Model Checking Security Protocols Using a Logic of Belief

Massimo Benerecetti[1] and Fausto Giunchiglia[1,2]

[1] DISA - University of Trento,
Via Inama 5, 38050 Trento, Italy
[2] IRST - Istituto Trentino di Cultura,
38050 Povo, Trento, Italy
{bene,fausto}@cs.unitn.it

Abstract. In this paper we show how model checking can be used for the verification of security protocols using a logic of belief. We model principals as processes able to have beliefs. The idea underlying the approach is to treat separately the temporal evolution and the belief aspects of principals. Therefore, when we consider the temporal evolution, belief formulae are treated as atomic propositions; while the fact that principal A has beliefs about another principal B is modeled as the fact that A has access to a representation of B as a process. As a motivating example, we use the framework proposed to formalize the Andrew protocol.

1 Introduction

In this paper we show how model checking (see, e.g., [5,6]) can be used for the verification of security protocols using a logic of belief (see [3] for an example of the use of a belief logic in security applications). Our approach allows us to reuse with almost no variations all the technology and tools developed in model checking.

Model checking allows us to verify concurrent reactive finite state *processes*. 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.
– We deal with beliefs as follows. The fact that principal a_1 has beliefs about another principal a_2 is modeled as the fact that a_1 has access to a representation of a_2 as a process. Then, any time it needs to verify the truth value of some belief atom about a_2, e.g., $B_2\phi$, a_1 simply tests whether, e.g., ϕ holds in its (appropriate) representation of a_2. Beliefs are essentially used to control the "jumping" among processes. This operation is iterated in the obvious way in case of nested beliefs.

The paper is structured as follows. In Section 2 we describe a well known protocol, the Andrew protocol, as a motivating example. Section 3 presents the theoretical framework we employ (called MultiAgent Finite State Machines). The description is given incrementally over the standard model checking notions. In particular, we adopt CTL [5] as the propositional temporal logic used to state temporal specifications. Section 4 shows how the Andrew protocol can be formalized in our framework. Section 5 describes the model checking procedure we propose, while Section 6 illustrates how the algorithm described in Section 5 works in verifying a property of the Andrew protocol. Finally, some conclusions are drawn.

2 The Andrew Protocol

In this section we briefly recall a simple authentication protocol, known as the Andrew protocol, which has been proved to be vulnerable to various attacks (see, e.g., [3]). The protocol involves two principals, A and B, which share a secret key K_{ab}, and carry out a handshake to authenticate each other. The ultimate goal of the protocol is to exchange a new secret session key K'_{ab} between A and B. B is intended to be the key server, while A is the recipient.

We use standard notation, and present the version of the protocol proposed in [3]. N_i denotes a nonce (a fresh message) newly created by principal i for the current session; K_{ij} is a shared key between principals i and j; $\{M\}_{K_{ij}}$ denotes a message M encrypted with the key K_{ij}; M_1, M_2 is the message resulting from the concatenation of the two messages M_1 and M_2; while $i \rightarrow j : M$ denotes the fact that principal i sends the message M to principal j. The Andrew protocol can be formulated as follows:

$$\begin{array}{lll} 1 & A \rightarrow B & : & \{N_a\}_{K_{ab}} \\ 2 & B \rightarrow A & : & \{N_a, N_b\}_{K_{ab}} \\ 3 & A \rightarrow B & : & \{N_b\}_{K_{ab}} \\ 4 & B \rightarrow A & : & \{K'_{ab}, N'_b\}_{K_{ab}} \end{array}$$

Intuitively, the protocol works as follows: with message 1, A sends B the (fresh) nonce N_a encrypted with the key K_{ab}, which is supposed to be a good key. The goal of this message is to request authentication from B. With message 2, B sends back to A the nonce N_a concatenated with a newly created nonce N_b, both encrypted. At this point, since B must have decrypted message 1 to be able to generate message 2, A knows that it is talking with B. Then, in message 3, A sends back to B N_b encrypted. This allows B to conclude that it is actually talking to A (as it must have decrypted message 2 to obtain N_b and generate message 3). The two principals are now authenticated. Finally with message 4, B sends A the new session key K'_{ab} together with a new nonce N'_b encrypted with the shared key. The final message is the one subject to attacks. Indeed, in message 4 there is nothing that A can recognize as fresh. An intruder might send A an old message, possibly containing a compromised key.

