LCMAS 2004

second international workshop on

Logic and Communication in Multi-Agent Systems

Nancy, France 16-20 August, 2004

W. van der Hoek, A. Lomuscio, E. de Vink and M. Wooldridge
(eds.)

a workshop of

ESSLLI 2004
The 16th European Summer School in
Logic, Language and Information

Table of Contents

Epistemic Logics of Imperfect Information for Multi-agent Communication

Ahti-Veikko Pietarinen

Department of Theoretical Philosophy
Siltavuorenpenger 20 A, FIN-00014 University of Helsinki

Abstract. To express knowledge and communication in multi-agents systems, a hybrid of possible-worlds and game-theoretic semantics is defined for the language of independence-friendly (IF) first-order epistemic logic. The language distinguishes between specific and non-specific knowledge by hiding worlds. Identification on stratified domains is via imperfect information that agrees with cross-world references by coordinating information sets with world-bound aspects of individuals. Applications range from intentional identities (cross-modal anaphora) to reasoning and communication about focussed knowledge.

Modelling and model checking web services

Holger Schlingloff[1,2]

[1] Institut für Informatik, Humboldt-Universität zu Berlin
Rudower Chaussee 25, 12489 Berlin
[2] Fraunhofer Institut FIRST, Kekuléstraße 7, 12489 Berlin

Abstract. Web services are a new paradigm for business agents communicating via standardized protocols over the web, mutually offering and accessing services. Typical languages used in this context include BPEL4WS, WSDL, SOAP, and others. Whereas many current industrial activities on web services are aimed at topics like standardization of formats and interoperability between sites, there are also specific semantical problems which should be dealt with: Usually, web services can be combined to complex business process architectures, they offer the possibility of retraction, and allow approximative and time-dependent results. Moreover, in the design of a web service security aspects such as authentication, authorization and non-repudiation must be taken into account. We outline some of the approaches which have been proposed for the formalization of the above problems, and report on methods and tools for the specification, model checking and automated testing of web services.

Liar Detection within Agent Communications[*]

Guillaume MULLER and Laurent VERCOUTER
SMA/G2I/ÉNS des Mines de Saint-Étienne
158 cours Fauriel, 42023 Saint-Étienne CÉDEX 2
{muller, vercouter}@emse.fr

May 12, 2004

Abstract

Recent works suggest to model the reputation of the agents in order to evaluate the sincerity of an agent. The very first step of this mechanism is the detection of fraud and lie. However, most of the works concerning reputation consider this detection as implicit either because it is obvious in their specific application or because they assume that this detection is already made. In the present paper, we propose a definition of the concepts of lie, deception and fraud. From these definitions, we also propose a decentralized detection of lie and deception. Finally, this detection mechanism is used to maintain a representation of the agents' reputation.

1 Introduction

Agent communications is a very important issue in multi-agent systems. Agents have to interact in order to solve a problem, exchange informations, etc. The collective activities within the system strongly depend on the good functioning of communications and can fail if some communications are, voluntarily or not, wrong. Some guarantees such as authentication, integrity, confidentiality, etc. can be obtained by the use of security techniques. However, there are also some threats upon the *veracity of the content* of the messages. If the system is open, malicious agents may be introduced in order to *lie* to other agents and disturb the good functioning of the system.

Recent works [21, 5, 18, 3, 17] suggest some solutions to this problem by the calculation and the use of other agents' *reputation*. The reputation of an agent is usually evaluated based on its previous behavior. The more an agent had bad behaviors, the lower its reputation is. Therefore the very first step of this evaluation relies on the detection of a fraudulent behavior. There are few works [12, 1, 23, 16] about formalization or detection of fraud and lie. Most of the works about reputation consider this detection as implicit either because it is obvious in their specific application or because they assume that this detection is already made (e.g., in E-commerce, the user judges the outcome of a transaction and gives the feedback).

[*] PhD Student Paper.

The goal of the work described in this paper is to detect some agents that lie. More specifically, we focus on the detection of agents that send messages which are inconsistent when considered together. Therefore we first need to define precisely lies, deceptions and violations in this context. Then, a framework in which agents can detect, in a decentralized way, when lies occur is built and agent's reputations updated. In section 2, we define the concepts of lie, deception and violation. These definitions are used in section 3 to show how an agent can locally reason about lie and deception. We distinguish in particular different situations where an agent simply observes some communications, detects that a lie occurred or identifies a given lie or liar. In section 4, we describe a system that can detect lies by the way of a collaborative activity of agents, where each agent locally updates its own reputation model about other agents, and then uses reputation in its behavior when interacting with other agents.

2 Definitions of lie, deception and violation

It is generally admitted [23] that **fraud** is composed of two parts: a **violation** and a **deception**. A violation appears when an agent acts contrary to a given obligation. A deception occurs each time an agent i manages to make another agent j believe something i either does not believe or that is contrary to one of i's own beliefs. FIROZABADI *et al.* [12] followed this decomposition in order to formalize the different concepts involved in the definition of fraud: obligations and violations using standard deontic logic (SDL) [27, 8], actions performed by agents using a logic of action [14, 9, 19] and local beliefs of agents and deceptions using doxatic logic [10].

Based on their work, we define the concepts of **lie** and **deception** to address the specific case of communication. In this section, we briefly present the modal operators that are used and a speech act representation to be able to define the concepts of lie and deception that are described at last.

2.1 Background

The formalization of violations and deceptions requires the use of four modal operators taken from the modal logics quoted above. In the remaining of the paper we use the axiomatizations given in [12].

Violation The O operator, defined in Standard Deontic Logic [27, 8], denotes an obligation such that $O\phi$ means "it ought to be the case that ϕ". A violation can then be written as:

$$O\phi \wedge \neg\phi \tag{1}$$

The same operator is also used to denote the violation of a prohibition:

$$O\neg\phi \wedge \phi \tag{2}$$

Action A logic of action [14, 9, 19] provides two modal operators to represent that an agent performed an action:

- $E_i\phi$ means "agent i brings it about that ϕ";
- $H_i\phi$ means "agent i tries to bring it about that ϕ" but does not give any information on the success of the action.

These two operators are logically related by the following formula:

$$E_i\phi \rightarrow H_i\phi \tag{3}$$

Deception The doxatic logic provides the B operator to represent an agent's local belief. $B_i\phi$ means that agent i believes ϕ. FIROZABADI *et al.* [12] combine the B operator with the logic of action to represent deceptions (agent i deceives the agent j) by the two following formulae:

$$\neg B_i\varphi \wedge E_iB_j\varphi \tag{4}$$

$$B_i\neg\varphi \wedge E_iB_j\varphi \tag{5}$$

It is often interesting to detect the intention of an agent to deceive and not only its success. The two following formulae denote an attempt to deceive:

$$\neg B_i\varphi \wedge H_iB_j\varphi \tag{6}$$

$$B_i\neg\varphi \wedge H_iB_j\varphi \tag{7}$$

2.2 Representation of agent communications

In order to formally define what are lies, deceptions and frauds, we need to represent communications that are specific actions performed by agents. The speech act theory [22] defines each speech act by a performative and its associated preconditions and postconditions. Preconditions should be verified by the sender in order to be able to send the message, while postconditions represent the state of the world the sender wishes to reach. Various languages, including these preconditions and postconditions, exist (e.g., [11, 15]). We choose to use FIPA's as none of the formalisms is best suited for our purpose and because complete specifications are easy to find. In the FIPA specifications, the *feasibility preconditions* (noted FP) correspond to the preconditions and the *rational effects* (noted RE) to the postconditions.

We use these notations to represent that an agent has emitted a speech act by the fact that *it tries to reach the rational effect of the speech act*. If $sa_k^{i,j}$ is the kth speech act sent by agent i to agent j, the action of successfully sending this speech act is written by:

$$E_iRE(sa_k^{i,j}) \tag{8}$$

Since the receiver is autonomous, it may not take the message into account, and the sender cannot be certain that the rational effects are applied. The notation below denotes only that the speech act has been sent without information about its success:

$$H_iRE(sa_k^{i,j}) \tag{9}$$

2.3 Lies, deceptions and violations within communications

The sender of a speech act is also an autonomous agent. It is therefore possible that an agent sends a speech act without believing that its feasibility preconditions are true. We define such a case as a **lie**.

Definition of a lie

$$\text{Lie}(sa_k^{i,j}) \equiv H_i RE(sa_k^{i,j}) \wedge \neg FP(sa_k^{i,j}) \tag{10}$$

According to this definition, a lie from agent i to agent j is an attempt to reach the rational effects of a speech act whereas its feasibility conditions are not satisfied. If the lie is successful, a **deception** occurs as the rational effects are reached (we may aslo omit the B_i operator in this definition).

Definition of a deception

$$\text{Deception}(sa_k^{i,j}) \equiv E_i RE(sa_k^{i,j}) \wedge \neg FP(sa_k^{i,j}) \tag{11}$$

These formalizations can now be used to forbid lies in the agent communications. In the FIPA specifications, the *sincerity condition* assumes that an agent does not lie. This assumption is not realistic in open systems as it is impossible to have any guarantee on the internal implementation of the agents that enter in the system. The sincerity condition is made explicit by the following obligation:

Sincerity condition

$$O(\neg H_i RE(sa_k^{i,j}) \vee FP(sa_k^{i,j})) \equiv O(\neg \text{Lie}(sa_k^{i,j})) \tag{12}$$

This writing of the sincerity condition has two usages: (i) it can be used by agents to reason about this obligation and eventually to decide to violate it; (ii) it can be used by other agents to detect violations of this obligation and update their representations of the liars.

Violation of the sincerity condition

$$O(\neg \text{Lie}(sa_k^{i,j})) \wedge \text{Lie}(sa_k^{i,j}) \tag{13}$$

We use the formulae defined in this subsection in order to detect liars. Since a deception is a successful lie (formula 3 links formulae 10 and 11), deceivers may also be detected. Moreover, a fraud can be detected if a deception and a violation occur. Next sections focus on the detection processes.

3 Detection of Lies

In decentralized multi-agent systems agents have different beliefs. Some of them may detect lies whereas some others may not. In this section, we first distinguish different roles that an agent can play in a collaborative detection of lies. This detection process is then described.

3.1 Agents that Observe and Evaluate a Target

An agent may be unable to detect alone that another agent lied. According to its own local beliefs, it can, however, participate to a collective activity that aims at detecting liars. We consider three possible ways for agents to participate: an agent can either be the **Target** of the detection process or can be used as an **Observer** or an **Evaluator**.

An agent t, on which the fraud detection process focuses, is labelled as a **Target**. A set of messages sent by the **Target** is considered. We note SA^t any set of messages sent by agent t:

$$\mathbf{Target}(t, SA^t)$$

An **Observer** is an agent o that has observed some of the messages sent by a target t (where $SA'^t \subset SA^t$):

$$\mathbf{Observer}(o, SA'^t) \\ \equiv \\ \forall s \in SA'^t, B_o(H_t RE(s)).$$

An **Evaluator** is an agent that confronts observations about a set of messages from a given target t : $SA''^t \subset SA^t$. It is then able to detect if at least one lie exists or if the target has been sincere with respect to this set of messages.

$$\mathbf{Evaluator}(e, SA''^t) \\ \equiv \\ B_e(\ \exists s \in SA''^t, \mathrm{Lie}(s)) \vee B_e(\ \forall s \in SA''^t, \neg\mathrm{Lie}(s))$$

3.2 The Detection Processes

There are two kinds of processes for lies detection. The detection can occur after the observation of contradictory messages or during the evaluation of the sincerity of a specific target.

3.2.1 Observation-driven detection.

This detection can be performed by observers. An agent o is an observer of a target t. Agent o believes that a set of messages SA^t has been sent and that the sincerity condition holds. For each message $s \in SA^t$:

$$H_t RE(s) \wedge O(\neg\mathrm{Lie}(s))$$

If agent o assumes that the sender did not violate the sincerity condition we obtain:

$$H_t RE(s) \wedge O(\neg\mathrm{Lie}(s)) \wedge \neg(O(\neg\mathrm{Lie}(s)) \wedge \mathrm{Lie}(s))$$

This formula can be expanded into:

$$(\ H_t RE(s) \wedge \mathbf{O}(\neg\mathrm{Lie}(\mathbf{s})) \wedge \neg\mathbf{O}(\neg\mathrm{Lie}(\mathbf{s}))\ )\vee \tag{16a}$$

$$(\ \mathbf{H_t RE}(\mathbf{s}) \wedge O(\neg\mathrm{Lie}(s)) \wedge \neg\mathbf{H_t RE}(\mathbf{s})\ )\vee \tag{16b}$$

$$(\ H_t RE(s) \wedge O(\neg\mathrm{Lie}(s)) \wedge FP(s)\ ) \tag{16c}$$

The last part of this formula (16c) is, the only one determining for the truth value of the whole formula. Then, agent o assumes that $FP(s)$ holds.

If the assumption that $FP(s)$ holds leads to an inconsistency, an incorrect belief exists either in this assumption or in the previous beliefs of agent o. If this inconsistency is only caused by some assumptions of sincerity about agent t, then there exists at least one lie within the set of messages sent by t. The following formula expresses this situation:

$$\exists s_1, \ldots, s_n \in SA^t, (\bigwedge_{1 \leq i \leq n} FP(s_i)) \rightarrow \perp$$

Consequently, agent o both plays the roles of observer and evaluator. It plays the role of evaluator since it believes that there is a lie in SA^t. It is important to note that the process allows agent o to detect a lie occurred within the set of messages sent by agent t, therefore t is a liar; however, it is impossible at this stage and it may remain definitively impossible to detect which messages $s \in SA^t$ are lies.

3.2.2 Target-driven detection.

In this case we consider another agent, the **Beneficiary**. A beneficiary (noted b) is an agent that requires to know if a target t has lied in the past. The following process is used to gather some information about agent t:

1. Agent b asks some agents to become evaluators of the target t;

2. Each evaluator e looks for agents that have observed some messages sent by t and asks them for their observations;

3. An evaluator e confronts the observations that it received and may detect a lie if there is an inconsistency (this process is described in the previous section);

4. At last, each evaluator sends its own result to the beneficiary which merges them.

The choice of the target may be free or may be influenced by the occurrence of a conflict. In section 3.2.1 we define observation-driven detection as the detection of an inconsistency in the feasibility preconditions of some messages sent by the *same* agent t. However, if the messages involved in the conflict come from *different* agents, it is more difficult to know which one has lied. Moreover, the inconsistency may arise due to the application of the rational effects of some messages. In this case, all the agents that sent one of these messages are considered as targets for target-driven detections.

4 Reasoning about Lies

Each time a lie is detected, the beneficiary of this detection should use this information to update its representation of the target. The information is usually merged in an evaluation of the sincerity of the target: its reputation. Figure 1 shows how an agent links lies detection with reputation.

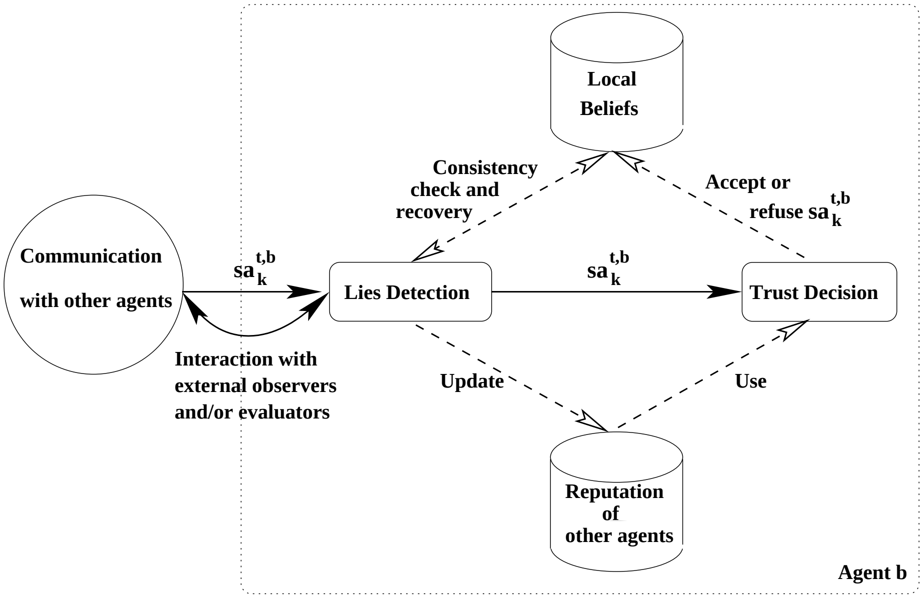


Figure 1: Using Lies Detection and Reputation in Agent Communications.

The *lies detection* module implements the processes described in section 3.2. During these processes the beneficiary may have to communicate with other agents (observers and evaluators) and uses its *local beliefs* to check if an inconsistency occurs. This process can result in an update of the reputation attached to some agents. Then, if the message $sa_k^{t,b}$ has not been detected as a lie, it is transmitted to the *trust decision* module that decides to accept the message (trust the sender) or refuse it (distrust the sender).

In this section we focus on the *reputation of other agents* and on the *trust decision* module. First, we describe different ways to use the detection of a lie in order to update reputations. We then show how an agent can use the reputation attached to other agents to avoid being deceived in the future.

4.1 Using different kinds of reputation

Even if a target has lied to another agent, it is not always a liar. In the same way, an agent that has not yet lied may not remain sincere in all its future communications. Then, it may be useful to estimate the sincerity of the target by a degree rather by a boolean value.

We represent reputation as a real number in the interval $[-1, +1]$. An agent which reputation is -1 is considered as a systematic liar whereas an agent with a reputation of $+1$ would be always sincere. In addition to this interval, an agent's reputation can take the value "unknown" if there is too few informations about it.

Our aim is not to propose a specific function to compute a reputation from several observations and evaluations. This function depends on the application and several different propositions exist [18, 20, 26, 28, 6, 13]. On the other hand, we are particularly interested by the source of observations and evaluations. Actually, informations communicated by other agents in order to estimate a reputation may not be as reliable as those that are observed or evaluated by the agent itself.

Therefore, there are several different reputation values attached to a same target. First, it depends on the agent that encapsulate the value. In a decentralized system, an agent computes the reputation attached to a target from its local beliefs and two distinct agents can obtain different results. Moreover, section 3.1 shows that the local beliefs of an agent allows it to play the role of observer and/or evaluator. It also means that an agent sometimes needs to rely on other agents to obtain some observations or evaluations. If an agent considers the information communicated by other agents is less reliable than the one observed or computed by itself, it can maintain different reputations for the same target that differ from one another by value and reliability.

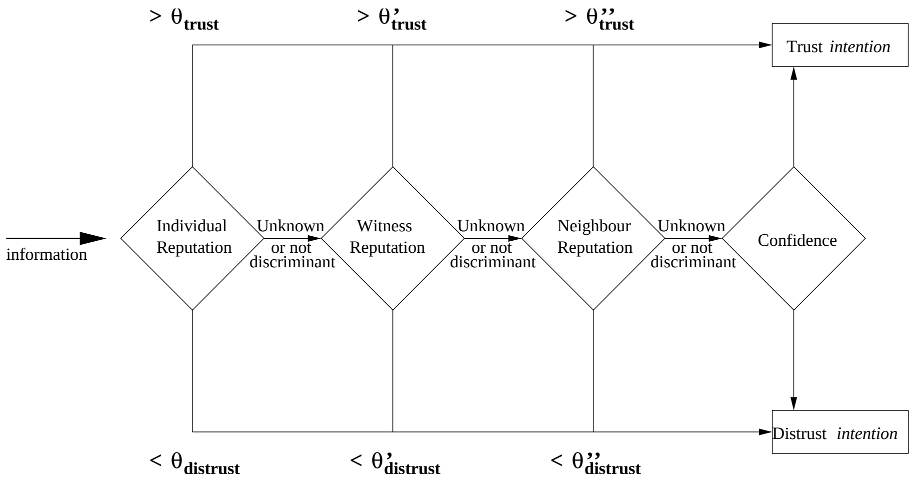


Figure 2: Trust Decision Process.

According to the source (internal or external) of the observations and evaluations, three kinds of reputation are distinguished:

- *Individual Reputation* is based on an agent's own observations and evaluations. In particular, an agent that has detected a lie by the way of an observation-driven detection (section 3.2.1) updates its individual reputation for the target;
- *Witness Reputation* is based on observations communicated by other agents (the witnesses). Thus, an agent confronts these external observations in order to detect if there is a lie and, consequently, updates its witness reputation;
- *Neighbor Reputation* relies on an external detection of lies. It is the integration of different reputations values communicated by other evaluators. It is usually updated by target-driven detection of lie (section 3.2.2).

An agent can use one of these reputations in order to decide if it can trust another agent. We need to consider an additional concept, the *Confidence*, to handle the specific case where an agent does not have relevant estimations about another agent in any kind of reputation. Confidence is a general disposition to trust. It is used by default to lead an agent to trust or distrust another agent which reputations are unknown. Confidence is not attached to a specific target.

Figure 2 shows an ordering of these reputation concepts that we think is common sense and that may be used in general case: Individual Reputation is considered the most reliable type of reputation by its owner since it is based on its own observations. Witness Reputation is less reliable since observation comes from other agents and may, therefore, be false. Neighbor Reputation is the less reliable of the three values since most of the information is external. Confidence not comparable to the other types since it does not target any agent; it is a general disposition to trust. Therefore Confidence comes last. However, it is clear that this ordering can vary in specific situations.

4.2 Preventing Future Deceptions

The aim of the *trust Decision* module is to decide whether the agent should trust or not a given target regarding a particular information. We consider here the specific case of communications where this information is a message sent by the target and where the decision process leads the agent to accept the message or refuse it. However, we think that this decision mechanism is general and can handle other situations where an agent should decide whether it trusts or not a target (e.g., anticipating if the target will fulfill or not its commitments, whether it will obey or not a norm…). Figure 2 shows how the various kinds of reputation presented in the previous section are used to implement the trust decision process.

Logically the agent considers in priority its most reliable belief: its *Individual Reputation* about the target. The *Individual Reputation* can be sufficient to decide to trust (respectively distrust) a target if it has an high (respectively low) value. This is represented in figure 2 by two thresholds θ_{trust} and $\theta_{distrust}$. If the *Individual Reputation* is greater than θ_{trust}, the agent trusts the target and accept the message it received from it. At the opposite, if the *Individual Reputation* is less than $\theta_{distrust}$, the agent distrusts the target and refuses its message. Otherwise, the *Individual Reputation* does not permit the agent to decide whether the target should be trusted or not. These other cases consist in specific values of the *Individual Reputation*: either the "unknown" value, or a moderate value (between θ_{trust} and $\theta_{distrust}$).

A similar decision function is then applied to the second most reliable reputation: the *Witness Reputation*. This value is also compared to two thresholds (θ'_{trust} and $\theta'_{distrust}$ that can be different from the thresholds used for *Individual Reputation*) in order to decide whether to trust or not the target. If this value is still not discriminant, *Neighbor Reputation* is considered for decision. As a last resort, the agent's *Confidence* makes the decision.

At the end of this decision mechanism, the agent has decided whether to trust or distrust the target. Then there are two ways to deal with the message received:

Trust decision The message $sa_k^{t,b}$ is accepted. The rational effects are applied by the receiver: $RE(sa_k^{t,b})$ and $E_t RE(sa_k^{t,b})$ are added to its *local beliefs*.

Distrust decision The message $sa_k^{t,b}$ is refused. The receiver does not apply the rational effects, but notes that the message has been sent: $H_t RE(sa_k^{t,b}) \wedge \neg E_t RE(sa_k^{t,b})$ are added to its *local beliefs*.

The main interest of the *trust decision* module is to preserve the agent from being deceived by some undetected lie. There are lies that are not detected by the *lies detection* module. Reputation can then be used not to believe messages sent by agents that have often lied in the past.

At the end of this decision process two undesirable cases may happen: (i) messages that are not lies may be rejected; (ii) undetected lies coming from agents with a high reputation may be accepted. If the former case should be avoided, the receiver does not have to definitely reject the message. It may rather asks some justifications to the target or to other agents. In the latter case, a deception occurs. The lie may be detected *a posteriori* if another message received later leads to an inconsistency with the undetected lie. However, few lies may remain undetected forever.

5 Conclusion

The work presented in this paper is related to two categories of existing works. The first category is the formalization of fraud with regards to the actions performed by agents. LOMUSCIO *et al.* [16] formalize violation in the bit transmission problem with deontic logic. However, their axiomatization is based on a representation of the world as a global state, which is composed of the state of each agent of the system plus the state of the environment. It is, therefore, necessary to enumerate each state of every agent and each state the environment may be in. In an open system, it is impossible to do this, since agents may come from different designers and their internal implementation is not available. FIROZABADI *et al.* [12] suggest a formalization of fraud in two parts: a formalization of violation and of deception. They propose a logical formalism to represent how an agent should behave (obligations) and what an agent has done (actions) in order to define a situation of violation and deception is represented by considering the agents' local beliefs. We propose in this paper to extend their work in order to formalize and detect lies within agent communications.

This extension allows us to show how some agents can detect that an agent has lied, and how they use this information to update a reputation model of the other agents. Reputation modeling is the second category of related works. Several different proposals [18, 20, 26, 28, 6, 13] exist to represent and to evaluate the reputation of some agents. However, most of these works focus on the evaluation of reputation but does not consider the very first step of this evaluation: the detection of a fraud. We use a reputation model issued from some previous work [17] to link the detection of lie to the updating of agents' reputation.

All the related works quoted above focus on one specific category without considering the others. One of the main interest of the work presented here is that it links all these categories to propose a mechanism that integrates the detection of lies, the representation of reputation and its use to prevent future deceptions. Moreover, this mechanism is decentralized and does not require the existence of a central entity. We are currently working on an implementation of this mechanism.

This work can still be extended in a few directions. A first extension consists in the formalism that we have chosen to represent violations and lies. A violation has been represented as an action that contradicts an obligation expressed with the help of standard deontic logic. It would be interesting to introduce an explicit representation of norms instead of the plain obligations. Thus, a violation would refer to a norm and we could benefit from some existing works [7, 4, 25] to define specific behaviors in case of a violation. Moreover, the reputation attached to an agent could also be contextualized by the norm violated or respected. Since several works about norms use deontic logic for obligations, we believe that our work remains open to the future introduction of norms. Moreover, the notion of time as not yet been taken into account. Using a temporal logic together with an ACL such as TACL [2] will allow to distinguish agents that really lies from agents that may have changed their opinion about an aspect of the world.

The formalization of lies relies on the specifications of FIPA-ACL [11]. The justification of this choice is that the detection of a lie requires the representation of a sincerity condition. The prevention of future deceptions also requires the representation of the success of a speech act (the rational effects in the FIPA specification). However, our work is not bound to the use of FIPA-ACL and other languages may be used if they represent the required preconditions and postconditions. For example, it is possible to use KQML [15] by using its sender preconditions and receiver postconditions. We are also exploring the social semantics approach of ACLs. Indeed, it does not need, for an agent that emits a speech act, to make strong assumptions on the internal representation of the receiver [24].

Another extension of this work would be to use lie detection in the messages exchanged *during* lie detection. If an agent lies when it sends a message containing an observation or an evaluation of a target, it can lead some other agents to believe that a liar is sincere (or the opposite). This problem is the main reason why different levels of reliability are distinguished for reputation (section 4.1). However, the messages containing some observations or evaluations may be considered as classical messages and may also be detected as lies or be rejected if they are sent by an agent with a low reputation.

References

[1] B. Bhargava, Y. Zhong, and Y. Lu. Fraud formalization and detection. In *Proceedings of DaWak'03*, 2003.

[2] T. Carron, H. Proton, and O. Boissier. A temporal agent communication language for dynamic multi-agents systems. In F. Garijo and M. Boman, editors, *Multi-Agent System Engineering - Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'99)*, volume 1647 of LNAI, pages 115–127. Springer-Verlag: Heidelberg, Germany, 1999.

[3] C. Castelfranchi, R. Falcone, and G. Pezzulo. Belief sources for trust: some learning mechanisms. In *Proceedings of the Workshop on Deception, Fraud and Trust in Agent Societies, AAMAS03*, July 2003.

[4] R. Conte, C. Castelfranchi, and F. Dignum. Autonomous norm acceptance. In *Proceedings of ATAL 1998*, pages 99–112, 1998.

[5] R. Conte and M. Paolucci. *Reputation in Artificial Societies. Social Beliefs for Social Order*. Kluwer Academic Publishers, 2002.

[6] C. Dellarocas. The digitalization of word of mouth: Promise and challenges of online reputation mechanisms. Draft, 2002.

[7] F. Dignum. Autonomous agents with norms. In *Artificial Intelligence and Law*, volume 7, pages 69–79, 1999.

[8] F. Dignum, J.-J. C. Meyer, R. Wieringa, and R. Kuiper. *Deontic Logic, Agency and Normative Systems*, chapter A modal approach to intentions, commitments and obligations: Intention plus commitment yields obligation, pages 80–97. Springer Verlag, 1996.

[9] D. Elgesem. *Action Theory And Modal Logic*. PhD thesis, Dept. of Philosophy, University of Oslo, 1993.

[10] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[11] FIPA. Fipa communicative act library specification. Technical Report SC00037J, FIPA: Fundation For Intelligent Phisical Agents, December 2002. Standard Status.

[12] B. S. Firozabadi, Y.-H. Tan, and R. M. Lee. Formal definitions of fraud. In *Proceedings of the Fourth International Workshop on Deontic Logic, DEON'98*, 1998.

[13] R. Jurca and B. Faltings. An incentive compatible reputation mechanism. In *Proceedings of the Workshop on Deception, Fraud and Trust in Agent Societies, AAMAS'03*, 2003.

[14] S. Kanger. Law and logic. *Theoria*, 38:105–132, 1972.

[15] Y. Labrou and T. Finin. A semantics approach for kqml - a general purpose communication language for software agents. In *Third International Conference on Information and Knowledge Management*, 1994.

[16] A. Lomuscio and M. Sergot. A formalisation of violation, error recovery, and enforcement in the bit transmission problem. *Journal of Applied Logic (selected articles from DEON02 - London)*, 1, 2003. A previous version of this paper appeared in the proceedings of DEON02.

[17] G. Muller, L. Vercouter, and O. Boissier. Towards a general definition of trust and its application to openness in mas. In *Proceedings of the Workshop on Deception, Fraud and Trust in Agent Societies, AAMAS03*, July 2003.

[18] J. Sabater and C. Sierra. Social regret, a reputation model based on social relations. *SIGecom Exchanges. ACM*, 3.1:44–56, 2002. http://www.acm.org/sigecom/exchanges/volume_3_(02)/3.1-Sabater.pdf.

[19] F. Santos, J. Carmo, and A. Jones. Action concepts for describing organised interaction. In R. A. Sprague, editor, *Thirtieth Annual Hawai International Conference on System Sciences*, pages 373–382, 1997.

[20] M. Schillo and P. Funk. Learning form and about other agents in terms of social metaphors. In J. M. Vidal and S. Sen, editors, *Proceedings of the "Agents Learning About, From and With other Agents" Workshop of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

[21] M. Schillo and P. Funk. Who can you trust: Dealing with deception. In *In Proceedings of the workshop Deception, Fraud and Trust in Agent Societies at the Autonomous Agents Conference*, pages 95–106, 1999.

[22] J. R. Searle. *Speech Acts: an essay in the philosophy of language*. Cambridge University Press, 1969.

[23] M. R. Simmons. Recognizing the elements of fraud, 1995. http://users.aol.com/marksimms/mrweb/fraudwww.htm.

[24] M. P. Singh. Agent communication languages: Rethinking the principles. In M.-P. Huget, editor, *Communication in Multiagent Systems*, volume 2650 of *Lecture Notes in Computer Science*, pages 37–50. Springer, 2003.

[25] J. Vàzquez-Salceda and F. Dignum. Modelling electronic organizations. In *Proceeding of CEEMAS'03*, pages 584–593, 2003.

[26] Y. Wang and J. Vassileva. Bayesian network-based trust model in peer-to-peer networks. In *Proceedings of of the Workshop on Deception, Fraud and Trust in Agent Societies, AAMAS'03*, pages 57–68, 2003.

[27] G. Wright. *New studies in Deontic Logic : Norms, Actions, and the Foundations of Ethics*, chapter On the Logic of Norms and Actions, pages 3–35. D. Reidel, 1981.

[28] G. Zacharia, A. Moukas, and P. Maes. Collaborative reputation mechanisms in electronic market places. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.

Bounded Model Checking for Deontic Interpreted Systems *

Bożena Woźna[1] and Alessio Lomuscio[1] and Wojciech Penczek[2]

[1] Department of Computer Science
King's College London
Strand, London WC2R 2LS,UK
email: {bozena,alessio}@dcs.kcl.ac.uk
[2] Wojciech Penczek
Institute of Computer Science, PAS
email: penczek@ipipan.waw.pl

Abstract. We propose a framework for the verification of specification in multiagent systems by symbolic model checking. The language CTLKD (an extension of CTL) allows for the representation of the temporal evolution of epistemic states of the agents, as well as their correct and incorrect functioning behaviour. We ground our analysis on the semantics of deontic interpreted systems. The verification approach is based on an adaption of the technique of bounded model checking, a mainstream approach in verification of reactive systems. We test our results on a typical communication scenario: the bit transmission problem with faults.

1 Introduction

The task of software engineers is to design and deploy a computer system that meets a particular set of specifications. This is by no means a trivial task. Only in the past few months the problems with NASA's and ESA's Mars exploration missions have made news showing how many unforeseen glitches even a thoroughly-designed distributed system may encounter. In mission-critical software as well as in particularly sensitive applications such as Internet protocols, the worldwide electronic banking system, etc., software engineers are interested in analysing the properties of software, and in particular in checking whether particular conditions, from the basic ones of deadlock to more complex ones, hold for a particular system.

The area of multiagent systems is also interested in a similar set of problems. Multiagent systems [15] are distributed systems in which the individual processes, or *agents*, are autonomous entities that engage in social activities such as coordination, negotiation, cooperation, etc. Since multiagent systems are autonomous and social, their range of possible behaviours is even greater than the

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

one of traditional distributed systems. It follows from this that the issue of verification of the properties a system satisfies is just as important in multiagent systems.

Software validation, i.e., the process of certifying that a piece of software satisfies certain characteristics, is currently conducted by means of three main techniques: testing, theorem proving, and model checking. Testing involves searching the state-space of the possible inputs of a program looking for potentially problematic outputs. Theorem proving techniques are based on the representation of a program by means of a system of formal logic; in its simplest instance, checking whether a property is satisfied amounts to checking whether a formula is a theorem of the logic that represents the program. Model checking [4] in its mainstream approach involves representing all possible computational traces of the program by means of a temporal model (appropriately represented) and checking whether or not a temporal formula, representing the property to be verified, holds in this model.

Software validation has in other words an intrinsic "deontic connotation". It amounts to checking whether the system under consideration behaves as it is prescribed by its specification. But it should be noted that, in this approach, this is a property that is *external* from the logical system. One could say, the correct functioning behaviour of the system is a *metalogical* property of the logic system representing the program. Differently from what happens in deontic logic, deontic concepts here are not explicitly used in the logic to represent the system, but they are built into the procedures that operate on the logic used to check the system.

Although a range of systems have been verified by means of standard verification techniques, multiagent systems applications call for a refined approach. Consider for example a number of automatic agents bidding for goods on an electronic auction. There may be rules as to how these agents may conduct their bidding, but it is often unfeasible and/or counterproductive to have many of these rules hard-wired in the auction protocol itself. As designers of the auction we may consider it beneficial that agents do not bid several times a second on the same good not to have resulting denial-of-service problems at server-level, but it seems difficult to enforce this when the agents are programmed by a variety of software houses on which we have no control. Other examples from traditional federated databases to more recent fault-control modules in mission-critical software point to similar conclusions: it is important to reason about the properties that hold in a system when the programs are functioning following their specification but also (and occasionally even more importantly) even they do not. In other words not only we would like to check whether a system satisfies its specification, but also we would like to derive the consequences resulting from the system not behaving as intended. Different deontic logics have been used to bring to the logical object level the distinction between correct (or ideal, normative, etc.) and incorrect states. In this paper we would like to take these ideas one step further and provide a technique by means of which we can not not specify but also automatically verify properties expressing compliance of a multiagent system with respect to specifications.

To carry out this analysis we use the formal machinery of verification by model checking, and in particular the one of *Bounded Model Checking* via SAT transformation [3]. In verification by model checking one typically describes a system S by means of a program in a language such as SMV [12]. This description is then supplied to the model checker which produces the (appropriately encoded) temporal model M_S representing all the possible executions of system S. To check whether a property P is satisfied in S one checks whether the temporal model M_S satisfies a formula ϕ_P representing P, i.e., $M_S \models \phi_P$. The key problem in this approach is to manage the representation of the resulting model M_S. One of the techniques available to keep the approach feasible is bounded model checking. This technique focuses on the attempt to discover faults in the specification of the system. Rather than checking the whole state space for the verification of the property, in bounded model checking one checks whether the negation of the property is actually satisfied on a fraction of the model, thereby producing a counterexample. Furthermore, the actual check is translated into a standard propositional satisfiability problem by computing appropriate translation into propositional formulas representing both the truncated model and the formula to be checked. By means of this approach subtle bugs in protocols for reactive systems have been discovered. We refer to [1, 3, 5, 14] for more details.

While in reactive systems it is enough to model a system by means of a purely temporal language, multiagent systems are defined following what is often referred to as the “intentional stance” [6]. In other words it is useful to describe autonomous agents in terms of their knowledge, belief, intentions, social context, etc. This implies that the model checking problem for multiagent systems cannot simply be stated as one on temporal logic but that richer formalisms need being used. In past research we have provided a model checking algorithm for a branching time temporal-epistemic logic (CTLK) [13]. In this paper we extend this work by providing a bounded model checking algorithm for a logic that comprises knowledge and a deontic component representing correct functioning behaviour of the system.

The paper is organised as follows. In Section 2, we fix the notation on the semantics of deontic interpreted systems. In Section 3 we present the language of CTLKD, an extension of CTLK, representing correct/incorrect functioning behaviour of the agents. In Section 4 we present a bounded semantics definition for satisfaction that we use in Section 5 to define the algorithm of bounded model checking. In Section 6 we apply the formalism to an example close to the multiagent systems literature: the bit transmission problem with faults.

2 Deontic Interpreted Systems

In this section we introduce *deontic interpreted systems*. These were defined in [10] to represent and reason about correct functioning behaviour of multiagent systems. They provide a semantics based on the computation states of the agents, on which it is possible to interpret a deontic modality $\mathcal{O}_i\phi$, representing the fact "in all correct functioning executions of agent i, ϕ holds", as well as a traditional epistemic modality $K_i\phi$ representing knowledge of ϕ by agent i, and standard branching time temporal operators[1]. An axiomatisation of deontic interpreted systems has been provided for the non-temporal fragment of the language; we refer the interested reader to [11] for more details.

The following is reported to fix the notation only; more details can be found in [10, 13]. Let $\mathcal{PV}$ be a set of propositional variables and $\mathcal{A} = \{1, 2, \ldots, n\}$ be a set of agents. Consider n non-empty sets $L_1, \ldots, L_n$ of local states, one for each agent of the system, and a set L_e of local states for the environment. For each agent $i \in \mathcal{A}$, consider a set of possible action Act_i and a set of protocols $P_i : L_i \to 2^{Act_i}$ representing the functioning behaviour of every agent, and consider a function $P_e : L_e \to 2^{Act_e}$ for the environment.

Further assume that for every agent, its set of local states can be divided into allowed and disallowed states. We call these states *green* and *red* respectively. For n agents and $n+1$ mutually disjoint and non-empty sets $\mathcal{G}_1, \ldots, \mathcal{G}_n, \mathcal{G}_e$ we define the set of all *possible global states* (S) as the Cartesian Product $L_1 \times \ldots \times L_n \times L_e$, such that $L_1 \supseteq \mathcal{G}_1, \ldots, L_n \supseteq \mathcal{G}_n, L_e \supseteq \mathcal{G}_e$. $\mathcal{G}_e$ is called the set of green states for the environment, and for any agent i, $\mathcal{G}_i$ is called the set of green states for agent i. The complement of $\mathcal{G}_e$ with respect to L_e (denoted by $\mathcal{R}_e$) is called the set of red states for the environment, and respectively the complement $\mathcal{G}_i$ with respect to L_i (denoted by $\mathcal{R}_i$) is called the set of red states for the agent i.

We can model the computation taking place in the system by means of a transition function $t : S \times Act \to S$, where $Act \subseteq Act_1 \times \ldots \times Act_n \times Act_e$ is the set of joint actions. Intuitively, given an initial state $\iota \in S$, the sets of protocols, and the transition function, we can build a (possibly infinite) structure that represents all the possible computations of the system. Indeed, we will deal with the systems, in which the state space consists of reachable global states only. A state $s \in S$ is *reachable* if there is a sequence of states $(s_0, \ldots, s_n)$ such that $s_0, \ldots, s_n \in S$, $s_0 = \iota$, $s_n = s$, and for all $i \in \{0, \ldots, n-1\}$ there exists an action $act_i \in Act$ such that $t(s_i, act_i) = s_{i+1}$, i.e., s_{i+1} is the result of applying the transition function t to the global state s_i, and a joint action act_i. If each of the components of act_i is prescribed by the corresponding protocol P_j at $l_j(s_i)$, for $j \in \mathcal{A}$, then the resulting state will only contain green local states, otherwise it may contain some red local states. For further considerations on this see [10]. In the following we abstract from the transition function, the actions, and the protocols, and simply use the relation T, but it should be clear that this is uniquely determined by the interpreted system under consideration.

Let $l_i : S \to L_i$ be a function which returns the local state of agent i from a global state. A *deontic interperted system* is defined as follows:

Definition 1. *Given a set of agents $\mathcal{A} = \{1, \ldots, n\}$, the corresponding green and red states, protocols, and transition function, a* deontic interperted system *(or simply a model) is a tuple $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ where*

[1] Temporal operators were not actually used in [10] but this is a straightforward extension.

- *$\mathcal{DS} \subseteq S$ is a finite set of reachable global states for $\mathcal{A}$,*
- *ι is an initial state,*
- *$T \subseteq \mathcal{DS} \times \mathcal{DS}$ is a serial binary relation on $\mathcal{DS}$ (i.e., each state has at least one T-successor),*
- *$R_i^O \subseteq \mathcal{DS} \times \mathcal{DS}$ is a relation for each agent $i \in \mathcal{A}$ defined by: $sR_i^O s'$ iff $l_i(s') \in \mathcal{G}_i$[2],*
- *$R_i^K \subseteq \mathcal{DS} \times \mathcal{DS}$ is a relation for each agent $i \in \mathcal{A}$ defined by: $sR_i^K s'$ iff $l_i(s') = l_i(s)$,*
- *$\mathcal{V} : \mathcal{DS} \longrightarrow 2^{\mathcal{PV}}$ is a valuation function such that true $\in \mathcal{V}(s)$ for all $s \in \mathcal{DS}$. $\mathcal{V}$ assigns to each state a set of proposition variables that are assumed to be true at that state.*

By $|M|$ we denote the number of states of M, whereas $\mathbb{N} = \{0, 1, 2, \ldots\}$ indicates the set of natural numbers, and $\mathbb{N}_+ = \{1, 2, \ldots\}$ the set of positive natural numbers.

A *computation* in M is an infinite sequence $\pi = (s_0, s_1, \ldots)$ of states such that $(s_i, s_{i+1}) \in T$ for each $i \in \mathbb{N}$. For a computation $\pi = (s_0, s_1, \ldots)$, let $\pi(k) = s_k$, and $\pi_k = (s_0, \ldots, s_k)$, for each $k \in \mathbb{N}$. In the rest of the paper we shall call π_k a *k-computation.* Moreover, a $k-$computation π_k is a *(k,l)-loop* if $(\pi_k(k), \pi_k(l)) \in T$ for some $0 \leq l \leq k$. We call π_k simply a *loop* if there is an $l \in \mathbb{N}$ with $l \leq k$ for which π_k is a (k,l)-loop. By $\Pi(s)$ we denote the set of all the infinite computations starting at $s \in M$, whereas by $\Pi_k(s)$ the set of all the $k-$computations starting at s.

3 The logic CTLKD

Here we fix syntax and semantics for CTLKD, an extension of CTL [7], introduced by Emerson and Clarke, enriched with modal operators representing correct functioning behaviour, and standard epistemic operators [8]. The bounded model checking problem of the temporal epistemic fragment of the language was analysed in [13]

Definition 2 (Syntax of CTLKD**).** *Let $\mathcal{PV}$ be a set of propositional variables also containing the symbol true. The set of* CTLKD *formulas $\mathcal{FORM}$ is defined inductively as follows:*

- *every member p of $\mathcal{PV}$ is a formula,*
- *if α and β are formulas, then so are $\neg\alpha$, $\alpha \wedge \beta$ and $\alpha \vee \beta$,*
- *if α is formula, then so are $\mathrm{EX}\alpha$, $\mathrm{EG}\alpha$ and $\mathrm{E}\alpha\mathrm{U}\beta$,*
- *if α is formula, then so are $\mathcal{P}_i\alpha$, and $\overline{\mathrm{K}}_i\alpha$, for $i \in \mathcal{A}$.*

Intuitivly, E means there exists a computation, $\mathrm{X}\alpha$ is true in a computation if α is true at the second state of the computation, $\alpha\mathrm{U}\beta$ is true in a computation if β is true at some state on the computation and always earlier α holds, and

[2] Since each R_i^O only depends on the target state, for what pertains this component we could have equally defined a model by means of green local states for agent i.

Gα is true in a computation if α is true at all the states of the computation. We use the indexed modal operator $\mathcal{P}_i$ to represent the *correctly functioning circumstances of agent i*. The formula $\mathcal{P}_i\alpha$ stands for "there is a state where agent i is functioning correctly, and in which α holds". We refer to [10, 11] for a discussion of this notion. Moreover we use the indexed modality $\overline{\mathrm{K}}_i$ to represent the diamond of an epistemic operator for agent i [8]: $\overline{\mathrm{K}}_i\alpha$ stands for "agent i considers possible that α".

The derived basic modalities are defined as follows: $\mathrm{EF}\alpha \overset{def}{=} \mathrm{E}(true\mathrm{U}\alpha)$, $\mathrm{AX}\alpha \overset{def}{=} \neg\mathrm{EX}\neg\alpha$, $\mathrm{AF}\alpha \overset{def}{=} \neg\mathrm{EG}\neg\alpha$, $\mathrm{A}(\alpha\mathrm{R}\beta) \overset{def}{=} \neg\mathrm{E}(\neg\alpha\mathrm{U}\neg\beta)$, $\mathrm{AG}\alpha \overset{def}{=} \neg\mathrm{EF}\neg\alpha$, $\mathcal{O}_i\alpha \overset{def}{=} \neg\mathcal{P}_i\neg\alpha$ for $i \in \mathcal{A}$, $\mathrm{K}_i\alpha \overset{def}{=} \neg\overline{\mathrm{K}}_i\neg\alpha$ for $i \in \mathcal{A}$. Moreover, $\alpha \to \beta \overset{def}{=} \neg\alpha \vee \beta$.

The logic ECTLKD is the restriction of CTLKD such that the negation can be applied only to elements of $\mathcal{PV}$, i.e., $\neg\alpha$ is replaced by $\neg p$ in the Definition 2.

The logic ACTLKD is also the restriction of CTLKD such that its language is defined as $\{\neg\varphi \mid \varphi \in \mathrm{ECTLKD}\}$. It is easy to see that ACTLKD consists of the temporal formulas of the form: $\mathrm{AX}\alpha$, $\mathrm{A}(\alpha\mathrm{R}\beta)$, $\mathrm{AF}\alpha$, $\mathrm{K}_i\alpha$ and $\mathcal{P}_i\alpha$.

Definition 3 (Semantics of CTLKD). *Let M be a model, s be a state, and α, β be formulas of* CTLKD. *$M, s \models \alpha$ denotes that α is true at the state s in the model M. M is omitted, if it is implicitly understood. The relation $\models$ is defined inductively as follows:*

$s \models p$ *iff* $p \in \mathcal{V}(s)$, $s \models \neg\alpha$ *iff* $s \not\models \alpha$,
$s \models \alpha \vee \beta$ *iff* $s \models \alpha$ *or* $s \models \beta$, $s \models \alpha \wedge \beta$ *iff* $s \models \alpha$ *and* $s \models \beta$,
$s \models \mathrm{EX}\alpha$ *iff* $(\exists\pi \in \Pi(s))\ \pi(1) \models \alpha$,
$s \models \mathrm{EG}\alpha$ *iff* $(\exists\pi \in \Pi(s))(\forall m \geq 0)\ \pi(m) \models \alpha$,
$s \models \mathrm{E}(\alpha\mathrm{U}\beta)$ *iff* $(\exists\pi \in \Pi(s))(\exists m \geq 0)\ [\pi(m) \models \beta$ *and* $(\forall j < m)\ \pi(j) \models \alpha]$,
$s \models \mathcal{P}_i\alpha$ *iff* $\exists s' \in \mathcal{DS}\ (sR_i^O s'$ *and* $s' \models \alpha)$,
$s \models \overline{\mathrm{K}}_i\alpha$ *iff* $\exists s' \in \mathcal{DS}\ (sR_i^K s'$ *and* $s' \models \alpha)$.

