v, k, g \rangle\) \(\in W^c\) be a state. Then the semantics of \(\varphi \in L_{DAL}\) and programs \(\pi \in P_{DAL}\) is defined by the same semantic clauses as for DEL (cf. def. 5) with DEL states replaced by DAL states, and the following additional clause:

- \(M^c, s \models \varphi \iff \forall v', k' (M^c, v', k', g \models g(0) \Rightarrow M^c, s[v' / v, k' / k, g^{i+1} / g] \models \varphi).\)
The modal semantics and the state-based semantics are equivalent and the expressive power thus is not reduced by introducing a state-based semantics.

**Theorem 2.** (Equivalence of $|$ and $|=_{c}$ for DAL)

The modal and the state-based semantics for DAL are equivalent:

for any $\varphi \in \mathcal{L}_{DAL}$: $\models \varphi$ iff $|=_{c} \varphi$

*Proof.* The proof is analogous to theorem 1, but also makes use of constraints 1 and 2. $\square$

### 3.2 Operational Semantics for DAL Programs

In this section, the details of an operational semantics for DAL programs are provided using the state-based semantics, which, with minor changes, can also be used to provide an operational semantics for DEL programs.

The main task in defining an operational semantics for DAL programs is to define a computational step relation, or, as it is usually called, a transition relation, by means of a set of transition rules (cf. [11]). The transition rules for the compositional program constructs (i.e. sequential composition $;$, nondeterministic choice $+$ and repetition $^{*}$) are common, but the rules for the basic operations in the programming language, execution of actions $a_{i}$ and tests $\varphi?$, must be provided with a computational interpretation in terms of the states of the programming language semantics.

**Computational Interpretation of Actions.** Actions change the world and the agent’s knowledge representing that world, but, as discussed, do not change the agent’s goals. Additionally, DAL states $\langle v, k, g \rangle$ satisfy $v \models k$ to ensure that knowledge correctly represents the current world state, and action execution must preserve this relation between the world state and knowledge base. Formally, these requirements are captured in the definition of a transition function.

**Definition 11.** (DAL Transition Function)

A DAL transition function $\mathcal{T}$ is a mapping from actions $a_{i}$ and DAL states $\langle v, k, g \rangle$ to sets of DAL states that satisfies the following condition:

- Any $\langle v', k', g' \rangle \in \mathcal{T}(a_{i}, v, k, g)$ satisfies $v' \models k'$ and $g' = g$.

**Computational Interpretation of Tests.** In an operational semantics, tests are evaluated locally in the states of a computation. As theorem 2 shows, DAL states $\langle v, k, g \rangle$ contain the information needed to evaluate formulas without dynamic modalities $[\pi]$. Tests in DAL programs therefore are restricted to formulas without occurrences of dynamic operators, to be able to provide an operational semantics and to avoid the need to introduce undecidable look ahead facilities.

DAL formulas $\varphi \in \mathcal{L}_{DAL}$ without occurrences of dynamic modalities $[\pi]$ are called intentional propositions. The set of intentional propositions is denoted by $\mathcal{L}_{g}$ and includes objective and knowledge propositions, i.e. $\mathcal{L}_{k} \subseteq \mathcal{L}_{g}$. DAL programs with restricted tests $\varphi?$ with $\varphi \in \mathcal{L}_{g}$ are called poor test DAL programs (cf. [1]). Poor test DAL programs thus include tests on the knowledge and goal bases of an agent. A computational interpretation of poor tests that is used to define the operational semantics of tests in terms of states is introduced next.
Definition 12. (Computational Interpretation of Intentional Propositions)
Let \( v \) be a world state, \( k \) a knowledge base, and \( g \) a goal base such that \( v \models k \). Then the truth conditions for intentional propositions \( \varphi \) are defined by:

- \( \models_{v,k,g} p \iff p \in v \),
- \( \models_{v,k,g} \neg \varphi \iff \not \models_{v,k,g} \varphi \),
- \( \models_{v,k,g} \varphi \land \psi \iff \models_{v,k,g} \varphi \) and \( \models_{v,k,g} \psi \),
- \( \models_{v,k,g} K \varphi \iff \forall v' (v' \models_k \Rightarrow \models_{v',k,g} \varphi) \),
- \( \models_{v,k,g} G \varphi \iff \forall v', k' (\models_{v',k',g} g(0) \Rightarrow \models_{v',k',g+1} \varphi) \).

Note that the computational interpretation \( \models_{v,k,g} \) can be used on the right hand side in the last clause of the inductive definition since \( g(0) \subseteq L_k \) does not itself contain goal operators. This clause also shows that first-order goals are interpreted by the set of level 0 goals \( g(0) \) and higher-order goals are interpreted by higher levels in the hierarchy. A formula \( G \varphi \) with \( \varphi \in L_k \) is evaluated in all models of \( g(0) \subseteq L_k \), i.e. all world state and knowledge base pairs that are models of \( g(0) \).

**Transition Semantics for DAL Programs** The transition semantics is defined inductively by a set of transition rules using the concepts introduced above.

Definition 13. (Transition Semantics for DAL Programs)
Let \( s, s' \) be DAL states, \( T \) a transition function, and \( \varphi \) an intentional proposition. Then the transition semantics for DAL programs is inductively defined by:

\[
\begin{align*}
\langle a, s \rangle \rightarrow \langle E, s' \rangle & \quad |_{=_{a, \varphi}} \quad \langle \pi_1 ; \pi_2, s \rangle \rightarrow \langle \pi_1', \pi_2, s' \rangle \\
\langle a, s \rangle \rightarrow \langle \varphi', s \rangle & \quad \langle \pi_1, s \rangle \rightarrow \langle \pi_1', s' \rangle \\
\langle a, s \rangle \rightarrow \langle \pi_1, s \rangle & \quad \langle \pi_2, s \rangle \rightarrow \langle \pi_2', s' \rangle \\
\langle \pi_1 + \pi_2, s \rangle \rightarrow \langle \pi_1', s' \rangle & \quad \langle \pi_1 + \pi_2, s \rangle \rightarrow \langle \pi_2', s' \rangle \\
\langle \pi, s \rangle \rightarrow \langle \varphi', s' \rangle & \quad \langle \pi, s \rangle \rightarrow \langle \varphi', \pi, s' \rangle \\
\langle \pi, s \rangle \rightarrow \langle E, s \rangle & \quad \langle \pi, s \rangle \rightarrow \langle E, s \rangle \\
\end{align*}
\]

The same transition semantics can essentially unmodified (replace DAL states with DEL states and restrict tests to knowledge propositions) also be used for DEL programs. An operational semantics that defines an input-output relation for DAL programs can now be defined in the standard way by means of the transitive closure \( \rightarrow^+ \) (cf. [11]).

Definition 14. (Operational Semantics for DAL Programs)
The operational semantics for test DAL programs \( \pi \) is defined by:

\[
O(\pi)(s) = \{ s' \mid \langle \pi, s \rangle \rightarrow^+ \langle E, s' \rangle \}
\]

3 The symbol \( E \) is used to denote termination of a program.
A standard technique to relate the logical and operational semantics is to define a denotational semantics for DAL programs (cf. [12]). The denotational semantics can directly be derived from the state-based semantics for DAL by fixing an interpretation of atomic actions. This interpretation should correspond with the transition function $T$ to ensure equivalence with the operational semantics. To this end, we introduce the notion of $T$-compatible DAL structures.

**Definition 15.** ($T$-Compatible Structures)
Let $M^c = \langle W^c, R^c \rangle$ be a state-based DAL structure and $s, s' \in W^c$. $M^c$ is said to be $T$-compatible if it satisfies: $s R^c a s'$ iff $s' \in T(a, s)$ for all atomic actions $a$.

**Definition 16.** (Denotational Semantics for DAL Programs)
Let $M^c$ be a state-based, $T$-compatible DAL structure. Then the denotational semantics for DAL programs is inductively defined by:

\[
[a](s) = \{ s' \mid s R^c a s' \} \\
[\pi_1; \pi_2](s) = \bigcup_{s' \in [\pi_1](s)} [\pi_2](s') \\
[\pi^?](s) = \begin{cases} \{ s \}, & \text{if } M^c, s \models \varphi \\ \emptyset, & \text{otherwise} \end{cases} \\
[\pi_1 + \pi_2](s) = [\pi_1](s) \cup [\pi_2](s) \\
[\pi^*](s) = \bigcup_{n \geq 0} [\pi^n](s)
\]

It is easy to show that $[\_]$ is well-defined. The next theorem states that the denotational semantics and the operational semantics are equivalent.

**Theorem 3.** (Equivalence of Denotational and Operational Semantics)
The denotational and operational semantics for DAL programs are equivalent:

\[
[\pi](s) = O(\pi)(s)
\]

**Proof.** Use induction on the program structure. \(\square\)

The equivalence of the denotational and operational semantics shows that the modal agent logic can be used to verify (partial) correctness properties of DAL programs. This fact is expressed mathematically in the following corollary. The left hand side of the corollary statement concerns the step-by-step behaviour of a (terminating) agent program whereas the equivalence with the right hand side shows that the agent logic can be used to (compositionally) reason about such programs, without the need to inspect the operational behaviour of an agent program.

**Corollary 1.** (Proving Partial Correctness Properties of DAL Programs)
Let $\pi$ be a poor test DAL program. Then we have:

\[
\forall s, s': \text{ if } \models s \varphi \text{ and } (\pi, s) \longrightarrow^* (E, s') \text{, then } \models s', \psi \\
\text{iff} \\
\models \varphi \rightarrow [\pi]\psi
\]

**Proof.** Immediate from theorems 2 and 3. \(\square\)
4 Goal Dynamics

In the previous sections it was shown how declarative goals can be incorporated into an agent programming theory. The main objective was to ensure that the agent logic can be used to reason about agent programs that have a computational semantics, which was achieved by introducing some reasonable constraints on the modal semantics. One of those constraints, however, required goals to be static. In this section, we relax this constraint, by introducing a mechanism for goal adoption based on second-order goals. The mechanism we introduce is based on a dispositional view of second-order goals: An agent with a second-order goal $\varphi$ is disposed to adopt $\varphi$ as a first-order goal.

In the literature, a goal has been interpreted primarily as a first-order goal denoting a state that can be achieved through action (cf. [13]). Second-order and higher-order goals have not been paid much attention to, though some related notions have been discussed. As mentioned, Sloman et al. in [8] discuss second-order motives as first-order goal generators. The notion of a motive in [13] and that of concern in [14] is also related to our dispositional view of second-order goals. A motive or concern is more stable than a first-order goal, just as, in our view, second-order goals are. In [13, 14] motives and concerns respectively provide reasons for goal adoption. In [13] it is claimed that (pro-active) goal adoption based on motives provides for a more efficient goal adoption mechanism, which is based on attention triggering in [13].

One intuitive interpretation of second-order goals is that they are dispositions to change one’s mental state and, given the right circumstances, result in the adoption of a (first-order) goal to achieve $\varphi$. Second-order goals thus have a “monitoring” function. As an example, consider a cleaning robot. A cleaning robot does not have (to have) a goal to clean garbage when there is no garbage, but it should be disposed to clean any garbage if there is any.

Goal Adoption in the Agent Programming Language An agent program is extended with goal adoption rules of the form $\varphi \Rightarrow_{G} \psi$ to implement the adoption of first-order goals $\psi$. The formula $\varphi$ is called the triggering condition of the rule and $\psi$ the goal of the rule. The triggering condition $\varphi$ and goal $\psi$ must be knowledge propositions, i.e. $\varphi, \psi \in L_k$. Informally, a goal adoption rule $\varphi \Rightarrow_{G} \psi$ fires when the agent is disposed to adopt the goal $\varphi$, i.e. $G\varphi$ holds and the circumstances are believed to be right by the agent, i.e. $K\varphi$ holds. Additionally, a goal is only adopted if it is consistent with the current set of goals of an agent. The rationale for this is the assumption that an agent that is disposed to adopt $\psi$, i.e. $G\varphi$ but also has a goal $\neg \psi$, will not have self-generated this goal, but e.g. have adopted such a goal upon request of another agent. A goal adoption rule in our programming language is similar to the notion of a goal template in [13], whereas, the notion of a so-called “repository of known goals” in [10] can be related to our operationalization of second-order goals.

Formally, an (extended) agent program is defined as a pair $A = (\pi, \Gamma)$ with $\Gamma$ a set of goal adoption rules. We introduce some additional notation to facilitate the introduction of the transition semantics: Given an agent $A = (\pi, \Gamma)$ we use
\( A[\pi'] \) to denote the agent \( \langle \pi', \Gamma \rangle \); given a state \( s = \langle v, k, g \rangle \) we use \( s[d/g(0)] \) to denote the state \( \langle v, k, g' \rangle \) with \( g'(0) = d \) and \( g'(i + 1) = g(i + 1) \). The operational semantics for agent programs is extended with the following transition rule.

**Definition 17. (Transition Rule for Goal Adoption)**

Let \( s = \langle v, k, g \rangle, s' = \langle v', k', g' \rangle \) be DAL states and define the adoption set \( e \) by:

\[
e = \bigcup_{\varphi \Rightarrow G \psi \in \Gamma} \{ \psi | \models_s \varphi, \models_s GG\psi, \models_s \neg G \neg \psi \}
\]

Then the transition semantics of an agent program \( A = \langle \pi, \Gamma \rangle \) is defined by:

\[
\langle \pi, s \rangle \rightarrow \langle \pi', s' \rangle
\]

\[
\langle A, s \rangle \rightarrow \langle A[\pi'], s'[g(0) \cup e/g(0)] \rangle
\]

As a simple example to illustrate the semantics, assume that a cleaning robot is disposed to adopt the goal \( \text{GarbageInBin} \), i.e. \( GG\text{GarbageInBin} \), initially has no goals, i.e. \( g(0) = \emptyset \), and executes the program:

\((G\text{GarbageInBin}; \text{clean} + \neg G\text{GarbageInBin}; \text{observe}; \text{wander})^*; \text{K(time = 6pm)}\)

with \text{clean}, \text{observe} and \text{wander} atomic actions. The robot is supposed to clean until 6pm every day. Additionally, a single goal adoption rule \( \text{garbage} \Rightarrow G \text{GarbageInBin} \) is part of the robot’s program. Since, initially, the robot does not have any goals it will execute the right branch of the choice program and perform the actions \text{observe} and \text{wander} (repeatedly). We assume that when the robot observes garbage it updates its knowledge base accordingly. In that case, all the conditions for firing the rule are satisfied: the robot knows there is garbage, \( K\text{garbage} \), is disposed to clean it, \( GG\text{GarbageInBin} \) and adopting this goal is not inconsistent with its current goals, \( \neg G \neg \text{GarbageInBin} \). Accordingly, the rule fires and the goal base of the agent is expanded with the first-order goal \( \text{GarbageInBin} \). Consecutively, the robot will execute the left branch in the choice program and start cleaning the garbage by performing the \text{clean} action.

**Goal Adoption in the Agent Logic** Finally, in the agent program logic, analogously logical rules can be introduced to reason about the effects of goal adoption rules \( \varphi \Rightarrow G \psi \). These effects can be formalized by goal adoption axioms of the form:

\[
(K\varphi \land GG\psi \land \neg G \neg \psi) \rightarrow \Box G\psi
\]

For any agent program, a set of goal adoption axioms corresponding to the goal adoption rules can be added to the agent program logic to reason about the goal adoption mechanism of the agent and to verify its correctness. Semantically, the constraint imposed on the semantics of goals which required the goals of an agent to be static needs to be relaxed. In particular, the third constraint in Constraint 1 needs to be adjusted accordingly for all models \( M = \langle W, R, K, G, V \rangle \) and atomic actions \( a \). To facilitate the introduction of the new semantic constraint
that corresponds with the goal adoption mechanism, as a preliminary step, we first define the set $e$ as follows:

$$e(w) = \bigcap_{\varphi \Rightarrow G\psi \in \Gamma} \{w'' \in W \mid (M, w \models K\varphi \land G\psi \land \neg G\neg\psi) \Rightarrow M, w'' \models \psi\}$$

Then the third constraint in Constraint 1 is to be replaced with:

$$\forall w, w'(wR_a w' \Rightarrow G(w') = G(w) \cap e(w))$$

This new constraint illustrates the fact that the goals of an agent are expanded by firing goal adoption rules since the all worlds that do not satisfy the adopted goals are removed from the set of alternative goal worlds $G(w)$.

**Remark 2.** It will be clear that care should be taken to avoid inconsistency when goal adoption axioms are added to the agent logic. A consistency check with the previously adopted goals of an agent is built into the semantics, but the consistency of the combined set of adopted goals is not incorporated. In the transition rule above, the consistency of the adoption set $e$ itself is not checked. This remains a task for the designer or programmer, who has several methods available to ensure consistency. The most simple technique is to ensure that the triggering conditions of the rules are mutually exclusive in the sense that only one triggering condition can be true at any time. This may be too restrictive, however, and a more advanced approach would be to prove that the set of goals of arbitrary rule sets that may fire in the same state (i.e. the triggering conditions of these rules are not mutually exclusive) are consistent.

Goal adoption rules provide a mechanism for goal adoption based on the (second-order) dispositions of an agent, but also provide a means to implement adjustable autonomy. Note that from the axiom $GG\varphi \lor G\neg G\varphi$, it follows that an agent will be either disposed to adopt $\varphi$ as a (first-order) goal or will not be disposed to do that. Using this principle, the autonomy of agents can be explicitly adjusted by restricting or relaxing by means of second-order goals, i.e. the set of first-order goals that an agent may generate itself (disposed to adopt) and which not. Those goals $\varphi$ that the agent is not disposed to adopt, $G\neg G\varphi$, still may be adopted by the agent, but not through self-generation but, e.g. through a mechanism for handling requests from other agents. Second-order goals thus provide a concrete mechanism for implementing adjustable autonomy (cf. [15]).

5 Conclusion

In this paper, an agent programming theory has been introduced. This agent programming theory consists of an agent programming language and a corresponding agent logic to verify the correctness of the agent programs. The agent logic incorporates the core agent concepts of action, knowledge and goals of an agent. To be useful as a program logic for agent programs, a precise correspondence of the logical and program semantics must be established. It has been shown how declarative goals can be incorporated into the operational program semantics of an agent programming language in a way that preserves this precise correspondence between the operational and logical semantics. A number of
constraints that were imposed on the logical semantics to establish this correspondence were discussed and motivated. Additionally, a mechanism for adopting goals has been introduced based on a dispositional view of second-order goals: second-order goals are dispositions to adopt goals and can be used by the agent to self-generate goals and by a programmer to adjust the autonomy of the agent.

In our agent programming theory, we have incorporated a mechanism for goal expansion but not yet one for goal revision. An approach based on the introduction of an axiom like $K\varphi \rightarrow \neg \Box \varphi$ combined with an axiom to capture goal persistence $\Box \varphi \rightarrow \Box (\varphi \lor K\varphi)$ does not seem to work: There is a simple counterexample to the goal persistence axiom with $\varphi = (p \land q)$ a conjunctive goal and the assumption that at the next time only $p$ is achieved. In [7] an interesting approach for goal change has been proposed and we would like to study such proposals for goal revision in our programming theory as well. Additionally, future research will involve a detailed comparison of our approach with other approaches to agent programming and to the verification of agents.

References

More Games in Belief Game Model: 
A Preliminary Report

Sébastien Konieczny

CRIL - CNRS
Université d’Artois
62300 Lens - France
konieczny@cril.univ-artois.fr

Abstract. Modelization of negotiation between agents that have conflictual beliefs/goals is an important question. Many negotiation protocols have been proposed in the multi-agent literature. But they are mainly descriptive works. We propose to look at this problem in a prescriptive way, trying to define what should be the result of a negotiation if one could avoid all communications/manipulations problems.

In previous works Belief Negotiation Models and Belief Game Models were proposed to this aim. Those operators rely on an iterative selection-weakening process to find a consensus (agreement). So a particular operator is defined once a choice function and a weakening function is chosen. In those works the choice function was defined arbitrarily. In this work we propose to take as such choice function a measure that allows to define the amount of conflict brought by each agent. So it gives a more intuitive way of defining these operators.

1 Introduction

Negotiation is defined as a process aimed at finding an agreement between several agents. Many negotiation protocols have been proposed in the multi-agent literature such as argumentation, dialog games, etc. All those approaches have in common that they can be viewed as a game between the agents, ruled by some fixed protocol, where one agent declares something, and where the others can accept what is declared, challenge it, or propose a competitive view, etc.

So negotiation protocols can be viewed as a kind of game with incomplete information, since, as the exchange between the agents is ruled by a given protocol, the result of the negotiation does not take into account the whole opinion of each agent, but only what each agent have declared in the course of the negotiation process. It means in particular that 1) it may happen that the optimal agreement is missed because some important points were not evoked during the negotiation process 2) the result of the negotiation process can differ depending of the order in which the agents intervened (and especially of the first one to talk). Those problems can be viewed as inherent drawbacks of those methods, that can not guarantee that the “optimal” agreement is reached.
So one interesting question is to try to define what this “optimal” agreement is. There is not a unique and definite answer to this question: what is the best agreement between several agents that pursue their own goals is one of the main question that is studied in game theory since years. The bargaining problem [1] can be viewed as the prototypical negotiation problem: among a set of possible outcomes\(^1\) (vector of utility of each player) a set of players (two in the prototypical case) have to agree on a particular outcome. If they do not agree, a given outcome will be the result. The problem is, given only these hypothesis, to define what is the optimal/fair result for the bargaining. As one could expect, there is no unique possible interpretation of optimality/fairness, and that gives rise to a large number of well defined solution concepts [2].

This part of cooperative game theory has to be related to non-cooperative game theory, where one does not allow simple agreements like above, but ask the agents to play a game (roughly a sequence of moves) to achieve the best result from their respective point of view. It is well known that for most games the cooperative solution is better for all agents than the non-cooperative one.

Negotiation protocols can be viewed as a kind of non-cooperative games (of incomplete information as explained above). So an interesting question is to study their cooperative counterpart. This would allow to find better solutions than with negotiation protocols. We call these kind of operators conciliation operators [3].

This does not mean that conciliation operators are better than negotiation protocols. They can provide better solutions, but they do not take into account the communication issues of real applications, they suppose that all the agents provide their full opinion base, and that they are all cooperative enough to let an outside judge compute the result. Those assumptions are too strong if one consider autonomous agents, nevertheless conciliation operators can be seen as an idealization of negotiation, when all those limitations do not interfere with the agreement seeking. So the results given by conciliation operators can be used as test cases to compare particular negotiation protocols (the closer from the conciliation operator results, the better).

The problem of modelization of negotiation has been investigated recently under the scope of belief change tools [4–9, 3]. The problem is to define operators that take as input belief profiles (multi-set of propositional belief bases representing the beliefs/goals of the agents) and that produce a new belief profile that aims to be less conflicting. The idea followed in [5, 6, 9] to define conciliation operators is to use an iterative process where at each step a set of agents is selected. These selected agents are the ones that have to weaken their point of view. The process stops when an agreement, called consensus, is reached. Many interesting operators can be defined when one fixes the choice function (the function that selects the agents that must be weakened at each round) and the weakening method. In [9] the choice function is based on a notion of distance. It can be sensible if such a distance is meaningful in a particular application. If not, it is only an arbitrary choice.

\(^1\) Usually the set of outcomes is a convex and bounded set.
What we propose in this paper is to use as choice function a measure that will allow to know the amount of conflict brought by each agent. So the agents that will have to concede are the ones that bring more conflict. As we will show existing inconsistency measures are not sufficient to do so, and we will need to define inconsistency measures based on Shapley value (that is a solution concept of cooperative game theory).

The rest of the paper will be articulated as follows. After a preliminary section with some definitions, in Section 3 we introduce Belief Game Models. In Section 4 we recall some definitions of inconsistency values and define Shapley inconsistency values. In Section 5 we will then introduce the BGM operators using the Shapley inconsistency values. We give some conclusions and perspectives in Section 6.

2 Preliminaries

We consider a propositional language \(\mathcal{L}\) over a finite alphabet \(\mathcal{P}\) of propositional symbols. An interpretation is a function from \(\mathcal{P}\) to \(\{0, 1\}\). The set of all the interpretations is denoted \(\mathcal{W}\). An interpretation \(\omega\) is a model of a formula \(\varphi\), noted \(\omega \models \varphi\), if and only if it makes it true in the usual classical truth functional way. Let \(\varphi\) be a formula, \(\text{mod}(\varphi)\) denotes the set of models of \(\varphi\), i.e. \(\text{mod}(\varphi) = \{\omega \in \mathcal{W} \mid \omega \models \varphi\}\). Conversely, let \(X\) be a set of interpretations, \(\text{form}(X)\) denotes the formula (up to logical equivalence) whose set of models is \(X\).

A belief base \(\varphi\) is a consistent propositional formula (or, equivalently, a finite consistent set of propositional formulae considered conjunctively). Let us note \(\mathcal{K}\) the set of all belief bases. A belief base will represent the beliefs/goals of an agent.

Let \(\varphi_1, \ldots, \varphi_n\) be \(n\) belief bases (not necessarily different). We call belief profile the multi-set \(\Psi\) consisting of those \(n\) belief bases: \(\Psi = (\varphi_1, \ldots, \varphi_n)\) (i.e. two agents can have the same belief base). We note \(\bigwedge \Psi\) the conjunction of the belief bases of \(\Psi\), i.e. \(\bigwedge \Psi = \varphi_1 \land \cdots \land \varphi_n\). We say that a belief profile is consistent if \(\bigwedge \Psi\) is consistent. The multi-set union will be noted \(\sqcup\) and the multi-set inclusion will be noted \(\subseteq\). The cardinal of a finite (multi-)set \(\Psi\) is noted \(#(\Psi)\) (the cardinal of a finite multi-set is the sum of the numbers of occurrences of each of its elements). Let \(\mathcal{E}\) be the set of all finite non-empty belief profiles.

Two belief profiles \(\Psi_1\) and \(\Psi_2\) are said to be equivalent (\(\Psi_1 \equiv \Psi_2\)) if and only if there is a bijection between \(\Psi_1\) and \(\Psi_2\) such that each belief base of \(\Psi_1\) is logically equivalent to its image in \(\Psi_2\).

3 Belief Game Model

In [4, 5] Richard Booth introduces Belief Negotiation Models, that are conciliation operators that reach a consensus through an iterative selection-weakening process. This work is an interesting abstraction of negotiation. The idea is that negotiation aimed at finding a consensus between several agents having conflictual points of view(i.e. jointly inconsistent bases). In this aim some agents will
have to concede on their point of view in order to reach a consensus (i.e. jointly consistent bases). So, one step of this negotiation process is to defined what are the agents that have to concede (they can be chosen one after the other, or one can choose the more problematic agents, etc...). Once a set of agents is selected, all the agents in this set has to weaken her point of view. This process is repeated until a consensus is reached. So one particular Belief Negotiation Models is defined when one define the way the agents are selected and the way they weaken their points of view.

This work was extended in [9], where a specialized case, Belief Game Models (or BGM) was studied. See [9] about the exact relationship between Belief Negotiation Models and Belief Game Models. Let us now introduce Belief Game Models.

**Definition 1.** A choice function is a function $g : \mathcal{E} \rightarrow \mathcal{E}$ such that:

- $g(\Psi) \sqsubseteq \Psi$
- If $\bigwedge \Psi \neq \top$, then $\exists \varphi \in g(\Psi)$ s.t. $\varphi \notin \top$
- If $\Psi \equiv \Psi'$, then $g(\Psi) \equiv g(\Psi')$

The choice function aims to find which are the agents that must weaken at a given round (see definition 3).

As the weakening function aims to weaken the belief base, and as there is no weaker base than a tautological one, the second condition states that at least one non-tautological base must be selected. So it states that at each round at least one base will be weaken. Last condition is an irrelevance of syntax condition. It states that the selection of the bases to weaken does not depend on the particular form of the bases, but only on their informational content. It also means that if two bases are logically equivalent then if one is chosen, the other one will be chosen as well. It means that it is only the opinion (beliefs/goals) of the agent that can be considered by the choice function and not the agent itself.

**Definition 2.** A weakening function is a function $\nabla : \mathcal{L} \rightarrow \mathcal{L}$ such that:

- $\varphi \vdash_\nabla(\varphi)$
- If $\varphi \equiv \nabla(\varphi)$, then $\varphi \equiv \top$
- If $\varphi \equiv \varphi'$, then $\nabla(\varphi) \equiv \nabla(\varphi')$

The weakening function aims to give the new beliefs of an agent that have been chosen to be weakened. The two first conditions ensure that the base will be replaced by a strictly weaker one (unless the base is already a tautological one). The last condition is an irrelevance of syntax requirement: the result of the weakening must only depend on the information conveyed by the base, not on its syntactical form.

We extend the weakening functions on belief profiles as follows: let $\Psi'$ be a subset of $\Psi$,

$$\nabla_{\Psi'}(\Psi) = \bigsqcup_{\varphi \in \Psi'} \nabla(\varphi) \sqcup \bigsqcup_{\varphi \in \Psi \setminus \Psi'} \varphi$$
This means that we only weaken the belief bases of $\Psi$ that are in $\Psi'$, and the other ones do not change.

In some cases, the result of the merging has to obey some constraints (physical constraints, norms, etc...). We will assume that these integrity constraints are encoded as a propositional formula, and we will note this formula $\mu$. Then we introduce Belief Game Models (BGM):

**Definition 3.** The solution to a belief profile $\Psi$ for a Belief Game Model $N = \langle g, \triangledown \rangle$ under the integrity constraints $\mu$, noted $N_{\mu}(\Psi)$, is the belief profile $\Psi_{\mu}$ defined as:

- $\Psi_0 = \Psi$
- $\Psi_{i+1} = \triangledown_g(\Psi_i)$
- $\Psi_{\mu}$ is the first $\Psi_i$ that is consistent with $\mu$

So the solution to a belief profile is the result of a game on the beliefs of the agents. At each round there is a contest to find out the weakest bases (given by the choice function), and these bases have to concede on their beliefs by weakening them.

Now let us give two examples of weakening functions and two families of choice functions.

**Definition 4.** Let $\varphi$ be a belief base.

- The drastic weakening function forget all the information about one agent, i.e. $: \triangledown_\top(\varphi) = \top$.
- The dilation weakening function is defined as:

$$\text{mod}(\triangledown_\delta(\varphi)) = \{ \omega \in \mathcal{W} \mid \exists \omega' \models \varphi \ d_H(\omega, \omega') \leq 1 \}$$

where $d_H$ is the Hamming distance between interpretations, i.e. the number of propositional symbols on which the two interpretations differ. Let $\omega$ and $\omega'$ be two interpretations, then $d_H(\omega, \omega') = \#(\{a \in \mathcal{P} \mid \omega(a) \neq \omega'(a)\})$.

Before giving examples of choice functions we have to state some definitions:

**Definition 5.** A (pseudo)distance $d$ between two belief bases is a function $d : \mathcal{L} \times \mathcal{L} \to \mathbb{N}$ such that $d(\varphi, \varphi') = 0$ iff $\varphi \wedge \varphi' \not\models \bot$ and $d(\varphi, \varphi') = d(\varphi', \varphi)$.

Two examples of such distances are:

$$d_D(\varphi, \varphi') = \begin{cases} 0 & \text{if } \varphi \wedge \varphi' \not\models \bot \\ 1 & \text{otherwise} \end{cases} \quad d_H(\varphi, \varphi') = \min_{\omega \models \varphi, \omega' \models \varphi'} d_H(\omega, \omega')$$

**Definition 6.** An aggregation function is a total function $f$ associating a non-negative integer to every finite tuple of nonnegative integers and verifying (non-decreasingness), (minimality) and (identity).

- if $x \leq y$, then $f(x_1, \ldots, x_n) \leq f(x_1, \ldots, y, \ldots, x_n)$. (non-decreasingness)
- $f(x_1, \ldots, x_n) = 0$ if and only if $x_1 = \ldots = x_n = 0$. (minimality)
for every nonnegative integer \( x \), \( f(x) = x \). (identity)

We say that an aggregation function is symmetric if it also satisfies:

- For any permutation \( \sigma \), \( f(x_1, \ldots, x_n) = f(x_{\sigma(1)}, \ldots, x_{\sigma(n)}) \) (symmetry)

**Definition 7.** A symmetric model-based choice function \( g^{d,h} \) is defined as:

\[
g^{d,h}(\Psi) = \{ \varphi_i \in \Psi \mid h(d(\varphi_i, \varphi_1), \ldots, d(\varphi_i, \varphi_n)) \text{ is maximal} \}
\]

where \( h \) is a symmetric aggregation function, and \( d \) is a distance between belief bases.

So these choice functions select the bases that are the “furthest” from the others, according to one chosen “distance” notion. The other family of choice functions is called formula-based choice functions, and rely on maximal consistent subsets of bases. These maximal consistent subsets are considered as the closest path to consistency (hence to consensus), and the selected bases are the ones that have the minimum “score” with respect to these maximal consistent subsets. Naturally there are several ways to compute this score, that gives rise to different operators.

**Definition 8.** Let \( \text{MAXCONS}(\Psi) \) be the set of the maxcons of \( \Psi \), i.e. the maximal (with respect to multi-set inclusion) consistent subsets of \( \Psi \). Formally, \( \text{MAXCONS}(\Psi) \) is the set of all multi-sets \( M \) such that:

- \( M \subseteq \Psi \),
- \( \bigwedge M \not\models \perp \) and
- if \( M \subseteq M' \subseteq \Psi \), then \( \bigwedge M' \models \perp \).

**Definition 9.** A formula-based choice function \( g^{mc} \) is a function of the set of the maxcons of \( \Psi \) and the belief base, i.e. :

\[
g^{mc}(\Psi) = \{ \varphi_i \in \Psi \mid h(\varphi_i, \text{MAXCONS}(\Psi)) \text{ is minimal} \}
\]

Examples of the use of maxcons are numerous, let us see two of them.

**Definition 10.**

- \( h^{mc1}(\varphi, \text{MAXCONS}(\Psi)) = \#(\{ M \mid M \in \text{MAXCONS}(\Psi) \text{ and } \varphi \in M \}) \)
- \( h^{mc2}(\varphi, \text{MAXCONS}(\Psi)) = \max(\{ \#(M) \mid M \in \text{MAXCONS}(\Psi) \text{ and } \varphi \in M \}) \)

For the first choice function the score of a base is the number of maxcons to which this base belongs. For the second function the score of a base is the size of the biggest maxcons to which this base belongs.

