A LOGIC OF BELIEF AND A MODEL CHECKING ALGORITHM FOR SECURITY PROTOCOLS

M. Benerecetti
DISA - Università degli Studi di Trento, Via Inama 5, 38100 Trento, Italy
bene@na.infn.it

F. Giunchiglia
DISA - Università degli Studi di Trento, Via Inama 5, 38100 Trento, Italy
fausto@cs.unitn.it

M. Panti
Istituto di Informatica, Università di Ancona, via Brecce Bianche,60131 Ancona, Italy
panti@inform.unian.it

L. Spalazzi
Istituto di Informatica, Università di Ancona, via Brecce Bianche,60131 Ancona, Italy
spalazzi@inform.unian.it

Abstract Model checking is a very successful technique which has been applied in the design and verification of finite state concurrent reactive processes. In this paper we show how this technique can be used for the verification of security protocols using a logic of belief. The underlying idea is to treat separately the temporal evolution and the belief aspects of principals. In practice, things work as follows: when we consider the temporal evolution of a principal we treat belief atoms (namely, atomic formulae expressing belief) as atomic propositions. When we deal with the beliefs of a principal A, we model its beliefs about another principal B as the fact that A has access to a representation of B as a process. Then, any time it needs to verify the truth value of some belief atom about B, e.g., $B_B\phi$, A simply tests whether, e.g., ϕ holds in its (appropriate) representation of B. Beliefs are essentially used to control the "jumping" among processes. Our approach allows us to reuse the technology and tools developed in model checking.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35533-7_26

Tommaso Bolognesi and Diego Latella (Eds.), *Formal Methods for Distributed System Development.*

1. INTRODUCTION

In this paper we show how model checking (see, e.g. [9]) can be used for the verification of security protocols using a logic of belief and time [2]. Our approach allows us to reuse with almost no variations all the technology and tools developed in model checking and it is on the line of the work by Fagin et al. [7]. We model *principals* participating to a protocol session as (concurrent reactive finite state) processes able to have beliefs. The specification of a principal has therefore two orthogonal aspects: a temporal aspect and a belief aspect. The key idea underlying our approach is to keep these two aspects separated. In practice things work as follows:

- When we consider the temporal evolution of a principal we treat belief atoms (namely, atomic formulae expressing belief) as atomic propositions. The fact that these formulae talk about beliefs is not taken into consideration.
- When we deal with the beliefs of a principal A, we model its beliefs about another principal B as the fact that A has access to a representation of B as a process. Then, any time it needs to verify the truth value of some belief atom about B, e.g., $B_B\phi$, a simply tests whether, e.g., ϕ holds in its (appropriate) representation of B. Beliefs are essentially used to control the "jumping" among processes. This operation is iterated in the obvious way in case of nested beliefs.

The application of model checking to security protocol verification is not new (see. e.g., [8, 10]). However, in the previous work, security protocols are verified by introducing the notion of intruder and, then, by verifying whether the intruder can attack a given protocol. This approach makes possible to directly find a trace of a possible attack but it may not be clear what the protocol flaw is. This work usually employs temporal logics or process algebras. Our approach is different, we do not use any model of intruders and our formalism is able to represent both time and belief. This work builds on the work described in [3], where model checking is applied to BDI attitudes (namely, Belief, Desire, and Intention) of agents. Various papers can be found in literature, where belief logics have been applied to testing security in protocol verification (see for instance [4]). However, in all the previous work (see, e.g., [4, 1]) verification is performed proof-theoretically.

The paper is structured as follows. In Section 2 we describe the logic of belief and time we employ (called MATL, MultiAgent Temporal Logic). We also briefly introduce a well-known protocol, the CCITT X.509 protocol, as a running example. Section 3 introduces finite presentations of MATL (called MAFSM, MultiAgent Finite State Machines). Section 4 describes the model checking procedure we propose. Finally, some conclusions are drawn in Section 5.

2. A LOGIC OF BELIEF AND TIME

We model principals engaged in an authentication session as finite state processes. We build the notion of principal incrementally over the notion of process. Suppose we have a set I of principals. Each principal is seen as a process having beliefs about (itself and) other principals. We adopt the usual syntax for beliefs: $B_i\phi$ means that principal i believes ϕ, and ϕ is a belief of i. B_i is the belief operator for i.

The idea is to associate to each (level of) nesting of belief operators a process evolving over time. Let $B = \{_B1, ..., _Bn\}$, where each index $1, ..., n \in I$ corresponds to a principal. Let B^* denote the set of finite strings of the form $_B1, ..., _Bn$ with $_Bi \in B$. We call any $\alpha \in B^*$, a *view*. Figure 1 depicts the general structure of the views. Each view in B^* corresponds to a possible nesting of belief operators. We also allow for the empty string, ϵ. The intuition is that ϵ represents the view of an external observer (e.g., the designer) which, from the outside, "sees" the behavior of the overall protocol. The beliefs of principal A correspond to the view $_BA$ and are modeled by a process playing A's role in a protocol. The beliefs of principal B (the view $_BB$) are modeled similarly. The beliefs that A has about (the behavior of) principal B correspond to the view $_BA_BB$ and are modeled by a process playing B's role in the protocol. Things work in the same way for any arbitrary nesting of belief operators.