Definition 4 (Validity). *A* CTLKD *formula φ is valid in a model $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ (denoted $M \models \varphi$) iff $M, \iota \models \varphi$, i.e., φ is true at the initial state of the model M.*

4 Bounded Semantics for ECTLKD

In this section we give a *bounded semantics* for ECTLKD in order to define the *bounded model checking problem* for ECTLKD, and to translate it subsequently into a satisfiability problem. This formalism is an extension of the one presented in [14].

Definition 5 (k−model). *Let $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ be a model and $k \in \mathbb{N}_+$. A structure $M_k = (\mathcal{DS}, \iota, P_k, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ is a k−model for M, where P_k is the set of all the k-computations of M, i.e., $P_k = \bigcup_{s \in \mathcal{DS}} \Pi_k(s)$.*

Satisfaction for the temporal operators in the bounded case depends on whether or not the computation π defines a loop, i.e., whether $loop(\pi) \neq \emptyset$, where $loop$ is defined below.

Definition 6. *Let* $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ *be a model,* $M_k = (\mathcal{DS}, \iota, P_k, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ *be a* $k-$*model for* M*, and* $\pi \in P_k$*. The function* $loop : P_k \rightarrow 2^{\{0,\ldots,k\}}$ *is defined as:*

$$loop(\pi) \;=\; \{l \mid 0 \leq l \leq k \; and \; (\pi(k),\; \pi(l)) \in T\}$$

The main reason for reformulating the semantics of the modalities in the following definition in terms of elements of $k-$computations rather than elements of $\mathcal{DS}$ or Π is to restrict the semantics to a part of the model.

Note that the interpretation of the temporal modalities on bounded semantics is different from the one of Definition 3.

Definition 7 (Bounded semantics).
Let M_k *be a* $k-$*model and* α, β *be* ECTLKD *formulas.* $M_k, s \models \alpha$ *denotes that* α *is true at the state* s *of* M_k*.* M_k *is omitted if it is clear from the context. The relation* $\models$ *is defined inductively as follows:*
$s \models p$ *iff* $p \in \mathcal{V}(s)$*, for* $p \in \mathcal{PV}$*,*
$s \models \neg p$ *iff* $p \notin \mathcal{V}(s)$*, for* $p \in \mathcal{PV}$*,*
$s \models \alpha \vee \beta$ *iff* $s \models \alpha$ *or* $s \models \beta$*,*
$s \models \alpha \wedge \beta$ *iff* $s \models \alpha$ *and* $s \models \beta$*,*
$s \models \mathrm{EX}\alpha$ *iff* $(\exists \pi \in P_k(s))\; \pi(1) \models \alpha$*,*
$s \models \mathrm{EG}\alpha$ *iff* $(\exists \pi \in P_k(s))(\forall 0 \leq j \leq k)(\pi(j) \models \alpha$ *and* $loop(\pi) \neq \emptyset)$*,*
$s \models \mathrm{E}(\alpha \mathrm{U} \beta)$ *iff* $(\exists \pi \in P_k(s))(\exists 0 \leq j \leq k)\big(\pi(j) \models \beta$ *and* $(\forall 0 \leq i < j)\pi(i) \models \alpha\big)$*,*
$s \models \mathcal{P}_i\alpha$ *iff* $(\exists \pi \in P_k(\iota))(\exists 0 \leq j \leq k)\big(\pi(j) \models \alpha$ *and* $sR_i^O\pi(j)\big)$*,*
$s \models \overline{\mathrm{K}}_i\alpha$ *iff* $(\exists \pi \in P_k(\iota))(\exists 0 \leq j \leq k)\big(\pi(j) \models \alpha$ *and* $sR_i^K\pi(j)\big)$*.*

The above extends to deontic modalities the bounded semantics of [13, 14]. As in [13] we note that the given Definition 1, the relations for the operator $\mathcal{P}_i$ used above are constructed on the basis of the *internal structure of the global states* of the system (i.e., they are defined on the basis of the local states of the agents), and not by means of an ad-hoc construction by the modeller. Note also that while the conditions for the temporal components require the states to be reachable from the state in consideration, this is not the case for operators $\mathcal{P}_i$ and $\overline{\mathrm{K}}_i$, where we consider whether or not there is *a* computation from the initial state that results in a state that is related for agent i from the global state under consideration. This guarantees reachability of such a state and corresponds to the usual interpretation of the modalities in the non-bounded model.

The theoretical results proved in [13] for CTLK can easily be extended for CTLKD.

Definition 8 (Validity for Bounded Semantics). *An* ECTLKD *formula* φ *is valid on a k-model* M_k *(denoted* $M \models_k \varphi$*) iff* $M_k, \iota \models \varphi$*.*

Next, we describe how the model checking problem ($M \models \varphi$) can be reduced to the bounded model checking problem ($M \models_k \varphi$).

Lemma 1. *Let M be a model, s be a state of M, and φ be an* ECTLKD *formula. Then, the following two conditions hold:*

a) $M_k, s \models \varphi$ implies $M_l, s \models \varphi$, for $l \geq k$,
b) $M_k, s \models \varphi$ implies $M, s \models \varphi$.

Proof. Straightforward by induction on the length of φ.

Lemma 2. *Let M be a model, φ be an* ECTLKD *formula, s be a state of M, and $k = |M|$. If $M, s \models \varphi$, then $M_k, s \models \varphi$.*

Proof. By induction on the length of φ. The lemma follows directly for the propositional variables and their negations.

Next, assume that the hypothesis holds for all the proper sub-formulas of φ. If φ is equal to either $\alpha \wedge \beta$ or $\alpha \vee \beta$, then it is easy to check that the lemma holds. Consider φ to be of the following forms:

- $\varphi = \mathrm{EX}\alpha \mid \mathrm{EG}\alpha \mid \mathrm{E}(\alpha \mathrm{U} \beta)$. By induction hypothesis — see [14] page 139.
- $\varphi = \mathcal{P}_i\alpha$. By definition, there is a state s' in M such that $l_i(s') \in \mathcal{G}_i$ and $M, s' \models \alpha$. By the inductive assumption, we have that $M_k, s' \models \alpha$. Since s' is reachable, it is reachable from ι in at most k steps as $k = |M|$. Thus, there is a $k-$computation $\pi \in P_k$ such that $\pi(0) = \iota$ and $\pi(i) = s'$ for some $i \leq k$. So, we have $M_k, s \models \mathcal{P}_i\alpha$.
- $\varphi = \overline{\mathrm{K}}_i\alpha$. By definition, there is a state s' in M such that $l_i(s) = l_i(s')$ and $M, s' \models \alpha$. By the inductive assumption, we have that $M_k, s' \models \alpha$. Since s' is reachable, it is reachable from ι in at most k steps as $k = |M|$. Thus, there is a $k-$computation $\pi \in P_k$ such that $\pi(0) = \iota$ and $\pi(i) = s'$ for some $i \leq k$. So, we have $M_k, s \models \overline{\mathrm{K}}_i\alpha$.

In this setting we can prove that in some circumstances satisfiability in the $|M|$-bounded semantics is equivalent to the unbounded one.

Theorem 1. *Let $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ be a model, φ be an* ECTLKD *formula and $k = |M|$. Then, $M \models \varphi$ iff $M \models_k \varphi$.*

Proof. Straightforward from Lemma 1 and Lemma 2 above.

Given that we reasoned on a bounded model of size $|M|$ there is nothing surprising about the results above. The rationale behind the method is that for particular examples checking satisfiability of a formula can be done on a small fragment of the model.

5 The BMC algorithm for ECTLKD

In this section we present a Bounded Model Checking (BMC) method for ECTLKD. This is an extension of the method appearing in [13, 14]. This construction first appeared in [14], and was then extended in [13] for the CTLK case.

Definition 9. *Let $M_k = (\mathcal{DS}, \iota, P_k, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ be a k-model of M. We say that a structure $M_k' = (\mathcal{DS}', \iota, {P_k}', R_1'^O, \ldots, R_n'^O, R_1'^K, \ldots, R_n'^K, \mathcal{V}')$ is a* submodel *of M_k if ${P_k}' \subseteq P_k$, $States({P_k}') \subseteq \mathcal{DS}' \subseteq \mathcal{DS}$, $R_i'^O = R_i^O \cap (\mathcal{DS}' \times \mathcal{DS}')$, for $i \in \mathcal{A}$, $R_i'^K = R_i^K \cap (\mathcal{DS}' \times \mathcal{DS}')$, for $i \in \mathcal{A}$, and $\mathcal{V}' = \mathcal{V}|_{\mathcal{DS}'}$, where $States({P_k}')$ defines the set of states reached in all computations in ${P_k}'$, and $\mathcal{V}|_{\mathcal{DS}'}$ denotes the restriction of the interpretation function $\mathcal{V}$ to $\mathcal{DS}'$, a subset of $\mathcal{DS}$ (upon which $\mathcal{V}$ is defined).*

For technical reasons we allow for having states in $\mathcal{DS}'$, which may not be reached in ${P_k}'$, but obviously all the states of $\mathcal{DS}'$ are reachable in M_k as $\mathcal{DS}' \subseteq \mathcal{DS}$. The bounded semantics of ECTLKD over submodels M_k' can still be defined as for M_k (see Def. 7). Our present aim is give a bound for the number of k-computations in M_k' such that the validity of φ in M_k is equivalent to the validity of φ in M_k'.

Definition 10. *Define a function $f_k : \mathcal{FORM} \rightarrow \mathbb{N}$ as follows:*

- $f_k(p) = f_k(\neg p) = 0$*, where* $p \in \mathcal{PV}$,
- $f_k(\alpha \vee \beta) = max\{f_k(\alpha), f_k(\beta)\}$,
- $f_k(\alpha \wedge \beta) = f_k(\alpha) + f_k(\beta)$,
- $f_k(\mathrm{EG}\alpha) = (k+1) \cdot f_k(\alpha) + 1$,
- $f_k(\mathrm{E}(\alpha \mathrm{U} \beta)) = k \cdot f_k(\alpha) + f_k(\beta) + 1$,
- $f_k(Y\alpha) = f_k(\alpha) + 1$*, for* $Y \in \{\mathrm{EX}, \overline{\mathrm{K}}_i, \mathcal{P}_i\}$.

The function f_k determines the number of k-computations of a submodel M_k' sufficient for checking an ECTLKD formula. Here we take this bound as given, but we provide a proof of the adequacy of this in the next section.

The main idea is that we can check φ over M_k by checking the satisfiability of a propositional formula $[M, \varphi]_k = [M^{\varphi,\iota}]_k \wedge [\varphi]_{M_k}$, where the first conjunct represents (part of) the model under consideration and the second a number of constraints that must be satisfied on M_k for φ to be satisfied. Once this translation is defined, checking satisfiability of an ECTLKD formula can be done by means of a SAT-checker. Although from a theoretical point of view the complexity of this operation is no easier, in practice the efficiency of modern SAT-checkers makes the process worthwhile in many instances. In this process, an important decision to take is the size k of the truncation. We do not discuss this issue in this paper, but we do point out the fact that there are heuristics that can be developed for particular classes of examples [2].

A trivial mechanism, for instance, would be to start with $k := 1$, test satisfiability for the translation, and increase k by one either until $[M^{\varphi,\iota}]_k \wedge [\varphi]_{M_k}$ becomes satisfiable or k reaches $|M|$.

Definition 11. *BMC algorithm for* ECTLKD*:*

1. *Let $\varphi := \neg\psi$ (where ψ is an* ACTLKD *formula).*
2. *Set $k := 1$.*
3. *Select the $k-$model M_k.*

4. *Select the submodels M'_k of M_k with $|P_k{}'| \leq f_k(\varphi)$.*
5. *Translate the transition relation of all the submodels M'_k of M_k into a propositional formula $[M^{\varphi,\iota}]_k$.*
6. *Translate φ over all M'_k into a propositional formula $[\varphi]_{M_k}$.*
7. *Check the satisfiability of $[M,\varphi]_k := [M^{\varphi,\iota}]_k \wedge [\varphi]_{M_k}$.*
8. *If $[M,\varphi]_k$ is satisfiable then return $M \models \varphi$ (i.e., $M \not\models \psi$) else set $k := k+1$.*
9. *If $k = |M|+1$, then return $M \not\models \varphi$ (i.e., $M \models \psi$) else go to 3.*

Now, we give details of this translation. We begin with the encoding of the transitions in the interpreted system under consideration. Recall that the set of reachable global states is $\mathcal{DS} \subseteq \prod_{i=1}^{n} L_i \times L_e$, where $L_i \supseteq \mathcal{G}_i$ for each agent $i \in \mathcal{A}$, and $L_e \supseteq \mathcal{G}_e$ for the environment. We assume that $L_i = \mathcal{G}_i \cup \mathcal{R}_i \subseteq \{0,1\}^{g_i} \times \{0,1\}^{r_i}$, where $g_i = \lceil log_2(|\mathcal{G}_i|)\rceil$, $r_i = \lceil log_2(|\mathcal{R}_i|)\rceil$, and $L_e = \mathcal{G}_e \cup \mathcal{R}_e \subseteq \{0,1\}^{g_e} \times \{0,1\}^{r_e}$, where $g_e = \lceil log_2(|\mathcal{G}_e|)\rceil$, $r_e = \lceil log_2(|\mathcal{R}_e|)\rceil$. Let $g_1 + r_1 \ldots + g_n + r_n + g_e + r_e = m$. Then, each global state $s = (l_1,\ldots,l_n,l_e) = (s[1],\ldots,s[m])$ can be represented by $w = (w[1],\ldots,w[m])$ (which we shall call a *global state variable*), where each $w[i]$ for $i = 1,\ldots,m$ is a propositional variable. Notice that we distinguish between global states being sequences of binary digits and their representations in terms of propositional variables $w[i]$. A finite sequence $(w_0,\ldots,w_k)$ of global state variables is called a *symbolic $k-$path*. In general we shall need to consider not just one but a number of symbolic $k-$paths. This number depends on the formula φ under investigation, and it is returned as the value $f_k(\varphi)$ of the function f_k. We refer to [14] for more details. To construct $[M,\varphi]_k$, we first define a propositional formula $[M^{\varphi,\iota}]_k$ that constrains the $f_k(\varphi)$ symbolic k-paths to be valid k-computations of M_k. For $j \in \{1,\ldots,f_k(\varphi)\}$, the j-th symbolic $k-$computation is denoted as $w_{0,j},\ldots,w_{k,j}$, where $w_{i,j}$ for $i \in \{0,\ldots,k\}$ are global state variables.

Let $\mathcal{SV}$ be a set of state variables, $\mathcal{SF}$ be a set of propositional formulas over $\mathcal{SV}$, and let $lit : \{0,1\} \times \mathcal{SV} \to \mathcal{SF}$ be a function defined as follows: $lit(0,p) = \neg p$ and $lit(1,p) = p$. Moreover, let $green_i : \mathcal{SV}^m \to \mathcal{SV}^{g_i}$, for $i = 1,\ldots,n,e$ be a function which returns the sequence of state variables encoding the green states of the i-th agent or environment, and let Idx_i and Idx_e be sets of the indexes of the bits of the local states of each agent i and environment in the global states. Furthermore, let w, v be global state variables. We define the following propositional formulas:

- $I_s(w) := \bigwedge_{i=1}^{m} lit(s[i], w[i])$.
  This formula encodes the state s of the model, i.e., $s[i] = 1$ is encoded by $w[i]$, and $s[i] = 0$ is encoded by $\neg w[i]$.
- $p(w)$ is a formula over $w[1],\ldots,w[m]$, which is true for a valuation $(s_1,\ldots,s_m) \in \{0,1\}^m$ of $(w[1],\ldots,w[m])$ iff $p \in \mathcal{V}((s_1,\ldots,s_m))$, where $p \in \mathcal{PV}$.
  This formula encodes a proposition $p \in \mathcal{PV}$.
- $H(w,v) := \bigwedge_{i=1}^{m} w[i] \Leftrightarrow v[i]$.
  This formula represents logical equivalence between global state encodings, representing the fact that they represent the same state.
- $HP_i(w) := \bigvee_{l \in \mathcal{G}_i} (\bigwedge_{j=1}^{g_i} lit(l[j], green_i(w)[j]))$.
  This formula encodes an accessibility of a global state in which agent i is running correctly.

- $HK_l(w,v) := \bigwedge_{i \in Idx_l} w[i] \Leftrightarrow v[i]$.
  This formula represents logical equivalence between l-local state encodings, representing the fact that they represent the same local state, i.e., the local state in the two states is the same.
- $TR(w,v)$ is a formula over the propositions $w[1], \ldots, w[m]$, $v[1], \ldots, v[m]$, which is true for a valuation $(s_1, \ldots, s_m)$ of $(w[1], \ldots, w[m])$ and a valuation $(s'_1, \ldots, s'_m)$ of $(v[1], \ldots, v[m])$ iff $((s_1, \ldots, s_m), (s'_1, \ldots, s'_m)) \in T$.
- $L_{k,j}(l) := TR(w_{k,j}, w_{l,j})$,
  This formula encodes a backward loop connecting the k-th state to the l-th state in the symbolic $k-$computation j, for $0 \leq l \leq k$.

The propositional formula $[M^{\varphi,\iota}]_k$, representing the transitions in the k-model, is given by the following definition.

Definition 12 (Unfolding of Transition Relation).
Let $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ be a model, $k \in \mathbb{N}_+$ be a bound, and φ be an ECTLKD *formula. The propositional formula $[M^{\varphi,\iota}]_k$ is defined as follows:*

$$[M^{\varphi,\iota}]_k := I_\iota(w_{0,0}) \wedge \bigwedge_{j=1}^{f_k(\varphi)} \bigwedge_{i=0}^{k-1} TR(w_{i,j}, w_{i+1,j})$$

where $w_{0,0}$, and $w_{i,j}$ for $0 \leq i \leq k$ and $1 \leq j \leq f_k(\varphi)$ are global state variables. $[M^{\varphi,\iota}]_k$ encodes the initial state ι by $w_{0,0}$ and constrains the $f_k(\varphi)$ symbolic k-paths to be valid k-computations in M_k.

The next step of the algorithm consists in translating an ECTLKD formula φ into a propositional formula.

Definition 13 (Translation of ECTLKD **formulas).** *Let a model M_k with initial state ι, and an* ECTLKD *formula φ be given. We inductively define the translation of φ at state $w_{m,n}$ into the propositional formula $[\varphi]_k^{[m,n]}$ as follows:*

$$
\begin{array}{ll}
[p]_k^{[m,n]} & := p(w_{m,n}), \\
[\neg p]_k^{[m,n]} & := \neg p(w_{m,n}), \\
[\alpha \wedge \beta]_k^{[m,n]} & := [\alpha]_k^{[m,n]} \wedge [\beta]_k^{[m,n]}, \\
[\alpha \vee \beta]_k^{[m,n]} & := [\alpha]_k^{[m,n]} \vee [\beta]_k^{[m,n]}, \\
[\mathrm{EX}\alpha]_k^{[m,n]} & := \bigvee_{i=1}^{f_k(\varphi)} \Big(H(w_{m,n}, w_{0,i}) \wedge [\alpha]_k^{[1,i]} \Big), \\
[\mathrm{EG}\alpha]_k^{[m,n]} & := \bigvee_{i=1}^{f_k(\varphi)} \Big(H(w_{m,n}, w_{0,i}) \wedge (\bigvee_{l=0}^{k} L_{k,i}(l)) \wedge \bigwedge_{j=0}^{k} [\alpha]_k^{[j,i]} \Big), \\
[\mathrm{E}(\alpha \mathrm{U} \beta)]_k^{[m,n]} & := \bigvee_{i=1}^{f_k(\varphi)} \Big(H(w_{m,n}, w_{0,i}) \wedge \bigvee_{j=0}^{k} \big([\beta]_k^{[j,i]} \wedge \bigwedge_{t=0}^{j-1} [\alpha]_k^{[t,i]} \big) \Big), \\
[\mathcal{P}_l \alpha]_k^{[m,n]} & := \bigvee_{i=1}^{f_k(\varphi)} \Big(I_\iota(w_{0,i}) \wedge \bigvee_{j=0}^{k} \big([\alpha]_k^{[j,i]} \wedge HP_l(w_{j,i}) \big) \Big), \\
[\overline{\mathrm{K}}_l \alpha]_k^{[m,n]} & := \bigvee_{i=1}^{f_k(\varphi)} \Big(I_\iota(w_{0,i}) \wedge \bigvee_{j=0}^{k} \big([\alpha]_k^{[j,i]} \wedge HK_l(w_{m,n}, w_{j,i}) \big) \Big).
\end{array}
$$

The meaning of the translations above can be intuitively reconstructed from the definition of propositional formulas presented earlier. For example, the formula $[\mathrm{EX}\alpha]_k^{[m,n]}$ expresses the condition that there exists a sub-path starting from $w_{m,n}$ in which the first point $w_{0,i}$ in this computation satisfies α. For $[\mathcal{P}_l\alpha]_k^{[m,n]}$ we insist on the existence of a point $w_{j,i}$ in which agent l is in a green local state, and that it is accessible from the initial state by some computation. For $[\overline{\mathrm{K}}_l\alpha]_k^{[m,n]}$ we insist on the existence of a point $w_{j,i}$ in which agent l is in the same local state, and that it is accessible from the initial state by some computation.

Given the translations above, we can now check φ over M_k by checking the satisfiability of the propositional formula $[M^{\varphi,\iota}]_k \wedge [\varphi]_{M_k}$, where $[\varphi]_{M_k} = [\varphi]_k^{[0,0]}$. The translation presented above is shown to be correct and complete in the next section.

6 Correctness of the translation

In this section we prove the correctness of the translation of the model checking problem into the SAT-problem as given by Definition 12.

Lemma 3. *$M_k, s \models \varphi$ iff there is a submodel M_k' of M_k with $|{P_k}'| \leq f_k(\varphi)$ such that $M_k', s \models \varphi$.*

Proof. (=>) By structural induction on φ. The lemma follows directly for the propositional variables and their negations.

Assume that the hypothesis holds for all the proper sub-formulas of φ.

- $\varphi = \alpha \vee \beta \mid \alpha \wedge \beta$. Straightforward.
- $\varphi = \mathrm{EX}\alpha \mid \mathrm{EG}\alpha \mid \mathrm{E}(\alpha \mathrm{U} \beta)$. By induction hypothesis — see [14] page 143.
- Let $\varphi = \mathcal{P}_i\alpha$. If $M_k, s \models \mathcal{P}_i\alpha$, then by definition: $(\exists \pi \in P_k)\big(\pi(0) = \iota$ and $\exists_{0 \leq j \leq k}(sR_i^O\pi(j))$ and $\pi(j) \models \alpha)\big)$. By the inductive assumption there is a submodel $M_k' = (\mathcal{DS}', \iota, {P_k}', R_1^{\prime O}, \ldots, R_n^{\prime O}, R_1^{\prime K}, \ldots, R_n^{\prime K}, \mathcal{V}')$ of M_k such that $|{P_k}'| \leq f_k(\alpha)$ and $M_k', \pi(j) \models \alpha$.
  Consider a submodel $M_k'' = (\mathcal{DS}'', {P_k}'', R_1^{\prime\prime O}, \ldots, R_n^{\prime\prime O}, R_1^{\prime\prime K}, \ldots, R_n^{\prime\prime K}, \iota, \mathcal{V}'')$ of M_k, where ${P_k}'' = {P_k}' \cup \{\pi\}$ and $\mathcal{DS}'' = \mathit{States}({P_k}'') \cup \{s\}$. Since π belongs to ${P_k}''$, by the construction of M_k'' and the definition of the bounded semantics, we have that $M_k'', s \models \mathcal{P}_i\alpha$ and $|{P_k}''| \leq f_k(\varphi) = f_k(\alpha) + 1$.
- Let $\varphi = \overline{\mathrm{K}}_i\alpha$. If $M_k, s \models \overline{\mathrm{K}}_i\alpha$, then by definition:
  $(\exists \pi \in P_k)\big(\pi(0) = \iota$ and $\exists_{0 \leq j \leq k}(sR_i^K\pi(j))$ and $\pi(j) \models \alpha)\big)$. By induction there is a submodel $M_k' = (\mathcal{DS}', \iota, {P_k}', R_1^{\prime O}, \ldots, R_n^{\prime O}, R_1^{\prime K}, \ldots, R_n^{\prime K}, \mathcal{V}')$ of M_k such that $|{P_k}'| \leq f_k(\alpha)$ and $M_k', \pi(j) \models \alpha$.
  Consider a submodel $M_k'' = (\mathcal{DS}'', \iota, {P_k}'', R_1^{\prime\prime O}, \ldots, R_n^{\prime\prime O}, R_1^{\prime\prime K}, \ldots, R_n^{\prime\prime K}, \mathcal{V}'')$ of M_k, where ${P_k}'' = {P_k}' \cup \{\pi\}$ and $\mathcal{DS}'' = \mathit{States}({P_k}'') \cup \{s\}$. Since π belongs to ${P_k}''$, by the construction of M_k'' and the definition of the bounded semantics, we have that $M_k'', s \models \overline{\mathrm{K}}_i\alpha$ and $|{P_k}''| \leq f_k(\varphi) = f_k(\alpha) + 1$.

(<=) The proof is straightforward.

From Lemma 3 we can now derive the following.

Corollary 1. *$M \models_k \varphi$ iff there is a submodel M'_k of M_k with $|P_k{}'| \leq f_k(\varphi)$ such that $M'_k, \iota \models \varphi$.*

Proof. It follows from Definition 8, and Lemma 3, by using $s = \iota$.

Lemma 4. *For each state s of M, the following condition holds: $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ is satisfiable iff there is a submodel M'_k of M_k with $|P_k{}'| \leq f_k(\varphi)$ such that $M'_k, s \models \varphi$.*

Proof. (=>) Let $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ be satisfiable. By the definition of the translation, the propositional formula $[\varphi]_{M_k}$ encodes all the sets of $k-$computations of size $f_k(\varphi)$ which satisfy the formula φ. By the definition of the unfolding of the transition relation, the propositional formula $[M^{\varphi,s}]_k$ encodes $f_k(\varphi)$ symbolic k-paths to be valid $k-$computations of M_k. Hence, there is a set of $k-$computations in M_k, which satisfies the formula φ of size smaller or equal to $f_k(\varphi)$. Thus, we conclude that there is a submodel M'_k of M_k with $|P_k{}'| \leq f_k(\varphi)$ and $M'_k, s \models \varphi$. The actual definition of M'_k can be reconstructed from Definition 13 and Definition 12.

(<=) The proof is by induction on the length of φ. The lemma follows directly for the propositional variables and their negations. Consider the following cases:

- For $\varphi = \alpha \vee \beta, \alpha \wedge \beta$ or the temporal operators the proof is like in [14].
- Let $\varphi = \mathcal{P}_l\alpha$. If $M'_k, s \models \mathcal{P}_l\alpha$ with $|P_k{}'| \leq f_k(\mathcal{P}_l\alpha)$, then by Definition 7 we have that there is a $k-$computation π such that $\pi(0) = \iota$ and $(\exists j \leq k)\ sR^O_l\pi(j)$ and $M'_k, \pi(j) \models \alpha$. Hence, by induction we obtain that for some $j \leq k$ the propositional formula $[\alpha]^{[0,0]}_k \wedge [M^{\alpha,\pi(j)}]_k$ is satisfiable. Let $ii = f_k(\alpha) + 1$ be the index of a new symbolic $k-$path which satisfies the formula $I_\iota(w_{0,ii})$. Therefore, by the construction above, it follows that the propositional formula $I_\iota(w_{0,ii}) \ \wedge\ \bigvee_{j=0}^{k} \left([\alpha]^{[j,ii]}_k \wedge HP_l(w_{j,ii})\right) \wedge [M^{\mathcal{P}_l\alpha,s}]_k$ is satisfiable. Therefore, the following propositional formula is satisfiable:
  $\bigvee_{1 \leq i \leq f_k(\mathcal{P}_l\alpha)} \Big(I_\iota(w_{0,i}) \ \wedge\ \bigvee_{j=0}^{k} \left([\alpha]^{[j,i]}_k \wedge HP_l(w_{j,i})\right) \wedge [M^{\mathcal{P}_l\alpha,s}]_k\Big)$.
  Hence, by the definition of the translation of an ECTLKD formula, the above formula is equal to the propositional formula $[\mathcal{P}_l\alpha]^{[0,0]}_k \wedge [M^{\mathcal{P}_l\alpha,s}]_k$.
- Let $\varphi = \overline{\mathrm{K}}_l\alpha$. If $M'_k, s \models \overline{\mathrm{K}}_l\alpha$ with $|P_k{}'| \leq f_k(\overline{\mathrm{K}}_l\alpha)$, then by Definition 7 we have that there is a $k-$computation π such that $\pi(0) = \iota$ and $(\exists j \leq k)\ sR^K_l\pi(j)$ and $M'_k, \pi(j) \models \alpha$. Hence, by induction we obtain that for some $j \leq k$ the propositional formula $[\alpha]^{[0,0]}_k \wedge [M^{\alpha,\pi(j)}]_k$ is satisfiable. Let $ii = f_k(\alpha) + 1$ be the index of a new symbolic $k-$path which satisfies the formula $I_\iota(w_{0,ii})$. Therefore, by the construction above, it follows that the propositional formula $I_\iota(w_{0,ii}) \ \wedge\ \bigvee_{j=0}^{k} \left([\alpha]^{[j,ii]}_k \wedge HK_l(w_{0,0}, w_{j,ii})\right) \ \wedge$ $[M^{\overline{\mathrm{K}}_l\alpha,s}]_k$ is satisfiable. Therefore, the following propositional formula is satisfiable:
  $\bigvee_{1 \leq i \leq f_k(\overline{\mathrm{K}}_l\alpha)} \Big(I_\iota(w_{0,i}) \ \wedge\ \bigvee_{j=0}^{k} \left([\alpha]^{[j,i]}_k \wedge HK_l(w_{0,0}, w_{j,i})\right) \wedge [M^{\overline{\mathrm{K}}_l\alpha,s}]_k\Big)$.
  Hence, by the definition of the translation of an ECTLKD formula, the above formula is equal to the propositional formula $[\overline{\mathrm{K}}_l\alpha]^{[0,0]}_k \wedge [M^{\overline{\mathrm{K}}_l\alpha,s}]_k$.

Theorem 2. *Let M be a model, M_k be a $k-$model of M, and φ be an* ECTLKD *formula. Then, $M \models_k \varphi$ iff $[\varphi]_{M_k} \wedge [M^{\varphi,\iota}]_k$ is satisfiable.*

Proof. Follows from Lemmas 3 and 4.

Corollary 2. *$M \models_k \neg\varphi$ iff $[\varphi]_{M_k} \wedge [M^{\varphi,\iota}]_k$ is unsatisfiable for $k = |M|$.*

This concludes our analysis of the translation technique. We now give an example to demonstrate how it can be put into practice.

7 Model checking the bit transmission problem with faults

The bit-transmission problem [8] involves two agents, a *sender* $\mathfrak{S}$, and a *receiver* $\mathfrak{R}$, communicating over a possibly faulty communication channel. $\mathfrak{S}$ wants to communicate some information—the value of a bit for the sake of the example—to $\mathfrak{R}$. One protocol to achieve this is as follows [8]. $\mathfrak{S}$ immediately starts sending the bit to $\mathfrak{R}$, and continues to do so until it receives an acknowledgement from $\mathfrak{R}$. $\mathfrak{R}$ does nothing until it receives the bit; from then on it sends acknowledgements of receipt to $\mathfrak{S}$. $\mathfrak{S}$ stops sending the bit to $\mathfrak{R}$ when it receives an acknowledgement. Note that $\mathfrak{R}$ will continue sending acknowledgements even after $\mathfrak{S}$ has received its acknowledgement. Intuitively $\mathfrak{S}$ will know for sure that the bit has been received by $\mathfrak{R}$ when it gets an acknowledgement from $\mathfrak{R}$. $\mathfrak{R}$, on the other hand, will never be able to know whether its acknowledgement has been received since $\mathfrak{S}$ does not answer the acknowledgement We refer to [8, 9] for further discussion.

In this section we are interested in applying the machinery of bounded model checking to verify a version of the scenario above where one agent does not operate as it is supposed to. This version of the scenario was first described in [10]. In particular we examine in detail only the possibility that $\mathfrak{R}$ is faulty[3]. Specifically, we shall consider in this section the possibility that $\mathfrak{R}$ may send acknowledgements without having received the bit. This is a simple example of an agent not following its specification. This scenario can be analysed by means of deontic interpreted systems. We report here briefly part of the analysis that was conducted in [10], and then proceed to model check the example.

There are three active components in the scenario: a sender, a receiver, and a communication channel. In line with the spirit of the formalism of (deontic) interpreted systems, it is convenient to see sender and receiver as agents, and the communication channel as the environment. Each of these can be modelled by considering their local states [4]. For the sender $\mathfrak{S}$, it is enough to consider four possible local states and since we are not admitting the possibility of faults, its

[3] The possibility that $\mathfrak{S}$ is faulty, and other combinations of faulty $\mathfrak{R}$, $\mathfrak{S}$ and $\mathfrak{E}$, can be treated in similar fashion.

[4] Recall that, in order to apply the machinery of deontic interpreted systems we have to split the set of local states of agent i into two disjoint sets: green ($\mathcal{G}_i$) and red ($\mathcal{R}_i$), representing correct and incorrect functioning behaviour respectively.

local states are all green. They represent the value of the bit that $\mathfrak{S}$ is attempting to transmit, and whether or not $\mathfrak{S}$ has received an acknowledgement from $\mathfrak{R}$. We thus have: $L_{\mathfrak{S}} = \mathcal{G}_{\mathfrak{S}} = \{0,\ 1,\ 0\text{-}ack,\ 1\text{-}ack\}$, $\mathcal{R}_{\mathfrak{S}} = \emptyset$. For the environment it is enough to consider a singleton: $L_{\mathfrak{E}} = \{\cdot\}$. Moreover, we assume that all local states of the environment are green, so we have: $L_{\mathfrak{E}} = \mathcal{G}_{\mathfrak{E}}$, $\mathcal{R}_{\mathfrak{E}} = \emptyset$. It remains to model the local states of the receiver $\mathfrak{R}$. Six different local states are enough to capture the state of $\mathfrak{R}$: the value of the received bit, the circumstance in which no bit has been received yet (represented by ϵ), the circumstance in which $\mathfrak{R}$ has sent an acknowledgement without having received the value of the bit (denoted by $\epsilon\text{-}ack$), and the circumstance in which $\mathfrak{R}$ has sent an acknowledgement having received the value of the bit (represented by $0\text{-}ack$ and $1\text{-}ack$). So, the local states of $\mathfrak{R}$ for this version of the problem are defined as follows: $L_{\mathfrak{R}} = \{0, 1, \epsilon, 0\text{-}ack, 1\text{-}ack, \epsilon\text{-}ack\}$ with $\mathcal{G}_{\mathfrak{R}} = \{0, 1, \epsilon,\text{0-ack,1-ack}\}$, $\mathcal{R}_{\mathfrak{R}} = \{\epsilon\text{-}ack\}$.

The set of actions available to the agents are as follows: $Act_{\mathfrak{S}} = \{sendbit,\ \lambda\}$, $Act_{\mathfrak{R}} = \{sendack,\ \lambda\}$, where λ stands for no action ('no-op'). The actions $Act_{\mathfrak{E}}$ for the environment correspond to the transmission of messages between $\mathfrak{S}$ and $\mathfrak{R}$ on the unreliable communication channel. We will assume that the communication channel can transmit messages in both directions simultaneously, and that a message travelling in one direction can get through while a message travelling in the opposite direction is lost. The set of actions for the environment is $Act_{\mathfrak{E}} = \{\leftrightarrow,\ \rightarrow,\ \leftarrow,\ -\}$, where $\leftrightarrow$ represents the action in which the channel transmits any message successfully in both directions, $\rightarrow$ that it transmits successfully from $\mathfrak{S}$ to $\mathfrak{R}$ but loses any message from $\mathfrak{R}$ to $\mathfrak{S}$, $\leftarrow$ that it transmits successfully from $\mathfrak{R}$ to $\mathfrak{S}$ but loses any message from $\mathfrak{S}$ to $\mathfrak{R}$, and $-$ that it loses any messages sent in either direction.

The protocols the agents are running are as follows:

- $P_{\mathfrak{S}}(0) = P_{\mathfrak{S}}(1) = \{sendbit\}$, $P_{\mathfrak{S}}(0\text{-}ack) = P_{\mathfrak{S}}(1\text{-}ack) = \{\lambda\}$,
- $P_{\mathfrak{R}}(0) = P_{\mathfrak{R}}(1) = P_{\mathfrak{R}}(0\text{-}ack) = P_{\mathfrak{R}}(1\text{-}ack) = \{sendack\}$, $P_{\mathfrak{R}}(\epsilon) = P_{\mathfrak{R}}(\epsilon\text{-}ack) = \{\lambda\}$,
- $P_{\mathfrak{E}}(l_{\mathfrak{E}}) = Act_{\mathfrak{E}} = \{\leftrightarrow,\ \rightarrow,\ \leftarrow,\ -\}$, for all $l_{\mathfrak{E}} \in L_{\mathfrak{E}}$.

It should be straightforward to infer the transition system that is induced by the informal description of the scenario we considered above together with the local states and protocols defined above. We refer to [10] for further discussion.

We now encode the local states in binary form in order to use them in the model checking technique. Since the sender $\mathfrak{S}$ can be in 4 different local green states we shall need 2 bits to encode its state; we take: $(0,0) = 0$, $(0,1) = 1$, $(1,0) = 0\text{-}ack$, $(1,1) = 1\text{-}ack$. Since the receiver $\mathfrak{R}$ can be in 5 different local green states and in 1 different local red states, we shall need 3 + 1 bits to encode its state; we take: $(1,0,0;0) = 0$, $(0,1,0;0) = 1$, $(0,0,0;0) = \epsilon$, $(1,1,0;0) = 0\text{-}ack$, $(1,1,0;1) = 1\text{-}ack$, $(1,1,1;0) = \epsilon\text{-}ack$. The modelling of the environment $\mathfrak{E}$ requires only one bit: $(0) = \cdot$

In view of this, a global state is modelled by a byte: $s = (s[1], s[2], s[3], s[4], s[5], s[6], s[7])$. For instance the initial state $\iota = (0, \epsilon, \cdot)$ is represented as a tuple of seven 0's. If we want to represent it in terms of propositional variables, we shall have to insist on the propositions encoding the state to be in the state of false. In other words, we would encode the initial state as follows: $I_{\iota}(w_{0,0}) = \bigwedge_{i=1}^{7} \neg w_{0,0}[i]$.

Let $\mathcal{PV} = \{\mathbf{bit=0}, \mathbf{bit=1}, \mathbf{recbit}, \mathbf{recack}\}$. We use the following interpretation for the proposition variables in $\mathcal{PV}$:

$$\begin{aligned}
(M,s) &\models \mathbf{bit=0} \text{ if } l_{\mathfrak{S}}(s) = 0 \text{ or } l_{\mathfrak{S}}(s) = 0\text{-}ack,\\
(M,s) &\models \mathbf{bit=1} \text{ if } l_{\mathfrak{S}}(s) = 1 \text{ or } l_{\mathfrak{S}}(s) = 1\text{-}ack,\\
(M,s) &\models \mathbf{recbit} \text{ if } l_{\mathfrak{R}}(s) = 1 \text{ or } l_{\mathfrak{R}}(s) = 0 \text{ or } l_{\mathfrak{R}}(s) = 0\text{-}ack \text{ or } l_{\mathfrak{R}}(s) = 1\text{-}ack,\\
(M,s) &\models \mathbf{recack} \text{ if } l_{\mathfrak{S}}(s) = 1\text{-}ack \text{ or } l_{\mathfrak{S}}(s) = 0\text{-}ack.
\end{aligned}$$

Some properties we may be interested in checking for the example above are the following:

1. $\mathrm{AG}\Big(\neg\mathbf{recack} \vee \mathrm{K}_{\mathfrak{S}}\big(\mathcal{O}_{\mathfrak{R}}\big(\mathrm{K}_{\mathfrak{R}}(\mathbf{bit=0}) \vee \mathrm{K}_{\mathfrak{R}}(\mathbf{bit=1})\big)\big)\Big)$
2. $\mathcal{O}_{\mathfrak{R}}(\mathbf{recack} \wedge \neg\big(\mathrm{K}_{\mathfrak{R}}(\mathbf{bit=0}) \vee \mathrm{K}_{\mathfrak{R}}(\mathbf{bit=1})\big))$
3. $\mathrm{AG}\Big(\mathcal{O}_{\mathfrak{R}}\big(\mathrm{K}_{\mathfrak{S}}\big(\mathrm{K}_{\mathfrak{R}}(\mathbf{bit=0}) \vee \mathrm{K}_{\mathfrak{R}}(\mathbf{bit=1})\big)\big)\Big)$
4. $\mathrm{A}\Big(\mathcal{O}_{\mathfrak{R}}\big(\mathrm{K}_{\mathfrak{R}}(\mathbf{bit=0}) \vee \mathrm{K}_{\mathfrak{R}}(\mathbf{bit=1})\big)\mathrm{U}\ \mathbf{recack}\Big)$

Property 1) says that forever in the future if an *ack* is received by $\mathfrak{S}$, then $\mathfrak{S}$ knows that in all the states where $\mathfrak{R}$ is functioning correctly, $\mathfrak{R}$ knows the value of the bit. Property 2) states that in all the states where $\mathfrak{R}$ is functioning correctly $\mathfrak{S}$ has received an acknowledgement and $\mathfrak{R}$ does not know the value of the bit. Property 3) says that forever in the future in all the states where $\mathfrak{R}$ is functioning correctly, $\mathfrak{S}$ knows that $\mathfrak{R}$ knows the value of the bit. Property 4) says that at one point at the future an *ack* is received by $\mathfrak{S}$ and at all the preceding points in time in all states where $\mathfrak{R}$ was operating as intended $\mathfrak{R}$ knew the value of the bit.

The property 1 is true on the interpreted system in consideration, but the properties 2, 3 and 4 are not. The formula 1 is an ACTLKD formula, so in order to check it we shall have to encode the whole model. We can do this in the BMC technique reported above, but, as mentioned already, the benefits of BMC are most apparent when only a *fraction* of the model is generated. For example this happens in formulas 2, 3 and 4 where we need to check validity of an ECTLKD formula in the model. For the purposes of this paper we check validity of the formula 3. The negated formula is:

$$\varphi := \mathrm{EF}\Big(\mathcal{P}_{\mathfrak{R}}\big(\overline{\mathrm{K}}_{\mathfrak{S}}\big(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit=0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit=1})\big)\big)\Big)$$

The translation for the propositions used in φ is as follows: $(\mathbf{bit=0})(w) := (\neg w[1] \wedge \neg w[2]) \vee (w[1] \wedge \neg w[2])$, which means that $(\mathbf{bit=0})$ holds at all the global states with the first local state equal to $(0,0)$ or $(1,0)$. $(\mathbf{bit=1})(w) := (\neg w[1] \wedge w[2]) \vee (w[1] \wedge w[2])$, which means that $(\mathbf{bit=1})$ holds at all the global states with the first local state equal to $(0,1)$ or $(1,1)$.

The translation for the equality of the $\mathfrak{R}$-local states is as follows: $HK_{\mathfrak{R}}(w,v) = \bigwedge_{i=3}^{6} w[i] \Leftrightarrow v[i]$, and the translation of an accessibility of a global state in which

$\mathfrak{R}$ is running correctly is as follows: $HP_{\mathfrak{R}}(v) = (v[3] \wedge \neg v[4]) \vee (\neg v[3] \wedge v[4]) \vee (\neg v[3] \wedge \neg v[4])$. The translation of the equality of the $\mathfrak{S}$-local states is as follows: $HK_{\mathfrak{S}}(w, v) = \bigwedge_{i=1}^{2} w[i] \Leftrightarrow v[i]$.

We calculate that $f_k(\varphi) = 5$ for all $k \in \mathbb{N}_+$ (see Definition 10), so we need to exploit five symbolic $k-$paths. To proceed with the translation, the first thing we need to translate is the initial state $\iota = (0, \epsilon, \cdot)$, where ι is binary represented by $(0, \ldots, 0)$. With the representation above this will be encoded by the propositional formula $I_\iota(w_{0,j}) := \bigwedge_{i=1}^{7} \neg w_{0,j}[i]$, for $0 \leq j \leq 5$.

The next step is to translate the transitions $T(w_{i,j}, w_{i+1,j})$; for simplicity we report only on one transition for the case $k = 1$, and in particular only on the formula $T(w_{0,1}, w_{1,1})$ representing the first transition of the first path. The remaining formulas are $T(w_{0,2}, w_{1,2})$, $T(w_{0,3}, w_{1,3})$, $T(w_{0,4}, w_{1,4})$ and $T(w_{0,5}, w_{1,5})$.

To encode the whole example we should model all the transitions for all the k's starting from $k := 1$. We do not do it here. Let us now encode the formula φ we would like to check.
$[\varphi]_1^{[0,0]} :=$
$\bigvee_{i=1}^{5} \Big(H(w_{m,n}, w_{0,i}) \;\wedge\; \bigvee_{j=0}^{k} [\mathcal{P}_{\mathfrak{R}}(\overline{\mathrm{K}}_{\mathfrak{S}}(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{1})))]_k^{[j,i]}\Big)$
Next:
$[\mathcal{P}_{\mathfrak{R}}(\overline{\mathrm{K}}_{\mathfrak{S}}(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{1})))]_k^{[j,i]} := \bigvee_{i=1}^{5} \Big(I_\iota(w_{0,i}) \wedge$
$\bigvee_{j=0}^{k} (HP_{\mathfrak{R}}(w_{j,i}) \;\wedge\; [\overline{\mathrm{K}}_{\mathfrak{S}}(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{1}))]_k^{[j,i]})\Big)$
Next:
$[\overline{\mathrm{K}}_{\mathfrak{S}}(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{1}))]_k^{[j,i]} := \bigvee_{i=1}^{5} \Big(I_\iota(w_{0,i}) \wedge$
$\bigvee_{j=0}^{k} ([(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{1}))]_k^{[j,i]} \;\wedge\; HK_{\mathfrak{S}}(w_{m,n}, w_{j,i}))\Big)$
Next:
$[(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{1}))]_k^{[j,i]} = [\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{0})]_k^{[j,i]} \wedge [\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{1})]_k^{[j,i]}$
Next:
$[\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{0})]_k^{[j,i]} := \bigvee_{i=1}^{5} \Big(I_\iota(w_{0,i}) \wedge \bigvee_{j=0}^{k} ([\neg(\mathbf{bit} = \mathbf{0})]_k^{[j,i]} \wedge HK_{\mathfrak{R}}(w_{m,n}, w_{j,i}))\Big)$
$[\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit} = \mathbf{1})]_k^{[j,i]} := \bigvee_{i=1}^{5} \Big(I_\iota(w_{0,i}) \wedge \bigvee_{j=0}^{k} ([\neg(\mathbf{bit} = \mathbf{1})]_k^{[j,i]} \wedge HK_{\mathfrak{R}}(w_{m,n}, w_{j,i}))\Big)$
Next:
$[\neg(\mathbf{bit} = \mathbf{0})]_k^{[j,i]} := \neg(\mathbf{bit} = \mathbf{0})(w_{j,i})$ and $[\neg(\mathbf{bit} = \mathbf{1})]_k^{[j,i]} := \neg(\mathbf{bit} = \mathbf{1})(w_{j,i})$.

Checking that the bit transmission protocol satisfies the temporal deontic formula above can now be done by feeding a SAT solver with the propositional formula generated in this method. This would produce a solution, thereby proving that the propositional formula is satisfiable.

8 Conclusions

In this paper we extended the methodology of bounded model checking for CTLK, presented in [13] by adding the deontic notion of correct functioning behaviours of the agents. This notion was explored in [10, 11].

Future work include an implementation of the algorithm presented here and a careful evaluation of experimental results to be obtained.