Let us see on an example [10], what is the behaviour of some BGM operators, namely the operators \( \langle g^{dH,\Sigma}, \blacksquare \rangle \), \( \langle g^{dH,\max}, \blacksquare \rangle \), \( \langle g^{mc1}, \blacksquare \rangle \) and \( \langle g^{mc2}, \blacksquare \rangle \).
Example 1. There are three agents $\Psi = \{\varphi_1, \varphi_2, \varphi_3\}$ with the following belief bases $\varphi_1 = \{\neg a \land (a \lor c)\}$, $\varphi_2 = \{(\neg a \land b) \land \neg a \land b \land c\}$, $\varphi_3 = \{a \land b \land c\}$. For the computations given below it is easier to consider those bases as sets of models, so $\text{Mod}(\varphi_1) = \{(1, 0, 0), (0, 0, 1), (1, 0, 1)\}$, $\text{Mod}(\varphi_2) = \{(0, 1, 0), (0, 0, 1)\}$, and $\text{Mod}(\varphi_3) = \{(1, 1, 1)\}$. There are no constraints on the result, so $\mu = \top$.

- $(g^{3H, \Sigma}, \nabla_3)$: As $\Psi$ is not consistent, let us do the first round. $d(\varphi_1, \varphi_2) = 0$, $d(\varphi_1, \varphi_3) = 1$, $d(\varphi_2, \varphi_3) = 2$. So $h^0_\Sigma(\varphi_1) = 1$, $h^0_\Sigma(\varphi_2) = 2$, $h^0_\Sigma(\varphi_3) = 3$. That gives $g^{3H, \Sigma}(\Psi) = \{\varphi_3\}$. So $\varphi_3$ is replaced by $\varphi_3 = \nabla_3(\varphi_3) = \text{form}(\{(1, 1, 1), (1, 1, 0), (1, 0, 1), (0, 0, 1)\})$. We have not yet reached a consistent $\Psi$, so let us do a further round. Let us first compute the new distances. $d(\varphi_1, \varphi_2) = 0$, $d(\varphi_1, \varphi_3) = 0$, $d(\varphi_2, \varphi_3) = 1$. So $h^1_\Sigma(\varphi_1) = 0$, $h^1_\Sigma(\varphi_2) = 1$, $h^1_\Sigma(\varphi_3) = 1$. That gives $g^{3H, \Sigma}(\Psi) = \{\varphi_2, \varphi_3\}$. So $\varphi_2$ is replaced by $\varphi_2 = \nabla_3(\varphi_2) = \text{form}(\{(0, 1, 0), (0, 0, 1), (1, 1, 0), (0, 0, 0), (0, 1, 1), (1, 0, 1)\})$, and $\varphi_3$ is replaced by $\varphi_3 = \nabla_3(\varphi_3) = \text{form}(\{(1, 1, 1), (1, 1, 0), (1, 0, 1), (0, 1, 0), (1, 0, 0), (0, 0, 1)\})$. We have reached a consistent belief profile, so the result is $\Psi' = \{\varphi_1, \varphi_2, \varphi_3\}$, and the conjunction (consensus/agreement) is the base whose models are $\{(0, 0, 1), (1, 0, 1)\}$.

- $(g^{3H, \text{max}}, \nabla_3)$: As $\Psi$ is not consistent, let us do the first round. $d(\varphi_1, \varphi_2) = 0$, $d(\varphi_1, \varphi_3) = 1$, $d(\varphi_2, \varphi_3) = 2$. So $h^{\text{max}}(\varphi_1) = 1$, $h^{\text{max}}(\varphi_2) = 2$, $h^{\text{max}}(\varphi_3) = 2$. That gives $g^{3H, \text{max}}(\Psi) = \{\varphi_2, \varphi_3\}$. So $\varphi_2$ is replaced by $\varphi_2 = \nabla_3(\varphi_2) = \text{form}(\{(0, 1, 0), (0, 0, 1), (1, 1, 0), (0, 0, 0), (0, 1, 1), (1, 0, 1)\})$, and $\varphi_3$ is replaced by $\varphi_3 = \nabla_3(\varphi_3) = \text{form}(\{(1, 1, 1), (1, 1, 0), (1, 0, 1), (0, 1, 0)\})$. The obtained profile is consistent, so the result is $\Psi' = \{\varphi_1, \varphi_2, \varphi_3\}$, and the model of the conjunction is $\{(1, 0, 1)\}$.

- $(g^{3H, \text{max}}, \nabla_3)$: As $\Psi$ is not consistent, and $\text{MAXCONS}(\Psi) = \{(\varphi_1, \varphi_2), (\varphi_3)\}$. So $h^{\text{max}}(\varphi_1) = h^{\text{max}}(\varphi_2) = h^{\text{max}}(\varphi_3) = 1$, and $g^{\text{max}}(\Psi) = \Psi$. So we weaken the three bases, which gives respectively $\varphi_1 = \nabla_3(\varphi_1) = \text{form}(\{(1, 1, 0), (0, 0, 1), (1, 0, 1), (0, 0, 0), (1, 1, 0), (0, 1, 1)\})$, $\varphi_2 = \nabla_3(\varphi_2) = \text{form}(\{(0, 1, 0), (0, 0, 1), (1, 1, 0), (0, 0, 0), (0, 1, 1), (1, 0, 1)\})$, and $\varphi_3 = \nabla_3(\varphi_3) = \text{form}(\{(1, 1, 1), (1, 1, 0), (1, 0, 1), (0, 1, 1)\})$. This belief profile is consistent, and the result is $\Psi' = \{\varphi_1, \varphi_2, \varphi_3\}$.

- $(g^{mc2}, \nabla_3)$: $\Psi$ is not consistent, and we have $\text{MAXCONS}(\Psi) = \{(\varphi_1, \varphi_2), (\varphi_3)\}$. So $h^{mc2}(\varphi_1) = h^{mc2}(\varphi_2) = 2$ and $h^{mc2}(\varphi_3) = 1$, and $g^{mc2}(\Psi) = \Psi$. So $\varphi_3$ is replaced by $\varphi_3 = \nabla_3(\varphi_3) = \text{form}(\{(1, 1, 1), (1, 0, 1), (0, 1, 1)\})$. The belief profile is still not consistent, so we need one more round. Now we have $\text{MAXCONS}(\Psi) = \{(\varphi_1, \varphi_2), (\varphi_1, \varphi_3)\}$. So $h^{mc2}(\varphi_1) = h^{mc2}(\varphi_2) = h^{mc2}(\varphi_3) = 2$, and $g^{mc2}(\Psi) = \Psi$. So we weaken the three bases, which gives respectively $\varphi_1 = \nabla_3(\varphi_1) = \text{form}(\{(1, 0, 0), (0, 0, 1), (1, 0, 1), (0, 0, 0), (1, 1, 0), (0, 1, 1)\})$, $\varphi_2 = \nabla_3(\varphi_2) = \text{form}(\{(0, 1, 0), (0, 0, 1), (1, 1, 0), (0, 0, 0), (0, 1, 1), (1, 0, 1)\})$, and $\varphi_3 = \nabla_3(\varphi_3) = \text{form}(\{(1, 1, 1), (1, 1, 0), (0, 1, 1), (0, 1, 1)\})$. The belief profile is consistent, and the result is $\Psi' = \{\varphi_1, \varphi_2, \varphi_3\}$.
4 Shapley Inconsistency Values

The aim of this paper is to improve BGM by using as choice function an inconsistency measure that allows to state how much conflict is brought by each agent. We will first introduce usual inconsistency measures. Those inconsistency measures are defined for one (unique) base/source/agent, and do not allow to associate to an agent in a group its amount of conflict. One possibility is to give as amount of conflict of an agent the difference between the total amount of conflict of the group minus the amount of conflict of the group without this agent. But we introduce a more precise measure in Section 4.3.

4.1 Inconsistency Measures based on Variables

A method to evaluate the inconsistency of a set of formulas is to look at the proportion of the language touched by the inconsistency. To this end, it is clearly not possible to use classical logic, since the inconsistency contaminates the whole language. But if we look at the two profiles $\Psi_1 = \{a, \neg a, b \land c, d\}$ and $\Psi_2 = \{a, \neg a, b \land \neg c, c \land \neg b, d, \neg d\}$, we can observe that in $\Psi_1$ the inconsistency is mainly about the variable $a$, whereas in $\Psi_2$ all the variables are touched by a contradiction. This is this kind of distinction that these approaches allow.

One way to circumscribe the inconsistency to the variables directly concerned only is to use multi-valued logics, and especially three-valued logics, with the third “truth value” denoting the fact that there is a conflict on the truth value (true-false) of the variable.

We do not have space here to detail the range of different measures that have been proposed. See [11–15] for more details on these approaches. We only give one such measure, that is a special case of the degrees of contradiction defined in [13]. The idea of the definition of these degrees in [13] is, given a set of tests on the truth value of some formulae of the language (typically on the variables), the degree of contradiction is the cost of a minimum test plan that ensures recovery of consistency.

The inconsistency measure we define here is the (normalized) minimum number of inconsistent truth values in the $LP_m$ models [16] of the belief base. Let us first introduce the $LP_m$ consequence relation.

An interpretation $\omega$ for $LP_m$ maps each propositional atom to one of the three “truth values” $F, B, T$, the third truth value $B$ meaning intuitively “both true and false”. $3^P$ is the set of all interpretations for $LP_m$. “Truth values” are ordered as follows: $F <_T B <_T T$.

\[
- \omega(\top) = T, \omega(\bot) = F \\
- \omega(\neg \alpha) = B \text{ iff } \omega(\alpha) = B \\
\omega(\neg \alpha) = T \text{ iff } \omega(\alpha) = F \\
- \omega(\alpha \land \beta) = \min_{\leq_T} (\omega(\alpha), \omega(\beta)) \\
- \omega(\alpha \lor \beta) = \max_{\leq_T} (\omega(\alpha), \omega(\beta))
\]

The set of models of a formula $\varphi$ is:

$Mod_{LP}(\varphi) = \{\omega \in 3^P \mid \omega(\varphi) \in \{T, B\}\}$

24
Define $\omega!$ as the set of “inconsistent” variables in an interpretation $w$, i.e.

$$\omega! = \{x \in P \mid \omega(x) = B\}$$

Then the minimum models of a formula are the “most classical” ones:

$$\min(\text{Mod}_{LP}(\varphi)) = \{\omega \in \text{Mod}_{LP}(\varphi) \mid \exists \omega' \in \text{Mod}_{LP}(\varphi) \text{ s.t. } \omega! \subset \omega'!\}$$

The $LP_m$ consequence relation is then defined by:

$$\varphi \models_{LP_m} \varphi \text{ iff } \min(\text{Mod}_{LP}(\varphi)) \subseteq \text{Mod}_{LP}(\varphi)$$

So $\varphi$ is a consequence of $\varphi$ if all the “most classical” models of $\varphi$ are models of $\varphi$. And the models of a profile $\Psi$ are straightforwardly the interpretations that are models of each of its formulae.

Then let us define the $LP_m$ measure of inconsistency, noted $I_{LP_m}$, as:

**Definition 11.** Let $\Psi$ be a profile. $I_{LP_m} = \frac{\min_{\omega \in \text{Mod}_{LP}(\Psi)}(|\omega!|)}{|P|}$

So the inconsistency measure of a belief base is defined as the minimum number of variables (out of the total number of variables) that are touched by an inconsistency in the $LP_m$ models of this base. That intuitively means that the inconsistency measure of a base is expressed as how inconsistent is the least inconsistent model of this base.

**Example 2.** $\Psi_4 = \{a, \neg a, b \land c, \neg b\}$. $I_{LP_m}(\Psi_4) = \frac{2}{3}$

So inconsistency measures based on variables like this one allow to finely describe the amount of conflict in one base (or profile), but they are unable to take into account the distribution of the contradiction among formulae. In fact the inconsistency measure would be exactly the same with $\Psi'_4 = \{a \land \neg a \land b \land \neg b \land c\}$. This is a real problem for the intended use of those inconsistency values for Belief Game Models, since we want to be able to know what is the part of the inconsistency brought by each formula (agent). To this aim we will use a notion coming from game theory.

### 4.2 Games in Coalitional Form - Shapley Value

In this section we give the definitions of games in coalitional form and of the Shapley value.

**Definition 12.** Let $N = \{1, \ldots, n\}$ be a set of $n$ players. A game in coalitional form is given by a function $v : 2^N \rightarrow IR$, with $v(\emptyset) = 0$.

This framework defines games in a very abstract way, focusing on the possible coalitions formations. A coalition is just a subset of $N$. This function gives what payoff can be achieved by each coalition in the game $v$ when all its members
act together as a unit. The problem is how this utility can be shared among the players\(^2\). Let us explain this on an example.

**Example 3.** Let \( N = \{1, 2, 3\} \), and let \( v \) be the following coalitional game:

\[
\begin{align*}
v(\{1\}) &= 1 & v(\{2\}) &= 0 & v(\{3\}) &= 1 \\
v(\{1, 2\}) &= 10 & v(\{1, 3\}) &= 4 & v(\{2, 3\}) &= 11 \\
v(\{1, 2, 3\}) &= 12
\end{align*}
\]

The grand coalition can bring 12 to the three players. This is the highest utility achievable by the group. But this is not the main aim for all the players. In particular one can note that two coalitions can bring nearly as much, namely \( \{1, 2\} \) and \( \{2, 3\} \) that gives respectively 10 and 11, that will have to be shared only between 2 players. So it is far from certain that the grand coalition will form in this case. Another remark on this game is that all the players do not share the same situation. In particular player 2 is always of a great value for any coalition she joins. So she seems to be able to expect more from this game than the other players. For example she can make an offer to player 3 for making the coalition \( \{2, 3\} \), that brings 11, that will be split in 8 for player 2 and 3 for player 3. As it will be hard for player 3 to win more than that, 3 will be tempted to accept.

A solution concept has to take into account these kinds of arguments. It means that one wants to **solve** this game by stating what is the payoff that is “due” to each agent. That requires to be able to quantify the payoff that an agent can claim with respect to the power that her position in the game offers.

**Definition 13.** A **value** is a function that assigns to each game \( v \) a vector of payoff \( S(v) = (S_1, \ldots, S_n) \) in \( \mathbb{R}^n \).

This function gives the payoff that can be expected by each player \( i \) for the game \( v \), i.e. it measures \( i \)'s power in the game \( v \).

Shapley proposes a solution to this problem that can be explained as follows: considering that the coalitions form according to some order (a first player enters the coalition, then another one, then a third one, etc), and that the payoff attached to a player is its marginal utility (i.e. the utility that it brings to the existing coalition), so if \( C \) is a coalition (subset of \( N \)) not containing \( i \), player’s \( i \) marginal utility is \( v(C \cup \{i\}) - v(C) \). As one can not make any hypothesis on which order is the correct one, suppose that each order is equally probable. This leads to the following formula:

\[
S_i(v) = \sum_{C \subseteq N} \frac{(c-1)!(n-c)!}{n!} (v(C) - v(C \setminus \{i\}))
\]

where \( c \) is the cardinality of \( C \).

---

\(^2\) One supposes the transferable utility (TU) assumption, i.e. the utility is a common unit between the players and sharable as needed (roughly, one can see this utility as some kind of money).
Example 4. The Shapley value of the game defined in Example 3 is \((17/27, 35/27, 20/27)\).

These values show that it is player 2 that is the best placed in this game, accordingly to what we explained when we presented Example 3.

4.3 Inconsistency Values using Shapley Value

Given an inconsistency measure, the idea is to take it as the payoff function defining a game in coalitional form, and then using the Shapley value to compute the part of the inconsistency that can be imputed to each base of the belief profile [17].

This allows us to combine the power of inconsistency measures based on variables (like Coherence measure in [12], or like the test action values of [13]), and the use of the Shapley value for knowing what is the responsibility of a given formula in the inconsistency of the base (or the responsibility of a given base in the inconsistency of the profile).

We just require some basic properties on the underlying inconsistency measure.

Definition 14. An inconsistency measure \(I\) is called a basic inconsistency measure if it satisfies the following properties,
\[
\begin{align*}
\bullet & \quad I(\varphi) = 0 \text{ iff } \varphi \text{ is consistent} \quad \quad \text{(Consistency)} \\
\bullet & \quad 0 \leq I(\varphi) \leq 1 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \q
Note that this SIV gives a value for each base of the profile $\Psi$, so if one considers the base $\Psi$ as the vector $\Psi = (\varphi_1, \ldots, \varphi_n)$, then we will use $S_I(\Psi)$ to denote the vector of corresponding SIVs, i.e.

$$S_I(\Psi) = (S_I^\varphi(\varphi_1), \ldots, S_I^\varphi(\varphi_n))$$

Let us see this on Example 1.

**Example 5.** There are three agents $\Psi = \{\varphi_1, \varphi_2, \varphi_3\}$ with the following belief bases:

$$\varphi_1 = \{\neg b \land (a \lor c)\}, \varphi_2 = \{\neg a \land b \land \neg c\} \lor (\neg a \land \neg b \land c)\}, \varphi_3 = \{a \land b \land c\}.$$

Then $I_{LP_m}(\varphi_1) = I_{LP_m}(\varphi_2) = I_{LP_m}(\varphi_3) = I_{LP_m}(\{\varphi_1, \varphi_2\}) = 0$, $I_{LP_m}(\{\varphi_1, \varphi_3\}) = \frac{1}{3}$, $I_{LP_m}(\{\varphi_2, \varphi_3\}) = \frac{2}{3}$, $I_{LP_m}(\{\varphi_1, \varphi_2, \varphi_3\}) = \frac{3}{3} = 1$.

So $S_{I_{LP_m}}(\varphi_1) = \frac{1}{18}$, $S_{I_{LP_m}}(\varphi_2) = \frac{4}{18}$, and $S_{I_{LP_m}}(\varphi_3) = \frac{7}{18}$.

So, according to this Shapley inconsistency value, this is agent $\varphi_3$ that brings the most conflicts in the group (profile), and $\varphi_1$ that is the less problematic agent.

To the best of our knowledge Shapley Inconsistency Values are the only inconsistency measures that allow to discriminate finely between inconsistencies by looking at the proportion of the language touched by the inconsistencies while describing the distribution of the conflicts among the different bases/formulas.

See [17] for more details on the properties of these inconsistency values. Let us see now how to use this idea to define BGM operators.

## 5 Shapley Belief Game Model

So the idea is to define Belief Game Model with a Shapley inconsistency measure as choice function, in order to have a more sensible way to select the bases that have to be weaken.

A Shapley inconsistency measure says how much of the global conflict is brought by each agent. So the choice function selects the more conflictual agents, and the bases of those agents are weaken with a given weakening function.

**Definition 16.** A **Shapley Belief Game Model** is a Belief Game Model $N = \langle S_I, \triangledown \rangle$, where $S_I$ is a Shapley Inconsistency Value.

The solution to a belief profile $\Psi$ for a Shapley Belief Game Model $N = \langle S_I, \triangledown \rangle$ under the integrity constraints $\mu$, noted $N_\mu(\Psi)$, is the belief profile $\Psi^\mu_N$ defined as:

- $\Psi_0 = \Psi$
- $\Psi_{i+1} = \triangledown_{\text{argmax}(S_I(\Psi_i))}(\Psi_i)$
- $\Psi^\mu_N$ is the first $\Psi_i$ that is consistent with $\mu$

So a Shapley Belief Game Model is defined once one has chosen a weakening function and a (basic) inconsistency measure.

Let us see an example of Shapley Belief Game Model.
Example 6. Consider the Shapley Belief Game Model $N = \langle S_{IP_{LP}}, \nabla_{S} \rangle$. There are three agents $\Psi = \{ \varphi_1, \varphi_2, \varphi_3 \}$ with the following belief bases $\varphi_1 = \{ \neg b \land (a \lor c) \}$, $\varphi_2 = \{ \neg a \land b \land \neg c \} \lor (\neg a \land \neg b \land c) \}$, $\varphi_3 = \{ a \land b \land c \}$.

As computed in Example 5, we have $S_{IP_{LP}}(\varphi_1) = \frac{1}{18}$, $S_{IP_{LP}}(\varphi_2) = \frac{1}{18}$, and $S_{IP_{LP}}(\varphi_3) = \frac{2}{18}$. The maximal value is $\frac{2}{18}$, that is $S_{IP_{LP}}(\varphi_3)$, so $\varphi_3$ is the agent that brings the more conflicts, and so it is selected by the choice function for weakening. So $\varphi_3$ is replaced by $\varphi_{3,1} = \nabla_{S}(\varphi_3) = \text{form}(\{(1,1,1), (1,1,0), (1,0,1), (0,1,1)\})$. We have not yet reached a consistent profile, so we must do a further round. Then the new computations of inconsistency values give: $S_{IP_{LP}}(\varphi_1) = 0$, $S_{IP_{LP}}(\varphi_2) = \frac{1}{9}$, and $S_{IP_{LP}}(\varphi_{3,1}) = \frac{1}{9}$. So here the two most problematic bases are $\varphi_2$ and $\varphi_{3,1}$, so they are weakened. $\varphi_2$ is replaced by $\varphi_{2,1} = \nabla_{S}(\varphi_2) = \text{form}(\{(0,1,0), (0,0,1), (1,1,0), (0,0,0), (0,1,1), (1,0,1)\})$, and $\varphi_{3,1}$ is replaced by $\varphi_{3,2} = \nabla_{S}(\varphi_{3,1}) = \text{form}(\{(1,1,1), (1,1,0), (1,0,1), (0,1,1), (0,0,1), (1,0,0), (0,0,1)\})$. We have reached a consistent belief profile, so the result is $\Psi' = \{ \varphi_1, \varphi_2, \varphi_{3,2} \}$.

6 Conclusion

In this paper, we proposed to improve Belief Game Models by using a more natural and suitable definition of choice functions. These choice functions are based on an inconsistency measure that allows to determine what part of the conflicts of the group is bringing by each agent. To this aim we use the Shapley value to be able to adequately define this measure. So Shapley Belief Game Model operators can be described as operators that, iteratively, select the bases that are the more conflictual, and weaken them, until an agreement (consensus) is reached.

This work is a first step to the study of conciliation operators and of Shapley Belief Game Models. Among the required extensions is the need of a more formal definition of conciliation operators as abstract negotiation operators. What are for instance the logical properties that are common to those operators? What are the specific properties of Shapley Belief Game Models? Another interesting question is to know if Shapley Belief Game Models are implementable, or approximable, by a negotiation protocol.

References

Modal tableaux for verifying security protocols

Mehmet A. Orgun¹, Guido Governatori² and Chuchang Liu³

¹ Department of Computing, Macquarie University, Sydney, NSW 2109, Australia
² School of ITEE, The University of Queensland, Brisbane, QLD 4072, Australia
³ Information Networks Division, DSTO, Edinburgh, SA 5111, Australia

Abstract. To develop theories to specify and reason about various aspects of multi-agent systems, many researchers have proposed the use of modal logics such as belief logics, logics of knowledge, and logics of norms. As multi-agent systems operate in dynamic environments, there is also a need to model the evolution of multi-agent systems through time. In order to introduce a temporal dimension to a belief logic, we combine it with a linear-time temporal logic using a powerful technique called fibering for combining logics. We describe a labelled modal tableau system for a fibred belief logic (FL) which can be used to automatically verify correctness of inter-agent stream authentication protocols. With the resulting fibred belief logic and its associated modal tableaux, one is able to build theories of trust for the description of, and reasoning about, multi-agent systems operating in dynamic environments.

1 Introduction

Multi-agent systems (MASs for short) consist of a collection of agents that interact with each other in dynamic and unpredictable environments. Agents communicate with one another by exchanging messages, and they have the ability to cooperate, coordinate and negotiate with each other to achieve their objectives. In order to develop theories to specify and reason about various aspects of multi-agent systems, many researchers have proposed the use of modal logics such as belief logics [4, 6] and logics of knowledge [5, 12]. As multi-agent systems operate in dynamic environments, there is also a need to model the evolution of multi-agent systems through time.

In order to introduce a temporal dimension to a belief logic, Liu et al. [16] have proposed a temporalized logic that provides a logical framework for users to specify the dynamics of trust and model evolving theories of trust for multi-agent systems. However, in this logic there are certain restrictions on the use of temporal and belief operators because of the hierarchical combination of belief and temporal logics used. Temporal operators can never be within the scope of a belief operator, hence we cannot express a statement asserting that some agent believes an event to happen at some time, e.g., the logic does not have a formula such as $B_{john} \text{first holds}(bob,k)$, which could be used to express an assertion that John believes that at the initial time Bob holds the key $k$. Such kind of assertions are often needed, for example, in analysing stream authentication protocols; we therefore consider a more powerful combination technique called fibering [8] that treats temporal operators and belief operators equally.

In this paper, we combine, using the fibering technique, the logic TML, a variant of the modal logic KD of belief [16], with the temporal logic SLTL which is suitable
for specifying events that may run on different clocks (time-lines) of varying rates of progress [15]. We show that in the resulting fibred belief logic (FL) we can specify and reason about not only agent beliefs but also the timing properties of a system effectively. We describe a labelled modal tableaux system for FL which can be used to automatically verify correctness of inter-agent stream authentication protocols. With this logical system one is able to build theories of trust for the description of, and reasoning about, multi-agent systems.

In the rest of the paper, Section 2 introduces the TESLA stream authentication protocol. Section 3 briefly discusses logics SLTL and TML. Section 4 presents the fibring technique as specifically applied for combining TML with SLTL, and provides an axiomatisation for the fibred logic called FL. Section 5 adapts KEM [2, 9], a labelled modal tableaux system to reason with FL. Section 6 develops a theory of trust in FL for specifying the TESLA protocol and discusses its correctness.

2 The TESLA Protocol

Multi-agent systems, typically real world systems, need to employ application specific protocols for transferring data, such as video, audio and sensory data, among agents. Such protocols are often different from the standard class of authentication protocols previously analysed by many researchers using belief logics and/or model checking techniques [4–6]. As an example, we consider the TESLA protocol, a multicast stream authentication protocol of Perrig et al. [19]. In TESLA, authentication is based on the timing of the publication of keys and the indirect relation of each new key to an original key commitment. The process for verifying data packets received to be authentic depends on trust of the receiver in the sender, and belief on whether an intruder can have prior knowledge of a key before it is published by the protocol.

We consider a basic scheme for the TESLA Protocol, called the PCTS scheme, in which each message $M_i$ is sent in a packet $P_i$, along with additional authentication information [3, 19]. The sender issues a signed commitment to a key. The key is only known to the sender. To send message $M_i$, the sender uses that key to compute a MAC (Message Authenticating Code) on a packet $P_i$, and later discloses the key in packet $P_{i+1}$, which enables the receiver (or receivers, when multiple receivers are involved) to verify the commitment and the MAC of packet $P_i$. A successful verification will imply that packet $P_i$ is authenticated and trusted. We assume that, apart from the initial contact messages between the sender and the receiver, for all $i \geq 2$, the packet $P_i$ from the sender to receiver has the standard form $(D_i, MAC(K'_j, D_i))$, where $D_i = (M_i, f(K_{i+1}, K_{i-1}))$, $K'_j = f'(K_i)$ for $j \geq 1$, and $f$ and $f'$ are two different pseudo-random functions.

In analysing the protocol it is assumed that [19]:

- The sender is honest and works correctly, following all requirements of the protocol strictly.
- The receiver accepts packet $P_i$ as authentic only when it believes the key commitment and the MAC of the packet have been successfully verified.
- The intruder has the ability to capture, drop, resend, delay, and alter packets, can access to a fast network with negligible delay, and can perform efficient computations, such as computing a reasonable number of pseudo-random function appli-
cations and MACs with negligible delay. Nonetheless, the intruder cannot invert a
pseudo-random function with non-negligible probability.

The security property for the TESLA protocol we need to guarantee is that the
receiver does not believe any packet $P_i$ to be authenticated unless the $M_i$ it contains was
actually sent by the sender. To prevent any successful attack by an intruder, the receiver
only needs to be sure that all packets $P_i$ arrive safely such that the intruder has no time
to change the message and commitment in $P_i$ and forge the subsequent traffic.

3 Two Logics: SLTL and TML

We now give a brief introduction to the logics SLTL and TML.

3.1 SLTL: Simple Linear-time Temporal Logic

SLTL offers two operators, first and next, which refer to the initial moment and the next
moment in time respectively. The formulas of SLTL are built with the usual formation
rules from standard connectives and quantifiers of classical first order logic, and the
temporal operators first and next.

The collection of moments in time is the set of natural numbers. We define the
global clock as the increasing sequence of natural numbers, i.e., $\langle 0, 1, 2, \ldots \rangle$, and a local
clock is an infinite subsequence of the global clock. Thus, we have

Definition 1 (time models) A time model for the logic SLTL has the form $c = (C, <, v)$,
where $C = \langle t_0, t_1, t_2, \ldots \rangle$ is a clock, $<$ is the usual ordering relation over $C$ and $v$ is an
assignment function giving a value $v(t, q) \in \{\text{true, false}\}$ for any atomic formula $q$ at
time $t$ in $C$.

We write $c, t \models A$ to stand for “$A$ is true at time $t$ in the model $c$”. Then the semantics of
the temporal operators with the notion of satisfaction in SLTL is given as follows:

- $c, t \models \text{first } A$ if $c, t_0 \models A$.
- $c, t \models \text{next } A$ if $c, t_{i+1} \models A$.
- satisfaction in the model $c = (C, <, v)$ is defined as satisfaction at some point on $C$.

A minimal axiomatic system for the propositional temporal logic consists of the fol-
lowing axioms (axiom schemata). We let $\nabla$ stand for first or next.

A0. all classical tautologies.
A1. $\nabla(\text{first } A) \leftrightarrow \text{first } A$.
A2. $\nabla(\neg A) \leftrightarrow \neg(\nabla A)$.
A3. $\nabla(A \land B) \leftrightarrow (\nabla A) \land (\nabla B)$.

Apart from the generic substitution rule, SLTL has two rules of inference defined as
follows:

MP. From $\vdash A$ and $\vdash A \rightarrow B$ infer $\vdash B$ (Modus Ponens)
TG. From $\vdash A$ infer $\vdash \nabla A$ (Temporal Generalisation)

The soundness and completeness of the axiomatisation system for SLTL with respect
the class $\mathcal{C}$ consisting of all local clocks are straightforward [15].

33
3.2 TML: Typed Modal Logic

We assume that there are \( n \) agents \( a_1, \ldots, a_n \) and, correspondingly, \( n \) modal operators \( B_1, \ldots, B_n \) in the logic, where \( B_i \) \((1 \leq i \leq n)\) stands for “agent \( a_i \) believes that”.

We assume the fixed-domain approach to quantification, that is, the domain of quantification is the same in all possible worlds. This means that we have the standard first-order logic semantics for the \( \forall \) and \( \exists \) quantifiers. We also employ rigid denotations for terms, that is, the only dynamic objects are predicates. We also assume that in TML all the wffs built according to the usual formation rules and are correctly typed.

A classical Kripke model [14] for the logic TML is a tuple \( m = \langle S, R_1, \ldots, R_n, \pi \rangle \), where \( S \) is the set of states or possible worlds; and each \( R_i (1 \leq i \leq n) \) is a relation over \( S \), consisting of state pairs \((s, t)\) such that \((s, t) \in R_i\) iff, at state \( s \), agent \( a_i \) considers the state \( t \) possible; and \( \pi \) is the assignment function, which gives a value \( \pi(s, q) \in \{true, false\} \) for any \( s \in S \) and atomic formula \( q \). Each \( R_i \) called the possibility relation according to agent \( a_i \). We write \( m, s \models \varphi \) to stand for “\( \varphi \) is true at the state \( s \) in the model \( m \)” or “\( \varphi \) holds at \( s \) in \( m \)”.

The semantics definition for the belief operators with the notion of satisfaction in TML is given as follows:

- \( m, s \models B_i \varphi \) iff, for all \( t \) such that \((s, t) \in R_i \), \( m, t \models \varphi \).
- A formula \( \varphi \) is satisfiable in a model \( m \) if there exists \( s \in S \) such that \( m, s \models \varphi \).

In preparation for fibering TML with S\( LTL\), we now consider monadic models for TML defined as follows:

**Definition 2 (monadic models)** A monadic model for TML is a structure \( m = \langle S, R_1, \ldots, R_n, \pi, u \rangle \) where \( \langle S, R_1, \ldots, R_n, \pi \rangle \) is a classical model for TML and \( u \in S \) is called the actual world. \( \varphi \) is satisfiable in the monadic model \( m \) if and only if \( m, u \models \varphi \).

We define \( \mathcal{K}_{\text{tml}} \) as a class of monadic models of the form \( \langle S, R_1, \ldots, R_n, \pi, u \rangle \), where

1. \( S = \{ x \mid \exists R_1 \ldots R_n \ u R_1 \circ \ldots \circ R_n x, R_1, \ldots, R_n \in \{ R_1, \ldots, R_n \} \} \),

where \( R_i \circ R_j \) represents the relative product (or composition) of \( R_i \) and \( R_j \). Furthermore, using the notation \( m \) for a model in \( \mathcal{K}_{\text{tml}} \), we write \( m = \langle S^m, R_1^m, \ldots, R_n^m, \pi^m, u^m \rangle \).

In addition we assume:

2. if \( m_1 \neq m_2 \), then \( S^{m_1} \cap S^{m_2} = \emptyset \).
3. \( m_1 = m_2 \) iff \( u^{m_1} = u^{m_2} \).

Assumption (2) indicates that all sets of possible worlds in \( \mathcal{K}_{\text{tml}} \) are all pairwise disjoint, and that there are infinitely many isomorphic (but disjoint) copies of each model; assumption (3) means that a model in \( \mathcal{K}_{\text{tml}} \) can in fact be identified by the actual world in it.

TML has the following axiom schemata and inference rules:

- **B0.** all axioms of the classical first-order logic.
- **B1.** \( B_i (\varphi \rightarrow \psi) \land B_i \varphi \rightarrow B_i \psi \) for all \( i (1 \leq i \leq n) \).
- **B2.** \( B_i (\neg \varphi) \rightarrow \neg (B_i \varphi) \) for all \( i (1 \leq i \leq n) \).
B3. \( \forall x \forall y \forall z (x = y \rightarrow x = z) \) for all \( i \) (1 \( \leq i \leq n \)).

I. \( \forall x \exists y (x = y) \) for all \( i \) (1 \( \leq i \leq n \)).

II. \( \forall x \exists y (x \neq y) \) for all \( i \) (1 \( \leq i \leq n \)).

III. \( \forall x \exists y (x = y) \) for all \( i \) (1 \( \leq i \leq n \)).

The soundness and completeness of the axiomatisation system for TML can be proved in a standard pattern [13].

