

# Parallel dynamic logic with communication

László Aszalós\*

Philippe Balbiani

## Abstract

This paper introduces a propositional dynamic logic in which sequences of message exchanges between two agents can be represented. It addresses the issues of axiomatization/completeness and decidability/complexity of this propositional dynamic logic.

## 1 Introduction

Encryption systems are an essential tool in computer security: they let agents transmit information in a hidden form over an insecure network [11, 13]. It is nevertheless true that encryption systems are not enough for the agents to assure that all transactions using the network will be secure. Using pre-established conventions such as cryptographic protocols, agents can achieve their security goals: data confidentiality, data integrity, availability of resources, etc. Cryptographic protocols give the description of how agents should exchange messages on the network. They are usually made up of encryption algorithms together with agreed-upon sequences of message exchanges. Their use in computers and communications enable agents to interact with each other and be convinced of fairness while they execute tasks such as negotiating contracts, voting, distributing information, etc. Because of the fast expansion of electronic commerce, cryptographic protocol verification is emerging as a very challenging issue.

Several formal methods have been proposed for verifying cryptographic protocols. Some of them are based on specialized logics such as the logic BAN for authentication protocols [3]. This logic formalizes the reasoning carried out by agents when they execute an authentication protocol and describes how the knowledge of agents evolves as messages are exchanged. Other approaches, see [9], are based on alternating-time temporal logic. Alternative ways of reasoning about cryptographic protocols is to liken their execution to sets of communication traces. The different ways in which this idea can be formalized are based on process calculi [1], rewriting systems [12], tree automata [10], etc. Notwithstanding the capacity of these approaches for analyzing cryptographic protocols and detecting their defects, the one drawback is that specialized logics (such as the logic BAN) are limited in the analysis of security concepts and alternative ways (based on process calculi, rewriting systems, tree automata, etc) lack an appropriate semantics for epistemic concepts.

What we have in mind is to tackle cryptographic protocol verification by defining a formal language based on epistemic logic and dynamic logic where

---

\*Corresponding author: László Aszalós, Institut de recherche en informatique de Toulouse, 118 route de Narbonne, F-31062 Toulouse Cedex 4; email address: aszalos@irit.fr.

both the knowledge of agents and the execution of cryptographic protocols can be represented. In this paper, we focus our attention on the dynamic part of our formal language. When two agents exchange messages on the network by executing some cryptographic protocol, each one of them executes its part of the protocol. This brings us to define our formal language as a many-dimensional dynamic logic. We start off with the idea of associating with each pair of programs  $\alpha, \beta$  a modality  $[\alpha \parallel \beta]$ , the formula  $[\alpha \parallel \beta]A$  being read “whenever the two agents terminate to execute in parallel  $\alpha$  and  $\beta$ , they must do so in a state satisfying  $A$ ”. The paper is organized as follows. We introduce in section 2 the basic concepts relating to the syntax and the semantics of our logic. In section 3, we study the commutativity and Church-Rosser properties of our modalities. We devote section 4 to the mechanization of our logic through a tableau method. We deal in section 5 with the axiomatization/completeness issue of a fragment of our modal logic and we briefly sketch the proof that the satisfiability problem for any given formula of this fragment is decidable.

## 2 Syntax and semantics

We consider a countably infinite set  $\{p_1, p_2, \dots\}$  of propositional variables and a countably infinite set  $\{\pi_1, \pi_2, \dots\}$  of atomic programs. The set of all formulae and the set of all programs are defined by induction as follows:

$$A := p_k \mid \neg A \mid (A \vee B) \mid [\alpha \parallel \beta]A$$

$$\alpha := \pi_k \mid \lambda \mid A? \mid (\alpha; \beta) \mid (\alpha \cup \beta) \mid \mathbf{send}(m) \mid \mathbf{rec}(m)$$

Like standard propositional dynamic logic [7, 8], a typical feature of our language is that it is a multi-modal language with an algebraic structure in the set of modalities, the formula  $[\alpha \parallel \beta]A$  being read “whenever the two agents terminate to execute in parallel  $\alpha$  and  $\beta$ , they must do so in a state satisfying  $A$ ”. Intended meanings of program constructs are: constant  $\lambda$  “continue”, test  $A?$  “continue if  $A$  is true, otherwise fail”, composition  $\alpha; \beta$  “do  $\alpha$  and then  $\beta$ ”, union  $\alpha \cup \beta$  “do either  $\alpha$  or  $\beta$  nondeterministically”,  $\mathbf{send}(m)$  “send message  $m$ ”,  $\mathbf{rec}(m)$  “receive message  $m$ ”, where  $m$  ranges over a countably infinite set of message constants. The other standard connectives are defined by the usual abbreviations. In particular  $\langle \alpha \parallel \beta \rangle A$  is  $\neg[\alpha \parallel \beta]\neg A$ . We follow the standard rules for omission of the parentheses. We may denote formulae by upper case Latin letters like  $A, B, C$ . Lower case Greek letters like  $\alpha, \beta, \gamma$  are reserved for programs. We assume that the function  $;$  is associative with unit element  $\lambda$ , i.e. we identify two programs if they are equivalent with respect to the least congruence  $\equiv$  on the set of all programs such that  $(\alpha; \beta); \gamma \equiv \alpha; (\beta; \gamma)$ ,  $\lambda; \alpha \equiv \alpha$  and  $\alpha; \lambda \equiv \alpha$ . It must be remarked that the question whether two given programs are equivalent modulo  $\equiv$  can be decided in polynomial time [2]. We shall say that  $\alpha$  is an elementary program if it is equivalent to a program of one of the following forms:  $\pi_k, \lambda, A?, \mathbf{send}(m), \mathbf{rec}(m)$ .

A model is a relational structure of the form  $\mathcal{M} = (W_1, W_2, R_1, R_2, V)$  where  $W_1$  and  $W_2$  are nonempty sets of local states,  $R_1$  and  $R_2$  are families of binary relations on  $W_1$  and  $W_2$ , i.e.  $R_1(\pi) \subseteq W_1 \times W_1$  and  $R_2(\pi) \subseteq W_2 \times W_2$  for all atomic programs  $\pi$ , and  $V$  is a valuation on  $W_1 \times W_2$ , i.e.  $V(p_j) \subseteq W_1 \times W_2$  for all propositional variables  $p_j$ . When the two agents exchange messages they

execute elementary programs of the form  $\mathbf{send}(m)$ ,  $\mathbf{rec}(m)$ . We assume that agents transmit messages over a perfect channel. Nevertheless, seeing that there may be a delay between emission and reception of messages, we need to evaluate formulae at quadruples of the form  $(s_1, c_1, s_2, c_2)$ , where  $s_1 \in W_1$  is the current local state of the first agent,  $c_1$  is the list of all the messages sent by the first agent but laid unread by the second agent,  $s_2 \in W_2$  is the current local state of the first agent and  $c_2$  is the list of all the messages sent by the second agent but laid unread by the first agent. Such quadruples will be called global states. If  $c$  is a list of messages and  $m$  is a message, then by  $c \star m$  we mean the new list whose last element is  $m$  and whose remaining elements are those of  $c$ , whereas by  $m \star c$  we mean the new list whose first element is  $m$  and whose remaining elements are those of  $c$ . In the sequel,  $\varepsilon$  will denote the empty list. Given a model  $\mathcal{M}$  we define the accessibility relation  $(s_1, c_1, s_2, c_2) R_{\alpha\|\beta}(t_1, d_1, t_2, d_2)$  between global states and the truth-relation  $(s_1, c_1, s_2, c_2) \models_{\mathcal{M}} A$  between global states and formulae as follows:

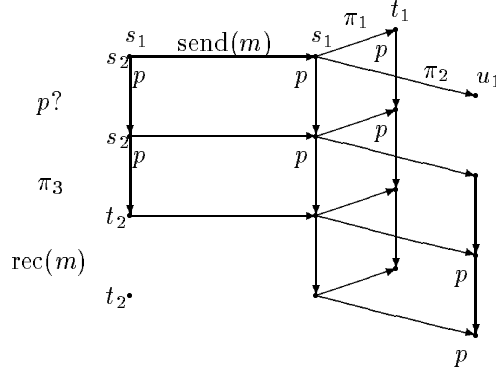
- $(s_1, c_1, s_2, c_2) R_{\lambda\|\lambda}(t_1, d_1, t_2, d_2)$  iff  $s_1 = t_1, c_1 = d_1, s_2 = t_2, c_2 = d_2$ .
- $(s_1, c_1, s_2, c_2) R_{\pi\|\lambda}(t_1, d_1, t_2, d_2)$  iff  $s_1 R_1(\pi)t_1, c_1 = d_1, s_2 = t_2, c_2 = d_2$ .
- $(s_1, c_1, s_2, c_2) R_{\lambda\|\pi}(t_1, d_1, t_2, d_2)$  iff  $s_1 = t_1, c_1 = d_1, s_2 R_2(\pi)t_2, c_2 = d_2$ .
- $(s_1, c_1, s_2, c_2) R_{A?\|\lambda}(t_1, d_1, t_2, d_2)$  iff  $s_1 = t_1, c_1 = d_1, s_2 = t_2, c_2 = d_2, (s_1, c_1, s_2, c_2) \models_{\mathcal{M}} A$ .
- $(s_1, c_1, s_2, c_2) R_{\lambda\|A?}(t_1, d_1, t_2, d_2)$  iff  $s_1 = t_1, c_1 = d_1, s_2 = t_2, c_2 = d_2, (s_1, c_1, s_2, c_2) \not\models_{\mathcal{M}} A$ .
- $(s_1, c_1, s_2, c_2) R_{\mathbf{send}(m)\|\lambda}(t_1, d_1, t_2, d_2)$  iff  $s_1 = t_1, d_1 = c_1 \star m, s_2 = t_2, c_2 = d_2$ .
- $(s_1, c_1, s_2, c_2) R_{\lambda\|\mathbf{send}(m)}(t_1, d_1, t_2, d_2)$  iff  $s_1 = t_1, c_1 = d_1, s_2 = t_2, d_2 = c_2 \star m$ .
- $(s_1, c_1, s_2, c_2) R_{\mathbf{rec}(m)\|\lambda}(t_1, d_1, t_2, d_2)$  iff  $s_1 = t_1, c_1 = d_1, s_2 = t_2, c_2 = m \star d_2$ .
- $(s_1, c_1, s_2, c_2) R_{\lambda\|\mathbf{rec}(m)}(t_1, d_1, t_2, d_2)$  iff  $s_1 = t_1, c_1 = m \star d_1, s_2 = t_2, c_2 = d_2$ .
- $R_{\lambda\|\varphi;\beta} = (R_{\lambda\|\varphi} \circ R_{\lambda\|\beta})$ .
- $R_{\varphi;\alpha\|\lambda} = (R_{\varphi\|\lambda} \circ R_{\alpha\|\lambda})$ .
- $R_{\varphi;\alpha\|\psi;\beta} = (R_{\varphi\|\lambda} \circ R_{\alpha\|\psi;\beta}) \cup (R_{\lambda\|\psi} \circ R_{\varphi;\alpha\|\beta})$  where  $\varphi$  and  $\psi$  are elementary programs.
- $R_{\alpha(\alpha_1 \cup \alpha_2)\|\beta} = R_{\alpha(\alpha_1)\|\beta} \cup R_{\alpha(\alpha_2)\|\beta}$ .
- $R_{\alpha\|\beta(\beta_1 \cup \beta_2)} = R_{\alpha\|\beta(\beta_1)} \cup R_{\alpha\|\beta(\beta_2)}$ .
- $(s_1, c_1, s_2, c_2) \models_{\mathcal{M}} p$  iff  $(s_1, s_2) \in V(p)$ .
- $(s_1, c_1, s_2, c_2) \models_{\mathcal{M}} \neg A$  iff  $(s_1, c_1, s_2, c_2) \not\models_{\mathcal{M}} A$ .
- $(s_1, c_1, s_2, c_2) \models_{\mathcal{M}} A \vee B$  iff  $(s_1, c_1, s_2, c_2) \models_{\mathcal{M}} A$  or  $(s_1, c_1, s_2, c_2) \models_{\mathcal{M}} B$ .

	$\pi_j$	$B?$	$\mathbf{send}(m)$	$\mathbf{rec}(m)$
$\pi_i$	•		•	•
$A?$		•		
$\mathbf{send}(m)$	•		•	
$\mathbf{rec}(m)$	•			•

Table 1: Commutativity properties.

- $(s_1, c_1, s_2, c_2) \models_{\mathcal{M}} [\alpha \parallel \beta] A$  iff for all  $(t_1, d_1, t_2, d_2)$ , if  $(s_1, c_1, s_2, c_2) R_{\alpha \parallel \beta} (t_1, d_1, t_2, d_2)$  then  $(t_1, d_1, t_2, d_2) \models_{\mathcal{M}} A$ .