References

1. N. Amla, R. Kurshan, K. McMillan, and R. Medel. Experimental analysis of different techniques for bounded model checking. In *Proc. of TACAS'03*, volume 2619 of *LNCS*, pages 34–48. Springer-Verlag, 2003.
2. A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu. Bounded Model Checking. In *Advances in Computers*, volume 58. Academic Press, 2003. pre-print.
3. E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.
4. E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, 1999.
5. F. Copty, L. Fix, R. Fraer, E. Giunchiglia, G. Kamhi, A. Tacchella, and M. Vardi. Benefits of bounded model checking at an industrial setting. In *Proc. of CAV'01*, volume 2102 of *LNCS*, pages 436–453. Springer-Verlag, 2001.
6. D. Dennet. *The Intentional Stance.* MIT Press, 1987.
7. E. A. Emerson and E. M. Clarke. Using branching-time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.
8. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge.* MIT Press, Cambridge, 1995.
9. J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: Knowledge-based derivations and correctness proofs for a family of protocols. *Journal of the ACM*, 39(3):449–478, 1992.
10. A. Lomuscio and M. Sergot. Violation, error recovery, and enforcement in the bit transmission problem. In *Proceedings of DEON'02*, London, May 2002. *Journal of Applied Logic*, 2:93-116, Elsevier, 2004.
11. A. Lomuscio and M. Sergot. Deontic interpreted systems. *Studia Logica*, 75, 2003.
12. K. McMillan. The SMV system. Technical Report CMU-CS-92-131, Carnegie-Mellon University, February 1992.
13. W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.
14. W. Penczek, B. Woźna, and A. Zbrzezny. Bounded model checking for the universal fragment of CTL. *Fundamenta Informaticae*, 51(1-2):135–156, 2002.
15. M. Wooldridge. *An introduction to multi-agent systems.* John Wiley, England, 2002.

On Epistemic Temporal Strategic Logic

Sieuwert van Otterloo
University of Liverpool
sieuwert@csc.liv.ac.uk

Geert Jonker
Universiteit Utrecht
geertj@cs.uu.nl

May 15, 2004

Abstract

ATEL is one of the most expressive logics for reasoning about knowledge, time and strategies. Several issues around the interpretation of this logic are still unresolved. This paper contributes to the ongoing discussion by showing that agents do not have to know a specific strategy for doing something in order to have a capability. Furthermore we claim that agents can possess so-called strategic knowledge that is derived from their knowledge of strategies being played. In order to prove these claims we present an alternative interpretation of ATEL over extensive game forms. For the definition of abilities we use strategy domination, and to deal with strategic knowledge we include strategy profiles in the model. We illustrate the interpretation issues mentioned using several small examples. Furthermore we show how perfect recall and perfect memory can be characterized.

1 Introduction

Reasoning about knowledge, time and strategies is important in the analysis of multi agent systems. Several logical frameworks and languages have been developed to capture these notions. In this paper we discuss one such language called ATEL [15], which is a logic to reason about knowledge, time and strategies. This logic can be applied to the formal analysis of a wide range of systems, for instance distributed protocols, synchronisation and security, but also to any issue that can be modeled as an extensive game, such as argumentation, auctions of language games. In this paper we are concerned with the interpretation of ATEL. The original interpretation of the language featured attractive computational properties, but suffered from some counterintuitive properties, which led to a number of refinements [16, 9, 8]. In this paper we provide a new interpretation for ATEL. In order to keep things simple we do this for turn based systems, but we see no reason why the approach could not be extended to the more general class of concurrent systems.

The history of ATEL started with the definition of Computation Tree Logic, a logic for branching time models [4]. Using this logic one can express many temporal properties of distributed or concurrent systems, such as 'all possible computations will reach this state' or 'this state eventually occurs in at least one computation'. It was discovered by Alur and others [1] that this logic could be extended to reason about multiagent systems without changing the complexity of model checking. They extended the language with coalition operators and called it Alternating-time Temporal Logic (ATL). Using ATL one could express properties like 'This coalition can ensure that a state is reached' or 'This coalition cannot avoid that p always holds'. Van der Hoek and Wooldridge realized that it would be useful to reason about the knowledge of agents within ATL, and extended the language with a knowledge operator calling it ATEL [15]. They gave the knowledge operator a very intuitive interpreted systems semantics [6] but did not alter the notion of a strategy used in the coalition operator. The result was that agents were assumed to always be able to make different choices in different states, even if the agent could not distinguish these states. Assuming that agents can make different choices in states they cannot distinguish is counterintuitive [9] and different semantics along the lines of imperfect recall ATL [13] were developed by Jonker, Van der Hoek and Jamroga [9, 8].

In plain English, ATEL originally assumed that a coalition of agents can achieve something if a strategy to achieve it exists. This condition has been strengthened in the sources cited:

- A coalition of agents can achieve X if they have a uniform strategy (or strategy-under-incomplete-knowledge) that achieves X [9]

- A coalition of agents can achieve X if they have a uniform strategy of which they know that it achieves X. [9, 8]

In section 2 we define the notion of a uniform strategy formally, but it can be thought of as a strategy with extra restrictions so that it does not use facts that an agent is not supposed to know. One of the main points of this paper is that this latter condition is too strong. A coalition does not have to be able to identify a strategy of which they know it will be successful. A coalition of sensible agents will choose, if they cannot identify a foolproof strategy, the best strategy that they can come up with. More precisely, they will choose a strategy that is not dominated by any other strategy. We say that a coalition of agents can do X if any undominated strategy achieves X. We will define domination later in this paper.

A second point of this paper is that the knowledge of a coalition does not have to depend only on the state of a game or system, but also on the strategies they employ. It seems safe to assume that agents know what strategies they employ, and that this gives them extra information about the future. This phenomenon can be called strategic knowledge [5]. The interpretation developed here addresses this issue by assuming all agents know their own strategy. We stress that this is intended not as the final answer on this issue, but as a demonstration how one can incorporate some form of strategic knowledge.

The language CTL is in fact a syntactical restricted subset of the language CTL*. The same is true for ATL and ATL* and also for ATEL and ATEL*. The unstarred versions have received the most attention, because they have a low model checking complexity [16]. In this paper however we are more interested in the meaning of interpretations for the languages than in complexity. Therefore we prefer to work with ATEL*, the language without restrictions, rather than to define an interpretation only for ATEL.

In section 2 we present necessary definitions. Section 3 contains examples. In section 4 we define an interpretation for ATEL*. Section 5 uses the logic to analyse the examples of section 3. In section 6 two theorems are proven and section 7 is the conclusion.

2 Extensive Game Forms

Games are models for interaction between agents with different and possibly competing objectives [12]. An extensive game gives a detailed description of such interaction. It shows which decisions are made in order to reach an outcome. It can be represented in a game tree, where each leaf corresponds to an outcome and each node corresponds to a choice between options. The preferences of all agents are part of an extensive game. In many cases one wants to study the structure of a game independent of the preferences of the agents. In that case one can use the idea of a game form, which is an extensive game without preference function. The notion of an extensive game goes back to Kuhn [11]. We have adapted the next definition from Osborne [12]. Since the structure encodes the fact that agents may not be sure what exactly the current state is when making a decision, we speak of an extensive game with *imperfect information.*

We have adopted notation fashionable in game theory [12] and notation used in coalition logic [16]. The set of all agents is denoted Σ and Γ is used for a subset of agents. Individual agents are denoted X, Y. For game forms and game form interpretations F, G are used. Strategies are called S, T or S_Γ, T_Γ, to indicate for which group the strategy is. A game form and a strategy together form a model, denoted by M. Formulas are denoted ϕ, ψ and atomic propositions p, q. P is a set of propositions. They are interpreted using a function π. Actions are a, b and histories are called h, h' or j. To improve readability, we sometimes abuse notation a little bit and write XY for the set of agents $\{X, Y\}$.

Game Form A *game form* is a tuple $(\Sigma, H, Ow, \sim)$, where Σ is a finite set of agents and H is a non-empty set of *histories.* The set H must be prefix-closed, which means that for any sequence $ha \in H$ also $h \in H$. We use the special symbol ϵ to denote the empty sequence. The set of all actions available after h is denoted $A(h) = \{a | ha \in H\}$. A history $h \in H$ is terminal if $A(h) = \emptyset$. The set of all terminal histories of H is denoted $Z(H)$.

The function $Ow(h) : H \setminus Z(H) \to \Sigma$ defines which player chooses the next action. Intuitively the agent $Ow(h)$ owns the history h, but we can also say that it decides h, controls h or has the initiative in h. For each agent $X \in \Sigma$, the relation $\sim_X$ is an equivalence relation between histories, where $h \sim_X j$ expresses the fact that agent X cannot tell the difference between having gone through history h and history j. One condition applies: if $Ow(h) = X$ and $h' \sim_X h$ then also $Ow(h') = X$ and further $A(h) = A(h')$. This condition ensures that an agent knows when it can select a action and that it knows which actions are available. This definition is taken from Osborne and Rubinstein [12], definition 200.1, where it is called an extensive game form. We have extended the information sets such that agents also have information when they are not in charge, which is a common extension for logical purposes [14, 3].

Game Form Interpretation A *game form interpretation* is defined as a tuple $(\Sigma, H, Ow, \sim, P, \pi)$. The first elements $(\Sigma, H, Ow, \sim)$ are a game form. The set P contains propositions and $\pi : H \to 2^P$ is a function that assigns to each history the set of propositions that are true in the final state of that history. The idea is that these propositions can be used to refer to certain histories, for instance to histories where an agent achieves a certain goal.

Strategies A strategy S_Γ for a coalition Γ is a function that takes a history h such that $Ow(h) \in \Gamma$ and returns a non-empty set of actions $S_\Gamma(h)$ such that $S_\Gamma(h) \subseteq A(h)$. This means that strategies can be non-deterministic. We sometimes call the strategy S_Γ a *strategy profile* to indicate that it contains a strategy for every agent in Γ. There is no fundamental difference between strategies and strategy profiles in this paper. A strategy S_Γ is uniform iff for all h, j with $h \sim_{Owh} j$ it is the case that $S_\Gamma(h) = S_\Gamma(j)$. A uniform strategy thus prescribes the same actions in histories that an agent cannot distinguish.

For the purposes of this paper, we think of a game form as a description of a commonly known protocol for interaction between autonomous agents. All agents have preferences and it is common knowledge that these are private, thus not known to other agents. It is also commonly known that agents with different objectives cannot communicate outside the structure of the game. If an agent adopts a certain strategy, for instance to reach a certain goal, then the agent itself knows which strategy it is playing, but other agents do not know this. If a coalition of agents has a goal, then it is assumed that agents have the right means of coordination in order to select a group strategy.

Mental capabilities are included in the definition of a game form. We assume that if an agent is for instance forgetful, that this has been encoded in the equivalence relation in the game form. The relations in the game form thus represent what an agent knows from all its sources, not just its observations. Under this assumption it makes sense to relate properties of agents, such as perfect recall and perfect memory in section 6, to properties of the equivalence relations.

3 Scenarios

To illustrate the principles of ATEL covered in this paper we will first have a look at some examples. The first game form we study is game form G_1 in figure 1. In this game agent A can first choose between action 1 and action 2. Agent B then chooses for either action 3 or action 4. The dashed line indicated that agent B does not know what agent A has done when B has to choose. For this game form we are interested in the strategic abilities of agent B. In the original ATEL interpretation agent B has a strategy in t_0 to satisfy q. If A chooses action 1 it would chooses action 3, if A chooses action 2 it would choose action 4. However, at t_1 agent B doesn't know whether agent A has played action 1 or 2. In terms of Jonker [9], agent B doesn't have a uniform strategy to satisfy q in t_0. In t_1 things are a bit different. Agent B does have a uniform strategy to achieve q, but doesn't know which one. In terms of Jonker and Van der Hoek and Jamroga, it cannot identify the right strategy. If agent B would want to satisfy p, it would be able to identify a uniform strategy in t_1: choosing action 3.

But what if agent B would want to satisfy $p \wedge q$? At t_1 it wouldn't be able to identify a uniform strategy, since it doesn't know whether agent A has played action 1 or 2. Nevertheless, it has a 'best bet' strategy: playing action 3. In the view of the agent, this strategy *might* make it reach its goal. There is no other strategy that is better than this one, so we call this strategy *undominated*. It would be dominated if there existed a strategy that performed better in some of the indistinguishable states, and equally in the others. We will define this more formally in section 4. Let us now assume that we are in the left situation, after action 1. We notice that if agent B plays his 'best bet' strategy, it will achieve his goal. We therefore say that agent B is *able* to satisfy $p \wedge q$, or that $p \wedge q$ is *achievable* for agent B.

Game form G_2 in figure 2 illustrates that an agent can have several undominated strategies. We assume the current situation is the one after action 2. Agent B wants to make p true in the next states. The agent has two actions to choose from, and it cannot identify which action is best for achieving p. In the left situation, the appropriate strategy would be action 4. In the right situation, the appropriate strategy would be action 5. For the middle situation however it does not matter which strategy the agent chooses. Either action leads to a desired state. Thus, the agent has two strategies that might make him reach his goal and to him it

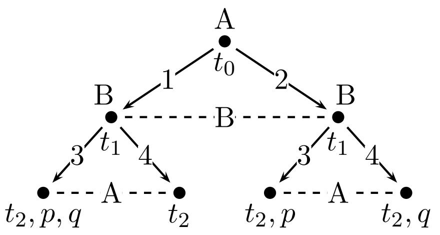


Figure 1: A simple game form G_1

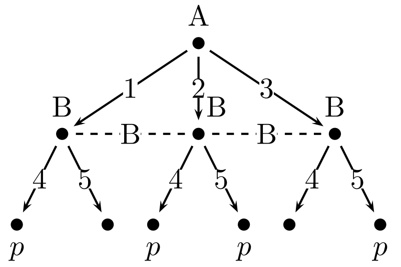


Figure 2: Game form G_2

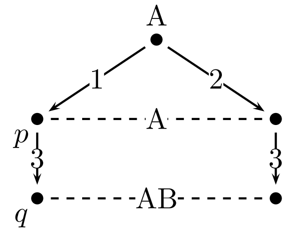


Figure 3: Imperfect Memory Game Form

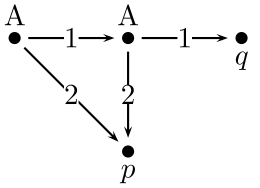


Figure 4: Block Pushing

is not clear whether to use one or the other, they are both undominated. The best he can do is to randomly choose one of the two. Fortunately, since agent A has played action 2, agent B will reach his goal with any of his undominated strategies. Therefore, we say that agent B is *able* to satisfy p, or that p is *achievable* for agent B.

Both in game forms G_1 and G_2, the ability of agent B to reach his goal was facilitated by agent A's choice. In G_1, we saw that agent B was able to satisfy p and q because agent A had chosen to play action 1. We say that agent A has the ability to give agent B the ability to satisfy p and q. The fact that agent A moves before agent B does, does not imply that agent B cannot enable agent A to reach certain goals. If agent B decides beforehand to play action 3, it in a way gives agent A the ability to satisfy (eventually) p and q.

In figure 3 another game form is depicted. This game form is unusual since agents lose information, according to the relations indicated. Agent A seems not to remember, after action 1 that it has chosen this action. Similarly, B knows the difference between histories 1 and 2 but one step later it cannot distinguish 13 and 23. These peculiarities are discussed in section 3 and section 6.

The last scenario pictured in figure 4 deals with the question whether agents can lose abilities by choosing certain strategies. This is a minor issue, not very related to the previous issues. Nevertheless it illustrates one issue in the interpretation of ATEL. Assume that the leftmost state is the current state. The interesting property of this model is that the agent has the ability to make p true in the next state. It cannot get rid of this ability in one step. If we assume that agents can unconditionally commit to strategies, so that they cannot change later their strategy, than A would be able, by committing to go to the rightmost state, to causing itself not to be able to achieve p. This is not what we think of as intuitive. Therefore we assume that agents can always change their own strategy at a later moment.

Finally, we turn back to game form G_1 again. Consider the scenario where agent A has decided to play the strategy of choosing action 2. At time t_0, this influences the knowledge of agent A. It now knows for example that agent B won't be able to satisfy $p \wedge q$ anymore.

This is called strategic knowledge [5]. It is a kind of provisional knowledge; provided that an agent will follow the strategy it has committed to, it has more knowledge about the future. The same holds for a coalition; provided that they follow the same strategy and that this is common knowledge between them, their knowledge about the future increases. The next section shows how strategic knowledge, along with the other principles explained above, can be formalized using a new interpretation of ATEL*.

4 Strategic Temporal Epistemic Logic

The language ATEL* is the smallest language L such that for any formula $\phi, \psi \in L$, any coalition of agents Γ and any agent X it is the case that:

$$\begin{array}{cccc} \phi \vee \psi \in L & \neg\phi \in L & \Box\phi \in L & \phi\,\mathcal{U}\,\psi \in L \\ \bigcirc\phi \in L & K_X\phi \in L & \ll \Gamma \gg \phi \in L & \end{array}$$

This language is in fact a mixture between ATL^* and epistemic logic [6]. The reader will be familiar with disjunction ($\phi \vee \psi$, 'or') and negation ($\neg\phi$, 'not'). The temporal operators $\Box$, $\mathcal{U}$ and $\bigcirc$ say something about the future. $\Box\phi$ means that ϕ is true in all future states. The formula $\phi\,\mathcal{U}\,\psi$ expresses the fact that at a certain point in the future ψ becomes true, and until that time ϕ is true. The next-state operator $\bigcirc\phi$ expresses the fact that in the next state ϕ is true. The epistemic operator $K_X\phi$ indicates that agent X knows that ϕ holds. The coalition operator $\ll \Gamma \gg \phi$ expresses that the set of agents Γ can ensure that ϕ holds.

We interpret formulas ϕ over a model M and a history h and write $M, h \models \phi$ if the formula is true. Unlike previous interpretations we include strategies in the model. A model M is a pair (F, S_Σ) where F is a game form with interpretation and S_Σ a strategy for all agents.

We define the neutral strategy S^0_Γ as the strategy which returns all available actions: $S^0_\Gamma(h) = A(h)$. Strategies for different coalitions can be combined using the function `combine` into a function for a larger coalition. The strategy $\texttt{combine}(\Gamma, S_\Gamma, T)$ is equal to S_Γ for agents in Γ and equal to T for other agents:

$$\texttt{combine}(\Gamma, S_\Gamma, T)(h) = \begin{cases} S_\Gamma(h) & \text{if } Ow(h) \in \Gamma \\ T(h) & \text{if } Ow(h) \notin \Gamma \end{cases}$$

A model $M = (F, S_\Sigma)$ contains information about all strategies that the agents currently use. An agent only knows his own strategy. It knows it will follow that strategy, but assumes that nothing more is known about others than that they adhere to the neutral strategy S^0_Σ. Thus, when evaluating the knowledge of an agent about the future, we use the model that is the result of the agent using its strategy, while the others use the neutral strategy. This strategy is in fact the least restrictive strategy indistinguishable to the agent. We define it using the operator $\texttt{k}(M, X)$ which is defined as

$$\texttt{k}((F, S_\Sigma), X) = (F, \texttt{combine}(X, S_\Sigma, S^0_\Sigma))$$

The function k is used in the definition of the knowledge operator.

In order to define the meaning of the strategic operator $\ll \Gamma \gg$, we use the idea of *undominated strategies*. Informally a strategy S dominates T if S is strictly better than T for reaching a certain goal. A coalition of rational agents, we assume, will not play a strategy if that strategy is dominated. To define domination, we first we need to define two other operators. The operator update is similar to k: it returns the model that represents the view of agents in coalition Γ after the adoption of strategy S_Γ. Agents in Γ adhere to S_Γ but other agents can act in any way they want:

$$\texttt{update}(M, S_\Gamma) = (F, \texttt{combine}(\Gamma, S_\Gamma, S^0_\Sigma))$$

Furthermore, we call a strategy S_Γ *successful* ($\texttt{success}(M, h, S_\Gamma, \phi)$) in history h (for Γ with respect to ϕ) if and only if $\texttt{update}(M, S_\Gamma), h \models \phi$. A strategy S_Γ *dominates* a strategy T_Γ with respect to a model M, a goal ϕ and a history h if two conditions are met : There is a history $h' \sim_\Gamma h$ such that $\texttt{success}(M, h, S_\Gamma, \phi)$ but not $\texttt{success}(M, h, T_\Gamma, \phi)$, and secondly there is no history j such that $\texttt{success}(M, j, T_\Gamma, \phi)$ but not $\texttt{success}(M, j, S_\Gamma, \phi)$. This definition of dominance makes the domination relation transitive and asymmetric. These properties ensure that any nonempty, finite set of strategies contains at least one strategy not dominated by another strategy in the set. We say that $\texttt{achievable}(M, h, \Gamma, \phi)$ if for all undominated strategies S_Γ it is the case that $\texttt{success}(M, h, S_\Gamma, \phi)$. The reason that we quantify over all undominated strategies is that we imagine that a coalition picks randomly any undominated strategy, since it has no reason to prefer one undominated strategy over the other. Therefore success is only guaranteed if all undominated strategies are successful. Using all these notions we define the interpretation of ATEL* as follows:

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

$M, h \models \neg\phi$ iff not $M, h \models \phi$

$M, h \models \phi \vee \psi$ iff $M, h \models \phi$ or $M, h \models \psi$

$M, h \models K_X \phi$ iff $\forall h' : h' \sim_X h \implies \texttt{k}(M, X), h' \models \phi$

$M, h \models \phi\, \mathcal{U}\, \psi$ iff $\forall j \in Z(H(S_\Sigma, h)) \exists i \forall k : |h| \leq k < i \implies$
$M, j_0 \dots j_k \models \phi) \wedge M, j_0 \dots j_i \models \psi$

$M, h \models \Box \phi$ iff $\forall h' \in H(S_\Sigma, h) : M, h' \models \phi$

$M, h \models \bigcirc \phi$ iff $\forall a \in S_\Sigma(h) : M, ha \models \phi$

$M, h \models \ll \Gamma \gg \phi$ iff $\texttt{achievable}(M, h, \Gamma, \phi)$

The set $H(S, h)$ contains all histories of H that start with h and are consistent (after h) with the strategy S. It can be defined recursively. $H(S, h)$ is the smallest set H' such that $h \in H'$ and $\forall h' \in H', \forall a \in S(h') : h'a \in H'$.

5 Examples

In section 3 we have introduced four game forms. In this section we use these game forms to show the interpretation of example formulas. For all examples the model M_i is defined as (G_i, S^0_Σ). The first examples deal with temporal properties.

$M_1, \epsilon \models t_0 \wedge \bigcirc t_1 \wedge \bigcirc \bigcirc t_2$	Initially t_0 holds, then t_1 and then t_2
$M_1, \epsilon \models \Box (t_2 \rightarrow \neg t_1)$	If t_2 holds, then not t_1
$M_1, \epsilon \models \top\, \mathcal{U}\, t_2$	Eventually t_2 holds

We have argued that in the game form G_1 after action 1, the agent B can achieve p, but not q. It can also achieve $p \wedge q$ but it does not know that it can achieve this fact. The translations of these facts are given here. In model G_2 similar properties hold and these are also given.

$M_1, 1 \models \ll B \gg \bigcirc p$	B can make p true in the next state
$M_1, 1 \models \neg \ll B \gg \bigcirc q$	B cannot make p true in the next state
$M_1, 1 \models \ll B \gg \bigcirc (p \wedge q)$	B can make p and q true in the next state
$M_1, 1 \models \neg K_B \ll B \gg \bigcirc (p \wedge q)$	B doesn't know it can make p and q true
$M_2, 1 \models \neg \ll B \gg \bigcirc p$	B cannot make p true in the next state
$M_2, 2 \models \ll B \gg \bigcirc p$	B can make p true in the next state
$M_2, 2 \models \neg K_B \ll B \gg \bigcirc p$	B does not know it can make p true in the next state

In game form G_3, agent A does not remember the choices it has made in the past. Agent B does not always know its previous observations. This is expressed in the next statements. The next model, G_4, shows that agents cannot in general commit themselves to act against their future preferences.

$M_3, \epsilon \models K_A \ll A \gg \bigcirc p$	A knows it can make p true in the next state
$M_3, \epsilon \models \neg \ll A \gg \bigcirc K_A p$	A cannot know p in the next state
$M_3, 1 \models K_B \bigcirc q$	B knows q is true in the next state
$M_3, 1 \models \neg \bigcirc K_B q$	In the next state B does not know that q is true
$M_4, \epsilon \models \ll A \gg \bigcirc \bigcirc p$	A can make p true in the next next state
$M_4, \epsilon \models \neg \ll A \gg \neg \ll A \gg \bigcirc p$	A cannot ensure it cannot make p true
$M_4, \epsilon \models \neg \ll A \gg \bigcirc \neg \ll A \gg \bigcirc p$	A cannot ensure that next it cannot make p true

Turning back to game form G_1, we give an example of how an agent can have strategic knowledge. Suppose that agent A has committed to the strategy of playing action 2. It then knows that agent B will not be able to achieve the satisfaction of $p \wedge q$ anymore. Let $S_{A,B}$ be the strategy profile consisting of the strategy of playing action 2 for agent A and the neutral strategy for agent B. We can then represent the knowledge described above as:

$(G_1, S_{A,B}), \epsilon \models K_A \neg \ll B \gg \bigcirc \bigcirc p \wedge q$	A knows that B won't be able to satisfy $p \wedge q$

6 Perfect Recall and Perfect Memory

Agents have *perfect recall* if they never forget their previous observations and the actions they have chosen [12]. Similarly the have perfect memory [10, 2] if they do not forget their observations. Traditionally game theory has focused on perfect recall agents, but artificial agents in multiagent systems often do not have these properties. For the complexity of solving games, or model checking temporal formulas, it is relevant whether the systems have perfect recall of perfect memory [10, 7]. Therefore we present here two theorems that characterize whether a game form interpretation has perfect recall and perfect memory. Especially we want to illustrate the difference between perfect recall and perfect memory, since this difference does not appear in temporal logic without strategies, but does exist in games and strategic logics. Making use of the strategy profiles that we have included in the model is necessary for the proof of the perfect recall property. The authors are not sure whether a different theorem regarding perfect recall could hold for previous interpretations of ATEL*.

It can be argued that *perfect recall* is not a property of a game, but a property of a player in a game. However we think of the equivalence relations $\sim_X$ in a game form $(\Sigma, H, Ow, [\sim_a]_{a \in \Sigma})$ as representing the knowledge of the agents. We thus assumed that the capabilities of the agents have been included in the equivalence relations. Thus we view perfect recall as a property that a game form can or cannot have. We define perfect recall in terms of observations;

an agent has perfect recall if it remembers all its observations, including the actions it has chosen. Let the $O_X(h)$ be the function returning the ordered list of all observations and actions chosen by agent X in history h. Then an agent X has perfect recall if and only if $h \sim_X j \;\leftrightarrow\; O_X(h) = O_X(j)$. The function O_X can be defined recursively. The observation function of ha contains all observations of h, plus maybe the action a (if $Ow(h) = X$) and the observation made in ha. Using such recursive definition, it is not hard to show that the property of perfect recall is equivalent to the next two properties.

$$ha \sim_X jb \rightarrow h \sim_X j$$
$$Ow(h) = X \wedge ha \sim_X jb \rightarrow a = b$$

This characterisation of perfect recall is the one we use in the next theorem.

Theorem 1 *Let $F = (\Sigma, H, Ow, [\sim_X]_{X\in\Sigma})$ and $X \in \Sigma$. X has perfect recall in F if and only if for every P, π, every ϕ, every strategy S_Σ and every h:*

$$((F, P, \pi), S_\Sigma), h \models K_X \bigcirc \phi \rightarrow \bigcirc K_X \phi$$

Suppose that X has perfect recall in F and let $P, \pi, \phi, S_\Sigma, h$ be given. Let $G = (F, P, \pi)$ and suppose that $(G, S_\Sigma), h \models K_X \bigcirc\phi$. Define $M' = (G, S'_\Sigma) = k((G, S_\Sigma), X)$. Let $a \in S_\Sigma(h)$ and $jb \sim_X ha$. From the perfect recall properties we know that $h \sim_X j$. From the definition of k one can see that for any h' it is the case that $S_\Sigma(h') \subseteq S'_\Sigma(h')$. Using $(G, S_\Sigma), h \models K_X \bigcirc\phi$ we can conclude that $M', j \models \bigcirc\phi$. If $Ow(h) = X$ then $Ow(j) = X$ and we know that $b = a$ and therefore $b \in S_\Sigma(h) \subseteq S'_\Sigma$. If $Ow(h) \neq X$ then $S'_\Sigma(j) = A(j)$ and thus $b \in S'_\Sigma(j)$. From $M', j \models \bigcirc\phi$ and $b \in S'_\Sigma(j)$ we can conclude that $M', jb \models \phi$. Since we have shown this for an arbitrary $jb \sim_X ha$ we conclude that $(G, S_\Sigma), h \models \bigcirc K_X\phi$.

For the second half, assume that for every P, π, every ϕ, every strategy S_Σ and every h:

$$((F, P, \pi), S_\Sigma), h \models K_X \bigcirc \phi \rightarrow \bigcirc K_X \phi$$

Let $G = (F, P, \pi)$ and let $ha \in H$. Take jb such that $ha \sim_X jb$. Let $P = \{p\}$ and define π such that $\pi(j'b') = \{p\}$ iff $h \sim_X j'$ and $b' \in A(j')$. Let $S = S^0_\Sigma$. This interpretation ensures that $(G, S), h \models K_X \bigcirc p$. We can derive from the assumptions that $(G, S), h \models \bigcirc K_X p$. Thus for every $a \in A(h)$ it is the case that $(G, S), ha \models K_X p$. Since $ha \sim_X jb$, we conclude that $(G, S), jb \models p$. By definition of $\pi(p)$ this gives us $h \sim_X j$.

Let $G = (F, P, \pi)$ and let $ha \in H$ be such that $Ow(h) = X$. take $jb \in H$ such that $ha \sim_X jb$. From the previous part we can already conclude that $h \sim_X j$. Let $P = \{p\}$ and let $S = S_\Sigma$ be a strategy such that $S(h) = \{a\}$. define π such that $\pi(j'b') = \{p\}$ iff $b' = a$ and $j' \sim_X h$. These definitions ensure that $(G, S), h \models K_X \bigcirc p$. We can derive $(G, S), h \models \bigcirc K_X p$. From the definition of next we know that $(G, S), ha \models K_X p$ and thus that $(G, S), jb \models p$. By definition of p we conclude that $a = b$. This concludes the proof.

An agent with perfect memory remembers all its previous observations. Let $M_X(h)$ be the function returning the ordered list of all observations by agent X in history h (excluding the actions it has chosen). We define that an agent X has perfect memory if and only if $h \sim_X j \;\leftrightarrow\; M_X(h) = M_X(j)$. Again, one can define the observation function M recursively. Using such recursive definition, it is not hard to show that the property of perfect memory is equivalent to $ha \sim_X jb \rightarrow h \sim_X j$.

Theorem 2 *Let $F = (\Sigma, H, Ow, [\sim_X]_{X\in\Sigma})$ be a game form and $X \in \Sigma$. X has perfect memory in F if and only if for every P, π, every ϕ, and every h:*

$$((F, P, \pi), S^0_\Sigma), h \models K_X \bigcirc \phi \rightarrow \bigcirc K_X \phi$$

In this theorem, instead of just any strategy we use the neutral strategy S^0_Σ. An important property of this strategy is that $k((F, S^0_\Sigma), X) = (F, S^0_\Sigma)$.

For the first half of the proof, let X have perfect memory in F and assume that P, π, ϕ and h are given. Let $M = ((F, P, \pi), S^0_\Sigma)$ and assume that $M, h \models K_X \bigcirc\phi$. Let $a \in A(h)$ and jb any history with $jb \sim_X ha$. From the perfect memory property we know that $j \sim_X h$. We can derive that $k(M, X), j \models \bigcirc\phi$. Since M contains the neutral strategy, $k(M, X) = M$ and thus $M, j \models \bigcirc\phi$. This means that $M, jb \models \phi$. Since we have shown this for an arbitrary jb with with $jb \sim_X ha$ we can conclude that $M, h \models \bigcirc K_X\phi$.

For the second part, assume that for every P, π, ϕ and h: $((F, P, \pi), S^0_\Sigma), h \models K_X \bigcirc \phi \rightarrow \bigcirc K_X\phi$. Let $ha \sim_X jb$ be given. Define $P = \{p\}$ and π such that $\pi(h'a') = \{p\}$ iff $h' \sim_X h$ and $a' \in A(h')$. This definition ensures that $M, h \models K_X \bigcirc p$. We can conclude that $M, h \models \bigcirc K_X p$. From this formula one can derive that $M, ha \models K_X p$ and thus that $M, jb \models p$. this means that $\pi(jb) = \{p\}$ and thus that $j \sim_X h$. Therefore X has perfect memory in F.

7 Conclusions and Further research

We have presented the logic ATEL* and given a new interpretation to the operator associated with strategic ability. Its advantages over previously suggested definitions are that the meaning corresponds, in the authors' opinions, best with the natural meaning of having a strategy in an extensive game. A characteristic feature is that under this interpretation agents may have abilities they do not know they have. We have shown that one can characterize the properties *perfect recall* and *perfect memory* with an ATEL* formula.

Future work could focus on extending the language, for instance with common knowledge. An interesting question is what the complexity is of evaluating this logic over finite state systems. Furthermore it would be interesting to see whether similar semantics can be given for agents that do know each others' strategy immediately, or for coalitions that are not able to coordinate their actions. Finally it would be interesting to apply this logic to example problems such as the Russian Cards problem [18] and the Dining Cryptographers problem [17].

References

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 100–109, Florida, October 1997.

[2] G. Bonanno. Memory and perfect recall in extensive games. *Games and Economic Behavior*, 47:237–256, 2004.

[3] G. Bonanno. Memory implies von neumann-morgenstern games. *Knowledge Rationality and Action*, to appear:1–20, 2004.

[4] E.M. Clarke and E.A. Emerson. Desing and synthesis of synchronisation skeletons using branching time temporal logic. *Lecture Notes in Computer Science*, 131:52–71, 1981.

[5] S. Druiven. Knowledge development in games of imperfect information, 2002. University Maastricht Master Thesis.

[6] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about knowledge.* The MIT Press: Cambridge, MA, 1995.

[7] J. Halpern, R van der Meyden, and M. Vardi. Complete axiomatizations for reasoning about knowledge and time. *SIAM Journal on Computing*, 33:674–703, 2004.

[8] W. Jamroga and W. van der Hoek. Some remarks on alternating-time temporal epistemic logic. submitted, 2003.

[9] G. Jonker. Feasible strategies in alternating-time temporal epistemic logic, 2003. Universiteit Utrecht Master Thesis.

[10] D. Koller and N. Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior 4*, 4:528–552, October 1992.

[11] H Kuhn. Extensive games and the problem of information. *Contributions to the Theory of Games*, II:193–216, 1953.

[12] M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* The MIT Press: Cambridge, MA, 1994.

[13] Pierre-Yves Schobbens. Alternating-time logic with imperfect recall. In Wiebe van der Hoek, Alessio Lomuscio, Erik de Vink, and Mike Wooldrige, editors, *Electronic Notes in Theoretical Computer Science*, volume 85. Elsevier, 2004.

[14] J. van Benthem. Games in dynamic-epistemic logic. *Bulletin of Economic Research*, 53(4):219–248, 2001.

[15] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, pages 1167–1174, Bologna, Italy, 2002.

[16] W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(4):125–157, 2003.

[17] R. van der Meyden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. submitted, 2002.

[18] H. P. van Ditmarsch. The russian cards problem. *Studia Logica*, 75(4):31–62, 2003.

A dialogue-game for agent resolving conflicts by verbal means

Maxime Morge
École Nationale Supérieure des Mines de Saint Etienne
158 cours Fauriel, F-42023 Saint Etienne cedex 2
morge@emse.fr

14th May 2004

Abstract

We present in this paper a formal framework for argumentation-based dialogues between agents. These latter manage the dialogues with the help of three components: an argumentative component to generate arguments, a social component to interprete arguments, and a conventional component to manage the sequence of coherent moves. We formalize the notion of dialogue-game to address the gap between individual moves and the extended sequence of coherent moves that arise between agents. The moves are not associated with an intention, however the dialogues have a goal.

1 Introduction

During the last decade, many Agent Communication Languages (ACL) were designed for the interaction in Multi-Agent Systems (MAS). These ACL do not succeed to address formal inter-agent dialogues.

Most of the existing ACL are based on speech acts theory [8]. For example, FIPA-ACL [7] or KQML [4] define communicative acts by pre/post conditions bearing on the mental attitudes of agents. Many shortcomings come from this approach. We have identified here three main shortcomings. (1) The illocutionary force, *i.e.* the intention of the speaker, is considered as the main characteristic of the speech act. This is the reason why the agents must be understood in terms of mental concepts. (2) The institutional value of the speech acts is implicit. Then, the communication has no social semantics to be judged in a public perspective [9]. (3) This approach considers a communicative act as an epiphenomenon. Therefore, the semantics of communicative acts is so rich that it is far too complex to determine an answer by just inferring mental states[3].

By contrast, our work is inspired by formal dialectic [10]. We present in this paper an extension of the framework for argumentation-based dialogues between agents proposed by Parsons *et al.* [5, 1, 2].. The agents manage the dialogues with the help of three components, each of them addressing one of the previous issues. We formalize the notion of dialogue-game to address the gap between individual moves and the extended sequence of coherent moves that arise between agents. The moves are not associated with an intention, however the dialogues have a goal.

Paper overview. Section 2 presents the argumentation-based reasoning as defined in [1]. In accordance with this background, we modify the formal framework for dialogues proposed by Parsons *et al.* [5] in section 3. The agents share a knowledge language and a communication language (cf section 3.1) in order to reason together (cf section 3.2). We formalize the notion of dialogue in section 4. Then, the proprieties of the dialogues can be studied (cf section 4.2).

2 Argumentation system

An argumentation system as defined in [1] is a set of arguments with a conflicting relation and a preference relation from which could be extracted a set of acceptable arguments.

The knowledges are factual judgments gather in a knowledge base, written Σ. This base contains formulae of a propositional language, written $\mathcal{L}$. $\vdash$ stands for classical deduction and $\equiv$ for logical equivalence.

In order to evaluate preferences between the judgments, the knowledge base has a preference order captured by a preordering relation. This preference relation $\ll$ denotes a binary relation that is reflexive and transitive. This preference ordering makes it possible to deduce a stratification of the base Σ into non-overlapping sets $\Sigma^n \ll ... \ll \Sigma^1$ such that facts in Σ^i are all equally preferred and are more preferred than those in Σ^j with $i \leq j$. The number of the highest numbered layer that has a member in a non-empty set H is written level(H).

An argument is composed of a formula, called conclusion, and a set of formulae, called support, from which the conclusion can be inferred.

Definition 1. *An **argument** is a pair $P = (H, h)$ where h is a formula of $\mathcal{L}$ and H a subset of Σ such that:*

1. *$H \subseteq \Sigma$ is consistent;*

2. $H \vdash h$;

3. H is minimal, so no subset of H satisfying both 1 and 2 exists.

H is called the ***support*** *of P, written $H = support(P)$ and h is the* ***conclusion*** *of P, written $h = conclusion(P)$.*

An argument P is **trivial** iff support(P) = {conclusion(P)}. Let $\mathcal{A}(\Sigma)$ denote the set of arguments built on Σ.

Since Σ can be inconsistent, arguments may conflict. The next definition precises the notion of undercutting to capture these conflicts. An argument is undercut iff there is one of the formulae of its support which is denied by another argument.

Definition 2. *Let P_1 and P_2 two arguments of $\mathcal{A}(\Sigma)$. P_1* ***undercuts*** *P_2 iff $\exists h \in support(P_2)$ such that $h \equiv \neg conclusion(P_1)$.*

Moreover, the preferences between arguments can be evaluated.

Definition 3. *Let P_1 and P_2 two arguments of $\mathcal{A}(\Sigma)$. P_1* ***is preferred to*** *P_2 (written P_1 pref P_2) iff: $level(support(P_2)) > level(support(P_1))$.*

These two orders make it possible to distinguish different types of relations between arguments.

Definition 4. *Let P_1, P_2 be two arguments of $\mathcal{A}(\Sigma)$ and $S \subseteq \mathcal{A}(\Sigma)$ be a set of arguments.*

- *P_1* ***defends itself against*** *P_2 iff P_2 undercut $P_1 \wedge P_1$ pref P_2. We denote P_1 defend_itself P_2;*
- *S* ***defends*** *P_1 iff $\forall P_2 \in \mathcal{A}(\Sigma)$ s.a P_2 undercuts P_1 and P_1 does not defend itself again P_2 then $\exists P_3 \in S$ s.a. P_3 undercut P_2 and P_2 does not defend itself against P_3. We denote S defend P_1;*

The notion of acceptability links the preference ordering and the undercutting relation.

Definition 5. *Let $AS = \langle \mathcal{A}(\Sigma),\ undercut\ ,\ pref\ \rangle$ be an argumentation system. The* ***set of acceptable arguments****, written $\mathcal{S}$ is the least fixpoint of a function $\mathcal{F}$: $S \in \mathcal{A}(\Sigma)$ and $\mathcal{F}(S) = \{P \in \mathcal{A}(\Sigma);\ S\ defend\ P\}$.*

The following section formalizes the framework for inter-agent dialogues based upon this argumentation-based reasoning.

3 Dialogical system

A dialogical multi-agent system consists of a set of agents. They share a knowledge language and a communication language. An agent is associated with an argumentation system in order to deliberate. The arguments of its peers must be taken into account to be interpreted and to generate counter-arguments.

3.1 Common languages

Since the judgments of agents may be different, each agent has its own belief base, Σ_i^B and its own preordering, $\ll_i$. These belief bases contain formulas of a **common knowledge language**, written $\mathcal{L}_\mho$. Consequently, the agents share the same inference rule, denoted $\vdash_\mho$.

Dialogue agents utter messages each its turns. Each message has an identifier M_k. The syntax of messages is in conformance with a **communication language**, $\mathcal{CL}_\mho$ defined in a similar way of FIPA-ACL or KQML. A message is also called dialogical move by reference to the game theory.

Definition 6. *A* ***dialogical move*** $M_k \in \mathcal{CL}_\mho$ *is defined by a 5-tuple,* $M_k = \langle S_k, H_k, R_k, DG_k, L_k \rangle$ *where:*

- M_k *is the identifier of the* k^{th} *move in the dialogue between the speaker and the hearer. It can be referenced later in the dialogue;*

- $S_k = speaker(M_k)$ *is the agent that utters the move;*

- $H_k = hearer(M_k)$ *is the addressee;*

- $R_k = reply(M_k)$ *is the identifier of the move to which* M_k *responds* $(R_1 = \emptyset)$*;*

- $DG_k = dialogue\text{-}game(M_k)$ *is the dialogue game used to generate the answer ;*

- $L_k = locution(M_k)$ *is the locution composed of a performative and a propositional content. The verb is one of the following: question, assert, unknow, accept, challenge, withdraw.*

A move uttered by a speaker is addressed to a hearer, i.e. one agent in the audience that receives and interprets the move in order to respond. The meaning of locutions is defined by the three components used to manage the dialogue. (cf sections 3.2.1,3.2.2,4.1). We propose a dialogue-game in section 4.2.

Thanks to these two languages, we present here the two components used by the agents to reason together. They take into account the arguments of their peer, interpret them and generate counter-arguments: they argue together.

3.2 Co-argumentation

During dialogue, agents take a stand for propositions. The commitment store, written CS_j^i, consists of the set of formulae perceived by the agent ag_i to which the agent ag_j commits [10]. An agent is in conformance with the following definition:

Definition 7. *A* ***dialogical agent*** $ag_i \in AG_{\mho}$ *is a triple* $ag_i = \langle \Sigma_i^B, \cup_{j \neq i} CS_j^i, \ll_i \rangle$ *such as:*

- Σ_i^B *is a belief base;*
- $\cup_{j \neq i} CS_j^i$ *is the set of commitment stores built by the agent* ag_i*;*
- $\ll_i$ *is the preordering relation on* Σ_i^B.

The formulae in the commitment stores are taken into account to generate arguments.

3.2.1 Argumentation component

The argumentation component precises the rational conditions of utterances and the relative tactics.

Since agents reason together, their arguments are built on their own beliefs and on the commitments of the agents it is speaking to. Then, each agent is associated with an extended argumentation system:

$$\mathrm{AS}_i^* = \langle \Sigma_i, \text{ undercut }, \text{ pref }_i^* \rangle$$

$$\text{where } \Sigma_i = \Sigma_i^B \cup [\bigcup_{i \neq j} \mathrm{CS}_j^i] \text{ the extended belief base}$$

$$\text{and } \text{ pref }_i^*\text{the extension of the preference relation on}\mathcal{A}(\Sigma_i).$$

We denote $\mathcal{S}_i^*$ the corresponding set of acceptable arguments. pref $_i^*$ will be explained in section 3.2.2.

The rational condition of a locution depends on its performative and its propositional content. An agent can assert a formula iff it has an argument for it.

Definition 8. *The predicate* $can\_assert(ag_i, H)$*, called* ***rational condition for the assertion*** *of a propositional content* H *by the agent* ag_i*, is defined s.a.:*

$$\forall h \in H\ \exists P \in \mathcal{A}(\Sigma_i)\ conclusion(P) = h.$$

Contrary to [5], the rational condition for the assertion and the rational condition for the acceptance of the same propositional content by the same agent distinguish themselves.

Definition 9. *The predicate* $can\_accept(ag_i, H)$*, called* ***rational condition for the acceptance*** *of a propositional content* H *by the agent* ag_i*, is defined s.a.:*

$$\forall h \in H\ \exists P \in \mathcal{A}(\Sigma_i)\ conclusion(P) = h \text{ with } (support(P) \neq \{h\} \wedge \\ support(P) \not\subseteq \cup_{j \neq i}\ CS^i_j).$$

Agents can assert propositions whatever they are supported by a trivial argument or not. By contrast, agents do not accept all the propositions he hears in spite of they are all supported by a trivial argument.

The other locutions (question(h), challenge(h), unknow(h), withdraw(h)) have no particular rational conditions.

Then, the rational conditions of utterances are not necessary mutually excluded. These nondeterministic situations make it possible for agents to choice. That is the reason why we define as Parsons *et al.* [5] a set of argumentative tactics.

Definition 10. *The predicate* $want\_assert(ag_i, H)$ *called* ***argumentative tactic for the assertion*** *of a propositional content* H *by the agent* ag_i*, depends on the* ***assertive attitude*** *of the agent* ag_i*:*

- *if* ag_i *is thoughtful then* $want\_assert(ag_i, H) \Leftrightarrow \forall h \in H\ \exists P \in \mathcal{S}^*_i$ $conclusion(P) = h$*;*
- *if* ag_i *is confident then* $want\_assert(ag_i, H) \Leftrightarrow\ can\_assert(ag_i, H)$*.*

The argumentative tactic for the acceptance is defined in a similar way.

Definition 11. *The predicate* $want\_accept(ag_i, H)$ *called* ***argumentative tactic for the acceptance*** *of a propositional content* H *by the agent* ag_i*, depends on the* ***acceptance attitude*** *of the agent* ag_i*:*

- *if* ag_i *is skeptical then* $want\_accept(ag_i, H) \Leftrightarrow \forall h \in H\ \exists P \in \mathcal{S}^*_i$ $conclusion(P) = h$ *with* $(support(P) \neq \{h\} \wedge$ $support(P) \not\subseteq \cup_{j \neq i}\ CS^i_j)$*;*
- *if* ag_i *is credulous then* $want\_accept(ag_i, H) \Leftrightarrow\ can\_accept(ag_i, H)$*.*

However the rational conditions of utterances are shared by all the agents, the argumentative tactics are individual choices. The social component makes it possible to interpret arguments.

3.2.2 Social component

This component provides the social semantics for the locution [9]. The move's meaning must not only have a private perspective to be expressed, but also a public perspective in order to be interpreted.

In a similar way with [5], we associate a set of commitment stores to each agent, which hold the commitments perceived during the dialogue. Commitments stores are updated according to the following rules:

Definition 12. ***Updating rules.***
Let $M_{k+1} \in \mathcal{CL}_{\mho}$, s.a. speaker$(M_{k+1}) = ag_j$ and $ag_i \in AG_{\mho}$ is in the audience.

- *if L_{k+1} = question(h) or L_{k+1} = unknow(h) or L_{k+1} = challenge(h) or L_{k+1} = withdraw(h) with h a formula of $\mathcal{L}_{\mho}$ then $CS^i_j(M_{k+1}) = CS^i_j(M_k)$;*
- *if L_{k+1} = assert(H) or L_{k+1} = accept(H) with H a set of formulae of $\mathcal{L}_{\mho}$ then $CS^i_j(M_{k+1}) = CS^i_j(M_k) \cup H$.*

The performative withdraw (not present in [5]) has no effect on the commitment stores but closes the dialogue (cf section 4.1.2). The arguments which are received must be valuated.