We associate a language $\mathcal{L}_\alpha$ to each view $\alpha \in B^*$. Intuitively, each $\mathcal{L}_\alpha$ is the language used to express what is true (and false) of the process of view α. We employ the logic CTL [6], a well known propositional branching-time temporal logic widely used in formal verification. For each α, let P_α be a set of propositional atoms. Each P_α allows for the definition of a different language, called a MultiAgent Temporal Logic (MATL) language (on P_α). A MATL language $\mathcal{L}_\alpha$ on P_α is the smallest CTL language containing the set of propositional atoms P_α and the belief atoms $B_i\phi$, for any formula ϕ of $\mathcal{L}_{\alpha B_i}$. In particular, $\mathcal{L}_\epsilon$ is used to speak about the whole protocol. The language $\mathcal{L}_{B_i}$ ($\mathcal{L}_{B_j}$) is the language adopted to represent i's (j's) beliefs. i's beliefs about j's beliefs are specified by the language of the view $_Bi_Bj$. Given a family $\{P_\alpha\}$ of sets of propositional atoms, the family of MATL languages on $\{P_\alpha\}$ is the family of CTL languages $\{\mathcal{L}_\alpha\}$. We write $\alpha : \phi$ (called labeled wff) to mean that ϕ is a formula of $\mathcal{L}_\alpha$.

Example 2.1 MATL is expressive enough to capture most security logics. For instance, we can easily express in MATL the analysis of the CCITT X.509 protocol, as described in [4]. N_i denotes a nonce (a fresh message) newly created by principal i for the current session; T_i is a timestamp which represents the time at which the message was generated; X_i and Y_i are user data. In public key cryptography, each principal i has a *public key*, denoted K_i, which any other principal can obtain from a key server; it also has a *private secret key*,

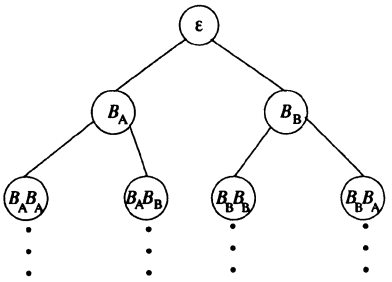

Figure 1 The set of views for the CCITT pro-tocol.

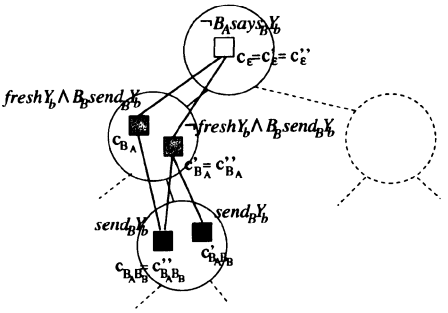

Figure 2 Some chains of the CCITT protocol.

$K_i{}^{-1}$, which is the inverse of K_i. Any message encrypted with the public key, $\{M\}_{K_i}$, can be decrypted only by i with its own private key; vice versa, any message encrypted (signed) by i with its private key, $\{M\}_{K_i{}^{-1}}$, can be decrypted by any principal with the public key K_i. The CCITT X.509 protocol can be formulated as follows:

$$\begin{array}{lll} 1 & A \rightarrow B & : \{T_a, N_a, X_a, \{Y_a\}_{K_b}\}_{K_a{}^{-1}} \\ 2 & B \rightarrow A & : \{T_b, N_b, N_a, X_b, \{Y_b\}_{K_a}\}_{K_b{}^{-1}} \\ 3 & A \rightarrow B & : \{N_b\}_{K_a{}^{-1}} \end{array}$$

Intuitively, with Message 1, A sends B the data X_a and Y_a. Y_a must be secret and thus it is encrypted with K_b. The whole message is signed with $K_a{}^{-1}$ to assure that the sender is A. Similarly, B sends A the data X_b and Y_b with Message 2. We then set the atoms P_α for the view $_{B_A}$ as follows:

$$P_{B_A} = \left\{ \begin{array}{l} send_B \{T_a, N_a, X_a, \{Y_a\}_{K_b}\}_{K_a{}^{-1}}, \\ rec \{T_b, N_b, N_a, X_b, \{Y_b\}_{K_a}\}_{K_b{}^{-1}}, \\ fresh \{T_b, N_b, N_a, X_b, \{Y_b\}_{K_a}\}_{K_b{}^{-1}}, \\ fresh\ N_a, \\ pubk_B\ K_b, \\ \ldots \end{array} \right\}$$

where the variables of the form *rec* M or $send_B\ M$ (where M is a (sub)message of the CCITT protocol) are called *message variables* and represent the act of receiving M, and sending M to B, respectively. For instance, the atom $send_B \{T_a, N_a, X_a, \{Y_a\}_{K_b}\}_{K_a{}^{-1}}$ in view $_{B_A}$ represent A sending Message 1 to B. Variables of the form *fresh* M (e.g., *fresh* N_a in view $_{B_A}$) mean that M (e.g., N_a) is a fresh (sub)message. Variables of the form $pubk_i\ K$ (e.g., $pubk_B\ K_b$in view $_{B_A}$) mean that K is the public key of i. The atomic propositions of the other views can be defined similarly. Since each view α_{B_i} (with $i = A, B$) models the (beliefs about the) behavior of principal i, the set of atomic proposition will be that of view $_{B_i}$ (see [2]).

In MATL we can formalize how messages sent and received during a protocol session by a party may affect its beliefs about the other parties. For instance, one might want to express the following property: at the end of a protocol session, A believes that B recently "said" X_b and Y_b. Formally:

$$\epsilon : \mathsf{AG}\, send_{A,B}\, \{N_b\}_{K_a{}^{-1}} \rightarrow B_A says_B\, X_b \wedge B_A says_B\, Y_b \qquad (1)$$

where, following [1], for each message M, in the view modeling the beliefs of principal j, the proposition $says_i\, M$ is defined as $fresh M \wedge B_i send_j M$.