4 FL: Fibred Logic

In this section, we discuss how the logic FL is obtained through the use of fibring technique for combining the logics TML and SMTL. Let \( \mathcal{O} = \{ B_1, \ldots, B_n, \text{first, next} \} \) be the set of modal connectives of FL. Then the formulas of FL are obtained from the usual formation rules. As before we assume that in FL all the wffs are correctly typed.

The discussion of the fibred semantics in the case of the Kripke monadic models for TML with time models for SLTL can be laid out in three levels: using a single time model, or considering a set of time models with the same clock, or based on different clock models. In this paper we restrict ourselves to the first level. Following Gabbay [8], we define the fibred semantics arising from the Kripke models for TML with a single time model based on simplified fibred models (simply, sfm models) defined as follows:

**Definition 3 (sfm models)** A simplified fibred model or sfm model is a tuple \( \langle W, W_1, W_2, R_0, R_1, \ldots, R_n, \pi, F, w_0 \rangle \) where

1. \( W \) is a set of worlds, \( w_0 \in W_1 \cup W_2 \)
2. \( W_1 \subseteq W, \text{ and } W_2 \) is a set of natural numbers, we also have \( W_1 \subseteq W \).
3. For \( s \in W_1 \), let \( S(s) = \{ x \mid sR_1 \circ \cdots \circ R_n x, \text{ for some } R_1, \ldots, R_n \in \{ R_1, \ldots, R_n \} \} \),
   then \( 1 \) for all \( s \in W_1, S(s) \cap W_2 = \emptyset; \) \( 2 \) for all \( s, r \in W_2, \text{ if } s \neq r, \text{ then } S(s) \cap S(r) = \emptyset; \) and \( 3 \) \( W = ( \bigcup_{s \in W_1} S(s) ) \cup W_2 \).
4. \( R_0 = \{ (x, y) \mid x, y \in W_1 \text{ and } x \neq y \} \) for all \( x, y \in W \).
5. For all \( u \in W_2, \text{ the model } \mathfrak{c} = (C, R_0, \pi^{(u)}) \text{ satisfies the condition that } u \in W_2 \text{ iff u is a time point in the clock } C, \text{ and is in the semantics of SLTL.} \)
6. For all \( u \in W_2, \text{ the model } \mathfrak{m}^{(u)} = (S^{(u)}, R_1 | S^{(u)} \times S^{(u)}, \ldots, R_n | S^{(u)} \times S^{(u)}, u, h | S^{(u)}) \text{ is in the semantics of } \mathcal{X}_{\text{timl}} \text{ of the logic TML.} \)
7. \( F \) is the fibred function consisting of two folds, \( F_0 \) and \( F_1 \). It satisfies the following conditions: \( 1 \) For all connectives \( \forall x \in \mathcal{O} \text{ and all worlds } w \in W, \)
   \[ F(\forall x, w) = \begin{cases} F_0(w) & \text{ if } \forall x \text{ is first or next} \\ F_0(w) & , \text{ otherwise}. \end{cases} \]
   \( 2 \) If \( x \in S^{(u)} \text{ and } u \in W_2, \text{ then } F_0(x) = x \); if \( x \in W_1, \text{ then } F_0(x) = W_2; \) if \( x \in W_2 \), then \( F_1(x) = x; \) and if \( x \notin W_1, \text{ then } F_1(x) = W_2. \)

**Definition 4 (semantics)** The semantics of formulas for the logic FL is defined inductively with respect to an sfm-model \( \langle W, W_1, W_2, R_0, R_1, \ldots, R_n, \pi, F, w_0 \rangle \). For any \( w \in W, \)
1. for any atomic formula \( q \), \( w \models q \) iff \( \pi(w, q) = \text{true} \).
2. \( w \models \neg \varphi \) iff it is not the case that \( w \models \varphi \).
3. \( w \models (\varphi \land \psi) \) iff \( w \models \varphi \) and \( w \models \psi \).
4. \( w \models \forall X \varphi(X) \) iff, for all \( d \in \mathcal{T} \), \( w \models \varphi(d) \), where \( \mathcal{T} \) is the type of \( X \).
5. \( w \models \bigtriangledown \varphi \) iff \( F(\bigtriangledown, w) \models \varphi \).
6. \( w \models \text{first} \ \varphi \) when \( w \in W_i \) iff \( \{ t \mid t \in W_i \} \models \varphi \).
7. \( w \models \text{next} \ \varphi \) when \( w \in W_i \) iff \( \{ t \mid wR_0t \} \models \varphi \).
8. \( w \models B_i \varphi \) when \( w \notin W_i \) and \( 1 \leq i \leq n \) iff, for all \( s \) such that \( wR_0s \), \( s \models \varphi \), assuming \( s \in S^{|m|} \) and \( m \in \mathcal{K}_d \).

With the sfm model \( \langle W, W', W_b, R_0, R_1, \ldots, R_n, \pi, F, w_0 \rangle \) we say that it satisfies the formula \( \varphi \) iff \( w_0 \models \varphi \). Furthermore, \( m = \langle W, W', W_b, R_0, R_1, \ldots, R_n, \pi, F, w_0 \rangle \) is called a regular fibred semantics model for the logic FL. We say \( \varphi \) is valid in the model \( m \), and written as \( m \models \varphi \), if, for all \( w_0 \in W_0 \cup W_b \), the model \( \langle W, W', W_b, R_0, R_1, \ldots, R_n, \pi, F, w_0 \rangle \) satisfies \( \varphi \); we say that \( \varphi \) is satisfied in the model \( m \) if, for some \( w_0 \in W_0 \cup W_b \), the model \( \langle W, W', W_b, R_0, R_1, \ldots, R_n, \pi, F, w_0 \rangle \) satisfies \( \varphi \). Let \( \mathcal{K}_{FL} \) be the set of regular fibred semantics models which defines the fibred logic FL, then we say \( \varphi \) is valid in the logic FL if, for all \( m \in \mathcal{K}_{FL}, m \models \varphi \).

The axiom set of FL consists of the combination of the axioms for SLTL and TML and their inference rules. The soundness for the logic FL depends on the soundness theorems for logics TML and SLTL, and is not difficult to prove; the completeness can be proved by the techniques used in Gabbay [8].

5 Labelled Tableaux for FL

In this section we show how to adapt KEM, a labelled modal tableau system, to reason with FL. The system can be used to automatically check for formal properties of security protocols, in particular for TESLA, in FL.

A tableau system is a semantic based refutation method that systematically tries to build a (counter-)model for a set of formulas. A failed attempt to refute (invalidate) a set of formulas generates a model where the set of formulas is true. To show that a property \( A \) follows from a theory (a protocol) \( B_1, \ldots, B_n \) we verify whether a model for \( \{B_1, \ldots, B_n, \neg A\} \) exists. If it does not then \( A \) is a consequence of the protocol.

In labelled tableau systems, the object language is supplemented by labels meant to represent semantic structures (possible worlds in the case of modal and temporal logics). Thus the formulas of a labelled tableau system are expressions of the form \( A: i \), where \( A \) is a formula of the logic and \( i \) is a label. The intuitive interpretation of \( A: i \) is that \( A \) is true at (the possible world(s) denoted by) \( i \).

KEM is a labelled tableau for logics admitting possible world semantics whose inferential engine is based on a combination of standard tableau linear expansion rules and natural deduction rules supplemented by an analytic version of the cut rule. In addition it utilises a sophisticated but powerful label formalism that enables the logic to deals with a large class of (quantified) modal and non-classical logics. Furthermore the label mechanism corresponds to fibering thus it is possible to define tableau systems for multi-modal logic by a seamless combination of the (sub)tableaux systems for the component logics of the combination.
5.1 Label Formalism

KEM uses Labelled Formulas (L-formulas for short), where an L-formula is an expression of the form $A^i$, where $A$ is a wff of the logic, and $i$ is a label. For FL, we have a type of labels to various modalities for each agent (belief) plus a type of labels for the temporal modalities. The set of atomic labels is

$$\Phi = \Phi_T \cup \bigcup_{i \in \text{Agt}} \Phi^i,$$

where $\Phi_T = \{h_0,t_1,\ldots\}$ and every $\Phi^i$ is partitioned into (non-empty) sets of variables and constants: $\Phi^i = \Phi_V^i \cup \Phi_C^i$, where $\Phi_V^i = \{W_1^i,W_2^i,\ldots\}$ and $\Phi_C^i = \{w_1^i,w_2^i,\ldots\}$. Finally, we add a sets of auxiliary unindexed atomic labels $\Phi^A = \Phi^A \cup \Phi_C^i$ where $\Phi^A = \{W_1,W_2,\ldots\}$ and $\Phi_C^i = \{w_1,w_2,\ldots\}$. $\Phi^A$ will be used in unifications and proofs.

$\Phi_C$ and $\Phi_V$ denote the set of constants and the set of variables. The set of labels is denoted by $\mathfrak{S}$.

Definition 5 (labels) A label is either (i) an element of the set $\Phi_C$, or (ii) an element of the set $\Phi_V$, or (iii) a path term $(u',u)$ where (iiiia) $u' \in \Phi_V \cup \Phi_V$ and (iiib) $u \in \Phi_C$ or $u = (v',v)$ where $(v',v)$ is a label.

As an intuitive explanation, we may think of a label $u \in \Phi_C$ as denoting a world (a given one), and a label $u \in \Phi_V$ as denoting a set of worlds (any world) in some Kripke model. A label $u = (v',v)$ may be viewed as representing a path from $v$ to a (set of) world(s) $v'$ accessible from $v$ (the world(s) denoted by $v'$).

For any label $u = (v',v)$ we shall call $v'$ the head of $u$, $v$ the body of $u$, and denote them by $h(u)$ and $b(u)$ respectively. Notice that these notions are recursive (they correspond to projection functions): if $b(u)$ denotes the body of $u$, then $b(b(u))$ will denote the body of $b(u)$, and so on. We call each of $b(u)$, $b(b(u))$, etc., a segment of $u$. The length of a label $u$, $\ell(u)$, is the number of world-symbols in it, $s^0(u)$ will denote the segment of $u$ of length $n$ and we shall use $h^n(u)$ as an abbreviations for $h(s^n(u))$. Notice that $h(u) = h^{\ell(u)}(u)$. Let $u$ be a label and $u'$ an atomic label. We use $(u';u)$ as a notation for the label $(u';u)'$ if $u' = h(u)$, or for $u$ otherwise.

For any label $u$, $\ell(u) > 1$, we define the counter-segment-$n$ of $u$, as follows (for $n < k < \ell(u)$):

$$c^n(u) = h(u) \times \cdots \times (h^3(u) \times \cdots \times (h^{n+1}(u),w_0)))$$

where $w_0$ is a dummy label, i.e., a label not appearing in $u$ (the context in which such a notion occurs will tell us what $w_0$ stands for). The counter-segment-$n$ defines what remains of a given label after having identified the segment of length $n$ with a ‘dummy’ label $w_0$. The appropriate dummy label will be specified in the applications where such a notion is used. However, it can be viewed also as an independent atomic label in the set of auxiliary labels.

So far we have provided definitions about the structure of the labels without regard of the elements they are made of. The following definitions will be concerned with the type of world symbols occurring in a label.

We say that a label $u$ is $\tau$-preferred iff $\tau = i$ and $h(u) \in \Phi^i$, or $\tau = t$ and $h(u) \in \Phi_T$; a label $u$ is $\tau$-pure iff each segment of $u$ of length $n > 1$ is $\tau$-preferred. With $\mathfrak{S}^i$ we denote the set of $i$-preferred labels where $i \in \text{Ag}$. 


5.2 Label Unifications

One of the key features of KEM is its logic dependent label unification mechanism. In the same way as each modal logic is characterised by a combination of modal axioms (or semantic conditions on the model), KEM defines a unification for each modality and axiom/semantic condition and then combines them in a recursive and modular way. In this case for SLTL we have to provide a characterisation of the two modalities first and next in terms of relationships over labels. In particular we use what we call unification to determine whether the denotation of two labels have a non empty intersection, or in other terms whether two labels can be mapped to the same possible world in the possible worlds semantics.

The second key issue is the ability to split labels and to work with parts of labels. The mechanism permits the encapsulation of operations on sub-labels. This is an important feature that, in the present context, allows us to correlate unifications and fibring functions. Given the modularity of the approach the first step of the construction is to define unifications (pattern matching for labels) corresponding to the single modality in the logic we want to study.

Every unification is built from a basic unification defined in terms of a substitution $\rho : \Phi \mapsto \Sigma$ such that:

$$\rho : \Phi_c \mapsto \Sigma$$
$$\Phi_i \mapsto \Sigma_i \text{ for every } i \in \text{Ag}$$
$$\Phi^t_i \mapsto \Sigma$$

This means that a substitution $\rho$ replaces a constant with the same constant; a variable of type $i$ can be replaced by any $i$-preferred label, while an auxiliary variable can be freely replaced by any label. This is in agreement with the intuitive meaning of labels that a constant stands for a possible world, and a variable stands for a set of possible world (of the appropriate type). Accordingly, we have that two atomic ("world") labels $u$ and $v$ $\sigma$-unify iff there is a substitution $\rho$ such that $\rho(u) = \rho(v)$. We shall use $[u;v]\sigma$ both to indicate that there is a substitution $\rho$ for $u$ and $v$, and the result of the substitution.

The $\sigma$-unification is extended to the case of composite labels (path labels) as follows:

$$[u;v]\sigma = z \text{ iff } \exists \rho : h(z) = \rho(h(u)) = \rho(h(v)) \text{ and } b(z) = [h(u);h(v)]\sigma$$

Clearly $\sigma$ is symmetric, i.e., $[u;v]\sigma$ iff $[v;u]\sigma$. Moreover this definition offers a flexible and powerful mechanism: it allows for an independent computation of the elements of the result of the unification, and variables can be freely renamed without affecting the result of a unification.

We are now ready to introduce the unifications corresponding to the modal operators at hand. For these unification we assume that the labels involved are $\tau$-pure. The first unification is that for first.

$$[u;v]\sigma_{\text{first}} = (t_0;[h^1(u);h^1(v)]\sigma) \text{ iff } h(u) = h(v) = t_0 \text{ and } [h^1(u);h^1(v)]\sigma$$

The unification for first ($\sigma_{\text{first}}$-unification) corresponds to a constant function (the initial time is unique for the model). Accordingly if two labels end with the same atomic label ($t_0$) then the two labels denote the same time instant, namely the start of the clock.
For the unification for \textbf{next} we will use the fact that the time line is a discrete total order, thus two labels denote the same time instant if they have the same length.

\[ [u; v]\sigma_{\text{next}} = u \text{ iff } \ell(u) = \ell(v), \quad [h^1(u); h^1(v)]\sigma \text{ and } c^1(u), c^1(v) \text{ do not contain } t_0. \]

The unification for the logic SLTL is defined by the combination of the unifications for \textbf{first} and \textbf{next}. Formally

\[ [u; v]\sigma_{\text{SLTL}} = \begin{cases} [u; v]\sigma_{\text{first}} & \text{if } u, v \text{ are i-pure, or} \\ [c^m(u); c^m(v)]\sigma_{\text{next}} & [s^m(u); s^m(v)]\sigma_{\text{SLTL}} \end{cases} \]

The belief logic can be understood as the combination of multiple KD modal logics, one for each agent \( i \in \text{Agt} \). Thus we first give the unification for each of such logics and then we combine in a single unification to be used with the unification for SLTL for FL.

\[ [u; v]\sigma_{\text{TML}} = [u; v]\sigma \]

where \( u \) and \( v \) are i-pure. Notice that using the mechanism of counter-segment it is always possible to split labels into pure sub-labels. Accordingly the definition of the unification for TML is

\[ [u; v]\sigma_{\text{TML}} = \begin{cases} [u; v]\sigma_{\text{TMLi}} & u, v \text{ are i-pure, or} \\ [c^m(u); c^m(v)]\sigma_{\text{TMLi}} & c^m(u), c^m(v) \text{ are i-pure, and} \\ w_0 = [s^m(u); s^m(v)]\sigma_{\text{TMLi}} \end{cases} \]

The logic FL is the fibred combination of TML and SLTL, thus according to [9] we can obtain the unification for it based on the unifications for the component logics. With \( \sigma_{\text{BL}} \) we understood either \( \sigma_{\text{SLTL}} \) or \( \sigma_{\text{TML}} \). The unification for FL is:

\[ [u; v]\sigma_{\text{FL}} = \begin{cases} [u; v]\sigma_{\text{BL}} & c^m(u), c^m(v) \text{ are i-pure, and} \\ w_0 = [s^m(u); s^m(v)] \end{cases} \]

Theorem 1: The \( \sigma_{\text{FL}} \)-unification of two labels \( u \) and \( v \) can be computed in linear time.

5.3 Inference Rules

For the presentation of the inference rules we assume familiarity with Smullyan-Fitting unifying notation [7].

\[ \frac{\alpha : u}{\alpha_1 : u} \quad \frac{\alpha : u}{\alpha_2 : u} \quad \frac{\neg (A \lor B) : u}{A : u} \quad \frac{\neg (A \rightarrow B) : u}{A : u} \quad \frac{\neg (A \rightarrow B) : u}{A : u} \quad \frac{\neg (A \rightarrow B) : u}{A : u} \quad (\alpha) \]

The \( \alpha \)-rules are just the familiar linear branch-expansion rules of the tableau method. For the \( \beta \)-rules (formulas behaving disjunctively) we exemplify only the rules for \( \rightarrow \).

\[ \frac{\beta : u}{\beta_i : u} (i = 1, 2) \quad \frac{\beta : u}{\beta_i : u} \quad \frac{\beta : v}{\beta_i : u} \quad \frac{\beta : v}{\beta_i : u} \quad \frac{\beta : v}{\beta_i : u} \quad \frac{\beta : v}{\beta_i : u} \quad (\beta) \]
The β-rules are nothing but natural inference patterns such as Modus Ponens, Modus Tollens and Disjunctive syllogism generalised to the modal case. In order to apply such rules it is required that the labels of the premises unify and the label of the conclusion is the result of their unification.

\[
\begin{array}{c}
\gamma : u \\
\emptyset(x_n) : u \\
(\exists x P(x)) : u \\
\neg(\exists x P(x)) : u \\
\end{array}
\]  
\[
(\gamma)
\]

The γ rules are the usual “universal” rules of tableaux method with the usual proviso that \(x_n\) is a variable not previously occurring in the tree [7, 2].

\[
\begin{array}{c}
\delta : u \\
\emptyset(n_0) : u \\
(\exists x P(x)) : u \\
\neg(\exists x P(x)) : u \\
\end{array}
\]  
\[
(\delta)
\]

The δ rules are the usual “existential” rules of the tableau method, where \(c_n\) is a constant that does not occur previously in the tree.

The rules for B are the normal expansion rule for modal operators of labelled tableaux with free variable. The intuition for the ν rule is that if \(BA\) is true at \(u\), then \(A\) is true at all worlds accessible via \(R_i\) from \(u\), and this is the interpretation of the label \((W_i^n, u)\); similarly if \(BA\) is false at \(u\) (i.e., \(\neg BA\) is true), then there must be a world, let us say \(w_n^u\) accessible from \(u\), where \(\neg A\) is true.

\[
\begin{array}{c}
\text{first} A : u \\
\neg \text{first} A : u \\
\text{next} A : u \\
\neg \text{next} A : u \\
\end{array}
\]  
\[
(\mu_F)
\]

where \(W_i^n\) and \(w_n^u\) are new labels.

The “Principle of Bivalence” represents the semantic counterpart of the cut rule of the sequent calculus (intuitive meaning: a formula \(A\) is either true or false in any given world). PB is a zero-premise inference rule, so in its unrestricted version can be applied whenever we like. However, we impose a restriction on its application. PB can be only applied w.r.t. immediate sub-formulas of unanalysed β-formulas, that is β formulas for which we have no immediate sub-formulas with the appropriate labels in the tree.

\[
\begin{array}{c}
A : u \\
\neg A : u \\
\end{array}
\]  
\[
(PB)
\]

Given the functional interpretation of the temporal accessibility relation and that the initial instant is fixed, we have the same expansion of the labels and there is no need to introduce variables.
5.4 Soundness and Completeness

The resulting tableau system is sound and complete for the logics presented in this paper. As usual with tableau systems a proof of $A$ is a closed tableau for $\neg A$, thus a tableau systems is sound and complete for a particular logic if it is able to generate closed tableaux for all negation of valid formula, and open tableaux (models) for all satisfiable formulas. In proving the results for the logics at hand we will make use of the main result of [9] (Theorem 22) that allows one to obtain a sound and complete labelled tableau system for a fibred logic based on sound and complete labelled tableau systems (of the same type of the tableau system for the fibred logic) for the logics to be combined. The key idea of the Theorem is to conceive the join point of a unification where the labels are split in segments and counter-segments as the counterpart of the fibring function in fibred models.

Theorem 2 $KEM$ is sound and complete for $SLTL$, $TML$ and $FL$.

6 Analysing Authentication Protocols

In this section, we first build a theory of trust to specify the TESLA protocol, then discuss its correctness. With the purpose of making the logic FL appropriate for specifying the protocol, we restrict the time model of FL to guarantee that the time interval between any moment and its next moment in time has the same length, 1 unit time. This restriction matches the special timing property that the TESLA scheme satisfies: the sender sends packets at regular time intervals. The assumption simplifies our discussion without harming its correctness.

6.1 The Formalization of TESLA

We now establish a theory that describes the behaviour or functions of the protocol with the scheme PCTS. The basic types of the theory include: $Agents = \{A,B,S,R,I\}$, $Messages = \{X,Y,D,D'\}$ and $Keys = \{K,K_1,K_2\}$ where $S,R,I$ stand for the sender, the receiver, and the intruder, respectively. In case there are multiple receivers, we may have $R_1,R_2,...$ in the type $Agents$.

Through an analysis of the TESLA protocol, we set a theory to specify it consisting of four modules, $M_{sr}$ (send-receive mode specification), $M_{mk}$ (message receiving and knowledge gained), $M_{ms}$ (message sending), and $M_{ar}$ (authentication rules). Each module consists of several axiom schemata. Several predicates are used to express these axioms. Given the intuitive reading of the predicates we omit their explanations.

Send-receive mode specification depends on what kind of mode is adopted. We first consider a simple mode called the zero-delay mode, which is based on two assumptions: (1) Zero time is spent between sending a message and receiving this message, i.e., the sending time of a packet $P_i$ is equal to the receiving time of the packet on the synchronized receiver’s clock, for any $P_i$; and (2) the packet rate is assumed to be 1 (i.e., 1 packet per unit time). With this mode, module $M_{sr}$ consists of the following axiom schemata:
The first rule says that, if the sender sends the receiver a message, then the receiver will receive the message at the same time; and the second one says that the sender sends the receiver a message packet with a signed commitment to a key if it will send the receiver a packet containing that key at the next moment in time.

Zero-delay mode is an idealized mode. However, generally the time spent between sending and receiving messages cannot be zero. Considering this point, we give the definition of send-receive modes by introducing a generic form.

**Definition 6 (time intervals)** For a send-receive mode, all packets $P_i$ must arrive within a certain time interval $[u, v]$ relative to the current time defined as follows:

\[
sends(S, R, P_i) \rightarrow \text{next}^m \text{receives}(R, P_i), u \leq m \leq v.
\]

Let the current time be $t_c$ (time of sending a packet). Definition 6 indicates that any packet sent by the sender must arrive at a moment between $t_{c+u}$ and $t_{c+v}$.

**Definition 7 (time distance of sending)** Let $d = 1/r$, where $r$ is the packet rate (i.e., number of packets sent per unit time). We call $d$ the time distance of sending between two packets.

Noting that a send-receive mode is in fact determined based on the time interval of packet arrival and the time distance of sending, we have the formal definition of a mode as follows:

**Definition 8 (send-receive modes)** We use the notation $m([u, v], d)$ to represent a send-receive mode of the PCTS scheme of TESLA or, simply, a mode if $u, v, d \in \mathbb{N}$, the set of all natural numbers, and $u \leq v$, where $[u, v]$ is the time interval of this mode, and $d$ the time distance of sending with it. We say that $m([u, v], d)$ is a safe mode if $v < d$.

The following generic rules specify a given mode $m([u, v], d)$:

\[\begin{align*}
G1. & \quad sends(S, R, X) \rightarrow \text{next}^{(i)} \text{receives}(R, X) \lor \ldots \lor \text{next}^{(v)} \text{receives}(R, X). \\
G2. & \quad sends(S, R, \langle D, \text{MAC}(f^i(K), D) \rangle) \rightarrow \text{next}^{(d)} sends(S, R, X) \land K \in X.
\end{align*}\]

Mode-specific rules are determined when $u, v$ and $d$ are given. For example, within the mode $m([2, 3], 4)$, we have

\[\begin{align*}
S1. & \quad sends(S, R, X) \rightarrow \text{next}^{(2)} \text{receives}(R, X) \lor \text{next}^{(3)} \text{receives}(R, X). \\
S2. & \quad sends(S, R, \langle D, \text{MAC}(f^i(K), D) \rangle) \rightarrow \text{next}^{(4)} sends(S, R, X) \land K \in X.
\end{align*}\]

Modules $M_{mk}$, $M_{ms}$, and $M_{tar}$ are fixed for any mode. Due to space limitations, they are listed below without explanations.

**M新模式 (message receiving and knowledge gained)**

\[\text{next}^{m}\] In what follows we will use $\text{next}^{m}$ to indicate $m$ consecutive occurrences of $\text{next}$.
G3. \( \text{receives}(A, (X, Y)) \rightarrow \text{receives}(A, X) \land \text{receives}(A, Y) \).
G4. \( \text{receives}(A, X) \rightarrow \text{knows}(A, X) \).
G5. \( \text{knows}(A, K) \rightarrow \text{knows}(A, f(K)) \land \text{knows}(A, f'(K)) \).
G6. \( \text{knows}(A, \{ X \} SK(B)) \rightarrow \text{knows}(A, X) \).
G7. \( \text{knows}(A, K) \land \text{knows}(A, X) \rightarrow \text{knows}(A, MAC(K, X)) \).
G8. \( \text{knows}(A, X) \rightarrow \text{next} \text{knows}(A, X) \).

where \( SK(B) \) is the private key of agent \( B \) and its corresponding public key can be known by anybody, so we have G8.

\( M_{\text{int}}(\text{Message sending}) \)
G9. \( \text{send}(A, B, (X, Y)) \rightarrow \text{send}(A, B, X) \land \text{send}(A, B, Y) \).
G10. \( \text{send}(A, B, X) \rightarrow \text{has_sent}(A, B, X) \).
G11. \( \text{has_sent}(A, B, X) \rightarrow \text{next} \text{has_sent}(A, B, X) \).

\( M_{\text{au}}(\text{Authentication rules}) \)
G12. \( \text{is_auth}(X, MAC(f'(K), D)) \rightarrow \text{verify_success}(f(K)) \land \text{verify_success}(MAC(f'(K), D)) \).
G13. \( \text{is_auth}(X) \rightarrow \text{has_auth}(X) \).
G14. \( B_R \text{has_auth}(X) \rightarrow \text{next} B_R \text{has_auth}(X) \).
G15. \( \text{receives}(R, (X, MAC(f'(K), D))) \land \text{has_sent}(S, R, X) \rightarrow B_R \text{arrive_safe}(X) \).
G16. \( \text{arrive_safe}(X) \rightarrow \text{has_arrive_safe}(X) \).
G17. \( B_R \text{has_arrive_safe}(X) \rightarrow \text{next} B_R \text{has_arrive_safe}(X) \).
G18. \( B_R \text{verify_success}(f(K)) \rightarrow \text{B_R has_arrive_safe}(X, MAC(f'(K), D)) \land \text{knows}(R, K) \land B_R \text{has_auth}(D', MAC(f'(K), D')) \land f(K) \in D' \).
G19. \( \text{B_R verify_success}(MAC(f'(K), D)) \rightarrow \text{B_R has_arrive_safe}(X, MAC(f'(K), D)) \land \text{knows}(R, K) \land MAC(f'(K), X) = MAC(f'(K), D) \).

Thus, we have obtained a theory \( T = M_{\text{au}} \cup M_{\text{int}} \cup M_{\text{rst}} \cup M_{\text{str}} \) specifying the PCTS scheme of TESLA given in Section 2, where each module contains the relevant axioms given above.

6.2 Correctness Analysis

The correctness condition for a given TESLA scheme should guarantee that if the receiver (receivers) can verify that a packet is authentic, then the packet was indeed sent by the sender.

**Definition 9 (correctness condition)** The local correctness for a TESLA scheme to the receiver \( R \) who receives messages from the sender \( S \) means that, if \( R \) has verified that a packet is authentic, then the packet was indeed sent by \( S \). That is, \( \forall X (B_R \text{has_auth}(X) \land \text{has_sent}(A, R, X) \rightarrow A = S) \). Furthermore, the (global) correctness for the TESLA scheme means that the local correctness for the scheme to all receivers holds.

The theory discussed above is based on a time model where the clock is regarded as the synchronized receiver’s clock (correspondingly to the global clock). It provides a basis for the receiver to verify stream messages received through the PCTS scheme of TESLA if the scheme with its send-recv mode satisfies the correctness condition.

Based on the theory developed above, we can show that the correctness condition of the TESLA protocol holds within the scheme.
**Proposition 1** The PCTS scheme with the mode \( m([u,v], d) \) mode is secure (i.e., it satisfies the correctness condition) if \( m([u,v], d) \) is a safe mode.

We can also use the theory to show that the PCTS scheme with an unsafe mode, e.g., the mode \( m([1,4], 2) \), provides chances for the intruder to attack the system. Consider the case: assume that packets \( P_t \) and \( P_{t+1} \) are sent out by the sender at time \( t \) (the current moment in time) and at \( t + 2 \) (the next next moment), respectively. The intruder, \( I \), first intercepts \( P_t \) at \( t + 2 \) and then, at \( t + 3 \), again intercepts \( P_{t+1} \) when it arrives. By creating a packet \( P'_t \) instead of \( P_t \), using key \( K_t \) in packet \( P_{t+1} \), \( I \) masquerades as the sender send packet \( P'_t \) to the receiver. The attack will be successful if \( P'_t \) reaches the receiver at \( t + 4 \).

### 6.3 Mechanising Correctness Proofs

In order to automatically analyse the correctness of a scheme of the protocol, we need to mechanize the theory describing the behaviour of the protocol in an appropriate proof system. In our approach, such system-specific trust theories developed for specifying communications protocols do not depend on a specific implementation. The user is therefore allowed to freely choose the tools for mechanizing these theories. Below we will show how modal tableaux can be used to verify the properties of the TESLA protocol. Modular structure offers convenience to the user for translating a theory to an executable code (program) in a mechanised proof system, such as Isabelle [18] or the SMV model checker [17].

With the labelled modal tableaux system KEM, to show a safe mode satisfies the correctness condition, we only need to show that in this mode \( A = S \) is a KE consequence of a set of formulas \( \Gamma = \{ \text{B}_R \text{ has been auth}(X), \text{has sent}(A, R, X) \} \). Due to space limitations, we only give a simple case to show how the labelled tableaux system works on checking the properties of TESLA. With the send-receive mode \( m([2,3], 4) \), we assume that the message has arrived safely and it has been authenticated based on the time the message was received and the contents of the message:

\begin{align*}
\text{H1.} & \quad \text{first next}^3 \text{ receives}(R, (X,Y)) \\
\text{H2.} & \quad \text{first next}^7 \text{ receives}(R, X1) \wedge K \in X1 \\
\text{H3.} & \quad \text{MAC}(f^X(K), X) = Y \\
\text{H4.} & \quad \text{first next}^8 \text{ B}_R \text{ is auth}(X,Y) \\
\end{align*}

Then, we can prove the following property:

\begin{align*}
(A). & \quad \text{first next}^8 \text{ B}_R (is auth(X,Y)) \to (\text{first sends}(S, R, (X,Y)) \vee \text{first next sends}(S, R, (X,Y))))
\end{align*}

It basically says that if at time \( t_8 \), agent \( R \) believes that if the message is authenticated, then it must have been sent at either time \( t_0 \) or time \( t_1 \) (agent \( R \) does not really know the exact time when the message was sent, however, it knows about the time interval).

In the following we show the tableaux proof of the property. All the rules of the PCTS scheme of TESLA are at our disposal as well as the assumptions made above; each is labelled with a generic universal label that would unify with any given label. Tableaux rules have been applied exhaustively until all the branches have been completed (details of proof steps are omitted). We also assume a that biconditional (such as \( S_1 \) used in the proof) is the conjunction of two implications.
1. \( \text{ sends}(S,R,(X,Y)) \rightarrow \text{next}(\text{receives}(R,(X,Y))) \lor \text{next}(\text{receives}(R,(X,Y))) : W1 \)
2. \( \text{first next}(\text{receives}(R,(X,Y))) : W2 \)
3. \( \text{first next}(\text{is auth}(X,Y)) : W3 \)
4. \( \text{first next}(\text{is auth}(X,Y)) \rightarrow (\text{first sends}(S,R,(X,Y)) \lor \text{first next sends}(S,R,(X,Y))) : W4 \)
5. \( \text{next}(\text{is auth}(X,Y)) \rightarrow (\text{first sends}(S,R,(X,Y)) \lor \text{first next sends}(S,R,(X,Y))) : (t_0,W4) \)
6. \( \text{next}(\text{is auth}(X,Y)) \rightarrow (\text{first sends}(S,R,(X,Y)) \lor \text{first next sends}(S,R,(X,Y))) : (t_1,(t_0,W4)) \)
7. \( \ldots \) (expansion rule for \text{next} is applied 7 times, \( \mu_T \))
8. \( \neg B_R (\text{is auth}(X,Y)) \rightarrow (\text{first sends}(S,R,(X,Y)) \lor \text{first next sends}(S,R,(X,Y))) : (t_0,(\ldots,(t_1,(t_0,W4))\ldots)) \)
9. \( \neg (\text{is auth}(X,Y)) \rightarrow (\text{first sends}(S,R,(X,Y)) \lor \text{first next sends}(S,R,(X,Y))) : (w',(t_0,(\ldots,(t_1,(t_0,W4))\ldots)) \)
10. \( \text{is auth}(X,Y) : (w',(t_0,(\ldots,(t_1,(t_0,W4))\ldots)) \)
11. \( \neg (\text{first sends}(S,R,(X,Y)) \land \text{first next sends}(S,R,(X,Y))) : (w',(t_0,(\ldots,(t_1,(t_0,W4))\ldots)) \)
12. \( \text{first sends}(S,R,(X,Y)) : (w',(t_0,(\ldots,(t_1,(t_0,W4))\ldots)) \)
13. \( \neg (\text{first next sends}(S,R,(X,Y))) : (w',(t_0,(\ldots,(t_1,(t_0,W4))\ldots)) \)
14. \( \text{first next sends}(S,R,(X,Y)) : (w',(t_0,(\ldots,(t_1,(t_0,W4))\ldots)) \)
15. \( \text{first sends}(S,R,(X,Y)) : (t_0,((w',(t_0,(\ldots,(t_1,(t_0,W4))\ldots))\ldots)) \)
16. \( \text{first sends}(S,R,(X,Y)) : (t_1,(t_0,((w',(t_0,(\ldots,(t_1,(t_0,W4))\ldots))\ldots)) \)
17. \( \text{first sends}(S,R,(X,Y)) \rightarrow \text{next}(\text{receives}(R,(X,Y))) \lor \text{next}(\text{receives}(R,(X,Y))) : W1 \)
18. \( \text{next}(\text{receives}(R,(X,Y))) \lor \text{next}(\text{receives}(R,(X,Y))) \rightarrow \text{first sends}(S,R,(X,Y)) : W1 \)
19. \( \text{next}(\text{receives}(R,(X,Y))) \lor \text{next}(\text{receives}(R,(X,Y))) : W1 \)
20. \( \text{next}(\text{receives}(R,(X,Y))) : (t_1,(t_0,((w',(t_0,(\ldots,(t_1,(t_0,W4))\ldots))\ldots)) \)
21. \( \text{next}(\text{receives}(R,(X,Y))) : (t_1,(t_0,((w',(t_0,(\ldots,(t_1,(t_0,W4))\ldots))\ldots)) \)
22. \( \text{next}(\text{receives}(R,(X,Y))) : (t_0,W2) \)
23. \( \text{next}(\text{receives}(R,(X,Y))) : (t_1,(t_0,W2)) \)
24. \( \times ([t_1,(t_0,((w',(t_0,(\ldots,(t_1,(t_0,W4))\ldots))\ldots))\ldots)) \) and \( (t_1,(t_0,W2)) \) unify]

This proof has only one branch which is closed. This shows that agent R’s belief has been justified based on the assumptions.