Since the agents are more or less authoritative, the commitments are considered in accordance with the estimated reliability of the agents from whom the information is obtained. For this purpose, each agent ag_i ranks the competence of the other agents with a strict total order on $\text{AG}_{\mho}$, denoted $\prec_i$. Contrary to[2], this preference relation defines a subjective power relation between the agents.

The preference between formulae are evaluated in accordance with the following cooperative principle of arguments adoption: *"ag_i will prefer ag_j's statements iff ag_j is regarded as more competent".* This principle defines $\ll^*_i$ as a preordering relation on Σ_i and so pref^*_i on $\mathcal{A}(\Sigma_i)$. Then, the preference between arguments coming from different sources, the belief base ($\mathcal{A}(\Sigma^B_i)$) or the different commitment stores ($\mathcal{A}(\text{CS}^i_j)$,($\mathcal{A}(\text{CS}^i_k)$,...), can always be evaluated.

Thanks to the formal framework described here, the agents argue together. They take into account the arguments of their peer, interpret them and generate counter-arguments. However, the agents do not jointly reason to reach common goals. We formalize the notion of dialogue-game to address this gap.

4 Dialogue-game

Walton and Krabbe [10] have proposed a categorization of dialogues. This classification is especially based upon the initial informational status of the participants and the goals they share, also called the goals of the dialogue.

A dialogue-game describes the possible sequence of coherent moves to reach a goal. The conventional component manages the sequence of moves.

4.1 Conventional component

In order to manage the sequence of moves, this component uses dialogical rules, sequence rules, and related tactics.

4.1.1 Dialogical rules

The following basic rules regulate the dialogues whatever the dialogue-game is. The first rule initializes the dialogue with a question on a topic. The second one avoids redundancy of information in assertions [6]. Therefore, no loop will happen in dialogues. The third rule takes care of turn-taking. The fourth warrants to keep the same dialogue-game during the dialogue.

Definition 13. *The moves $M_1, M_{k+1} \in \mathcal{CL}_{\mho}$ (with $k \geq 0$) are in conformance with the following **dialogical rules**:*

1. *initialization*
   $locution(M_1) = question(p)$. *p is called the topic of the dialogue ;*

2. *non-redundancy*
   $locution(M_{k+1}) = assert(H) \rightarrow \forall p \in H \; \forall l \leq k \; locution(M_l) = assert(H')$, $p \notin H'$;

3. *turn-taking*
   $hearer(M_{k+1}) = speaker(M_k) \wedge speaker(M_{k+1}) = hearer(M_k)$;

4. *dialogue-game keeping*
   $dialogue\text{-}game(M_{k+1}) = dialogue\text{-}game(M_k)$.

We immediately deduce that a dialogue takes place between the speaker and the hearer of the first move. A participant play one of the following **conventional roles**: *initiator* (init), i.e. the agent beginning the dialogue or *partner* (part), i.e. the agent it is speaking to. The agents that do not participate directly are the *bystanders* of the dialogue.

All the agents use these four dialogical rules whatever the dialogue-game is. However these rules are canonical, sequence rules specify the answers allowed or not in a given situation. The following section enumerates a set of sequence rules. The section 4.2 presents the dialogue-game using these rules.

4.1.2 Sequence rules

The sequence rules specify the answers that are (or not) allowed in a given situation by constraining the locution and the reply field. The argumentative tactics of the allowed moves are not necessary mutually excluded. These nondeterministic situations renders a choice possible. That is the reason why we define as Parsons *et al.* [5] a set of conventional tactics and attitudes.

Respond to a question. The rule of "Question/Answer" allows the hearer of a question (question(h)) to respond: either with a confirmation (assert(h)), either with an invalidation (assert($\neg h$)), or with a plea of ignorance (unknow(h)).

In replying to a question, an agent that can either give its opinion, a confirmation or an invalidation, or plead ignorance is *cooperative* if it responds to the request. Otherwise, it is *egoist.* An agent that can either respond with a confirmation or with an invalidation is: *positive* if it confirms whenever possible ; *negative* if it invalidates whenever possible.

Respond to an assertion. the rule of "Assertion/Refutation" allows the hearer of an assertion (assert(H)) to respond: either with a hearty welcome (accept(H)), either with a refutation (assert($\neg h$), with $h \in H$), or with a challenge (challenge(h'), with $h' \in H$).

In replying to an assertion, an agent that can either give its opinion, an hearty welcome or a refutation, or challenge is: *argumentative* if it challenges; *open-minded* if it gives its opinion. an agent that can either respond with a hearty welcome or with a refutation is: *optimistic* if it accepts whenever possible ; *pessimistic* if it refutes whenever possible.

Respond to a challenge. The rule of "Challenge/Argument" allows the hearer ag$_i$ of a challenge (challenge(h)) to respond: either with an argument (assert(H), with $H =$ support(P), $P \in \mathcal{A}(\Sigma_i)$ s.a. $h =$ conclusion(P)), or with a withdrawal ($withdraw(h')$) making reference to its first assertion.

In replying to a challenge, an agent *patient* respond with an argument whenever possible. Otherwise, it is *impatient.*

Then, an algorithm selects the privileged responding move for each sequence rule. These algorithms are defined such as there is a single effective responding move which is in conformance with the corresponding sequence rule.

Closing the dialogue. The moves with performatives: unknow, accept or withdraw close the dialogue.

A dialogue-game of persuasion consists of the combinaison of these sequence rules.

4.2 Dialogue-game of persuasion

The topic of persuasion dialogues is only discursive. The participants try to reach an agreement, not a decision to act (or not to act). We aim at proving the termination of persuasion dialogues whatever the initial situation is. By contrast, the goals of a persuasion dialogue are reached if some particular initial conditions are verified.

The figure 4.2 shows a persuasion dialogue-game in the extensive form game representation where nodes are game situations and edges are associated with moves. For example, 2.3^{init} denotes a game situation where the exponent indicates that the initiator is the speaker of the next move. $2.1^{\square}$,$3.2^{\square}$, and $4.2^{\square}$ denote game-over situations.

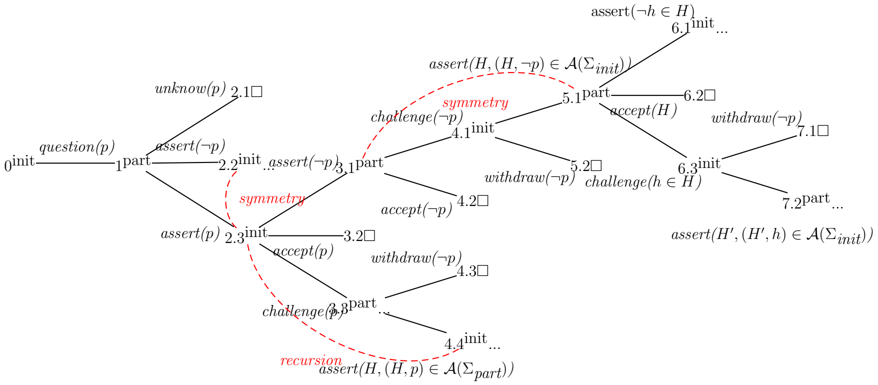


Figure 1: Persuasion dialogue in an extensive form game representation

4.2.1 Termination

The termination of persuasion dialogues can be warranted, whatever the (argumentative and conventional) attitudes and the initial informational status of the participants are.

Theorem 1. *A persuasion dialogue which takes place between two agents of $AG_{\mho}$ and with a topic in $\mathcal{L}_{\mho}$ always terminates.*

Proof. Thanks to the definition of the algorithms selecting the privileged responding move, the hearer can always respond whatever the sequence rule is. The game situations 2.2^{init} and 2.3^{init} are equivalent by symmetry on the propositional content of the previous assertion. The game situation 2.3^{part} is equivalent to the game situation 4.4^{init} by symmetry on the propositional content of the previous assertion even if the conventional roles are inverted. The game situations 3.1^{part} and 5.1^{part} are equivalent by symmetry on the propositional content of the previous assertion. Moreover, the second dialogical rule avoids redundancy of information in assertions. Then, no loop will happen in dialogues. Σ_{part} and Σ_{init} are finite because the belief bases of participants are finite. Consequently, the recursion is finite and the dialogue closes. □

4.2.2 Success

The goal of a persuasion dialogue is to reveal the position of the participants, to spread the participants'arguments and to verbally resolve the conflict. Contrary to the termination of a persuasion dialogue, the goals are reached if some particular initial conditions are verified.

Theorem 2. *Let a persuasion dialogue take place between two agents of $AG_{\mho}$ such as the topic p is a formula of $\mathcal{L}_{\mho}$. If the initial informational status of participants are such as they have conflicting thesis, even if it inverts:*

- *the initiator is convinced of $\neg p$:*
  $[\exists P'_{init} \in \mathcal{S}^*_{init}\ conclusion(P'_{init}) = \neg p] \wedge$
  $[\nexists P_{init} \in \Sigma_{init}\ conclusion(P_{init}) = p]$;
- *the partner is convinced of p:*
  $\left[\exists P_{part} \in \mathcal{S}^*_{part}\ conclusion(P_{part}) = p\right] \wedge$
  $\left[\nexists P'_{part} \in \Sigma_{part}\ conclusion(P'_{part}) = \neg p\right]$.

*Let a **witness** agent (denoted bystander) be a bystander of the dialogue with an initially empty belief base.*
If the partner is cooperative and the initiator is open-minded then the three goals will be reached at the end of the dialogue:

1. ***revealing position:***
   $p \in CS^{init}_{part} \wedge \neg p \in CS^{part}_{init}$;
2. ***spread of argument:***
   $\exists P \in \mathcal{A}(CS^{init}_{part}) \cap \mathcal{A}(CS^{bystander}_{part})\ conclusion(P) = p$
   $\wedge\ \exists P' \in \mathcal{A}(CS^{part}_{init}) \cap \mathcal{A}(CS^{bystander}_{init})\ conclusion(P') = \neg p$;
3. ***resolving the conflict by verbal means:*** *the witness agent is prone to one of the participants' thesis (even if inverts p with $\neg p$):*
   $\exists P' \in \mathcal{S}^*_{bystander}\ conclusion(P') = \neg p \wedge$
   $\nexists P \in \mathcal{S}^*_{bystander}\ conclusion(P) = p$.

Proof. The partner is convinced of p and it is cooperative. Therefore, the game situation 2.3^{init} is reached. The commitment store is updated then the partner has revealed its position. The initiator is convinced of $\neg p$ and it is open-minded. Then, the game situation 3.1^{part} is reached. The commitment store is updated then the initiator has revealed its position. Whatever the participants'arguments are, each of them has spread a trivial argument for its thesis.
In game-over situation $4.2^{\Box}$ and $5.2^{\Box}$, the witness agent has a trivial argument for p and a trivial argument for $\neg p$. They undercut each other. Because the sources of the arguments are different, only one is acceptable. Then, the witness is prone to one of the participant thesis. In the game-over situation $6.2^{\Box}$ and $7.1^{\Box}$, $P' = (H', \neg p)$ is the only acceptable argument of the witness. Therefore, this agent is prone to $\neg p$. The other game situations are equivalent by recursion on the content of the previous move. Consequently, the witness is prone to one of the participant's thesis however the dialogue is closed. □

However we define the resolution of the conflict by verbal means in a different way than Walton and Krabbe [10], we can demonstrate that these two definitions are equivalent.

5 Conclusions

We have presented in this paper a formal framework for the argumentation-based dialogues between agents. These latter manage the dialogues with the help of three components: the argumentation component specify the rational condition of utterances and the relative tactics ; the social component provides the meaning of the locutions to be interpreted ; and the conventional component manages the sequence of moves. We have formalized the notion of dialogue-game to address the gap between individual moves and the extended sequence of coherent moves that arise between agents. However the moves are not associated with an intention, the dialogues have a goal. The termination of the dialogue is demonstrated, whatever the initial status and attitudes of the participants are. By contrast, the goals of a dialogue are reached if some particular initial conditions are verified.

We are currently implementing this dialogical multi-agent system with MAST[1], which is an environment for the development of multi-agent applications. It provides some tools to design agents in a component-based approach, in particular a component for inter-agent communication and an interaction model of the agent-level components.

[1] http://www.emse.fr/∼vercoute/mast

We aim at extending this dyadic dialogue framework to a multi-party one. At first, removing the restriction of two participants makes it possible to have participants that may join and/or leave the system during the dialogue. At second, the division of the multi-party dialogue among ontology-based channels is not limited to unobtrusive observations but allows unsolicited suggestions like in a newsgroup.

Acknowledgements

We would like to thank the referees for their revelant suggestions which helped improve this paper.

References

[1] L. Amgoud and C. Cayrol. On the acceptability of arguments in preference-based argumentation framework. In *Proc. of 14th Conference on Uncertainty in Artificial Intelligence*, pages 1–7, 1998.

[2] Leila Amgoud and Simon Parsons. Agent dialogues with conflicting preferences. In *Proc. of the International Workshop on Agent Theories, Architectures and Languages*, 2001.

[3] Brahim Chaib-draa and Frank Dignum. Trends in agent communication language. *Trends in Agent Communication Language*, 18(2), 2002.

[4] Yannis Labrou and Tim Finin. Semantics and conversations for an agent communication language. In *Proc. of the 15th international joint conf. on Artifical Intelligence*. Morgan Kaufmann Publisher, 1997.

[5] Simon Parsons, Michael Wooldridge, and Leila Amgoud. An analysis of formal inter-agent dialogues. In Cristano Castelfranchi and W. Lewis Johnson, editors, *Proc. of the first international joint conf. on autonomous agent and multiagent systems*, volume part 1, pages 394–401. ACM press, 2002.

[6] Monty Python. The argument clinic. Monty Python's Flying Circus: Just the Words, Volume 2, 1972. episode 29. Methuen, ISBN 0-413-62550-8.

[7] M.D. Sadek and P. Breiter. A rational agent as a kernel of a cooperative dialogue system : Implementing a logical theory of interaction. In *Proc. of ECAI 96 Workshop ATAL*, pages 261–276. Springer-Verlag, 1996.

[8] J.R. Searle. *Speech Acts : An Essay in the Philosophy of Language.* Cambridge University Press, 1969.

[9] Munindar Singh. A social semantics for agent communication languages. In *Proc. of the IJCAI Workshop on Agent Communication Languages.* Springer-Verlag, 2000.

[10] D. Walton and E. Krabbe. *Commitment in Dialogue.* SUNY Press, 1995.

Algebra and sequent calculus for epistemic actions

Alexandru Baltag and Bob Coecke
Oxford University Computing laboratory
baltag / coecke@comlab.ox.ac.uk

Mehrnoosh Sadrzadeh
Université du Québec À Montréal
sadrzadeh.mehrnoosh@courrier.uqam.ca

Abstract

We introduce an algebraic approach to Dynamic Epistemic Logic. This approach has the advantage that: (i) its semantics is a transparent algebraic object with a minimal set of primitives from which most ingredients of Dynamic Epistemic Logic arise, (ii) it goes with the introduction of non-determinism, (iii) it naturally extends beyond boolean sets of propositions, up to intuitionistic and non-distributive situations, hence allowing to accommodate constructive computational, information-theoretic as well as non-classical physical settings, and (iv) introduces a structure on the actions, which now constitute a quantale. We also introduce a corresponding sequent calculus (which extends Lambek calculus), in which propositions, actions as well as agents appear as *resources* in a resource-sensitive dynamic-epistemic logic.

1 Introduction

Dynamic Epistemic Logic (DEL) is a PDL-style logic to reason about epistemic actions and updates in a *multi-agent system*. It focuses in particular on epistemic programs, i.e. programs that update the information state of agents, and it has applications to modelling and reasoning about information-flow and information exchange between agents. This is a major problem in several fields such as *secure communication* where one has to deal with the privacy and authentication of communication protocols, *Artificial Intelligence* where agents are to be provided with reliable tools to reason about their environment and each other's knowledge, and *e-commerce* where agents need to have knowledge acquisition strategies over complex networks.

The standard approach to information flow in a multi-agent system has been presented in [8] but it does not present a formal description of epistemic programs and their updates. The first attempts to formalize such programs and updates were done by Plaza [21], Gerbrandy and Groeneveld [12], and Gerbrandy [10, 11]. However, they only studied a restricted class of epistemic programs. A general notion of epistemic programs and updates for DEL was introduced in [4, 5]. However, in this approach the underlying logic on propositions is boolean. For computational purposes one might want to relax this to an intuitionistic setting, hence conceiving propositions as being structured in a Heyting algebra. On the other hand, continuous lattices are also models of partiality of knowledge [9], and are in general not distributive. Finally, actual physical computational situations such as quantum computation require (at least) a non-boolean setting.

In this paper we generalize 'boolean' DEL by introducing the notion of an *abstract epistemic system*. This generalization goes hand-in-hand with the introduction of non-determinism for states and actions and brings algebraic clarity to the semantics. The particular algebraic object which we introduce is a refinement of previously used objects tailored to study concurrency in computer science [1, 22] and the dynamics and interaction of physical systems [6]. Such an abstract epistemic system consists of a *quantale Q of epistemic programs*, a *Q-right module M of epistemic propositions*, and each agent is encoded by an *appearance map* i.e. an *endomorphism of the (M, Q)-structure*. We show that the boolean DEL of [5] is a concrete example of such an abstract epistemic system. The axioms of the modal operators follow immediately from abstract properties of quantales and modules over them. Crucial notions of DEL are definable abstractly and some new notions emerge naturally. The passage to a non-boolean theory also provides a new insight into epistemic programs such as *public announcement* and, of a surprisingly different status, *public refutation*. We sketch an analysis of the muddy children puzzle and of a cryptographic attack in our setting and also provide a motivating example for the passage to a non-boolean theory. We also provide a corresponding sequent calculus in which sequents will typically look like

$$m_1, \ldots, q_1, \ldots, A_1, \ldots, m_k, \ldots, q_l, \ldots, A_n \vdash \delta$$

where $m_1, \ldots, m_k$ are propositions, $q_1, \ldots, q_l$ are actions and $A_1, \ldots A_n$ are agents which resolve into a single proposition or action δ. The fragment of the calculus restricted to actions is the Lambek calculus [18], hence resource sensitive.

2 Epistemic propositions and epistemic programs

In this section we slightly recast and enrich the Dynamic Epistemic Logic of [5] in such a way that it enables a smooth passage to the algebraic setting to be introduced in Section 4. Part of this involves the introduction of non-determinism for both states and actions.

State models. For a set of *facts* Φ and a finite set of *agents* $\mathcal{A}$, a *state model* is a triple

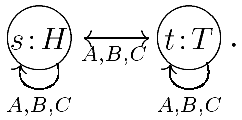
$$\mathbf{S} = (S, \xrightarrow{A}, \mu)_{A\in\mathcal{A}}$$

where S is the set of *states*, $\xrightarrow{A} \subseteq S \times S$ the *accessibility relation* for each agent $A \in \mathcal{A}$, and $\mu : S \to \mathcal{P}(\Phi)$ the *valuation map* which encodes satisfaction $s \models \varphi \Leftrightarrow \varphi \in \mu(s)$. The "facts" $\varphi \in \Phi$ are simple, objectives features of the world ("objective" in the sense of non-epistemic, i.e. independent of the agents' knowledge or beliefs), and the valuation map tell us what facts hold in a given state $s \in S$. Each accessibility relation can be repackaged as a map

$$f_A : S \to \mathcal{P}(S) :: s \mapsto f_A(s) := \{t \in S \mid s \xrightarrow{A} t\} ,$$

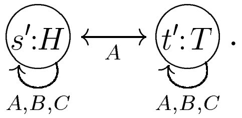
called the *appearance map* of agent A. The significance of the appearance maps is as follows: if $t \in f_A(s)$ then, whenever agent A is in state s he considers state t as a 'possible world'. In other words, if the actual state of the system is s, agent A thinks t may be the actual state.

As an example,[1] consider two players A, B and a referee C. In front of everybody, the referee throws a fair coin, catches it in his palm and fully covers it, before anybody (including himself) can see on which side the coin has landed. There are two possible states here, state s in which 'the coin lies Heads' up ($= H \in \Phi$), hence $\mu(s) = \{H\}$, and state t in which the coin lies Tails up ($= T \in \Phi$), hence $\mu(t) = \{T\}$. We depict the state model **Toss** as



For every agent there are arrows between any two states (including identical states), which means that nobody knows the 'real state'.

We can also consider a case in which agents B and C can see the face of the coin, but agent A cannot see it (although he knows that the others see it), so he is still uncertain if the coin is heads or tails. In this case only agent A has several arrows between states whereas agents B and C have only one arrow in each state, which means that if the coin is heads up they know it and similarly for tails up. Hence **PToss** gets depicted as

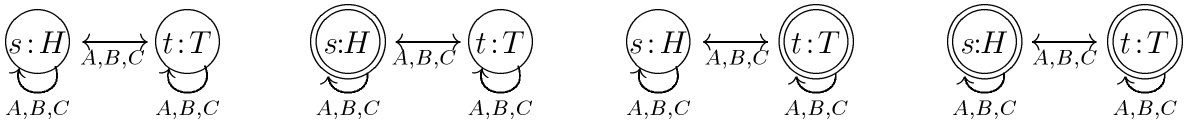


An *epistemic proposition P over a state model* **S** is a subset P of S, containing all the states at which the proposition is 'true'. The maps μ and f_A of the state model are extended to elements of P as follows

$$\mu(P) := \bigcap\{\mu(s) \mid s \in P\} \in \mathcal{P}(\Phi) \qquad \text{and} \qquad f_A(P) := \bigcup\{f_A(s) \mid s \in P\} \in \mathcal{P}(S)\,.$$

Note that we have to use intersection and not union in defining $\mu(P)$ since a fact is entailed by an epistemic proposition when it holds at all the states of the proposition. This makes the passage from $\mathcal{P}(S)$ to $\mathcal{P}(\Phi)$ contravariant. In other words, the actual algebra of facts is $\mathcal{P}(\Phi)^{op}$, that is, the complete boolean algebra $\mathcal{P}(\Phi)$ where the order is reversed i.e. $\varphi_1 \leq^{op} \varphi_2 \Leftrightarrow \varphi_1 \supseteq \varphi_2$. While facts are simple and non-epistemic, and thus cannot be altered by epistemic actions (see further), epistemic propositions can express complex features of the world, which may depend on the agents' knowledge (and so may be changed by epistemic actions). However, notice that each fact $\varphi \in \Phi$ corresponds to an epistemic proposition $P_\varphi := \{s \in S \mid \varphi \in \mu(s)\}$, saying that the fact holds in the current state.

In the **Toss** model, H and T are facts expressing the heads up or tails up of the coin. The epistemic propositions that correspond to these facts are the states in which the fact holds. The epistemic propositions are $\emptyset, \{s\}, \{t\}, \{s, t\} \subseteq \{s, t\}$. We depict an epistemic proposition over a state model by double-circling the included states, hence



represent the four epistemic propositions of **Toss**.

When a proposition P has exactly one state $s \in P$ (i.e. $P = \{s\}$ is a singleton), we shall use systematic ambiguity, identifying the proposition with the state and writing e.g. $P = \{P\}$.

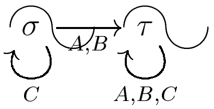
[1] For a more elaborated example of an authentication protocol we refer the reader to [2].

Action models. Given a state model S, *an action model over* S is a triple

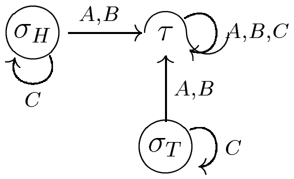
$$\sum = (\Sigma, \xrightarrow{A}, \mu)$$

similar to a state model except that we think of the elements of Σ as possible *actions* instead of possible states and the valuation $\mu : \Sigma \to \mathcal{P}(S)$ assigns to each action σ a *precondition,* i.e. a proposition $\mu(\sigma)$ definining the domain of applicability of σ: action σ can happen in a state s iff $s \in \mu(\sigma)$; e.g. a truthful announcement of a fact can only happen in those states where that fact holds. Note that since $\mathcal{P}(S)$ is boolean we can equivalently consider the states at which the action *cannot take place* , denoted as $Ker(\sigma) := S \setminus \mu(\sigma)$ for each $\sigma \in \Sigma$. The *effect* of an action on states and appearance maps will be defined below in terms of an *epistemic update* product.

We introduce an action model over **Toss**. After catching the coin in his hand the referee might secretly take a peek at the coin before covering it while nobody notices this. The action model is now depicted as



where σ stands for 'cheating' and τ for 'nothing happens' and $\mu(\sigma) = \{s, t\}$. The action model can be refined when replacing σ by σ_H and σ_T where $\mu(\sigma_H) = H$ and $\mu(\sigma_T) = T$, specifying what the referee saw in case of deceit:

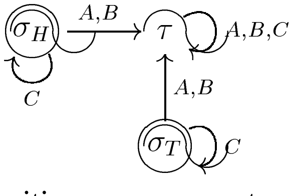


An *epistemic program* π *over an action model* Σ is a subset π of Σ; the μ and f_A maps are both extended covariantly by continuity

$$\mu(\pi) := \bigcup\{\mu(\sigma) \mid \sigma \in \pi\} \in \mathcal{P}(S) \quad \text{and} \quad f_A(\pi) := \bigcup\{f_A(\sigma) \mid \sigma \in \pi\} \in \mathcal{P}(\Sigma)\,.$$

The union in the definition of μ maps for programs says that an epistemic program is applicable where at least one of its actions is applicable. This makes the Ker map follow contravariantly by boolean negation i.e. $Ker(\pi) = S \setminus \mu(\pi)$. Epistemic programs introduce non-determinism: whenever $\pi_1 \subseteq \pi_2$ then π_2 is obtained from π_1 by increasing nondeterminism; $\pi = \{\sigma_1, \sigma_2\}$ stands for "either action σ_1 or action σ_2 takes place".

In our example with actions σ_H, σ_T and τ the epistemic program $\{\sigma_H, \sigma_T\}$ stands for the non-deterministic action σ, in the sense that the outcome of the toss can be either. We depict the program over an action by double-circling the including actions. Hence the picture of the program $\pi = \{\sigma_H, \sigma_T\}$ over $\sum$ is

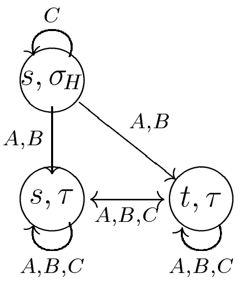


As in the case of states and propositions, we use systematic ambiguity to identify *deterministic* programs $\pi = \{\sigma\}$ with their unique undferlying action σ.

Update. Given a state model **S** and an action model $\sum$ over **S** we define their *update product* **S** $\otimes \sum$ to be a new state model given by

$$S \otimes \Sigma := \bigcup_{\sigma \in \Sigma} \mu(\sigma) \times \{\sigma\} \qquad f_A(s,\sigma) := (f_A(s) \times f_A(\sigma)) \cap (S \otimes \Sigma) \qquad \mu(s,\sigma) := \mu(s)\,,$$

hence $S \otimes \Sigma \subseteq S \times \Sigma$ and $f_A(s,\sigma) \subseteq f_A(s) \times f_A(\sigma)$. In our example, after the cheating action σ_H where the coin has lied Heads up, A and B think that nobody knows on which side the coin is lying. But they are wrong! The system after this action can be updated by taking the update product of the two models **Toss** and σ_H depicted above:



Note that in general $S \otimes \Sigma$ and S are not necessarily disjoint.[2]

Definition 2.1 We define the *update product of an epistemic proposition P over* **S** *and an epistemic program π over* $\sum$ as the epistemic proposition

$$P \otimes \pi := \bigcup_{\sigma \in \pi} (\mu(\sigma) \cap P) \times \{\sigma\} \subseteq \; P \times \pi \;\; \text{over} \;\; \mathbf{S} \otimes \sum .$$

The proposition $P \otimes \pi$ provides the *strongest postcondition* for P with respect to epistemic program π: for each state in $P \otimes \pi$ the proposition P holds before running the π. It can be seen that $P \otimes \pi = \emptyset$ iff $P \cap \mu(\pi) = \emptyset$, where $\emptyset$ is the *falsum* (i.e. the trivially false epistemic proposition over **S**).

Modalities. We define the *epistemic modality* for each agent $A \in \mathcal{A}$ as the unary connective which assigns to proposition $P \subseteq S$ over **S** another proposition

$$\Box_A P := \big\{ s \in S \;\big|\; f_A(s) \subseteq P \big\} \;\; \text{over} \;\; \mathbf{S}.$$

We read $\Box_A P$ as ‘agent A knows or believes P’.[3]

We define the *dynamic modality* for each epistemic program π over $\sum$ as the unary connective which assigns to proposition $P \subseteq S$ over **S** another proposition

$$[\pi]P := \big\{ s \in S \;\big|\; \{s\} \otimes \pi \subseteq P \big\} = \bigcup \big\{ Q \in \mathcal{P}(S) \;\big|\; Q \otimes \pi \subseteq P \big\} \;\; \text{over} \;\; \mathbf{S}\,.$$

Note that (as mentioned before) some states $s \in S$ can be themselves pairs of states and actions (s, σ) which make the above definition well defined. The proposition $[\pi]P$ provides the *weakest precondition* for P with respect to the epistemic program π: for each state in $[\pi]P$ the proposition P holds after running π.

[2] In fact later, the most important models we shall consider later (DEL models) are closed with respect to update product, i.e. $S \otimes \Sigma \subseteq S$.

[3] Taking either ‘knows’ or ‘beliefs’ depends on the context.

Sequential composition. The *sequential composition* $\sum_1 \bullet \sum_2$ over S of two action models $\sum_1$ and $\sum_2$ both over S means 'first do $\sum_1$ and then do $\sum_2$' and is defined as

$$\Sigma_1 \bullet \Sigma_2 := \Sigma_1 \times \Sigma_2 \quad f_A(\sigma_1, \sigma_2) := f_A(\sigma_1) \times f_A(\sigma_2) \quad \mu(\sigma_1, \sigma_2) := \mu(\sigma_1) \cap [\sigma_1]\mu(\sigma_2).$$

Again note that $\Sigma_1 \bullet \Sigma_2$ and Σ_1 (or Σ_2) are not necessarily disjoint.[4] The action model over a state model S contains an action *skip* in which nothing happens iff [5]

$$\text{skip} = \{\text{skip}\} \qquad \mu_{\text{skip}} = S = \top_{P(S)} \qquad f_A(\text{skip}) = \{\text{skip}\}\,.$$

Notice the use of systematic ambiguity: we denoted with the same name (*skip*) both the program *skip* and its only action. It is easy to see that skip is a unit, up to isomorphism, both for update product and sequential composition.

Definition 2.2 We define the *sequential composition of two epistemic programs* π_1 *over* $\sum_1$ *and* π_2 *over* $\sum_2$ as the epistemic proposition $\pi_1 \bullet \pi_2 := \pi_1 \times \pi_2$ over $\sum_1 \bullet \sum_2$.

Concrete epistemic systems. We now have all the tools to make the passage of DEL in the sense of [5] to 'concrete epistemic systems' which we put forward as a stepping-stone towards 'abstract epistemic systems'. A DEL model is essentially one that is closed under update product and sequential composition (and contains a *skip*), while a concrete epistemic system consists of all the epistemic propositions and all the epistemic programs of a DEL model:

Definition 2.3 A DEL model is a pair $(\mathbf{S}, \sum)$ where $\mathbf{S}$ is a state model and $\sum$ is an action model over $\mathbf{S}$ such that *skip* $\in \Sigma$, $(S \otimes \Sigma) \subseteq S$ and $(\Sigma \bullet \Sigma) \subseteq \Sigma$.

Definition 2.4 Given a DEL model $(\mathbf{S}, \sum)$, a *concrete epistemic system* is the pair $(\mathcal{P}(S), \mathcal{P}(\Sigma))$ which goes equipped with valuation μ, appearance maps $\{f_A\}_{A \in \mathcal{A}}$ and all other operations of the DEL model extended to $\mathcal{P}(S)$ and $\mathcal{P}(\Sigma)$ as we showed above.

3 The algebra of programs and propositions

A *sup-lattice* L is a complete lattice with maps which preserve arbitrary joins as homomorphism. Recall that each sup-lattice also has arbitrary meets, namely

$$\bigwedge_i a_i = \bigvee \{b \in L \mid \forall i, b \leq a_i\}$$

for any $A \subseteq L$. Hence the designation 'sup-lattice refers to the fact that we require structure-preserving maps only to preserve arbitrary joins (cf. the designations *locales* and *frames* for complete Heyting algebras [16]). We denote *bottom* and *top* of L by $\perp$ and $\top$ respectively and define its set of *atoms* as

$$Atm(L) := \{p \in L \setminus \{\perp\} \mid a \leq p \Rightarrow a = \perp\}.$$

[4] In fact later we only consider models where $\Sigma \bullet \Sigma \subseteq \Sigma$.

[5] This action has been denoted as τ in the preceding examples.

A lattice L is *atomistic* iff

$$\forall a \in L, a = \bigvee \{p \in Atm(L) \mid p \leq a\}\,.$$

Every sup-morphism $f^* : L \to M$ has a (unique) right Galois adjoint f_* satisfying

$$\frac{f^*(a) \leq b}{\overline{a \leq f_*(b)}}$$

and can be explicitly given as

$$f_* : M \to L :: b \mapsto \bigvee \{a \in L \mid f^*(a) \leq b\}.$$

The *left Galois adjoint* f^* moreover preserves arbitrary meets. We denote an adjoint pair by $f^* \dashv f_*$. In computational terms, one can think of the left Galois adjoint f_* as assigning weakest preconditions with respect to the program f^*.

A *quantale*[6] is a sup-lattice Q equipped with a monoid structure $(Q, \bullet, 1)$ satisfying

$$a \bullet \left(\bigvee_i b_i\right) = \bigvee_i (a \bullet b_i) \qquad\qquad \left(\bigvee_i a_i\right) \bullet b = \bigvee_i (a_i \bullet b)\ .$$

Hence for all $a \in Q$ the maps $a \bullet - : Q \to Q$ and $- \bullet a : Q \to Q$ preserve arbitrary joins and hence they have Galois adjoints $(a \bullet -) \dashv (a \setminus -)$ and $(- \bullet a) \dashv (-/a)$ explicitly given by

$$a \setminus b := \bigvee \{c \in Q \mid a \bullet c \leq b\} \qquad\qquad b/a := \bigvee \{c \in Q \mid c \bullet a \leq b\}.$$

We refer to $(a \setminus -)$ and $(-/a)$ as the *residual* operations. A *quantale homomorphism* is both a sup-homomorphism and a monoid-homomorphism. Examples of quantales are: the set $\mathsf{sup}(L)$ of all sup-endomorphisms of a complete lattice L ordered pointwisely; the set of all relations from a set X to itself ordered by pointwise inclusion — this quantale is isomorphic to $\mathsf{sup}(\mathcal{P}(X))$; the powerset of any monoid with composition extended by continuity.

A *Q-right module* for a quantale Q is a sup-lattice M which goes equipped with a *module action* $- \otimes - : M \times Q \to M$, that is,

$$m \otimes 1 = m$$

$$m \otimes (q_1 \bullet q_2) = (m \otimes q_1) \otimes q_2$$

$$m \otimes (\bigvee_i q_i) = \bigvee_i (m \otimes q_i) \qquad (\bigvee_i m_i) \otimes q = \bigvee_i (m_i \otimes q)$$

Again we have two right Galois adjoints $- \otimes q \dashv [q]-$ and $m \otimes - \dashv \{m\}-$ where

$$[q]m := \bigvee \{m' \in M \mid m' \otimes q \leq m\} \qquad\qquad \{m\}m' := \bigvee \{q \in Q \mid m \otimes q \leq m'\}.$$

As for some examples, a quantale Q is a Q-right module over itself with composition as the tensor and a complete lattice L is a $\mathsf{sup}(L)$-right module with function application as the tensor.

[6] The term 'quantale' was introduced in [20]. For a survey on quantales we refer to [23]. For insightful categorical perspectives on quantales and Q-modules we refer to [17] and [24].

Definition 3.1 [1] A *system* is a pair (M, Q) with Q a quantale and M a Q-right module.

A system is *atomistic* when both M and Q are atomistic and the following equations hold

$$\begin{aligned} m \in Atm(M), q \in Atm(Q) &\implies m \otimes q \in Atm(M) \cup \{\bot\} \\ q_1, q_2 \in Atm(Q) &\implies q_1 \bullet q_2 \in Atm(Q) \,. \end{aligned}$$

These conditions can be interpreted as the fact that 'the atoms of both the quantale and the module behave deterministically'.

Proposition 3.2 i. *Epistemic programs $\mathcal{P}(\Sigma)$ with $\bigcup$ as $\bigvee$, sequential composition as $\bullet$ and 'skip' as 1 form a quantale.*[7] **ii.** *Epistemic propositions $\mathcal{P}(S)$ with $\bigcup$ as $\bigvee$ and update product as $\otimes$ form a right $\mathcal{P}(\Sigma)$-module.* **iii.** *The pair $(\mathcal{P}(S), \mathcal{P}(\Sigma))$ is an atomistic system. The atoms of the module $\mathcal{P}(S)$ correspond to the* states $s \in S$, *while the atoms of the quantale $\mathcal{P}(\Sigma)$ correspond to the* actions $\sigma \in \Sigma$.

Proposition 3.3 i. *The appearance maps $f_A : \mathcal{P}(S) \to \mathcal{P}(S)$, and for all $\pi \in \Sigma$ the maps $- \otimes \pi : \mathcal{P}(S) \to \mathcal{P}(S)$ are all sup-homomorphisms.* **ii.** *The appearance maps $f_A : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$, and for all $\pi \in \Sigma$ the maps $\pi \bullet -, - \bullet \pi : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$ are quantale-homomorphisms.* **iii.** *For every epistemic proposition $P \in \mathcal{P}(S)$ and every epistemic program $\pi \in \mathcal{P}(\Sigma)$, we have*

$$f_A(P \otimes \pi) \subseteq f_A(P) \otimes f_A(\pi) \,.$$

iv. *For every* state *(i.e. atomic proposition) $s \in S$ and every* action *(i.e. atomic program) $\sigma \in \Sigma$ we have that:*

$$\textit{if} \quad s \otimes \sigma \neq \emptyset \quad \textit{then} \quad f_A(s \otimes \sigma) = f_A(s) \otimes f_A(\sigma) \,.$$

The last property can be generalised by introducing a notion of *coherence*:

Definition 3.4 A pair (P, π) where P is an epistemic proposition and π is an epistemic program is *coherent* iff

$$\forall s \in P,\ \forall \sigma \in \pi \ \ s \otimes \sigma \neq \emptyset$$

i.e. iff $P \subseteq \mu(\sigma)$ for every $\sigma \in \pi$. This means that proposition P ensures the possibility of all the actions subsumed by program π. An equivalent definition which doesn't refer to states or actions is the following:

$$\forall P' \subseteq P,\ \forall \pi' \subseteq \pi \ (P' \otimes \pi' = \emptyset \ \Rightarrow \ P' = \emptyset \text{ or } \pi' = \emptyset) \,.$$

Proposition 3.5 *If (P, π) is a coherent pair then we have*

$$f_A(P \otimes \pi) = f_A(P) \otimes f_A(\pi) \,.$$

[7] This construction is implicit in the relational composition of dynamic actions in [14].

Proposition 3.6 **i.** *For $A \in \mathcal{A}$ the right Galois adjoint to appearance $f_A^{\mathbf{S}}(-) : \mathcal{P}(S) \to \mathcal{P}(S)$ is knowledge $\Box_A^{\mathbf{S}}-$ (=the epistemic modality).* **ii.** *For $\pi \in \mathcal{P}(\Sigma)$ the right Galois adjoint to update $- \otimes \pi : \mathcal{P}(S) \to \mathcal{P}(S)$ is the dynamic modality $[\pi]-$.* **iii.** *The right Galois adjoint to appearance $f_A^{\Sigma}(-) : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$ introduces an epistemic modality $\Box_A^{\Sigma}-$ on actions.* **iv.** *The right Galois adjoint to left- and right-composition $\pi \bullet -, - \bullet \pi : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$ introduce respectively weakest pre-specification $\pi \backslash -$ and strongest post-specification $\pi / -$, and the right Galois adjoint to $P \otimes - : \mathcal{P}(\Sigma) \to \mathcal{P}(S)$ introduces $\{m\}-$, a variant on this.*[8]

Proof. All follows by construction and basic facts on sets, cartesian products and relations. □

4 Abstract epistemic systems

The propositions of the previous section lead us to the following definitions.

Definition 4.1 A *system-endomorphism* $(M, Q) \xrightarrow{f} (M, Q)$ is a pair

$$\big(f^M : M \to M\ ,\ f^Q : Q \to Q\big)$$

where f^M is a sup-homomorphism, f^Q is a quantale homomorphism and

$$f^M(m \otimes q) \leq f^M(m) \otimes f^Q(q) \tag{1}$$

for all $m \in M$ and $q \in Q$.

Definition 4.2 An *epistemic system* is a tuple $(M, Q, \{f_A\}_{A \in \mathcal{A}})$ where (M, Q) is a system and $\{f_A\}_{A \in \mathcal{A}}$ are system-endomorphisms.

Interpretation. The elements of the quantale Q are to be thought of as the *epistemic programs* and its unit as *skip*, the elements of the module M are to be thought of as the *epistemic propositions*, or if one wants, the not necessarily deterministic states, the labels $A \in \mathcal{A}$ are the *agents* with the endomorphisms $\{f_A\}_{A \in \mathcal{A}}$ as their *appearance maps*. The *kernel* of a program $q \in Q$ is

$$Ker(q) := \{m \in M \mid m \otimes q = \bot\}$$

and comprises the *preconditions*: it contains the epistemic propositions to which q cannot be applied. The *stabilizer*

$$Stab(Q) := \{m \in M \mid \forall q \in Q, [q]m = m\}$$

comprises the *facts*: it consists of those epistemic propositions which are stable under epistemic actions. The *satisfaction* relation is included in the partial ordering of M: for a state $m \in M$ and fact $\varphi \in Stab(Q)$ we have $m \models \varphi \Leftrightarrow m \leq \varphi$. All modalities and other right Galois adjoints discussed and introduced in Proposition 3.6 arise also here as right Galois adjoints and hence there interpretation still holds e.g. “knowledge $\Box_A^M$ is the adjoint to appearance f_A^M”.

[8] The residual $\pi \setminus -$ assigns to its argument δ the weakest program $\pi \setminus \delta$ which one has to effectuate *after* effectuating π such that the net effect is below δ. The residual $-/\pi$ assigns to its argument δ the strongest program δ/π which one has to effectuate *before* effectuating π such that the net effect is below δ. The right Galois adjoint does $\{m\}-$ assigns to its argument δ the weakest proposition $\{m\}P$ before effectuating π which guarantees P after. For a discussion on pre- and post-specification we refer to [7, 15].

Nature of the modalities. We identify the basic properties of the modalities.

Proposition 4.3 *In any epistemic system we have*

$$\Box_A^M \top = \top \qquad \Box_A^M (m \wedge m') = \Box_A^M m \wedge \Box_A^M m' \qquad \frac{m \leq m'}{\Box_A^M m \leq \Box_A^M m'}\,.$$

Proof: Since $\Box_A^M$ is a right Galois adjoint it preserves arbitrary meets, that is $\Box_A^M(\bigwedge_i m_i) = \bigwedge_i \Box_A^M m_i$, and hence it preserves the empty meet and binary meets, and is monotone. □

Since all other modalities preserve arbitrary meets the same result holds for them and for all other right Galois adjoints. In an intuitionistic context where one might take M to be a *frame* (i.e. a (complete) Heyting algebra with sup-homomorphisms) we can internalize the partial order using the defining property of a Heyting algebra so we obtain

$$\frac{\vdash m \to m'}{\vdash \Box_A^M m \to \Box_A^M m'}\,.$$

Hence in the special case that $Q = \{1\}$ and $A = \{*\}$ we obtain the intuitionistic modal logic $\mathbf{IntK}_\Box$ of [26]. We conclude that *intuitionistic epistemic systems*, that is epistemic systems for which M is a frame, generalize intuitionistic modal logic to multiple agents and dynamics in terms of epistemic programs. If M is moreover a complete boolean algebra such as the powerset of Section 2 then Kripke's axiom **K** follows i.e.

$$\Box_A^M (m \to m') \to (\Box_A^M m \to \Box_A^M m').$$

Diamonds and corresponding rules arise in that case by duality.

Learning. The fact that eq(1) in definition 4.1 is an *inequality* expresses learning of agents. Some of the clauses of the appearance of an agent on an update product might get eliminated from the left hand side of eq(1) simply because some of the sub-action of the program might not be applicable on some of the sub-states of the proposition. This implies that the agent learns something new as the result of update (left hand side is stronger than the right hand side).

We can also force the equality by introducing the notion of coherence:

Definition 4.4 A pair (m, q) where $m \in M$ and $q \in Q$ is *coherent* iff

$$\forall m' \leq m,\ \forall q' \leq q\ (m' \otimes q' = \bot\ \Rightarrow\ m' = \bot \text{ or } q' = \bot)$$

In an *atomistic* system, every pair $(m, q) \in Atm(M) \times Atm(Q)$ of an atomic proposition and an atomic action where $m \notin ker(q)$ is coherent.

Definition 4.5 A *strong system endomorphism* is a system endomorphism where for all coherent pairs (m, q) we have

$$f^M(m \otimes q) = f^M(m) \otimes f^Q(q)\,.$$

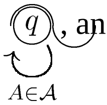
A *strong epistemic system* is a tuple $(M, Q, \{f_A\}_{A \in \mathcal{A}})$ where (M, Q) is a system and $\{f_A\}_{A \in \mathcal{A}}$ are strong system-endomorphisms.

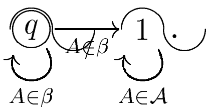
Theorem 4.6 *Every atomistic strong epistemic system for which both M and Q are completely distributive boolean algebras can be represented as a concrete epistemic system.*

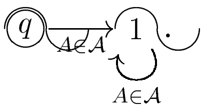
Proof: It suffices to set $S := Atm(M)$, $\Sigma := Atm(Q)$ and $\Phi := Stab(Q)$. The accessibility relations arise from the appearance maps, satisfaction from $\varphi \in \mu(s) \Leftrightarrow s \leq \varphi$ for $s \in S$ and $\varphi \in \Phi$ and preconditions from $\mu(\sigma) := S \setminus Ker(\sigma)$ for $\sigma \in \Sigma$. □

5 Some dynamic epistemic situations

For a given epistemic system $(M, Q, f_A)_{A\in\mathcal{A}}$ the following are some examples of some special epistemic programs that can be defined in the system. Note that $Ker(q) = \downarrow(\bigvee Ker(q))$, where $\downarrow a := \{b \in L \mid b \leq a\}$, and hence "being not in the precondition of q" exists as a proposition in M for all $q \in Q$.

1. **Public refutation** of the proposition $m \in M$ is an epistemic program $q \in Q$ with $\{f_A(q)\}_{A\in\mathcal{A}} = q$, that is: , and for which $Ker(q) = \downarrow m$.

2. **Private refutation to subgroup** This is also a program that privately refutes a proposition m to the subgroup β of agents. $Ker(q)$ is the same as before and $\{f_A(q)\}_{A\in\beta} = q$ and $\{f_A(q)\}_{A\in\mathcal{A}\setminus\beta} = 1$, that is: .

3. **Failure test** of a proposition m is a program q that tests when m fails. It is a particular case of private refutation where m is refuted to an empty set of agents $Ker(q) = \downarrow m$ and $\{f_A(q)\}_{A\in\mathcal{A}} = 1$, that is: .

4. **Public announcement** is also definable in our setting. However, while "being not in the precondition of q" is a proposition in M for all $q \in Q$, this is not the case for "being in the precondition of q". To see this consider the lattice $\{\bot \leq a, b, c \leq \top\}$ with q such that $Ker(q) = \{\bot, a\}$ where in the language of Section 2 we have $\mu(q) = \{b, c\}$, which can not be represented by a single element of M. The reason for this is that this lattice is non-boolean. Hence public announcement of the proposition $m \in M$ is an epistemic program $q \in Q$ for which $f_A(q) = q$ and for which $\bigvee Ker(q)$ has a *boolean complement* $(\bigvee Ker(q))^c$, satisfying $(\bigvee Ker(q))^c = m$.

We now present some case studies. Given an epistemic system $(M, Q, f_A)_{A\in\mathcal{A}}$ on which we impose particular conditions which encode the desired state and action models.