The kind of analysis we're interested in is based on the idea, originally proposed in [3] (but see also [1]), of studying how messages sent and received during a protocol session by a trusted party may affect its beliefs about the other parties. In the present case, one might want to prove the following property: after message 4 has been received by A, A (respectively B) believes that the new key is a good key between A and B. Another property is that after message 4, A (respectively B) believes that B (respectively A) believes that the new key is a good key for communication between A and B. As pointed out in [3], the second property is stronger than the first, as it ensures that both principals believe that the protocol ended correctly and that they both possess a good session key. It turns out that neither of the properties above can be attained by principal A.

3 Multiagent Finite State Machines

Principals engaged in an authentication session can be modeled as finite state processes. We build the notion of principal (agent) 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. Therefore, let $B = \{\text{B}_1, ..., \text{B}_n\}$, where each index $1, ..., n \in I$ corresponds to a principal. The set B^* denotes the set of finite strings of elements of B, i.e., strings of the form $\text{B}_1, ..., \text{B}_n$ with $\text{B}_i \in B$. We call any $\alpha \in B^*$, a *view*. Each view in B^* corresponds to a possible nesting of belief operators. We also allow for the empty string, ϵ. Figure 1 depicts the general structure of the views. 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. To each nesting of belief operators we associate a view of the corresponding principal. Intuitively, in Figure 1, the beliefs of principal 1 correspond to the view B_1 and can be modeled by a process playing 1's role. The beliefs that 1 has about (the behavior of) principal 2 correspond to the view B_1B_2 and can be modeled by a process playing 2's role in (1's view of) the protocol. Things work in the same way for the beliefs of 2 and the beliefs that 2 can have about 1.

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, a well known propositional branching-time temporal logic widely used in formal verification [5]. 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 Multi-Agent 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 $\mathcal{L}_{\alpha \text{B}_i}$. In particular, $\mathcal{L}_\epsilon$ is used to speak about the whole protocol. The language $\mathcal{L}_{\text{B}_i}$ is the language adopted to represent i's beliefs. The language $\mathcal{L}_{\text{B}_i\text{B}_j}$ is used to specify i's beliefs about j's beliefs, and so on. For instance, the formula $\mathsf{AG}\,(p \supset B_i\neg q) \in \mathcal{L}_\epsilon$, (denoted by

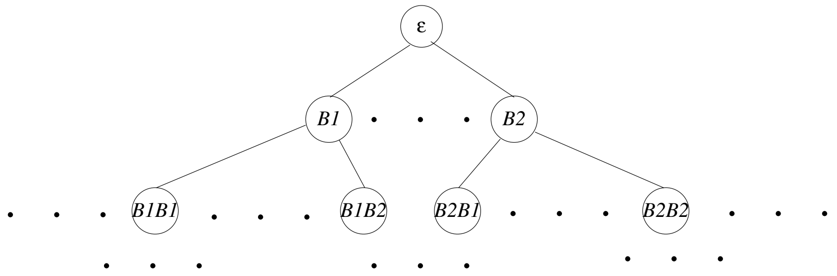

Fig. 1. A set of views

$\epsilon : \mathsf{AG}\,(p \supset B_i \neg q))$, intuitively means that in every future state, if p is true then principal i believes q is false. 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 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 tuple $f = \langle S, J, R, L\rangle$, where S is a finite set of states, $J \subseteq S$ is the set of *initial states*, the transition relation R is a total binary relation on S, and $L : S \rightarrow \mathcal{P}(P)$ is a *labelling function*, which associates to each state $s \in S$ the set $L(s)$ of propositional atoms true at s. Our solution is to extend the notion of FSM to that of *MultiAgent 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.