7 Concluding Remarks

With the logic FL, we use a simple case of the fibred semantics arising from Kripke models with a single time model. However, it is not difficult to extend it with other different time models. Such extensions would be needed when one wants to deal with different local clocks for multi-agent systems.

We have discussed an application of the logic FL in analysing the TESLA protocol. Archer [1] uses the theorem prover TAME, and Broadfoot et al [3] use model checking techniques, to analyse TESLA. The advantage of these methods is that some properties of the protocol can easily be captured through proof systems, but a drawback is that the formal representations involved in such proofs are often not easily validated by the user. Our approach separates the theory from its implementation and helps a protocol designer to capture the meanings of the theory as a whole. Our analysis has shown that the PCTS scheme of TESLA with a safe send- is secure given that the correctness condition is satisfied. We believe that this approach can be easily extended for the analysis of other schemes of the TESLA protocol, and for other security protocols.
Acknowledgements

This work is supported in part by Australia Research Council under Discovery Project No. DP0452628 on “Combining modal logics for dynamic and multi-agents systems”.

References

Modelling Emotional BDI Agents

David Pereira¹, Eugénio Oliveira², and Nelma Moreira¹

¹ DCC – FCUP & LIACC, Rua do Campo Alegre, 823, 4150-180 Porto, Portugal
{dperreira,nam}@dcc.up.pt
² NIAD&R – FEUP & LIACC, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
eco@fe.up.pt

Abstract. Emotional-BDI agents are agents whose behaviour is guided not only by beliefs, desires and intentions, but also by the role of emotions in reasoning and decision-making. In this paper we introduce the logic \( \mathcal{E}_{BDI} \) for specifying Emotional-BDI agents in general and a special kind of Emotional-BDI agent under the effect of fear. The focus of this work is in the expressiveness of \( \mathcal{E}_{BDI} \) and on using it to establish some properties which agents under the effect of an emotion should exhibit.

1 Introduction

Emotional-BDI agency describes computational agents whose behaviour is guided by the interactions existing between beliefs, desires and intentions, along the lines of the classical BDI architecture [1], but where these interactions are influenced by an additional emotional component [2]. This component produces data which will bound the BDI interaction by imposing some of the set of positive aspects that emotions play in reasoning and decision-making [3].

The conceptual architecture which defines the Emotional-BDI model of agency was recently introduced in [2] and is mainly based on recent works of Oliveira & Sarmiento’s about an emotional agents architecture [4], although adapted to fit in the original BDI architecture [1, 5, 6].

In this paper we introduce \( \mathcal{E}_{BDI} \), a multi-modal logic for specifying Emotional-BDI agents. We define the various axioms which properly characterise each of the modal operators of \( \mathcal{E}_{BDI} \) and after we give the specification of the basic Emotional-BDI agent and a specification of a fearful Emotional-BDI agent.

This paper is organised as follows: in Section 2 we provide the motivation for the current work; in Section 3 we introduce the logic \( \mathcal{E}_{BDI} \) and define its syntax and semantics, together with the axioms for the modal operators; in Section 4 we present the specification of a basic Emotional-BDI agent and a fearful Emotional-BDI agent. Finally, in Section 5 we refer related work and in Section 6 we draw some conclusions and point the path to current and future work.

2 Motivation

The main motivation for the current work was to provide a formal system in which the concepts of the Emotional-BDI model of agency could be logically ex-
pressed. Using these concepts, we can build distinct specifications of Emotional-BDI agents which describe the behaviours which are expected from the agents under the influence of emotions. The existing formal systems, namely BDICTL [6] and the KARO [7,8] framework, if used independently, are not suited for our goals. However, both have properties which we need to combine in order to properly model Emotional-BDI agents. Plus, we integrate some important concepts of Oliveira & Sarmiento’s emotional agent architecture [4], which were mapped into abstract concepts for fitting the structure of $\mathcal{E}_BDI$’s syntax.

3 The Logic $\mathcal{E}_BDI$

We will now introduce the logic $\mathcal{E}_BDI$. We first give a resumed informal description of the purpose of each of its components and afterwards we provide its syntax and semantics.

3.1 Informal semantics

The logical structure which supports $\mathcal{E}_BDI$ is a two dimensional structure introduced by Schild [9], which is a simplified approach to Rao & Georgeff’s BDICTL [10] semantics. One dimension is a set of possible worlds that corresponds to the different perspectives of the agent, such as its beliefs, desires, etc. The other is a set of temporal states which describe the temporal evolution of the agent. We call a pair ($\text{world}, \text{temporal\_state}$) a situation.

In $\mathcal{E}_BDI$, as in the KARO framework, we consider explicit complex actions. Actions can be either atomic or regular: the first are actions which cannot be sub-divided into a combination of smaller ones, while regular actions are constructions of atomic actions through a set of regular rules. Actions are a labelling of the temporal structure underlying $\mathcal{E}_BDI$.

In order to properly execute any action, we need the notion of capability (abstract plan) already studied in [11,7] and also the explicitly notion of resource. We use these to specify under which conditions the agent is able to effectively execute any action.

Finally, we introduce the concepts of fear and fundamental desire. The first refer to fearing something or being fearful that, and brings concepts into objects of fear in $\mathcal{E}_BDI$. To properly establish the notion of fear, we require to have special information in which are described the vital desires of an agent, like, for instance, to be alive. The notion of fundamental desire plays such a role. Although it is a desire, a fundamental desire has special properties which guarantee the existence of the agent in an environment.

3.2 Syntax

We now define the language of $\mathcal{E}_BDI$ which extends Rao & Georgeff’s BDICTL [10] for containing explicit actions, capabilities, resources and modal operators representing fear and fundamental desires. This language distinguishes between state-
formulas (which are evaluated in a given situation) and path-formulas (which are evaluated along a given temporal path).

**Definition 1.** Given an infinite numerable set \( P = \{ p, q, p_1, \ldots \} \) of propositional variables and an infinite numerable set of atomic actions \( A_{At} = \{ a, b, a_1, \ldots \} \), the set of \( \mathcal{E}_{BDI} \) well-formed formulas is defined by the following BNF-grammar:

- **State-formulas (SF):**
  \[ \phi_s ::= p | \neg \phi_s | \phi_s \land \phi_s | \langle \alpha \rangle \phi_s | \langle \alpha \rangle \phi_s | \text{EF}_p | \text{AE}_s | \text{BEL}(\phi_s) | \text{DES}(\phi_s) | \text{INT}(\phi_s) | \text{FEAR}(\phi_s) | \text{FDES}(\phi_s) | \text{CAP}(\alpha) | \text{RES}(\alpha) \]

- **Path-formulas (PF):**
  \[ \varphi_p ::= X(\phi_s) | \phi_s \text{U} \phi_s \]

- **Regular-actions (A_{At}):**
  \[ \alpha ::= \text{id} | a_i | \alpha; \alpha | \alpha + \alpha | \alpha^\ast \]

In addition, we introduce the following abbreviations: \( \top, \bot, \phi \lor \psi \) and \( \phi \rightarrow \psi \) are abbreviations of \( \neg (p \land \neg p) \) (with \( p \) being a fixed element of \( P \)), \( \neg \top, \neg (\neg \neg \phi \land \neg \psi) \) and \( \neg (\phi \land \neg \psi) \), respectively; \( \text{AF}_\varphi, \text{EF}_\varphi, \text{AG}_\varphi \) and \( \text{EG}_\varphi \) are abbreviations of \( \text{A(} \text{TU}_\varphi \text{)}, \text{E(} \text{TU}_\varphi \text{)} \), \( \neg \text{EF}_\varphi \) and \( \neg \text{AF}_\varphi \), respectively. Iterated actions \( \alpha^n \), with \( n \geq 0 \), are inductively defined by \( \alpha^0 = \text{id} \) and \( \alpha^{n+1} = \alpha; \alpha^n \).

### 3.3 Semantics

In this section we introduce the semantics of \( \mathcal{E}_{BDI} \). We start by defining the notion of situation.

**Definition 2.** Given a non-empty set \( W = \{ w_0, w_1, w_2, \ldots \} \) of worlds (also known as agent’s perspectives or scenarios), and a non-empty set \( S = \{ s_0, s_1, s_2, \ldots \} \) of temporal states (also known as time points), a situation is a pair \( \sigma = (w_i, s_j) \), with \( i \geq 0 \) and \( j \geq 0 \). The set of situations is denoted by \( \Sigma \), which verifies \( \Sigma \neq \emptyset \) and \( \Sigma \subseteq W \times S \).

Situations define particular temporal states, in scenarios that the agent has information about. For instance, in a situation \( \langle \text{desire}, s \rangle \) the desire of winning the lottery may be considered as true, although in the same temporal state, lets say in the situation \( \langle \text{believe}, s \rangle \), the agent may not believe in it. However, at some temporal state \( s' \) both may be considered true by the agent.

Given a set of situations \( \Sigma \) we can map the evolution of time and action execution by defining two relations. One is a branching time relation \( R_T \) and the other is an action execution relation that associates to each element of \( R_T \) an atomic action.

**Definition 3.** Given a non-empty set of situations \( \Sigma \) we define the relation \( R_T \) as follows:
1. It is serial, i.e., \( \forall \sigma \in \Sigma, \exists \sigma' \in \Sigma \) such that \( (\sigma, \sigma') \in R_T \);
2. If \((w_i, s_i), (w_k, s_i)\) \(\in R_T\) then \(w_i = w_k\).

Only imposing that \(R_T\) is only serial, and not a total (linear) order, leads to a branching-time structure.

**Definition 4.** Given a set of atomic actions \(A_{At}\) and a branching time relation \(R_T\), for \(a_i \in A_{At}\) we define an action execution relation \(R_{a_i}\), such that:

1. \(R_{a_i} \in R_T\);
2. If \((\sigma, \sigma') \in R_{a_i}\), then it is false that exists \(a_j \in A_{At}\) such that \(i \neq j\) and \((\sigma, \sigma') \in R_{a_j}\).

The previous relation can be extended to regular actions, as follows.

**Definition 5.** Given a regular action \(\alpha\) and a set of situations \(\Sigma\), we inductively define the regular action accessibility relation by:

\[
\begin{align*}
R^\alpha & : A_{R_{a}} \rightarrow (\Sigma \times \Sigma) \\
R^\alpha(a_i) & = \{ (\sigma, \sigma') \mid (\sigma, \sigma') \in R_{a_i} \} \\
R^\alpha(id) & = \{ (\sigma, \sigma') \mid \sigma = \sigma' \} \\
R^\alpha(\alpha;\beta) & = \{ (\sigma, \sigma') \mid \exists \sigma'' \in \Sigma((\sigma, \sigma'') \in R^\alpha(\alpha) \land (\sigma'', \sigma') \in R^\alpha(\beta)) \} \\
R^\alpha(\alpha;\beta) & = \{ (\sigma, \sigma') \mid (\sigma, \sigma') \in R^\alpha(\alpha) \lor (\sigma, \sigma') \in R^\alpha(\beta) \} \\
R^\alpha(\alpha^n) & = \{ (\sigma, \sigma') \mid (\sigma, \sigma') \in R^\alpha(id) \} \\
R^\alpha(\alpha^{n+1}) & = \{ (\sigma, \sigma') \mid (\sigma, \sigma') \in R^\alpha(\alpha;\alpha^n) \} \\
R^\alpha(\alpha^*) & = \{ (\sigma, \sigma') \mid \exists n \in \mathbb{N}((\sigma, \sigma') \in R^\alpha(\alpha^n)) \}
\end{align*}
\]

The main interest behind using both approaches is mainly guided by the properties which emotions exhibit. The emotions can be triggered either by an action which will lead to some wanted/unwanted situation or triggered by believing that a situation may or will inevitably be true in the future.

The distinction, in the syntax, between path formulas and state formulas must reflect also in the semantics. In \(\mathcal{E}_{BDI}\), as in \(\mathcal{E}_{BDI_{CTL}}\), the former are analysed along a path (a time branch) and the second in a particular situation. In \(\mathcal{E}_{BDI}\), a path is defined as follows:

**Definition 6.** Let \(\Sigma\) be a set of situations and \(R_T\) a branching time relation defined on \(\Sigma\). A path is a subset \(\pi \sigma = (\sigma_0, \sigma_1, \sigma_2, \ldots)\) such that \(\sigma = \sigma_0\) and \(\forall i \geq 0, (\sigma_i, \sigma_{i+1}) \in R_T\). The \(k^{th}\) element of a path \(\pi \sigma\) is denoted as \(\pi \sigma[k]\).

We already saw that we can analyse the several perspectives the agent may be aware of at the same state. For that we have to vary the world component of any situation \(\langle \text{world, temporal\_state} \rangle\). The accessibility relations which establish this relationship are the ones which are going to be used for modelling the
mental states of the agent. These relations are denoted by $R^O$, with $O$ belonging to a set of modal operators and that must respect the following condition: if $\langle (w_i, s_i), (w_k, s_k) \rangle \in R^O$ then $s_j = s_i$.

Finally, we also have to provide a semantic interpretation for capabilities and resources. We mainly follow the ideas of modelling capabilities in the KARO framework, i.e., by considering local functions in each situation which establish which atomic actions the agent has capabilities/resources to execute properly. The capabilities/resources for regular actions are interpreted by relating these local functions to regular action accessibility relations, in the following way.

**Definition 7.** Given a regular action $a$, a set of situations $\Sigma$ and a function $\nu_f(a_i)$ which establishes a subset of $\Sigma$ where the agent has capabilities/resources to execute atomic actions $a_i \in A_{at}$, resources and capabilities are interpreted by similar functions. Therefore, we inductively define them in a function $f$, with $f \in \{c, r\}$, such that:

\[
\begin{align*}
& f^A : A_{at} \to \wp(\Sigma) \\
& f^A(a_i) = \nu_f(a_i) \\
& f^A(id) = \Sigma \\
& f^A(\alpha; \beta) = \{ \sigma | \sigma \in f^A(\alpha) \land \exists \sigma' \in \Sigma((\sigma, \sigma') \in R^A(\alpha) \land \sigma' \in f^A(\beta)) \} \\
& f^A(\alpha + \beta) = \{ \sigma | \sigma \in f^A(\alpha) \lor \sigma \in f^A(\beta) \} \\
& f^A(\alpha^0) = \{ \sigma | \sigma \in f^A(id) \} \\
& f^A(\alpha^{n+1}) = \{ \sigma | \sigma \in f^A(\alpha^n) \} \\
& f^A(\alpha^*) = \{ \sigma | \exists n \in \mathbb{N}(\sigma \in f^A(\alpha^n)) \}
\end{align*}
\]

The interpretation of $\mathcal{E}_{BDI}$-formulae is done over Kripke-models, as defined below.

**Definition 8.** Given a set of worlds $W$, a set of temporal states $S$, a set of propositional variables $P$, a set of atomic actions $A_{at}$ and a set of modal operators $Op = \{\text{BEL}, \text{DES}, \text{INT}, \text{FDES}, \text{FEAR}\}$, we define an $\mathcal{E}_{BDI}$-model as a tuple:

\[ M = \langle \Sigma, R_T, \{R_a : a \in A_{at}\}, R^A, \{R^O : O \in Op\}, c^A, r^A, v_p, v_c, v_r \rangle \]

where

- $\Sigma$ is the set of situations;
- $R_T$ is a branching time relation on $\Sigma$;
- each $R_a$ is an atomic action accessibility relation on $\Sigma$;
- $R^A$ is a accessibility relation for regular actions;
- $R^O$ are accessibility relations for the corresponding modal operators;
- $v_p$, $v_c$ and $v_r$ are functions which define in which states the propositions hold, the capabilities for atomic actions hold and the resources for atomic actions hold, respectively.

The satisfiability of a well-formed formula in $\mathcal{E}_{BDI}$ is given by the following definition.
Definition 9. Let \( M \) be an \( \mathcal{E}_{\text{BDI}} \)-model. The satisfiability of an \( \mathcal{E}_{\text{BDI}} \)-formula with respect to \( M \) and a situation \( \sigma \in \Sigma \) is inductively defined as follows, considering \( O \in O_p \):

- **satisfaction for state-formulas:**
  1. \( M, \sigma \models p \) iff \( p \in \nu_p(\sigma) \)
  2. \( M, \sigma \models \neg \varphi \) iff \( M, \sigma \models \varphi \)
  3. \( M, \sigma \models \varphi \land \psi \) iff \( M, \sigma \models \varphi \) and \( M, \sigma \models \psi \)
  4. \( M, \sigma \models \exists \pi \varphi \) such that \( M, \pi \sigma \models \psi \)
  5. \( M, \sigma \models \exists \pi \sigma \models \psi \)
  6. \( M, \sigma \models (\alpha) \varphi \) iff \( \exists (\sigma, \sigma') \in R^k(\alpha) \) such that \( M, \sigma' \models \varphi \)
  7. \( M, \sigma \models (\alpha) \varphi \lor (\sigma, \sigma') \in R^k(\alpha), M, \sigma' \models \varphi \)
  8. \( M, \sigma \models \mathcal{O}(\omega) \) iff \( \forall (\sigma, \sigma') \in R^0, M, \sigma' \models \varphi \)
  9. \( M, \sigma \models \mathcal{C}(\omega) \) iff \( \sigma \in c^1(\alpha) \)
  10. \( M, \sigma \models \mathcal{R}(\alpha) \) iff \( \sigma \in r^0(\alpha) \)

- **satisfaction for path-formulas:**
  1. \( M, \pi \sigma \models X \varphi \) iff \( M, \pi \sigma[1] \models \varphi \)
  2. \( M, \pi \sigma \models \varphi_2 \) iff \( \exists k \geq 0 \text{ such that } M, \pi \sigma[k] \models \varphi_2 \) and \( \forall j, 0 \leq j < k, M, \pi \sigma[j] \models \varphi_1 \)

If \( M, \sigma \models \varphi \) in all \( \mathcal{E}_{\text{BDI}} \)-models \( M \) and situations \( \sigma \in \Sigma \), then \( \varphi \) is valid. If it is the case that \( M, \sigma \models \varphi \) only for some \( M \) and \( \sigma \), then \( \varphi \) is satisfiable in \( M \) and situation \( \sigma \).

**Properties of time** The temporal layer of \( \mathcal{E}_{\text{BDI}} \) corresponds to CTL logic \([10]\). Therefore, we have the formulas \( \mathcal{A} \psi \) and \( \mathcal{E} \psi \) which assert that \( \psi \) holds over all paths, and at least in one of them, respectively. For reasoning about the properties of a particular path, we have the formulas \( \varphi \mathcal{U} \psi \) and \( \chi \). These express the conditions that \( \varphi \) holds until \( \psi \) holds, and \( \psi \) holds at the next state of the path. As in CTL, the following axioms verify:

- (cl1) \( \mathcal{A}(\varphi \rightarrow \psi) \rightarrow (\mathcal{E} \varphi \rightarrow \mathcal{E} \psi) \)
- (cl2) \( \mathcal{E} \top \land \mathcal{A} \top \)
- (cl3) \( \mathcal{E}(\varphi \mathcal{U} \psi) \rightarrow \psi \lor (\varphi \land \mathcal{E} \mathcal{E} \varphi \mathcal{U} \psi) \)
- (cl4) \( \mathcal{A}(\varphi \mathcal{U} \psi) \rightarrow \psi \lor (\varphi \land \mathcal{A} \mathcal{A} \varphi \mathcal{U} \psi) \)
- (cl5) \( \mathcal{A}(\varphi \rightarrow (\mathcal{E} \varphi)) \rightarrow (\varphi \rightarrow \mathcal{A}(\varphi \mathcal{U} \psi)) \)
- (cl6) \( \mathcal{A}(\varphi \rightarrow (\mathcal{E} \varphi)) \rightarrow (\varphi \rightarrow \mathcal{A}(\varphi \mathcal{U} \psi)) \)
- (cl7) \( \mathcal{A}(\varphi \rightarrow (\mathcal{E} \varphi)) \rightarrow (\varphi \rightarrow \mathcal{A}(\varphi \mathcal{U} \psi)) \)
- (cl8) \( \mathcal{A}(\varphi \rightarrow (\mathcal{E} \varphi)) \rightarrow (\varphi \rightarrow \mathcal{E} \mathcal{E} \varphi \mathcal{U} \psi) \)

The set containing only the above axioms is denoted by CTL.
Properties of regular actions Regular actions provide high-level constructs which are suited to describe actions which an agent can execute upon its environment. \( \mathcal{E}_{\text{BDI}} \) is based in PDL [12] and therefore the following axioms verify

(a1) \((\alpha; \beta)\varphi \leftrightarrow (\alpha)(\beta)\varphi\)
(a2) \((\alpha + \beta)\varphi \leftrightarrow (\alpha)\varphi \lor (\beta)\varphi\)
(a3) \((\alpha^*\varphi \leftrightarrow \varphi \lor (\alpha)(\alpha^*)\varphi\)
(a4) \(\varphi \land (\alpha^*)\varphi \leftrightarrow (\alpha)\varphi \land (\alpha^*)\varphi\)

The set containing only the above axioms is denoted by \(PDL\).

Let now define some properties relating regular actions to temporal formulae.

**Lemma 1.** Let \(\mathcal{M}\) be a \(\mathcal{E}_{\text{BDI}}\)-model and \(\sigma\) a situation. If \(\mathcal{M}, \sigma \models (\alpha^*)\varphi\) then \(\mathcal{M}, \sigma \models \varphi \lor (\alpha^n)\varphi\), for \(n \in \mathbb{N}, n \geq 1\).

** Lemma 2.** Let \(\mathcal{M}\) be a \(\mathcal{E}_{\text{BDI}}\)-model and \(\sigma\) a situation. If \(\mathcal{M}, \sigma \models (\alpha^n)\varphi\), for \(n \geq 1\), then \(\mathcal{M}, \sigma \models (\alpha)E(\langle \alpha \rangle\top\varphi)\).

** Theorem 1.** Let \(\mathcal{M}\) be a \(\mathcal{E}_{\text{BDI}}\)-model and \(\sigma\) a situation. If \(\mathcal{M}, \sigma \models (\alpha^*)\varphi\) then \(\mathcal{M}, \sigma \models \varphi \lor (\alpha)E(\langle \alpha \rangle\top\varphi)\).

Relations between time and actions Time and action interact with each other in the following sense: if after successfully executing a particular action \(\alpha\) the proposition \(\varphi\) holds, then it is also true that there exists in the future a state where the proposition \(\varphi\) also holds. However, the inverse case is not true, since \(\varphi\) may hold as the result of executing an action \(\beta\) different from \(\alpha\). Formally, we have the following two axioms:

**Theorem 2.** Let \(\mathcal{M}\) be an \(\mathcal{E}_{\text{BDI}}\)-model, and \(\sigma \in \Sigma_M\). Then the following formulae are theorems of \(\mathcal{E}_{\text{BDI}}\):

\[(ta1) (\alpha)\varphi \rightarrow EX\varphi\]
\[(ta2) (\alpha)\varphi \rightarrow EF\varphi\]

As an example, consider the following scenarios:

- the agent, after driving a vehicle at high-speed, was not able to stop properly and crashed.

\((\text{KeepHighSpeed})\text{CrashedCar}\)

- the agent, after driving a vehicle for some time crashed it.

\(EF(\text{CrashedCar})\)
It is perfectly acceptable that the crashed car after some high-speed driving imply that the car will be crashed in the future. However, the vehicle being crashed in the future does not necessarily imply that the cause was driving at high speed.

**BDi layer** For beliefs we use the KD-45 axiom system and the axiom system KD for both desires and intentions, as in [10]. Therefore, the set $BEL_{K,D}$, for beliefs contains the following axioms:

- $(belK) \ \ BEL(\varphi \rightarrow \psi) \rightarrow (BEL(\varphi) \rightarrow BEL(\psi))$
- $(belD) \ \ BEL(\varphi) \rightarrow \neg BEL(\neg \varphi)$
- $(belI) \ \ BEL(\varphi) \rightarrow BEL(BEL(\varphi))$
- $(bel5) \ \ \neg BEL(\varphi) \rightarrow BEL(\neg BEL(\varphi))$

while $DES_{K,D}$ and $INT_{K,D}$ sets, for desires and intentions, contain respectively the first two and second two of the following axioms:

- $(desK) \ \ DES(\varphi \rightarrow \psi) \rightarrow (DES(\varphi) \rightarrow DES(\psi))$
- $(desD) \ \ DES(\varphi) \rightarrow \neg DES(\neg \varphi)$
- $(intK) \ \ INT(\varphi \rightarrow \psi) \rightarrow (INT(\varphi) \rightarrow INT(\psi))$
- $(intD) \ \ INT(\varphi) \rightarrow \neg INT(\neg \varphi)$

**Capabilities, resources and actions** Informally, we can see both the capabilities and resources as prerequisites for successful action-execution.

Resources and capabilities are defined in the Emotional-BDI model as follows:

**Resources**: these are physical/virtual means which may be drawn in order to make the agent capable of executing actions. If the resources for executing some action α do not exist, the action’s success may be at stake.

**Capabilities**: these are abstract means which the agent has to change the environment in some way, thus resembling to abstract plans of action. In fact, we can consider the set of capabilities as a dynamic set of plans which the agent has available to decide what to do in each of its execution states.

In $\mathcal{E}_{BDI}$, the axioms which characterise these concepts are

- $(f1) \ \ f(\alpha;\beta) \rightarrow f(\alpha) \land (\alpha)f(\beta)$
- $(f2) \ \ f(\alpha + \beta) \rightarrow f(\alpha) \lor f(\beta)$
- $(f3) \ \ f(\alpha^*) \rightarrow f(\alpha) \land (\alpha)f(\alpha^*)$
- $(f4) \ \ f(\alpha) \land (\alpha)^*(f(\alpha) \rightarrow (\alpha)f(\alpha)) \rightarrow f(\alpha^*)$

with $f \in \{CAP, RES\}$, and define the sets $CAP$ and $RES$, respectively.

Since agents live in complex and highly dynamic environments, the information they capture may contain too much noise. However, it is in this noisy information the agent relies on, and which affects the information the agent has about its own means. This is what we call **effective capabilities** [4, 2], which are
the (possibly wrong) beliefs about capabilities and resources. Formally it is expressed as \( \text{EffCap}(\alpha) = \text{BEL}(\text{CAP}(\alpha)) \land \text{BEL}(\text{RES}(\alpha)) \). This allows us to model acceptable facts such as \( \text{EffCap}(\alpha) \land (\alpha) \bot \), which expresses the fact that, based on insufficiently wrong information about resources and capabilities, an agent may not succeed in performing an action, as expected.

On the other hand, if we know that an action was successfully executed, then it is true that the agent had effective capabilities which lead him to execute the action. Formally this is written as \( (\alpha) \top \rightarrow \text{EffCap}(\alpha) \).

**Theorem 3.** Let \( M \) be a \( \mathcal{EBDI} \)-model, and \( \sigma \) a situation. Then, if \( M, \sigma \models \text{CAP}(\alpha^*) \) then \( M, \sigma \models \text{E}(\text{CAP}(\alpha) \land (\alpha)\text{CAP}(\alpha))\text{UT} \).

**Theorem 4.** Let \( M \) be a \( \mathcal{EBDI} \)-model, and \( \sigma \) a situation. Then, if \( M, \sigma \models \text{RES}(\alpha^*) \) then \( M, \sigma \models \text{E}(\text{RES}(\alpha) \land (\alpha)\text{RES}(\alpha))\text{UT} \).

**Fear** Fear, in \( \mathcal{EBDI} \), is explicitly referred by the modal operator \( \text{FEAR} \). This operator should be read as the agent fears that \( \varphi \) verifies.

For fear we require only the Kripke-axiom

\[ \text{FEAR}(\varphi \rightarrow \psi) \rightarrow (\text{FEAR}(\varphi) \rightarrow \text{FEAR}(\psi)) \]

to verify, and the set containing only this axiom is denoted by \( \text{FEAR}_K \).

**Fundamental Desires** Fundamental desires are special desires which are vital desires of the agent, or desires which cannot be failed to achieve, in any condition, since may put in danger the agent’s own existence. Fundamental desires should always be true and the agent must always do its best to maintain them valid.

The set of axioms which describe \( \text{FDES} \) are the following

\[
\text{fdesK} \quad \text{FDES}(\varphi \rightarrow \psi) \rightarrow (\text{FDES}(\varphi) \rightarrow \text{FDES}(\psi))
\]

\[
\text{fdesD} \quad \text{FDES}(\varphi) \rightarrow \neg \text{FDES}(\neg \varphi)
\]

and we denote this set by \( \text{FDES}_{KDT} \). This operator was introduced to facilitate the specification of triggering conditions for fear.

**The basic Emotional-BDI system** Now that all the modal operators were characterised, we are in conditions to define the simplest Emotional-BDI agents. This is called the basic Emotional-BDI agent.

**Definition 10.** A basic Emotional-BDI system is a set of formulae which is contain the union of the following sets of axioms

1. the set of all propositional tautologies
2. the time axiom set \( \text{CTL} \)
3. the action axiom set \( \text{PDL} \)
4. the belief axiom set \( \text{BEL}_{KD45} \)
5. the desire axiom set \( \text{DES}_{KD} \)
6. the intention axiom set \( INT_{KD} \)
7. the capabilities axiom set \( CAP \)
8. the resources axiom set \( RES \)
9. the fear axiom set \( FEAR_{K} \)
10. the fundamental desire axiom set \( FDES_{KD} \)

and that are closed under the inference rules of modus ponens \( \varphi, \varphi \rightarrow \psi \rightarrow \psi \)
and the necessitation rule \( \vdash \varphi \rightarrow \vdash \square \varphi \), where \( \square \in \{ \operatorname{BEL}, \operatorname{DES}, \operatorname{INT}, \operatorname{FDES}, \operatorname{FEAR}, \operatorname{AG}, [\alpha] \} \),
with \( \alpha \) being a regular action.

Any other system to specify an agent in \( E_{BDI} \) must extend this system. One such case is going to be presented in Section 4.

4 Modelling Fear

Agents are affected by fear in different ways, depending on how their internal representations differentiate between what are dangerous situations or non-dangerous situations. These differences of fear reactions have a direct impact on how agents may react in distinct ways with respect to some situation. For instance, a civilian may elicit fear about get shot just by hearing some fire shots, while a policeman or a soldier element may get only alert, due to its everyday contact with highly dangerous situations.

4.1 Threats

Negative emotions like fear are generally elicited when some possibly dangerous conditions of the environment (or generated by the agent) put at stake one of the agent’s fundamental goals. This may also put in cause the agent’s own self-preservation. Here, these conditions are called \textit{threats}.

Threats can be scaled in terms of their dangerousness and time occurrence. By this we mean that there are threats which are more dangerous than others, and threats which already are present on the environment and others which most likely will end up by occurring in the environment.

\textbf{Current threats}: the source of the threat is occurring now, and the agent has information about the fact that the existence of such source may put at stake its fundamental goals.
- VeryDangerousCThreat(\( \psi, \varphi \)) \( \equiv \) \( FDES(\varphi) \land \operatorname{BEL}(\psi \rightarrow \neg \varphi) \land \psi \)
- DangerousCThreat(\( \psi, \varphi \)) \( \equiv \) \( FDES(\varphi) \land \operatorname{BEL}(\psi \rightarrow \operatorname{AF}(\neg \varphi)) \land \psi \)
- CThreat(\( \psi, \varphi \)) \( \equiv \) \( FDES(\varphi) \land \operatorname{BEL}(\psi \rightarrow \operatorname{EF}(\neg \varphi)) \land \psi \)

\textbf{Future threats}: the source of the threat will eventually occur in the future.
- VeryDangerousPThreat(\( \psi, \varphi \)) \( \equiv \) \( FDES(\varphi) \land \operatorname{BEL}(\psi \rightarrow \neg \varphi) \land \operatorname{AF}\psi \)
- DangerousPThreat(\( \psi, \varphi \)) \( \equiv \) \( FDES(\varphi) \land \operatorname{BEL}(\psi \rightarrow \operatorname{AF}(\neg \varphi)) \land \operatorname{AF}\psi \)
- PThreat(\( \psi, \varphi \)) \( \equiv \) \( FDES(\varphi) \land \operatorname{BEL}(\psi \rightarrow \operatorname{EF}(\neg \varphi)) \land \operatorname{AF}\psi \)
In this paper, we formally model these classes of agents in order to show that our logic is expressive enough to model different kinds of agents, which generally react differently to distinct types of threats.