In the definitions above,  $\alpha(\beta)$  always denotes a program with occurrence of program  $\beta$  as a subprogram. As a result,  $\alpha(\gamma)$  denotes the result of the replacement of  $\beta$  in its place in  $\alpha$  with another program  $\gamma$ . Note that, in consequence of the definition above we have  $R_{\alpha \parallel \beta \parallel \lambda} = R_{\alpha \parallel \lambda} \circ R_{\beta \parallel \lambda}$ . We say that formula  $A$  is *satisfiable* in model  $\mathcal{M}$  if  $(s_1, \varepsilon, s_2, \varepsilon) \models_{\mathcal{M}} A$  for some  $s_1 \in W_1$  and for some  $s_2 \in W_2$ , whereas we say that formula  $A$  is *valid* in model  $\mathcal{M}$  if  $(s_1, \varepsilon, s_2, \varepsilon) \models_{\mathcal{M}} A$  for all  $s_1 \in W_1$  and for all  $s_2 \in W_2$ . A formula is satisfiable if it is satisfiable in some model whereas it is valid if it is valid in every model. To make everything clear, we give an example. In this example, let  $W_1 = \{s_1, t_1, u_1\}$ ,  $W_2 = \{s_2, t_2\}$ ,  $s_1 R_1(\pi_1)t_1$ ,  $s_1 R_1(\pi_2)u_1$ , and  $s_2 R_2(\pi_3)t_2$ . The valuation of propositional variable  $p$  is given by  $V(p) = \{(s_1, s_2), (t_1, s_2), (u_1, t_2)\}$ . We can see that in this case at  $(s_1, \varepsilon, s_2, \varepsilon)$  the formula  $\langle \mathbf{send}(m); (\pi \cup \pi_2) \parallel p?; \pi_3; \mathbf{rec}(m) \rangle p$  is true, but the formula  $[\mathbf{send}(m); (\pi_1 \cup \pi_2) \parallel p?; \pi_3; \mathbf{rec}(m)] p$  is false, because  $(s_1, \varepsilon, s_2, \varepsilon) R_{\mathbf{send}(m); (\pi_1 \cup \pi_2) \parallel p?; \pi_3; \mathbf{rec}(m)} (t_1, \varepsilon, t_2, \varepsilon)$  and  $p$  is false at  $(t_1, \varepsilon, t_2, \varepsilon)$ .



### 3 Properties

To understand the meaning of our modalities, a good start is to find formulae that are valid in every model. The most obvious formulae are of the form:

$$[\varphi \parallel \lambda][\lambda \parallel \psi] A \leftrightarrow [\lambda \parallel \psi][\varphi \parallel \lambda] A$$

$$\langle \varphi \parallel \lambda \rangle [\lambda \parallel \psi] A \rightarrow [\lambda \parallel \psi] \langle \varphi \parallel \lambda \rangle A$$

	$\pi_j$	$B?$	$\mathbf{send}(m)$	$\mathbf{rec}(m)$
$\pi_i$	•		•	•
$A?$		•		
$\mathbf{send}(m)$	•		•	•
$\mathbf{rec}(m)$	•		•	•

Table 2: Confluence properties.

where  $\varphi$  and  $\psi$  are elementary programs. Table 1 and table 2 give the different cases when such formulae are valid in every model. Table 1 says, for example, that formulae like  $[\pi_1 \parallel \lambda][\lambda \parallel \pi_2]A \leftrightarrow [\lambda \parallel \pi_2][\pi_1 \parallel \lambda]A$  are always valid whereas table 2 says, for example, that formulae like  $\langle \pi_1 \parallel \lambda \rangle [\lambda \parallel \pi_2]A \rightarrow [\lambda \parallel \pi_2] \langle \pi_1 \parallel \lambda \rangle A$  are always valid. The proof that every • in table 1 and table 2 corresponds to a formula valid in every model is left as an exercise for the reader. Let us examine why formulae like

$$\langle \mathbf{send}(m) \parallel \lambda \rangle [\lambda \parallel B?]A \rightarrow [\lambda \parallel B?] \langle \mathbf{send}(m) \parallel \lambda \rangle A$$

are not valid in every model. Let us regard the special case when  $A$  is constant  $\perp$  “false” and  $B$  is formula  $[\lambda \parallel \mathbf{rec}(m)]\perp$ . Obviously, in any model  $\mathcal{M}$

$$(s_1, \varepsilon, s_2, \varepsilon) \models_{\mathcal{M}} \langle \mathbf{send}(m) \parallel \lambda \rangle [\lambda \parallel [\lambda \parallel \mathbf{rec}(m)]\perp?] \perp$$

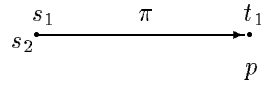
for all  $s_1 \in W_1$ , and for all  $s_2 \in W_2$ . However

$$(s_1, \varepsilon, s_2, \varepsilon) \not\models_{\mathcal{M}} [\lambda \parallel [\lambda \parallel \mathbf{rec}(m)]\perp?] \langle \mathbf{send}(m) \parallel \lambda \rangle \perp$$

for all  $s_1 \in W_1$ , and for all  $s_2 \in W_2$ . Now, let us examine why formulae like

$$[\pi \parallel \lambda][\lambda \parallel B?]A \leftrightarrow [\lambda \parallel B?][\pi \parallel \lambda]A$$

are not valid in every model. Let us regard the special case when formula  $A$  is constant  $\perp$  “false” and  $B$  is propositional variable  $p$ . Take the model  $\mathcal{M}$  where  $W_1 = \{s_1, t_1\}$ ,  $W_2 = \{s_2\}$ ,  $s_1 R_1(\pi)t_1$ , and  $V(p) = \{(t_1, s_2)\}$ .



The reader can easily prove that

$$(s_1, \varepsilon, s_2, \varepsilon) \not\models_{\mathcal{M}} [\pi \parallel \lambda][\lambda \parallel p?] \perp \text{ and } (s_1, \varepsilon, s_2, \varepsilon) \models_{\mathcal{M}} [\lambda \parallel p?][\pi \parallel \lambda] \perp.$$

## 4 Tableau method

What we have in mind is to prove that satisfiability in our logic is decidable. In this respect, we consider in this section a tableau method for the full language. We will show that our method is complete. Unfortunately, it does not terminate. As a result, we consider in section 5 a decidable fragment of our language. A prefix is a quadruple  $(\xi, c_1, \eta, c_2)$ , where  $\xi$  is a finite sequence of indexed atomic

programs,  $c_1$  is a finite sequence of messages,  $\eta$  is a finite sequence of indexed atomic programs and  $c_2$  is a finite sequence of messages.  $\xi$  is called the left-prefix of  $(\xi, c_1, \eta, c_2)$ , whereas  $\eta$  is called its right-prefix. The left-prefix  $\xi$  and the right-prefix  $\eta$  of prefix  $(\xi, c_1, \eta, c_2)$  encode the accessibility relations  $R_1$  and  $R_2$  in a way that will become clear later. A *prefixed formula* is a pair  $(\xi, c_1, \eta, c_2)A$  where  $(\xi, c_1, \eta, c_2)$  is a prefix and  $A$  is a formula. Proofs are written in tree form, branching downward. Every node of the tree is labelled by a prefixed formula. An attempted proof of  $A$  begins with the tree containing exactly one node labelled by  $(\lambda, \varepsilon, \lambda, \varepsilon) \neg A$ . Then the tree is enlarged step-by-step using certain extension rules. Apart from standard rules devoted to Boolean connectives, the set of branch extension rules contains the following rules:

$$\begin{array}{l} \langle \lambda \lambda \rangle: \frac{(\xi, c_1, \eta, c_2) \langle \lambda \parallel \lambda \rangle A}{(\xi, c_1, \eta, c_2) A} \quad [\lambda \lambda]: \frac{(\xi, c_1, \eta, c_2) [\lambda \parallel \lambda] A}{(\xi, c_1, \eta, c_2) A} \\ \langle ? \lambda \rangle: \frac{(\xi, c_1, \eta, c_2) \langle A? \parallel \lambda \rangle B}{(\xi, c_1, \eta, c_2) A \quad (\xi, c_1, \eta, c_2) B} \quad [? \lambda]: \frac{(\xi, c_1, \eta, c_2) [A? \parallel \lambda] B}{(\xi, c_1, \eta, c_2) B \mid (\xi, c_1, \eta, c_2) \neg A} \\ \langle \pi \lambda \rangle: \frac{(\xi, c_1, \eta, c_2) \langle \pi \parallel \lambda \rangle A}{(\xi; \pi^n, c_1, \eta, c_2) A} \quad [\pi \lambda]: \frac{(\xi, c_1, \eta, c_2) [\pi \parallel \lambda] A}{(\xi; \pi^n, c_1, \eta, c_2) A} \end{array}$$

At the rule  $\langle \pi \lambda \rangle$ ,  $n$  is an integer such that  $\xi; \pi^n$  does not already occur in the branch. At the rule  $[\pi \lambda]$ ,  $n$  is an integer such that  $\xi; \pi^n$  already occurs in the branch. Now we introduce rules concerning the elementary programs of the form  $\mathbf{send}(m)$ ,  $\mathbf{rec}(m)$ .

$$\begin{array}{l} \langle \mathbf{s} \lambda \rangle: \frac{(\xi, c_1, \eta, c_2) \langle \mathbf{send}(m) \parallel \lambda \rangle A}{(\xi, c_1 \star m, \eta, c_2) A} \quad [\mathbf{s} \lambda]: \frac{(\xi, c_1, \eta, c_2) [\mathbf{send}(m) \parallel \lambda] A}{(\xi, c_1 \star m, \eta, c_2) A} \\ \langle \mathbf{r} \lambda \rangle: \frac{(\xi, c_1, \eta, m \star c_2) \langle \mathbf{rec}(m) \parallel \lambda \rangle A}{(\xi, c_1, \eta, c_2) A} \quad [\mathbf{r} \lambda]: \frac{(\xi, c_1, \eta, m \star c_2) [\mathbf{rec}(m) \parallel \lambda] A}{(\xi, c_1, \eta, c_2) A} \\ \langle ; \rangle: \frac{(\xi, c_1, \eta, c_2) \langle \varphi; \alpha \parallel \psi; \beta \rangle A}{(\xi, c_1, \eta, c_2) \langle \varphi \parallel \lambda \rangle \langle \alpha \parallel \psi; \beta \rangle A \mid (\xi, c_1, \eta, c_2) \langle \lambda \parallel \psi \rangle \langle \varphi; \alpha \parallel \beta \rangle A} \\ [;]: \frac{(\xi, c_1, \eta, c_2) [\varphi; \alpha \parallel \psi; \beta] A}{(\xi, c_1, \eta, c_2) [\varphi \parallel \lambda] [\alpha \parallel \psi; \beta] A \quad (\xi, c_1, \eta, c_2) [\lambda \parallel \psi] [\varphi; \alpha \parallel \beta] A} \\ \langle \cup_1 \rangle: \frac{(\xi, c_1, \eta, c_2) \langle \alpha(\alpha_1 \cup \alpha_2) \parallel \beta \rangle A}{(\xi, c_1, \eta, c_2) \langle \alpha(\alpha_1) \parallel \beta \rangle A \mid (\xi, c_1, \eta, c_2) \langle \alpha(\alpha_2) \parallel \beta \rangle A} \quad [\cup_1]: \frac{(\xi, c_1, \eta, c_2) [\alpha(\alpha_1 \cup \alpha_2) \parallel \beta] A}{(\xi, c_1, \eta, c_2) [\alpha(\alpha_1) \parallel \beta] A \quad (\xi, c_1, \eta, c_2) [\alpha(\alpha_2) \parallel \beta] A} \end{array}$$

At the rules  $\langle ; \rangle$  and  $[;]$ ,  $\varphi$  and  $\psi$  denote elementary programs. Of course we have corresponding rules according to the second agent too. A *branch* is *closed* if one of the following conditions is satisfied:

- It contains  $(\xi, c_1, \eta, c_2) A$  and  $(\xi, c_1, \eta, c_2) \neg A$ .
- It contains  $(\xi, c_1, \eta, c_2) \langle \mathbf{rec}(m) \parallel \lambda \rangle$  and  $c_1$  does not begin with  $m$ .

- It contains  $(\xi, c_1, \eta, c_2)\langle \lambda \parallel \mathbf{rec}(m) \rangle$  and  $c_2$  does not begin with  $m$ .

A tree is *closed* if all its branches are closed. We say that a branch  $\mathcal{B}$  is *satisfiable* in a model  $\mathcal{M} = (W_1, W_2, R_1, R_2, V)$  if there are two functions  $f_1$  and  $f_2$  such that:

- $f_1$  maps every left-prefix in  $\mathcal{B}$  to a local state in  $W_1$ .
- $f_2$  maps every right-prefix in  $\mathcal{B}$  to a local state in  $W_2$ .
- If  $(\xi, c_1, \eta, c_2)A \in \mathcal{B}$  then  $(f_1(\xi), c_1, f_2(\eta), c_2) \models_{\mathcal{M}} A$ .
- If  $\xi$  and  $\xi; \pi^n$  are left-prefixes in  $\mathcal{B}$  then  $f_1(\xi)R_1(\pi)f_1(\xi; \pi^n)$ .
- If  $\eta$  and  $\eta; \pi^n$  are right-prefixes in  $\mathcal{B}$  then  $f_2(\eta)R_2(\pi)f_2(\eta; \pi^n)$ .

A branch  $\mathcal{B}$  is *satisfiable* if there is a model  $\mathcal{M}$  such that  $\mathcal{B}$  is satisfiable in  $\mathcal{M}$ . A tree  $\mathcal{T}$  is *satisfiable* if one of its branches is satisfiable. A proof of formula  $A$  is a closed tree built with the above rules beginning with the tree containing exactly one node labelled by  $(\lambda, \varepsilon, \lambda, \varepsilon)\neg A$ . To illustrate the difficulties of the termination the reader may examine the case where we have a tree with the nodes labelled with  $(\lambda, \varepsilon, \lambda, \varepsilon)\langle \varphi \parallel \lambda \rangle A$ ,  $(\lambda, \varepsilon, \lambda, \varepsilon)[\varphi \parallel \lambda]\langle \lambda \parallel \psi \rangle B$ , and  $(\lambda, \varepsilon, \lambda, \varepsilon)[\lambda \parallel \psi]\langle \varphi \parallel \lambda \rangle C$ . To prove the correctness and the completeness of our method we follow the line of reasoning suggested by Fitting [4]. By an easy verification, the reader may easily prove the following lemma.