Semantics. To understand the semantics of the family of languages $\{\mathcal{L}_\alpha\}_{\alpha \in B^*}$ (hereafter we drop the subscript), we need to understand two key facts. On the one hand the semantics of formulae depend on the view. For instance, the formula p in the view $_{B_i}$ expresses the fact that i believes that p is true. The same formula in the view $_{B_j}$ expresses the fact that j believes that p is true. As a consequence, the semantics associates *locally* to each view α a set of pairs $\langle m, s\rangle$, where: $m = \langle S, J, R, L\rangle$ is a CTL structure, with S a set of states, $J \subseteq S$ the set of *initial states*, R the transition relation, and $L : S \rightarrow \mathcal{P}(P)$ the *labeling function*; and s is a *reachable state* of m (a state s of a CTL structure is said to be reachable if there is a path leading from an initial state of the CTL structure to state s). On the other hand there are formulae in different views which have the same intended meaning. For instance $B_j p$ in view $_{B_i}$, and p in view $_{B_i B_j}$ both mean that i believes that j believes that p is true. This implies that only certain interpretations of different views are *compatible* with each other, and these are those which agree on the truth values of the formulae with the same intended meaning. To capture this notion of compatibility we introduce the notion of chain.

Let α be any view, a α*−chain* c is a finite sequence $\langle c_\epsilon, ..., c_\beta, ..., c_\alpha\rangle$, where $c_\beta = \langle m, s\rangle$ is an interpretation for $\mathcal{L}_\beta$ and β is a prefix of α (i.e., $\alpha = \beta\gamma$ for some index $\gamma \in B^*$). A *compatibility relation* C *on* $\{\mathcal{L}_\alpha\}$ is a set of α−chains, for every α. Intuitively, C will contain all those c's whose elements c_α, c_β (where α, β are two views in B^*) assign the same truth values to the formulae with the same intended meaning.

Example 2.2 Figure 2 (see previous page) shows some possible chains of the MATL structure for the CCITT protocol. The boxes in each view represent interpretations of the language associated with the corresponding view. The figure shows an interpretation for view ϵ, two interpretations for view $_{B_A}$ and two interpretations for view $_{B_A B_B}$. Links connecting boxes in different views represent $_{B_A B_B}$−chains. Figure 2 shows three $_{B_A B_B}$−chains, $c = \langle c_\epsilon, c_{B_A}, c_{B_A B_B}\rangle$, $c' = \langle c'_\epsilon, c'_{B_A}, c'_{B_A B_B}\rangle$ and $c'' = \langle c''_\epsilon, c''_{B_A}, c''_{B_A B_B}\rangle$, where $c_\epsilon = c'_\epsilon = c''_\epsilon$, $c'_{B_A} = c''_{B_A}$ and $c_{B_A B_B} = c''_{B_A B_B}$. Let us assume that the each interpretation satisfies the formula written close to it in figure. Therefore, the interpretation labeled c'_{B_A} $(= c''_{B_A})$ satisfies the formula $B_B send_A Y_b$. The

intended meaning of this formula in view $_{B_A}$ is that A believes that B believes it has sent message Y_b to A. The formula $send_A Y_b$ in view $_{B_A B_B}$ has the same intended meaning, as $_{B_A B_B}$ models the beliefs of B seen from (the beliefs of) A. Therefore any $_{B_A B_B}-$chain passing through the interpretation c'_{B_A} must reach an interpretation in $_{B_A B_B}$ which satisfies the argument $send_A Y_b$, as shown in the figure.

Let us now define the notion of satisfiability. We start with satisfiability local to views (first step) and suppose that for each view α there is a satisfiability relation between CTL structures and formulae of $\mathcal{L}_\alpha$. With an abuse of notation, we denote all these satisfiability relations with the same symbol $\models$. The context always makes clear which relation we mean. The second step is to define (global) satisfiability taking into account chains. To do this we need some further notation. Let $\models$ denote satisfiability also on chains. For any $\alpha-$chain c and for any formula in $\mathcal{L}_\beta$, satisfiability relation $\models$ is defined only when either α is a prefix of β or β is a prefix of α. (i.e., when either $\alpha = \beta\gamma$ or $\beta = \alpha\gamma$). If $\alpha = \beta\gamma$ then $c_\beta \models \phi$ iff ϕ is true in c_β. If $\beta = \alpha\gamma$ then $c_\beta \models \phi$ for any ϕ. In other words, if a chain stops at a given level (e.g. α), then it satisfies every formula of the views (e.g., $\alpha\gamma$) which are below that level in the tree. Let us extend the satisfiability relation to sets of formulae: $x \models Y$ if and only if for any $y \in Y$, $x \models y$.

We are now ready to define the notion of model for MATL (called MATL structure), and then that of satisfiability between MATL structures and formulae of a view.

Definition 2.1 (MATL structure) *A nonempty compatibility relation C for a family of MATL languages on $\{P_\alpha\}$ is a* MATL structure *on $\{P_\alpha\}$ if for any $\alpha\beta-$chain $c \in C$,*

1. *$c_\alpha \models B_i\phi$ iff for every $\alpha\gamma-$chain $c' \in C$, $c'_\alpha = c_\alpha$ implies $c'_{\alpha B_i} \models \phi$;*
2. *if $c_\alpha = \langle m, s\rangle$, then for any state s' of m, there is a $\alpha\beta-$chain $c' \in C$ such that $c'_\alpha = \langle m, s'\rangle$.*

Briefly: the nonemptyness condition for C guarantees that the external observer has a consistent view of the world. The *only if* part in condition 1 guarantees that each view has correct beliefs, i.e., any time $B_i\phi$ holds at a view then ϕ holds in the view one level down in the chain. The *if* part is the dual property and ensures the completeness of each view. Condition 2 can be understood on the basis of two crucial observations, concerning the mutual nesting of CTL operators and belief operators. The first, concerning the nesting of CTL operators inside belief operators is that $c_\alpha \models \phi$ is computed using the notion of satisfiability in a CTL structure. Therefore, a chain links the fact that a belief atom holds in a state of a CTL structure in one view with the fact that its argument holds in a state of a CTL structure in the view below. The

second observation concerns the nesting of belief operators inside temporal operators (temporal operators which involve no belief atoms are treated as in CTL structures, that is, without jumping among views). Consider for instance the formula $\mathsf{EX}\, B_i p$. To assess the truth of $\mathsf{EX}\, B_i p$ we need to be able to assess the truth of $B_i p$ in some future (reachable) state s' of the CTL structure we are considering, e.g., $\langle\langle S, J, R, L\rangle, s\rangle$. The only way to establish this is to request that in s' we have a chain c' which gives access to a CTL structure in the view below. Given the fact that chains connect CTL structures only for what holds in their (reachable) states, the only solution is to request that s' is the state component of the structure $c'_\epsilon = \langle\langle S, J, R, L\rangle, s'\rangle$ with $c' \in C$. Given the fact that temporal operators allow us to state facts about all the states in a CTL structure, this operation must be repeated for each state $s \in S$. But this is exactly what Item 2 of Definition 2.1 says.