Now, a general threat – being it current or possible in the future – is any threat, with any amount of associated danger. Formally,

\[
\begin{align*}
\text{AnyCThreat}(\psi, \varphi) & \equiv \text{VeryDangerousCThreat}(\psi, \varphi) \lor \text{DangerousCThreat}(\psi, \varphi) \lor \text{CThreat}(\psi, \varphi) \\
\text{AnyPThreat}(\psi, \varphi) & \equiv \text{VeryDangerousPThreat}(\psi, \varphi) \lor \text{DangerousPThreat}(\psi, \varphi) \lor \text{PThreat}(\psi, \varphi)
\end{align*}
\]

4.2 Special atomic actions

Based on the literature \cite{4,13}, we will introduce a set of special purpose actions, which represent specific behaviour exhibited by the agent under certain emotional conditions. These actions are information processing strategies identified in humans \cite{14} and which are applied by them for obtaining solutions under specific emotional states. Here we will only present the strategies which had been identified as being activated under fear conditions.

Besides these strategies, we also introduce an abstract self-preservation action, whose meaning is the reactive character of an agent when the urgency for avoiding a dangerous situation is so great that none of the other processing strategies will provide good solutions in an acceptable time.

The set of special actions we define is:

1. **Self-preservation**: the self-preservation behaviour is activated when the agent is fearing the failure of some of its fundamental desires. We can see this as atomic action which mainly reacts to threats in a self-protective way. In $\mathcal{E}_{\text{BDI}}$, this special action is represented by selfpreservation.

2. **Direct Access**: this processing strategy relies on the use of fixed pre-existing structures/knowledge. It is the simplest strategy and corresponds to a minimisation of the computational effort and to fast solutions. In $\mathcal{E}_{\text{BDI}}$, this kind of processing is abstracted into the specialised atomic action $\text{das}$.

3. **Motivated Processing**: this processing strategy is employed by the agent when some desire which directs its behaviour must be maintained but may be at risk. This strategy is computationally intensive, as it should produce complex data-structures for preserving desires. In $\mathcal{E}_{\text{BDI}}$, this kind of processing is abstracted into the specialised atomic action $\text{mps}$.

4. **Substantive Processing**: this is considered the most complex information processing strategy and is usually applied to obtain possible solutions for situations which require large amount of computational effort for obtaining complex plans. It is applied when there are enough resources and capabilities and not too much urgency on find a solution. In $\mathcal{E}_{\text{BDI}}$, this kind of processing is denoted by the atomic action $\text{sps}$.
Considering the above actions as being atomic actions is of course a big abstraction to the complexity of Emotional-BDI agents. These actions are usually complex planning and revision strategies.

4.3 Specifying a fearful agent

We will now present a formal specification of what we consider a fearful Emotional-BDI agent. Informally, a fearful Emotional-BDI agent describes a class of software agents which elicit fear in all the situations where threats (or possible threats) are detected, not distinguishing between really dangerous threats or only light or possible threats. However, the temporal characteristics of the threats are taken into account by the agent, which fears their proximity. Based on what are the fears of the agent, it will employ distinct deliberation strategies studied in the literature [14], which require distinct levels of resources and capabilities, depending on what kind of urgent situations they are to be applied to.

The formal specification of fearful Emotional-BDI agents will be done in two parts:
- **eliciting conditions**, which are $\mathcal{E}_{BDI}$ formulae which explicitly define in which situations the agent elicits fear about propositions;
- **behaviour effect**, which are $\mathcal{B}_{BDI}$ formulae which state what kind of behaviour is exhibited by the agent in order to avoid the fears it has elicited and are still present in the agent’s internal state.

**The elicitation of fear** The agent elicits fear about some proposition if that proposition describes a threatening situation to one of its fundamental desires.

$$\text{AnyCThreat}(\psi, \varphi) \rightarrow \text{FEAR}(\psi)$$

If the threat is still to occur, the agent will fear not the threat itself, but its future occurrence.

$$\text{AnyPThreat}(\psi, \varphi) \rightarrow \text{FEAR}(\text{AF}\psi)$$

Now, if the agent already has beliefs about how to achieve a certain fundamental desire (or on how to maintain it), the will fear situations where unexpected interruptions on the execution of the actions to achieve that occur. In a first case, if the agent detects that it doesn’t have effective capabilities to successfully accomplish the action, it will fear for that lack of effective capabilities.

$$\text{FDES}(\varphi) \land \text{BEL}((\alpha; \beta)\varphi) \rightarrow [\alpha](\neg\text{EffCap}(\beta) \rightarrow \text{FEAR}(\neg\text{EffCap}(\beta)))$$

But the agent may only detect the fact that, even though it has effective resources to execute the rest of the action, the successful execution of that action will possibly lead to a non wanted falsity of the fundamental desire. In this case, the agent will fear for a successfully execution of the action

$$\text{FDES}(\varphi) \land \text{BEL}((\alpha; \beta)\varphi) \rightarrow [\alpha](\text{BEL}(\beta) \neg\varphi \rightarrow \text{FEAR}(\beta T))$$

58
The effects of fear in behavior If the agent is present before a current threat and it does not believe that it will obtain a good solution using even the quickest and less computational requiring processing strategy before the threatened fundamental desire becomes false, it will execute the self-preservation action in order to, at least, guarantee its most basic safety condition

$$\text{FEAR}(\psi) \land \text{AnyCT} \land (\text{BEL}(\neg \text{Eff Cap}(\text{das}))) \land (\text{selfpreservation}) \land$$

However, if the agent believes it has effective capabilities to execute a direct processing strategy, and therefore obtain better solutions to avoid the threat, it will execute the direct processing instead of just safe-guarding itself

$$\text{FEAR}(\psi) \land \text{AnyCT} \land (\text{BEL}(\text{Eff Cap}(\text{das}))) \land (\text{das}) \land$$

In the case of the threat is still to occur, the agent will employ either the motivated processing or substantive processing strategies, since they have still some time until the threat occurs and during this time they main obtain plans detailed enough to have a better guaranteed of avoiding the threat

$$\text{FEAR}(\alpha \land \text{AnyCT} \land (\text{BEL}(\text{Eff Cap}(\text{mps}))) \land (\text{mps} + \text{sps}) \land$$

If the fear of the agent is elicited during the execution of one action supposed to achieve or maintain a fundamental desire, the agent must exhibit a behaviour which allow it to obtain an alternative action to fulfill the first action’s goal

$$\text{BEL}((\psi) \land \text{FDES}(\psi) \land \text{[selfpreservation]}) \land (\text{selfpreservation}) \land$$

If it does not has the effective capabilities to do it, the agent must self preserve itself before doing something else

$$\text{BEL}((\psi) \land \text{FDES}(\psi) \land \text{[selfpreservation]}) \land (\text{selfpreservation}) \land$$

5 Related work

The subject of formally modelling emotional agents was already addressed by J.J. Meyer in [16]. In his work, Meyer uses the KARO framework and imposes conditions on the structure where KARO is interpreted, so that the triggering of emotions (happiness, sadness, anger and fear) and their effects on the behaviour of the agent are conveniently defined.

Work was also done in introducing the notion of capability in Rao & Georgeff’s BDI CTL logic. This work was presented in [11] but do not explicitly refer actions. It is only considered as the ability to rationally act towards the achievement of desires.
6 Conclusions and future work

In this paper we presented the syntax and semantics of $\mathcal{E}_{BDI}$ logic, a logic developed for modelling Emotional-BDI agents. By introducing the notions of threat, fear elicitation and effects of fear we have showed its expressiveness to model a class of Emotional-BDI agents which we called fearful agents.

Our approach was based in $\mathcal{BDI}_{CTL}$ extended with explicit reference to actions, resources and capabilities. However, for satisfiability purposes, we can transform any $\mathcal{E}_{BDI}$ formula into an $\mathcal{BDI}_{CTL}$ formula. In this way, we can easily extend the decision procedures given for $\mathcal{BDI}_{CTL}$ [10] to $\mathcal{E}_{BDI}$. In particular, we can obtain the decidability of the satisfiability problem of $\mathcal{E}_{BDI}$ formulas, as well as the soundness and completeness of the basic $\mathcal{E}_{BDI}$ system, with respect to a class of models. This is part of our ongoing work. We are also interested in providing different Emotional-BDI systems reflecting other behaviour which Emotional-BDI agent can exhibit.

7 Acknowledgements

We thank the anonymous referees for their constructive comments that helped to improve this paper.

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

References

Models and Methods for Plan Diagnosis*

Nico Roos and Cees Witteveen

1 Department of Computer Science, Universiteit Maastricht
   P.O.Box 616, NL-6200 MD Maastricht
   roos@cs.unimaas.nl

2 Faculty EEMCS, Delft University of Technology
   P.O.Box 5031, NL-2600 GA Delft
   C.Witteveen@tudelft.nl

Abstract. We consider a model-based diagnosis approach to the diagnosis of plans. Here, a plan executed by some agent(s) is considered as a system to be diagnosed. We introduce a simple formal model of plans and plan execution where it is assumed that the execution of a plan can be monitored by making partial observations of plan states. These observations of plan states are used to compare them with predicted states based on (normal) plan execution. Deviations between observed and predicted states can be explained by qualifying some plan steps in the plan as behaving abnormally. A diagnosis is a subset of plan steps qualified as abnormal that can be used to restore the compatibility between the predicted and the observed partial state. In contrast to model-based diagnosis, where minimum and minimal diagnoses are preferred, we argue that in plan-based diagnosis maximum informative diagnoses should be preferred. These are diagnoses that make the strongest predictions with respect to partial states to be observed in the future. We show that in contrast to minimum diagnoses, finding a (minimal) maximum informative diagnosis can be achieved in polynomial time. Finally, we show how we can deal with diagnosis of a plan if an arbitrary sequence of partial observations is given.

1 Introduction

With a growing complexity of plans, the possibility that something goes wrong during their execution increases correspondingly. No wonder then that more attention is paid to the development of robust plans. One way to enhance robustness is to perform plan diagnosis in order to identify the causes of failures, to predict future failures and, if possible, to prevent failures to occur. Since there is a huge number of potential factors that might prevent correct plan execution, it is not surprising that current approaches to plan diagnosis are rather diverse. For example, a changing environment might be such an important disturbing factor, preventing some parts of the plan to be executed by

* This research is supported by the Technology Foundation STW, applied science division of the Dutch Science Foundation (NWO) and the technology programme of the Ministry of Economic Affairs (the Netherlands). Project DIT5780: Distributed Model Based Diagnosis and Repair.
changing the preconditions of some instances of actions occurring in the plan. Another important source of plan failures could be attributed to the agent(s) controlling the actions prescribed in the plan by being unable to perform some of the actions required or accidently changing some of the preconditions of actions. In a broader, multi-agent perspective, one could even concentrate on incompatibilities between different agents involved in the execution of a joint plan as a major factor that could prevent parts of a joint plan to be executed correctly.

The main goal of this paper is to specify a general framework for plan diagnosis where, in principle, such general aspects of plan diagnosis could be dealt with. In developing such a framework it seems unavoidable to concentrate on some aspects of plan diagnosis and to (temporarily) neglect others. In this paper, we decided to concentrate on internal failure sources and leave external failure sources such as the environment, failures of executing agents as in [1] or incompatibilities between agents as in [5, 6] for future research. In particular, we will confine ourselves to the identification of failing actions as the only source of plan failure. Our main motivation for this restriction is that if the plan is correctly specified, errors in the plan execution process become manifest in the incorrect behavior of one or more instances of actions\(^1\). Whether or not we should be satisfied with the mere identification of one or more of such failing actions, a diagnostic process that identifies a set of actions that can be shown to be responsible for the abnormalities observed seems to be a useful analysis on its own. In a multi-agent planning systems, for example, identification of such failing actions can be used to identify incompatibilities between plans, to identify failing agents responsible for executing plans or to identify incompatibilities between agents involved in the plans. In the conclusion section we will elaborate on the potential extensions of the framework to deal with these questions.

Concentrating on the identification of failing actions, one of the main goals of this paper is to show how a plan consisting of a partially ordered set of instances of actions can be viewed as a system to be diagnosed and how a diagnosis can be established using partial observations of a plan in progress. Distinguishing between normal and abnormal execution of actions in a plan, we then introduce a plan diagnosis as a set of instances of actions qualified as abnormal to explain the deviations between expected plan states and observed plan states.

**Results** The results obtained in this paper are threefold. First of all we present a formal framework for plan diagnosis that enables us to define exactly how observations of a plan in execution can be used to derive a plan diagnosis. We show that establishing a plan diagnosis comes down to finding a subset of actions in a plan such that if these actions are qualified as abnormal, the observed plan states are compatible with predicted plan states. Secondly, after introducing minimal maximum informative diagnoses (abbreviated as mini-maxi diagnoses) as a special kind of diagnoses that have to be preferred above the well-known subset-minimal and minimum diagnoses known from model-based diagnosis, we show that in contrast to minimum diagnoses, mini-maxi diagnoses can be computed efficiently. Thirdly, we extend the framework to plan

---

\(^1\) Of course, some of these actions might not be specified in the plan.
diagnosis based on iterative partial observations and we show how this case can be reduced to establishing diagnoses with a simple pair of observations.2

**Organization** We first introduce a simple formal framework for representing states, actions and plans. Then, in Section 3, we introduce the main concepts of plan-based diagnosis and we discuss the idea of maximum informative diagnosis. In this section, we also discuss an efficient algorithm to find minimal maximum informative diagnoses. In Section 4, we extend our framework to diagnosis with a sequence of observations and Section 5 concludes this paper with a brief outlook to future research. Due to lack of space, all proofs of results have been omitted.

2 Preliminaries

We consider plan-based diagnosis as a simple extension of the model-based diagnosis (MBD) approach [2, 3, 8], where the model is not a description of an underlying physical system but a plan of one or more agents. By executing plans we change a part of the world. Therefore, before we discuss plans, we need to introduce a simple state-based view on the world.

**States** We assume that for the planning problem at hand, the world can be described by a set \( Var = \{v_1, v_2, \ldots, v_n\} \) of variables and their respective value domains \( D_i \). A complete state of the world \( \sigma \) then is a value assignment \( \sigma : Var \rightarrow \bigcup_{i=1}^{n} D_i \) to the variables. Slightly abusing terminology, we simply denote a complete state by an \( n \)-tuple \( \sigma = (\sigma(v_1), \ldots, \sigma(v_n)) \). A partial state is an element \( \pi \in D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \), where \( 1 \leq k \leq n \) and \( 1 \leq i_1 < \ldots < i_k \leq n \). We use \( Var(\pi) \) to denote the set of variables \{\( v_{i_1}, v_{i_2}, \ldots, v_{i_k} \} \subseteq Var \) specified in such a partial state \( \pi \). The value \( \sigma(v_j) \) of variable \( v_j \in Var(\pi) \) will be denoted by \( \pi(v_j) \). The value of a variable \( v_j \in Var \) not occurring in a partial state \( \pi \) is said to be undefined (or unpredictable) in \( \pi \), denoted by \( \perp \). Including \( \perp \) in every value domain \( D_i \) allows us to consider every partial state \( \pi \) as an element of \( D_1 \times D_2 \times \ldots \times D_n \).

Partial states can be ordered with respect to their information content: Given values \( d \) and \( d' \), we say that \( d' \) is at least as informative as \( d \), abbreviated as \( d \leq d' \), iff \( d = \perp \) or \( d = d' \). The containment relation \( \subseteq \) between partial states is the point-wise extension of \( \leq \); \( \pi \) is said to be contained in \( \pi' \), denoted by \( \pi \subseteq \pi' \), iff \( \forall v \in Var[\pi(v) \leq \pi'(v)] \).

An important notion in plan diagnosis is the notion of compatibility between partial states. Intuitively, two states \( \pi \) and \( \pi' \) are said to be compatible if there is no essential disagreement about the values assigned to variables in the two states, i.e., in principle they could characterize the same state of the world. More exactly, compatibility implies that for every \( v \in Var \), either \( \pi(v) = \pi'(v) \) or at least one of the values \( \pi(v) \) and \( \pi'(v) \) is undefined:

**Definition 1 (compatibilityrelation).** Two partial states \( \pi \) and \( \pi' \) are said to be compatible, denoted by \( \pi \approx \pi' \), iff \( \forall v \in Var[\pi(v) \leq \pi'(v)] \) or \( \pi'(v) \leq \pi(v) \).

---

2 An earlier version of the framework has appeared in [9] without the discussion of maximum informative diagnoses, algorithms and iterative observations.
If two partial states $\pi_1$ and $\pi_2$ are compatible, their information content can be fused to obtain a new partial state $\pi = \pi_1 \sqcup \pi_2$ that contains them both: $\pi = \pi_1 \sqcup \pi_2$ holds iff $\forall v \in \text{Var}[\pi(v) = \max \{\pi(v), \pi'(v)\}]$.

**Actions, Plan operators and Plan Steps** In the preceding sections we used to term “actions” in a rather informal way. From now on, we make a distinction between actions, plan operators and plan steps. First of all, an action refers to an activity that results in some change of the (current) state of the world, such as loading a vehicle or assembling components. A plan operator refers to a description of such an action in a plan. More exactly, a plan operator $o$ is a function mapping partial states to partial states by replacing the values of a subset $V_o \subseteq \text{Var}$ by other values (dependent upon the values of another set $V'_o \supseteq V_o$ of variables). Hence, every plan operator $o$ can be modeled as a (partial) function $f_o : D_{i_1} \times \ldots \times D_{i_k} \rightarrow D_{j_1} \times \ldots \times D_{j_l}$, where $1 \leq i_1 < \ldots < i_k \leq n$ and $\{j_1, \ldots, j_l\} \subseteq \{i_1, \ldots, i_k\}$. The variables whose value domains occur in $\text{dom}(f_o)$ will be denoted by $\text{dom}_V(o) = \{v_{i_1}, \ldots, v_{i_k}\}$ and, likewise, $\text{ran}_V(o) = \{v_{j_1}, \ldots, v_{j_l}\}$. It is required that $\text{ran}_V(o) \subseteq \text{dom}_V(o)$, i.e., the function $f_o$ associated with a plan operator is range-restricted. This functional specification $f_o$ constitutes the normal behavior of the plan operator $o$, also denoted by $f_o^{\text{nor}}$.

**Example 1.** Figure 1 depicts two states $\sigma_0$ and $\sigma_1$ (the white boxes) each characterized by the values of four variables $v_1, v_2, v_3$ and $v_4$. The partial states $\pi_0$ and $\pi_1$ (the gray boxes) characterize a subset of variables in a (complete) state. Plan operators are used to model state changes. The domain of the plan operator $o$ is the subset $\{v_1, v_2\}$, denoted by the arrows pointing to $o$. The range of $o$ is the subset $\{v_1\}$, which is denoted by the arrow pointing from $o$. Finally, the dashed arrow denotes that the value of variable $v_2$ is not changed by operator causing the state change.

![Fig. 1. Plan operators, states and partial states](image)

A plan operator $o$ may be used at several places in a plan. A specific occurrence of $o$ is called a plan step mapping a specific partial state into another partial state. A plan step $s$ as an occurrence of $o$ then describes a specific function application of the function specified by the operator $o$ at a specific place in the plan. Therefore, given a set $O$ of plan operators, we consider a set $S = \text{inst}(O)$ of instances of plan operators in $O$, called the set of plan steps. A plan step will be denoted by a small roman letter $s_i$. The plan operator $o$ the instance $s_i$ was instantiated from is denoted by $\text{op}(s_i)$. If $\text{op}(s_i) = o$, the instance $s_i$ is said to be of type $o$. 

4
**Plans and Plan Execution** A plan is a tuple $P = \langle O, S, \prec \rangle$ where $S \subseteq Inst(O)$ is a set of plan steps occurring in $O$ and $(S, \prec)$ is a partial order. The partial order relation $\prec$ specifies an execution relation between plan steps: for each $s \in S$ it holds that $s$ is executed immediately after all plan steps $s'$ such that $s' \prec s$ have been finished. We will denote the transitive reduction of $\prec$ by $\ll$, i.e., $\ll$ is the smallest subrelation of $\prec$ such that the transitive closure $\ll^+$ of $\ll$ equals $\prec$.

**Example 2.** Figure 2 gives an illustration of a plan. Arrows relate the objects a plan step uses as inputs and the objects it produces as its outputs to the plan step itself. In this plan, the direct execution relation is specified as $s_1 \ll s_3, s_2 \ll s_4, s_4 \ll s_5$ and $s_4 \ll s_6$.

![Fig. 2. Plans and plan steps. Each state characterizes the values of four variables $v_1, v_2, v_3$ and $v_4$. States are changed by application of plan steps $s_i$ for $i = 1, 2, \ldots, 6$.](image)

Without loss of generality, we assume that every plan step $s \in S$ takes a unit of time to execute and the execution of the first plan step starts at time $t = 0$. Using this assumption and the definition of the execution relation $\prec$, the time $t$ at which a plan step will be executed is uniquely determined: Let $depth_P(s)$ be the depth of plan step $s$ in plan $P = \langle O, S, \prec \rangle$. Then the time $t_s$ at which the plan step $s$ is executed is $t_s = depth_P(s)$ and $s$ will be completed at time $t_s + 1$. Let $P_t$ denote the set of plan steps $s$ with $depth_P(s) = t$, let $P_{>t} = \bigcup_{t' > t} P_{t'}$, $P_{<t} = \bigcup_{t' < t} P_{t'}$ and let $P_{[t, t']} = \bigcup_{k=t}^{t'} P_k$.

**Example 3.** Consider again Figure 2. In this plan, the depth of $s_1$ and $s_2$ is 0, the depth of $s_3$ and $s_4$ is 1, and the depth of $s_5$ and $s_6$ is 2. Therefore, $P_0 = \{s_1, s_2\}$, $P_1 = \{s_3, s_4\}$ and $P_2 = \{s_5, s_6\}$.

---

5 Here, $depth_P(s) = 0$ if $\{s' \in S \mid s' \ll s\} = \emptyset$ and $depth_P(s) = 1 + \max\{depth_P(s') \mid s' \ll s\}$, else. If the context is clear, we omit the subscript $P$.
Given a state $\sigma$ at some time $t$ and the set $P_t$ of plan steps to be executed at time $t$ we want to be sure that the next state $\sigma'$ at time $t+1$ is uniquely defined. If $P_t$ contains two plan steps $s$ and $s'$ with overlapping ranges, i.e., if $\text{ran}_{\text{var}}(s) \cap \text{ran}_{\text{var}}(s') \neq \emptyset$, the final result of a variable $v$ occurring in this intersection is not uniquely defined in $\sigma'$. We therefore assume the following condition to hold:

**Determinism:** If $P$ is a plan and $s, s'$ are plan steps in $P$ such that $\text{ran}_{\text{var}}(s) \cap \text{ran}_{\text{var}}(s') \neq \emptyset$ then $\text{depth}_P(s) \neq \text{depth}_P(s')$.

It is not difficult to see (and can be easily proven using the derivability relations to be discussed) that Determinism guarantees that a future plan state can be defined uniquely given a plan and a currently uniquely defined plan state.

2.1 Qualifications

The correct execution of a plan step may fail either because of an inherent malfunctioning or because of a malfunctioning of an agent responsible for executing the action, or because of unknown external circumstances. In all these cases we would like to model the effects of executing such failing plan operators. Therefore, we introduce a set of health modes $H_s$ for each plan step $s$. This set $H_s$ contains at least the normal mode $\text{nor}$, the mode $\text{ab}$ indicating the most general abnormal behavior, and possibly several other specific fault modes. The most general abnormal behavior of plan step $s$ is specified by the function $f_{ab}^s$, where $f_{ab}^s(d_{i_1}, d_{i_2}, \ldots, d_{i_k}) = (\perp, \perp, \ldots, \perp)$ for every partial state $(d_{i_1}, d_{i_2}, \ldots, d_{i_k}) \in \text{dom}(f_o)$.

To keep the discussion simple, in the sequel we distinguish only the health modes $\text{nor}$ and $\text{ab}$.

Let us assume, for the moment, that each plan step can be viewed as an independent component of a plan. To each plan step then a a health mode $h_s \in \{\text{nor}, \text{ab}\}$ can be assigned and the result is called a qualified plan. In establishing which part of the plan fails, we are only interested in those plan steps qualified as abnormal. Therefore, we define a qualified version $P_Q$ of a plan $P = \langle O, S, \prec \rangle$ as a tuple $P_Q = \langle O, S, \prec, Q \rangle$, where $Q \subseteq S$ is the subset of plan steps qualified as abnormal (and therefore, $S - Q$ is the subset of plan steps qualified as normal).

Since a qualification $Q$ corresponds to assigning the health mode $\text{ab}$ to every plan step in $Q$ and since $f_{\text{ab}}^s(d_{i_1}, d_{i_2}, \ldots, d_{i_k}) = (\perp, \perp, \ldots, \perp)$ for every plan step $s \in Q$ with $\text{type}(s) = o$, the results of anomalously behaving plan steps are unpredictable. Note that a “normal” plan $P$ corresponds to the qualified plan $P_\emptyset$ and that in our context “undefined” is considered to be equivalent to “unpredictable”.

2.2 Derivability relations induced by plan execution

Note that $P$ on a given initial state $\pi_0$ will induce a sequence of states $\pi_0, \pi_1, \ldots, \pi_k$, where $\pi_{t+1}$ is generated from $\pi_t$ by applying the set of plan steps $P_t$ to $\sigma_t$. To define this relation between partial states at different time points we denote a partial state $\pi$ at a given time $t$ by a tuple, also called a timed state, denoted by $(\pi, t)$.

---

4 This definition implies that the behavior of abnormal steps is essentially unpredictable.
Let \( s \) be a plan step. We say that \( s \) is \emph{enabled} in a state \( \pi \) if \( \text{dom}_{V,\var} (s) \subseteq \text{Var}(\pi) \). Intuitively, we can predict the timed state \((\pi', t + 1)\) using the timed state \((\pi, t)\) and the set \( P_i \) of to be executed plan steps as follows:

1. whenever a variable \( v \) does not occur in the range of a plan step \( s \in P_i \), its value in state \( \pi' \) is the same as its value in \( \pi \), i.e., \( \pi(v) = \pi'(v) \);
2. if the variable \( v \) occurs in the range of a normally qualified plan step \( s \) that is enabled in \( \pi \), then \( \pi'(v) = f_{s}^\text{nor}(\pi(v)) \);
3. in all other cases, there is not sufficient information to predict the value of \( \pi'(v) \), either because \( v \) occurs in the range of an abnormally qualified plan step \( s \) or \( v \) occurs in the range of some plan step \( s \in P_i \) not enabled in \( \pi \).

Formally, this relation is defined as follows:

**Definition 2.** We say that \((\pi', t + 1)\) is (directly) generated by execution of the \( Q \)-qualified plan \( P_Q \) from \((\pi, t)\), abbreviated by \((\pi, t) \rightarrow_{Q,P} (\pi', t + 1)\), iff for every \( v \in V \) the following conditions hold:

1. if \( v \notin \bigcup_{s \in P_i} \text{ran}_{V,\var}(s) \) then \( \pi'(v) = \pi(v) \);
2. if \( v \in \bigcup_{s \in P_i - Q} \text{ran}_{V,\var}(s) \) then \( \pi'(v) = f_{s}^\text{nor}(\pi(v)) \);
3. else \( \pi'(v) = \bot \).

It is easy to see that thanks to Determinism, the immediate derivability relation \( \rightarrow_{Q,P} \) is well-defined and deterministic:

**Proposition 1.** Let \( P_Q \) be a qualified plan and let \((\pi, t)\) a timed state. Then \((\pi, t) \rightarrow_{Q,P} (\pi', t + 1) \) and \((\pi, t) \rightarrow_{Q,P} (\pi'', t + 1) \) implies \( \pi'' = \pi' \).

We extend this direct derivability relation to a general derivability relation in a straightforward way:

**Definition 3.** For arbitrary values of \( t \leq t' \) we say that \((\pi', t')\) is (directly or indirectly) generated by execution of \( P_Q \) from \((\pi, t)\), denoted by \((\pi, t) \rightarrow_{Q,P}^* (\pi', t') \), iff the following conditions hold:

1. if \( t = t' \) then \( \pi' = \pi \);
2. if \( t' = t + 1 \) then \((\pi, t) \rightarrow_{Q,P} (\pi', t') \);
3. if \( t' > t + 1 \) then there exists some state \((\pi'', t' - 1)\) such that \((\pi, t) \rightarrow_{Q,P}^* (\pi'', t' - 1) \) and \((\pi'', t' - 1) \rightarrow_{Q,P} (\pi', t') \).

Note that \((\pi, t) \rightarrow_{Q,P}^* (\pi', t') \) denotes the normal execution of a normal plan \( P_Q \). Such a normal plan execution will also be denoted by \((\pi, t) \rightarrow_{P}^* (\pi', t') \).

Using these definitions, it is not difficult to show that for every \( 0 \leq t \leq k \), the timed state \((\pi', t)\) where \((\pi, 0) \rightarrow_{Q,P}^* (\pi', t) \) is uniquely defined if \( t \) satisfies the Determinism requirement.

**Example 4.** Figure 3 gives an illustration of an execution of a plan with abnormal plan steps. Suppose plan step \( s_3 \) is abnormal and generates a result that is unpredictable (\( \bot \)). Given the qualification \( Q = \{ s_3 \} \) and the partially observed state \( \pi_0 \) at time point \( t = 0 \), we predict the partial states \( \pi_i \) as indicated in Figure 3, where \((\pi_i, t_i) \rightarrow_{Q,P}^* (\pi', t') \) for \( i = 1, 2, 3 \). Note that since the value of \( v_1 \) and of \( v_5 \) cannot be predicted at time \( t = 2 \), the result of plan step \( s_6 \) and of plan step \( s_8 \) cannot be predicted and \( \pi_3 \) contains only the value of \( v_3 \). \[\blacksquare\]
3 Observations and Diagnoses

To establish plan diagnosis in our framework we need to make observations. Our framework provides a natural candidate for representing such observations: an observation \( \text{obs}(t) \) at time \( t \) is a timed state \((\pi, t)\) where \( \pi \) is a partial state. This implies that we do not require observations to specify a complete state. Suppose we have an observation \( \text{obs}(t) = (\pi, t) \) and an observation \( \text{obs}(t') = (\pi', t') \) at some later time \( t' > t \geq 0 \) during the execution of the plan \( P \). We would like to use these observations to infer the health modes of the plan steps occurring in \( P \). First, assuming a normal execution of \( P \), we can predict the timed state \((\pi', t')\) given the observation \( \text{obs}(t) \): if all plan steps behave normally, we predict the timed state \((\pi', t')\) such that \( \text{obs}(t) \rightarrow^* P(\pi', t') \). Such a prediction has to be compared with the actual observation \( \text{obs}(t') = (\pi', t') \) made at time \( t' \). It is easy to see whether the predicted state and the observed state match: in that case we should be able to find a state \( \sigma \) such that both the observed state \( \pi' \) and the predicted state \( \pi' \) are contained in \( \sigma \), that is, \( \pi' \sqsubseteq \sigma \) and \( \pi' \sqsubseteq \sigma \). By definition of compatibility, such a \( \pi' \) can only exist if \( \pi' \approx \pi' \) holds.\(^\text{5}\) If this is not the case, the execution of some plan steps must have gone wrong and we have to determine a qualification \( Q \) such that the predicted state \( \pi' \approx \pi' \) derived using \( Q \) is compatible with \( \pi' \). Hence, we have the following straight-forward extension of the diagnosis concept in MBD to plan diagnosis (cf. [3]):

\[ \text{Definition 4. Let } P = \langle \mathcal{O}, S, \prec \rangle \text{ be a plan with observations } \text{obs}(t) = (\pi, t) \text{ and } \text{obs}(t') = (\pi', t'), \text{ where } t < t' \leq \text{depth}(P) \text{ and let } \text{obs}(t) \rightarrow^* Q, P(\pi', t') \text{ be a derivation using } P_Q. \text{ Then } Q \text{ is said to be a plan diagnosis of } \langle P, \text{obs}(t), \text{obs}(t') \rangle \text{ iff } \pi' \approx \pi' \text{.} \]

In order to be able to establish a diagnosis, we simply assume that for every variable \( v \) there exists at least some plan step \( s \) and some time \( t \leq t'' \leq t' \) such that \( s \in P_{t''} \) and \( v \in \text{ran}_{\text{var}}(s) \).

\(^5\) See the definition in the preliminaries.
Example 5. Consider again Figure 3 and suppose that we did not know that plan step \( s_3 \) was abnormal and that we observed \( \text{obs}(0) = ((d_1, d_2, d_3, d_4), 0) \) and \( \text{obs}(3) = ((d'_1, d'_2, d'_3), 3) \). Using the normal plan derivation relation starting with \( \text{obs}(0) \) we will predict a state \( \pi'_0 \) at time \( t = 3 \) where \( \pi'_0 = (d'_1, d'_2, d'_3) \). If everything is ok, the values of the variables predicted as well as observed at time \( t = 3 \) should correspond, i.e. we should have \( d'_j = d'_j \) for \( j = 1, 3 \). If, for example, only \( d'_1 \) would differ from \( d'_1 \), then we could qualify \( s_0 \) as abnormal, since then the predicted state at time \( t = 3 \) using \( Q = \{ s_0 \} \) would be \( \pi'_0 = (d'_1) \) and this partial state agrees with the predicted state on the value of \( v_3 \).

Remark 1. Note that for all variables in \( \text{Var}(\pi') \cap \text{Var}(\pi'_0) \), the qualification \( Q \) provides an explanation for the observation \( \pi' \) made at time point \( t' \). Hence, for these variables the qualification provides an abductive diagnosis [2]. For all observed variables in \( \text{Var}(\pi') - \text{Var}(\pi'_0) \), no value can be predicted given the qualification \( Q \). Hence, by declaring them to be unpredictable, possible conflicts with respect to these variables if a normal execution of all plan steps is assumed, are resolved. This corresponds with the idea of a consistency-based diagnosis [8].

3.1 Maximal informative diagnoses

On intuitive grounds, one would prefer, like in MBD, smaller diagnoses above larger ones. One of the intuitions behind this preference is that, normally, we expect a plan to deliver correct results. Any deviation from this normality assumption should be as small as possible and we prefer a qualification that does not contain more actions qualified as abnormal than necessary. This, like in MBD, would be an obvious reason to prefer subset-minimal diagnoses and especially minimum diagnoses among the set of minimal diagnoses. These notions can be easily defined in our framework as follows: Given plan observations \( \langle P, (\pi, t), (\pi', t') \rangle \), a qualification \( Q \) is said to be

1. a (subset) minimal plan diagnosis if for every plan diagnosis \( Q' \) such that \( Q' \subseteq Q \), it holds that \( Q = Q' \).
2. a minimum plan diagnosis if for every plan diagnosis \( Q' \), it holds that \( |Q| \leq |Q'| \).