**Lemma 1** *Suppose that  $\mathcal{T}$  is a satisfiable tree. Let  $\mathcal{T}'$  be a tree resulting from a single extension rule applied to  $\mathcal{T}$ . Then  $\mathcal{T}'$  is also satisfiable.*

Let  $A$  be a satisfiable formula. Hence the initial tree containing exactly one node labelled with the prefixed formula  $(\lambda, \varepsilon, \lambda, \varepsilon)A$  is satisfiable. Therefore using lemma 1, we can conclude that every tree built step-by-step from this initial tree will be satisfiable. Consequently there is no proof of formula  $\neg A$ . Then we have the following theorem.

**Theorem 2 (Soundness)** *If formula  $A$  has a proof built with the above rules then  $A$  is valid.*

Let  $\mathcal{C}$  be a collection of sets of prefixed formulae.  $\mathcal{C}$  is a *consistency property* if it meets the following conditions for each  $S \in \mathcal{C}$ :

1. There is no atomic formulae  $p$  such that  $S$  contains both  $(\xi, c_1, \eta, c_2)p$  and  $(\xi, c_1, \eta', c_2)\neg p$ .
2. If  $(\xi, c_1, \eta, c_2)(A \wedge B) \in S$  then  $S \cup \{(\xi, c_1, \eta, c_2)A, (\xi, c_1, \eta, c_2)B\} \in \mathcal{C}$ .
3. If  $(\xi, c_1, \eta, c_2)(A \vee B) \in S$  then  $S \cup \{(\xi, c_1, \eta, c_2)A\} \in \mathcal{C}$  or  $S \cup \{(\xi, c_1, \eta, c_2)B\} \in \mathcal{C}$ .
4. If  $(\xi, c_1, \eta, c_2)\langle \lambda \parallel \lambda \rangle A \in S$  or  $(\xi, c_1, \eta, c_2)[\lambda \parallel \lambda]A \in S$  then  $S \cup \{(\xi, c_1, \eta, c_2)A\} \in \mathcal{C}$ .
5. If  $(\xi, c_1, \eta, c_2)\langle A? \parallel \lambda \rangle B \in S$  then  $S \cup \{(\xi, c_1, \eta, c_2)A, (\xi, c_1, \eta, c_2)B\} \in \mathcal{C}$ .
6. If  $(\xi, c_1, \eta, c_2)[A? \parallel \lambda]B \in S$  then  $S \cup \{(\xi, c_1, \eta, c_2)B\} \in \mathcal{C}$  or  $S \cup \{(\xi, c_1, \eta, c_2)\neg A\} \in \mathcal{C}$ .
7. If  $(\xi, c_1, \eta, c_2)\langle \pi \parallel \lambda \rangle A \in S$  then  $S \cup \{(\xi; \pi^n, c_1, \eta, c_2)A\} \in \mathcal{C}$  for some positive integer  $n$ .

8. If  $(\xi, c_1, \eta, c_2)[\pi \parallel \lambda]A \in S$  then  $S \cup \{(\xi; \pi^n, c_1, \eta, c_2)A\} \in \mathcal{C}$  for all positive integers  $n$  such that  $\xi; \pi^n$  occurs as a left-prefix in  $S$ .
9. If  $(\xi, c_1, \eta, c_2)\langle \mathbf{send}(m) \parallel \lambda \rangle A \in S$  or  $(\xi, c_1, \eta, c_2)[\mathbf{send}(m) \parallel \lambda]A \in S$  then  $S \cup \{(\xi, c_1, \eta, c_2 \star m)A\} \in \mathcal{C}$ .
10. If  $(\xi, m \star c_1, \eta, c_2)\langle \mathbf{rec}(m) \parallel \lambda \rangle A \in S$  or  $(\xi, m \star c_1, \eta, c_2)[\mathbf{rec}(m) \parallel \lambda]A \in S$  then  $S \cup \{(\xi, c_1, \eta, c_2)A\} \in \mathcal{C}$ .
11. If  $(\xi, c_1, \eta, c_2)\langle \alpha(\alpha_1 \cup \alpha_2) \parallel \beta \rangle A \in S$  then  $S \cup \{(\xi, c_1, \eta, c_2)\langle \alpha(\alpha_1) \parallel \beta \rangle A\} \in \mathcal{C}$  or  $S \cup \{(\xi, c_1, \eta, c_2)\langle \alpha(\alpha_2) \parallel \beta \rangle A\} \in \mathcal{C}$ .
12. If  $(\xi, c_1, \eta, c_2)[\alpha(\alpha_1 \cup \alpha_2) \parallel \beta]A \in S$  then  $S \cup \{(\xi, c_1, \eta, c_2)[\alpha(\alpha_1) \parallel \beta]A, (\xi, c_1, \eta, c_2)[\alpha(\alpha_2) \parallel \beta]A\} \in \mathcal{C}$ .
13. If  $(\xi, c_1, \eta, c_2)\langle \varphi; \alpha \parallel \psi; \beta \rangle A \in S$  then  $S \cup \{(\xi, c_1, \eta, c_2)\langle \varphi \parallel \lambda \rangle \langle \alpha \parallel \psi; \beta \rangle A\} \in \mathcal{C}$  or  $S \cup \{(\xi, c_1, \eta, c_2)\langle \lambda \parallel \psi \rangle \langle \varphi; \alpha \parallel \beta \rangle A\} \in \mathcal{C}$ .
14. If  $(\xi, c_1, \eta, c_2)[\varphi; \alpha \parallel \psi; \beta]A \in S$  then  $S \cup \{(\xi, c_1, \eta, c_2)[\varphi \parallel \lambda][\alpha \parallel \psi; \beta]A, (\xi, c_1, \eta, c_2)[\lambda \parallel \psi][\varphi; \alpha \parallel \beta]A\} \in \mathcal{C}$ .

At the conditions 13 and 14,  $\varphi$  and  $\psi$  denote elementary programs. Of course we require that similar conditions for the second agent corresponding to conditions 5–12 hold too. If  $S \in \mathcal{C}$  then we say that  $S$  is  $\mathcal{C}$ -consistent. The reader may easily verify the following lemmas.