Example 2.3 Consider again Figure 2. The formula $\neg B_A says_B Y_b$ is satisfied by the interpretation in view ϵ. The formula $B_A says_B Y_b$ is defined as $B_A(fresh Y_b \wedge B_B send_A Y_b)$. By Item 1 of Definition 2.1 (only if direction), there must be a chain starting from the interpretation in view ϵ whose component for view $_{BA}$ does not satisfy the argument of the belief, namely $fresh Y_b \wedge B_B send_A Y_b$. This is indeed the case for both c' and c'', as $c'_{B_A} = c''_{B_A}$ and both of them do not satisfy $fresh Y_b$. Assume now that c' and c'' are the only two chains passing through interpretation c'_{B_A}. The components for view $_{BABB}$ of both chains passing through that interpretation, namely $c'_{B_A B_B}$ and $c''_{B_A B_B}$, satisfy the argument $send_B Y_b$. Thus, by Item 1 of Definition 2.1 (if direction), c'_{B_A} must satisfy $B_B send_B Y_b$, as shown in Figure 2.

Given a MATL structure C, a formula ϕ and a view α, $C \models \alpha : \phi$ is read as ϕ is true in C (or equivalently, ϕ holds in C, or ϕ is satisfied by C) at view α, and it is defined as follows:

$$C \models \alpha : \phi \text{ iff for all } c \in C \text{ s.t. } c_\alpha = \langle\langle S, J, R, L\rangle, s\rangle \text{ and } s \in J,\ c_\alpha \models \phi \tag{2}$$

The intuition is that in order to check the satisfiability of ϕ at the view α we need to check all the interpretations of $\mathcal{L}_\alpha$ allowed by the compatibility imposed by the chains we are considering.

3. MULTIAGENT FINITE STATE MACHINES

We are interested in extending CTL model checking to the model checking of belief formulae. In model checking, finite state processes are modeled as finite state machines. A *finite state machine* (FSM) is a finite CTL structure $f = \langle S, J, R, L\rangle$. Our solution is to extend the notion of FSM to that of *Multi-*

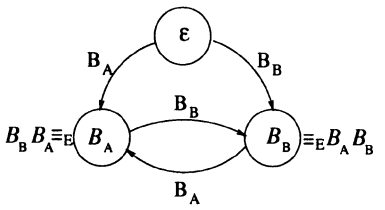

Figure 3 The set of views for the CCITT protocol.

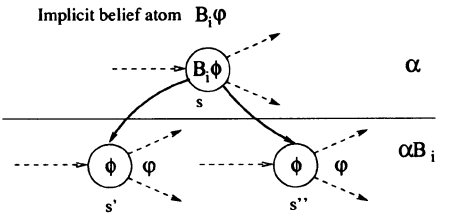

Figure 4 Explicit belief atoms and satisfiability.

Agent Finite State Machine (MAFSM), where, roughly speaking, a MAFSM is a finite set of FSMs.

A first step in this direction is to restrict ourselves to a finite number of views α. Let B^n denote a finite subset of B^* obtained by taking the views in any finite subtree of B^* rooted at ϵ. This restriction is not enough, as a finite set of views still allows for an infinite number of belief atoms. Even if we had a finite number of processes we would not be able to model them as FSMs. This problem can be solved introducing the notion of *explicit belief atoms* as a finite subset of the set of belief atoms. Explicit belief atoms are the only belief atoms which are explicitly represented in a FSM, allowing for a finite set of states. As we explain below, the truth value of all the other belief atoms not explicitly represented in a FSM is computed starting from explicit beliefs. Formally, if $\mathcal{L}_\alpha$ is a MATL language, then for each belief operator B_i, the set of *explicit belief atoms* of B_i for α (in symbols $Expl(B_i, \alpha)$) is a (possibly empty) *finite* subset of the belief atoms of $\mathcal{L}_\alpha$.

In this way, though, one can only model systems whose specifications must be expressed with an "a priori" bound on the nesting of belief operators. As a consequence, it would not be possible to deal with, e.g., (infinite) positive/negative introspection or an arbitrary nesting of belief operators. Moreover, each view must be specified separately even when different views of principals happen to behave in the same way. One way to overcome these limitations is to extend the finite tree of views to a graph of views. Cycles in the graph allow us to represent arbitrary nesting of beliefs and to model equivalent views using the same specification.

Example 3.1 Consider for instance the CCITT protocol as represented in Figure 3. The view B_B is modeled by a process whose (possible) behaviors are those of B as specified by the protocol. View $B_A B_B$ is modeled by a process whose (possible) behaviors are those of B as seen from A's point of view. Since protocols are usually supposed to be publicly known, and each honest principal's behavior is assumed to comply to the protocol, those two views in the CCITT protocol should exhibit the same set of behaviors. In other words, the two views are behaviorally equivalent. Therefore, both views can be mod-

eled by the same process. Similar argument can be given for B_A and for any level of nesting of belief operators. The resulting graph is represented in Figure 3.