Example 6. Consider the plan depicted in Figure 4. Suppose \( \text{obs}(0) = (\pi_0, 0) \) and \( \text{obs}(3) = (\pi'_3, 3) \) and \( \pi'_3 \) equals \( \pi_3 \) except that there is a deviation in the value of \( v_2 \) at time \( t = 3 \) (as indicated by the black dot). Note that there are three possible minimal diagnoses that are also minimum diagnoses: \( Q_1 = \{ s_1 \} \), \( Q_2 = \{ s_3 \} \) and \( Q_3 = \{ s_0 \} \). Let \( \pi'_Q \) denote the state derived at time \( t = 3 \) by using \( Q \) as a qualification. Then \( \text{Var}(\pi'_{Q_1}) = \emptyset \), \( \text{Var}(\pi'_{Q_2}) = \{ v_4, v_5 \} \) and \( \text{Var}(\pi'_{Q_3}) = \{ v_3, v_4, v_5 \} \), so these partial states predicted differ in their information content.

Example 6 shows that, in general, minimum or minimal diagnoses might considerably differ in their predictive power. For example, if we take \( D_1 \) as a diagnosis, the values of all variables predicted at time \( t = 3 \) will be undefined, while taking \( D_3 \) as a diagnosis, only \( v_1 \) and \( v_2 \) are undefined. Hence, it seems that minimality as the single criterion to choose a suitable diagnosis is not sufficient. Intuitively, besides minimizing
the number of abnormal plan steps, we would prefer those diagnoses \( Q \) that generate a state \( \pi'_Q \) that minimizes the number of undefined values. We call such diagnoses maximally informative diagnoses:

**Definition 5 (maxi-diagnosis).** Given plan observations \( \langle P, (\pi, t), (\pi', t') \rangle \), a diagnosis \( Q \) is said to be a maximally informative plan diagnosis, abbreviated maxi-diagnosis, if there exists no diagnosis \( Q' \) such that \( \text{Var}(\pi_Q) \subset \text{Var}(\pi_{Q'}) \).

Note that for a given state \( \pi \), \( \text{Var}(\pi) \) is the set of variables defined in \( \pi \).

**Remark 2.** Analogous to the distinction between minimal diagnoses and minimum diagnoses, we could introduce the notion of a maximum informative diagnosis as a diagnosis \( Q \) for which there exists no diagnosis \( Q' \) such that \( |\text{Var}(\pi_Q)| < |\text{Var}(\pi_{Q'})| \). Unlike minimal and minimum diagnoses, however, it turns out that every maximally informative diagnosis is also a maximum informative diagnosis, i.e., there is no distinction between the subset-maximal and the cardinality-maximal notions of informative diagnoses. In fact, we can show an even stronger result: given some plan observations \( \langle P, (\pi, t), (\pi', t') \rangle \), for every two maxi-diagnoses \( Q \) and \( Q' \), it holds that \( \text{Var}(\pi_Q) = \text{Var}(\pi_{Q'}) \), i.e., they are equally informative in a strict sense.

Such maxi-diagnoses, however, are not always subset minimal diagnoses. By combining the two criteria, however, we obtain a qualification that is able to achieve compatibility with the observations, being as exact in its predictions as possible, without considering too many actions as behaving abnormally. We therefore define a minimal maximally-informative diagnosis as follows:
**Definition 6 (mini-maxi diagnosis).** Given plan observations \( \langle P, (\pi, t), (\pi', t') \rangle \), a diagnosis \( Q \) is said to be a minimal maximally informative plan diagnosis, abbreviated as mini-maxi diagnosis, if (i) \( Q \) is a maxi-diagnosis and (ii) there exists no maxi-diagnosis \( Q' \) such that \( Q' \subset Q \).

### 3.2 Finding maxi-diagnoses

Finding minimum diagnoses is computationally hard, even in our simple framework. Surprisingly, however, finding maxi-diagnoses and even finding mini-maxi-diagnoses is tractable. We will first give an intuitive description of an efficient procedure to find a (mini-) maxi diagnosis and then give a polynomial algorithm for finding a mini-maxi diagnosis.

Suppose we have plan observations \( \langle P, (\pi, t), (\pi', t') \rangle \). To determine a maxi-diagnosis, we first determine the disagreement set \( \text{Dis}_\varphi \) of all those variables whose values are defined in both the observed state \( \pi' \) and the predicted state \( \pi'_\varphi \) at time \( t' \) but differ:

\[
\text{Dis}_\varphi = \{ v \in \text{Var} | [\pi'_\varphi(v) \neq \pi'(v) \land (\pi'_\varphi(v) \geq \bot) \land (\pi'(v) > \bot)] \}. 
\]

Next, we collect all plan steps \( s \) at time \( t' - 1 \) such that there exists a variable \( v \in \text{ran}_{\text{Var}}(s) \cap \text{Dis}_\varphi \). By the determinism requirement, two different plan steps \( s \) and \( s' \) occurring in some set \( P_t \) cannot have a variable in common in their range, hence for every \( v \in \text{Dis}_\varphi \) there is at most one plan step \( s_v \in P_{t-1} \) such that \( v \in \text{ran}_{\text{Var}}(s_v) \). Then we remove all variables \( v \) that occur in the range of the plan steps just selected from the disagreement set. For \( i = 2, 3, \ldots \), we iteratively select new plan steps at times \( t' - i \) having a variable in their range that also occurs in the disagreement set and we remove these variables until the disagreement set is empty. It is not difficult to see that this procedure generates the set \( Q_{\text{max}} = \{ s_v | v \in \text{Dis}_\varphi \} \) where \( s_v \) is the latest plan step in the plan causing the value \( v \) to occur in the disagreement set. It can be easily proven that \( Q_{\text{max}} \) is a maxi-diagnosis.

In order to obtain a mini-maxi diagnosis, we have to refine this procedure slightly. Firstly, let us introduce the notion of a scope of a plan step \( s \). Intuitively, the scope of a plan step \( s \) contains all plan steps \( s' \) such that all variables \( v \in \text{ran}_{\text{Var}}(s') \) will become undefined whenever \( s \) is qualified as abnormal. This scope \( \text{scope}_P(s) \) is inductively defined as follows: (i) \( s \in \text{scope}_P(s) \) and (ii) if there exist plan steps \( s' \) and \( s'' \) such that \( \text{depth}_P(s') < \text{depth}_P(s'') \) and \( \text{ran}_{\text{Var}}(s') \cap \text{dom}_{\text{Var}}(s'') \neq \emptyset \) then \( s' \in \text{scope}_P(s) \) implies \( s'' \in \text{scope}_P(s) \). Now, in the above procedure to generate a maxi-diagnosis, if we simply add a set \( S_i \) of new plan steps belonging to \( P_{t-i} \) to the already selected set of plan steps \( S \), some of the plan steps \( s \) occurring in \( S_i \) might contain variables \( v' \) in their scope that also occur in the domain of plan steps \( s' \in S \) already selected. That implies \( \text{scope}(s) \supset \text{scope}(s') \); hence, adding such a plan step \( s \) makes the inclusion of the previously added plan steps \( s' \) superfluous. Therefore, at each iteration step, we remove such redundant plan steps \( s' \) to obtain a mini-maxi diagnosis.

The following algorithm (see Algorithm 1 states an iterative procedure to obtain a mini-maxi diagnosis \( Q_{\text{max}} \):

**Example 7.** Consider again the plan execution depicted in Figure 4. Given \( \text{obs}(0) \) and \( \text{obs}(3) \) and a deviation in the value of \( s_2 \) at time \( t = 3 \), we determine the disagreement set \( \text{Dis} = \{ s_2 \} \). After selecting \( s_2 \) as a plan step to be included in the diagnosis, the disagreement set is empty. Hence, \( D = \{ s_0 \} \) is a maxi-diagnosis.
Algorithm 1 Algorithm to compute mini-maxi diagnoses

Require: plan observations \( \langle P, (\pi, t), (\pi', t') \rangle \)
Ensure: a mini-maxi informative diagnosis \( Q_{\text{max}} \)

Let \( \text{Dis}_0 = \emptyset \) and let \( Q_{\text{max}} = \emptyset \);
\( i := 0 \)
while \( \text{Dis}_0 \neq \emptyset \) do
\( i := i + 1; \)
\( S_i := \{ s \in P_{t-1} \mid \exists v \in \text{Dis}_0[v \in \text{ranVar}(s)] \}; \)
\( Q_i := \{ s \in Q_{\text{max}} \mid \exists s' \in S_i[s \in \text{scope}(s')] \}; \)
\( Q_{\text{max}} := (Q_{\text{max}} - Q_i) \cup S_i; \)
\( \text{Dis}_0 := \text{Dis}_0 - \bigcup_{s \in Q_{\text{max}}} \text{ranVar}(s) \)
end while
return \( Q_{\text{max}} \)

4 Diagnosing a sequence of observations

Until now we discussed the diagnosis of a plan \( P \) using (simple) plan observations: we
considered diagnoses based on two observations of \( P \) at different time points \( t < t' \).
Considering the plan \( P \) as a system to be diagnosed, there is a direct correspondence
between MBD and plan diagnosis: the observation \( \text{obs}(t) \) at the earliest time point cor-
responds to observing the inputs of the system, while the observations \( \text{obs}(t') \) at the
latest time point corresponds to observing the outputs. Plan diagnosis, however is not
limited to making observations at two different points of time. For it may happen that
during the execution of a plan we are able to make a sequence of \( k > 2 \) observations at
some specific time points \( t_1 < t_2 < \ldots < t_k \).

In this section we will adapt the definition of a plan diagnosis to such a sequence
of observations. We will first make a careful analysis of the adaptations to be made by
discussing a simple example.

Example 8. Consider the plan \( P \) as depicted in Figure 5 (a). There are three observations
\( (\pi_0, t_0) \), \( (\pi_1, t_1) \) and \( (\pi_2, t_2) \). Using the observation \( (\pi_0, t_0) \) and assuming no
faulty plan steps, we predict the partial state \( (\pi_0', 1) \) at time \( t = 1 \) as depicted in
Figure 5 (b). Note that this predicted state is compatible with the observed state \( (\pi_1, t_1) \).
Using the same observation \( (\pi_0, t_0) \), we also predict an observed state \( (\pi_0', 2) \) at time
\( t = 2 \) where only the variables \( v_1 \) and \( v_2 \) are defined. Suppose that this prediction is
compatible with the observed state \( \pi_2 \) at time \( t = 2 \). The observation \( (\pi_1, t_1) \) can also
be used to obtain information about the state of the plan at time \( t = 2 \). In this case, how-
ever, using \( (\pi_1, t_1) \) the empty partial state \( \pi_{1,2}' = (\bot, \ldots, \bot) \) is predicted. This state,
by definition, is compatible with any prediction or observation made at time \( t = 2 \).
Therefore, we could conclude that the fusion \( \pi_{0,2}' \sqcup \pi_{1,2}' \sqcup \pi_2 \) represents the total
information that can be derived from both observations at time \( t = 2 \), assuming that the plan
is executed correctly and that the prediction \( \pi_{0,2}' \) is compatible with the observation \( \pi_2 \).

However, in this way we did not use all the information available at time \( t = 1 \)
to make a prediction for the state of the plan at time \( t = 2 \). For example, we are not
able to detect whether the value of \( v_3 \) deviates from the prediction that can be made if
we systematically combine the predictions using both the observations \( \pi_0 \) and \( \pi_1 \). For
example, since the predicted state $\pi'_{0,1}$ and the observed state $\pi_1$ are compatible, the total state information available at time $t = 1$ is the fused state $\pi'_{0,1} \sqcup \pi_1$. From this latter state we are able to predict the partial state $\pi'_2$ at time $t = 2$ where $\text{Var}(\pi'_2) = \{v_1, v_2, v_3, v_4\}$ and therefore, we could detect whether $\pi_2$ at variable $v_3$ is compatible with this prediction. We conclude that we have to carefully combine all the derivations made from previous observations with the current observed state information to make predictions for the state of the plan at a future time.

To model the case where a sequence of $k > 2$ observations is made, we consider a plan $P = \langle O, S, < \rangle$ with a sequence $\text{Obs} = (\text{obs}(t_1), \ldots, \text{obs}(t_k))$ of observations where $\text{obs}(t_i) = (\pi_i, t_i)$ for $i = 1, \ldots, k$ and $t_1 < t_2 < \ldots < t_k \leq \text{depth}(P)$.

Let us first consider, given a plan $P$ and such a sequence of observations $\text{Obs}$, the constraints a diagnosis $Q \subseteq S$ has to satisfy. Consider the first observation $(\pi_1, t_1)$. From this observation we can make predictions $\pi'_{1,i}$ for all time points $t_i$ with $i > 1$, using the derivations

$$(\pi_1, t_1) \rightarrow^*_Q (\pi'_{1,i}, t_i).$$

Clearly, since $Q$ is assumed to be a diagnosis of $P$ using $\text{Obs}$, we should require $\pi'_{1,i} \approx \pi_i$ for all $1 < i \leq k$.

Now consider the second time point $t_2$. Note that the total state information available at time $t_2$ consists of the observed partial state $\pi_2$ and the predicted partial state $\pi'_{1,2}$ compatible with it. Hence, the total information available at $t_2$ is represented by the fused state $\pi_2 \sqcup \pi'_{1,2}$. Using this fused state and the qualification $Q$ we can make predictions $\pi'_{2,i}$ for the partial states $\pi_i$ observed at $t_i$ for $i > 2$:

$$(\pi_2 \sqcup \pi'_{1,2}, t_2) \rightarrow^*_Q (\pi'_{2,i}, t_i).$$
Again, since $Q$ is a diagnosis, all these predictions $\pi'_{2,i}$ should be compatible with $\pi_i$ for all $2 < i \leq k$, i.e., it should hold that $\pi'_{2,i} \approx \pi_i$ for all $2 < i \leq k$.

Proceeding inductively, assume that predictions $\pi'_{h,i}$ have been made using all information available at times $t_h$ where $h = 1, 2, \ldots, i - 1$. Then the predictions $\pi'_{i,j}$, where $j = i + 1, \ldots, k$, can be obtained as follows: The total information available at time $t_i$ is $\pi_i \cup \pi'_{i-1} \cup \ldots \cup \pi'_{i-1,i}$. We can make predictions $\pi'_{i,j}$ for all times $t_j$ where $j = i + 1, \ldots, k$ using the derivations: $(\pi_i \cup \pi'_{i-1} \cup \ldots \cup \pi'_{i-1,i}, t_i) \rightarrow_{Q,P} (\pi'_{i,j}, t_j)$.

The representation of the total information available at time $t_i$ can be simplified, since it turns out that $\pi_i \cup \pi'_{i-1} \cup \ldots \cup \pi'_{i-1,i} = \pi_i \cup \pi'_{i-1,i}$. This can be seen as follows:

For $1 \leq h < i - 1$ it holds that $(\pi_h \cup \pi'_{i-1,h} \cup \ldots \cup \pi'_{i-1,i}, t_h) \rightarrow_{Q,P}^* (\pi'_{h,i}, t_i)$. Since the derivability relation is deterministic, $(\pi'_{h,i}, t_i)$ must hold for $h = 1, \ldots, i - 2$. Since we have $\pi'_{h,i-1} \subseteq \pi_{i-1} \cup \pi'_{i-1} \cup \ldots \cup \pi'_{i-1,i}$ and $(\pi_{i-1} \cup \pi'_{i-1} \cup \ldots \cup \pi'_{i-1,i}, t_i) \rightarrow_{Q,P} (\pi'_{i-1,i}, t_i)$, hence we obtain $\pi_{i-1} \cup \pi'_{i-1} \cup \ldots \cup \pi'_{i-1,i} \approx \pi_i \cup \pi'_{i-1,i}$. Therefore, at time $t_i$ we only need to make a prediction $\pi'_{i,i+1}$ for time $t_{i+1}$ using the derivation

$$(\pi_i \cup \pi'_{i-1,i}, t_i) \rightarrow_{Q,P}^* (\pi'_{i,i+1}, t_{i+1})$$

This line of reasoning underlies the following definition of a diagnosis using a sequence of observations:

**Definition 7.** Let $P = \langle O, S, < \rangle$ be a plan with a sequence $\text{Obs} = (\text{obs}(t_1) = (\pi_1, t_1), \ldots, \text{obs}(t_k) = (\pi_k, t_k))$ of observations, where $t_1 < t_2 < \ldots < t_k \leq \text{depth}(P)$. Then the qualification $Q \subseteq S$ is said to be a plan diagnosis of $P$ using $\text{Obs}$ iff there exist partial states $\pi'_{i,i+1}$ for $1 \leq i < k$ such that

1. $(\pi_1, t_1) \rightarrow_{Q,P}^* (\pi'_{1,2}, t_2)$,
2. $(\pi_i \cup \pi'_{i-1,i}, t_i) \rightarrow_{Q,P}^* (\pi'_{i,i+1}, t_{i+1})$ for every $2 \leq i < k$ and
3. $\pi_{i+1} \approx \pi_i$ for every $1 \leq i < k$.

By slightly changing this definition, we can make a closer connection between the definition of a diagnosis based on a sequence of observations and the definition of a diagnosis based on a pair of observations. To this end, given the sequence of observations $\text{Obs}$, the qualification $Q$ and Definition 7, we construct a new sequence of observations $(\text{obs}^*(t_1), \ldots, \text{obs}^*(t_k))$ as follows:

1. $\text{obs}^*(t_1) = \text{obs}(t_1)$;
2. for $i = 2, \ldots, k$, $\text{obs}^*(t_i) = (\pi_i \cup \pi'_{i,i}, t_i)$, where $(\pi_i', t_i)$ satisfies $\text{obs}^*(t_{i-1}) \rightarrow_{Q,P}^* (\pi_i', t_i)$.

Now we can establish the following connection between diagnosis based on a sequence of observations and diagnosis based on a pair of observations:

**Proposition 2.** $Q$ is a diagnosis of $P$ using $\text{Obs} = (\text{obs}(t_1), \ldots, \text{obs}(t_k))$ iff for $i = 1, \ldots, k - 1$, $Q$ is a diagnosis of the pair of observations $(P, \text{obs}^*(t_i), \text{obs}(t_{i+1}))$. 

14

76
Algorithm 2 Computing a mini-maxi diagnosis based on a sequence of observations

Require: a plan $P$ with a sequence $\text{Obs}$ of observations $\text{obs}(t_1) = (\pi_1, t_1), \ldots, \text{obs}(t_k) = (\pi_k, t_k)$ where $t_1 < t_2 < \ldots < t_k \leq \text{depth}(P)$.

Ensure: a mini-maxi diagnosis $\mathcal{Q}_{max}$.

1: Find a mini-maxi diagnosis $\mathcal{Q}_{max,1}$ for $(P, (\pi_{t_1}, t_1), (\pi_{t_2}, t_2))$ using Algorithm 1 and compute the predicted state $\mathcal{\pi}'_{max,1}$ using $\mathcal{Q}_{max,1}$;
2: $i := 2$;
3: while $i < k$ do
4: Find a mini-maxi diagnosis $\mathcal{Q}_{max,i}$ for $(P, (\pi_{t_i} \sqcup \mathcal{\pi}'_{max,i}, t_i), (\pi_{t_{i+1}}, t_{i+1}))$ using Algorithm 1 and compute the corresponding predicted state $\mathcal{\pi}'_{max,i+1}$ using $\mathcal{Q}_{max,i}$;
5: end while
6: return $\mathcal{Q}_{max} := \bigcup_i \mathcal{Q}_{max,i}$

It is not difficult to adapt the idea of mini-maxi diagnoses to a sequence of observations. To construct such a diagnosis $\mathcal{Q}_{max}$, it suffices to construct the separate qualifications $\mathcal{Q}_{max,1}, \ldots, \mathcal{Q}_{max,k}$ as follows:

Note that this algorithm makes use of Proposition 2 to compute the resulting mini-maxi diagnosis using an algorithm developed for diagnosis based on a pair of observations.

5 Conclusion

We have presented a simple formal framework to specify an executable plan and we have defined the notion of a diagnosis using partial observations of a plan in execution. We based our analysis of plans and observations upon a model-based diagnosis approach and considered a plan as a description of a system that can be observed and can be used to make predictions about its (future) behavior.

Using this framework, we derived a definition for a plan diagnosis as a set of abnormally qualified plan steps that are able to derive a partial plan state compatible with an observed partial plan state. In contrast to model-based diagnosis, where minimal and minimum diagnoses are aimed for, we have shown that minimality in plan diagnosis not always leads to the results we prefer. The reason is that making observations of plans is not completely comparable to making observations of input-output behavior of systems in model-based diagnosis. Often we make observations during plan execution and would like to make predictions of future outcomes of plan execution based on a plan diagnosis established so far. That implies that predictions about future behavior are as important as explanations of already observed behavior. In order to make powerful predictions, we argued that we should therefore aim at maximal informative diagnoses.

We showed that in contrast to minimum diagnosis, a minimal maximum informative diagnosis can be found efficiently, although maximum informative diagnoses of minimum size are difficult to compute.

Finally, we extended our approach to diagnosis with iterative observations, showing that in such cases both the general definition of what constitutes a diagnosis as well as the computation of maximum informative diagnoses can be reduced to their counter-
parts discussed for the simple case where only two successive observations are involved.

Current work can be extended in several ways. We mention three possible extensions: First of all, we could improve our current notion of diagnosis by taking into account the difference between plan operators and plan steps. In some cases it could be useful to make a distinction between establishing diagnoses at the plan step level and diagnoses at the plan operator level. For example, if instances of a driving action (i.e. plan steps) pertain to a plan operator that refers to the use of one single vehicle and all these instances are qualified as being abnormal, there is sufficient reason to believe that the vehicle itself (the plan operator) is faulty. Such a distinction requires the inclusion of causal rules linking different plan steps to each other. By means of such causal rules the number of plan steps qualified as abnormal often can be significantly reduced. Secondly, going beyond plan operators, we could improve the diagnostic model to include a model of the executing agent(s) that is involved in executing one or more plan steps. In particular we need to consider cases where the agent might evolve through several abnormal states. We suspect the resulting model to be related to diagnosis in Discrete Event Systems [4, 7]. Thirdly, we hope to extend our current approach by including methods for plan repair in the context of the inferred agent’s current (abnormal) state. Such methods especially seem to be useful in the context of iterative observations as discussed in the final part of this paper.

Acknowledgements This research is supported by the Technology Foundation STW, applied science division of the Dutch Science Foundation (NWO) and the Technology Programme of the Ministry of Economic Affairs. Project DIT5780: Distributed Model Based Diagnosis and Repair.

References

BAN Logic is *Not* ‘Sound’,
Constructing Epistemic Logics for Security is Difficult

Wouter Teepe

Department of Artificial Intelligence, University of Groningen, Grote Kruisstraat 2/1, 9712 TS Groningen, The Netherlands, email: wouter@teepe.com

Abstract. We show that BAN logic, an epistemic logic for analyzing security protocols, contains an inference rule that wrongly ascribes a certain property to cryptographic hash functions. This faulty inference rule makes the BAN logic not ‘sound’. That is, it is possible to derive counterintuitive beliefs which cannot be computationally justified. We will prove this in this paper. This result should count as a warning to those who wish to extend their BAN-descendant logic to one that captures ‘all’ cryptographic primitives.

1 Introduction: BAN Logic

Formal analysis of security protocols requires one to reason about the knowledge of the participants (principals) in the protocol. Critical for a security protocol is that it should not only guarantee that certain information is communicated, but also that certain other information is *not* communicated. For example, external observers should typically not be able to infer session keys which are exchanged in a security protocol.

BAN logic, introduced by Burrows, Abadi and Needham \[1, 2\] is an epistemic logic specially crafted for analyzing security protocols. It models at an abstract level the knowledge of the principals in a protocol. The principals are supposed to have only polynomially many computational resources. It was the first logic of its kind, and has had a tremendous influence on protocol analysis: it has helped revealing weaknesses in known protocols, and many logics are based on it. This is not to say that there has been no criticism of BAN logic. For one thing, a full semantics is lacking, and many attempts have been made to fix this problem \[3–10\]. Moreover, the logic fails to detect some very obvious protocol flaws \[11\].

Though the semantics of BAN logic is generally considered unclear, it is for our purposes important to note that BAN logic does have a partial semantics, which is defined over a part of the formal language of BAN logic.\(^1\)

The general consensus about BAN-descendant logics appears to be that these logics are computationally sound (detected protocol flaws are indeed flaws), but certainly not computationally complete (they may fail to detect certain protocol flaws).

\(^1\) This partial semantics is defined in Section 13 of the original BAN papers \[1, 2\].
flaws). Recent work includes attempts to bridge the gap between the formal (i.e., BAN-descendant) approach and the computational approach to security logics [12], and attempts to obtain completeness results for BAN-descendant logics in a kind of Kripke-style semantics [13, 14]. In the Multi-Agent Systems world, BAN logic has been widely used (see for example, [15])

In this paper we show a problem of BAN logic that has, to our knowledge, not yet been identified, despite all research into formal protocol analysis. BAN logic is not ‘sound’; false statements can be obtained by ‘valid’ inference rules from true assumptions. A questionable inference rule causes this behavior. In Sect. 2 we will explain the reasoning mistake behind this questionable inference rule and how this works out in BAN logic. Section 3 discusses the need for computational justification of inference rules, and the justification for the inference rule questioned in this paper. Section 4 shows the protocol we use in our unsoundness proof and Sect. 5 shows all inference rules used in our proof. Section 6 shows the actual proof. In Sect. 7 we will give an alternative proof, but in the questionable semantics of BAN logic, therefore, we regard our proof of Sect. 6 more important. We close with some remarks on the relevance of our results.

2 Cryptographic Hash Functions and Justified Beliefs

A cryptographic hash function is a function $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$ which is computationally feasible to compute, but for which the inverse is computationally infeasible. In particular, computing the inverse of a hash function takes $O(2^k)$ operations. Thus, a cryptographic hash function is one-way: it is computationally infeasible to construct a message $x$ such that $H(x)$ yields a given value $h$ [16].

Cryptographic hash functions have a lot of applications, including password protection, manipulation detection and the optimization of digital signature schemes. Unfortunately however, the class of applications is sometimes over-estimated. Consider for example the following quote from security expert Bruce Schneier [18, page 31]:

“If you want to verify someone has a particular file (that you also have), but you don’t want him to send it to you, then you ask him for the hash value. If he sends you the correct hash value, then it is almost certain that he has that file.”

Unfortunately, this claim is false. The problem is that in the above situation sketch, there is no mention that the file should be kept totally secret. If there is somebody who is willing to publish the hash value of the file, anybody can ‘prove’ possession of the file.

2 Cryptographic hash functions have more properties than the ones described here. For an extensive treatment, see [17].
The authors of BAN logic [1, 2] made the same reasoning mistake as Bruce Schneier, and incorporated into their logic an inference rule reflecting the above-mentioned questionable reasoning\(^3\). The name of the questionable rule is H and the rule will be shown in Sect. 5. As a result of this, BAN logic is not ‘sound’. Essential in our proof is the fact that belief in BAN logic is considered to be justified belief.

But first, let us recapitulate what soundness is. A proof procedure is sound if it proves only valid formulae. In particular, from true formulae it should be impossible to infer a false formula. A proof of soundness generally involves a formal system and a class of models (a semantics): a proof of soundness essentially shows that every formula that is derivable (\(\|-\)) in the formal system is observable (\(\|=\)) in all relevant models. Our proof of unsoundness does not require a model. Instead we rely on the definition of the modal operator belief (\(\equiv\)) in BAN logic which denotes true justified belief. As opposed to beliefs in general, which may be ungrounded and false, a true justified belief should be true. To see what the authors of BAN logic consider belief, let us look at the following excerpt from [19, page 7):

“More precisely, define knowledge as truth in all states (as in [20]\(^4\)): our notion of belief is a rudimentary approximation to knowledge, and it is simple to see that if all initial beliefs are knowledge then all final beliefs are knowledge and, in particular, they are true.”

In our ‘unsoundness’ proof, all initial beliefs are clearly knowledge, though one of the obtained final beliefs is not knowledge, in particular, it is false. Thus, by inferring an unjustified belief in BAN logic from true assumptions, we prove that BAN logic is not sound. In particular, this means that it is impossible to create a semantics in which BAN logic is sound.

### 3 On the Computational Justification of Beliefs

In the analysis of security protocols, if a principal obtains a new belief, there has to be a computational justification for the newly obtained belief. For example, if a principal sees a message cryptographically signed with private key \(K_1^{-1}\), it is justified to believe that the message originates from the principal owning private key \(K_1^{-1}\). The computational justification is in this case that it is computationally infeasible for principals other than the one owning private key \(K_1^{-1}\) to construct a message signed with this key. This type of justification is essential if security is of concern.\(^5\)

---

3 See Appendix A for a detailed discussion of the papers presenting BAN logic, and which papers exactly contain the reasoning mistake.

4 This is a reference to a preliminary paper. The final paper is [21] — WT.

5 Consider the alternative: we do not want principals to believe a message is sent by Santa Claus just because the name ‘Santa Claus’ is written beneath it; writing the name ‘Santa Claus’ is an exercise just as easy for Santa Claus himself as it is for anybody else.
With this consideration in mind, it is worth noting the following bit from page 266 of the BAN paper [1], (resp. pages 41–42 of [2]):

“Obviously, trust in protocols that use hash functions is not always warranted. If $H$ is an arbitrary function, nothing convinces one that when $A$ has uttered $H(m)$ he must have also uttered $m$. In fact, $A$ may never have seen $m$. This may happen, for instance, if the author of $m$ gave $H(m)$ to $A$, who signed it and sent it. This is similar to the way in which a manager signs a document presented by a subordinate without reading the details of the document. However, the manager expects anyone receiving this signed document to behave as though the manager had full knowledge of the contents. Thus, provided the manager is not careless and the hash function is suitable, signing a hash value should be considered the same as signing the entire message.”

This quote contains an assumption which is, in our opinion, unreasonable: The manager expects anyone receiving the signed document as though something would be the case which may not be the case. Of course, any principal including the manager may be free to desire any behavior from other principals. But is it reasonable to expect beliefs to be obtained which are not computationally justified?

It is reasonable to assume that any principal, upon seeing $\{H(N)\}_{K^{-1}}$ will believe the manager signed $H(N)$, since it is computationally too difficult for any principal other than the manager to construct the signature. However, it is not reasonable to assume that any principal, upon believing a manager signed $H(N)$, is willing to believe the manager has seen $N$, as there is no computational problem that would justify such a belief. Anybody may have computed $H(N)$ from $N$, in particular someone may have told the manager $H(N)$ not not $N$. Therefore, the expectation of a manager that other principals should act as if the manager knows $N$, is not warranted.

In fact, the text quoted above is the justification of the inference rule $H$ in BAN logic. We believe the identified problematic assumption explains the problems that arise from the inference rule $H$.

4 The Two Parrots Protocol

To prove the ‘unsoundness’ of BAN logic, we rely on a protocol. The rather simple two parrots protocol, shown in Fig. 1, will demonstrate the ‘unsoundness’. Alice (denoted $A$) chooses a random number $N$, sends it to Cecil (denoted $C$), who returns the number. Then Alice sends the cryptographic hash of the number to Bob (denoted $B$), and Bob signs this hash value and returns it to Alice. As Bob only sees the cryptographic hash value of $N$, and a cryptographic hash function is one-way, Bob does not learn $N$ itself. Nevertheless, if the two parrots protocol is analyzed in BAN logic, Alice will believe that Bob knows $N$.

Of course, Cecil might privately disclose $N$ to Bob, but this does not happen in the two parrots protocol. Thus, though by private channels Bob might learn $N$, the protocol certainly does not guarantee this.
In the two parrots protocol, the message $N$ is transmitted without protection. Thus, one can argue that Bob could learn $N$ by mere eavesdropping. For the sake of simplicity, we use a very simple protocol that suffices to demonstrate our observation on BAN logic. Of course, protection of $N$ can be achieved by encryption of the messages between Alice and Cecil. Our proof can be easily extended to obtain the same result for such an altered protocol. Moreover, our proof does not rely on Bob eavesdropping.

Thus, though Bob could learn $N$ through either an assistant (Cecil disclosing $N$ to Bob) or through eavesdropping, the communication in the two parrots protocol simply does not warrant Bob knowing $N$, and therefore also does not warrant Alice believing that Bob knows $N$.

When we want to formally analyze the protocol in BAN logic, we need to transcribe it into BAN logic. First, we have the protocol assumptions

$$A \equiv^K B, \quad A \equiv N, \quad A \equiv \sharp(N)$$

which state that $A$ knows the public key $K$ of $B$, $A$ knows $N$, and $A$ believes $N$ to be fresh. A newly generated random number is particularly fresh. Then, we have the protocol itself:

1. $S_1 : A \rightarrow C : N$
2. $S_2 : C \rightarrow A : N$
3. $S_3 : A \rightarrow B : H(N)$
4. $S_4 : B \rightarrow A : \{H(N)\}_{K^{-1}}$

This protocol description is rather straightforward. In general, the message $X$ cryptographically signed with the private key corresponding to public key $K$ is denoted as $\{X\}_{K^{-1}}$. Thus, any agent that knows $K$ can verify the signature and read $X$.

What is achieved by a protocol can be stated in claims. For the two parrots protocol, the following claim is true:

It will not be the case that $B \equiv N$

which essentially states that $B$ will not know $N$. Note that this is true because

1. $B$ only sees $H(N)$,
2. the inverse of $H(\cdot)$ is hard to compute ($H(\cdot)$ is a one-way function), and
3. $B$ has only polynomially many computational resources.

The problem that we identify in BAN logic (see Sect. 6) has the effect that the following statement can also be inferred in BAN logic:

$$A \equiv B \equiv N$$

which states that $A$ will believe that $B$ will know $N$. This belief of $A$ is not computationally justified (cf. Sect. 3).

5 Used Inference Rules

The proof of ‘unsoundness’ in Sect. 6 involves three inference rules of BAN logic$^{6}$:

1. the message meaning inference rule number $ii$ as given on page 238 of [1] (resp. page 6 of [2]):

$$\text{MM} \quad P \equiv K, \quad P \equiv \{X\}_{K^{-1}}$$

$$\frac{P \equiv Q \sim X}{P \equiv Q \sim X}$$

This rule formalizes that if $P$ knows $Q$’s public key, and $P$ receives a message $X$ signed with $Q$’s private key, $P$ may infer that $Q$ once sent $X.$

2. the hashing inference rule $H$ as given on page 266 of [1] (resp. page 42 of [2]):

$$\text{H} \quad P \equiv Q \sim H(X), \quad P \equiv Q \sim X$$

$$\frac{P \equiv Q \sim X}{P \equiv Q \sim X}$$

This rule is problematic, as it essentially infers belief (by $P$) of “possession” (by $Q$) of the message $X$ from $P$ believing that $Q$ once said $H(X).$ This rule leads to the ‘unsoundness’ of BAN logic. Fortunately, none of the authentication logics that descend from BAN logic, adopts the $H$ inference rule.