**Lemma 3** *Suppose  $\mathcal{C}$  is a consistency property. Let  $\mathcal{C}'$  be the set of all subsets of all members of  $\mathcal{C}$ . Then  $\mathcal{C}'$  is also a consistency property extending  $\mathcal{C}$  and closed under subsets.*

**Lemma 4** *Suppose  $\mathcal{C}'$  is a consistency property closed under subsets. Let  $\mathcal{C}''$  be the set of all sets of prefixed formulae whose finite subsets are in  $\mathcal{C}'$ . Then  $\mathcal{C}''$  is also a consistency property extending  $\mathcal{C}'$  and such that for all sets  $S$  of prefixed formulae,  $S \in \mathcal{C}''$  iff each finite subset of  $S$  belongs to  $\mathcal{C}''$ .*

A collection  $\mathcal{C}$  of sets of prefixed formula is said to be of *finite character* if for all sets  $S$  of prefixed formulae,  $S$  belongs to  $\mathcal{C}$  iff each finite subset of  $S$  belongs to  $\mathcal{C}$ . The following lemma immediately follows from lemma 3 and 4.

**Lemma 5** *Any consistency property can be extended to a consistency property of finite character.*

Now we can prove the following important theorem.

**Theorem 6 (Model existence)** *Let  $\mathcal{C}$  is a consistency property. If  $S$  is  $\mathcal{C}$ -consistent, then  $S$  is satisfiable.*

PROOF. Let  $\mathcal{C}$  be a consistency property. Suppose that  $S$  is  $\mathcal{C}$ -consistent. We have to show that  $S$  is satisfiable. According to lemma 5,  $\mathcal{C}$  can be extended to a consistency property  $\mathcal{C}'$  of finite character. We use  $\mathcal{C}'$  to create a model. Let  $S'$  be a maximal members of  $\mathcal{C}'$  which is an extension of  $S$ . Let  $W_1$  be  $\{\xi \mid (\xi, c_1, \eta, c_2)A \in S'\}$  and let  $W_2$  be  $\{\eta \mid (\xi, c_1, \eta, c_2)A \in S'\}$ . Moreover we define  $R_i$  as  $uR_i(\pi)v$  iff  $v = u; \pi^n$  for all  $u, v \in W_i$ . We define an interpretation  $V$  as follows: for each  $w_1 \in W_1$ , for each  $w_2 \in W_2$  and for each propositional variable

$p$ , let  $(w_1, w_2) \in V(p_i)$  iff  $(w_1, c_1, w_2, c_2)p \in S'$  for some lists of messages  $c_1$  and  $c_2$ . Now we defined model  $\mathcal{M} = (W_1, W_2, R_1, R_2, V)$ . We claim: for all prefixed formulae  $(w_1, c_1, w_2, c_2)A$ , if  $(w_1, c_1, w_2, c_2)A$  is in  $S'$ , then  $(w_1, c_1, w_2, c_2) \models_{\mathcal{M}} A$ . We can prove this by induction on the size of  $A$ , but we left the details of the proof as an exercise for the reader.

Consequently, we have

**Theorem 7 (Completeness)** *Let  $A$  be a formula. If  $A$  valid then  $A$  has a tableau proof.*

PROOF. Suppose  $A$  has no tableau proof. Then there is no closed tableau for  $\{(\lambda, \varepsilon, \lambda, \varepsilon) \neg A\}$ . Remark that  $\{(\lambda, \varepsilon, \lambda, \varepsilon) \neg A\}$  belongs to the set  $\mathcal{C}$  of all consistent sets of prefixed formulae. Obviously,  $\mathcal{C}$  is a consistency property. Therefore  $\{(\lambda, \varepsilon, \lambda, \varepsilon) \neg A\}$  is  $\mathcal{C}$ -consistent. By theorem 6,  $\{(\lambda, \varepsilon, \lambda, \varepsilon) \neg A\}$  is satisfiable. Therefore,  $A$  is not valid.

## 5 Axiomatization of a fragment

From now on, we study the language that results when the formalism does not include elementary programs like  $\mathbf{send}(m)$  or  $\mathbf{rec}(m)$ . We wish to define the relation “ $A$  is deducible”, denoted  $\vdash A$ , and show that  $\vdash A$  iff  $A$  is valid. Let  $\mathcal{L}$  be the least normal modal logic that contains the axioms given below:

- $[\lambda \parallel \lambda]A \leftrightarrow A$ ;
- $[A? \parallel \lambda]B \leftrightarrow (A \rightarrow B)$ ;
- $[\lambda \parallel A?]B \leftrightarrow (A \rightarrow B)$ ;
- $[\lambda \parallel \varphi; \beta]A \leftrightarrow [\lambda \parallel \varphi][\lambda \parallel \beta]A$ ;
- $[\varphi; \alpha \parallel \lambda]A \leftrightarrow [\varphi \parallel \lambda][\alpha \parallel \lambda]A$ ;
- $[\varphi; \alpha \parallel \psi; \beta]A \leftrightarrow [\varphi \parallel \lambda][\alpha \parallel \psi; \beta]A \wedge [\lambda \parallel \psi][\varphi; \alpha \parallel \beta]A$ ;
- $[\alpha(\alpha_1 \cup \alpha_2) \parallel \beta]A \leftrightarrow [\alpha(\alpha_1) \parallel \beta]A \wedge [\alpha(\alpha_2) \parallel \beta]A$ ;
- $[\alpha \parallel \beta(\beta_1 \cup \beta_2)]A \leftrightarrow [\alpha \parallel \beta(\beta_1)]A \wedge [\alpha \parallel \beta(\beta_2)]A$ ;
- $[\pi_1 \parallel \lambda][\lambda \parallel \pi_2]A \leftrightarrow [\lambda \parallel \pi_2][\pi_1 \parallel \lambda]A$ ;
- $\langle \pi_1 \parallel \lambda \rangle [\lambda \parallel \pi_2]A \rightarrow [\lambda \parallel \pi_2] \langle \pi_1 \parallel \lambda \rangle A$ .

Formula  $A$  is deducible, in symbols  $\vdash A$ , if  $A$  can be derived from the axioms of  $\mathcal{L}$  by applying the inference rules of  $\mathcal{L}$ . Obviously models satisfy the conditions which are needed to verify the axioms of  $\mathcal{L}$  and the rules of inference of  $\mathcal{L}$  are correct with respect to validity. As a result:

**Theorem 8 (Soundness of  $\mathcal{L}$ )** *Let  $A$  be a formula. If  $\vdash A$  then  $A$  is valid.*

The completeness of  $\mathcal{L}$  is more difficult to establish than its soundness and we defer proving it till the end of this section. The canonical frame for  $\mathcal{L}$  is the relational structure of the form  $\mathcal{F}' = (W', R')$  where:

- $W'$  is the set of all maximal consistent sets of formulas.

- $R'$  is a function assigning to each pair  $(\alpha, \beta)$  of programs the binary relation  $R'_{\alpha\|\beta}$  on  $W'$  such that  $s'R'_{\alpha\|\beta}t'$  iff  $\{A: [\alpha \|\ \beta]A \in s'\} \subseteq t'$ .