Cheating. Consider the 'cheating' scenario of the first section where the set of agents is $\mathcal{A} = \{A, B, C\}$. Recall that there are two possibilities in the state model **Toss**, s in which the coin is Heads up and t in which it is Tails up. We model this abstractly by assuming as given an epistemic system (M, Q), with $s, t \in M$ and $\sigma_H \in Q$. The facts are encoded as stabilizers, i, e. we are given propositions $H, T \in Stab(Q)$. All these are assumed to satisfy the following conditions: $f_i(s) = f_i(t) = s \vee t$ for all $i \in \mathcal{A}$ $s \leq H, t \leq T, H \wedge T = \perp$; the epistemic program $\sigma_H \in Q$ has maps $f_A(\sigma_H) = f_B(\sigma_H) = 1$ and $f_C(\sigma_H) = \sigma_H$, and kernel $Ker(\sigma_H) =\downarrow t$. This program describes an instance of cheating where the coin is heads up. $s \otimes \sigma_H \in M$ is the proposition s after it is updated by σ_H.

Let us reason about this scenario, using our algebraic setting, e. to prove that $s \otimes \sigma_H \leq \Box_C H$. Indeed by $\{f_A\}_{A \in \mathcal{A}}$ being system homomorphisms and eq(1) we have

$$f_A(s \otimes \sigma_H) \leq f_A(s) \otimes f_A(\sigma_H) = (s \vee t) \otimes 1 = s \vee t\,,$$

and the same goes for f_B. On the other hand

$$f_C(s \otimes \sigma_H) \leq f_C(s) \otimes f_c(\sigma_H) = (s \vee t) \otimes \sigma_H = (s \otimes \sigma_H) \vee (t \otimes \sigma_H) = s \otimes \sigma_H$$

since $t \in Ker(\sigma_H)$. We have $s \leq H$ iff $s \otimes \sigma_H \leq H \otimes \sigma_H$ and by the definition of $Stab(Q)$ we get $s \otimes \sigma_H \leq H$. Thus $f_C(s \otimes \sigma_H) \leq H$ and by adjunction we get $s \otimes \sigma_H \leq \Box_C H$ which means after updating his initial state by taking a peek, the referee knows that the coin is heads up.

If the referee is honest he uncovers the coin without taking a peek. He then publicly refutes the ‘coin being tails’. The epistemic program in this case is the public refutation of proposition t where $f_A(q) = f_B(q) = f_C(q) = q$ and $Ker(q) = \{t\}$. It follows that $s \otimes q \leq \Box_A H$, and the same goes for B and C. Hence all the agents know that the coin is Heads up after the public refutation.

The muddy children puzzle. We refer the reader for the details of the general case to [8]. Here we discuss the case of three children A, B, C playing in the mud with A and B having muddy foreheads. Their father publicly announces that at least one of them has mud on his forehead and asks once if they know that they are dirty. After they all simultaneously reply “No!” once, the muddy children A and B will know that they are muddy. This simple case has only one round (since the number of dirty children is 2), but the general case with k dirty children shall have $k - 1$ rounds of ”No!” replies.

As before, we model this by postulating as given an epistemic system (M, Q). The set of agents $\mathcal{A}$ includes children $\{A, B, C\}$. The module M includes all possible initial states s_β with $\beta \subseteq \mathcal{A}$ being those children that are dirty. Since the children cannot see their own foreheads (which might be dirty or not) we have $f_i^M(s_\beta) = s_{\beta \setminus \{i\}} \vee s_{\beta \cup \{i\}}$ for each child i. Let $D_\emptyset$ be the fact that no child has a dirty forehead and D_i be the fact that child i has a dirty forehead, hence $\{D_\emptyset\} \cup \{D_i \in M \mid i \in \mathcal{A}\} \subseteq Stab(Q)$, and also $s_\beta \leq D_i$ for all $i \in \beta$. Let q be a round of no answers of the 3 children, i.e. q is the public refutation of $\Box_A D_A \vee \Box_B D_B \vee \Box_C D_C$ and hence $Ker(q) = \Box_A D_A \vee \Box_B D_B \vee \Box_C D_C$ and $f_i(q) = q$ for each child i. Let $q_0 \in Q$ be the be father’s announcement that at least one child has mud on his forehead hence $Ker(q_0) =\downarrow D_\emptyset$ and $f_i(q_0) = q_0$ for each child i. We have to show that after the first round of refutation q each muddy child (e.g. A) knows that he is dirty, i.e. $s_{\{A,B\}} \leq [q_0 \bullet q]\Box_A D_A$ and similarly for child B. By adjunction on dynamic and epistemic modalities and module equation $(m \otimes q_1) \otimes q_2 = m \otimes (q_1 \bullet q_2)$ we get

$$f_A((s_{\{A,B\}} \otimes q_0) \otimes q) \leq D_A\,. \tag{2}$$

By the f_A inequality (i.e. eq(1)) it suffices to show

$$f_A(s_{\{A,B\}} \otimes q_0) \otimes f_A(q) \leq D_A$$

Again by eq(1) and the assumption $f_A(q_0) = q_0$

$$f_A(s_{\{A,B\}} \otimes q_0) \leq f_A(s_{\{A,B\}}) \otimes q_0$$

update both sides by $f_A(q) = q$

$$f_A(s_{\{A,B\}} \otimes q_0) \otimes q \leq (f_A(s_{\{A,B\}}) \otimes q_0) \otimes q$$

So to prove eq(2) it suffices to show

$$(f_A(s_{\{A,B\}}) \otimes q_0) \otimes q \leq D_A$$

Replacing f_A by its value will get us

$$((s_{\{A,B\}} \vee s_{\{B\}}) \otimes q_0) \otimes q \leq D_A$$

hence

$$((s_{\{A,B\}} \otimes q_0) \otimes q) \vee ((s_{\{B\}} \otimes q_0) \otimes q) \leq D_A\,.$$

The first disjunct is given by the assumptions $s_{\{A,B\}} \leq D_A$ and D_A being a fact and thus stable under updates, i.e. $(D_A \otimes q_0) \otimes q \leq D_A$. For the other disjunct we shall show that $s_{\{B\}} \otimes q_0 \leq \Box_B D_B \in Ker(q)$ which gives us $(s_{\{B\}} \otimes q_0) \otimes q = \bot$ and $\bot \leq D_A$. To see this use the adjunction to get $f_B(s_{\{B\}} \otimes q_0) \leq D_B$, by eq(1) it suffices to show $f_B(s_{\{B\}}) \otimes f_B(q_0) \leq D_B$. Now replace f_B with its values and get $(s_{\{B\}} \vee s_{\{A,B\}}) \otimes q_0 \leq D_B$ which is equal to $(s_{\{B\}} \otimes q_0) \vee (s_{\{A,B\}} \otimes q_0) \leq D_B$. This inequality holds since by assumption $s_{\{B\}} \leq D_B$ and also $s_{\{A,B\}} \leq D_B$. Hence the result follows.

Note that this proof can be straightforwardlly extended to the general case by induction on the number of dirty children.

A cryptographic attack. Two agents A and B share a secret key so that they can send each other encrypted messages over some communication channel. The channel is not secure: some outsider C may interpret the messages or prevent them from being delivered (although he cannot read them because he does not have the key). Suppose the encryption method is publicly known but the key is secret. It is also known that A is the only one who knows an important secret for example if some fact P holds or not. Suppose now that A sends an encrypted message to B communicating the secret. B gets the message and he is convinced that it must be authentic. Now both A and B are convinced that they share the secret and that C doesn't. However suppose that C notices two features of the specific encryption method: first that the shape of the encrypted message can show whether it contains a secret or it is just junk, second that without knowing the key or the content of the message he can modify the encrypted message to its opposite i.e. if it originally said P hold, it will now say that P does not hold. Now the outsider C will secretly intercept the message, change it appropriately and send it to B without knowing the secret. Now

A and B mistakenly believe that they share the secret, while in fact B got the wrong secret instead! C has succeeded to manipulate their beliefs.

We can encode this situation in an epistemic system. The agents include $\{A, B, C\}$. Let $s, t \in M$ satisfy $s \leq P$ and $t \nleq P$. The only agent that knows if P holds or not is A thus $f_A(s) = s$ and similarly $f_A(t) = t$. On the other hand B and C do not know this so $f_B(s) = f_C(s) = f_B(t) = f_C(t) = s \vee t$. Call the message in which P holds P and the one in which it does not hold $\bar{P}$. The epistemic actions that correspond to the cryptographic attack are the following: α in which the message P is intercepted, modified and sent to B, β in which the message $\bar{P}$ is intercepted, modified and sent to B, α' in which A sends the message P to B, β' in which A sends the message $\bar{P}$ to B, and finally γ which corresponds to sending a junk message. Thus $\{\alpha, \beta, \alpha', \beta', \gamma\} \subseteq Q$ and $P, \bar{P} \in Stab(Q)$ and $P \wedge \bar{P} = \bot,\ P \vee \bar{P} = \top$. In actions α and β agent C is uncertain about which message P or $\bar{P}$ has been sent so $f_C(\alpha) = f_C(\beta) = \alpha \vee \beta$. On the other hand, agent A is sure that he has sent a message (either that P holds or that it doesn't) to B and that B has received exactly the same secret i.e. $f_A(\alpha) = \alpha'$ and $f_A(\beta) = \beta'$. However if P has been sent, B has received $\bar{P}$ so $f_B(\alpha) = \beta'$ and the other way around $f_B(\beta) = \alpha'$. Further $f_A(\alpha') = f_B(\alpha') = \alpha'$ and $f_A(\beta') = f_B(\beta') = \beta'$ and $f_C(\alpha') = f_C(\beta') = \alpha' \vee \beta' \vee \gamma$. C also considers possible that only a junk message has been sent and that is why he sees γ while in α' and β'. If a junk message has been sent, A and B are sure about it $f_A(\gamma) = f_B(\gamma) = \gamma$ while C is unsure if it was a junk message or P or $\bar{P}$, thus $f_C(\gamma) = \alpha' \vee \beta' \vee \gamma$. The kernel of each action is the states which they cannot be applied to i.e. $Ker(\alpha) = Ker(\alpha') = \downarrow \bar{P}$ and $Ker(\beta) = Ker(\beta') = \downarrow P$.

The epistemic program $\alpha \vee \beta$ expresses the action of communicating the secret P or $\bar{P}$ in the above scenario. Now let us update the state s with the epistemic program $\alpha \vee \beta$ and show that after update, if P holds, then A knows that B knows that P holds i.e. $s \otimes (\alpha \vee \beta) \leq \Box_A \Box_B P$. Since this is equal to $(s \otimes \alpha) \vee (s \otimes \beta) \leq \Box_A \Box_B P$ and $s \leq P \in Ker(\beta)$ we get $s \otimes \beta = \bot$, so it suffices to show that $s \otimes \alpha \leq \Box_A \Box_B P$. By adjunction $f_B(f_A(s \otimes \alpha)) \leq P$. By eq(1) we get $f_A(s \otimes \alpha) \leq f_A(s) \otimes f_A(\alpha)$, order preservation of f_B will give us

$$f_B(f_A(s \otimes \alpha)) \leq f_B(f_A(s) \otimes f_A(\alpha)) \leq f_B(f_A(s)) \otimes f_B(f_A(\alpha)).$$

Now it suffices to show

$$f_B(f_A(s)) \otimes f_B(f_A(\alpha)) \leq P.$$

Replace the f_A with its values and show $f_B(s) \otimes f_B(\alpha') \leq P$, do the same for f_B and get $(s \vee t) \otimes \alpha' \leq P$, hence $(s \otimes \alpha') \vee (t \otimes \alpha') \leq P$ which is equal to $(s \otimes \alpha') \leq P$ since $t \leq \bar{P} \in Ker(\alpha')$. By the assumption $s \leq P$ we obtain $s \otimes \alpha' \leq P \otimes \alpha'$ which leads to $s \otimes \alpha' \leq P$ because P is a fact.

A non-boolean example. An intuitive example of an epistemic system $(M, Q, f_A)_{A \in \mathcal{A}}$ where refutations are first class citizens rather than announcements is the refutation of theories in scientific practice. Hence the underlying lattice M is naturally non-boolean. Let the elements of the module M be *theories* written in some logical language e.g. DEL; a theory being a consistent set of sentences closed under logical deduction. For obvious reasons negating a theory $th \in M$ is in general itself not a theory — algebraically a theory should be conceived as a filter. The join in M is the intersection of the sentences belonging to the corresponding theories while the meet is the closure of their union. The quantale Q consists of *experiments* performed by (groups of) agents in order to check some testable consequences of theories. This experiment might be public or private, and some of the outsiders might be deluded into rejecting, misunderstanding or misinterpreting the outcome.[9] The appearance $f_A^M(m)$ of a theory to an agent can be thought of as the agent's interpretation of the theory m, and similarly the appearance $f_A^Q(q)$ is the agent's interpretation of the outcome of an experiment q. Following Popper's conception, a positive result of an experiment cannot provide a proof of a theory but a negative one provides a falsification of the theory, hence we can refute it. For each such refutation $r \in Q$ we have a kernel $Ker(r) \in M$ which tells us which theories can be refuted, namely those which satisfy $th \otimes r = \bot$.

6 The sequent calculus of epistemic systems

We define the objects of our sequent calculus by mutual induction on two sets, the set of *formulas* denoted as $m \in L_M$ and the set of epistemic programs denoted as $q \in L_Q$, respectively

$$\begin{array}{rcl} m & ::= & \bot \mid \top \mid p \mid s \mid m \wedge m \mid m \vee m \mid \Box_A m \mid f_A(m) \mid [q]m \mid m \otimes q \\ q & ::= & \bot \mid \top \mid 1 \mid \sigma \mid q \bullet q \mid q \vee q \mid f_A(q) \end{array}$$

where A is in the set $\mathcal{A}$ of agents, p is in the set Φ of facts, s is in a set V_M of atomic propositional variables, and σ is in a set V_Q of atomic action variables. We denote by L_M the set of all m-formulas, L_Q the set of all q-formulas, and $\mathcal{A}$ the set of agents. We have two kinds of sequents, M-sequents $\Gamma \vdash_M \delta$ where $\Gamma \in (L_M \cup L_Q \cup \mathcal{A})^*$ and $\delta \in L_M \cup L_Q$, and Q-sequents $\Gamma \vdash_Q \delta$ where $\Gamma \in (L_Q \cup \mathcal{A})^*$ and $\delta \in L_Q$. To describe what these sequents mean, we extend the notation to two operations

$$- \odot - : L_M \times (L_M \cup L_Q \cup \mathcal{A}) \to L_M \cup L_Q \quad \text{and} \quad - \odot - : L_Q \times (L_Q \cup \mathcal{A}) \to L_Q$$

by putting $q \odot q' := q \bullet q'$, $m \odot A := f_A(m)$, $q \odot A := f_A(q)$, $m \odot q := m \otimes q$, and $m \odot m' := m \wedge m'$. For a sequent

$$\Gamma = (\gamma_1, \cdots, \gamma_n) \in (L_M \cup L_Q \cup \mathcal{A})^* \cup (L_Q \cup \mathcal{A})^*$$

we put $\bigodot \Gamma := ((((\sharp \odot \gamma_1) \odot \gamma_2) \odot \gamma_3) \cdots) \odot \gamma_n$, where $\sharp$ is the top element of M for M-sequents, and the unit element of Q for Q-sequents.[10] Obviously we have

$$\Gamma \in (L_M \cup L_Q \cup \mathcal{A})^* \Rightarrow \bigodot \Gamma \in L_M \quad \text{and} \quad \Gamma \in (L_Q \cup \mathcal{A})^* \Rightarrow \bigodot \Gamma \in L_Q\,.$$

Define a *satisfaction relation* $\models$ on L_M as $m \models m' \Leftrightarrow m \leq m'$, similarly on L_Q we define $q \models q' \Leftrightarrow q \leq q'$, and finally on both as[11]

$$m \models q \ \Leftrightarrow \ (m, q) \text{ coherent and } \ q \neq \bot\,.$$

Now a sequent $\Gamma \vdash \delta$ (for either $\vdash_M$ or $\vdash_Q$) is said to be *valid* iff $\bigodot \Gamma \models \delta$. We also allow sequents with empty consequents, denoted as $\Gamma \vdash$. We interpret such a sequent as being equivalent to $\Gamma \vdash \bot$, or in other words $\bigodot \Gamma = \bot$.

[9] E.g. arguments for Darwinism such as the discovery of fossils are interpreted by creationists as "the fossils have been put in place by God".

[10] Note that the top element of M is the unit for $\bigodot$ on M (i.e. $\wedge$) and that the unit element of Q (i.e. 1) is the unit for $\bigodot$ on Q (i.e. $\bullet$)

[11] For the definition of coherence refer to definition 4.4

The meaning of a sequent. To provide the reader with a way to "read" our sequents, we can express the *intuitive meaning* of a sequent $\Gamma \vdash \delta$ in the following inductive manner:

- $A, \Gamma \vdash_M \delta$ means that agent A knows, or believes, that $\Gamma \vdash_M \delta$ holds. So this captures features of A's own reasoning: the sequent $\Gamma \vdash_M \delta$ is accepted by A as a valid argument.
- $q, \Gamma \vdash_M \delta$ means that, after action q happens, the sequent $\Gamma \vdash_M \delta$ will hold.
- $m, \Gamma \vdash_M \delta$ means that, in context m (i.e. in any situation in which m is true), the sequent $\Gamma \vdash_M \delta$ must hold.
- $A, \Gamma \vdash_Q \delta$ means that $\Gamma \vdash \delta$ is a tautology which implies that each agent (with no assumption) knows all the tautologies (i.e. the necessitation rule of classical modal logic).
- Finally, $q, \Gamma \vdash_Q \delta$ simply means that the sequent $q \bullet \Gamma \vdash_Q \delta$ holds.

For instance, the sequent $m, A, q, B, m' \vdash_M m''$ can be read as: in context m, agent A believes that after action q agent B will believe that, in context m', proposition m'' must hold .

This reading shows that our sequent calculus expresses two forms of resource sensitivity. One is the use-once form of linear logic [13] that comes from the quantale structure on epistemic programs. This, as will be seen later, is encoded in the Lambek calculus rules on Q-sequents. One could call these *dynamic resources* . The other form deals with *epistemic resources* : the resources available to each agent that enable him to reason in a certain way (i.e. to deduct a result from some assumptions). These resources are encoded in the way the context appears to the agent in sequents, for instance Γ in the sequent $\Gamma, A, \Gamma' \vdash_M \delta$ is the context and hence the $f_A(\Gamma)$ is the resource that enables agent A to do the $\Gamma' \vdash_M \delta$ reasoning. Note that $\Gamma' \vdash_M \delta$ might not be a valid sequent in the context Γ, but it is valid in the context given by Γ's appearance to agent A. To summerize, in our setting not only propositions, but also actions and agents are treated as resources (available or not for other actions or for reasoning of other agents).

Sequent rules. The axioms for identity and $\bot$ and $\top$ are the same as in any sequent calculus. The **operational rules for M-sequents** are

$$\frac{\Gamma, q \vdash_M m}{\Gamma \vdash_M [q]m} \qquad \frac{m, \Gamma \vdash_M \delta}{[q]m, q, \Gamma \vdash_M \delta} \qquad \frac{\Gamma, A \vdash_M m}{\Gamma \vdash_M \Box_A m} \qquad \frac{m, \Gamma \vdash_M \delta}{\Box_A m, A, \Gamma \vdash_M \delta}$$

$$\frac{\Gamma \vdash_M m}{\Gamma, A \vdash_M f_A(m)} \qquad \frac{m, A, \Gamma \vdash_M \delta}{f_A(m), \Gamma \vdash_M \delta}$$

$$\frac{\Gamma, m, m', \Gamma' \vdash_M \delta}{\Gamma, m \wedge m', \Gamma' \vdash_M \delta} \qquad \frac{\Gamma \vdash_M m \qquad \Gamma \vdash_M m'}{\Gamma \vdash_M m \wedge m'} \qquad \frac{\Gamma, q, q', \Gamma' \vdash_M \delta}{\Gamma, q \bullet q', \Gamma' \vdash_M \delta} \qquad \frac{\Gamma \vdash_M q \qquad \Gamma' \vdash_M q'}{\Gamma, \Gamma' \vdash_M q \bullet q'}$$

$$\frac{\Gamma \vdash_M \delta}{\Gamma \vdash_M \delta \vee \delta'} \qquad \frac{\Gamma \vdash_M \delta'}{\Gamma \vdash_M \delta \vee \delta'} \qquad \frac{m, \Gamma_Q \vdash_M \delta \qquad m', \Gamma_Q \vdash_M \delta}{m \vee m', \Gamma_Q \vdash_M \delta} \qquad \frac{\Gamma, q, \Gamma' \vdash_M \delta \qquad \Gamma, q', \Gamma' \vdash_M \delta}{\Gamma, q \vee q', \Gamma' \vdash_M \delta}$$

$$\frac{\Gamma, \Gamma_A \vdash_M m \qquad \Gamma_Q, \Gamma_A \vdash_Q q}{\Gamma, \Gamma_Q, \Gamma_A \vdash_M m \otimes q} \qquad\qquad \frac{\Gamma_M, q, \Gamma' \vdash_M \delta}{\Gamma_M \otimes q, \Gamma' \vdash_M \delta}$$

where $\Gamma_M \in L_M^*, \Gamma_Q \in L_Q^*, \Gamma_A \in \mathcal{A}^*, \delta \in L_M \cup L_Q$ and if $\Gamma_M = (m_1, \cdots, m_n)$ then $\Gamma_M \otimes q := (m_1 \otimes q, \cdots, m_n \otimes q)$. The **operational rules for Q-sequents** consist of Lambek calculus rules plus rules for f_A, namely

$$\frac{\Gamma \vdash_Q m}{\Gamma, A \vdash_Q f_A(m)} \qquad\qquad \frac{q, A, \Gamma \vdash_Q \delta}{f_A(q), \Gamma \vdash_Q \delta}\,.$$

As **structural rules** we have M-Contraction, two M-Weakenings and M-Exchange, respectively

$$\frac{\Gamma, m, m, \Gamma' \vdash_M \delta}{\Gamma, m, \Gamma' \vdash_M \delta} \qquad\qquad \frac{\Gamma \vdash_M \delta}{\Gamma', \Gamma \vdash_M \delta} \qquad \frac{\Gamma, \Gamma' \vdash_M \delta}{\Gamma, m, \Gamma' \vdash_M \delta} \qquad\qquad \frac{\Gamma, m, m', \Gamma'' \vdash_M \delta}{\Gamma, m', m, \Gamma'' \vdash_M \delta}\,,$$

two rules expressing *Invariance of facts (under epistemic actions)* (rules which can be seen as “Action Weakening’ and “Action Strengthening” in M-sequents)

$$\frac{\Gamma \vdash_M P}{\Gamma, q \vdash_M P} \qquad\qquad \frac{\Gamma, q \vdash_M P}{\Gamma \vdash_M P}\,,$$

where $P \in \Phi$ (the set of facts), and finally several restricted versions of the Cut Rule: Propositional Cut and Action Cut in M-sequents, as well as Action Cut in Q-sequents, respectively

$$\frac{\Gamma \vdash_M m \qquad m, \Gamma' \vdash_M \delta}{\Gamma, \Gamma' \vdash_M \delta} \qquad\qquad \frac{\Gamma, q \vdash_M \qquad \Gamma \vdash_M q}{\Gamma \vdash_M} \qquad\qquad \frac{\Gamma \vdash_Q q \qquad q, \Gamma' \vdash_Q \delta}{\Gamma, \Gamma' \vdash_Q \delta}\,.$$

We end with **encoding rules for concrete epistemic systems**, where $M = \mathcal{P}(S)$ and $Q = \mathcal{P}(\Sigma)$. To encode a state model $S = \{s_i\}_{i \in I}$ with the precondition $\Phi_i := \mu(s_i) \subseteq \Phi$ and the appearance maps are defined by $S_A^i := f_A(s_i) \subseteq S$, we add the axioms $s_i \vdash_M P$ and $s_i, Q \vdash_M$, for all $P \in \Phi_i$ and $Q \in \Phi \setminus \Phi_i$, and the rules

$$\frac{\{t, \Gamma \vdash_M \delta\}_{t \in S_A^i}}{s_i, A, \Gamma \vdash_M \delta} \qquad\qquad \frac{s_i, A, \Gamma \vdash_M \delta}{t, \Gamma \vdash_M \delta} \text{ for each } t \in S_A^i$$

To encode an action model $\Sigma = \{\sigma_i\}_{i \in I}$ with $\mu(\sigma_i) = m_i \in M$ and $f_A(\sigma_i) = \Sigma_A^i \subseteq \Sigma$ we add the axiom $m_i \vdash_M \sigma_i$ and rules

$$\frac{\Gamma, m_i \vdash_M}{\Gamma, \sigma_i \vdash_M} \qquad\qquad \frac{\Gamma \vdash_M \sigma_i}{\Gamma \vdash_M m_i} \qquad\qquad \frac{\{\tau, \Gamma \vdash_Q \delta\}_{\tau \in \Sigma_A^i}}{\sigma_i, A, \Gamma \vdash_Q \delta}$$

$$\frac{\sigma_i, A, \Gamma \vdash_Q \delta}{\tau, \Gamma \vdash_Q \delta} \text{ for each } \tau \in \Sigma_A^i \qquad\qquad \frac{\{\Gamma, m_i, A, \tau \vdash_M \delta\}_{\tau \in \Sigma_A^i}}{\Gamma, \sigma_i, A \vdash_M \delta}.$$

We can add one additional rule specifically for strong epistemic systems:[12]

$$\frac{\Gamma, q, A \vdash_M m \qquad \Gamma \vdash_M q}{\Gamma, A, f_A(q) \vdash_M m}$$

[12] For the definition of strong epistemic system refer to definition 4.5

Soundness. One verifies that these rules are sound with regard to the model of section 4. Concerns about completeness constitute on-going work.

7 Conclusion and elaborations

We have developed an algebraic axiomatics in terms of a simple mathematical object: a sup-lattice M, which encodes states, epistemic propositions as well as facts; a quantale Q (acting on M) which encodes update by epistemic programs; and a family of endomorphisms of the $(M, Q, \bigvee_M, \bigvee_Q, \otimes, \bullet, 1)$-structure encoding the agents in terms of their epistemic modalities. From this structure many useful other modalities arise, including dynamic modalities and residuals. This algebraic axiomatics generalizes Dynamic Epistemic Logic to non-boolean settings, while still capturing the same concepts. Furthermore it provides an algebraic way of dealing with epistemic scenarios such as the muddy children puzzle. We list some possible further elaborations on this line of thought.

- We would like to have a more refined version of Theorem 4.6, one which exposes alternative but concrete variations on Dynamic Epistemic Logic which it then axiomatically classifies.

- Also of interest would be investigations towards blending this algebraic approach with coalgebraic epistemic features which are currently intensively studied e.g. [3].

- Part of the motivation of this work was a marriage of epistemics and resource-sensitivity [19]. Although we only contributed in a very limited fashion to such a project in this paper (quantales provide a semantics for non-commutative linear logic and hence we obtain a linear structure on epistemic programs) it would be interesting to obtain a better handle on resources in our model.

- Each system (M, Q) can be equivalently represented as a Q-enriched category [24]. This would allow a passage from a dynamic epistemic theory of programs to one about program transformations when substituting the quantale by a one-object biclosed bicategory — a quantale is a one-object biclosed bicategory which is locally thin. The two-cells in the bicategory would then encode the program transformations.

Acknowledgements

M. S. thanks Samson Abramsky and Oxford University Computer Laboratory for their hospitality and thanks Mathieu Marion for logistic support. B. C. and M. S. thank Isar Stubbe for valuable discussions on the theory of quantales both recently and in the past.

References

[1] S. Abramsky and S. Vickers, ‘Quantales, observational logic and process semantics’, *Mathematical Structures in Computer Science* **3**, 161-227, 1993.

[2] A. Baltag, ‘Logics for communication: reasoning about information flow in dialogue games’, *Second North American Summer School in Logic, Language, and Information*, `http://www.indiana.edu/~nasslli/program.html`, Indiana University, 2003.

[3] A. Baltag, ‘A coalgebraic semantics for epistemic programs’, Proceedings of *Coalgebraic Methods in Computer Science 03*, 2003.

[4] A. Baltag, and L.S. Moss, ‘Logics for epistemic programs’, *Synthese* **139**, 2004.

[5] A. Baltag, L.S. Moss and S. Solecki, ‘The logic of public announcements, common knowledge and private suspicions’, CWI Technical Report SEN-R9922, 1999.

[6] B. Coecke, D.J. Moore and I. Stubbe, ‘Quantaloids describing causation and propagation of physical properties’, *Foundations of Physics Letters* **14**, 133-145, 2001.

[7] E.W. Dijkstra, *A Discipline of Programming*, Prentice-Hall, 1976.

[8] R. Fagin, J.Y. Halpern, Y. Moses and M.Y. Vardi, *Reasoning about Knowledge*, MIT Press, 1995.

[9] G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M.W. Mislove and D.S. Scott, *A Compendium of Continuous Lattices*, Springer-Verlag, 1980.

[10] J. Gerbrandy, ‘Dynamic Epistemic Logic’, in L.S. Moss, et al (eds.) *Logic, Language, and Information* **2**, Stanford University, CSLI Publication, 1999.

[11] J. Gerbrandy, *Bisimulation on Planet Kripke*, Ph.D. dissertation, University of Amesterdam, 1999.

[12] J. Gerbrandy, and W. Groenveld, ‘Reasoning about information change’, *Journal of Logic, Language, and Information* **6**, 1997.

[13] J-Y. Girard, ’Linear logic’, *Theoretical Computer Science* **50**,1-102, 1987.

[14] D. Harel, D. Kozen and J. Tiuryn, *Dynamic Logic*, MIT Press, 2000.

[15] C.A.R. Hoare and Jifeng, HE, ‘The weakest prespecification’, *Information Processing Letters* **24**, 127-132, 1987.

[16] P.T. Johnstone, *Stone Spaces*, Cambridge University Press, 1982.

[17] A. Joyal and M. Tierney, ‘An extension of the Galois theory of Grothendieck’, *Memoirs of the American Mathematical Society* **309**, 1984.

[18] J. Lambek, ‘The mathematics of sentence structure’, *American Mathematics Monthly* **65**, 154-169, 1958.

[19] M. Marion and M. Sadrzadeh, ‘Reasoning about knowledge in linear logic: modalities and complexity’, D. Gabbay, S. Rahman, J.M. Torres and J.-P. Van Bendegem (eds.), *Logic, Epistemology, and the Unity of Science*, Kluwer, 2004.

[20] C.J. Mulvey, &, *Supplemento ai Rendiconti del Circolo Matematico di Palermo* **II**, 99-104, 1986.

[21] J. Plaza, ‘Logics of public communications’, *Proceedings of 4th International Symposium on Methodologies for Intelligent Systems*, 1989.

[22] P. Resende, ‘Quantales and observational semantics’, B. Coecke, D.J. Moore and A. Wilce (eds.), *Current Research in Operational Quantum Logic*, Kluwer, 263-288, 2000.

[23] K.I. Rosenthal, *Quantales and their Applications*, Pitman Research Notes in Mathematics Series **234**, Longman, 1990.

[24] I. Stubbe, *Categorical Structures Enriched in a Quantaloid: Categories and Semicategories*, Ph.D. Thesis, Université Catholique de Louvain, 2003.

A Semantic Approach for Reasoning about Security Protocols (extended abstract)

Arjen Hommersom[1,*], John-Jules Meyer[2], Erik de Vink[3,4]

[1] Nijmegen Institute of Information and Computing Sciences, University of Nijmegen
[2] Institute of Information and Computer Science, University of Utrecht
[3] Department of Mathematics and Computer Science, Technische Universiteit Eindhoven
[4] Leiden Institute of Advanced Computer Science, Leiden University

Abstract We present a model-theoretic approach for reasoning about security protocols, applying recent insights from dynamic epistemic logics. This enables us to describe exactly the subsequent epistemic states of the agents participating in the protocol, using Kripke models and transitions between these based on updates of the agents' beliefs associated with steps in the protocol. As a case study we will consider the SRA Three Pass protocol.

1 Introduction

In today's world of e-commerce and the Internet, the role of security protocols is getting increasingly important. The design of these security protocols is difficult and error-prone [Sch00, And01], which makes (automatic) verification of protocols of crucial importance. Since the late eighties, one line of research, amongst others, for reasoning about security protocols is based on the use of the so-called BAN logic, proposed by Burrows, Abadi and Needham in [BAN90]. This is an epistemic logic augmented by constructs that are relevant for reasoning about security, such as the property of having the disposal of a cryptographic key to be able to decode a message and therefore to know its contents. Although many useful results having been reported (e.g., [KN98, AHV01, SW02]), due to their complexity and their semantic underpinning the use of BAN logics to prove the correctness of security protocols has so far not been very successful (cf. [AT91, BM97, WK96, SC01]).

In this paper we will apply insights from dynamic epistemic logics as recently developed by Gerbrandy [Ger97, Ger99], Baltag [BMS99, Bal00], Van Ditmarsch [Dit00, Dit01], and Kooi [Koo03]. Moreover, contrary to the traditional BAN logic approach, our approach is semantic or model-theoretic. We use Kripke models to represent the epistemic state of the agents involved in a protocol, similarly to the S5 preserving approach of Van Ditmarsch to analyze certain kinds of games involving knowledge. From Baltag's action models we import the idea to describe belief updates of the agents by semantic operators transforming the Kripke models at hand by copying and deleting parts of these models, although we use traditional Kripke models rather than Baltag's action models. To this end we need also operators for unfolding

[*] Corresponding author: `arjenh@cs.kun.nl`

[25] J. Van Benthem, ‘Logic in action’, *Journal of Philosophical Logic* **20**, 225-263, 1989.

[26] F. Wolter and M. Zakharyaschev, ‘The relation between intuitionistic and classical modal logics, *Algebra and logic* **36**, 73-92, 1997.

models, which is in its turn inspired by Gerbrandy's work on possibilities. The difference being that in our approach only *partial* unfolding is called for. We furthermore propose a language to express belief updates in the context of security protocols as well as properties of these updates, and give a semantics of this language in terms of the models mentioned and the operators on them. Since our approach is model-theoretic, we believe that it may serve as a starting point for the automatic verification of (properties of) security protocols.

As a case study illustrating our approach we will consider the so-called SRA Three Pass protocol and prove a property of it. It is not our intention to prove that the protocol is completely secure (as it is not in full generality), but we will prove that if the agents participating in the protocol are honest, then an intruder watching the communication does not learn anything about the plain-text messages in a single run. Furthermore we show what the intruder is able to learn about the agents participating.

2 Preliminaries

In this section we briefly discuss some preliminaries and background regarding the updates we will handle and the epistemic model we will use. First, we define objective formulas and so-called o-seriality.

Definition 2.1. Fix a set $\mathcal{P}$ of propositional variables. The class of objective formulas is the smallest class such that:

- all propositional variables or atoms $p \in \mathcal{P}$ are objective;
- if ϕ is objective, then $\neg\phi$ is objective;
- if ϕ_1 and ϕ_2 are objective, then $\phi_1 \wedge \phi_2$ is objective.

So, objective formulas do not involve beliefs. For our purposes it is important that every agent distinguishes a world with the same 'objective' information. This leads to the notion of an o-serial model.

Definition 2.2. A model $M = \langle S, \pi, R_1, ..., R_m\rangle$ is *o-serial* iff for all i, $1 \leq i \leq m$, and $w \in S$, there exists $v \in S$ such that $(w, v) \in R_i$ and for all objective formulas ϕ it holds that $(M, w) \models \phi \Leftrightarrow (M, v) \models \phi$.

We use a, b, c, etc. as typical agents, taken from a class $\mathcal{A}$. We use the notation $\{x\}_{k_a}$ to denote a message x encrypted with the cryptographic key k_a of agent a. Furthermore, B is used as a doxastic modal operator. For example, $B_a\phi$ should be read as 'a believes ϕ'. We interpret formulas on standard Kripke models $(M, s) = (\langle S, \pi, R_1, ..., R_m\rangle, s)$, where $(M, s) \models B_i\phi$ iff $\forall t \in S\colon R_i(s, t) \rightarrow (M, t) \models \phi$.

We require the relations R_i to be o-serial, transitive and euclidean. This yields a class of models that we will call Kt45, a proper subset of the class of models of the well-known doxastic logic *KD45*. The lower case t refers to the axiom

$$B_i\phi \Rightarrow \phi, \tag{t}$$

for every objective ϕ. The system Kt45 is sound with respect to the class of o-serial, transitive and euclidian models [Hom03]. We will show that the operations we introduce preserve Kt45.

The point is that in worlds of Kt45 models, we cannot both have $B_i\phi$ and $B_i\neg\phi$, for an objective formula ϕ. This is reasonable from our assumption that agents are conscious about the protocol. Therefore, they will not infer objective contradictions. This objectivity is captured locally for each state. As a consequence, the operations that we introduce can restrict the set of states without destroying objective information.

For the analysis of security protocols below, we assume that we are omniscient about the values of the variables in different runs of a protocol. For example, the program variable p in a protocol run has the value $[\![p]\!]$. In the real world it is, obviously, always true that $p = [\![p]\!]$. However, it is cumbersome to keep track of what is the real world in the operations on Kripke structures that we employ below. Therefore, we assume that an interpretation $[\![\cdot]\!]$ is given, that provides the 'real' values of the program variables when needed. It might very well be the case that $p \neq [\![p]\!]$ in a certain state. From now on, we will abbreviate $p = [\![p]\!]$ to p on (thus transforming a program expression into a propositional variable). Similarly, $\neg p$ is an abbreviation of $p \neq [\![p]\!]$. For example, agent a that learns $B_b p \vee B_b \neg p$, learns that agent b has assigned a value to the program variable p.

The types of updates we consider are (i) public announcement of a variable, (ii) the private learning of a variable and (iii) the private learning about the knowledge of other agents.

The first type of update typically runs as follows: In an open network agent a sends a message to agent b. From a security perspective, it is customary [DY83] to assume that all agents in the network can read this message too. However, also in open networks private learning, the second type of update, can take place. For example, agent b receives a message $\{x\}_k$ from agent a. Here $\{x\}_k$ denotes a message with content x encrypted with the (symmetric) key k. If b possesses the key k, then b privately learns the message content x (assuming that the key k is shared among a and b). The final type of update, learning about knowledge of others, is probably the most interesting. It is realistic to assume that the steps in a protocol run are known to all agents. Therefore, observing that an agent receives a message will increase the knowledge of the other agents. For example, if agent a sends a message $\{x\}_k$ to agent b, then agent c learns that b has learned the information contained in the message $\{x\}_k$, but typically, c does not learn x if c does not possess the key k.

Stronger types of updates we do not consider here. For example, we will not update the beliefs of an honest agent such that it learns that an intruder has learned about others. In the present paper, we restrict ourselves to updating beliefs about objective formulas and beliefs about objective formulas.

3 Update constructions

In this section we describes various types of updates in detail. We will start by defining an update for propositions in subsection 3.1. In subsection 3.2 we will define a belief update for agents that learn something about the belief of others. We do this in two slightly different ways by varying in the functions that describe a side-effect for an agent.

3.1 Objective updates

The belief update of objective formulas we will use is based on [RHM02, BMS99]. The construction works as follows: We will make copies of the states of the model such that the *old* worlds in *old*(S) correspond to the information in the original model and the *new* worlds in *new*(S) correspond to the new information.

Definition 3.1. Let a world $(M, w) = (\langle S, \pi, R_1, ..., R_m\rangle, w)$, a group of agents $\mathcal{B}$, and an objective formula ϕ be given. Then $\textsc{expand}_{(\phi,\mathcal{B})}(M, w) = (\langle S', \pi', R'_1, ..., R'_m\rangle, w')$, where

- $S' = \{\mathit{new}(s) \mid (M, s) \models \phi\} \cup \mathit{old}(S)$
- $w' = \mathit{new}(w)$
- for all $p \in \mathcal{P}$: $\pi'(\mathit{old}(u))(p) = \pi'(\mathit{new}(u))(p) = \pi(u, p)$
- for $1 \leq i \leq m$ the relation R'_i on S' is minimal such that:

$$\begin{array}{lcll} R'_i(\mathit{old}(u), \mathit{old}(v)) & \Leftrightarrow & R_i(u, v) & \\ R'_a(\mathit{new}(u), \mathit{new}(v)) & \Leftrightarrow & R_a(u, v) \wedge (M, v) \models \phi & \text{if } a \in \mathcal{B} \\ R'_b(\mathit{new}(u), \mathit{old}(v)) & \Leftrightarrow & R_b(u, v) & \text{if } b \notin \mathcal{B} \end{array}$$

The following example shows how this works on a concrete model.

Example 3.1. Consider the model (M, s) in Figure 1 where $\pi(s)(p) = \texttt{true}$ and $\pi(t)(p) = \texttt{false}$. The operation we execute is that b learns p. This results in the model (M, u) in Figure 2 where $\pi(u)(p) = \pi(v)(p) = \texttt{true}$ and $\pi(w)(p) = \texttt{false}$ and $\mathit{new}(s) = u, \mathit{old}(s) = v$ and $\mathit{old}(t) = w$. The world $\mathit{new}(t)$ is unreachable from the actual world and is therefore omitted from the figure. (In fact, in all figures in this paper, we will omit the unreachable worlds.)

We can see that the belief of agent a has not changed: it still considers its old worlds possible. The belief of agent b however, has changed. It now only considers the state u possible where p holds.

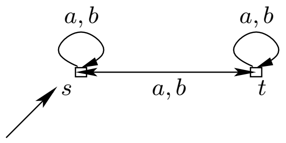


Figure 1: (M, s)

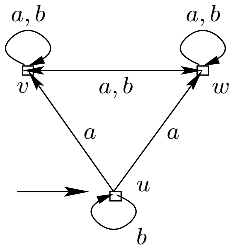


Figure 2: (M, u)

The update $\textsc{expand}_{(\phi,\mathcal{B})}$ is based on a formula ϕ and a set of agents $\mathcal{B}$. Roorda *et al.* [RHM02] have given a characterization of the formulas that are altered by such an operation with a single learning agent. Here we extend the definition for multi-agent purposes.

Definition 3.2. An update function $(\cdot)[\phi, \mathcal{B}]$ is called proper if

$$\begin{array}{lcl} (M, w)[\phi, \mathcal{B}] \models p & \Leftrightarrow & (M, w) \models p \\ (M, w)[\phi, \mathcal{B}] \models \alpha \wedge \beta & \Leftrightarrow & (M, w)[\phi, \mathcal{B}] \models \alpha \text{ and } (M, w)[\phi, \mathcal{B}] \models \beta \\ (M, w)[\phi, \mathcal{B}] \models \neg\alpha & \Leftrightarrow & (M, w)[\phi, \mathcal{B}] \not\models \alpha \\ (M, w)[\phi, \mathcal{B}] \models B_a\alpha & \Leftrightarrow & (M, w) \models B_a\alpha \;\text{ if } a \notin \mathcal{B} \\ (M, w)[\phi, \mathcal{B}] \models B_b\alpha & \Leftrightarrow & \forall u\colon ((B_b(w, u) \wedge (M, u) \models \phi) \Rightarrow (M, u)[\phi, \mathcal{B}] \models \alpha) \;\text{ if } b \in \mathcal{B} \end{array}$$

Following Roorda *et al.* we have that $\text{EXPAND}_{(\phi,\mathcal{B})}$ is proper. Moreover, $\text{EXPAND}_{(\phi,\mathcal{B})}$ is uniquely characterized by Definition 3.2 upto elementary equivalence, i.e. if $(\cdot)[\phi,\mathcal{B}]$ is a proper update function, then $(M,w)[\phi,\mathcal{B}]$ and $\text{EXPAND}_{(\phi,\mathcal{B})}(M,w)$ are elementary equivalent. We collect the following properties of $\text{EXPAND}_{(\phi,\mathcal{B})}$.

Lemma 3.1.

- *For all ϕ it holds that $(M,w) \models \phi \Rightarrow \text{EXPAND}_{(\phi,\mathcal{B})}(M,w) \models B_{\mathcal{B}}\phi$.*
- *If (M,w) satisfies the Kt45 properties and ϕ is objective, then $\text{EXPAND}_{(\phi,\mathcal{B})}(M,w)$ satisfies the Kt45 properties as well.*
- $\text{EXPAND}_{(\psi,\mathcal{C})}(\text{EXPAND}_{(\phi,\mathcal{B})}(M,w))$ *and* $\text{EXPAND}_{(\phi,\mathcal{B})}(\text{EXPAND}_{(\psi,\mathcal{C})}(M,w))$ *are bisimilar.*

Note that in Lemma 3.1 ϕ ranges over arbitrary formulas, including non-objective ones. However, for a non-objective formula $B_i\phi$, it can happen that, unintended, an agent increases the objective knowledge encapsulated by the formula ϕ. This is illustrated by the next example.

Example 3.2. Suppose we are interested in agent a learning the formula $B_bp \vee B_b\neg p$, but not p itself. Consider the Kripke model (M,s) in Figure 3, where $\pi(s)(p) = \pi(u)(p) = \texttt{true}$ and $\pi(t)(p) = \texttt{false}$. This models the state where b knows that p is true. Agent a does not know p or $\neg p$, and it does not know if b knows p.

If we apply the definition of the belief expansion function, it results in the model (M,v) from Figure 4, where $\pi(v)(p) = \pi(s)(p) = \pi(u)(p) = \texttt{true}$ and $\pi(t)(p) = \texttt{false}$. The reason that it turns out like this, is because the only state where $B_bp \vee B_b\neg p$ holds, is the state s. Thus, all the other states have no corresponding *new* states. Figure 4 illustrates that a has learned $B_bp \vee B_b\neg p$, but also that a has learned p itself.

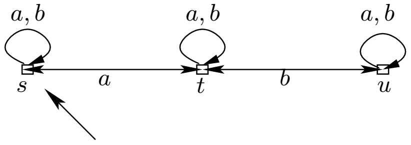


Figure 3: (M,s)

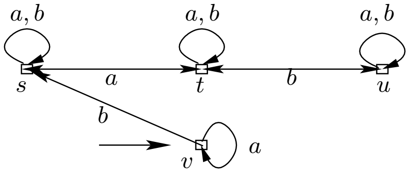


Figure 4: (M,v)

In the next subsection we will define a side-effect function such that a will learn about others, but does not learn any objective formulas itself.

3.2 Side-effects

The main reason that an update of $B_bp \vee B_b\neg p$ for agent a fails, is that it actually deletes the wrong arrows. The belief expansion function deletes arrows of a to gain the states that satisfy the updating formula. This is not what we intend. We want a to keep all the states it considers possible, but at the same time we would like to update all the possible states of a such that the formula $B_bp \vee B_b\neg p$ holds in these states. Moreover, we do not want to change the knowledge of other agents. In this section we define the functions that accomplish these requirements.

A technical obstacle is that states can be shared among agents. It is obvious that if we change a state with the intention to change the belief of one agent, then the belief of the other agents that consider this state possible, is changed as well. Therefore, the first thing to do, is separate the states of learning agents from the states of agents that do not learn. This procedure will be called *unfolding*. The functions $new_{\mathcal{B}}$ and *orig* are generalizations of *new* and *old* from the previous section, but the function *orig* is only defined on the point of the model (the actual world).

Definition 3.3. Given a model (M, w) with $M = \langle S, \pi, R_1, ..., R_m \rangle$, a partitioning $\mathcal{X}$ of $\mathcal{A}$, we define a function $\text{UNFOLD}_{\mathcal{X}}(M, w) = (\langle S', \pi', R'_1, ..., R'_m \rangle, w')$, where

- $S' = (\bigcup_{\mathcal{B} \in \mathcal{X}} new_{\mathcal{B}}(S)) \cup orig(w)$
- $w' = orig(w)$
- $\pi'(new_{\mathcal{B}}(v))(p) = \pi(v)(p)$, $\pi'(w')(p) = \texttt{true}$ for all $p \in \mathcal{P}, \mathcal{B} \in \mathcal{X}$
- the relation R'_i on S' is minimal such that

  $$\begin{array}{lcl} R'_i(new_{\mathcal{B}}(u), new_{\mathcal{C}}(v)) & \Leftrightarrow & R_i(u, v) \wedge (\mathcal{B} = \mathcal{C}) \\ R'_i(orig(w), new_{\mathcal{B}}(u)) & \Leftrightarrow & R_i(w, u) \wedge i \in \mathcal{B} \end{array}$$

  where $\mathcal{B}, \mathcal{C}$ range over $\mathcal{X}$.

So for every group of agents $\mathcal{B}$ there is copy of the original states (viz. $new_{\mathcal{B}}(s)$ for every $s \in S$). This function does indeed preserve our Kt45 properties and it models the same knowledge, which is captured in the following lemma.

Lemma 3.2.