3. the nonce-verification inference rule as given on page 238 of [1] (resp. page 6 of [2]):

$$\text{NV} \quad P \equiv t(X), \quad P \equiv Q \sim X$$

$$\frac{P \equiv Q \sim X}{P \equiv Q \equiv X}$$

This rule formalizes that if $P$ believes $X$ to be fresh (it originates in the current session), and $P$ believes $Q$ once sent $X$, then $P$ may infer that $Q$ believes $X$ (in the current session).$^{8}$

$^{6}$ The names of these inference rules are given by the writer of this text.

$^{7}$ Inference rule MM has been questioned by Wedel and Kessler, as it is invalid if interpreted according to their semantics $^{[6]}$. However, they point out that it is unclear whether BAN logic itself or their semantics of BAN logic is to blame for that.

$^{8}$ This rule relies on the assumption that only beliefs are communicated.
6 Proof of ‘Unsoundness’ of BAN logic

In this section, we will present our formal proof. In our proof, we use the term “false belief”. This might be perceived as unnecessarily harsh or misleading, but we will argue that this is the right formulation, even in lack of a clear semantics of BAN logic as a whole. The central construct of BAN logic, $\equiv$, is defined as follows on page 236 of [1] (resp. page 4 of [2]):

"$P \equiv X$: $P$ believes $X$, or $P$ would be entitled to believe $X$. In particular, the principal $P$ may act as though $X$ is true. This construct is central to the logic."

In our proof, we obtain a result of the form $P \equiv X$, where $X$ is not warranted. It might be the case that $X$ were true, if some more communication were to occur than considered in our proof. Therefore, and in this way, we deem “false belief” the appropriate term for such an $X$. With this explanation given, let us formulate our main theorem:

**Theorem 1 (‘Unsoundness’ of BAN logic).** Within BAN logic [1, 2] it is possible to derive false beliefs from true premises.

**Proof (derivability).** Consider the two parrots protocol, whose BAN idealization is given in Sect. 4. It is trivial to verify that all of $A$, $C$ and $B$ are capable of sending the messages they ought to send in the two parrots protocol.

As a result of protocol step 2 ($S_2$), the following statement is inserted:

$$A \triangleleft N$$

(1)

As a result of protocol step 4 ($S_4$), the following statement is inserted:

$$A \triangleleft \{H(N)\}_{K-1}$$

(2)

Using inference rule MM, assumption $A \equiv^K B$ and (2), we can infer:

$$A \equiv B \vdash H(N)$$

(3)

Using inference rule H, (3) and (1), we can infer:

$$A \equiv B \vdash N$$

(4)

Using inference rule NV, assumption $A \equiv \sharp(N)$ and (4), we can infer:

$$A \equiv B \equiv N$$

(5)

Statement (5) should definitely not be derivable from the two parrots protocol. With all premises true and based on valid inferences, a false belief is established. More precisely, there exist no valid protocol annotations of the two parrots protocol which contain $B \equiv N$. \[\Box\]
The culprit is the inference rule $H$. This problem cannot be fixed by adding inference rules in such a way that $B \equiv N$ can be inferred, as this would thwart the definition of a cryptographic hash function: then $N$ would be derivable from $H(N)$. Such a ‘fix’ would increase the number of computationally unjustified inference rules from (at least) one to two.

Note that one more inference step is needed after application of the $H$ rule before a false belief is established. This is because we need to obtain belief of belief, which cannot be directly inferred from $H$.

7 The Semantic Approach

In the original BAN papers [1, 2], a rather limited semantics is given for a part of the formal language of BAN logic. This semantics has been subject to an enormous amount of criticism. For one thing, the semantics is very closely tied to the formal language of BAN logic: what is derivable in the logic is by definition observable in the semantics. One might even argue the semantics is so closely tied to the formal language that it is of no additional value. Except for it being the subject of criticism, the semantics has hardly ever been used.

In Sect. 6 we have explained why we used the formulation “false belief” in a proof that does not rely on any formal semantics. Therefore, we have consistently used quotes around the term unsoundness. In this section we will provide a proof based on a semantics: therefore, we may omit the quotes around unsoundness. However, for this proof we need to disregard all criticisms of the semantics of BAN logic. Therefore, we regard our proof in the previous section as more important. But it is of course to the reader to choose what he likes best:

1. to agree with our use of “false belief” in the previous section, and with it agree with the semantics-free proof of ‘unsoundness’ (shown in the previous section), or
2. to accept the semantics of BAN logic, regardless of all its shortcomings, and with it agree to our proof of unsoundness (shown in this section).

Before we show a run of the two parrots protocol in the semantics of BAN logic, it is appropriate to summarize this semantics:

- A local state of a principal $P$ is a tuple $(M_P, B_P)$, where $M_P$ is the set of messages seen ($\triangleleft$) by $P$, and $B_P$ is the set of beliefs ($\equiv$) of $P$. These sets enjoy closure properties which correspond to the inference rules of the logic. For compactness and ease of reading, we have only included elements in these sets which are relevant for our purposes.
- A global state $s$ is a tuple containing the local states of all principals. If $s$ is a global state, then $s_P$ is the local state of $P$ in $s$ and $M_P(s)$ and $B_P(s)$ are the corresponding sets of seen messages and beliefs. In our case the principals are $A$, $B$ and $C$, and a global state $s$ is the triple $(s_A, s_B, s_C)$.

Note that in BAN logic, the semantics of belief ($\equiv$) is defined, while the semantics of once said ($\triangleright$) is still “largely a mystery” (literal quote from [1, 2, 22]).
A run is a finite sequence of global states \(s_0, \ldots, s_n\).

A protocol run of a protocol of \(n\) steps of the form \((P_i \rightarrow Q_i : X_i)\) is a run of length \(n + 1\), where \(s_0\) corresponds to the protocol assumptions and where \(X_i \models M_{Q_i}(s_i)\) for all \(i\) such that \(0 < i \leq n\).

To be able to show a run of the two parrots protocol which is convenient to read, we will first name and give all local states. Then, we will give the full protocol run in which the names of these local states are used. For naming the local states, we adhere to the following convention: \(s_{0nn}^P\) is the local state of principal \(P\) in the global states \(nn\).

The local states of principals \(A\), \(B\) and \(C\) are as follows:

\[
\begin{align*}
\mathcal{M}_A & \quad \mathcal{B}_A \\
\text{s}_{0}^A & = (\emptyset, \{K \mapsto B, N, \sharp(N)\} ) \\
\text{s}_{2,3}^A & = (\{N\}, \{K \mapsto B, N, \sharp(N)\} ) \\
\text{s}_{4}^A & = (\{N, \{H(N)\}_{K-1}\}, \{K \mapsto B, N, \sharp(N), B \not\sim H(N), B \not\sim N, B \not\equiv N\} ) \\
\mathcal{M}_B & \quad \mathcal{B}_B \\
\text{s}_{0,1,2}^B & = (\emptyset, \emptyset ) \\
\text{s}_{3,4}^B & = (\{H(N)\}, \{H(N), \{H(N)\}_{K-1}\} ) \\
\mathcal{M}_C & \quad \mathcal{B}_C \\
\text{s}_{0}^C & = (\emptyset, \emptyset ) \\
\text{s}_{1,2,3,4}^C & = (\{N\} ) \\
\end{align*}
\]

The following is a run of the two parrots protocol:

\(s_0, s_1, s_2, s_3, s_4\) (7)

where \(s_i\) are the global states after the consecutive steps of the protocol:

\[
\begin{align*}
\text{s}_A & \quad \text{s}_B \quad \text{s}_C \\
\text{s}_0 & = (s_{0}^A \quad s_{0,1,2}^B \quad s_{0}^C ) \\
\text{s}_1 & = (s_{0,1}^A \quad s_{0,1,2}^B \quad s_{1,2,3,4}^C ) \\
\text{s}_2 & = (s_{2,3}^A \quad s_{0,1,2}^B \quad s_{1,2,3,4}^C ) \\
\text{s}_3 & = (s_{3}^A \quad s_{3,4}^B \quad s_{1,2,3,4}^C ) \\
\text{s}_4 & = (s_{4}^A \quad s_{3,4}^B \quad s_{1,2,3,4}^C ) \\
\end{align*}
\]

Now that we have specified a protocol run of the two parrots protocol, we can give our alternative proof of unsoundness:
Proof (observability). As shown in statement (5) of the derivability proof in Sect. 6, we can derive in BAN logic the sentence \( A \models B \models N \) in a run \( S_1, S_2, S_3, S_4 \) of the two parrots protocol. Thus, we have:

\[
S_1, S_2, S_3, S_4 \vdash A \models B \models N \tag{9}
\]

Global state \( s_4 \) corresponds to the semantics after a protocol run \( S_1, S_2, S_3, S_4 \) of the two parrots protocol. When we take the model as given in equations (6)–(8), we can observe that ‘A believes B knows N’: \( B \models N \in \mathcal{B}_A(s_4) \), which gives us:

\[
s_4 \models A \models B \models N \tag{10}
\]

On the other hand, we can also observe in our model that ‘B does not know N’: \( N \not\in \mathcal{B}_B(s_4) \), which gives us:

\[
s_4 \not\models B \models N \tag{11}
\]

Thus, the belief of A as given in (10) is not true as shown in (11). The false belief of A as given in (10), is nevertheless derivable (9). Thus, it is possible to derive a false belief within BAN logic.

Let us quote one last excerpt from Section 13, on page 269 of [1] (resp. pages 47–48 of [2]):

“Clearly, some beliefs are false. This seems essential to a satisfactory semantics. […] Most beliefs happen to be true in practice, but the semantics does not account for this coincidence. To guarantee that all beliefs are true we would need to guarantee that all initial beliefs are true.”

The existence of false beliefs in the semantics as such is not a problem, the problem is that these beliefs are derivable.

8 Discussion

The formal approach to protocol analysis essentially started with BAN logic. Many critiques of BAN logic have appeared, mentioning its incompleteness (i.e. inability to detect some obvious problems, cf. [11]) and its poor semantics (among many others, see [3]). Nevertheless, these critiques have not been a reason to abandon the way of thinking introduced by BAN logic [23]. The many augmentations to BAN logic (most notably, AT [3], GNY [4], AUTLOG [5, 6], VO [7], SVO [8, 9] and SVD [10]) show the trust in the formal approach which originates from BAN logic. In our opinion, this consensual trust in the way of thinking introduced by BAN logic is justified. While obtaining completeness has long been regarded as impossible, the soundness of BAN logic itself has never been seriously doubted. Wedel and Kessler identified rules in BAN, AT and GNY which are invalid in their semantics, but they point out that it is unclear whether the inference rules or their semantics are to blame for that [6]. Various more recent
results \[12–14, 24\] provide directions on how completeness could be obtained for formal protocol analysis.

Our unsoundness result does not at all invalidate the formal approach to protocol analysis. It should merely count as a warning to those who wish to complete their logic. All augmentations of BAN logic are incomplete in the sense that they do not accommodate all cryptographic primitives known to date. These logics are essentially ‘just big enough’ to capture the problems the authors intend to capture. And to be fair, this has been difficult enough already. Just a few BAN-descendant logics accommodate cryptographic hash functions, none of them accommodate fancy primitives like (to name just an example) oblivious transfer \[25\].

The fact that none of the hash-accommodating BAN-descendant logics adopts the \(H\) inference rule, can probably be explained by the observation that constructing a good logic is already so difficult that none of the authors will have felt the urge to include an inference rule into their logic that was not needed to capture the problem the author intended to capture. Nevertheless, it is remarkable that we are apparently the first to find this result on a paper which has been so extensively studied and is 17 years old.

So far, we know of only one publication which relies on the faulty \(H\) inference rule \[15\]. In this publication, the SET protocol\[10\] is analyzed in BAN logic. It remains open whether the authors’ assessment of SET holds in a BAN logic with the inference rule \(H\) omitted.

Acknowledgments

I would like to thank the three anonymous referees and Rineke Verbrugge for their enthusiasm, insightful comments and very useful suggestions.

References


\[10\] SET stands for Secure Electronic Transactions. The protocol was introduced by VISA and Mastercard for online payments, but it has never been widely adopted or deployed.
A Taxonomy of Versions of the BAN Paper

The seminal paper “A Logic of Authentication” has a respectable number of versions. Its precursor, “Authentication: A Practical Study in Belief and Action” was presented at the second conference on Theoretical Aspects of Reasoning About Knowledge in March 1988 [22, 18 pages]. Then, there is the DEC technical report, which was published in February 1989 and revised in February 1990 [2, 49 pages]. In April 1989, the work was submitted to the Royal Society of London, which published it in December 1989 [1, 39 pages]. Also in December 1989, a revised version of the article was presented on the twelfth ACM Symposium on Operating Systems Principles, which was also published in the ACM SIGOPS Operating Systems Review [26, 13 pages]. This led to a paper in the ACM Transactions on Computer Systems in February 1990 [27, 19 pages]. In May 1994, an appendix to the DEC technical report was published [19, 10 pages].

The most notable distinction between these versions is that in the ACM-published versions and the DEC appendix, the notation of many operators has changed from symbols (e.g., $≡$) to linguistic terms (e.g., believes). These versions refer to the DEC technical report for full reference. The DEC technical report and the Royal Society version [2, 1] should be considered the most complete versions, due to their size and the fact that these papers are most often used in self-references of the authors. Martín Abadi considers the Royal Society version the most definite one (on his homepage). These two versions of the article contain a Section 12, “On Hashing”, which introduces and discusses the inference rule essential in this paper. These two versions also contain a Section 13, “Semantics”, which defines the partial semantics for BAN logic, used in Sect. 7 of this paper.
A Hybrid Representation of Knowledge and Belief

Ricardo Wehbe

Institut für Informatik und angewandte Mathematik, Universität Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland. E-mail: wehbe@iam.unibe.ch.

Abstract. Agents working in an environment with incomplete information may need not only knowledge, but also beliefs to supplement their information. Default Logics have been frequently used to represent beliefs. Since inconsistent beliefs give rise to different extensions (scenarios), priorities are introduced to establish a preference among scenarios. We present a hybrid framework in which beliefs are represented by both monotonic and non-monotonic sets of clauses, avoiding thus the need for introducing priorities. Finally we give a glimpse of further work in process and comment briefly on the possibility of updating the database with new information.

1 Introduction

Agents working in an environment with incomplete information may need not only knowledge, but also beliefs to supplement their information. In this paper we present a representation of the knowledge and beliefs of one agent. Knowledge is monotonically represented as a set of clauses (similar to Logic Programming but without NAF and with classical negation) and beliefs as a combination of establishing a preference among scenarios. We present a hybrid framework in which beliefs are represented by both monotonic and non-monotonic sets of clauses, avoiding thus the need for introducing priorities.

The paper is organised as follows. In sections 2 and 3 we define respectively the syntax and semantics of sets of clauses, which represent the knowledge of an agent, and of sets of defaults, which represent its beliefs. In both sections we emphasize a constructive view of the semantics, providing ways to construct the knowledge and belief sets. In section 4 we combine the monotonic approach (i.e., adding clauses to the doxastic part of the database) with the non-monotonic approach. This way, the monotonic part provides a natural way of establishing preferences between extensions without having to use priorities. Section 5 concludes with a comment on further developments in progress.

2 Syntax and Semantics of Sets of Clauses

We assume a set of propositional symbols \( \Pi \). An atom will be either a propositional symbol of \( \Pi \) (a positive atom) or the negation of a propositional symbol
of $\Pi$ (a negative atom.) A sequent will be a (possibly empty) sequence of atoms, which will be denoted by Greek capital letters. Given a set $S$, the notation $\Gamma \in S$ will be a shorthand to denote that for all atoms $p$ in the sequence $\Gamma$, $p \in S$. We will use the notation $p, \neg p$ to refer to two complementary atoms when we do not specify which one is positive and which one is negative. Otherwise, we will write simply $p, \neg p$. The set of all atoms that can be formed with the propositional symbols in $\Pi$ will be denoted by $\text{AT}(\Pi)$. Given a set of propositional symbols $\Pi$, a normal clause (or a clause for short) on $\Pi$ has the form $\Gamma \Rightarrow p$, where $\Gamma$ is a sequent and $\Gamma, p \in \text{AT}(\Pi)$.

A clause with an empty sequent will be called a fact. We will omit the symbol $\Rightarrow$ when dealing with facts whenever no ambiguity arises.

If $\varphi$ is a set of clauses, we will denote by $\Pi_\varphi$ the set of propositional symbols occurring in $\varphi$. The Herbrand base of $\varphi$, denoted $H_\varphi$, is $\text{AT}(\Pi_\varphi)$.

The intended meaning of a clause $p_1, \ldots, p_n \Rightarrow q$ is that if the agent knows $p_1 \land \ldots \land p_n$, then the agent knows $q$. Facts are thus trivially true.

**Definition 1.** Let $\Pi$ be a set of propositional symbols and let $S \subseteq \text{AT}(\Pi)$. We say that $S$ is consistent iff it does not contain complementary atoms. It is maximally consistent with respect to $\Pi$ iff for each propositional symbol $p \in \Pi$, either $p \in S$ or $\neg p \in S$ but not both.

It is easy to see that if $S$ is a maximally consistent set with respect to $\Pi$, we cannot add an atom $q \in \text{AT}(\Pi)$ without making it inconsistent, unless it is already contained in it. Maximally consistent sets will also be called worlds.

**Definition 2.** Let $\varphi$ be a set of clauses and let $\Pi_\varphi$ be the set of propositional symbols that occur in $\varphi$. A model of $\varphi$ is a set $M \subseteq \text{AT}(\Pi_\varphi)$ that is maximally consistent with respect to $\Pi_\varphi$ such that for any normal clause $\Gamma \Rightarrow q \in \varphi$, if $\Gamma \in M$, then $q \in M$.

A set of clauses does not necessarily have a model. Take, for instance, the set consisting of two complementary facts.

**Definition 3.** Let $\varphi$ be a set of clauses. We say that $\varphi$ is consistent iff it has a model.

Now we turn to a more “operational” notion of semantics. The following definition introduces the important concept of invariant of a set of clauses.

**Definition 4.** Let $\varphi$ be a consistent set of clauses. The invariant of $\varphi$, denoted $\mathcal{J}(\varphi)$, is the intersection of all its models.

The invariant has some important properties. Since it contains the atoms that are true in all models, it may be used to represent the knowledge of an agent. We are interested in finding a way to construct this set.

**Definition 5.** Let $\varphi$ be a set of clauses. An answer set of $\varphi$ is a set $A \subseteq H_\varphi$ such that for any normal clause $\Gamma \Rightarrow p \in \varphi$, if $\Gamma \in A$, then $p \in A$. An answer set is minimal if there is no answer set that is a proper subset of it.
Since sets of clauses may be considered as propositional logic programs with classical negation, it is easy to see that the concept of minimal answer sets corresponds to the minimal model in those programs. Hence, every set of clauses has a minimal answer set (see for instance [5].) The difference is that minimal answer sets may be inconsistent. We will denote the minimal answer set of a set $\varphi$ of clauses by $A_{\varphi}$.

We will be interested only in consistent sets of clauses. We will now relate the notions of answer sets, invariants, and models.

**Definition 6.** Let $\varphi$ be a set of clauses. The operator $T_{\varphi} : 2^{H_{\varphi}} \rightarrow 2^{H_{\varphi}}$ is defined as follows: $q \in T_{\varphi}(S)$ if there is a clause $\Gamma \Rightarrow p \in \varphi$ and $\Gamma \in S$.

A well-known result of logical program theory is that the $T_{\varphi}$ operator has a minimal fixpoint and that this fixpoint is $A_{\varphi}$. The proof will be omitted. Besides, if we define:

- $T^0_{\varphi} = \emptyset$
- $T^{k+1}_{\varphi} = T_{\varphi}(T^k_{\varphi})$

Then, $A_{\varphi} = \bigcup T^k_{\varphi}$, and the fixpoint is reached after a finite number of iterations. For a proof, see [5].

Summing up, every set of clauses has a minimal answer set and this set can be computed within a finite number of steps. Of course, the existence of such minimal answer set does not guarantee the existence of a model. We will establish the relation between the minimal answer set, the invariant and the models.

**Lemma 1.** Let $\varphi$ be a consistent set of clauses. Then for any model $M$ of $\varphi$, $A_{\varphi} \subseteq M$.

**Proof.** Induction on the construction of $A_{\varphi} = \text{lfp}(T_{\varphi})$.

**Corollary 1.** Let $\varphi$ be a consistent set of clauses. Then $A_{\varphi} \subseteq J(\varphi)$.

**Proof.** By definition 4, $J(\varphi)$ is the intersection of all models $M$ of $\varphi$. The result follows from lemma 1.

The reason of the gap between the minimal answer set and the invariant is that there may be atoms that belong to any model, not because they are the consequent of a clause whose antecedent is true, but because this is the only possibility to avoid inconsistency, as shown next.

**Example 1.** In the following cases, we have atoms that must be included in all models of $\varphi$, although they may not be in $\text{lfp}(T_{\varphi})$.

1. If $\overline{p} \Rightarrow p \in \varphi$, then $p \in J(\varphi)$.
2. If $\overline{a} \Rightarrow a \in \varphi$ and $\overline{p} \Rightarrow \overline{a} \in \varphi$, then $p \in J(\varphi)$.
3. If $\overline{q} \Rightarrow p \in \varphi$ and $q \Rightarrow p \in \varphi$, then $p \in J(\varphi)$.
These cases illustrate the three basic cases in which atoms in the invariant may be not in \( \text{lfp}(T_\varphi) \). Adding ad-hoc rules to \( T_\varphi \) would be of no use, since these cases may appear “hidden” (for instance, the clauses \( p \Rightarrow a \) and \( \neg a \Rightarrow p \) are an instance of the first case.) In all three cases there occur complementary atoms. We will show that if no complementary atoms occur in \( \varphi \), then \( J(\varphi) = A_\varphi \).

**Definition 7.** Let \( \varphi \) be a consistent set of clauses. An atom \( p \in H_\varphi \) is bound in \( \varphi \) if \( \varphi \cup \{p\} \) is consistent and \( \varphi \cup \{\neg p\} \) is inconsistent. A proposition \( p \in \Pi_\varphi \) is free in \( \varphi \) if both \( \varphi \cup \{\neg p\} \) and \( \varphi \cup \{p\} \) are consistent.

**Lemma 2.** Let \( \varphi \) be a consistent set of clauses, let \( M \subseteq H_\varphi \). Then \( M \) is a model of \( \varphi \) iff for all \( p \in M \), it is a model of \( \varphi \cup \{p\} \).

**Proof.** Omitted.

**Lemma 3.** Let \( \varphi \) be a consistent set of clauses. Then \( J(\varphi) \) contains exactly the bound atoms of \( \varphi \).

**Proof.** Let us assume that \( p \) is bound in \( \varphi \). Thus \( \varphi \cup \{p\} \) is inconsistent. Then, by lemma 2 no model may include \( p \). Thus, \( p \in J_\varphi \).

Assume now that \( p \in J(\varphi) \). Then, \( p \) belongs to all models of \( \varphi \). Hence, \( \varphi \cup \{p\} \) is inconsistent, and thus \( p \) is bound in \( \varphi \).

**Lemma 4.** Let \( \varphi \) be a set of clauses such that the set of consequents of \( \varphi \) contains no complementary atoms. Then \( \varphi \) is consistent.

**Proof.** Let \( \Pi_\varphi \) the set of propositional symbols occurring in \( \varphi \). By hypothesis, the set of consequents of \( \varphi \) is consistent. Any extension of this set to a maximally consistent set with respect to \( \Pi_\varphi \) is a model of \( \varphi \).

**Lemma 5.** Let \( \varphi \) be a consistent set of clauses such that no complementary atoms occur in \( \varphi \). Then, any proposition \( p \) such that \( p, \neg p \not\in A_\varphi \) is free in \( \varphi \).

**Proof.** We must show that both \( \varphi \cup \{p\} \) and \( \varphi \cup \{\neg p\} \) are consistent. Assume \( \varphi \cup \{p\} \) is inconsistent. Since by lemma 4 \( \varphi \) is consistent, and \( \neg p \not\in A_\varphi \), there would be a clause whose consequent is \( \neg p \) and whose antecedent turns true by the addition of the fact \( p \). But since \( \varphi \) does not contain complementary atoms, it cannot contain such a clause. The same reasoning applies to \( \varphi \cup \{\neg p\} \).

**Proposition 1.** Let \( \varphi \) be a set of clauses with no occurrence of complementary atoms. Then \( A_\varphi = J(\varphi) \).

**Proof.** By lemma 5, if \( \varphi \) does not contain complementary atoms, all bound atoms in \( \varphi \) are in \( A_\varphi \). The result follows from corollary 1.

The restriction of sets of clauses by not allowing complementary clauses is too strong. We will show a constructive way to compute the invariant set of clauses even when it contains complementary atoms. Some new definitions will be needed first.
Definition 8. Let $\varphi$ be a consistent set of clauses and let $S \subseteq H_\varphi$. The reduction of $\varphi$ with respect to $S$, denoted by $R_S(\varphi)$, is the set of clauses constructed from $\varphi$ as follows:

1. Eliminate all clauses whose consequents are in $S$.
2. Eliminate all clauses containing in their antecedents atoms whose complements are in $S$.
3. Eliminate all other occurrences of atoms in $S$.
4. Rewrite all clauses $\Gamma, \overline{a} \implies \overline{q}$ where $a \in A_\varphi$ as $\Gamma \implies \overline{q}$.

Intuitively, if $S$ contains atoms which are taken to be true in a set of clauses, the reduction is the set of the clauses whose consequents are still unsolved.

Lemma 6. Let $\varphi$ be a consistent set of clauses and let $M$ be a model of $\varphi$. Then $M$ is a model of $R_{A_\varphi}(\varphi)$.

Proof. A clause of $R_{A_\varphi}(\varphi)$ is either:

1. A clause $\Gamma \implies p$ of $\varphi$. Then, if $\Gamma \in M$, then $p \in M$.
2. A clause $\Gamma' \implies p$ which is obtained from a clause $\Gamma \implies p$ of $\varphi$ by eliminating from $\Gamma$ all atoms that belong to $A_\varphi$. Then, since $A_\varphi \subseteq M$, we have eliminated atoms that are true in $\Gamma$. Thus, if $M$ is a model of the original clause, it is a model of the modified one.
3. A clause $\Gamma \implies p$ which corresponds to a clause $\Gamma, \overline{a} \implies p$ of $\varphi$. Since $a \in A_\varphi$, either $\Gamma \notin M$ or $p \in M$. In both cases, $M$ is a model of the modified normal clause.
4. A normal clause $\Gamma, \Sigma \implies p$ which corresponds to a clause $\Gamma, \overline{a}, \Sigma \implies p$ in $\varphi$. It is immediate that if $M$ is a model of the original normal clause, it is a model of the modified one.

Lemma 7. Let $\varphi$ be a consistent set of clauses. Then, any model of $R_{A_\varphi}(\varphi)$ which includes $A_\varphi$ is a model of $\varphi$.

Proof. Let $M$ be a model of $R_{A_\varphi}(\varphi)$ such that $A_\varphi \subseteq M$. Let us consider the following cases:

1. Let $\Gamma \implies p$ be a clause of $\varphi$ that was eliminated using rule 1. Then, since $p \in A_\varphi$, $M$ is a model of the clause.
2. Let $\Gamma \implies p$ be a clause of $\varphi$ that was eliminated using rule 2. Then, since there is at least some $\overline{p}$ in $\Gamma$ such that $p \in A_\varphi$, $M$ is a model of the clause.
3. Let $\Gamma \implies p$ be a clause of $\varphi$ that was rewritten using rule 3. Then, since the rewritten clause is stronger than the original one, if $M$ is a model of the former it is a model of the latter.
4. Let $\Gamma, \overline{a} \implies p$ be a clause of $\varphi$ that was rewritten using rule 4 yielding $\Gamma \implies p$. Then, if $\Gamma \in M$ then $p \in M$. Since $a \in A_\varphi$, it follows that $M$ is a model of the original clause.

Corollary 2. Let $\varphi$ be a consistent set of clauses. If $p$ is bound in $\varphi$, then either $p \in A_\varphi$ or $p$ is bound in $R_{A_\varphi}(\varphi)$. 
Proof. From lemmas 6 and 7, it is immediate that a set \( M \) is a model of \( \varphi \) iff it is a model of \( \mathcal{R}_{\mathcal{A}_\varphi}(\varphi) \) that includes \( \mathcal{A}_\varphi \). Thus, if \( p \) is bound in \( \varphi \) there are no models of \( \varphi \) which include \( \overline{\varphi} \). Thus there are no models of \( \mathcal{R}_{\mathcal{A}_\varphi}(\varphi) \) which include \( \overline{\varphi} \).

The preceding result is important because it allows the construction of the invariant of a set of clauses.

**Definition 9.** Let \( \varphi \) be a set of clauses. We define the operator \( \mathcal{T}_\varphi = \bigcup_k \mathcal{T}_\varphi^k \), where

\[
\mathcal{T}_\varphi^0 = \emptyset \quad \varphi_0 = \varphi \\
\mathcal{T}_\varphi^{k+1} = \mathcal{A}_{\varphi_k} \varphi_{k+1} = \mathcal{R}_{\mathcal{T}_\varphi^k}(\varphi_k)
\]

The operator \( \mathcal{T}_\varphi \) is the union of least fixpoints, since each term \( \mathcal{T}_\varphi^k \) is the least fixpoint of \( \mathcal{T}_\varphi \). The following proposition shows that the union has only finitely many elements.

**Proposition 2.** Let \( \varphi \) be a set of clauses. Then there is a finite natural number \( m \) such that \( m > n \) implies \( \mathcal{T}_\varphi^m = \mathcal{T}_\varphi^n \).

Proof. The proof (here omitted) is based on the fact that the reduction of a set of clauses has either less normal clauses or less atoms than the original set. Since we have always less atoms, the process cannot go on forever.

The problem is still what to do with the complementary atoms. It is possible that the \( \mathcal{T} \) operator does not find all bound atoms, as we saw in example 1. The strategy will be based on successive “splittings” of the set of clauses. For each pair of complementary atoms \( p, \overline{p} \) appearing in a set of clauses \( \varphi \), two separate sets will be considered, \( \varphi \cup \{ \Rightarrow p \} \) and \( \varphi \cup \{ \Rightarrow \overline{p} \} \). Some previous results will be needed. The following lemma is a straightforward extension of previous results.

**Lemma 8.** Let \( \varphi \) be a consistent set of clauses. Then:

1. If \( p \) is bound in \( \varphi \), either \( p \in \mathcal{T}_\varphi \) or \( p \) is bound in \( \mathcal{R}_{\mathcal{T}_\varphi} \).
2. Any model of \( \mathcal{R}_{\mathcal{T}_\varphi} \) that includes \( \mathcal{T}_\varphi \) is a model of \( \varphi \).

Proof. (Part 1) Induction on the construction of \( \mathcal{T}_\varphi \).

Base case: immediate, since \( \mathcal{T}_\varphi^0 = \emptyset \) and \( \varphi_0 = \varphi \).

Induction step: \( \mathcal{T}_\varphi^{k+1} = \mathcal{A}_{\varphi_k} \varphi_{k+1} = \mathcal{R}_{\mathcal{T}_\varphi^k}(\varphi_k) \). By corollary 2, we have that if \( p \) is bound in \( \varphi_k \), then either \( p \in \mathcal{T}_\varphi^{k+1} \) or \( p \) is bound in \( \varphi_{k+1} \).

(Part 2) Induction on the construction of \( \mathcal{T}_\varphi \).

Base case: immediate, since \( \mathcal{T}_\varphi^0 = \emptyset \) and \( \varphi_0 = \varphi \).

Induction step: \( \mathcal{T}_\varphi^{k+1} = \mathcal{A}_{\varphi_k} \varphi_{k+1} = \mathcal{R}_{\mathcal{T}_\varphi^k}(\varphi_k) \). By lemma 7, any model of \( \varphi_{k+1} \) that includes \( \mathcal{T}_\varphi^{k+1} \) is a model of \( \varphi_k \).

**Proposition 3.** Let \( \varphi \) be a set of clauses. Then, if \( \mathcal{R}_{\mathcal{T}_\varphi}(\varphi) \) contains no complementary atoms, either \( \mathcal{J}(\varphi) = \mathcal{T}_\varphi \) or \( \mathcal{T}_\varphi \) is inconsistent.
Proof. Assume first that \( \varphi \) is consistent. Observe that \( \mathcal{R}_{T_\varphi}(\varphi) \) contains no facts, because \( T_\varphi \) has reached a fixpoint. If it does not contain complementary atoms, we may construct two models of \( \mathcal{R}_{T_\varphi}(\varphi) \), one setting all atoms to true and the other one setting all atoms to false. Thus, for any propositional symbol \( p \) occurring in \( \mathcal{R}_{T_\varphi}(\varphi) \), both \( \mathcal{R}_{T_\varphi}(\varphi) \cup \{ p \} \) and \( \mathcal{R}_{T_\varphi}(\varphi) \cup \{ \neg p \} \) are consistent. By lemma 8, any atom that is bound in \( \varphi \) is either in \( T_\varphi \) or is bound in \( \mathcal{R}_{T_\varphi}(\varphi) \) If all atoms in \( \mathcal{R}_{T_\varphi}(\varphi) \) are free, then \( p \in T_\varphi \). Thus, \( J(\varphi) = T_\varphi \).

Assume now that \( \varphi \) is inconsistent. Since \( \mathcal{R}_{A_\varphi}(\varphi) \) contains no complementary atoms, it is consistent. By lemma 8, any model of \( \mathcal{R}_{A_\varphi}(\varphi) \) that includes \( T_\varphi \) is a model of \( \varphi \). Thus, \( T_\varphi \) must be inconsistent.

Corollary 3. Let \( \varphi \) be a consistent set of clauses.

1. An atom \( p \) is bound in \( \varphi \) iff for any atom \( q \) that is free in \( \varphi \), \( p \) is bound in \( \varphi \cup \{ q \} \) and in \( \varphi \cup \{ \neg q \} \).
2. An atom \( p \) is bound in \( \varphi \) iff for any atom \( q \) that is bound in \( \varphi \), then \( p \) is bound in \( \varphi \cup \{ q \} \).