The axioms of  $\mathcal{L}$  force the binary relations  $R'_{\alpha\|\beta}$  to satisfy the following conditions:

- $s'R'_{\lambda\|\lambda}t'$  iff  $s' = t'$ .
- $s'R'_{A\|\lambda}t'$  iff  $s' = t'$  and  $A \in s'$ .
- $s'R'_{\lambda\|A}t'$  iff  $s' = t'$  and  $A \in s'$ .
- $R'_{\varphi;\alpha\|\psi;\beta} = (R'_{\varphi\|\lambda} \circ R'_{\alpha\|\psi;\beta}) \cup (R'_{\lambda\|\psi} \circ R'_{\varphi;\alpha\|\beta})$ .
- $R'_{\alpha(\alpha_1 \cup \alpha_2)\|\beta} = R'_{\alpha(\alpha_1)\|\beta} \cup R'_{\alpha(\alpha_2)\|\beta}$ .
- $R'_{\alpha\|\beta(\beta_1 \cup \beta_2)} = R'_{\alpha\|\beta(\beta_1)} \cup R'_{\alpha\|\beta(\beta_2)}$ .
- $R'_{\pi_1\|\lambda} \circ R'_{\lambda\|\pi_2} = R'_{\lambda\|\pi_2} \circ R'_{\pi_1\|\lambda}$ .
- If  $s'R'_{\pi_1\|\lambda}t'$  and  $s'R'_{\lambda\|\pi_2}u'$  then there is  $v' \in W'$  such that  $t'R'_{\lambda\|\pi_2}v'$  and  $u'R'_{\pi_1\|\lambda}v'$ .

Consider any formula  $A$  such that  $\not\vdash A$ . Hence there is a maximal consistent set  $s'$  of formulas such that  $A \notin s'$ . Let us define by induction a sequence  $(W_1^0, W_2^0, R_1^0, R_2^0), (W_1^1, W_2^1, R_1^1, R_2^1), \dots$  of relational structures and a sequence  $f^0, f^1, \dots$  of homomorphisms that map  $W_1^0 \times W_2^0, W_1^1 \times W_2^1, \dots$  to  $W'$ .

**Basis:** Let  $W_1^0 = \{s_1^0\}$  and  $W_2^0 = \{s_2^0\}$  be singleton sets,  $R_1^0(\pi)$  and  $R_2^0(\pi)$  be the empty relations on  $W_1^0$  and  $W_2^0$  respectively, and  $f^0(s_1^0, s_2^0) = s'$ . It should be remarked that  $f_0$  is a homomorphism of  $(W_1^0, W_2^0, R_1^0, R_2^0)$  to the canonical frame for  $\mathcal{L}$ .

**Hypothesis:** Let  $i$  be a positive integer and  $f_i$  be a homomorphism of  $(W_1^i, W_2^i, R_1^i, R_2^i)$  to the canonical frame for  $\mathcal{L}$ .

**Horizontal step:** Let  $\langle s_1, s_2, \pi, t' \rangle$  be what we will call a horizontal defect in  $(W_1^i, W_2^i, R_1^i, R_2^i)$  with respect to  $f_i$  and the canonical frame for  $\mathcal{L}$ , i.e.  $s_1$  is in  $W_1^i$ ,  $s_2$  is in  $W_2^i$ ,  $\pi$  is an atomic program and  $t'$  is in  $W'$  such that:

- $f_i(s_1, s_2)R'_{\pi\|\lambda}t'$ ;
- There is no  $t_1 \in W_1^i$  such that  $s_1R_1^i(\pi)t_1$  and  $f_i(t_1, s_2) = t'$ .

Our aim is to put right this defect. In this respect, let  $W_1^{i+1} = W_1^i \cup \{t_1\}$  where  $t_1$  is a new element and  $W_2^{i+1} = W_2^i$ . If  $\pi' \neq \pi$  then let  $R_1^{i+1}(\pi') = R_1^i(\pi')$  whereas let  $R_1^{i+1}(\pi) = R_1^i(\pi) \cup \{(s_1, t_1)\}$ . For all atomic programs  $\pi'$ , let  $R_2^{i+1}(\pi') = R_2^i(\pi')$ . Finally, let  $f_{i+1}$  be the function that maps  $W_1^{i+1} \times W_2^{i+1}$  to  $W'$  defined in the following way. First, the restriction of  $f_{i+1}$  to  $W_1^i \times W_2^i$  must be equal to the function  $f_i$ . Second, what remains to be done is to define the value of  $f_{i+1}(t_1, s)$  and the value of  $V_{i+1}(t_1, s)$  for each  $s \in W_2^i$ . Let  $(s^0, \dots, s^K)$  be an enumeration of  $W_2^i$  such that  $s^0 = s_2$  and for all positive integers  $k$ , if  $0 < k \leq K$  then there is a unique positive integer  $l$  such that  $l < k$  and there is a unique atomic program  $\pi'$  such that either  $s^kR_2^i(\pi')s^l$  or  $s^lR_2^i(\pi')s^k$ . Such an enumeration can be obtained as follows: take the unique  $R_2^i$ -path starting from  $s_2^0$  and ending with  $s_2$ ; enumerate it backwards; enumerate all the other

elements in  $W_2^i$  in the order of their creation. We define  $f_{i+1}(t_1, s^k)$  for all positive integers  $k$  such that  $k \leq K$  by induction in the following way. Let  $f_{i+1}(t_1, s^0) = t'$ . Let  $k$  be a positive integer such that  $0 < k \leq K$ . Suppose that  $f_{i+1}(t_1, s^l)$  is defined for all positive integers  $l$  such that  $l < k$ . We know that there is a unique positive integer  $l$  such that  $l < k$  and there is a unique atomic program  $\pi'$  such that either  $s^k R_2^i(\pi')s^l$  or  $s^l R_2^i(\pi')s^k$ . We consider the following two cases.

**Case 1:**  $s^k R_2^i(\pi')s^l$ . In this case,  $f_i(s_1, s^k)R'_{\lambda\|\pi'}f_i(s_1, s^l)$  and  $f_i(s_1, s^l)R'_{\pi\|\lambda}f_i(t_1, s^l)$ . Hence there is  $u' \in W'$  such that  $f_i(s_1, s^k)R'_{\pi\|\lambda}u'$  and  $u'R'_{\lambda\|\pi'}f_i(t_1, s^l)$ . We define  $f_{i+1}(t_1, s^k) = u'$ .

**Case 2:**  $s^l R_2^i(\pi')s^k$ . In this case,  $f_i(s_1, s^l)R'_{\lambda\|\pi'}f_i(s_1, s^k)$  and  $f_i(s_1, s^l)R'_{\pi\|\lambda}f_i(t_1, s^l)$ . Hence there is  $u' \in W'$  such that  $f_i(s_1, s^k)R'_{\pi\|\lambda}u'$  and  $f_i(t_1, s^l)R'_{\lambda\|\pi'}u'$ . We define  $f_{i+1}(t_1, s^k) = u'$ .