(a) If (M, w) is a Kt45 model and $\mathcal{X}$ a partition, then $\text{UNFOLD}_{\mathcal{X}}(M, w)$ is a Kt45 model.

(b) For every model (M, w) and partition $\mathcal{X}$, it holds that (M, w) and $\text{UNFOLD}_{\mathcal{X}}(M, w)$ are bisimilar.

Example 3.3 *(unfold).* Consider the Kripke model (M, s) in Figure 5 with $\pi(s)(p) = \pi(u)(p) = \texttt{true}$, $\pi(t)(p) = \texttt{false}$. So, b knows that p is true, while a does not. Furthermore, a does not know if b knows p. Now the operation we perform is $\text{UNFOLD}_{\{\{a\},\{b\}\}}(M, s)$ which results in the model (M', s) in Figure 6.

So we have split the knowledge of a and b. The state s is the original state, the primed states model a's knowledge and the double primed states model b's knowledge. So the upper half of the model represents the knowledge of a, and the lower half represents the knowledge of b. Note that no states are shared, in particular because the point of the model is not reflexive.

Now we give some preparatory definitions. First we define the notion of a submodel.

Definition 3.4. A model $M = \langle S, \pi, R_1, ..., R_m \rangle$ is a submodel of $M' = \langle S', \pi', R'_1, ..., R'_m \rangle$, written as $M \sqsubseteq M'$, iff $S \subseteq S'$, $\pi(s)(p) = \pi'(s)(p)$ for all $s \in S$, $p \in \mathcal{P}$ and $R_i \subseteq R'_i$.

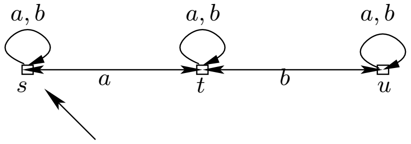


Figure 5: (M, s)

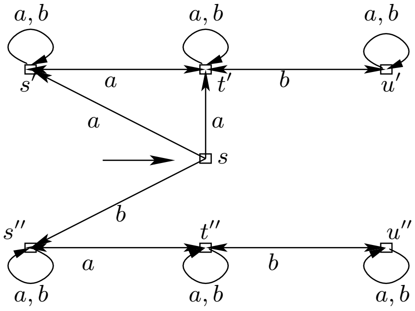


Figure 6: (M', s)

Next, we construct a submodel that represent the knowledge of an agent a.

Definition 3.5. Given a model $(M, w) = (\langle S, \pi, R_1, ..., R_m \rangle, w) = \text{UNFOLD}_{\{\{a\}, \mathcal{B}\}}(M', w')$ such that $\{\{a\}, \mathcal{B}\}$ is a partition of $\mathcal{A}$, for some (M', w'), define the a-submodel $\text{SUB}_a(M) = \langle S', \pi', R'_1, ..., R'_m \rangle$ where

- $s \in S' \Leftrightarrow new_a(s) \vee orig(s)$,
- for all $p \in \mathcal{P}$ it holds that $\pi'(new_a(s))(p) = \pi'(orig(s))(s) = \pi(s)(p)$,
- $R'_i(s, t) \Leftrightarrow R_i(s, t) \wedge s, t \in S'$.

Clearly an a-submodel is a submodel in the sense of Definition 3.4. The restmodel is the part of the model that is complements the submodel with respect to the accessibility relation.

Definition 3.6. Given a model $(M, w) = \langle S, \pi, R_1, ..., R_m \rangle$ and a submodel $N = \langle S'', \pi'', R''_1, ..., R''_m \rangle$ of M, define the restmodel $\text{REST}_N(M) = \langle S', \pi', R'_1, ..., R'_m \rangle$ where

- $s \in S' \Leftrightarrow s \in S \wedge \exists (u, v) \in R'_i\colon\ u = s \vee v = s$
- $\pi'(s)(p) = \pi(s)(p)$ for all $p \in \mathcal{P}$
- $R'_i(s, t) \Leftrightarrow R_i(s, t) \wedge \neg R''_i(s, t)$

We can see the a-submodel and restmodel definitions in action by taking the model of example 3.3 and applying the above definitions. See Figure 7. This exactly corresponds to the idea of two submodels that represent knowledge of different agents.

Now we would like to update the belief of some agents. To this end, we want to replace the submodel that represents their belief by a new model. We will apply the following definition.

Definition 3.7. Given a model $N = \langle S, \pi, R_1, ..., R_m \rangle$, a model M, a model N' such that $N \sqsubseteq N' \sqsubseteq M$ with $\text{REST}_{N'}(M) = \langle S', \pi', R'_1, ..., R'_m \rangle$, we define $\text{REPLACE}_{N'}(N, M) = \langle S'', \pi'', R''_1, ..., R''_m \rangle$ where

- $s \in S'' \Leftrightarrow s \in S \vee s \in S'$,
- for all $p \in \mathcal{P}$ it holds that $\pi''(s)(p) = \pi(s)(p)$ for $s \in S''$,

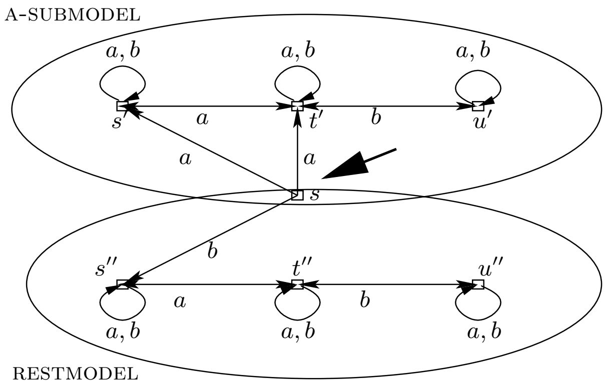


Figure 7: a-submodel outline

- $R''_i(s,t) \Leftrightarrow (s,t) \in R_i \vee (s,t) \in R'_i$.

The idea is that once the belief is completely separated, we can not only safely change the belief of a certain agent, but also preserve the Kt45 properties. The function $\textsc{atomsplit}_{(p,b)}$ removes the arrows of b between states that have a different valuation for p.

Definition 3.8. Given a model $M = \langle S, \pi, R_1, ..., R_m \rangle$, we define a function $\textsc{atomsplit}_{(p,b)}(M) = \langle S', \pi', R'_1, ..., R'_m \rangle$ as follows:

- $s \in S' \Leftrightarrow s \in S$,
- $\pi'(s)(p) = \pi(s)(p)$ for all $p \in \mathcal{P}$,
- $R'_i(s,t) \Leftrightarrow R_i(s,t)$ $(i \neq b)$,
- $R'_b(s,t) \Leftrightarrow R_b(s,t) \wedge \pi(s)(p) = \pi(t)(p)$.

Finally we are in a position to define the actual side-effect function that ties these things together.

Definition 3.9. For a model (M', w') such that $(M', w') = \textsc{unfold}_{\{\{a\}, \mathcal{A}\setminus\{a\}\}}(M, w)$ and $N = \textsc{sub}_a(M')$ we define $\textsc{side-effect}_{(p,a,b)}(M, w) = (\textsc{replace}_N(\textsc{atomsplit}_{p,b}(N), M'), w')$.

Example 3.4. We continue Example 3.3. To the a-submodel of M we now apply $\textsc{atomsplit}_{p,b}$ which results in the model (M'', s) in Figure 8. The arrow (t', u') has disappeared, since $\pi(t')(p) \neq \pi(u')(p)$. Therefore, u' is not reachable anymore, and can be dropped. Notice that a believes $B_b p \vee B_b \neg p$, while a has learned nothing about p itself.

We have the following results.

Lemma 3.3.

(a) If (M, w) is a Kt45-model, then $\textsc{side-effect}_{(p,a,b)}(M, w)$ *is a Kt45-model as well.*

(b) Given a model (M, w), agents a, b, c, d, two propositions $p, q \in \mathcal{P}$, it holds that $\textsc{side-effect}_{(p,c,d)}(\textsc{side-effect}_{(q,a,b)}(M, w))$ *and* $\textsc{side-effect}_{(q,a,b)}(\textsc{side-effect}_{(p,c,d)}(M, w))$ *are bisimilar.*

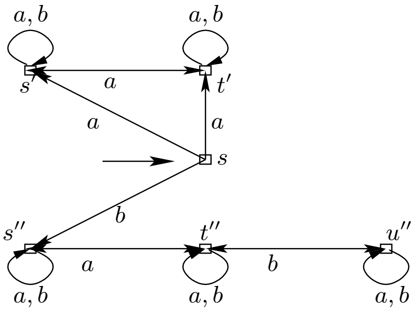


Figure 8: (M'', s)

(c) Given a model (M, w), agents a, b, c, a propositions $p \in \mathcal{P}$ and an objective formula ϕ, it holds that $\text{SIDE-EFFECT}_{(p,c,d)}(\text{EXPAND}_{(\phi,a)}(M, w))$ *and* $\text{EXPAND}_{(\phi,a)}(\text{SIDE-EFFECT}_{(p,c,d)}(M, w))$ *are bisimilar.*

Next, we consider how the formulas are altered by the side-effect function. We will partially answer this by presenting a few interesting formulas that hold in the resulting model. We will look at groups of agents instead of a single agent:

1. the group of agents that learn about other agents, ranged over by a;
2. the group of agents that is learned about, ranged over by b;
3. other agents, ranged over by c.

The fact that the agents a are the only agent that learn at all, is clear. The other agents consider their old worlds possible; their belief has not changed. With this in mind, we present a few properties of the side-effect function.

Lemma 3.4. *Let a, b and c be three different agents. Given a model (M, w) and the model $(M', w') = \text{SIDE-EFFECT}_{(p,a,b)}(M, w)$, it holds that*

(a) $(M', w') \models B_a(B_b p \vee B_b \neg p)$;

(b) $(M', w') \models B_a\phi$ *iff* $(M, w) \models B_a\phi$, *where ϕ objective;*

(c) $(M', w') \models B_a C_{abc}(B_b p \vee B_b \neg p)$;

(d) $(M', w') \models B_i\phi$ *iff* $(M, w) \models B_i\phi$ *for* $i \neq a$.

In the above a is an agent that learns about another agent, viz. agent b; agent b is the agent about whom is learned; agent c is an agent different from a and b. In part (a) of the above lemma, agent a obtains derived knowledge of an agent b. Part (b) states that no object knowledge is learned. Part (c) phrases that agent a considers the rest of the agents as smart itself. Finally, part (d) captures that other agents do not learn.

Property (c) is a reasonable assumption of a about the other agents. If one agent believes that another agent knows the value of p, then it is reasonable to assume that another agent will believe the same. On the other hand common knowledge might be too strong to assume. Next, we address the issue that an agent a shares its belief about an agent b with other agents. We represent this by linking a's beliefs of those other agents back to the original (unmodified) states. We distinguish four different type of groups of agents.

1. the group A of agents that learn about other agents, ranged over by a;
2. the group B of agents that is learned about, ranged over by b;
3. the group C of agents of which agents a believe they have commonly learned about agents in group b with, ranged over by c;
4. the group D of agents of which agents in a believe they have learned nothing about, ranged over by d.

For simplicity of presentation, we assume that there is exactly 1 agent present in each group, i.e. $\mathcal{A} = \{a, b, c, d\}$. We define the new side-effect operation 0-UNFOLD (where 0 refers to zero-knowledge). Note that the 0-UNFOLD operation depends on the particular partinioning of agents $\mathcal{A}$. Since the operation S0-IDE-EFFECT will depend on the unfold operation, the side-effect function is also taken with respect to some chosen partitioning of the agent set.

Definition 3.10. Given a model (M, w) such that $M = \langle S, \pi, R_1, ..., R_m\rangle$ and the set of agents $\{a, b, c, d\}$, we define a function $0\text{-UNFOLD}(M, w) = (\langle S', \pi', R'_1, ..., R'_m\rangle, w')$, where:

- $S' = \mathit{new}_a(S) \cup \mathit{new}_{bcd}(S) \cup \mathit{orig}(w)$
- $w' = \mathit{orig}(w)$
- $\pi'(\mathit{new}_{\mathcal{B}}(v))(p) = \pi(v)(p)$ and $\pi'(\mathit{orig}(w))(p) = \pi(w)(p)$ for all $p \in \mathcal{P}$, $\mathcal{B} \in \{\{a\}, \{bcd\}\}$
- R'_i on S' is minimal such that

$$\begin{array}{lcl} R'_d(\mathit{new}_a(u), \mathit{new}_{bcd}(v)) & \Leftrightarrow & R_d(u, v) \\ R'_i(\mathit{new}_a(u), \mathit{new}_a(v)) & \Leftrightarrow & R_i(u, v)\ (i \neq d) \\ R'_i(\mathit{new}_{bcd}(u), \mathit{new}_{bcd}(v)) & \Leftrightarrow & R_i(u, v) \\ R'_a(\mathit{orig}(w), \mathit{new}_a(v)) & \Leftrightarrow & R_a(w, v) \\ R'_i(\mathit{orig}(w), \mathit{new}_{bcd}(v)) & \Leftrightarrow & R_i(w, v) \wedge i = b, c, d \end{array}$$

So instead of completely separating the knowledge of te agent a with the other agents, we share the knowledge of a about d with the other agents. Since the other agents do not learn anything, a does not gain knowledge about d. We present a lemma similar to Lemma 3.2.

Lemma 3.5.

(a) If (M, w) is a Kt45 model, then so is 0-UNFOLD(M, w).

(b) For every model (M, w) it holds that (M, w) and 0-UNFOLD(M, w) *are bisimilar.*

Due to the case distinction for $R'_a(\mathit{orig}(w), \mathit{new}_{\mathcal{B}}(v))$ in Definition 3.10 above, it holds that the knowledge of a about b is not interconnected with the knowledge of other agents about b or of b itself. So we can 'cut out' the submodel containing the b arrows from the belief of a:

Definition 3.11. Given a model $(M', w') = \langle S', \pi', R'_1, ..., R'_m\rangle$ such that $(M', w') =$ $0\text{-UNFOLD}(M, w)$, for some (M, w), define $\text{B-SUB}(M') = \langle S'', \pi'', R''_1, ..., R''_m\rangle$ where

- $S'' = \{\ \mathit{new}_a(s) \mid s \in S\ \}$
- $\pi''(s)(p) \Leftrightarrow \pi(s)(p)$ for all $p \in \mathcal{P}$

- $R''_b(s,t) \Leftrightarrow R'_b(s,t)$ for $s,t \in S''$
- $R''_i = \emptyset \ \ (i \neq b)$.

The operation 0-SIDE-EFFECT is then given by $0\text{-SIDE-EFFECT}_p(M,w) = (\text{REPLACE}_N(\text{ATOMSPLIT}_{(p,b)}(N), M'), w'$ where $N = \text{B-SUB}(M')$.

Lemma 3.6. *If (M,w) is a Kt45-model, then so is $0\text{-SIDE-EFFECT}_p(M,w)$.*

Similarly to Lemma 3.3 and Lemma 3.4 we have the following result.

Lemma 3.7. *(a) If (M,w) is a Kt45-model, then so is $0\text{-SIDE-EFFECT}_p(M,w)$.*

(b) Given a model (M,w) and the model $(M',w') = 0\text{-SIDE-EFFECT}_p(M,w)$, it holds that

(i) $(M',w') \models B_a(B_b p \vee B_b \neg p)$

(ii) $(M',w') \models B_a \phi$ *iff* $(M,w) \models B_a \phi$ *for ϕ objective*

(iii) $(M',w') \models B_a C_{abc}(B_b p \vee B_b \neg p)$

(iv) $(M',w') \models B_i \phi$ *iff* $(M,w) \models B_i \phi$ *for $i \neq a$*

(v) $(M',w') \models B_a B_d \phi$ *iff* $(M,w) \models B_a B_d \phi$

Parts (i) to (iv) are compare to the properties given in Lemma 3.4. Part (v) states that the knowledge of agent a about agent d has not changed, which exactly as desired.

Example 3.5 *(side-effect function).* Recall the model (M,s) from Example 3.3 (Figure 5). We now present this model with four agents $\{a,b,c,d\}$ in Figure 9 such that $\pi(s)(p) = \pi(u)(p) =$ `true` and $\pi(t)(p) =$ `false`. Now apply $0\text{-SIDE-EFFECT}_p(M,w)$ and we gain the model (M',s) from Figure 10 such that $\pi(s)(p) = \pi(s')(p) = \pi(s'')(p) = \pi(u'')(p) =$ `true`, $\pi(t')(p) = \pi(t'')(p) =$ `false`. Note that in this model, a still knows exactly the same about d as it did before.

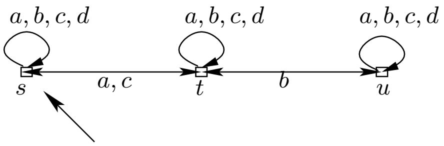


Figure 9: (M,s)

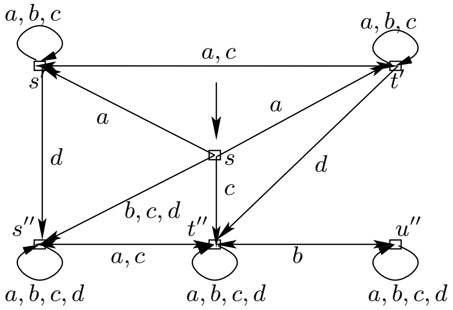


Figure 10: (M',s)

4 A logical language for security protocols

In this section we exploit the ideas of the previous section for a logical language to reason about security protocols. The EXPAND and SIDE-EFFECT operations are used for its semantics. It is illustrated for the case of the SRA Three Pass protocol how the various steps of the protocols change the initial knowledge of the agents involved and what Kripke-structure is finally obtained.

Definition 4.1. Fix a set of proposition $\mathcal{P}$, ranged over by p, and a set of agents $\mathcal{A}$ of m elements, ranged over by a. The language $\mathcal{L}_C$ is given by

$$\begin{gathered} \phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid B_i\phi \mid [\sigma]\phi \\ \sigma ::= \mathit{Priv}(i \to j, p) \mid \mathit{Pub}(i, p) \mid \sigma ; \sigma' \end{gathered}$$

The σ symbol denotes a (possibly composed) communication action. The action $\mathit{Priv}(i \to j, p)$ is a private or peer-to-peer message p from i to j; the action $\mathit{Pub}(i,p)$ means a public announcement or a broadcast by i about p. In the latter every agent on the network learns p, whereas in the former only j learns p. The bracket operator $[\sigma]\phi$ has the interpretation that after executing the communication action, ϕ holds.

The subscript in $\mathcal{L}_C$ refers to a set of so-called transition rules $\mathcal{C}$. The transition rules capture the updates, i.e. the expansions and side-effects, necessary for the interpretation of the constructs $\mathit{Priv}(i \to j, p)$ and $\mathit{Pub}(i,p)$. The transitions rules enforce consistency among the propositions that hold. For example, if an agent believes that the value of a message m is $[\![m]\!]$ and possesses a key $k(a)$, then it must believe that the value of the encryption $E_{k(a)}(m)$ of m has a value that corresponds with $[\![m]\!]$.

A transition rule has either the form $B_ip \Rightarrow \beta$ or $H_ip \Rightarrow \beta$. B_ip and H_ip are conditions, expressing that p must be believed by agent i or that p has been delivered to agent i, respectively. The body β of a transition rule is a sequence of actions $\alpha_1; \ldots ; \alpha_n$. Actions come in two flavours $L_{\mathcal{B}}p$ and $S_{i,j}p$. Here, $L_{\mathcal{B}}p$ expresses that p is learned among the agents in the set $\mathcal{B}$ and corresponds to belief expansion, whereas $S_{a,b}p$ expresses the side-effect that the agent a has learned that agent b now knows about p.

As an example, we will have the transition rule $B_b\{x\}_k \Rightarrow L_{ab}\{x\}_k$, when agents a and b share the key k and a sends b the message $\{x\}_k$. A typical application of an $H_ip \Rightarrow \beta$ transition rule is in a protocol that uses a challenge. In the situation described above, agent a sends the message x to agent b and agent b returns the message $\{x\}_k$. Since it is shared, a already can compute $\{x\}_k$ itself, so the delivery of $\{x\}_k$ does not teach a anything about this value. But, since it must come from b, it does learn that b has the shared key and authenticates b toward a. The transition rule in this case is $H_a\{x\}_k \Rightarrow S_{a,b}k$, stating that agent a, on observing $\{x\}_k$, learns that agent b knows the right value of the key k.

The semantics for the language $\mathcal{L}_C$, provided in the next definition, follows the set-up of [BMS99, Dit00]

Definition 4.2. Let $\mathcal{C}$ be a set of transition rules. For $\sigma \in \mathcal{L}_C$ the relation $[\![\sigma]\!]$ on models for $\mathcal{A}$ over $\mathcal{P}$ is given by

$$\begin{array}{lcl}
(M,w)[\![\mathit{Priv}(i \rightarrow j, p)]\!](M',w') & \Leftrightarrow & (M,w) \models B_i p \Rightarrow (\textsc{expand}_{p,j}(M,w) \triangleright_p (M',w')) \\
(M,w)[\![\mathit{Pub}(i,p)]\!](M',w') & \Leftrightarrow & (M,w) \models B_i p \Rightarrow (\textsc{expand}_{p,\mathcal{A}}(M,w) \triangleright_p (M',w')) \\
(M,w)[\![\sigma;\sigma']\!](M',w') & \Leftrightarrow & (M,w)[\![\sigma]\!](M'',w'')[\![\sigma']\!](M',w') \text{ for some model } (M'',w'')
\end{array}$$

$$\begin{array}{lcl}
(M,w) \triangleright_p (M',w') & \Leftrightarrow & \texttt{if } (x \Rightarrow \beta) \in \mathit{Sel}(M,w,p) \\
 & & \texttt{then } (M,w)\ \langle\beta\rangle\ (M'',w'') \triangleright_p (M',w') \text{ for some } (M'',w'') \\
 & & \texttt{else } (M,w) = (M',w') \\
 & & \texttt{end}
\end{array}$$

$$\begin{array}{lcl}
(M,w)\ \langle\rangle\ (M',w') & \Leftrightarrow & (M,w) = (M',w') \\
(M,w)\ \langle L_{\mathcal{B}}p;\beta\rangle\ (M',w') & \Leftrightarrow & \textsc{expand}_{(p,\mathcal{B})}(M,w)\langle\beta\rangle(M',w') \\
(M,w)\ \langle S_{i,j}p;\beta\rangle\ (M',w') & \Leftrightarrow & \textsc{side-effect}_{(p,i,j)}(M,w)\ \langle\beta\rangle\ (M',w')
\end{array}$$

$$\begin{array}{lcl}
(M,w) \models p & \Leftrightarrow & \pi(w)(p) = \texttt{true} \\
(M,w) \models \neg\phi & \Leftrightarrow & (M,w) \not\models \phi \\
(M,w) \models \phi \wedge \psi & \Leftrightarrow & (M,w) \models \phi \text{ and } (M,w) \models \psi \\
(M,w) \models B_i\phi & \Leftrightarrow & (M,v) \models \phi \text{ for all } v \text{ such that } wR_iv \\
(M,w) \models [\sigma]\phi & \Leftrightarrow & (M',w') \models \phi \text{ if } (M,w)[\![\sigma]\!](M',w')
\end{array}$$

where

$$\begin{array}{lcl}
\mathit{Sel}(M,w,p) & = & \{\ B_ip \Rightarrow \beta\ \in \mathcal{C} \mid (M,w)\ \langle\beta\rangle\ (M',w'),\ (M',w') \models \phi \not\Leftrightarrow (M,w) \models \phi,\ (M,w) \models B_ip\ \} \\
 & \cup & \{\ H_ip \Rightarrow \beta\ \in \mathcal{C} \mid (M,w)\ \langle\beta\rangle\ (M',w'),\ (M',w') \models \phi \not\Leftrightarrow (M,w) \models \phi\ \}
\end{array}$$

The operation $\triangleright_p$ applies a number of transition rules after the execution of a communication action. For the transition rules of type $B_ip \Rightarrow \beta$ it is checked if the precondition B_ip holds.

The selection $\mathit{Sel}(M,w)$ is organized in such a way that no transition rule is applied over and over again. The recursive definition of $\triangleright$ therefore stops if no fresh transition rule can be applied. In the definition of *Sel* it is checked if the belief of the agents changes under the transition rules, preventing an infinite chain of rewrites for $\triangleright_p$. Note that because of the results of the previous section, the order of applying these transition rules does not matter.

5 The SRA Three Pass protocol

In this section we discuss how the machinery developed above works out for a concrete example. Preparatory for this, in order to keep the models within reasonable size, we employ two helpful tricks. The first one is the disregarding of propositions not known to any agent. Thus, if a proposition is not part of the model, then the interpretation is that no agent has any knowledge about it. What we have to specify is how we add that proposition into the model. We accomplish this by making two copies of the original states. One of them we assign 'positive' and the other 'negative'. In the positive states, the proposition will be `true`, and in the negative states, the proposition will be `false`.

Definition 5.1. Given a model $(M,w) = \langle S, \pi, R_1, ..., R_m\rangle$ we define the function $\textsc{addatom}_p$ such that $(M',w') = \textsc{addatom}_p(M,w) = \langle S', \pi', R'_1, ..., R'_m\rangle$ where

- $S' = \mathit{pos}(S) \cup \mathit{neg}(S)$

- $\pi'(pos(s))(q) =$ if $p = q$ then `true` else $\pi(s)(q)$
- $\pi'(neg(s))(q) =$ if $p = q$ then `false` else $\pi(s)(q)$
- $R_i'(\alpha(s), \beta(t)) \Leftrightarrow R_i(s,t),\ \alpha, \beta = pos, neg$
- $w' = pos(w)$

We have the following property.

Lemma 5.1. *Given a model (M, w) and $(M', w') = \text{ADDATOM}_p(M, w)$ it holds that i) $(M', w') \models p$, ii) $(M, w) \models \phi \Leftrightarrow (M', w') \models \phi$ for $p \notin \phi^*$ with ϕ^* the closure under subformulas of ϕ and iii) for all $i \in \mathcal{A}$: $(M', w') \not\models B_i p$.*

The second trick helps to prevent the useless applying of rules which keeps the model in a reasonable size.

Lemma 5.2. *Given a model (M, w), the model (M', w') such that $(M, w)\langle L_i p; S_{ji} p\rangle x[Pub(i, p)](M', w')$ (for some x, i, j) and the model (M'', w'') such that $(M, w)\langle L_{\mathcal{A}} p\rangle(M'', w'')$ are bisimilar.*

That is to say, if an agent i learns p and then all other agents learn about i that it has learned p, followed by the action where everyone learns p (commonly), then it is equivalent to say that they have just learned p commonly.

Shamir, Rivest and Adelman have suggested the three-pass protocol [CJ97] for the transmission a message under minimal assumptions for commutative encryption. It is known to be insecure and various attacks have been suggested. However, it serves an illustrative purpose here. The protocol has the following steps:

1. $a \rightarrow b : \{x\}_{k_a}$
2. $b \rightarrow a : \{\{x\}_{k_a}\}_{k_b}$
3. $a \rightarrow b : \{x\}_{k_b}$

Both agent a and b have their own encryption key, k_a and k_b, respectively. The encryption key can be a symmetric key or a the private key of a private key-public key-pair for that matter. Agent a wants to send message x to agent b through an insecure channel and therefore wants to send x encrypted to b. It does this by sending x encrypted with its own key. Next, b will encrypt this message with b's key and sends this back. Since the encryption is commutative, a can now decrypt this message and sends this to b. Finally, b can decrypt the message it has just received and learn the value of x.

We consider three agents $\{a, b, i\}$ where a and b are honest agents that will run this protocol and i is the intruder that looks at the messages that are being transmitted through the network. So we're interested, if i does not actively attack the protocol, what i can learn during the run of this protocol.

Next, we define the transition rules. The first transition rule models the fact that agents can encrypt with their own key: $B_j m_{\mathcal{K}} \Rightarrow L_j m_{\mathcal{K}+j}; S_{ij} m_{\mathcal{K}+j}$ and, for $j \in \mathcal{K}$, $B_j m_{\mathcal{K}} \Rightarrow L_j m_{\mathcal{K}-j}; S_{ij} m_{\mathcal{K}-j}$, for all $j \in \mathcal{A}$, where $\mathcal{K}$ is some set of agents, $+$ a function that adds an agent and $-$ a function that deletes one. Here, $m_{\mathcal{K}}$ represent a message successively encrypted with the keys of the agents in $\mathcal{K}$. In the modeling, we limit ourselves by defining the list of useful propositions. The propositions we want to consider here are $\mathcal{P} = \{m, m_a, m_b, m_{ab}\}$ where m_a abbreviates $\{x\}_{k_a} = [\![\{x\}_{k_a}]\!]$ and m_{ab} abbreviates $\{\{x\}_{k_a}\}_{k_b} = [\![\{\{x\}_{k_a}\}_{k_b}]\!]$.

We assume that

- encryption is commutative;
- every agent has its own key not possessed by any other agent;
- during the run of the protocol, the value of the keys do not change.

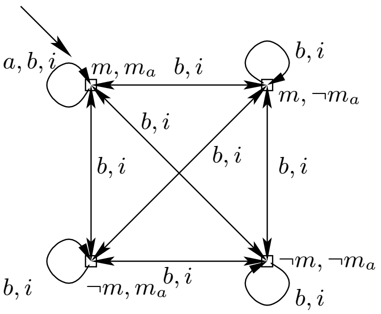


Figure 11: Starting point

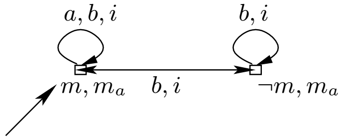


Figure 12: after $Pub(a, m_a)$

The next assumption we must make is about the knowledge state of the agents *before* the run of the protocol. We will assume that a is the only agent that knows m and m_a. Furthermore, we will assume that the other agents know this about a. The corresponding Kripke structure is in Figure 11.

The first step is executed. That is, m_a is propagated on the network, so all agents will learn this value. Thus, we execute the action $Pub(a, m_a)$. If we discard the states that become unreachable, this results in the model of Figure 12. Note that this model models $B_b m_a$. This triggers one of the transition rules, that is, it triggers $B_b m_a \Rightarrow L_b m_{ab}; S_{ib} m_{ab}$ since the antecedent holds in the point now.

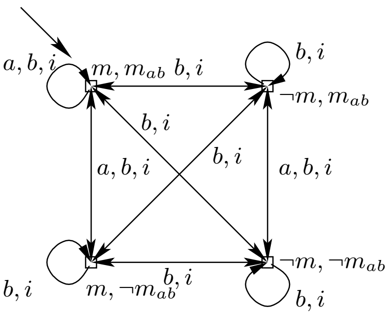


Figure 13: added m_{ab}

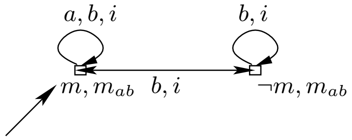


Figure 14: after $Pub(b, m_{ab})$)

We have not modelled m_{ab} yet, so this is the first step. We will not repeat m_a in the figure since this holds in any state of the model. The function $\textsc{addatom}_{m_{ab}}$ results in the model of Figure 13. Observe that in the next step of the protocol $L_{\mathcal{A}} m_{ab}$ ($Pub(b, m_{ab})$) is executed, since the message is being transmitted to all agents on the network. So with Lemma 5.2 it is justified to skip the steps that are required by the transition rules. Thus, we get the model in

Figure 14. This results in the triggering of the transition rule: $B_a m_{ab} \Rightarrow L_a m_b; S_{ia} m_b$. This is in fact a completely similar case as in the previous step of the protocol. Again we dismiss the m_{ab} proposition since every agent has learned this.

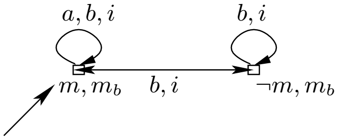


Figure 15: after $Pub(a, m_b)$

We introduce m_b and execute $Pub(a, m_b)$ because we can again skip the actions in the transition rules (Lemma 5.2). So we end up with the model in Figure 15. The last transition rule that is triggered is $B_b m_b \Rightarrow L_b m; S_{ib} m$. Again, we discard the proposition that holds in every state: m_b. We now only focus on the most interesting proposition m. First b learns m which results in the model in Figure 16.

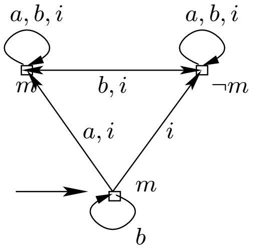


Figure 16: after $L_b m$

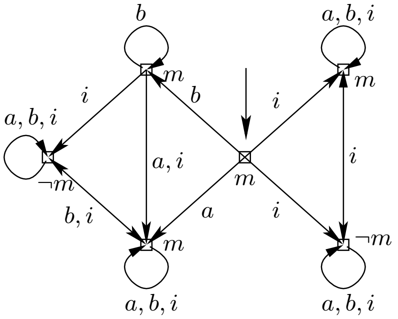


Figure 17: after $S_{ib} m$

The second action for the transition rule is that i learns that $B_b p \vee B_b \neg p$. If we execute this on the model we get the model (M', w') which can be seen in Figure 17. Recall that we have $(M', w') = [Pub(a, m_a); Pub(b, m_{ab}); Pub(a, m_b)](M, w)$. It holds that: $(M', w') \models \neg B_i m$, $(M', w') \models B_b m$ and $(M', w') \models B_i((B_b m \vee B_b \neg m) \wedge B_a(B_b m \vee B_b \neg m))$.

6 Conclusion

Inspired by recent work on dynamic epistemic logics, we have proposed a logical language for describing (properties of) runs of security protocols. The language contains constructs for the two basic types of epistemic actions that happen during such runs. The semantics of the language is based on traditional Kripke models representing the epistemic state of the agents involved in the protocol at hand. Changes in the epistemic state of the agent system as a result of the execution of a protocol are described by means of transition rules that precisely indicate what belief updates happen under certain preconditions. These belief updates give rise to modifications of the models representing the agents' epistemic state in a way that is precisely given by semantic operations on these models. We have illustrated our approach for a well-known security protocols such as the SRA Three Pass protocol. We also have analyzed the Needham-Schröder public key protocol and Andrew RPC, see [Hom03]. It should be noted, that we focus here on e single protocol runs with passive intruder. A further research goal is to extend our approach to deal with a setting of multiple protocols/multiple runs and active intruder.

The semantic updates we used operate on traditional Kripke models as opposed to updates in the approaches of Gerbrandy and Baltag. We believe that this will make it less troublesome to integrate these updates into existing model checkers, which hopefully will lead to better and new tools for verifying properties of security protocols.

Although future research will have to justify this, we are confident that our method can be employed for a broad class of verification problems concerning security protocols because of the flexibility of our approach using transition rules for epistemic updates.

References

[AHV01] N. Agray, W. van der Hoek, and E.P. de Vink. On BAN logics for industrial security protocols. In B. Dunin-Keplicz and E. Nawarecki, editors, *From Theory to Practice in Multi-Agent Systems*, pages 29–38. LNAI 2296, 2001.

[And01] R.J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2001.

[AT91] M. Abadi and M. Tuttle. A semantics for a logic of authentication. In *Proc. PODC'91*, pages 201–216. ACM, 1991.

[Bal00] A. Baltag. A logic for suspicous players: epistemic actions and belief-updates in games. Technical Report SEN–0044, CWI, 2000.

[BAN90] M. Burrows, M. Abadi, and R.M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8:16–36, 1990.

[BM97] A. Bleeker and L. Meertens. A semantics for BAN logic. In *Proceedings DIMACS Workshop on Design and Formal Verification of Protocols*. DIMACS, Rutgers University, `http://dimacs.rutgers.edu/Workshops/Security`, 1997.

[BMS99] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge and private suspicions. *Annals of Pure and Applied Logic*, 1999.

[CJ97] J.A. Clark and J.L Jacob. A survey of authentication protocols 1.0. Technical report, University of York, 1997.

[Dit00] H.P. van Ditmarsch. *Knowledge games*. PhD thesis, Universiteit van Amsterdam, 2000.

[Dit01] H.P. van Ditmarsch. The semantics of concurrent knowledge actions. In M. Pauly and G. Sandu, editors, *Proc. ESSLLI Workshop on Logic and Games*, Helsinki, 2001.

[DY83] D. Dolev and A.C. Yao. On the security of public-key protocols. *IEEE Transaction on Information Theory*, 29:198–208, 1983.

[Ger97] J. Gerbrandy. Dynamic epistemic logic. Technical Report LP–97–04, ILLC, 1997.

[Ger99] J. Gerbrandy. *Bisimulations on Planet Kripke*. PhD thesis, University of Amsterdam, 1999.

[Hom03] A.J. Hommersom. Reasoning about security. Master's thesis, Universiteit Utrecht, 2003.

[KN98] V. Kessler and H. Neumann. A sound logic for analyzing electronic commerce protocols. In J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollman, editors, *Proc. ESORICS'98*, pages 345–360. LNCS 1485, 1998.

[Koo03] B. Kooi. *Knowledge, Chance, and Change*. PhD thesis, Rijksuniversiteit Groningen, 2003.

[RHM02] J.-W. Roorda, W. van der Hoek, and J.-J.Ch. Meyer. Iterated belief change in multi-agent systems. *Proceedings of the first International Joint Conference on Autonomous Agents and Multi-Agent Systems: Part 2*, 2002.

[SC01] P. Syverson and I. Cervesato. The logic of authentication protocols. In R. Foccardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design: Tutorial Lectures*. LNCS 2171, 2001.

[Sch00] B. Schneier. *Secrets and Lies: Digital Security in a Networked World*. Wiley, 2000.

[SW02] S.G. Stubblebine and R.N. Wright. An authentication logic with formal semantics supporting synchronization, revocation and recency. *IEEE Transactions on Software Engineering*, 28:256–285, 2002.

[WK96] G. Wedel and V. Kessler. Formal semantics for authentication logics. In E. Bertino, H. Kurth, G. Martello, and E. Montolivo, editors, *Proc. ESORICS'96*, pages 219–241. LNCS 1146, 1996.

Verifying the Contract Net Protocol: A Case Study in Interaction Protocol and Agent Communication Language Semantics

Shamimabi Paurobally
University of Southampton
Southampton SO17 1BJ, UK.
sp@ecs.soton.ac.uk

Jim Cunningham
Imperial College
London SW7 2BZ, UK.
rjc@doc.ic.ac.uk

Nicholas R. Jennings
University of Southampton
Southampton SO17 1BJ, UK.
nrj@ecs.soton.ac.uk

May 14, 2004

Abstract

Multi-agent conversations are built upon two components: agent communication languages (ACLs) that specify the individual messages that can be exchanged and interaction protocols (IPs) that specify the sequences in which these message can be arranged. Although informative, the semantic definition proposed for the most standard ACL (FIPA 1997) is complicated and contentious, while published IPs tend to be ambiguous, incomplete, and unverified with respect to message semantics. As a case study to clarify and help rectify these problems, we have investigated verification of the contract net protocol when its messages are presumed to be expressed in FIPA ACL. In order to help both informal comprehension and formal verification we separate several concerns. We suggest a revised and simpler core semantics for many of the FIPA ACL speech acts, using the same belief-intention style of logic, although the underlying ideas are not dependent on this detail. An extended form of propositional dynamic logic and statecharts is used to express IPs. States are interpreted using mutual beliefs and intention, and properties such as termination and consistency of joint beliefs are shown.

1 Introduction

Social interactions, such as cooperation, coordination and negotiation, are a fundamental feature of multi-agent systems. They are enacted through a variety of agent communication languages (ACLs) and interaction protocols (IPs). An ACL (for example

KQML [3], FIPA ACL [4]) specifies the individual communicative acts (CAs), typically as classes of asynchronous messages modelled on the Theories of Speech Acts enunciated by Austin [1] and Searle [12] . The 1997 semantic specification for FIPA ACL is expressed using a logic of belief and intention and derived from work associated with Sadek [11]. This specification is informative has been criticised on various grounds [10] [6], not least that it is unverifiable [16]. Relatedly, an IP (for example the contract net protocol (CNP) [15] or an English auction protocol) specifies message sequences that can lead towards a goal state. However to date, may of the published specifications for these protocols suffer from ambiguities and incompleteness [8]. This lack of precision can arise from the inherent inexpressiveness of diagrammatic representations such as Petri-nets [7] and AUML [4], or from the level of abstraction chosen when using informal language or formal logic. Such representations can nevertheless be informative and helpful for comprehension. Commitment-based semantics have been used for modeling multi-agent interactions [17]. However this work remains focussed on the creation, fulfilment and discharge of an agreement, tends to be centralised, through for example an instituion, and discourages interactions whose goals are to share experience and model their environment. The legal issues also are often not considered.

Against this background, we use the CNP (arguably the most widely adopted IP) as a case study in this paper, to expose and apply a simpler semantics for an ACL, and to provide a compatible semantics for the protocol, which itself is represented in extended propositional dynamic logic (PDL [5]) and statecharts. Together these allow us to prove termination and consistency regarding some of the group's beliefs when interpreting the CNP. At the same time, in the light of criticisms about ambiguities in ACL semantics, our proposed semantics for an ACL and IPs serve as an example of how belief semantics [2] can provide insight in the clear specification of agent interactions.

Thus, this paper addresses the unresolved problem of a suitable ACL semantics for expediting agent interactions. Our first contribution is to show how to separate treatment of message delivery, sincerity, and implicit protocol issues within the existing style of FIPA-ACL logic (in the current work they are all implicitly assumed). This separation of issues has enabled us to simplify verification of the contract net protocol with respect to the message semantics. Our semantics are not dependent on the actual choice of ACL or the particular style of logic that we use. This means that a protocol specified in extended PDL does not depend on the actual ACL being used or the semantics of the ACL. The second contribution is also partly methodological. We overcome the lack of expressiveness of weaker graphical representations by using an enhanced form of statechart and extended PDL to represent the protocol both visually and using a logical theory. Finally, we show how the definitions of the states allows a semantic representation using a logic in the belief-intention style. This enables us to derive more specific properties of the protocol in terms of joint beliefs. We refer to a belief-intention *style* of logic but we do not provide a formal definition of any such logic in this paper (see section 5). Monadic modal operators for a belief (B_i) and an intention (I_i) of an agent i are used as intellectual props for deductive inference and treated as independent except for explicit interaction axioms. A belief logic is a useful ideal for giving epistemic status to the consistent but not necessarily true internal propositions that can be used by a designer to express and reason about information internal to an agent. Treating intention in a similar way is also a useful ideal for succinct reasoning about a goal state without getting into more temporal reasoning. The durability of the Belief-Desire-Intention paradigm for practical deliberative agents also provides a heuristic justification for this sort of reasoning.

The remainder of this paper is structured in the following way. In the next section we provide an informal summary of the separate issues in simplifying reasoning about the FIPA ACL. Section 3 critically analyses a representation of the CNP in Petri nets and those aspects of the specification that cannot be captured with in this approach. Section 4 develops a novel formal representation of the CNP in extended PDL and extended statecharts. In section 5, we discuss the axioms and assumptions of the belief logics and ACL. Sections 6 and 7, respectively, summarize our use of communicative acts and interpret the states in the CNP. In section 8, we validate our approach by proving desirable properties of the CNP in our framework. Section 9 presents our conclusions and future work.

2 Issues in Reasoning about the FIPA ACL

As defined in FIPA-ACL 1997 there is a feasibility pre-condition (FP) and a rational effect (RE) associated with each CA. The FP conjoins a sincerity condition (SC), with a typically more complex Gricean condition (GC) to preclude a redundant message. The RE expresses the condition that the sender may use in planning the communicative act. The FIPA *inform*, (or KQML *tell*) is the basic CA. As a message it has parameter for sender s, receiver r and propositional content ϕ. The SC is $B_s\ \phi$ and the RE is $B_r\ \phi$ The GC expresses the belief that the sender does not believe the receiver believes ϕ, or is uncertain about it.

The obvious criticism is that the Gricean condition introduces inessential complexity, even if the term “inform” is inappropriate without it, but there are deeper issues. The semantic conditions as expressed are sender oriented, and there is no overt association between the semantic conditions and the occurrence of the message itself, after all, sincerity and non-redundancy are social conditions, not mechanical. Although these concerns have been pointed out in [13], [16] we take a new step in removing much of their impact by re-expressing the semantics in the belief-intention logic itself.

The key step is not obvious and indeed exploits an obvious ”hack” in the logic, which is to avoid the detailed expression of temporality or causation by using the special proposition $done(a, act)$, for any agent a and action act. Using PDL notation, we express $done(a, act)$ as $done(a.act)$. Now the propositions $I_s\ done(s.m)$ and $B_r\ done(s.m)$ respectively can express that the sender agent s intends that m be sent, and receiver agent r believes that m has been sent by s. These are the tightest pre- and post-conditions we can express in the logic, and suggest the following PDL axiom schema for such messages, assuming there is a sender *s* and a receiver *r*:

$$I_s done(s.m) \leftrightarrow [s.m] B_r done(s.m)$$

This causality style of schema is potentially verifiable in a logic that is grounded in machine states, but for our purposes it enables us to separate the FP and RE from the message transport. FIPA-like semantics are re-instated by assuming that the receiver believes that the message transport post-condition entails the message RE ($B_r done(s.m) \rightarrow B_r RE_m$), and that the pre-condition $I_s done(s.m)$ is itself a primitive plan by which the sender can attain the RE. The FIPA semantics do not assume the sender to intend the RE, but one can take the view that the FP should be strenghened to entail $I_s done(s.m)$.

Indeed, each agent can access a common ontology of messages and infer from the receipt of a message what it means, and this may vary with any other environmental context that is available. We will simply drop the GC part of the FP, on the grounds that it is really a social protocol for human communication which need not be presumed in the context of any other interaction protocol, but retain the SC part as simpler FIPA-like pre-condition, and allow the FIPA RE as a trustworthiness assumption (see section 5.2).

It turns out that the remaining FIPA-ACL messages that we use can be re-expressed as special cases of the *inform* act in context. This was known when the standard was prepared, but in some cases obscured and made erroneous by the inessential complexity we have sought to remove. For example a *propose* message, sent by a potential CNP contractor to the manager is an *inform* with the propositional content that *if* the sender believes that the receiver intends that the action be done by the sender, *then* the sender (will) intend this (too). So a sincere *accept* or *reject* requires belief by the manager in the propositional content of such a proposal, and becomes an *inform* message with content expressing the manager's intention, so that the contractor can discharge the conditional ”promise”. Given these issues in the semantics of ACLs, there also exist issue in the current representations of IPs.

3 Issues in Representing IPs

To illustrate the required expressiveness of a specification language for realistic IPs, we consider the CNP since this is probably the most widely used protocol in the field. Figure 1, from [7], presents the CNP as a “Coloured” Petri Net. The interaction is started by a manager issuing a *call for proposals (cfp)*. Potential contractors respond with *proposals*, which the manager either rejects or accepts. Accepted proposals can be either cancelled by the manager or executed by the proposer, who later informs the manager of success or failure of their execution. The manager may also re-select other proposals or issue a new *cfp*.
However figure 1 is both ambiguous and incomplete:

- A manager has no means of deciding whether the overall contract net process has succeeded or failed. For example, if out of *n* proposals, *m* are done and *p* fail, the state of the overall process is undefined.

- This Petri net illustrates the same states and triggers for all three contractors. This is schematically redundant and is particularly problematic in open multi-agent systems; it would be hard to express the state of many contractors, which join the interaction dynamically, without more formal techniques.

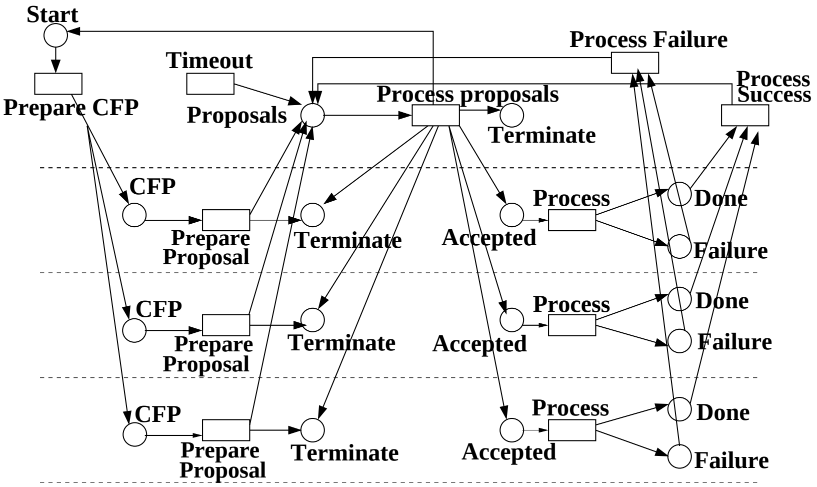


Figure 1: Contract Net conversation with three contractors in Coloured Petri Net

- Even if roles are used, as in AUML [4], there is no binding of an action to an agent's identity. Thus the manager should only send acceptance messages to those contractors who proposed, and only those proposers should inform the manager of the result of their execution.
- A deadlock will occur if all contractors refuse to bid or if the manager does not issue a new call. This is because for termination or for issuing a revised call, at least one contractor has to send a proposal to the current call.
- Alternative actions cannot be distinguished. For example, instead of the manager being able to both accept and reject each proposal, as in figure 1, it should either accept or reject each proposal.