Let $Front(B^n)$ be the set of views of B^* not in B^n which are immediately below a view in B^n. Each view in $Front(B^n)$ will be the first view, in a path of the tree of views, that do not belong to B^n. Formally $Front(B^n) = \{\alpha B_i \mid \alpha \in B^n \text{ and } \alpha B_i \notin B^n\}$.

Let E be any functional relation included in $Front(B^n) \times B^n$. The intuitive interpretation of the relation E between two views α and β (i.e., $E(\alpha, \beta)$) is that view α is "behaviorally equivalent" to view β. In other words α can be modeled by the same process as β. Intuitively, every time we need to verify satisfiability of a formula of (the process modelling) view α, we need to jump to view β to assess the truth of that formula. We write $\equiv_E$ to mean the smallest equivalence relation on B^* including E. Let $[\ ]_E$ be the equivalence class induced by $\equiv_E$ on B^*. By means of the equivalence relation induced by E, any view below α can be considered behaviorally equivalent to the appropriate view below β. That is, suppose we have to verify a formula in a given view α which doesn't belong to B^n but is equivalent (via $\equiv_E$) to a view $\beta \in B^n$ (that is, there is a view $\beta \in B^n$ with $[\beta]_E = [\alpha]_E$). In this case, we would jump to β and verify the given formula in that view.

Example 3.2 See again Figure 3. The protocol considered in the example involves two principals, A and B. Therefore, we have $I = \{A, B\}$. We can consider only three views in B^n, namely ϵ, B_A and B_B. ϵ (the external observer) is modeled as a process which "sees" all the messages sent and received by the principals. B_A models the behavior of principal A, while B_B models the behavior of principal B. In this case, we set $Front(B^n) = \{B_A B_A, B_A B_B, B_B B_A, B_B B_B\}$. The beliefs of B (A, respectively) about A (about B, respectively) are then modeled by the view B_A (B_B, respectively). Therefore, both $E(B_A, B_B B_A)$ and $E(B_B, B_A B_B)$ holds. Figure 3 illustrates this intuition by representing the relation E as arrows connecting the views B_A with B_B. Each arrow connects two views and is labeled with a belief operator. The arrow with label B_A from view ϵ to view B_A tells us that the beliefs the external observer has about principal A are represented by view B_A. The arrow from view B_A to view B_B labeled B_B means that the beliefs that A has about B (corresponding to view $B_A B_B$ in MATL) are modeled by view B_B, as expressed by the relation $E(B_A, B_B B_A)$. Similarly for the arrow linking views B_B and B_A in the other direction. As a consequence, the equivalence class $[B_B]_E$ contains, for instance, the view $B_A B_B$, while $[B_A]_E$ contains view $B_B B_A$. Whenever we need to verify a formula in the language of $B_A B_B$ ($B_B B_A$, respectively) we need to jump to view B_B (B_A, respectively) and verify the formula in that view.

Thus, we have the following:

Definition 3.1 (MAFSM) *Let $B^n \subset B^*$ be a finite subtree of views rooted at ϵ. A* **MAFSM** *is a pair $MF = \langle E, F \rangle$ where: E is any functional relation included in $Front(B^n) \times B^n$ and $F = \{F_\alpha\}$ is total recursive function such that:*

1. *$F_\epsilon \neq \emptyset$;*
2. *for all views $\alpha \in B^n$, it associates a finite set F_α of FSMs on the MATL language on the following atoms: P_α and, for all $i \in I$, $Expl(B_i, \alpha)$;*
3. *for all the views $\alpha \in B^* \setminus B^n$, $F_\alpha = \emptyset$.*

The first condition on F ensures that the protocol specification is not empty; the second allows us to deal, in each view, with finite sets of FSMs; and the third allows us to deal with a finite number of views. In general, there may be more than one FSM associated to each view. This allows for situations in which a view can be only partially specified, and consequently there can be more than one process modeling that view. If it is completely specified, a view contains only one FSM.

Let us extend the set $Expl(B_i, \alpha)$ and the function F to the equivalence classes induced by E over B^* in the following way:

$$Expl(B_i, [\alpha]_E) = \begin{cases} Expl(B_i, \beta) & \text{if } \beta \in [\alpha]_E \cap B^n \\ \emptyset & \text{otherwise} \end{cases} \qquad F_{[\alpha]_E} = \begin{cases} F_\beta & \text{if } \beta \in [\alpha]_E \cap B^n \\ \emptyset & \text{otherwise} \end{cases}$$

Intuitively, $Expl(B_i, [\alpha]_E)$ ($F_{[\alpha]_E}$, respectively) is the set of explicit belief atoms (set of FSMs, respectively) of a view β belonging to B^n in the equivalent class of α, if it exists, the empty set, otherwise. Notice that, if it exists, β must be unique. This follows from the fact that $\equiv_E$ is smallest equivalence relation including E and from the fact that E is a functional relation. Therefore there can be at most one view belonging to B^n equivalent to a view in $B^* \setminus B^n$.

In the rest of the paper we assume that each set of explicit belief atoms only contains belief atoms of depth one, i.e., atoms of the form $B_i\phi$, where ϕ does not contain other belief atoms. This is clearly a restriction in general, but it greatly simplifies the framework when cycles are allowed, while being sufficient for modeling most security protocols (see, e.g., the informal discussion in [2]).

Example 3.3 We need to choose the explicit belief atoms of each view. In general, the choice of the appropriate set of explicit belief atoms of (the views in) a MAFSM depends on what kind of aspects of the protocol one wants to analyze, and on what kind of properties need to be verified. In the case of security protocols, principals can only gain information carried by the messages they receive. We choose, therefore, the beliefs about other principals having

sent or received a given message as explicit beliefs atoms. In our case, we have:

$$Expl(B_B,B_A)=\left\{ \begin{array}{l} B_B send_A \{T_b, N_b, N_a, X_b, \{Y_b\}_{K_a}\}_{K_b{}^{-1}}, \\ \ldots \end{array} \right\}$$

where, for instance, $B_B send_A \{T_b, N_b, N_a, X_b, \{Y_b\}_{K_a}\}_{K_b{}^{-1}}$ in $_{B_A}$ intuitively means that A believes that B believes that it has sent Message 2 of the CCITT protocol to A. Similar explicit belief atoms are contained in view $_{B_A}$ and in view ϵ, the letter concerning beliefs of each principal about messages it sends and receives.