It should be remarked that  $f_{i+1}$  is a homomorphism of  $(W_1^{i+1}, W_2^{i+1}, R_1^{i+1}, R_2^{i+1})$  to the canonical frame for  $\mathcal{L}$ .

**Vertical step:** Let  $\langle s_1, s_2, \pi, t' \rangle$  be what we will call a vertical defect in  $(W_1^i, W_2^i, R_1^i, R_2^i)$  with respect to  $f_i$  and the canonical frame for  $\mathcal{L}$ , i.e.  $s_1$  is in  $W_1^i$ ,  $s_2$  is in  $W_2^i$ ,  $\pi$  is an atomic program and  $t'$  is in  $W'$  such that:

- $f_i(s_1, s_2)R'_{\lambda\|\pi}t'$ ;
- For all  $t_2 \in W_2^i$ , if  $s_2 R_2^i(\pi)t_2$  then  $f_i(s_1, t_2) \neq t'$ .

This case is treated similarly to the horizontal step. In conclusion, let  $W_1 = \bigcup\{W_1^i: i \text{ is a positive integer}\}$  and  $W_2 = \bigcup\{W_2^i: i \text{ is a positive integer}\}$ . Let  $R_1$  be the function that maps each atomic program  $\pi$  to the binary relation  $R_1(\pi) = \bigcup\{R_1^i(\pi): i \text{ is a positive integer}\}$  and  $R_2$  be the function that maps each atomic program  $\pi$  to the binary relation  $R_2(\pi) = \bigcup\{R_2^i(\pi): i \text{ is a positive integer}\}$ . Let  $V$  be the valuation on  $W_1 \times W_2$  defined by  $V(p) = \{(s_1, s_2) : \text{there is a positive integer } i \text{ such that } s_1 \in W_1^i, s_2 \in W_2^i \text{ and } p \in V_i(s_1, s_2)\}$ . Finally, the function  $f$  that maps  $W_1 \times W_2$  to  $W'$  is defined as follows: if  $s_1$  is in  $W_1^i$  and  $s_2$  is in  $W_2^i$  then let  $f(s_1, s_2) = f_i(s_1, s_2)$ . Let  $\mathcal{M} = (W_1, W_2, R_1, R_2, V)$ .

**Theorem 9 (Completeness of  $\mathcal{L}$ )** *Let  $A$  be a formula. If  $A$  is valid then  $\vdash A$ .*

Nevertheless, the problem of finding a good axiomatization of the set of all valid formulae remains open. Now we address the issue of the decidability of the satisfiability problem for the fragment of our language. Following the line of reasoning suggested by Gabbay et al. in chapter 6 of [5], we can prove that if a formula of the fragment is satisfiable then it is satisfiable in a finite model. As a result we have the following theorem:

**Theorem 10 (Decidability)** *The satisfiability problem for the fragment of our language is decidable.*

PROOF. The decidability of our fragment becomes evident as soon as we observe that

- deducible formulae are recursively enumerable.
- formulae satisfiable in some finite model are recursively enumerable.

- deducible formulae and negations of formulae satisfiable in some finite model constitute a partition of our fragment.

Nevertheless it is well worth noting that the satisfiability problem for the formulae of our fragment is nonelementary, seeing that:

- the satisfiability problem for the formulae of  $K \times K$ , the smallest product of modal logic is nonelementary [6].
- the satisfiability problem for the formulae of  $K \times K$  can be reduced to the satisfiability problem for the formulae of our fragment.

## 6 Conclusion

We would like to emphasize that so far our main concern has been the connection between the semantical concepts of validity and satisfiability and the concepts of axiomatization/completeness and decidability/complexity. More precisely, we have given a complete axiomatization of a fragment of the language of our logic as well as a proof of its decidability. Much remains to be done. We wish to axiomatize and to decide the full language of our logic. In this respect, it should be remarked, that modalities of one of the following forms are not normal:  $[\mathbf{send}(m) \parallel \lambda]$ ,  $[\lambda \parallel \mathbf{send}(m)]$ . To illustrate the truth of this, one has only to mention that formula  $[\lambda \parallel \mathbf{rec}(m)]\perp$  is valid, whereas  $[\mathbf{send}(m) \parallel \lambda][\lambda \parallel \mathbf{rec}(m)]\perp$  is not. Important matters in the development of reasoning systems that are concerned with the evolution of the knowledge of agents that exchange messages imply that future work will also include encryption algorithms as they are used in cryptographic protocols as well as epistemic operator as they are used in the modal logics for reasoning about knowledge.

## 7 Acknowledgement

The results of this paper have been obtained during the visit of the first author in the faculty of computer science at the Paul Sabatier university at Toulouse, which makes possible the collaboration between the two authors. Special thanks are due to Andreas Herzig and the Lilac group for valuable comments and discussions.

## References

- [1] Kamel Adi, Mourad Debabbi, and Mohamed Mejri. A new logic for electronic commerce protocols. *Theoretical Computer Science*, 291:223–283, 2003.
- [2] Dan Benanav, Deepak Kapur, and Paliath Narendran. Complexity of matching problems. In Jean-Pierre Jouannaud, editor, *RTA*, volume 202 of *Lecture Notes in Computer Science*, pages 417–429. Springer, 1985.
- [3] Michael Burrows, Martín Abadi, and Roger Needham. A logic for authentication. In *Proceedings of the Royal Society of London*, volume 426, pages 233–271, 1989.

- [4] Melvin Fitting. *Proof Methods for Modal and Intuitionistic Logics*, volume 169 of *Synthese Library*. D. Reidel, 1983.
- [5] Dov M. Gabbay, Ágnes Kurucz, Frank Wolter, and Michael Zakharyashev. Many-Dimensional Modal Logics: Theory and Applications. To appear.
- [6] Dov M. Gabbay and Valentin B. Shehtman. Products of modal logics. I. *Log. J. IGPL*, 6(1):73–146, 1998.
- [7] Robert Goldblatt. *Logics of Time and Computation*, volume 7 of *Lecture Notes*. Center for the Study of Language and Information, 1987.
- [8] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2000.
- [9] Steve Kremer and Jean-François Raskin. A game-based verification of non-repudiation and fair exchange protocols. In *CONCUR 2001—concurrency theory (Aalborg)*, volume 2154 of *Lecture Notes in Comput. Sci.*, pages 551–565. Springer, Berlin, 2001.
- [10] David Monniaux. Abstracting cryptographic protocols with tree automata. In *SAS'99*, volume 1694 of *Lecture Notes of Computer Science*, pages 149–163. Springer, 1999.
- [11] Charles P. Pfleeger. *Security in computing*. Prentice-Hall, 1988.
- [12] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with finite number of sessions is NP-complete. To appear.
- [13] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1986.