There are also other aspects missing in the CNP representation. It is not shown that different contractors have different beliefs depending on what they sent and received. Thus, since proposals are not broadcasted, a contractor will not know who else other than itself has sent a proposal, whilst the manager knows all proposers. In turn, the manager has to ensure that it accepts or rejects only those agents which sent a proposal in the first place. Thus, there is a shared belief between a contractor and the manager, (e.g. whether that contractor has sent a proposal or whether its proposal has been accepted). In the same protocol, some states are public, some are private to an agent, while yet others are mutually believed. Sometimes, the private states of an agent have to be expressed; for example, the state when the manager is deliberating a proposal is needed in order to show that: *i)* there is a specific set of contractors whose proposals a manager has chosen to accept and another distinct set whose proposals are rejected; and *ii)* contractors wait for the manager to finish its deliberation. These required features of the CNP are captured through our representation in the next section.

Table 1: **Semantics of Extended PDL**

$M, w \models p$	iff $w \in V(p), p \in PROP$
$M, w \models [\gamma]A$	iff $\forall w_1(wR_\gamma w_1$ implies $M, w_1 \models A)$
$M, w \models A(X)$	iff $M, w \models A$ and $X \in Ag\_group$
$M \models (\gamma_1 :: \gamma_2)$	iff $R_{\gamma_1} \subseteq R_{\gamma_2}$
$M, w \models none\_of(S)$	iff $\forall A(A \in S$ implies $M, w \not\models A)$
$M, w \models one\_of(S)$	iff $\exists A_1 \forall A_2((A_1 \in S$ and $M, w \models A_1)$ and $(A_2 \in S\text{and}M, w \models A_2))$ implies $A_1 \leftrightarrow A_2)$
$R_{Ag\_group.\gamma} \subseteq R_\gamma$	
$R_{A?} = \{(w, w) : M, w \models A\}$	
$R_{\gamma?} = \{(w_1, w_2) : (w_1, w_2) \in R_\gamma\}$	

4 A Formal Representation for Interaction Protocols

Here we show how to represent IPs formally in extended PDL and graphically in extended statecharts. First, however we specify our extensions to PDL.

4.1 Extended PDL

We extend PDL to enable us to reason about the effect of processes on interaction states (refer to [8] for more details and a complete axiomatisation). Specifically, let *A* denote a formula and γ denote a process. In general, a formula may be in multi-modal logic, including beliefs, desires and intentions, as well as actions. The formula $[\gamma]A$ has the intended meaning: *A* holds after executing process γ. (The formula $[\gamma]A$ is also the weakest precondition for γ to terminate with *A*). The syntax of the extended logic is defined below, where $\mathcal{A}$ denotes one agent or a set of agents and the term *States* denotes set of formulae.

$$\begin{array}{lll} \text{Formulae:} & A ::= p \mid \bot \mid A_1 \rightarrow A_2 \mid [\gamma]A \mid B_{\mathcal{A}}A \mid I_{\mathcal{A}}A \mid A(\mathcal{A}) \mid \gamma_1 :: \gamma_2 \\ & \qquad \mid none\_of(States) \mid one\_of(States) \\ \text{Processes:} & \gamma ::= \varpi \mid \gamma_1 ; \gamma_2 \mid \gamma_1 \cup \gamma_2 \mid \gamma^* \mid A? \mid null \mid abort \mid \mathcal{A} \ \centerdot \gamma \mid \gamma? \end{array}$$

The complex process ($\gamma_1; \gamma_2$) denotes the sub-process γ_1 followed by γ_2, the process ($\gamma_1 \cup \gamma_2$) either γ_1 or γ_2 non-deterministically, while γ^* denotes zero or more iterations of process γ. A state test operator "?" allows sequential composition to follow only if successful. A *null* process represents no execution, while an *abort* process results in a failed state. We extend the program logic of PDL so as to express multi-agent interactions. The semantics of the additional constructors are specified in table 1, and are based on a Kripke model denoted by *M = (W, R_γ, V)* [5]. We add types for agents and roles. We assume throughout that each atomic formula *p*, agent and instance of an atomic process ϖ can be denoted by a distinct identifying term. Set notation is used to manipulate sets of agents and interaction states. The formula *none_of(B)* holds if none of the formulae in the set *B* are true. The formula *one_of(B)* holds if only one of the formulae in the set *B* is true.

An agent or a group of agents, $\mathcal{A}$, may execute atomic actions or complex processes, γ. So the term $\mathcal{A}.\gamma$ can be read as $\mathcal{A}$ executes process γ, as for example in *r:retailer.display* means *retailer r* executes the *display* process, but the agent role may be omitted. Using set notation, we can denote a joint process between two parties as $\{$*r,c*$\}$*.shopping*. The formula $A(\mathcal{A})$ allows state A to have an agent or a group, $\mathcal{A}$, as parameters. The semantics of $M, w \models (\gamma_1 :: \gamma_2)$ states that all the worlds obtained through execution of process γ_1 are elements of the set of worlds possible through performing γ_2. For example,
EbayAuction::EnglishAuction means that all the rules in the English auction apply to the Ebay auction.

4.2 CNP in Extended PDL and Statecharts

Figure 2 shows the CNP in an extended form of statecharts and each state is fully defined in figure 4. The process $[X \backslash Y]$ means the process of replacing occurences of *X* with *Y* in the resulting state.

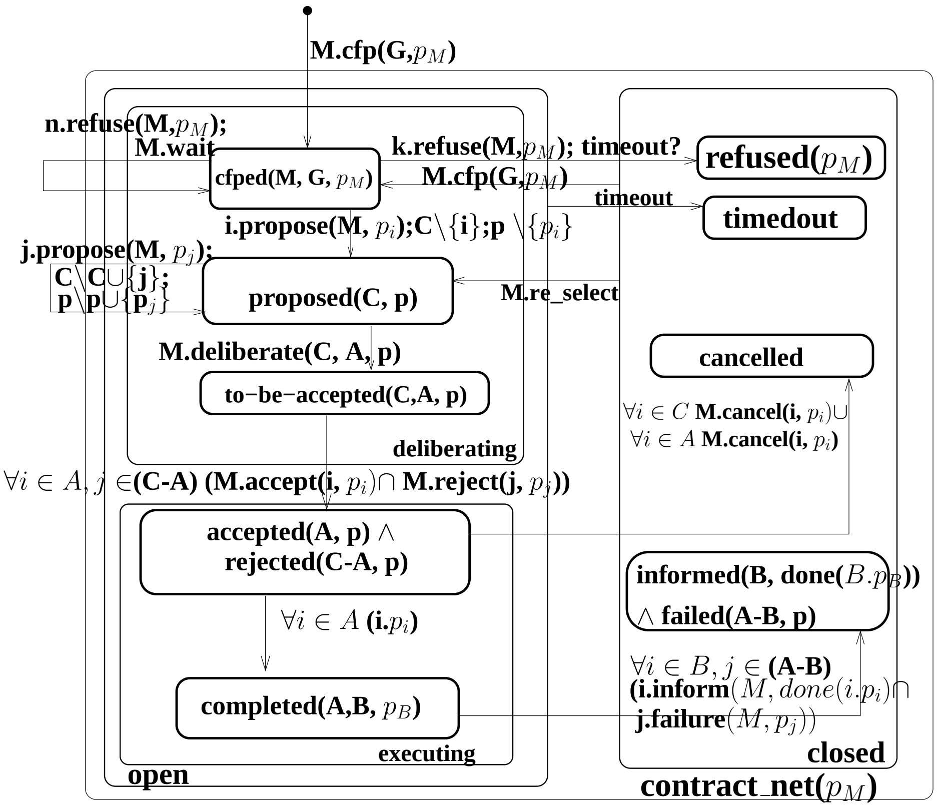

M.cfp(G,p_M)
n.refuse(M,p_M);
M.wait
cfped(M, G, p_M)
k.refuse(M,p_M); timeout?
refused(p_M)
M.cfp(G,p_M)
timeout
timedout
i.propose(M, p_i);C$\backslash\{$i$\}$;p $\backslash\{p_i\}$
j.propose(M, p_j);
C$\backslash$C$\cup\{$j$\}$;
p$\backslash$p$\cup\{$p$_j\}$
proposed(C, p)
M.re_select
M.deliberate(C, A, p)
to−be−accepted(C,A, p)
cancelled
$\forall i \in C$ M.cancel(i, p_i)$\cup$
$\forall i \in A$ M.cancel(i, p_i)
deliberating
$\forall i \in A, j \in$(C-A) (M.accept(i, p_i)$\cap$ M.reject(j, p_j))
accepted(A, p) $\wedge$
rejected(C-A, p)
informed(B, done($B.p_B$))
$\wedge$ failed(A-B, p)
$\forall i \in A$ (i.p_i)
$\forall i \in B, j \in$ (A-B)
(i.inform$(M, done(i.p_i)\cap$
j.failure$(M, p_j))$
completed(A,B, p_B)
executing
closed
open
contract_net(p_M)

Figure 2: CNP in Extended Statechart Notation (see section 7 for interpretation of the states

$$\neg\textit{contract\_net}\leftrightarrow[\textit{G.contract\_net\_process}]\textit{closed} \tag{1}$$

$$\textit{contract\_net} \leftrightarrow \textit{one-of}\;(\{\textit{ open , closed}\}) \tag{2}$$

$$\textit{closed} \leftrightarrow \textit{one-of}(\{\textit{(failed}\wedge\textit{informed), timedout, cancelled, refused}\}) \tag{3}$$

$$\textit{open} \leftrightarrow \textit{one-of}(\{\textit{deliberating, executing}\}) \tag{4}$$

$$\textit{deliberating} \leftrightarrow \textit{one-of}(\{\textit{cfped, proposed, to-be-accepted }\}) \tag{5}$$

$$\textit{executing} \leftrightarrow \textit{one-of}(\{\textit{(accepted}\wedge\textit{rejected), completed }\}) \tag{6}$$

$$\neg\;\textit{contract\_net}\leftrightarrow[\textit{M:manager.cfp(G ,}p_M\textit{)}]\textit{cfped(M,G,}p_M\textit{)} \tag{7}$$

Figure 3: Theory of CNP’s diagram structure in extended PDL

$$\begin{aligned}
&\textit{cfped(M,G,}p_M\textit{)}\leftrightarrow\textit{(}\\
&\quad[\textit{n:contractor.refuse(M,}p_M\textit{); M.wait]cfped(M,G, }p_M\textit{)}\\
&\quad\vee\;[\textit{n:contractor.refuse(M,}p_M\textit{); timeout?] refused(}p_M\textit{)}\\
&\quad\vee\;[\textit{i:contractor.propose(M,}p_i\textit{);C} \setminus \{\textit{i}\}\textit{; p}\setminus \{p_i\}\textit{]}\\
&\quad\textit{(proposed(C, p)} \wedge\forall i(i\in C\leftrightarrow B_M done(i\textit{.propose(M.}p_i\textit{))))}
\end{aligned} \tag{8}$$

$$\begin{aligned}
&\textit{proposed(C, p)} \leftrightarrow \textit{[M:manager.deliberate(C,A, p)]}\\
&\quad\textit{(to-be-accepted(C,A, p)} \wedge\forall i\in A,\forall j\notin A(B_M(i\in A)\wedge B_M(i\notin A))\textit{)}\\
&\quad\vee\;\textit{[j:contractor.propose(M,}p_j\textit{);C} \setminus C\cup\{j\};p\setminus p\cup\{p_j\}\textit{]}
\end{aligned} \tag{9}$$

$$\begin{aligned}
&\textit{to-be-accepted(C,A,p)}\leftrightarrow\textit{[}\forall i\in A,\forall j\in(C-A)\textit{(M:manager.accept(}i,p_i)\cap\textit{ M:manager.reject}(j,p_j)\textit{)]}\\
&\quad\textit{(accepted(A, p)} \wedge\forall i\in A(B_i(accepted(i,p_i))))\\
&\quad\wedge\;\textit{(rejected(C-A, p)} \wedge\forall j\in(C-A).B_j(rejected(j,p_j))\textit{)}
\end{aligned} \tag{10}$$

$$\begin{aligned}
&\textit{accepted(A, p)}\leftrightarrow\textit{[}\forall i\in A(i.p_i)\textit{)] (completed(A,B,p)} \wedge(\forall i\in B(B_i succeeded(p_i))\\
&\quad\vee\forall j\in(A-B)B_j\neg succeeded(p_j))\vee\textit{) [}\forall i\in A(M:manager.cancel(i,p_i)\cup\\
&\quad\forall i\in C(M:manager.cancel(i,p_i)\textit{]cancelled}
\end{aligned} \tag{11}$$

$$\begin{aligned}
&\textit{completed(A,B, p)} \leftrightarrow [\forall i\in B,\forall j\in(A-B)\textit{ [i.inform(M, done(i.}p_i))\cap j\textit{.failure(M,}p_j\textit{]}\\
&\quad\textit{(informed(B, done(}B.p_B\textit{))} \wedge \textit{failed(A-B, p))}
\end{aligned} \tag{12}$$

$$\textit{open}\leftrightarrow\textit{[timeout] timedout} \tag{13}$$

$$\textit{closed }\leftrightarrow\textit{[M:manager.cfp(G,}p_M\textit{)}]\textit{cfped(M,G,}p_M\textit{)} \vee \textit{[M:manager.re\_reselect]proposed(C, p)} \tag{14}$$

Figure 4: Definition of the states of the CNP in extended PDL

Figures 3 and 4 provide a corresponding logical theory of the protocol in extended PDL. Axioms (1)-(7) in figure 3 specify the relation between the states as given in the CNP diagram in figure 2. Axioms (8)-(14) both define the states and state transitions of the CNP. In more detail, axiom (1) defines that a group of agents *G* adheres to the CNP in a process instance called *contract_net_process*. Double implication in the action-condition rules allows an agent to infer the history of an interaction. Axioms (2)-(6) define the relations between parent and sub-states, as seen in the hierarchy of figure 2. There are 5 other axioms (not shown here) for ensuring when a parent state is false, none of its sub-states are true.

A manager may initiate a contract net process into a *cfped* state by issuing a call for proposal to a group of contractors *G*, only if the interaction has not yet started (Axiom 7), leading to the *cfped* state. Contractors may refuse the manager’s *cfp*, and if the manager receives only refusals by the deadline, the process terminates in a *refused* state (Axiom 8). Otherwise, some contractors may refuse whilst others (proposers) send a proposal, leading to *proposed(C, p)* where *C* is the set of proposers and *p* the set of proposals, each associated to its proposer. Further proposals from other proposers are added to the sets *C* and *p*. The expression $\forall i(i \in C \leftrightarrow B_M i$*.done(i.propose(M,*$p_i$*)))* means that the manager's beliefs include the identity of all proposers in *C* (Axiom 8).

Then the contractor deliberates to record those proposals it will accept as set *A*. The condition $\forall i \in A, j \notin A(B_M(i \in A)) \wedge (B_M(i \notin A))$ ensures that the manager's beliefs include accepted and rejected proposers (axiom 9). The manager concurrently sends an *accept* message to all chosen proposers in *A*, and rejections to those in *(C-A)* (axiom 10). The manager can also cancel the *cfp* or reject all proposals through a *cancel* message.

The action $\forall i \in A(i.p_i)$ and the state *completed(A,B,p)* express the fact that all accepted contractors in *A* execute their proposals and if successful are implicitly part of the set *B*. Each contractor privately believes whether it has succeeded or not in its execution (axiom 11). Finally, the contract net process terminates after all contractors in *B* have informed the manager of success, and the rest in *(A-B)* of failure (axiom 12). In an *open* state, a timeout can occur at any point (axiom 13), while from the *closed* state the manager can re-issue a call for proposal (axiom 14).

5 Beliefs and Intentions in a Group

Given the formal representation of the CNP in extended PDL, we now give meaning to the actions and states in the protocol in terms of the beliefs and intentions of the group. To do this, we use the modal operators for beliefs and intentions [2] [14]. Specifically, an axiomatic system for belief may be defined in terms of axioms for consistency and introspection [2]. We assume that each agent in a group has such a system of beliefs. The formula $B_i\alpha$ is read as agent *i* believes α, $I_i\alpha$ is read as agent *i* intends to do α, and $E_G\alpha$ is read as everyone in a group of agents, *G*, believes α, where α itself may express an agent's beliefs and intentions. The joint beliefs ($B_{i_1}\alpha_1 \wedge \ldots B_{i_n}\alpha_n$) of a group are the sentences that are consequences of the union of the individual beliefs of the agents. We also re-use a modified version of the FIPA SL *done* operator [4]. Here, *done(i.a)* (*done(i,a)* in FIPA SL) means that agent *i* has performed action *a*. Pre-conditions *p* for doing *a* can be expressed as part of *a* (e.g. *p?;a*).

As mentioned in section 2, we ignore the Gricean condition involving uncertain beliefs. We also specify intention axioms and other axioms for sincerity and trust that hold in our environment. These axioms are independent of the ACL and the protocol, but apply to reasoning about an interacting group of agents. Thus, although the semantics or the ACLs may differ, these core axioms should nevertheless hold in all agent interactions.

5.1 Belief and Intention Axioms

We assume the tradition axioms K, D, 4, 5 for belief (but not normality). They express closure under implication, consistency, and positive and negative introspection. Together they make iterated belief redundant, i.e. $B_iB_i\alpha \leftrightarrow B_i\alpha$. In addition to a KD5 axiom for belief, we have axioms for intentions and beliefs about intentions. Like positive introspection for belief and knowledge, we assume an agent's iterated intentions collapse to a single intention (A1): $I_i I_i \alpha \leftrightarrow I_i \alpha$.

Axiom (A2) states that an agent *i* is aware of its intentions and, intends what it believes it wants to intend: $B_i I_i \alpha \leftrightarrow I_i \alpha$.

We can have a stronger system where an agent is aware of what it does not intend to do, which is some kind of negative introspection (A3): $\neg I_i \alpha \rightarrow B_i \neg I_i \alpha$.

An agent is rational, that is it does not intend what it believes it does not intend (A4): $B_i \neg I_i \alpha \rightarrow \neg I_i \alpha$.

Intentions for negative intentions also collapse to not intending α (A5):
$I_i \neg I_i \alpha \rightarrow \neg I_i \alpha$.

An agent has control over its beliefs. If it intends to believe α, then it intends it (A6): $I_i B_i \alpha \rightarrow I_i \alpha$. This also implies $I_i B_i \neg\alpha \rightarrow I_i \neg\alpha$.

From axioms (A1) and (A2), we obtain the following axiom, which can be further simplified to: $I_i \alpha$ (A7): $B_i I_i B_i I_i \alpha \leftrightarrow B_i I_i \alpha$.

Definitions for common intentions may be formulated in the same way as common knowledge and common beliefs. Axioms (A1) to (A6), except for axiom (A3) hold in our framework, where α can itself include belief and intention modalities.

5.2 Assumptions

In this section, we formulate the axioms holding in our model. These are the foundations of our reasoning and proofs and combined infer a sincere and trustworthy behaviour, and that messages are successfully delivered. All of these assumptions are implicit in FIPA ACL. Although they may seem to require an ideal environment, our goal in this paper is to provide a simple and well-founded semantics that work in such a social context. Untrustworthy environments and relaxing these are avenues for future work. Let *s* (sender) and *r* (receiver) represent two different agents interacting with one another.

Sincerity Axioms: $I_s B_r B_s \alpha \rightarrow B_s \alpha$.
$I_s B_r I_s \alpha \rightarrow I_s \alpha$.

Agents are sincere and the sender does not intend the receiver to believe what it does not believe itself.

Trust Axiom: $B_r I_s B_r \alpha \rightarrow B_r \alpha$.

This states that if *r* believes that agent *s* sent a message to agent *r* that α holds, then *r* believes α. Receivers trust the sender. If α is a proposition or a belief formula, then *r* also believes *s* believes α; that is, $B_r I_s B_s \alpha \rightarrow B_r \alpha \wedge B_r B_s \alpha$ (Trust Axiom 2).

Cooperative Axiom: $B_r I_s I_r \alpha \rightarrow (I_r B_s I_r \alpha \vee I_r B_s \neg I_r \alpha)$.

The agents are co-operative. Thus on receiving a message, an agent replies, even if it is a refusal or a rejection.

Similar to the FP and RE of a CA, there are preconditions and postconditions when sending a message, that are independent of the meaning of the CA. Our precondition states that a sender intends for the message to be sent. For example, the CA *s.inform(r,p)* means that the sender *s* informs the receiver *r* that *p* holds. The FIPA semantics define the FP of this CA as the sender believing *p*. However, we also need to specify that the sender intends to send *r* the *inform* CA about *p*. The FP for sending all the sender's beliefs will hold. Let *done(m)* denote a sent message *m* containing a CA. Let *FP(m)* denote the FP of the CA and *RE(m)* denote the RE of the CA being sent. The axioms that apply to message exchange are:

Transport Precondition: $I_s done(m)$. Before sending a message, sender *s* intends to send it.

Transport Postcondition: $B_r done(m)$. The receiver received the message.

Message Sending: $I_s done(m) \rightarrow FP(m)$.

Message Receipt: $B_r done(m) \rightarrow RE(m)$.

If the sender intends to send a message *m*, then the FP of the message (its communicative act) should hold and likewise for the receiver to believe the RE of the message on receiving it. Given the above system for beliefs and intentions, we specify the semantics of the speech-acts and states in the CNP in terms of the intentions of a message sender and the receiver's beliefs.

6 The Semantics of CNP Actions

We analyse the semantics of the most commonly used CAs, as given by FIPA in the SL language, and discuss the incorrectness of these semantics with respect to the intended meaning of the CA. As a remedy, we provide a simpler and more intuitive semantics for the CAs in table 2. Let *s* and *r* denote the sender and receiver respectively.

Table 2: **BIS Semantics for FIPA CAs**

CA	**FP**	**RE**
$s.inform(r, \phi)$	$B_s \phi$	$B_r \phi$
$s.propose(r, \gamma)$	$B_s(B_s I_r done(s.\gamma)$ $\rightarrow I_s done(s.\gamma))$	$B_r(B_s I_r done(s.\gamma)$ $\rightarrow I_s done(s.\gamma))$
$s.accept(r, \gamma)$	$B_s(B_r I_s done(r.\gamma)$ $\rightarrow I_r done(r.\gamma)),$ $B_s I_s done(r.\gamma)$	$B_r I_s done(r.\gamma)$
$s.reject(r, \gamma)$	$B_s(B_r I_s done(r.\gamma)$ $\rightarrow I_r done(r.\gamma)),$ $B_s \neg I_s done(r.\gamma)$	$B_r \neg I_s done(r.\gamma)$
$s.request(r, \gamma)$	$B_s I_s done(r.\gamma)$	$B_r I_s done(r.\gamma)$
$s.agree(r, \gamma)$	$B_s I_r done(s.\gamma)$, $B_s I_s done(s.\gamma)$	$B_r I_s done(s.\gamma)$
$s.refuse(r, \gamma)$	$B_s I_r done(s.\gamma),$ $B_s \neg I_s done(s.\gamma)$	$B_r \neg I_s done(s.\gamma)$
$s.cfp(r, \gamma)$	$B_s I_s($ $done(r.propose(s, \gamma)) \vee$ $done(r.refuse(s, \gamma)))$	$B_r I_s($ $done(r.propose(s, \gamma)) \vee$ $done(r.refuse(s, \gamma)))$

6.1 Speech Acts Semantics in BIS Semantics

Henceforth, we refer to the SL semantics as "FIPA semantics" and we refer to our proposed revised semantics as "BIS" (Belief Intention Semantics) semantics. As in FIPA SL, we give the preconditions (FP) and postconditions (RE) holding, respectively, before sending and after receiving a CA. In general, the FP of a CA includes the intention of the sender for conveying that CA and the RE of a CA includes the receiver's beliefs. To be compatible with the way FIPA semantics are expressed, using axiom (A2), we prefix the intentions of a sender with its beliefs about those intentions, (e.g. $B_iI_iB_j\alpha$ instead of $I_iB_j\alpha$). Table 2 presents our BIS semantics for most of the FIPA CAs. We discuss below the semantics of some of the salient CAs.

6.1.1 *s.inform(r,ϕ).*

s informs *r* that ϕ holds.

Fipa	FP:$B_s\phi\wedge \neg B_s(Bif_r\phi \vee Uif_r\phi)$
inform	RE: $B_r\phi$
BIS	FP: $B_s\phi$
inform	RE:$B_r\phi$

In the FIPA semantics, the FP includes the fact that the sender believes ϕ and the RE that the receiver believes ϕ. As mentioned in section 5.2, $B_s\phi$ is not strong enough since we need to represent the intention of *s* to send the message. We could express this as $B_sI_sB_r\phi$, the sender intends the receiver to believe ϕ. By the sincerity and the trust axioms, in the BIS semantics (as shown in table 2), the FP and RE respectively simplify to $B_s\phi$ and $B_r\phi$.

6.1.2 *s.propose(r,γ).*

s proposes *r* for *s* itself to do γ.

Fipa	FP: $B_s\alpha\wedge \neg B_s(Bif_r\alpha \vee Uif_r\alpha)$
propose	RE: $B_r\alpha$ where,
	$\alpha = I_rdone(<s,\gamma>) \rightarrow I_sdone(<r,\gamma>)$
BIS	FP: $B_s(B_sI_rdone(s.\gamma) \rightarrow I_sdone(s.\gamma))$
propose	RE: $B_r(B_sI_rdone(s.\gamma) \rightarrow I_sdone(s.\gamma))$

In the FIPA semantics FP, the sender *s* believes that if *r* intends *s* to do γ, then *s* will intend it. However, *s* may not know what *r* intends and therefore cannot consequently infer that it should intend to do γ. For *s* to be aware that *r* intends $done(s.\gamma)$, it must have received an accept to its proposal. As such, the FIPA semantics specifies *s* adopts an intention by being privy to the individual beliefs of *r*. In our BIS semantics, both the FP and the RE specify that *s* (the proposer) believes that *r* intends $done(s.\gamma)$, for *s* to adopt the same intention. Therefore, $B_sI_rdone(s.\gamma)$ is the premise for *s* to adopt the intention to do γ.

6.1.3 *s.accept(r,γ).*

s sends an accept proposal to *r*.

Fipa	FP: $B_s\alpha \wedge \neg B_s(Bif_r\alpha \vee Uif_r\alpha)$
accept	RE: $B_r\alpha$, where $\alpha = I_s done(<r,\gamma>)$.
BIS	FP: $B_s(B_r I_s done(r.\gamma) \rightarrow I_r done(r.\gamma))$,
accept	$B_s I_s done(r.\gamma)$
	RE: $B_r I_s done(r.\gamma)$

The FIPA semantics for *accept*ing a proposal do not consider the context of sending an accept. As FP, *s* believes that it intends *r* to do γ. There is no notion in either the FP or in the RE, that *s* is accepting a proposal that *r* must have sent. These FP and RE could also hold in other speech-acts such as *tell* and does not distinguish an *accept-proposal* from them. In our BIS semantics, we also include that both sender and receiver are aware that *r* sent a proposal previously and it is up to *s* to accept it. Our FP and RE also specify the context of the CA, this being an acceptance, there was a proposal before. The other part is the choice of the sender to intend the receiver to do γ.

The same remarks as for *accept-proposal* apply to the FIPA semantics for *reject-proposal*. The BIS semantics for *reject* can be found in table 2

6.1.4 *s.cfp(r,γ).*

s sends a call for proposal to *r* to do γ. In the FIPA semantics, the FP for a call for proposal includes that both sender and receiver intend for the receiver to perform the request. However, these intentions are premature given that *r* has yet to propose and *s* to accept for *r* to do γ. It does not leave the possibility for refusal or rejection. The rest of the semantics for *cfp* is so complicated that its meaning is unclear.

In our semantics, a call for proposal from *s* to *r* is equivalent to a request from *s* to *r* for *r* to make a proposal to *s*. Thus *s.cfp(r, γ)* is equivalent to *s.request(r, r.propose(s,γ))*.

Using our BIS semantics for *s.request(r, γ)*, as shown in table 2, we can specify a call for proposal by *s* as having FP $B_s I_s(done(r.propose(s,\gamma)) \vee done(r.refuse(s,\gamma)))$. This means that *s* intends that *r* either sends a proposal or refuses to do γ (because may be *r* cannot do γ). In turn, the RE of a call for proposal is that *r* believes *s* intends *r* to make a proposal or to refuse.

6.1.5 *s.refuse(r,γ).*

s sends a refusal to *r* for *r* to do γ. Again for the refusal CA, the FIPA semantics are obscure. For example, the FP given by the FIPA semantics is:
$B_s \neg Feasible(< s,\gamma >) \wedge B_s(B_r Feasible(< s,\gamma >) \vee$
$U_r Feasible(< s, act >)) \wedge B_s\alpha \wedge \neg B_s(Bif_r\alpha \vee Uif_r\alpha)$
where $\alpha = \beta \wedge \neg done(< i,\gamma >) \wedge \neg I_s done(< s,\gamma >)$.
β is the reason for the refusal and γ is the action being refused. The predicate *Feasible* is unclear and the formula α is hard to understand. In the BIS semantics, the precondition for a refusal is that the sender *s* believes that *r* intends *s* to do γ and *s* does not

intend to do so. The RE is that the receiver then believes that the sender does not intend γ.

6.2 Internal Actions

There are two internal actions in the CNP protocol — ***M.deliberate(C,A,p)*** from a *proposed* state and *i.p* from an *accepted* state. The process *i.p* expresses that agent *i* executes *p* and it semantics are given in terms of the semantics for extended PDL. In the process $M.deliberate(C, A, Act)$, the set *C* contains those agents which sent a proposal, *A* contains the set of agents whose proposals *M* will accept and *A*ct is the set of proposals subscripted with their corresponding proposer. A manager internally performs the $M.deliberate(C, A, Act)$ process to select which proposals to accept. The semantics can be found in table 3. The precondition requires that *M* believes that the set *C* contains those agents which sent a proposal. The postcondition specifies that after a *deliberate* action, the set *A* contains the contractors whose proposals *M* will accept and the set *(C-A)*, those who will be sent rejections.

7 The Semantics of CNP States

The interaction states of an IP specify the beliefs of an agent or group of agents. Given this, the interaction states in the CNP can be grouped into three types: public, shared and individual. Public states are believed by all the agents, shared states are mutually believed by a particular subset of the group, and individual states are the beliefs of one agent that others are unaware of. Internal actions are assumed to succeed. In the CNP, state s_i is equivalent to $done(a_i)$. For example, the state *cfped(M, G, p)* can be written as *done(M.cpf(G,p))* and likewise for the other states. For the sake of readability, in figure 2 we prefer to name a state as the past tense of an action leading to it, instead of a parameterised *done*. Let a group of contractors be denoted by *G* and the manager by *M*. Let $E_G\alpha$ be read as everyone in a group of agents, *G*, believes α. We specify below the semantics of the interaction states of the CNP, which together with the semantics of the processes in the CNP, constitute the semantics of the protocol.

7.1 Public States

The public states in the CNP are *cfped, timedout, open* and *closed* (see figure 2). These states are believed by the manager and all the contractors in *G*. The semantics of the state *cfped(M, G, p)* is that everyone in the group *GM*, (where $GM = G \cup \{M\}$) believes *done(M.cpf(G,p))*. That is:
$E_{GM}\forall i \in G(done(M.request(i, i.propose(M, p))))$. This entails that everyone believes the FP and RE of a call for proposal:
$E_{GM}I_s(done(r.propose(s, p)) \vee done(r.refuse(s, p)))$.

Similarly the semantics of the other public states are specified in terms of the beliefs of the group *GM*.

7.2 Shared States

In the CNP, these shared states are mutually believed by the manager and a contractor. For example, only the manager and a contractor sending a proposal believe and mutually believe that this particular contractor has sent the manager a proposal. The shared states of the CNP are *proposed, accepted, rejected, cancelled, refused, informed and failed* (see figure 2). Their semantics are given in terms of the beliefs of the manager and a contractor. We explain the semantics of the *proposed* and *accepted* states. The semantics of the other shared states are given in figure 3.

7.2.1 *proposed(C,p).*

Can be re-written as $\forall i \in C(done(i.propose(M, p_i)))$. The beliefs between *M* and each *i* in *C* are given below. The FP and RE of the *propose* CA are:
$B_M(B_i I_M done(i.p_i) \rightarrow I_i done(i.p_i))$ and
$B_i(B_i I_M done(i.p_i) \rightarrow I_i done(i.p_i))$ leading to the shared belief between *i* and *M*:
$\forall i \in C(E_{\{M,i\}}(B_i I_M done(i.p_i) \rightarrow I_i done(i.p_i)))$.

7.2.2 *accepted(A, p).*

Every contractor *i* in the set *A* has been sent an acceptance message from *M*. This can be re-written as $\forall i \in A(done(M.accept(i, p_i)))$.

From the FP and RE effect of *accept* CA, we have $\forall i \in A$:
$E_{\{M,i\}}(B_i I_M done(i.p_i) \rightarrow I_i done(i.p_i))$, *i* and *M* believe *i* previously sent a proposal to *M*.
$E_{\{M,i\}} I_M done(i.p_i)$ holds i.e. both *i* and *M* believe *M* has accepted *i*'s proposal, leading to the belief $E_{\{M,i\}} I_i done(i.p_i)$, both believe that *i* has adopted the intention to do γ.

7.3 Individual States

There are two individual states in the CNP process, *to-be-accepted* and *completed*, as the private state of the manager and a contractor respectively (see figure 2). The state *to-be-accepted(C,A, Act)* expresses the private belief of *M* that it will send an acceptance to all contractors in *C*. The following holds in the manager's beliefs for the *to-be-accepted(C,A, Act)* state:

- $B_M \forall i \in C(B_i I_M done(i.p_i) \rightarrow I_i done(i.p_i))$, *M* believes that all agents in *C* sent it a proposal.

- $B_M \forall i \in A(I_M done(i.p_i) \wedge I_M B_i I_M done(i.p_i))$. By axioms (A6) and (A1), this is equivalent to $B_M \forall i \in A(I_M done(i.p_i))$ *M* intends that all agents in *A* execute their proposal and *M* intends to let them know about its acceptance.

- $B_M \forall i \in (C - A)(\neg I_M done(i.p_i) \wedge I_M B_i \neg I_M done(i.p_i))$, likewise for the agents that *M* has decided to reject. Again by axioms (A6) and (A1), this is equivalent to $B_M \forall i \in (C - A)(\neg I_M done(i.p_i))$.

8 Proving Properties of the CNP

We validate our semantics for the CNP and its CAs by proving useful properties of the protocol. We do this by reasoning about the possible paths in the CNP. To this end, figure 5 shows an interpretation of the CNP from its start with a call for proposal to its completion with *refusals*, *cancellations* or *informs*. Figure 5 is derived from the CNP statechart in figure 2 and includes all possible paths in the execution of the CNP (apart from timeout). Let us refer to the states s_i in figure 2 as interaction states, and to the states S_i in figure 5 as execution states. An execution state, S_i, represents the beliefs of the manager and the contractors, given by the RE of the action leading to S_i, the FP of the next action, and the group beliefs in the interaction state s_i. The path S_0 to S_6 is the longest execution path in the CNP, and the beliefs holding at these states are given in table 3. For example, the action *cfp* leads to state S_1 in figure 5, and thus in table 3, the state S_1 gives the RE of *cfp*, the group's beliefs of the *cfped* state, and the FP of the next action *propose*. In table 3, let $GM = G \cup \{M\}$. Let α_i denote $B_i I_M done(i.p) \rightarrow I_i done(i.p)$ and let α_j denote $B_j I_M done(j.p) \rightarrow I_j done(j.p)$.

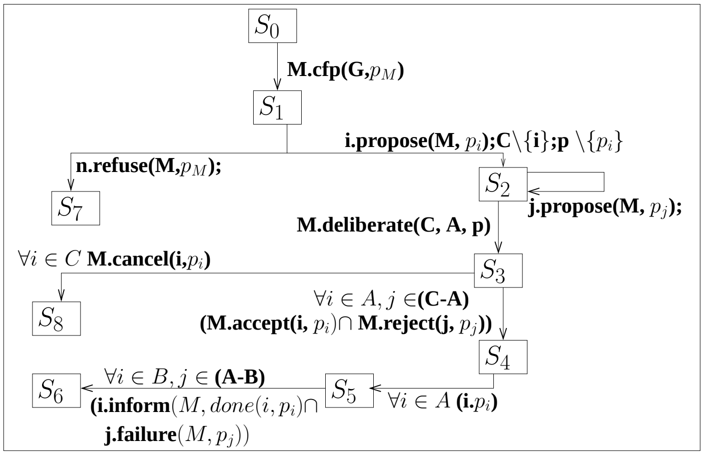

S_0
M.cfp(G,p_M**)**
S_1
i.propose(M, p_i**);C**$\backslash\{$**i**$\}$**;p** $\backslash\{p_i\}$
n.refuse(M,p_M**);**
S_2
j.propose(M, p_j**);**
S_7
M.deliberate(C, A, p)
$\forall i \in C$ **M.cancel(i,**p_i**)**
S_3
S_8
$\forall i \in A, j \in$**(C-A)**
(M.accept(i, $p_i$$)\cap$ **M.reject(j,** p_j**))**
S_4
$\forall i \in B, j \in$ **(A-B)**
S_6
S_5
$\forall i \in A$ **(i.**p_i**)**
(i.inform$(M, done(i, p_i)\cap$
j.failure$(M, p_j))$

Figure 5: Flowchart showing CNP Interpretation

8.1 Termination

We prove that any interpretation of the CNP will terminate.

Theorem 1. *In a CNP between a manager M and a group of contractors G, the formula (*$cfped(M, G, p_M) \rightarrow [Paths_{ALL}]closed$*) holds, where the process* $Paths_{ALL}$ *expresses all complete paths of execution in our framework.*

Proof. We prove that the *closed* state is eventually reached from the *cfped* interaction state. The simplest path is a timeout, which terminates. Let the process ρ_i lead to the interaction state s_i and the execution state S_i. The CNP defines that process ρ_i may be

Table 3: **Execution Paths from State S_0 to S_6**

S_0	FP_{cfp}	$\forall i \in G(B_M I_M(done(i.propose(M,p)) \vee done(i.refuse(M,p))))$
S_1	RE_{cfp}	$\forall i \in G(B_i I_M(done(i.propose(M,p)) \vee done(i.refuse(M,p))))$
	$State_{cfped}$	$\forall i \in G(E_{GM} I_M(done(i.propose(M,p)) \vee done(i.refuse(M,p))))$
	$FP_{propose}$	$B_i \alpha_i$
S_2	$RE_{propose}$	$B_M \alpha_i$
	$State_{proposed}$	$\forall i \in C(E_{\{M,i\}} \alpha_i)$
	$Pre_{deliberate}$	$\forall i \in C(B_M \alpha_i)$
S_3	$Post_{deliberate}$	$A \subseteq C \wedge \forall i \in A(B_M I_M B_i I_M done_i)$, $\forall j \in (C - A)(B_M I_M B_j \neg I_M done_j)$
	to-be-accepted	$\forall i \in C(B_M \alpha_i), \forall i \in A(B_M(I_M done_i \wedge I_M B_i I_M done_i))$, $\forall j \in (C - A)(B_M(\neg I_M done(j, p_j) \wedge I_M B_j \neg I_M done_j))$
	$FP_{M.accept(i,p_i)}$	$B_M \alpha_i, B_M I_M done(i.p_i)$
	$FP_{M.reject(j,p_j)}$	$B_M \alpha_j, B_M \neg I_M done(j.p_j)$
S_4	$RE_{M.accept(i,p_i)}$	$B_i I_M done(i.p_i)$
	$RE_{M.reject(j,p_j)}$	$B_j \neg I_M done(j.p_j)$
	$State_{accepted}$	$E_{\{M,i\}} \alpha_i, \forall i \in A(E_{\{M,i\}} I_M done(i.p_i))$
	$State_{rejected}$	$\forall j \in$ (C-A)$(E_{\{M,j\}} \alpha_j \wedge E_{\{M,j\}} \neg I_M done_j, E_{\{M,j\}} \neg I_j done_j)$
	$Pre_{i.p}$	Beliefs of *accepted* state
S_5	$Post_{\forall i \in A(i.p)}$	Beliefs of *completed* state
	$State_{completed}$	$\forall i \in B(B_i(done(i.p_i) \wedge I_i B_M done_i)$ $\forall j \in$ (A-B)$\wedge B_j(\neg done_j \wedge \neg I_j done_j \wedge I_j B_M \neg done_j))$
	$FP_{i.inform}$	$B_i done(i.p_i)$
	$FP_{j.failure(M,p_j)}$	$B_j \neg I_j done_j$
S_6	$RE_{i.inform}$	$B_M done(i.p_i)$
	$RE_{j.failure(M,p_j)}$	$B_M(\neg done(j.p_j) \wedge \neg I_j done_j)$
	$State_{informed}$	$\forall i \in B(E_{\{M,i\}} done(i.p_i))$
	$State_{failed}$	$\forall j \in (A - B)(E_{\{M,j\}}(\neg done_j \wedge \neg I_j done_j))$

followed by the process ρ_{i+1}. Let the notations in table 3 be used (e.g. for α_i). Proving termination for all processes in $Paths_{ALL}$ implies proving that all paths in figure 5 terminate. Thus, we prove that all actions in the paths in figure 5 are feasible in their source state (i.e, the FP of all processes ρ_{i+1} may hold after action ρ_i and in interaction state s_i). The premise is that the CNP has been started with a call for proposals. The RE_{cfp} (RE of cpf) holds, stating that contractors should either reply with a refusal or a proposal. But this is what is required by the CNP to trigger the next state, so the process proceeds to the state *proposed* or *refused*, since the RE_{cfp} renders it possible. Since *refused* is a sub-state of *closed*, all paths from *refuse* terminate. So using table 3, we now prove that the paths following a *propose* action terminate. That is, for all actions ρ_i, after execution state S_1, the $FP_{\rho_{i+1}}$ is possible from the interaction state s_i and the RE_{ρ_i}. This can be seen in table 3 where the pre-condition of *deliberate* holds from the $RE_{propose}$.

The FP of the action $\forall i \in A(M.accept(i, p_i))$ holds since the state *to-be-accepted* includes the belief $\forall i \in C(B_M \alpha_i)$ and $\forall i \in A(B_M I_M done_i)$.

Similarly the FP of the *reject* action holds from the beliefs in the *to-be-accepted* state. Thus, both acceptance and rejection processes can occur. Then the pre-condition of the next action $i.p$ hold by virtue of being the beliefs of the resulting interaction state from an *inform* or *failure*.

From table 3 it can be seen that the FP of both *accept* and *reject* can be derived from the state *completed*, leading to the sub-states of *closed*. □

Corollary 1. *After a call for proposal, the FP of all actions may hold from the FP and RE of the previous action. That is,* $FP_{\rho_{i+1}}$ *from* FP_{ρ_i} *and the* RE_{ρ_i}

We can also show there are no deadlocks in the CNP interpretation. The corollary holds because when proving theorem 1, we proved $FP_{\rho_{i+1}}$ is possible from the interaction state s_i and the RE_{ρ_i}. From our semantics, interaction state s_i is itself defined from the FP and RE of the action ρ_i leading to it.

8.2 A Failed or Succeeded CNP

We can also show that a CNP always terminates with the beliefs of whether the CNP process has satisfied the goal of the interaction:

Theorem 2. *The interpretation of the CNP terminates with either the shared belief between a manager and a contractor i of either* $done_i$ *or* $\neg done_i$*, or the group beliefs of* $\neg done$*.*

Proof. By theorem 1, a CNP interpretation always terminates. $\neg done$ obviously holds in the *timedout* state. Terminal states are *refused, timedout, cancelled* and *(informed* $\wedge$ *failed)*. Section 7 and table 3 both show that $\neg done$ holds in states *refused* and *cancelled*. It can also be seen that in the *accepted* and *rejected* states, $E_{\{M,i\}}done_i$ and $E_{\{M,j\}}\neg done_j$ respectively hold. Thus, all accepted contractors *i* believe $done_i$ and all rejected contractors *j* believe $\neg done_j$, while the manager appropriately believes $done_i$ and $\neg done_j$. □

8.3 Consistent Joint Beliefs in a Group

Consistent joint belief about μ in group *G* entails that everyone in *G* believes μ and no-one believes $\neg\mu$. Below we show that for public and shared states, there are some beliefs that are consistent between the agents in the group (for public states) or sub-group (for shared states). Thus, there is a state of affairs which every agent in the group (or sub-group) believes.

Theorem 3. *In the CNP interpretation, for all public and shared states, there are some consistent joint beliefs in, respectively, the group or sub-group. That is, for all states public to G, and states shared between* G_{sub}*:*

$$\exists\mu(E_G\mu \wedge (\neg\exists i \in G(B_i\neg\mu)))\wedge$$
$$\exists\beta(E_{G_{sub}}\beta \wedge (\neg\exists i \in G_{sub}(B_i\neg\beta)))$$

Proof. In the public *cfped* state, it can be seen that:
$\forall i \in G(E_{GM}I_M(done(i.propose(M,p)) \vee done(i.refuse(M,p))))$.
Regarding shared states in the CNP, they have been formulated in table 3 in such a way to show the joint beliefs between the manager and each contractor *i*. For example, $\forall i \in B(E_{\{M,i\}}done(i.p_i))$ in the *informed* state. Since individual beliefs are consistent, then if everyone in that sub-group believes $done(i.p_i)$, no-one will believe the contrary. The same is true for the other shared states *accepted, failed, rejected* and *proposed*. □

9 Conclusions and Future Work

We believe there is a lack of consensus on a suitable ACL for agent interaction because of the bewildering array of approaches to formalising an ACL's semantics. Likewise, there is a similarly strong need for formal specification and verification of interaction protocols and their semantics. To highlight these needs and in attempting to satisfy them, this paper uses the contract net protocol as a non-trivial case study. In our framework, we formulate axioms for reasoning about an agent's beliefs about its intentions and present a simplified and revised semantics for the FIPA communicative acts that appear in the contract net protocol. We accompany these ACL semantics with those of the states and internal actions in the contract net protocol in order to obtain (for the first time) a complete semantics for that protocol. In so doing, we can prove properties when interpreting the protocol such as termination and consistency in joint beliefs. Even though the case study has raised several issues about ACL and IP semantics, it is still incomplete and we intend that future work analyses other interesting open issues. Future work includes relaxing the assumptions detailed in section 5.2 and analysing properties such as liveness, completeness, complexity and decidability. A denotational semantics for the speech-acts can be specified in addition to the given BDI semantics and these semantics may be combined with our previous work [9] in modeling agent interactions in imperfect communication environments for an analysis of the performance of agent interactions in realistic environments.

References

[1] J. L. Austin. *How to Do Things with Words.* Oxford University Press, 1962.

[2] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.

[3] Y. Finin, T.and Labrou and J. Mayfield. KQML as an agent communication language. In *Software Agents, Jeff Bradshaw (Ed)*, pages 267–273, 1995.

[4] Foundation for Intelligent Physical Agents, http://www.fipa.org. *FIPA Agent Communication Language Specification*.

[5] R. Goldblatt. *Logics of Time and Computation*. CSLI, 1987.

[6] F. Guerin and J. Pitt. Agent communication frameworks and verification. In *AAMAS 2002 Workshop on Agent Communication Languages, Bologna, LNCS volume 2650*, 2002.

[7] M. Nowostawski, M. Purvis, and S. Cranefield. A layered approach for modelling agent conversations. In *Proc. 2nd Int. Work. on Infrastructure for Agents, MAS, and Scalable MAS,*, 2001.

[8] S. Paurobally. *Rational Agents and the Processes and States of Negotiation*. PhD thesis, Imperial College, 2002.

[9] S. Paurobally, R. Cunningham, and N. R. Jennings. Ensuring consistency in joint beliefs of interacting agents. In *Proc. AAMAS*, pages 662–669, 2003.

[10] J. Pitt and A. Mamdani. Communication protocols in mas. In *Work. on Specifying and Implementing Conversation Policies*, pages 39–48, 1999.

[11] D. Sadek. A study in the logic of intentions. In *3rd Conf. on Principles of Knowledge Representation and Reasoning*, pages 462–473, 1992.

[12] J. R. Searle. *Speech acts: An essay in the philosophy of language*. Cambridge University Press, 1969.

[13] M. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, 1998.