Given the notion of MAFSM, the next step is to give a notion of satisfiability in a MAFSM. We start from the notion of satisfiability of CTL formulae in an FSM at a state (defined as in CTL structures). Since FSMs are built on the propositional and explicit belief atoms of a view, to assess satisfiability of the propositional and explicit belief atoms (and the CTL formulae build out of them) we do not need to use the machinery associated to belief operators. However, this machinery is needed in order to deal with the (infinite) number of belief atoms which are not memorized anywhere in MAFSM. Let the set of *implicit belief atoms* of a view α, written $Impl(_{B_i}, \alpha)$, be defined as the (infinite) subset of all belief atoms of $\mathcal{L}_\alpha$ which are not explicit belief atoms. Formally:

$$Impl(_{B_i}, \alpha) = \{B_i\phi \in \mathcal{L}_\alpha \setminus Expl(_{B_i}, \alpha)\}$$

The idea is to use the information explicitly contained in the labeling function of each state s of a FSM f of a view α to assess the truth value of the implicit belief atoms at a state s. Figure 4 illustrates the underlying intuition. Intuitively, the principal modeled by FSM f (in view α), when in state s, ascribes to principal i the explicit belief atoms of the form $B_i\phi$ true at s. This means that the FMSs of view α_{B_i}, which model the beliefs of i, must be in any of the states (s' and s'' in Figure 4) in which the formulae ϕ, occurring as arguments of the explicit belief atoms, are true. This motivates the following definitions. Let $ArgExpl(_{B_i}, \alpha, s)$ be defined as follows:

$$ArgExpl(_{B_i}, \alpha, s)=\{\phi \in \mathcal{L}_{\alpha_{B_i}} \mid B_i\phi \in L(s) \cap Expl(_{B_i}, \alpha)\}$$

Thus, $ArgExpl(_{B_i}, \alpha, s)$ consists of all the formulae $\phi \in \mathcal{L}_{\alpha_{B_i}}$ such that the explicit belief atom $B_i\phi$ is true at state s (i.e., it belongs to the labeling function of s). $ArgExpl(_{B_i}, \alpha, s)$ contains the formulae which identify the states in which the FSMs in view α_{B_i} can be, whenever the process in view α is in state s.

We are now ready to define the notion of satisfiability of implicit belief atoms. Let $B_i\psi$ be an implicit belief atom of a view α. For each state s of a FSM of α, we can compute $ArgExpl(_{B_i}, \alpha, s)$. As shown in Figure 4, we just need to check whether all the reachable states of the FSMs of view α_{B_i}, which satisfy $ArgExpl(_{B_i}, \alpha, s)$ (namely, the set $\{\phi\}$ in Figure 4), also satisfy the argument ψ of the implicit belief atom. If this is the case, then s satisfies $B_i\psi$.

Definition 3.2 (Satisfiability in a MAFSM) *Let MF be a MAFSM, α a view, $f = \langle S, J, R, L\rangle \in F_{[\alpha]}$ an FSM, and $s \in S$ a state. Then, for any formula ϕ of $\mathcal{L}_\alpha$, the satisfiability relation $MF, \alpha, f, s \models \phi$ is defined as follows:*

1. *$MF, \alpha, f, s \models p$, where p is a propositional atom or an explicit belief atom: the same as FSM satisfiability;*
2. *satisfiability of propositional connectives and CTL operators: the same as FSM satisfiability;*
3. *$MF, \alpha, f, s \models B_i\phi$ if and only if for all $f' \in F_{[\alpha B_i]}$ and s' state of f', $MF, \alpha_{B_i}, f', s' \models \bigwedge ArgExpl(B_i, \alpha, s) \rightarrow \phi$, where $B_i\phi$ is an implicit belief atom,*

In the definition of satisfiability above, Item 3 is the crucial step. In the definition $\bigwedge ArgExpl(B_i, \alpha, s)$ denotes the conjunction of all the elements of the set $ArgExpl(B_i, \alpha, s)$. Notice that, under the assumption that the set of explicit belief atoms contains only belief atoms of depth one, the definition of satisfiability of implicit belief atoms above is well defined.

4. THE MODEL CHECKING ALGORITHM

The basic operation of a standard CTL model checking algorithm is to extend the labeling function of an FSM (which considers only propositional atoms) to all the (atomic and not atomic) subformulae of the formula being model checked. Let us call Extended FSM (or, simply, FSM when the context makes clear what we mean) the result of this operation. The generation of an extended FSM relies on the fact that the labeling function explicitly defines the truth value of all atoms. The problem is that in the FSMs of a MAFSM the labeling function is not defined on implicit belief atoms, whose truth value is therefore left undefined; and that we need to know the truth values of the implicit belief atoms occurring in the formula to be model checked. The definition of satisfiability in a MAFSM (Item 3 in Definition 3.2) tells us how to solve this problem.