Proof. (Part 1) Let \( p \) be bound in \( \varphi \). Then all models \( M \) of \( \varphi \) include \( p \). Since \( q \) is free, there are models including \( q \) and models including \( \neg q \). By lemma 2, the former are the models of \( \varphi \cup \{ q \} \) and the latter are the models of \( \varphi \cup \{ \neg q \} \). Since all of them include \( p \), it is bound in both sets of clauses. Now let \( p \) be bound in \( \varphi \cup \{ \Rightarrow q \} \) and in \( \varphi \cup \{ \Rightarrow \neg q \} \). Then all models of \( \varphi \) that include \( q \) include \( p \) and all models of \( \varphi \) that include \( \neg q \) include \( p \). Thus, \( p \) is bound in \( \varphi \).

(Part 2) Immediate from lemma 2.

Corollary 4. Let \( \varphi \) be an inconsistent set of clauses. Then either \( A_\varphi \) contains complementary atoms or \( \mathcal{R}_{A_\varphi}(\varphi) \) is inconsistent.

Proof. First observe that the atoms in \( A_\varphi \) do not occur in \( \mathcal{R}_{A_\varphi}(\varphi) \). Thus, they should be free therein. If \( \varphi \) is inconsistent, then there are no models of \( \mathcal{R}_{A_\varphi}(\varphi) \) containing \( A_\varphi \). Thus, all the complementary atoms in \( A_\varphi \) should be bound in \( \mathcal{R}_{A_\varphi}(\varphi) \), contradicting the fact that they are free.

Corollary 5. Let \( \varphi \) be an inconsistent set of clauses such that \( \mathcal{R}_{A_\varphi}(\varphi) \) contains no complementary atoms. Then \( A_\varphi \) contains complementary atoms.

Proof. Immediate from corollary 4.

We are ready to provide a procedure for the construction of the invariant of a set of clauses. In the next definitions we assume some ordering in the Herbrand base of a set of clauses.

Definition 10. Let \( \varphi \) be a set of clauses. We put \( \varphi_0 = \varphi \) and define:

- \( I(\varphi) = T_\varphi \) if \( \mathcal{R}_{T_\varphi}(\varphi) \) has no complementary atoms,
- \( I(\varphi) = T_\varphi \cup (I(\varphi_1) \cap I(\varphi_2)) \), where \( \varphi_1 = \mathcal{R}_{T_\varphi}(\varphi) \cup \{ p \} \), \( \varphi_2 = \mathcal{R}_{T_\varphi}(\varphi) \cup \{ \neg p \} \) and \( p, \neg p \) is the first pair of complementary atoms appearing in \( \mathcal{R}_{T_\varphi}(\varphi) \).
Note that the process is finite, since we eliminate one pair of complementary atoms in each step.

**Proposition 4.** Let $\varphi$ be a set of clauses. Then $\mathcal{J}(\varphi) = \mathcal{I}(\varphi)$.

**Proof.** Induction on the structure of $\mathcal{J}(\varphi)$.

1. **Base case:** If $\mathcal{R}_{T_\varphi}(\varphi)$ has no complementary atoms, then by lemma 3, $\mathcal{J}(\varphi) = \mathcal{I}(\varphi)$.

2. **Induction step:** If $\mathcal{R}_{T_\varphi}(\varphi)$ has complementary atoms $p$, $\neg p$, then by lemma 8, $\mathcal{J}(\varphi) = \mathcal{T}_\varphi \cup \mathcal{J}(\mathcal{R}_{T_\varphi}(\varphi))$. Let us suppose that $p$ is bound in $\mathcal{R}_{T_\varphi}(\varphi)$ and let $q$, $\neg q$ be two complementary atoms occurring in $\mathcal{R}_{T_\varphi}(\varphi)$. Then, either one of them is bound in $\mathcal{R}_{T_\varphi}(\varphi)$ or both are free. Suppose $q$ is bound. Then by corollary 3, part 2, we have that $p \in \mathcal{T}(\mathcal{R}_{T_\varphi}(\varphi) \cup \{q\})$, and that $\mathcal{R}_{T_\varphi}(\varphi) \cup \{\neg q\}$ is inconsistent. Thus, by induction hypothesis $\mathcal{J}(\mathcal{R}_{T_\varphi}(\varphi)) = \mathcal{I}(\mathcal{R}_{T_\varphi}(\varphi) \cup \{q\})$ and $\mathcal{J}(\mathcal{R}_{T_\varphi}(\varphi) \cup \{\neg q\}) = \mathcal{H}_\varphi$. Hence, $\mathcal{J}(\mathcal{R}_{T_\varphi}(\varphi)) = \mathcal{I}(\mathcal{R}_{T_\varphi}(\varphi) \cup \{q\}) \cap \mathcal{I}(\mathcal{R}_{T_\varphi}(\varphi) \cup \{\neg q\})$.

Assume now that both $q$ and $\neg q$ are free in $\mathcal{R}_{T_\varphi}(\varphi)$. By corollary 3, part one, we have that $p \in \mathcal{J}(\mathcal{R}_{T_\varphi}(\varphi))$ implies $p \in \mathcal{J}(\mathcal{R}_{T_\varphi}(\varphi) \cup \{q\})$ and $p \in \mathcal{J}(\mathcal{R}_{T_\varphi}(\varphi) \cup \{\neg q\})$. The result follows by induction hypothesis.

**Example 2.** Let $\varphi$ be

\[
q \Rightarrow p \\
q \Rightarrow \neg p \\
\neg q, s, t \Rightarrow r \\
\Rightarrow s \\
\Rightarrow t
\]

And assume an ordering \{p, $\neg p$, $\neg q$, r, $\neg r$, s, $\neg s$, t, $\neg t$\} in the Herbrand base.

We have:

\[
\mathcal{J}(\varphi) = \{s, t\} \cup \{(p, \neg p, r, \neg r, s, \neg s, t, \neg t) \cap \{\neg p, \neg q, r\}\} = \{\neg q, r, s, t\}
\]

Finally we mention some properties of invariants that will be useful later.

**Proposition 5.** Let $\varphi_1$ and $\varphi_2$ be two consistent sets of clauses such that $\varphi_1 \subseteq \varphi_2$. Then $\mathcal{J}(\varphi_1) \subseteq \mathcal{J}(\varphi_2)$.

**Proof.** Let $\mathcal{M}$ be a model of $\varphi_2$. Then $\mathcal{M} \setminus H_{\varphi_1}$ is a model of $\varphi_1$. Thus for all $p$, $p \in \mathcal{M}$ and $p \in H_{\varphi_1}$, implies $p \in \mathcal{J}(\varphi_1)$. Thus, $\mathcal{J}(\varphi_1) \subseteq \mathcal{J}(\varphi_2)$.

**Proposition 6.** Let $\varphi$ be a set of clauses. Then $\mathcal{J}(\varphi) = \mathcal{J}_\varphi(E \cup \mathcal{J}(\varphi))$.

**Proof.** Let $\mathcal{M}$ be a model of $\varphi \cup \mathcal{J}(\varphi)$. Then, it is a model of $\varphi$. Thus, the intersection of all models of $\varphi \cup \mathcal{J}(\varphi)$ is the intersection of all models of $\varphi$.

**Corollary 6.** Let $\varphi_1$ and $\varphi_2$ be two sets of clauses. Then $\mathcal{J}(\varphi_1 \cup \varphi_2) = \mathcal{J}(\varphi_1 \cup \varphi_2)$.

**Proof.** Take $\varphi = \varphi_1 \cup \varphi_2$. The result follows from lemmas 6 and 5.
3 A Non-Monotonic Representation of Belief

We will use clauses to represent knowledge. The invariant will represent the knowledge of an agent, in the sense that it consists of the atoms which are true in all possible worlds. We will distinguish knowledge from belief in the sense that knowledge is true, whereas belief may be false. This assumption, although usual, is rather strong: we have problems when several agents are concerned, since private communication might lead to inconsistency.

We will represent beliefs by defaults, which we define next.

Definition 11. Given a set of propositional symbols $\Pi$, a normal default (or a default for short) on $\Pi$ has the form $\Gamma : p \Rightarrow p$ where $\Gamma$ is a sequent and $\Gamma, p \in \text{AT}(\Pi)$. The sequent $\Gamma$ is the prerequisite of the default and $p$ is the justification of the default (left side) and the consequent of the default (right side.)

If $D$ is a set of defaults, then we will denote by $\text{CONS}(D)$ the set of consequences of $D$.

Strictly speaking, we will use a subset of defaults, namely the so-called normal defaults [6], [3]. In normal defaults, the consequent is the justification; general defaults allow arbitrary sequents as justifications. Since we will use only normal defaults, we call them simply “defaults.”

The intended meaning of a default $\Gamma : p \Rightarrow p$ is that if $\Gamma$ is true and $p$ is not inconsistent with the knowledge the agent has, then it will be taken to be true. Of course, it will not be in the same level as the atoms of the invariant; the latter will be known; the former will just be believed. The use of defaults implies that the representation is no longer monotonic.

Definition 12. Let $\Pi$ be a set of propositions. A knowledge and belief database (KB-database for short) is a pair $\zeta = (E, D)$ where $E$ is a set of clauses and $D$ is a set of defaults on $\Pi$. The set of clauses $E$ is the epistemic part of the database and the set $D$ is the doxastic part of the database. A KB-database where $E$ is consistent will be said to be epistemically consistent.

As before, given a KB-database $\zeta$, we will denote by $\Pi_\zeta$ the set of propositional symbols occurring in the epistemic and the doxastic parts of $\zeta$. The Herbrand base will be defined in the same way as for sets of clauses: the Herbrand base of $\zeta$, denoted by $H_\zeta$, is $\text{AT}(\Pi_\zeta)$.

As usual, semantics of sets of defaults will be based on the concept of extensions. Informally speaking, an extension is the set of beliefs we may form starting from a KB-Database. We point out that when we apply defaults, we assume some external circumstances, expressed as a set of atoms. This will be called a context. Now we define the belief sets we may form in the presence of a context. This will lead to the formal definition of extensions.

Definition 13. Let $\zeta = (E, D)$ be a KB-database and let $S \subseteq H_\zeta$. Then $\Lambda_\zeta(S)$ is defined as the smallest set such that the following properties are fulfilled:
1. $\mathcal{J}(E) \subseteq \Lambda_\zeta(S)$
2. If $p \in \mathcal{J}(E \cup \Lambda_\zeta(S))$ then $p \in \Lambda_\zeta(S)$
3. If $\Gamma : p \Rightarrow p \in D$ and $\Gamma \in \Lambda_\zeta(S)$ and $\exists \not\in S$, then $p \in \Lambda_\zeta(S)$

Informally, $\Lambda_\zeta(S)$ is the minimal set of beliefs that an agent whose KB-database is $\zeta$ may have in view of the context $S$. The set $\Lambda_\zeta(S)$ may be inconsistent, as the following example shows.

Example 3. Let $\zeta = (\emptyset, \{p \Rightarrow p, \neg p \Rightarrow \neg p\})$. Then, $p, \neg p \in \Lambda_\zeta(\emptyset)$.

A natural way to avoid this problem is to take the fixpoint.

Definition 14. Let $\zeta = (E, D)$ be a KB-Database. A set $S \subseteq H_\zeta$ is an extension for $\zeta$ iff $S$ is a fixpoint of $\Lambda_\zeta$, i.e., $S = \Lambda_\zeta(S)$.

The concept of extensions may seem elusive. We will give several characterisations of it.

Proposition 7. Let $\zeta = (E, D)$ be a KB-database and let $S$ be an extension for $\zeta$. Then $S$ is inconsistent iff $E$ is inconsistent.

Proof. It is trivial that if $E$ is inconsistent, then $S$ is inconsistent for the first condition of the definition of $\Lambda_\zeta(S)$. First we show that if $S$ is inconsistent, then $S \subseteq \mathcal{J}(E)$. It is clear that the first and the second conditions of the definition of $\Lambda_\zeta(S)$ are fulfilled by $\mathcal{J}(E)$, since $\mathcal{J}(E) = \mathcal{J}(E \cup \mathcal{J}(E))$. Besides, the third condition is also fulfilled, since all justifications of defaults belong to $S$. Thus, $\Lambda_\zeta(S) \subseteq \mathcal{J}(E)$ by the minimality of $\Lambda_\zeta(S)$. Since $S$ is inconsistent, so must be $\mathcal{J}(E)$.

Corollary 7. Let $\zeta$ be a KB-database. If $S$ has an inconsistent extension, then it has no other extension.

Proof. Since all extensions for $\zeta$ include $\mathcal{J}$ and this invariant is inconsistent by proposition 7, then all extensions must also be inconsistent.

Proposition 7 states that the addition of defaults may not turn a KB-database inconsistent, unless it is epistemically inconsistent.

Now we relate the notion of extensions to that of invariants. We need some more notation first.

Definition 15. Let $\zeta = (E, D)$ be a KB-database and let $S$ be an extension for $\zeta$. Then the set of generating defaults for $S$ in $\zeta$, denoted by $GD_\zeta(S)$, is the set $GD_\zeta(S) = \{\Gamma : p \Rightarrow p \in D \mid \Gamma \in S \text{ and } \not\in S\}$

Lemma 9. Let $\zeta = (E, D)$ be a KB-database and let $S$ be an extension for $\zeta$. Then $S = \mathcal{J}(E \cup \text{CONS}(GD_\zeta(S)))$. 
Proof. Let \( p \in \text{CONS} \left( \text{GD}_\zeta(S) \right) \). Then, there is a default \( \Gamma : p \Rightarrow p \in D \) such that \( \Gamma \in S \) and \( \overline{p} \notin S \). Thus, \( p \in S \). Therefore, \( \text{CONS} \left( \text{GD}_\zeta(S) \right) \subseteq S = \Lambda_\zeta(S) \) and thus \( \mathcal{J}(E \cup \text{CONS} \left( \text{GD}_\zeta(S) \right)) \subseteq S \).

Now let \( \Phi \) be an abbreviation for \( \text{CONS} \left( \text{GD}_\zeta(S) \right) \). We will show that \( \mathcal{J}(E \cup \Phi) \) includes \( \Lambda_\zeta(S) \).

On the one hand, we have by proposition 5 that \( \mathcal{J}(E) \subseteq \mathcal{J}(E \cup \Phi) \). On the other hand, by proposition 6, \( \mathcal{J}(E \cup \mathcal{J}(E \cup \Phi)) = \mathcal{J}(E \cup \Phi) \).

Assume now that there is a default \( \Gamma : p \Rightarrow p \in D \) such that \( \Gamma \in \mathcal{J}(E \cup \Phi) \) and \( \overline{p} \notin S \). Since \( \mathcal{J}(E \cup \Phi) \subseteq S \), then \( \Gamma \in S \). Thus, \( \Gamma : p \Rightarrow p \in \text{GD}_\zeta(S) \). Therefore, \( p \in \mathcal{J}(E \cup \Phi) \). Thus all three conditions of the definition of \( \Lambda_\zeta(S) \) are fulfilled and \( \Lambda_\zeta(S) \subseteq \mathcal{J}(E \cup \Phi) \). The result follows immediately.

The following lemma states an important property of defaults.

Lemma 10. Let \( \zeta_1 = (E, D_1) \) and \( \zeta_2 = (E, D_2) \) be two KB-databases such that \( D_1 \subseteq D_2 \) and let \( S_1 \) be an extension for \( \zeta_1 \). Then there is an extension \( S_2 \) for \( \zeta_2 \) such that \( S_1 \subseteq S_2 \).

Proof. We construct first the sequence of sets of defaults \( \Delta_0, \ldots, \Delta_m \) such that:

- \( \Delta_0 = \text{GD}_\zeta(S_1) \)
- \( \Delta_{k+1} = \{ \Gamma : p \Rightarrow p \in D_2 \mid \Gamma \in \bigcup_{i=0}^{k} \text{CONS}(\Delta_i) \) and \( \overline{p} \notin \bigcup_{i=0}^{k+1} \text{CONS}(\Delta_i) \} \)

The sequence is finite, since the set of defaults is finite. Now let us define the abbreviation \( \Sigma = \mathcal{J}(E \cup \bigcup_{i=0}^{m} \text{CONS}(\Delta_i)) \). We show first that \( \Lambda_\zeta(\Sigma) \subseteq \Sigma \).

It is clear that \( \mathcal{J}(E) \subseteq \Sigma \). Besides, by proposition 6, \( \mathcal{J}(E \cup \Sigma) = \mathcal{J}(\Sigma) \).

Consider now a default \( \Gamma : p \Rightarrow p \in D_2 \), with \( \Gamma \in \Sigma \) and \( \overline{p} \notin \Sigma \). Then there is some \( k \) such that \( \Gamma \in \mathcal{J}(E \cup \left( \bigcup_{i=0}^{k} \text{CONS}(\Delta_i) \right)) \), and \( \overline{p} \notin \mathcal{J}(E \cup \left( \bigcup_{i=0}^{k} \text{CONS}(\Delta_i) \right)) \). Hence, \( \Gamma : p \Rightarrow p \in \Delta_{k+1} \) and \( p \in \Sigma \). Thus, by the minimality of \( \Lambda_\zeta(\Sigma) \) we get \( \Lambda_\zeta(\Sigma) \subseteq \Sigma \).

Now assume that \( \Lambda_\zeta(\Sigma) \neq \Sigma \). Then there is some \( k \) such that \( \mathcal{J}(E \cup \left( \bigcup_{i=0}^{k} \text{CONS}(\Delta_i) \right)) \subseteq \Lambda_\zeta(\Sigma) \), even though \( \mathcal{J}(E \cup \left( \bigcup_{i=0}^{k+1} \text{CONS}(\Delta_i) \right)) \not\subseteq \Lambda_\zeta(\Sigma) \). Thus there is a default \( \Gamma : p \Rightarrow p \in D_2 \) such that \( p \in \Sigma \) and \( p \notin \Lambda_\zeta(\Sigma) \). Hence \( \Gamma \in \mathcal{J}(E \cup \left( \bigcup_{i=0}^{k+1} \text{CONS}(\Delta_i) \right)) \) and \( \overline{p} \notin \mathcal{J}(E \cup \left( \bigcup_{i=0}^{k+1} \text{CONS}(\Delta_i) \right)) \). But then, \( \Gamma \in \Lambda_\zeta(\Sigma) \). Besides, \( \overline{p} \notin \Sigma \) and thus \( p \) must be in \( \Lambda_\zeta(\Sigma) \), which contradicts the hypothesis.

Finally, we have that by construction \( \mathcal{J}(E \cup \text{CONS} \left( \text{GD}_\zeta(S_1) \right)) \subseteq \Sigma \) and by lemma 9 we get the result.

The last lemma shows a property that is sometimes called “semimonotonicity” [3]. It is also important because it leads to the following result.

Corollary 8. Let \( \zeta = (E, D) \) be a KB-database and let \( E \) be a consistent set of clauses. Then \( \zeta \) has an extension.

Proof. Since \( E \) is consistent, then clearly the KB-Database \( (E, \emptyset) \) has an extension, namely \( \mathcal{J} \). Then by lemma 10, it follows that \( \zeta \) has an extension.
We have seen that a KB-database that is epistemically consistent has at least one extension and in general a family of extensions. Now we will characterise the families of extensions that a KB-database may have.

**Lemma 11.** Let \( \zeta = (E, D) \) be a KB-database and let us define:

- \( S^0 = \mathcal{J}(E) \)
- \( S^{k+1} = \mathcal{J}(E \cup \{\text{CONS}(\Gamma : p \Rightarrow p \in D) \mid \Gamma \in S^k \text{ and } \overline{p} \notin S}\) 

Then \( S \) is an extension for \( \zeta \) iff \( S = \bigcup_i (S^i) \).

**Proof.** First we show that \( \Lambda_\zeta(S) \subseteq \bigcup_i (S^i) \). We have that \( \mathcal{J}(E) \subseteq \bigcup_i (S^i) \) and by proposition 6 we have that \( \mathcal{J}(E \cup \bigcup_i (S^i)) = \mathcal{J}(\bigcup_i (S^i)) \). Now let us consider a default \( \Gamma : p \Rightarrow p \in D \) such that \( \Gamma \in \bigcup_i (S^i) \) and \( \overline{p} \notin \bigcup_i (S^i) \). Then \( p \in \bigcup_i (S^i) \).

Now we show that \( \bigcup_i (S^i) \subseteq \Lambda_\zeta(S) \). Assume there is some \( p \in \bigcup_i (S^i) \) such that \( p \notin \Lambda_\zeta(S) \). Then there is some \( k \) such that \( S^k \subseteq \Lambda_\zeta(S) \) although \( S^{k+1} \not\subseteq \Lambda_\zeta(S) \). Thus, there must be a default \( \Gamma : p \Rightarrow p \in D \) such that \( \Gamma \in S^k \), \( \overline{p} \notin S \) and \( p \notin \Lambda_\zeta(S) \). Since \( S^k \subseteq \Lambda_\zeta(S) \), then \( p \) must be in \( \Lambda_\zeta(S) \) contradicting thus the hypothesis.

**Lemma 12.** Let \( \zeta = (E, D) \) be a KB-database and let \( S_1 \) and \( S_2 \) be two extensions for \( \zeta \) such that \( S_1 \subseteq S_2 \). Then \( S_1 = S_2 \).

**Proof.** By lemma 11 we have: \( S_1 = \bigcup_i (S^i_1) \) and \( S_2 = \bigcup_i (S^i_2) \). The lemma is proved by induction on \( k \).

Base case: \( S^0 = S^0_1 = \mathcal{J}(E) \)

Induction step: \( S^{k+1}_2 = \mathcal{J}(E \cup \{\text{CONS}(\Gamma: p \Rightarrow p \in D) \mid \Gamma \in S^k \text{ and } \overline{p} \notin S_2}\) 

By induction hypothesis, \( S^k = S^k_2 \) and, since \( S_1 \subseteq S_2 \), then \( \overline{p} \notin S_1 \). Hence \( S^{k+1}_1 = S^{k+1}_2 \).

**Lemma 13.** Let \( \zeta = (E, D) \) be a KB-database and let \( S_1, S_2 \) be two distinct extensions for \( \zeta \). Then \( S_1 \cup S_2 \) is inconsistent.

**Proof.** If \( S_1 \) and \( S_2 \) are extensions, we have by lemma 10 that \( S_1 = \bigcup_i S^i_1 \) and \( S_2 = \bigcup_i S^i_2 \). We have also by lemma 9 that \( S_1 = \mathcal{J}(E \cup \text{CONS}(\text{GD}_\zeta(S_1})) \) and \( S_2 = \mathcal{J}(E \cup \text{CONS}(\text{GD}_\zeta(S_2))) \). If we take \( \Delta = \text{GD}_\zeta(S_1) \cap \text{GD}_\zeta(S_2) \), then we have that \( \text{GD}_\zeta(S_1) \setminus \Delta \neq \emptyset \) and \( \text{GD}_\zeta(S_2) \setminus \Delta \neq \emptyset \), since otherwise we would be in the case of lemma 12. Let us suppose that there is an ordering of the defaults of \( D \) and that the first \( k \) defaults are those in \( \Delta \). Let thus \( \delta_{k+1} \) be the first default such that \( \delta_{k+1} \in \text{GD}_\zeta(S_1) \setminus \Delta \). Then we have that \( S^{k+1}_1 = S^{k+1}_2 \) and \( S^{k+1} = \mathcal{J}(E \cup \{\text{CONS}(\Gamma: p \Rightarrow p \in D) \mid \Gamma \in S^k \text{ and } \overline{p} \notin S_1\}) \).

Since \( \Gamma \in S^k_1 \), \( \Gamma \in S^k_2 \) by the construction of the sequence. Hence, if \( p \notin S_2 \), the only possibility is that \( \overline{p} \in S_2 \). Thus \( S_1 \cup S_2 \) contains complementary atoms.

The process to construct the extensions of a KB-database \( \zeta = (E, D) \) will be a “naïve” one, similar to one the proposed in [6]. Starting from the invariant \( \mathcal{J}(E) \), we apply one default each time until no further defaults are applicable. This process terminates, since the set of defaults is finite. The question is, whether this process is sound (i.e., if it yields an extension) and whether it is complete (i.e., if all extensions can be obtained this way.)
Definition 16. Let $\xi = (E, D)$ be a KB-database. Then we define the operator $\chi(\xi) = \bigcup_i \chi^i(\xi)$ as follows:

$$\chi^0(\xi) = \mathcal{J}(E)$$

- $\chi^{k+1}(\xi) = \begin{cases} 
\mathcal{J}(E \cup \chi^k(\xi) \cup \{p\}) & \text{if there is a default } \Gamma : p \Rightarrow p \in D \text{ such that } \Gamma \in \mathcal{J}(\chi^k(\xi)) \\
\chi^k(\xi) & \text{otherwise}
\end{cases}$$

Note that this definition describes a family of sets rather than one.

Lemma 14. Let $\xi = (E, D)$ be a KB-Database. Then if $E$ is consistent, so is $\chi(\xi)$.

Proof. Induction on the structure of $\chi(\xi)$.

Base case: immediate, since $\chi^0(\xi) = \mathcal{J}(E)$ and $E$ is consistent by hypothesis.

Induction step: $\chi^{k+1}(\xi) = \mathcal{J}(E \cup \chi^k(\xi) \cup \{p\})$ where there is a default $\Gamma : p \Rightarrow p \in D$ such that $\Gamma \in \mathcal{J}(\chi^k(\xi))$ and $p$ is free in $E \cup \chi^k(\xi)$. By induction hypothesis, $\chi^k(\xi)$ is consistent and by proposition 6, so is $E \cup \chi^k(\xi)$. Thus, since $p$ is free in $E \cup \chi^k(\xi)$, then $E \cup \chi^k(\xi) \cup \{p\}$ is consistent.

Proposition 8. Let $\xi = (E, D)$ be a KB-Database and let $E$ be consistent. Then $\chi(\xi)$ is an extension for $\xi$.

Proof. We prove first that $\Lambda_\xi(\chi(\xi)) \subseteq \chi(\xi)$. We have that $\mathcal{J}(E) \subseteq \chi(\xi)$. Besides, by proposition 6 we have that $\mathcal{J}(E \cup \chi(\xi)) = \chi(\xi)$. Now consider a default $\Gamma : p \Rightarrow p \in D$ such that $\Gamma \in \chi(\xi)$ and $\overline{p} \not\in \chi(\xi)$. Then there is some $k$ such that $\Gamma \subseteq \chi^k(\xi)$ and also $\overline{p} \not\in \chi^k(\xi)$. Thus, $p \in \chi^{k+1}(\xi) \subseteq \chi(\xi)$. Hence, $\Lambda_\xi(\chi(\xi)) \subseteq \chi(\xi)$.

Assume now that $\chi(\xi) \not\subseteq \Lambda_\xi(\chi(\xi))$. Since $\chi^0(\xi) \subseteq \Lambda_\xi(\chi(\xi))$ and for all $j \leq 0$ $\chi^j(\xi) \subseteq \chi^{j+1}(\xi)$, we have that there must be some $k$ such that $\chi^k(\xi) \subseteq \Lambda_\xi(\chi(\xi))$ even though $\chi^{k+1}(\xi) \not\subseteq \Lambda_\xi(\chi(\xi))$. Then there is some default $\Gamma : p \Rightarrow p \in D$ such that $\Gamma \in \chi^k(\xi)$ and $p$ is free in $\chi^k(\xi)$. But then, since $p \not\in \Lambda_\xi(\chi(\xi))$, then $\overline{p} \in \Lambda_\xi(\chi(\xi))$ and since $\Lambda_\xi(\chi(\xi)) \subseteq \chi(\xi)$, then $\overline{p} \not\in \chi(\xi)$. Hence, $\chi(\xi)$ would be inconsistent, contradicting thus lemma 14.

We have thus that the application of the operator $\chi(\xi)$ indeed yields an extension for $\xi$. The question is now whether all extensions may be found with some application of this operator.

Proposition 9. Let $\xi = (E, D)$ be a KB-Database. Then for any extension $S$ for $\xi$, there is a set $\chi(\xi)$ such that $\chi(\xi) = S$.

Proof. Let $S$ be an extension for $\xi$. We define $\chi(\xi)$ such that the choice of the defaults is restricted to defaults whose consequents are in $S$. We show that $\chi(\xi) \subseteq S$ by induction on the construction of $\chi(\xi)$.

Base case: $\chi^0(\xi) = \mathcal{J}(E) \subseteq S$. 

Induction step: let $q \in \chi^{k+1}(\zeta)$. Then $q \in J(E \cup \chi^k(\zeta) \cup \{\Rightarrow q\})$, where $\Gamma : q \Rightarrow q \in D$ and $\Gamma \in S^k$ and $q$ is free in $E \cup \chi(\zeta)$. Besides, we have imposed the condition $q \in S$. Thus, using the induction hypothesis, we get that $\chi^{k+1}(\zeta) \subseteq S$.

Now we show that $S \subseteq \chi(\zeta)$. Since $S$ is an extension, $S = \bigcup S^i$. We prove the inclusion by induction on $i$.

Base case: $S^0 = J(E) \subseteq \chi(\zeta)$.

Induction step: $S^{k+1} = J(E \cup \{\text{CONS}(\Gamma : p \Rightarrow p \in D) \mid \Gamma \in S^k$ and $\overline{p} \notin S\})$

Since $\chi(\zeta) \subseteq S$, $\overline{p} \notin S$ implies $\overline{p} \notin \chi(\zeta)$. Besides, by induction hypothesis we have that $S^k \subseteq \chi(\zeta)$; hence, $\Gamma \in \chi(\zeta)$ and thus $\text{CONS}(\Gamma : p \Rightarrow p) \subseteq \chi(\zeta)$.

Therefore, $\chi(\zeta) = S$.

Summing up: we have the representation of the knowledge and beliefs of an agent by a KB-database $\zeta = (E, D)$, where $E$ is a (monotonic) set of clauses that represents knowledge and $D$ is a set of defaults that represents the beliefs of the agent. We have a constructive way to obtain the invariant $J(E)$ and the extensions of $D$ by means of the $\chi(\zeta)$ operator. The following example shows that this representation may not correspond to the intuition of what a belief is.

Example 4. Let $\zeta$ be

\[
\Rightarrow p \\
q \Rightarrow r \\
p : q \Rightarrow q \\
: \neg q \Rightarrow \neg q \\
r : s \Rightarrow s
\]

We have here two extensions: $\text{Ext}_1 = \{p, q, r, s\}$ and $\text{Ext}_2 = \{p, \neg q\}$.

In the example we cannot say that the agent “believes” either $q$ or $\neg q$; it rather takes these possible scenarios under consideration. If the agent believes $q$, then some kind of priority must be given to extension $E_1$. One possibility is to add some extra-logical features, such as priorities [6]. In the next section we will combine the monotonic and the non-monotonic approaches.

4 A Hybrid Representation of Belief

The representation of belief through defaults does not correspond exactly to the usual notion of beliefs. For instance, lemma 13 allows different and incompatible sets of beliefs (as a matter of fact, if two belief sets are different, they must be incompatible.) This approach corresponds to the consideration of possible scenarios rather than a set of possible states that is considered more plausible than other.

A possible approach to establish some hierarchy between possible worlds is to give priorities to the defaults. Another one, which is the one we propose here, is to combine the monotonic and the non-monotonic approach. In this case, we re-define a KB-Database as follows:
**Definition 17.** Let \( \Pi \) be a set of propositions. A hybrid knowledge and belief database (HKB-database for short) is a triple \( \zeta = (E, D_{cl}, D_{def}) \) where \( E \) and \( D_{cl} \) are sets of clauses and \( D_{def} \) is a set of defaults on \( \Pi \). The set of clauses \( E \) is the epistemic part of the database and the sets \( D_{cl} \) and \( D_{def} \) constitute the doxastic part of the database. An HKB-database where \( E \) is consistent is epistemically consistent. An HKB-database where both \( E \) and \( D_{cl} \) are consistent is doxastically consistent.

Note that doxastic consistency implies the consistency of both \( E \) and \( D_{cl} \) and not only the consistency of \( D_{cl} \). This is because everything that is known is also believed. The knowledge of an agent is defined as \( J(E) \) (as before) and the belief of an agent are defined as the extensions constructed starting from \( \chi_0 = J(E \cup D_{cl}) \). In this way, only the extensions of \( \zeta \) that are consistent with \( J(E \cup D_{cl}) \) are taken into account.

**Example 5.** Let \( \zeta = (E, D_{cl}, D_{def}) \) be

\[
\begin{align*}
E & \{ \Rightarrow p \\
    & \quad q \Rightarrow r \\
D_{cl} & \{ \Rightarrow q \\
D_{def} & \{ p : q \Rightarrow q \\
    & \quad r : s \Rightarrow s \\
\end{align*}
\]

We have here that \( J(E) = \{ p \} \), \( J(E \cup D_{cl}) = \{ p, q, r \} \). The only extension is \( Ext = \{ p, q, r, s \} \).

What is the rôle of the default \( \neg q \Rightarrow \neg q \)? It seems that the clause in \( D_{cl} \) has made it redundant and that it could be eliminate. This would be true in a static environment, in which the knowledge and beliefs of the agent are permanent. But in a dynamic environment, where new information may become available and the beliefs must be eventually revised, this default may become relevant if new information gives preference to \( \neg q \) over \( q \). Recall that belief, in contrast to knowledge, is not necessarily true.

## 5 Discussion and Future Work

We have presented a representation of the knowledge and beliefs of an agent by a hybrid representation that includes a monotonic part (a set of clauses) and a non-monotonic part (a set of defaults.) The corresponding knowledge- and belief-sets may be iteratively computed. If we represent beliefs purely by defaults, we have no way of selecting among the different extensions, unless we include explicit priorities. The division of the doxastic part into a monotonic and a non-monotonic one provides a natural way to introduce preferences among the different extensions.

We are currently working on several directions. A natural extension of this representation in the case of multiple agents is the introduction of introspection,
that is, agents have knowledge about other agents' knowledge. This poses several interesting problems, such as common knowledge [4], [2]. Besides, private communication may entail inconsistency, since knowledge of an agent about the knowledge of agent $b$ may turn false as a result of a private communication from a third party. Thus, the axiom that knowledge is always true should be adapted.

Besides, we are interested in the effect of updating the databases. Since beliefs may be wrong, it may be necessary to change the doxastic part. In the case of a pure non-monotonic representation, this poses no problems, since this update amounts to the creation of a new extension, which is incompatible with the other existing ones. In the case of a hybrid representation, it may be possible that some older beliefs must be given up to preserve consistency. The idea is then to decide whether to accept the new belief and incorporate it with minimal changes to the belief database [1] or to reject it.

Finally, we want to explore the situation in which the state of the world changes. Then, unless agents are instantly informed of any change (not always a realistic assumption), their knowledge database may become inconsistent. Once more, the axiom that knowledge is always true may be too strong for practical purposes.

References