[14] I. Smith, P. Cohen, J. Bradshaw, Greaves M., and H. Holmack. Designing conversation policies using joint intention theory. In *ICMAS*, pages 269–276, 1998.

[15] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1981.

[16] M. Wooldridge. Semantic issues in the verification of agent communication languages. *Journal of Autonomous Agents and MAS*, 3(1):9–31, 2000.

[17] P. Yolum and M. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proc. AAMAS*, pages 527–534, 2002.

Modelling Resource Bounded Reasoners: An Example

Mark Whitsey[*]

School of Computer Science
University of Nottingham
Nottingham, NG8 1BB, UK
mtw@cs.nott.ac.uk

Abstract

We present Timed Reasoning Logics (TRL) as a framework that allows one to model multiagent systems over time without assuming agents are logically omniscient. TRL can be seen as a combination of syntactic epistemic logics and temporal logics. This paper is primarily concerned with illustrating TRL concepts using a propositional natural deduction-style reasoner, which derives consequences of its beliefs over time. We show the logic used to model the reasoner, which we call TRL(ND), is sound, complete and decidable.

1 Introduction

Timed Reasoning Logics (TRL) [1] are a family of logics that have been developed to address the problem of modelling time bounded reasoners, i.e. agents that are able to produce plans or derive consequences of their beliefs but take time to deliberate. Most research in epistemic and doxastic logics (see, for example, [6, 13, 15, 16, 17, 19, 20, 22, 23, 25]) makes the strong assumption that, whatever reasoning abilities an agent may have, the results of applying those abilities to a given problem are available immediately. Agents that reason using a classically complete set of logical rules, such as the rules of classical natural deduction, cannot then believe a formula without believing all logical consequences of that formula. In some situations, this is a reasonable assumption to make. An agent may do a

[*] This research is supported by the AHRB. Thanks to Natasha Alechina, Brian Logan and Thorsten Altenkirch for commenting on earlier drafts of the paper.

very simple kind of deliberation in a non-time-critical environment, for example, and we may safely ignore a small delay involved in deliberation.

However, there are many cases in which the assumption that results of deliberation are available immediately is not permissible. One example in which the time taken to deliberate is of critical importance is that of planning in a dynamic environment: an agent may be able to produce a perfectly good plan to reach its goal but, if planning takes longer than the time it takes the environment to alter significantly, the result may be irrelevant. Another example involves agents that verify cryptographic protocols. An agent intercepting a coded message may have all the "inference rules" necessary to break the code, eventually; but, in practice, it may take millennia to actually derive the key and decode the message. Again, in modelling a theorem-proving agent whose sole purpose is to check whether a formula is a tautology, to assume that it already knows all tautologies defeats the purpose of modelling. In general, TRL is concerned with modelling resource bounded agents and proving results of the form *agent i is capable of reaching conclusion ϕ within time bound t*.

Note that incorporating impossible and/or incomplete worlds (e.g. [11], [16], [21]) does not help matters. Impossible worlds are ones at which both a formula and its negation hold, for some formulas; incomplete worlds are ones at which neither hold. Such approaches have the effect of allowing a formula to be assigned various subsets of the set $\{true, false\}$ (and not just the classical singleton sets). Such epistemic logics have *relevance logic* as their propositional basis. Alternatively, ϕ may be assigned any subset of $\{true, false\}$ except $\emptyset$; this results in an epistemic logic with *paraconsistent logic* as its propositional basis. Both relevance logic and paraconsistent logic have a subclassical consequence relation; the resulting epistemic logics therefore model an agent's beliefs as being closed under that relation. Although an agent's beliefs will therefore be a proper subset of the beliefs ascribed according to the standard possible worlds paradigm [12], the considerations addressed above apply just as forcefully in these cases.

In this paper, we illustrate the basic concepts of TRL by considering a model of a classical reasoner as an example. The reasoner we model derives formulas in a step-by-step way using a system of natural deduction. The reasoner begins with a set of initial formulas Γ and derives consequences over time; we say that a formula is believed when an agent has derived it from the initial set. Now, a formula that is easily derivable from Γ (i.e. for which the derivation is short) should take our agent less time to add to its beliefs than formulas whose derivation from Γ requires more computational effort. Since our purpose is to model the agent's derivations in a step-by-step manner, it would clearly be wrong to assume that, in believing some formula ϕ at time t, our agent also believes all classical consequences of ϕ at t.

Reasoning in natural deduction requires assumptions to be made. Our model of the natural deduction reasoner comprises many helper agents subordinate to the main reasoner, where assumptions are modelled as the beliefs of these subordinate agents. Our principal reasoner may not believe ϕ, but it can communicate with a subordinate agent who does believe ϕ (as well as everything the principal agent believes and nothing else) to discover what would happen to it's beliefs if ϕ were the case. For example, if the subordinate agent derives a contradiction from ϕ, then the principal agent can infer $\neg\phi$. This process can be iterated, to model assumptions within assumptions. We formulate this type of principal agent–subordinate agent interaction along the lines of a *reductio ad absurdum* rule. For simplicity, we restrict this exposition to a language whose only connectives are $\neg$ and $\wedge$; however, we could just as easily use subordinate agents to introduce implications or to eliminate disjunctions in a similar way to introducing negation. Although we model a classical reasoner, an intuitionistic reasoner can be modelled just as easily (by replacing *reductio ad absurdum* and double negation elimination with rules for the introduction and elimination of implication and disjunction, for example).

We allow agent each agent to see inside the states of its subordinates in order to extract the information it needs. This assumption is used here as a simplifying feature; in general, there is nothing to stop TRL agents having private (unobservable) beliefs which can only be communicated to certain other agents, or even to none at all. In other words, although the specific example of TRL provided in this paper requires a simplistic notion of agent communication, the concepts developed allow for far more realistic models of communication in multiagent systems.

It should be emphasised that TRL is not conceived as an automated theorem proving tool; the example involving a natural deduction reasoner is used here merely to demonstrate how a simple system of time-bounded agents can be successfully modelled, i.e. without assuming an agent's beliefs to be closed under some consequence relation at each timepoint. It is not the particular formalisation of reasoning with assumptions that is of most interest here; attention should be given to the general framework of modelling step-by-step reasoning. That is not to say that the framework developed for modelling reasoning with assumptions is of no interest. The same approach could be used to distinguish an agent's firmly held beliefs from working hypotheses in a formalisation of belief revision, for example (although we would then be describing different sections of a single agent's architecture, rather than distinct agents).

TRL represents agents as repeatedly executing some fixed program at each tick of the clock, matching a set of condition-action rules against the contents of its working memory and firing a subset of these rules. Following standard rule-based system terminology, the process of deciding which subset of rule instances is to be fired at any given cycle is known as *conflict resolution*. TRL is capable of modelling various rule application and conflict resolution strategies. The reasoner we model in this paper fires every rule instance in its conflict set exactly once on each cycle; we call this the *all rules at each cycle* strategy. Accordingly, we restrict our exposition of TRL to those subsystems which use the the *all rules at each cycle* strategy exclusively.[1]

One of the advantages of TRL is its separation of temporal and epistemic information: its rules either affect how an agent's internal state changes over time, or affect what agents believe relative to their own or other agent's beliefs (section 4.2 below). The former kind of rule may incorporate features of more familiar temporal logics, such as operations on timepoints, whereas the latter can admit features of epistemic logics, such as Konolige's Deduction Model of Belief [13]. We may, for example, model an introspective agent, who knows what he knows, in the sense that he can introspect to discover whether he knows a particular formula, but for whom this process takes time, and so the results of introspection are not available immediately.

A major difference between TRL and approaches reliant upon possible worlds semantics is that TRL represents an agent's beliefs *syntactically*. Whilst this has the advantage of allowing one to differentiate between syntactically different but semantically indistinguishable formulas, it has been claimed that such accounts do not preserve meanings of believed formulas in the right way. An agent modelled in a syntactic account may believe, e.g., $p \wedge q$ but not believe $q \wedge p$ and so, it is claimed, the meaning of '$\wedge$' as conjunction is not preserved.[2]

Indeed, an agent with appropriate rules for conjunction and appropriate capacities should, according to any good model, be able to switch the order of conjuncts in a believed conjunction. But consider two logically equivalent formulas $\phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4 \ldots \wedge \phi_{n-1} \wedge \phi_n$ and $\phi_2 \wedge \phi_1 \wedge \phi_4 \wedge \phi_3 \ldots \wedge \phi_n \wedge \phi_{n-1}$. To arrive at belief in the latter on the basis of belief in the former, an agent may need to permute $n/2$ conjuncts which, for large values of n, will take some degree of computational effort; we could not assume that the permutations take place instantly. In fact, we should be able to model the fact that the amount of effort required by such an activity increases as n does. This is precisely the kind of example to which TRL is well suited.

In asserting that *agent i believes that ϕ (at time t)*, we mean that ϕ contributes towards the characterisation of the internal state of agent i at t and, given other appropriate information, gives us information pertaining to how i will act. We therefore take one of the rôles of ascribing a belief to an agent to be a (partial)

[1] We call such systems TRL(STEP), as this is the strategy implicit in step logic models [4]: see section 2 below.

[2] Syntactic accounts of belief are sometimes referred to as *sentential* accounts in the philosophical literature, for they relate an agent to a believed sentence, rather than the believed proposition expressed or state of affairs depicted by that sentence.

description of how it will act in certain circumstances.

The remainder of this paper is organised as follows. We begin by reviewing related research (section 2) and providing a brief outline of TRL (section 3). Section 4 describes our example of TRL, namely our natural deduction reasoner, and explains how we model it in TRL. We also show that the resulting logic, TRL(ND), is sound, complete, and decidable. We then briefly comment (section 5) on how this approach is preferable to other comparable logics for modelling resource bounded agents and suggest several avenues for further research to take.

2 Other Relevant Work

The literature contains many attempts to provide a logic of limited or restricted reasoning by attempting to avoid the *logical omniscience* problem: if ϕ logically implies ψ and an agent believes ϕ, then it must believe ψ too (as a consequence, an agent must believe all tautologies).[3] Levesque's [16] logic of implicit and explicit belief restricts an agent's explicit beliefs (the classical possible worlds notion) by allowing non-classical (either incomplete or impossible) worlds to enter an agent's epistemic accessibility relation. Although agents need not then believe all classical tautologies, they are nevertheless modelled as being logically omniscient in *relevance logic*, i.e. believing ϕ results in the the belif that ψ whenever $\phi \rightarrow \psi$ is valid in relevance logic. Such agents could hardly be described as resource bounded in our sense (see [26] for more on this issue).

In [7], Fagin & Halpern propose an alternative approach to restricting possible worlds semantics which involves a syntactic *awareness* filter, such that an agent only believes a formula if it (or its subterms) are in his awareness set. Agents are modelled as perfect (i.e. logically omniscient) reasoners whose beliefs are restricted to some syntactic class compatible with the awareness filter. Belief in all classical tautologies is again avoided but, as Konolige notes, this approach is "is no more powerful than current sentential [i.e. syntactic] logics, and can be re-expressed in terms of them" [14, p.248].

Konolige's Deduction Model [13] represents beliefs as sentences belonging to an agent's belief set **B**, which is closed under the agent's deduction rules. A deduction model assigns a set of rules to each agent, allowing representation of agents with differing reasoning capacities within a single system. Resource boundedness is thus accommodated by modelling an agents as having rules that are restricted in some respect (a rule may only be applicable, for example, if matched antecedent formulas are less than a certain length).

[3] The terminology *logical omniscience* was first introduced by Hintikka [12].

However, all these accounts provide what [2, p.1] call a "final tray model" of belief: they tell us what a set of agents will believe after an indefinitely long period of deliberation. Step logic [4], on the other hand, is an attempt to model agents' reasoning as a process situated in time, by incorporating the notion of a *step* into its rules of inference. Such rules are of the form:

$$\frac{t\colon \quad \phi_1, \ldots, \phi_n}{t+1\colon \qquad \psi}$$

where t is a variable ranging over steps. As such, step logics can be viewed as particular cases of Gabbay's Labelled Deduction Systems [8]. The rules are intuitively interpreted as: if the agent's set of beliefs contains $\phi_1, \ldots, \phi_n$ at step t, then it will contain ψ at step $t+1$.[4] Agents may lose as well as gain beliefs from step to step; if required, monotonicity must be stipulated using a rule.

Within this general framework, particular step logics are characterised according to three mechanisms: *self-knowledge*, i.e. an agent knowing what it does or does not believe; *time*, allowing an agent to reason about its reasoning over time; and *retraction*, or the ability to handle contradictions. The simplest such logic is named SL_0 and features none of these mechanisms. The other logics (SL_1 to SL_7) studied in [5] become more progressively more complicated as their index increases, each featuring a different combination of these mechanisms.

No semantics is given in [4] for any of these logics. A semantics for SL_5 is considered in [18] and [3], based on Montague's *neighbourhood structures*.[5] This has the effect of re-introducing a version of the logical omniscience problem: equivalent formulas cannot be distinguished, and so a formula cannot be believed without believing all equivalent formulas. This appears to be an unavoidable feature of neighbourhood semantics and of accounts whose semantics is based on possible worlds in general; see [26] for a more detailed critique of this approach.

[10] provides a proper semantics for step logics which does not suffer from logical omniscience. However, their first-order theories are undecidable; they also fail to separate the temporal from the epsitemic/doxastic aspects of reasoning, as TRL does. We should be able enrich a theory with a calculus on timepoints, for example, without affecting its epistemic properties.

The framework we propose borrows several notions from *context logics* (see, for instance, [9]). Contexts may be thought of as localised theories, each containing a language and a set of inference rules, but intuitively represent the points of view

[4] Note that an agents beliefs are represented as derived formulas in this language; a metalanguage containing a belief predicate can then be defined such that $\texttt{bel}(t, \ulcorner\phi\urcorner)$ holds just in case ϕ is derived at step t in the object language.

[5] Since the key idea is to model time, *timelines* replace sets of worlds; worlds thus become time point constants, such that timelines are *rays*, i.e. infinite in one direction only.

of a set of agents. An example given in [9] involves two contexts, each of which describes a different agent's perspective of the same object; neither view uniquely determines the other, but some combinations of views are impossible.

Contexts are connected to one another through inference rules known as *bridge rules*. The semantic counterpart of a context is the *local model*, defined as non-empty sets of classical models.[6] Contexts are thus closed under the classical consequence relation and, so interpreted, are not suitable for modelling resource bounded agents in our sense.

However, one can take the idea of using bridge-style rules to establish relationships between contexts, understood as sets of formulas, without recourse to classical localised models. One application is provided by the rules of step logic, which establish relationships between the an agent's set of beliefs at t and its beliefs at $t + 1$. Another application, much discussed in the literature on context logics, is in modelling the links between agents in a multi-agent system. In presenting our framework of Timed Reasoning Logics in the following section, we combine these two ideas to give us a system for modelling multiple resource bounded agents whose reasoning is modelled as a process situated in time.

3 The General Approach: Timed Reasoning Logics

The goal of the Timed Reasoning Logics (TRL) project [1] is to develop a system that can model the internal state of any number of agents at each stage of deliberation. One criterion of success is that, if we were to stop an agent at some stage of its deliberation and inspect its internal state, we should find there precisely the formulas that our framework tells us we should.

The internal state of an agent i at time t is modelled by a finite set of arbitrary formulas $\{\phi_1, \ldots, \phi_n\}$, which we call a *TRL local state*. Each local state is labelled with the time t and agent name i that it models. Note that we do not require local states to be internally consistent (in fact, states containing contradictory formulas are vital for the way we represent reasoning with assumptions in TRL(ND): see section 4 below). In this sense, TRL local states are more akin to the belief sets of Konolige's Deduction Model [13] than the local models of context logic, which are closed under the classical notion of consequence. But whereas an agent's belief set **B** in the Deduction Model is closed under that agent's deduction rules, which has the effect of modelling the agent as deducing new beliefs instantaneously, TRL local states are closed under no rules whatsoever. Rules play their rôle in TRL by

[6] This allows for modelling incompleteness: in a given context, the agent may accept neither ϕ nor $\neg\phi$

linking local states together; i.e. by establsihing what one state must look like on the basis of what another contains.

A TRL(ND) model is then a set of local states, each indexed by an element of the index set $I = A \times \mathbb{N}$. Intuitively, TRL models can be viewed as a grid, with timepoints along its horizontal axis and agent names forming the vertical axis. At the intersection of t on the horizontal axis and i on the vertical axis, we find the local model labelled with i and t, which is our representation of the contents of that agent's internal state at that time.

Each model has an obs function and a set of inf_i functions (one for each agent i). Intuitively, obs models communication between agents (which, in the case of TRL(ND), amounts to simply looking into another agent's state at a timepoint; see section 4 below). Note that, since different agents may use different languages, or even augment their language from timepoint to timepoint (e.g., by aquiring new constants through observing the environment), we write $\mathcal{L}^i_t$ to denote agent i's language at timepoint t. obs takes an agent i and a timepoint t as its arguments and returns a finite set of formulas in the agent's language at that timepoint, $\mathcal{L}^i_t$, which are added to the agent's local state instantaneously (i.e. at the same timepoint). Each function inf_i models agent i's computation of a new state, mapping a finite set of formulas in i's language at t, $\mathcal{L}^i_t$, to formulas in the language $\mathcal{L}^i_{t+1}$.

TRL has two types of rule: those that model an agent's internal deductive process, and those that model how agents interact with one another (see section 4.2 below). We call the former type *internal rules*, which are of the form:

$$\frac{(i,t) : \phi_1 \ldots (i,t) : \phi_n}{(i, t+1) : \psi}$$

Semantically, we model these rules using a function inf_i for each agent i (see definition 1 below) and, for a set of TRL rules R, write R^{inf} to denote the set of internal rules.

We call the latter type *communication rules*, which have the form:

$$\frac{(i,t) : \phi}{(j, t+1) : \psi}$$

For simplicity, we use a very basic notion of communication between agents: an agent simply looks into another's state and observes the formulas stored there. We model communication using an obs function, and so communication rules correspond to constraints on obs. We therefore write R^{obs} to denote the set of communication rules.

4 Modelling the ND Reasoner in TRL

In this section, present the example of the title; namely, a model of a classical natural deduction style reasoner. We do so in a logic we call TRL(ND). Our aim in this section is twofold. Firstly, we show that the model is faithful to the reasoner, in the sense that theorems are derived in a step-by-step manner, i.e. beliefs are not closed under a consequence relation at any *particular* timepoint, but any classical tautology or consequence of a set of initial formulas Γ will be derived at *some* timepoint. Secondly, we show TRL(ND) to be sound and complete with respect to TRL semantics; and that our notion of derivability in TRL(ND) is decidable.

4.1 The Natural Deduction Reasoner

Our reasoner is an agent which can reason in the following system of natural deduction for propositional logic:

$$\frac{\phi \qquad \psi}{\phi \wedge \psi}\wedge_{int} \qquad\qquad \frac{\phi \wedge \psi}{\phi}\wedge_{elimL}$$

$$\frac{\phi \wedge \psi}{\psi}\wedge_{elimR} \qquad \frac{\phi \quad \neg\phi}{\bot}\bot \qquad \frac{\neg\neg\phi}{\phi}\mathrm{DN}_{elim} \qquad \frac{\begin{matrix}[\phi \\ \vdots \\ \bot]\end{matrix}}{\neg\phi}\mathrm{RAA}$$

The main issue we have to deal with in modelling ND is the problem of modelling assumptions. The way we have chosen to do this in TRL is to model a hierarchy of agents, each with the same reasoning abilities, where each agent is labelled with a finite sequence of formulas from the language of the reasoner. The agent labelled with the sequence $\langle \phi_1, \ldots, \phi_n \rangle$ represents a series of assumptions, each within the scope of the previous one, beginning with the assumption that ϕ_1 holds and continuing until the assumption that ϕ_n holds has been made.

The agents in the system are linked by TRL communication rules which model how the ND reasoner makes and later closes assumptions. Observation rules also force a subordinate agent $\langle \phi_1, \ldots, \phi_n \rangle \psi$ to have same beliefs as the agent $\langle \phi_1, \ldots, \phi_n \rangle$, the agent one step up the hierarchy, plus the extra belief that ψ. Accordingly, we write $i\psi$ as the name of the agent representing the ND reasoner making an assumption that ψ holds, within the scope of whatever assumptions have been previously modelled by agent i. If the agent named $i\psi$ derives a contradiction, it then returns this result to agent i at the next step, who will then believe the negation of ψ.

For example, suppose our agent wants to prove $\neg(\phi \wedge \neg\phi)$, by first assuming $\phi \wedge \neg\phi$ and then deriving a contradiction. In our model, there will be an agent named $\phi \wedge \neg\phi$ at timepoint 0, who will apply first $\wedge_{elimL}$, then $\wedge_{elimR}$, followed by the $\bot$ rule to derive $\bot$ at timepoint 3. At timepoint 4, the agent labelled with the empty sequence will observe that $\phi \wedge \neg\phi$ has derived $\bot$ and accordingly believe $\neg(\phi \wedge \neg\phi)$. Since this agent represents beliefs with no assumptions, we say our modelled agent has indeed derived $\neg(\phi \wedge \neg\phi)$.

We can therefore consider the TRL(ND) system to be a (somewhat unusual) multi-agent system. Our model of the ND reasoner is this entire multi-agent system. However, when we consider what our modelled agent actually derives, we restrict out attention those formulas found in the states of the agent labelled with the empty sequence, which we denote $\cdot$, i.e. the formulas derived without any assumptions.

In the next section, we give details of our model of ND in TRL(ND). Although the definitions and proofs we give are specific to TRL(ND), much of what we say goes for the TRL system in general.[7] Wherever possible, we indicate where this is the case.

4.2 TRL(ND) Syntax

In this particular example of TRL, all agents reason in the same propositional language $\mathcal{L}$, which does not change over time. (As noted above, TRL allows for agents to have unique languages which may change over time). For every agent i and time t, we can thus write $\mathcal{L}$ in place of $\mathcal{L}^i_t$ to denote that agent's language. The well-formed formulas of $\mathcal{L}$ (which represent the formulas found in the internal states of each agent) are defined over a set of primitive propositions Φ along with $\bot$, $\neg$ and $\wedge$ in the usual way. Other formulas, such as implications, may also be added if appropriate definitions are given, i.e. $\phi \to \psi := \neg(\phi \wedge \neg\psi)$; we only provide rules for $\bot$, $\neg$ and $\wedge$ here.

In TRL(ND), we use labelled formulas in a language $\mathcal{L}^{TRL}$ to distinguish between unlabelled formulas belonging to internal states of distinct agents at distinct timepoints. Given a set of logical timepoints T (which we will assume to be the set of natural numbers) and a non-empty finite set of agents (each represented as a finite sequence of formulas) $A = \{1, \ldots, i, \ldots\}$, $(i, t) : \phi$ is a well-formed labelled formula of $\mathcal{L}^{TRL}$ for some $t \in T$ and $i \in A$ whenever ϕ is a well-formed formula of $\mathcal{L}$.

Theorems of TRL(ND) are obtained from the following rules, each of which is applied where applicable at every timepoint:

$$\frac{(i,t) : \phi \quad (i,t) : \psi}{(i,t+1) : \phi \wedge \psi}\wedge_{int} \qquad \frac{(i,t) : \phi \wedge \psi}{(i,t+1) : \phi}\wedge_{elimL} \qquad \frac{(i,t) : \phi \wedge \psi}{(i,t+1) : \psi}\wedge_{elimR}$$

[7] In fact, much of what is said goes for the system TRL(STEP), the subsystem of TRL which exclusively uses the *all rules at each step* rule application strategy, of which TRL(ND) is a particular example.

$$\frac{(i,t):\phi \quad (i,t):\neg\phi}{(i,t+1):\bot}\bot \qquad \frac{(i,t):\neg\neg\phi}{(i,t+1):\phi}\text{DN}_{elim} \qquad \frac{(i\phi,t):\bot}{(i,t+1):\neg\phi}\textsc{raa}$$

Most of these rules (which correspond to the rules of natural deduction above) are self-explanatory. In RAA, $i\phi$ is the name of an agent which is just like i (i.e. believes whatever i believes) but in addition believes a formula ϕ.

In addition, each agent reasons monotonically (since the reasoner ND does too):

$$\frac{(i,t):\phi}{(i,t+1):\phi}\textsc{mon}$$

Suppose the ND reasoner is given a set of unlabelled formulas Δ from which it is to infer new formulas. In TRL(ND), derivations always use labelled formulas and so the TRL(ND) derivation will begin with a set of labelled formulas Γ, which is formed by labelling each formula in Δ. To keep our account general, we should not assume that all formulas in Δ are introduced in the derivation at timepoint 0. Indeed, we allow formulas in Γ to have any label (i,t); consequently, we need to ensure that, at every step, all subordinate agents $i\phi$ inherit whatever beliefs i has at that step. We also have to make sure that $i\phi$ believes ϕ:

$$\frac{(\langle\psi_1\ldots\psi_n\rangle,t):\phi}{(\langle\ldots\psi_1\ldots\psi_n\ldots\rangle,t):\phi}\textsc{Inherit}^* \qquad \frac{}{(\langle\ldots\phi\ldots\rangle,t):\phi}\textsc{ax}$$

* where $(\langle\psi_1\ldots\psi_n\rangle,t):\phi\in\Gamma$.

Together, we call the set of these rules R_{ND}. With the exception of RAA, INHERIT and AX, which are communication rules (R^{obs}_{ND}), the rules in R_{ND} are internal rules (R^{inf}_{ND}).

A labelled formula $(i,t):\phi$ is *derivable* from a set of labelled formulas Γ using R_{ND}, written $\Gamma\vdash(i,t):\phi$, if there is a sequence of labelled formulas $(i_1,t_1):\phi_1,\ldots,(i_n,t_n):\phi_n$ such that:

1. each formula in the sequence is either a member of Γ, or is obtained from Γ by one of the inference rules in R_{ND}; and
2. the last labelled formula in the sequence is $(i,t):\phi$, namely $(i_n,t_n):\phi_n = (i,t):\phi$.

Note that there is a special case of derivation, namely of a formula $(\cdot,t):\phi$. Such derivations tell us what the modelled ND agent believes: an ND agent starting with a set Δ of unlabelled formulas believes ϕ whenever $\exists t.\Gamma\vdash(\cdot,t):\phi$, where $\Gamma=\{(\cdot,0):\psi\mid\psi\in\Gamma\}$.

Any formula that can be derived using the standard propositional natural deduction rules given in section 4.1, for which we write $\vdash_{nd}$), can be derived as a formula labelled $(\cdot, t)$ for some t in TRL(ND) and *vice versa*. This establishes that the reasoner modelled by TRL(ND) can, given enough time, derive any consequence of the formulas Δ it begins with, and nothing more:

Theorem 1 *Let Δ be a set of unlabelled propositional formulas, and ϕ an unlabelled propositional formula in the agent's language. Then $\Delta \vdash_{nd} \phi$ iff, for some t there exists a derivation $\Gamma \vdash (\cdot, t) : \phi$ where $\Gamma = \{(\cdot, 0) : \psi \mid \psi \in \Delta\}$.*

The proof is given in the appendix, section A.

Corollary 1 *For any finite set of labelled formulas Γ and a labelled formula ϕ, it is decidable whether $\Gamma \vdash \phi$.*

Proof. Since for any unlabelled set of formulas Δ and unlabelled formula ϕ it is decidable whether $\Gamma \vdash_{cl} \phi$ in propositional natural deduction, it follows from theorem 1 that it is decidable whether $\Gamma \vdash \phi$. $\dashv$

4.3 TRL(ND) Models

Definition 1 (Models) *Let A be a finite set of finite sequences of formulas over $\mathcal{L}$. A TRL(ND) model M_A over A is a tuple $\langle obs, \{inf_i \mid i \in A\}, \{m^i_t \mid i \in A, t \in \mathbb{N}\}\rangle$ where:*

1. *each m^i_t is a finite set of formulas in $\mathcal{L}$ such that $m^i_{t+1} = inf_i(m^i_t) \cup obs(i, t+1)$.*
2. *inf_i is a function from finite sets of formulas in $\mathcal{L}$ to finite sets of formulas in $\mathcal{L}$ and satisfies the following conditions:*

$$\phi_1 \wedge \phi_2 \in m^i_t \Longrightarrow \phi_1, \phi_2 \in inf_i(m^i_t)$$
$$\phi_1, \phi_2 \in m^i_t \Longrightarrow \phi_1 \wedge \phi_2 \in inf_i(m^i_t)$$
$$\phi, \neg\phi \in m^i_t \Longrightarrow \bot \in inf_i(m^i_t)$$
$$\neg\neg\phi \in m^i_t \Longrightarrow \phi \in inf_i(m^i_t)$$
$$\phi \in m^i_t \Longrightarrow \phi \in inf_i(m^i_t)$$

3. *obs is a function which maps a pair (i, t) to a finite set of formulas in $\mathcal{L}$ and satisfies the following conditions:*

$$\bot \in m^{i\phi}_t \Longrightarrow \neg\phi \in obs(i, t+1)$$

$$(i,t) : \phi \in \Gamma \Longrightarrow \phi \in obs(\langle \ldots i \ldots \rangle, t)$$
$$\phi \in obs(\langle \ldots \phi \ldots \rangle, t)$$

Definition 2 (Satisfaction and Entailment) *A labelled formula $(i,t) : \phi$ is true in a TRL model, written $M \models (i,t) : \phi$, iff $\phi \in m^i_t$. A formula $(i,t) : \phi$ is said to be A-valid, written $\models_A (i,t) : \phi$, whenever $M_A \models (i,t) : \phi$ for all models M_A over A. We say $(i,t) : \phi$ is an A-consequence of a set of labelled formulas Γ, written $\Gamma \models_A (i,t) : \phi$, if $(i,t) : \phi$ is true in all models M_A over A in which Γ is true.*

Again, there is a special case of truth in a model for TRL(ND), namely when $\Gamma \models_A (\cdot, t) : \phi$, which tells us what, according to our definition of truth, the ND reasoner should believe at step t.

Since different derivations may rely on making different sequences of assumptions, there is a special class of model, comprising only those that ensure all such assumptions are represented. However, we cannot simply assume that A contains all possible sequences over $\mathcal{L}$ in models in this class, for this would allow the possibility of infinitely many formulas being introduced to a local state by the obs condition corresponding to RAA. We thus need the notion of a *sufficient* model for a particular derivation:

Definition 3 (Sufficient Models) *A TRL(ND) model M_A is said to be sufficient for a pair $(\Gamma, (i,t) : \phi)$ if either $\Gamma \nvdash (i,t) : \phi$ or else there exists a derivation $\Gamma \vdash (i,t) : \phi$ containing a sequence of labelled formulas*

$$\begin{gathered}(i_1, t_1) : \phi_1 \\ \vdots \\ (i_n, t_n) : \phi_n\end{gathered}$$

we have $i_k \in A \in M_A$ for every $1 \leq k \leq n$.

Corollary 2 below establishes that the special case of consequence, $\Gamma \models_A (\cdot, t) : \phi$, holds for sufficient models M_A whenever ϕ is a consequence (w.r.t. the classical notion) of the formulas in Γ (minus their labels). In the following section, we establish that TRL(ND) is sound and complete with respect to sufficient models.

4.4 Completeness of TRL(ND)

The following definition and lemma apply to any system of TRL which uses the *all rules at each step* rule application strategy; it is thus easy to see how completeness and decidability proofs are constructed for systems of TRL other than TRL(ND).

Definition 4 (Minimal Model) *A TRL(ND) model M_A is a minimal model for a set of labelled formulas Γ if, for every i, t and every formula ϕ, $\phi \in m_t^i$ iff one of the following holds:*

1. *there is a rule in R_{ND}^{inf} whose antecedent formulas are matched by formulas $\phi_k \in m_{t-1}^i$ and the consequent is $(i,t) : \phi$, or*
2. *$\phi := \neg\psi$ and $\bot \in m_{t-1}^{i\psi}$, i.e.* RAA *matches, or*
3. *$i = \langle \ldots \psi_1 \ldots \psi_n \ldots \rangle$ and $(\langle \psi_1 \ldots \psi_n \rangle, t) : \phi \in \Gamma$, i.e.* INHERIT *matches, or*
4. *$i = \langle \ldots \phi \ldots \rangle$, i.e.* AX *matches.*

A minimal model for Γ thus only satisfies the formulas in Γ and their consequences. Note that, for each of these conditions, there corresponds a rule in R_{ND}. We use this fact to establish the following lemma:

Lemma 1 *Let M_A be a minimal TRL(ND) model for Γ which is also sufficient for $(\Gamma(i,t) : \phi)$. Then for every formula ϕ, $\phi \in m_t^i$ iff $\Gamma \vdash (i,t) : \phi$.*

Proof. The proof goes by induction on t. When $t = 0$, $(i,0) : \phi \in m_0^i$ iff either $(i,0) : \phi \in \Gamma$ or $i = \langle \ldots \phi \rangle$ (definition 4). In the former case, $\Gamma \vdash (i,0) : \phi$ by the definition of $\vdash$. In the latter case, $\Gamma \vdash (i,0) : \phi$ by AX.

Inductive hypothesis: suppose that for all agents j and all $s \leq t$, $\phi \in m_s^j$ iff $\Gamma \vdash (j,s) : \phi$. Now consider any $\phi \in m_{t+1}^i$; either $\phi \in inf_i(m_t^i)$ or $\phi \in obs(i, t+1)$. In the former case, there is a rule in R_{ND}^{inf} of the form

$$\frac{(i,s) : \phi_1, \ldots, (i,s) : \phi_n}{(i,s+1) : \psi}$$

such that $\psi = \phi$ and $\phi_1, \ldots, \phi_n \in m_t^i$. By the inductive hypothesis, $\Gamma \vdash (i,t) : \phi_i$. Hence, by the same rule, $\Gamma \vdash (i,t+1) : \phi$.

In the latter case, either condition 2, 3 or 4 of definition 4 holds. If 2 holds, then $\phi := \neg\psi$ and $\bot \in m_t^{i\psi}$ (definition 4) and, by hypothesis, $\Gamma \vdash (i\psi, t) : \bot$. Then $\Gamma \vdash (i, t+1) : \neg\psi$ by RAA. If 3 holds, then $i = k\, j\, k'$ for some (possibly empty) sequences j, k, k' and $(j, t+1) : \phi \in \Gamma$. Then by the definition of $\vdash$, $\Gamma \vdash (j, t+1) : \phi$ and, by INHERIT, $\Gamma \vdash (i, t+1) : \phi$. Finally, if 4 holds then $i = \langle \ldots \phi \ldots \rangle$ and, by AX, we have $\Gamma \vdash (i, t+1) : \phi$. $\dashv$

Theorem 2 *TRL(ND) is sound and complete w.r.t. sufficient models, i.e. for any finite set of labelled formulas Γ, a labelled formula ϕ and a set of agents A such that M_A is sufficient for (Γ, ϕ), $\Gamma \vdash \phi$ iff $\Gamma \models_A \phi$.*

Proof. Soundness ($\Gamma \vdash \phi \Rightarrow \Gamma \models_A \phi$) is standard: clearly, the rules in R_{ND} preserve validity in TRL(ND) models. For completeness, suppose $\Gamma \models_A \phi$. Consider a minimal TRL(ND) model for Γ, M_Γ. Since $\Gamma \models_A \phi$ and our particular model M_Γ satisfies Γ, $M_\Gamma \models_A \phi$. From Lemma 1, $\Gamma \vdash \phi$. $\dashv$

Corollary 2 *For all models M_A sufficient for $(\Gamma, (\cdot, t)$, for some t, $\Gamma \models_A (\cdot, t) : \phi \Longleftrightarrow \Gamma \models_{cl} \phi$, where $\models_{cl}$ is the classical notion of consequence.*

Proof. By theorem 1; soundness and completeness of TRL(ND) (theorem 2); and classical soundness and completeness for natural deduction. $\dashv$

Corollary 3 *For any finite set of labelled formulas Γ, labelled formula ϕ and finite set of finite sequences A, it is decidable whether $\Gamma \models_A \phi$.*

Proof. From Theorem 2 above, the questions whether $\Gamma \vdash (i, t) : \phi$ and whether $\Gamma \models_A (i, t) : \phi$ are equivalent and so, from corollary 1, it is decidable whether $\Gamma \models_A \phi$. $\dashv$

5 Conclusions

We have shown that resource bounded (i.e. real) agents *can* be modelled as having a complete set of rules without ignoring the time required to deliberate. Using the general framework of TRL, we have presented a classical reasoner which takes time to form new beliefs. We developed the logic TRL(ND) in which to model this reasoner and demonstrated it to be sound, complete and decidable. We also indicated how completeness and decidability can be shown for similar systems of of TRL. The availability of complete semantics distinguishes TRL from step logics, as presented in [2]; similarly, the decidability result distinguishes TRL from general first-order approaches such as [10].

It is these features of TRL, as highlighted by our TRL(ND) example, that have formed the primary focus of this paper. However, that is not to disregard the formalisation of assumptions developed here; as mentioned in the introduction, the approach may have other applications (for example, in distinguishing an agent's firmly-held beliefs from working hypotheses). Another distinguishing feature of TRL is its ability to model difference rule application strategies. The example here of a classical reasoner used an *all rules at each cycle* strategy; in [1], a CLIPS-style depth resolution strategy [24] is discussed. TRL thus allows one to model an agent and derive results pertaining to which of a number of possible strategies would be most effective, given the agent's goals and limitations.

A more realistic account of multiagent communication can be achieved in TRL by partitioning agent belief states into private and public sections. An agent may then perform deductive reasoning in private, and choose whether to allow other agents access to the results. Modification of what are here called *communication rules* allows representations in which only certain agents can access the beliefs of a given agent. Importantly for the current project, such representations can be built without modification of the underlying TRL framework, by adding an extra parameter (either `private` or `public`) to each agent name-sequence.

Semantically, an agent would then be treated as two distinct local states, one for private formulas and one for the publicly observable ones. Internal rules then pertain to private formulas; extra rules must be added to deal with the moving of formulas from private to public storage. A goal of further research is thus to present concrete representations of multiagent communication along these lines. A variation on this theme would be to model contingent communication between groups of agents, where agents communicate only to other agents within the same group (i.e. the "public" part of an agent's state is observable only to agents in the same group).

We can also model communication with a delay involved, such that a message is not guaranteed to arrive at the timepoint immediately after being sent. We could, for example, investigate how different rule application strategies behave when the delivery time of sent messages in a multiagent system is not constant, or differs depending on the sender or recipient of the message, e.g., where some agents in the system are close together and some are far apart.

Another goal of further research is to represent speech acts in TRL. Enriching TRL in this way would allow models of communication in which agents explicitly query, command and share beliefs amongst one another. One way in which this can be achieved is to augment the labelling language, which currently contains only agent and time parameters, with *force parameters*. The notion of force is intended to indicate how the propositional content of an utterance is used in a speech act; i.e. whether it is used to query, command or as declarative information. An agent's internal rules would then have to be suitably embellished to deal with different kinds of speech act; for example, a helpful agent should respond to a query $?\phi$ with either ϕ, $\neg\phi$ or "I don't know," depending on its beliefs at that timepoint. It appears that this would be a very fruitful extension of TRL from the point of view of modelling communication within multiagent systems.

References

[1] N. Alechina, B. Logan, and M. Whitsey. A complete and decidable logic for resource-bounded agents. In *Proc. Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*. ACM Press, July 2004.

[2] J. Drapkin and D. Perlis. A preliminary excursion into Step-Logics. *Proceedings of the SIGART International Symposium on Methodologies for Intelligent Systems*, pages 262–269, 1986.

[3] J. Elgot-Drapkin, S. Kraus, M. Miller, M. Nirkhe, and D. Perlis. Active logics: A unified formal approach to episodic reasoning. Technical Report CS-TR-4072, University of Maryland, Department of Computer Science, 1999.

[4] J. Elgot-Drapkin, M. Miller, and D. Perlis. Memory, reason and time: the Step-Logic approach. In R. Cummins and J. Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 79–103. MIT Press, Cambridge, Mass., 1991.

[5] J. Elgot-Drapkin and D. Perlis. Reasoning situated in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(1):75–98, 1990.

[6] R. Fagin and J.Y. Halpern. Belief, awareness and limited reasoning. In *Ninth International Joint Conference on AI*, pages 491–501, LA, California, 1985.

[7] R. Fagin and J.Y. Halpern. Belief, awareness and limited reasoning. *Artificial Intelligence*, 34:39–76, 1988.

[8] Dov M. Gabbay. *Labeled Deductive Systems: Volume I - Foundations*. Oxford UNiversity Press, 1996.

[9] C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning = locality + compatability. *Artificial Intelligence*, 127(2):221–259, 2001.

[10] John Grant, Sarit Kraus, and Donald Perlis. A logic for characterizing multiple bounded agents. *Autonomous Agents and Multi-Agent Systems*, 2000.

[11] J. Hintikka. Impossible possible worlds vindicated. *Journal of Philisophical Logic*, 4:475–484, 1675.

[12] J. Hintikka. *Knowledge and belief : an introduction to the logic of the two notions*. Cornell University Press, Ithaca, N.Y., 1962.

[13] K. Konolige. *A Deduction Model of Belief*. Morgan Kaufman, 1986.

[14] K. Konolige. What awareness isn't. In J.Y. Halpern, editor, *Theoetical Aspects of Reasoning About Knowledge*, pages 241–249. 1986.

[15] G. Lakemeyer. Steps towards a first-order logic of explicit and implict belief. In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the 1986 Conference*, pages 325–340, San Francisco, Calif., 1986. Morgan Kaufmann.

[16] H.J. Levesque. A logic of implicit and explicit belief. In *Proc. National COnference on Artificial Intelligence (AAAI '84)*, pages 198–202, 1984.

[17] R. C. Moore. *Logic and Representations*. Number 39 in CSLI Lecture Notes. CSLI Publications, 1995.

[18] M. Nirkhe, S. Kraus, and D. Perlis. Thinking takes time: a modal active-logic for reasoning in time. Technical Report CS-TR-3249, University of Maryland, Department of Computer Science, 1994.

[19] R. Parikh. Knowledge and the problem of logical omniscience. In *Methodologies for Intelligent Systems, Proceedings of the Second International Symposium*, pages 432–439. North-Holland, 1987.

[20] Y. Moses R. Fagin, J.Y. Halpern and M.Y Vardi. *Reasoning about Knowledge*. MIP Press, Cambridge, Mass., 1995.

[21] V. Rantala. Impossible worlds semantics and logical omniscience. *Acta Philosophica Fennica*, 35:18–24, 1982.

[22] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, 1991.

[23] M. P. Singh. Know-how. In Michael Wooldridge and Anand Rao, editors, *Foundations of Rational Agency*, pages 81–104. Kluwer Academic, Dordrecht, 1999.

[24] Software Technology Branch, Lyndon B. Johnson Space Center, Houston. *CLIPS Reference Manual: Version 6.21*, June 2003.

[25] W. van der Hoek, B. van Linder, and J-J. Ch. Meyer. An integrated modal approach to rational agents. In M. Wooldridge and A. Rao, editors, *Foundations of Rational Agency*, pages 133–168. Kluwer Academic, Dordrecht, 1999.

[26] M. Whitsey. Logical omniscience: a survey. Technical Report NOTTCS-WP-2003-2, School of Computer Science and IT, University of Nottingham, 2003.

A Proof of Theorem 1

Proof. Let $\bar{\psi} = \langle \psi_1, \ldots, \psi_k \rangle$ and $\Gamma = \{(0, \cdot) : \phi \mid \phi \in \Delta\}$. 'If' direction: we show $\Gamma \vdash (\bar{\psi}, t)\phi \Longrightarrow \Delta \cup \{\psi_1 \ldots \psi_k\} \vdash_{nd} \phi$ for any t, of which the 'if' direction is the special case where $\bar{\psi} = \cdot$. The proof is by induction on t. When $t = 0$, either $\phi \in \Gamma$ or else $\phi = \psi_{j \leq k}$. In the former case, we have $\phi \in \Delta$ and so $\Delta \vdash_{nd} \phi$. In the latter case, $\psi_j \vdash_{nd} \phi$ and so $\Delta \cup \{\psi_1 \ldots \psi_j \ldots \psi_k\} \vdash_{nd} \phi$. Induction hypothesis: $\Gamma \vdash (\bar{\psi}, s) : \phi \Longrightarrow \Delta, \psi_1, \ldots, \psi_k \vdash_{nd} \phi$ for all timepoints $s < t$. Now assume $\Gamma \vdash (\bar{\psi}, t) : \phi$. ϕ must have been obtained in TRL(ND) using one of the following rules:

- $\wedge_{int}$, i.e. $\phi := \phi_1 \wedge \phi_2$, so $\Gamma \vdash (\bar{\psi}, t-1) : \phi_n$ and by hypothesis $\Delta \cup \{\psi_1 \ldots \psi_k\} \vdash_{nd} \phi_n$. By the ND rule $\wedge_{int}$, we have $\Delta \cup \{\psi_1 \ldots \psi_k\} \vdash_{nd} \phi_1 \wedge \phi_2$.

- $\wedge_{elimL}$ or $\wedge_{elimR}$. Then $\Gamma \vdash_(\bar{\psi}, t-1) : \phi_1 \wedge \phi_2$ s.t. $\phi = \phi_n$, $n \in \{1, 2\}$. By hypothesis, $\Delta \cup \{\psi_1 \ldots \psi_k\} \vdash_{nd} \phi_1 \wedge \phi_2$ and so, by ND $\wedge_{elimL}$ or $\wedge_{elimR}$, $\Delta \cup \{\psi_1 \ldots \psi_k\} \vdash \phi_n$.

- $\neg_{elim}$. Then $\Gamma \vdash (\bar{\psi}, t-1) : \neg\neg\phi$ and by hypothesis, $\Delta \cup \{\psi_1 \ldots \psi_k\} \vdash_{nd} \neg\neg\phi$. By the ND rule $\neg_{elim}$, we obtain $\Delta \cup \{\psi_1 \ldots \psi_k\} \vdash_{nd} \phi$.

- RAA Then $\phi := \neg\chi$ and $\Gamma \vdash (\bar{\psi}\chi, t-1) : \bot$. By hypothesis, $\Delta \cup \{\psi_1 \ldots \psi_k, \chi\} \vdash_{nd} \bot$, i.e. $\Delta \cup \{\psi_1 \ldots \psi_k\} \vdash_{nd} \chi\bot$. Consider the derivation in which χ is an assumption; applying ND RAA, discharge the assumption and write $\neg\chi$. Hence $\Delta \cup \{\psi_1 \ldots \psi_k\} \vdash_{nd} \phi$.

- INHERIT Then, for some j, $(j, t) : \phi \in \Gamma$, hence $\phi \in \Delta$ and so $\Delta \cup \{\psi_1 \ldots \psi_k\} \vdash \phi$.

- AX: Then $\phi := \psi_j$, $1 \leq j \leq k$, and so $\Delta, \psi_1, \ldots, \psi_k, \phi \vdash_{nd} \phi$.

'Only if' direction: we show $\Delta \cup \{\psi_1, \ldots, \psi_n\} \vdash_{nd} \phi$ only if, for some t, $\Gamma \vdash (\langle \psi_1, \ldots, \psi_n \rangle, t) : \phi$. The proof is by induction on the length l of the derivation of ϕ. When $l = 0$, $\Delta \cup \{\psi_1 \ldots \psi_k\} \vdash_{nd} \phi$ either if $\phi \in \Delta$, in which case we have $(\bar{\psi}, 0) : \phi \in \Gamma$, or else $\phi = \psi_k$, $k \leq n$; either way, $\Gamma \models (\bar{\psi}, 0) : \phi$. Induction hypothesis: $\Gamma \vdash (\bar{\psi}, t) : \chi$, for some t, whenever $\Delta, \psi_1, \ldots, \psi_n \vdash_{nd} \chi$ in a derivation of length $k < l$. Suppose $\Delta \cup \{\psi_1, \ldots, \psi_n\} \vdash_{nd} \phi$. Either $\phi \in \Delta$, in which case $(\cdot, 0) : \phi \in \Gamma$ and hence $\Gamma \vdash (\cdot, t) : \phi$, or else ϕ is obtained using one of the natural deduction rules. For $\wedge_{int}$, $\wedge_{elimL}$, $\wedge_{elimR}$ and $\neg_{elim}$, the proof is trivial. For RAA, suppose $\phi := \neg\chi$. Then $\bot$ is derived from an assumption χ in the derivation, i.e. $\Delta, \psi_1, \ldots \psi_k, \chi \vdash_{nd} \bot$. By hypothesis, we have $\Gamma \vdash (\bar{\psi}\chi, t) : \bot$ for some t and so, by the TRL(ND) rule RAA, $\Gamma \vdash (\bar{\psi}, t) : \neg\chi$. $\dashv$