The crucial observation is that $ArgExpl(B_i, \alpha, s)$, in Item 3 of Definition 3.2, is generated from the formulae in $Expl(B_i, \alpha)$ and the labeling functions of the FSMs in α; $ArgExpl(B_i, \alpha, s)$ is a finite set; and it only depends on the MAFSM specified (and thus independent of the formula to be model checked). For each belief operator B_i, C_{B_i} is called the *(MAFSM) compatibility relation* of B_i, and it is a relation defined as follows. Let $ex \subseteq Expl(B_i, \alpha)$ be a subset of the explicit belief atoms of a view $\alpha \in B^n$ and $MF = \langle E, F\rangle$ a MAFSM. The notion of compatibility relation just defined for a MAFSM is the dual notion of the compatibility relation defined in MATL. Then:

$$C_{B_i}(\alpha, ex) = \left\{ \begin{array}{c} \langle f', s'\rangle \mid f' \in F_{\alpha B_i}, s' \text{ a reachable state of } f' \text{ and} \\ MF, \alpha_{B_i}, f', s' \models \{\phi \mid B_i\phi \in ex \end{array} \right\}$$

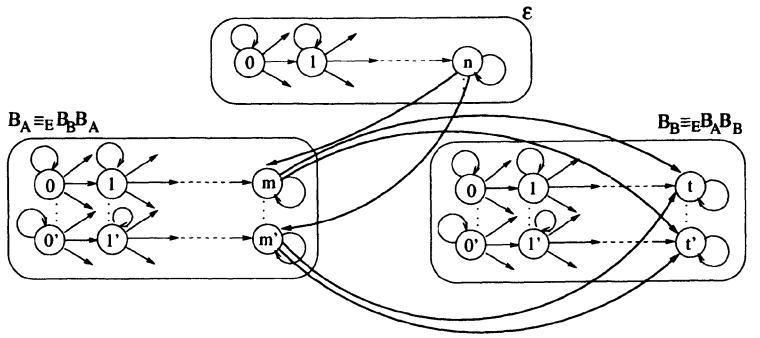

Figure 5 The MAFSM for the CCITT protocol.

Starting from a view α and a subset of explicit belief atoms ex of α, $C_{B_i}(\alpha, ex)$ collects all the FSMs f' and reachable states s' of f' (in the view α_{B_i}) which satisfy the arguments of the explicit belief atoms ex (formally: $\{\phi \mid B_i\phi \in ex\}$). Intuitively, $C_{B_i}(\alpha, ex)$ contains all the pairs $\langle f', s'\rangle$ of view α_{B_i} where the process associated to view α_{B_i} can be, whenever the process modeling view α is in a state that satisfies all the explicit belief atoms in ex. It can be easily seen that, for f' a FSM of view α_{B_i}, s' any reachable state of f', and s a state of a FSM of view α:

$$\langle f', s'\rangle \in C_{B_i}(\alpha, L(s) \cap Expl(B_i, \alpha)) \text{ iff } s' \text{ is a reachable state of } f' \text{ and } MF, \alpha_{B_i}, f', s' \models \bigwedge ArgExpl(B_i, \alpha, s)$$

where $L(s) \cap Expl(B_i, \alpha)$ is the set of explicit belief atoms true at state s in the view $\alpha \in B^n$. Hence, the following holds:

$$MF, \alpha, f, s \models B_i\phi \text{ iff for all } \langle f', s'\rangle \in C_{B_i}([\alpha]_E, L(s) \cap Expl(B_i, [\alpha]_E)),\ MF, \alpha_{B_i}, f', s' \models \phi \tag{3}$$

where $\alpha \in B^*$ and $B_i\phi$ is any implicit belief atom of view α. For any set Γ of formulae of $\mathcal{L}_\alpha$, $C_{B_i}([\alpha]_E, \Gamma)$ is the extension of the compatibility relation C_{B_i} to the equivalence classes induced by $\equiv_E$:

$$C_{B_i}([\alpha]_E, \Gamma) = \begin{cases} C_{B_i}(\beta, \Gamma) & \text{if } \beta \in [\alpha]_E \cap B^n \\ \emptyset & \text{otherwise} \end{cases}$$

Example 4.1 Figure 5 represents the underlying intuition on the MAFSM for the CCITT protocol. The relation $C_{B_i}(\alpha, L(s) \cap Expl(B_i, \alpha))$ is depicted as arrows connecting states of different views. Let us consider view ϵ. When the process associated to ϵ is in state n, the external observer believes that principal A, modeled by view B_A, can be in any one of the states m and m' of the FSM of that view, which are compatible with state n. The states of view B_A are completely identified by the explicit beliefs of ϵ true at state n. Therefore, ϵ

Algorithm MAMC-View(α, Γ)

$Sub := \bigcup\{sub(\phi) \mid \phi \in \Gamma\}$ } *Phase A*

for each B_i **do** } *Phase B*
  $ArgImpl(B_i, \alpha, Sub) := \{\phi \mid B_i\phi \in Sub \setminus Expl(B_i, [\alpha]_E)$
  if $ArgImpl(B_i, \alpha, Sub) \neq \emptyset$ **then**
    MAMC-View($\alpha_{B_i}, ArgImpl(B_i, \alpha, Sub)$)
  endif
end

for each $f \in F_{[\alpha]_E}$ **do** $/* f = \langle S, J, R, L\rangle */$ } *Phase C1*
  if ⟨Sub contains implicit belief atoms⟩ **then**
    for each $s \in S$ **do**
      for each B_i **do**
        $ArgImpl(B_i, \alpha, Sub) := \{\phi \mid B_i\phi \in Sub \setminus Expl(B_i, [\alpha]_E)\}$
        for each $\langle f', s'\rangle \in C_{B_i}([\alpha]_E, L(s) \cap Expl(B_i, [\alpha]_E))$ **do**
        $/* f' = \langle S', J', R', L'\rangle */$
          $ArgImpl(B_i, \alpha, Sub) := ArgImpl(B_i, \alpha, Sub) \cap L'(s')$
        end
        $L(s) := L(s) \cup B_i ArgImpl(B_i, \alpha, Sub)$
      end
    end
  endif
end CTLMC(f, Γ) } *Phase C2*

Figure 6 The model checking algorithm.

believes $B_i\phi$ in state n if and only if each state of B_A, in which ϵ believes A to be (i.e., states m and m' in Figure 5), satisfies ϕ. Given a state s' of a FSM f' of view α_{B_i}, we say that s' is *compatible* with a state s of a FSM of view α if the pair $\langle f', s'\rangle$ belongs to $C_{B_i}(\alpha, L(s) \cap Expl(B_i, \alpha))$.

The model checking algorithm MAMC(α, ϕ) takes two arguments, namely a view α and the MATL formula $\phi \in \mathcal{L}_\alpha$ that we want to model check. It calls the algorithm MAMC-View (shown in Figure 6) on view α and the set of formulae $\{\phi\}$. As a result, after this step, MAMC can return the appropriate truth value simply by testing whether ϕ is contained in the label set of the initial states J of all the FSMs $f \in F_{[\alpha]_E}$.

Notationally, let $sub(\phi)$ denote the set of subformulae of ϕ (remember that $B_i\phi$ is atomic and that, therefore, it is the only subformula of itself). The algorithm MAMC-View(α, Γ) takes two arguments: a view α, and a set of MATL formulae $\Gamma \subset \mathcal{L}_\alpha$. MAMC-View($\alpha, \Gamma$) performs the following phases:
Phase A. This phase, corresponding to the first line of the algorithm, collects in Sub all the subformulae of the formulae in Γ.

Phase B. MAMC-View(α, Γ) considers in turn the belief operators B_i. For each of them, $ArgImpl(_{B_i}, \alpha, \Gamma)$, the set of all the formulae ϕ which are arguments of the implicit belief atoms $B_i\phi$ occurring in Γ, is computed. MAMC-View(α, Γ) calls itself recursively on the view below (e.g., α_{B_i}) and on the set $ArgImpl(_{B_i}, \alpha, \Gamma)$. In this process, the algorithm recursively descends the tree structure which needs to be model checked. The leaves of this tree are the views for which there is no need to model check implicit belief atoms, as there are no more implicit belief atoms occuring in Γ.

Phase C. This phase is a loop over all the FSMs f of the equivalence class of current view α to extend the labeling functions of the visited FSMs. This loop iteratively performs the following two phases:

Phase C.1. In this phase, all the states of f of the (equivalence class of the) current view α, where the algorithm is, are labeled with the implicit belief atoms. This phase is executed only if there occur implicit belief atoms in the input formulae. The labeling of states of f is computed according to definition of satisfiability of implicit belief atoms in a MAFSM. Therefore, for each reachable state s of f, the set $L(s) \cap Expl(_{B_i}, \alpha)$ is computed. Then, the implicit belief atoms occurring in Γ are added to the labeling function of s. Therefore, for each implicit belief atom $B_i\phi$, $B_i\phi$ is added to $L(s)$ if ϕ is satisfied by all the pairs $\langle f', s' \rangle$ belonging to the compatibility relation $C_{B_i}([\alpha]_E, L(s) \cap Expl(_{B_i}, [\alpha]_E))$.

Phase C.2. This phase simply calls a standard CTL model algorithm on the FSM f of the current view. Indeed, at this point every state s in the current FSM f is labeled (by phase C.1) with all the atoms (i.e, propositional atoms, explicit and implicit belief atoms) occurring in the input formulae. Notice that in this phase we can employ any model checker (in our case NuSMV [5]) as a black box.

The following result states that MAMC-View actually solves the model checking problem for MATL.

Theorem 1 (Correctness of MAMC-View) *Let $f = \langle S, J, R, L \rangle \in F_{[\alpha]_E}$, with α any view, and s a state in S. $MF, \alpha, f, s \models \phi$ iff MAMC-View$(\alpha, \{\phi\})$ applied to MF constructs a new MF such that $\phi \in L(s)$, with s state of f.*

This result extends a similar result proved in the case of trees of views and reported in [3]. The termination of MAMC−View is guaranteed by the fact that input formulae always have beliefs of finite depth.

5. CONCLUSIONS

In this paper we have described a model-checking based verification procedure for security protocols employing a logic of belief. Our approach allows us to reuse the technology and tools developed in model checking. To model beliefs in security protocols, we have defined MATL – MultiAgent Tempo-

ral Logic – a logic where the temporal aspect and the belief aspect are kept separated. This semantics is the basis for introducing the notion of MultiAgent Finite State Machine (MAFSM) as an extension of the usual notion of Finite State Machine. Then, we have described a model checking algorithm (MAMC) which allows us to verify formulae containing belief (sub)formulae in a MAFSM.

A semantics for MATL was originally proposed in [3] for MultiAgent systems. The version proposed in this paper extends the work in [3] and makes it applicable to security protocols. [2] describes in some detail how a specific security protocol (the Andrew protocol) can be modeled and model checked using the logic defined in [3]. The analysis reported in [2] has some limitations due to the limited expressibility of the logic defined in [3].

References

[1] M. Abadi and M. Tuttle. A semantics for a logic of authentication. In *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 201–216, 1991.

[2] M. Benerecetti and F. Giunchiglia. Model checking security protocols using a logic of belief. In *Proceedings of the Sixth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000). March 27th - April 1st, 2000, Berlin, Germany*. To appear.

[3] M. Benerecetti, F. Giunchiglia, and L. Serafini. Model Checking Multiagent Systems. *Journal of Logic and Computation, Special Issue on Computational & Logical Aspects of Multi-Agent Systems*, 8(3):401–423, 1998.

[4] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[5] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. Nusmv: a new symbolic model verifier. In *Proceedings of the International Conference on Computer-Aided Verification (CAV'99) Trento, Italy. July 1999.*, 1999.

[6] E.A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publisher B.V., 1990.

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

[8] W. Marrero, E.M. Clarke, and S. Jha. Model checking for security protocols. In *DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997.

[9] K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publ., 1993.

[10] J.C. Mitchell, M Mitchell, and U. Stern. Automated Analysis of Cryptographic Protocols Using Murphi. In *IEEE Symp. Security and Privacy*, pages 141–153, 1997.