Formally, if $\mathcal{L}_\alpha$ is a MATL language of view α, then for each belief operator B_i, the set $Expl(B_i, \alpha)$ of *explicit belief atoms* of B_i for α is a (possibly empty) *finite* subset of the belief atoms of $\mathcal{L}_\alpha$. We have the following:

Definition 1. *Let $\{\mathcal{L}_\alpha\}$ be a family of MATL languages on $\{P_\alpha\}$. A* MultiAgent Finite State Machine *(MAFSM) $F = \{F_\alpha\}$ for $\{\mathcal{L}_\alpha\}$ is a recursive total function such that:*

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

where $B^* \setminus B^n$ denotes the difference between B^* and B^n, namely the set of all views not contained in B^n.

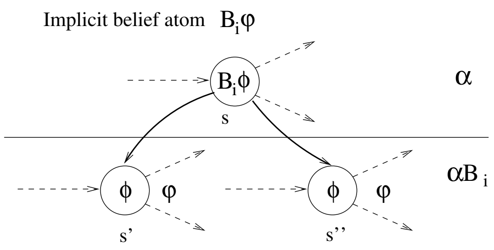

Fig. 2. Explicit belief atoms and satisfiability

The first condition 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 restricts us to a finite number of views. In general, there may be more than one FSM associated with 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.

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 with 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 $Impl(B_i, \alpha)$, the set of *implicit belief atoms* of a view α, be the (infinite) subset of all belief atoms of $\mathcal{L}_\alpha$ which are not explicit belief atoms, i.e., $Impl(B_i, \alpha) = \{B_i\phi \in \mathcal{L}_\alpha \setminus Expl(B_i, \alpha)\}$. The idea is to use the information explicitly contained in the labelling 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 2 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 αB_i, which model the beliefs of i, must be in any of the states (s' and s'' in Figure 2) in which the formulae ϕ, occurring as arguments of the explicit belief atoms, are true. This motivates the following definition. Let $ArgExpl(B_i, \alpha, s)$ be defined as follows:

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

$ArgExpl(B_i, \alpha, s)$ 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 labelling function of s). The set $ArgExpl(B_i, \alpha, s)$ contains the formulae which identify the states in which the FSMs in view αB_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 2, we just need to check whether all the *reachable states* [1] of the FSMs of view αB_i, which satisfy $ArgExpl(B_i, \alpha, s)$ (namely, the set $\{\phi\}$ in Figure 2), also satisfy the argument ψ of the implicit belief atom. If this is the case, then s satisfies $B_i\psi$.

Definition 2. (Satisfiability in a MAFSM) *Let F be a MAFSM, α a view in B^*, $f = \langle S, J, R, L\rangle \in F_\alpha$ an FSM, and $s \in S$ a state. Then, for any formula ϕ of $\mathcal{L}_\alpha$, the satisfiability relation $F, \alpha, f, s \models \phi$ is defined as follows:*

1. *$F, \alpha, f, s \models p$, 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. *$F, \alpha, f, s \models B_i\psi$, where $B_i\psi$ is an implicit belief atom, iff for all $f' \in F_{\alpha B_i}$ and s' reachable state of the FSM f', $F, \alpha B_i, f', s' \models \bigwedge ArgExpl(B_i, \alpha, s) \supset \psi$.*

We have furthermore:

4. *for every $s \in J$, $F, \alpha, f \models \phi$ iff $F, \alpha, f, s \models \phi$;*
5. *$F, \alpha \models \phi$ iff for all $f \in F_\alpha$, $F, \alpha, f \models \phi$;*
6. *$F \models \alpha : \phi$ iff $F, \alpha \models \phi$.*

In the definition of satisfiability above, Item 3 is the crucial step. The formula $\bigwedge ArgExpl(B_i, \alpha, s)$ is the conjunction of all the elements of $ArgExpl(B_i, \alpha, s)$.[2] Item 4 states that a FSM satisfies a formula if this formula is satisfied in all its initial states. Item 5 states that a formula is satisfied in a view if it is satisfied by all the FSMs of that view. Finally, Item 6 states that a labeled formula $\alpha : \phi$ is satisfied if ϕ is satisfied in the view corresponding to the label.

4 Modeling the Andrew Protocol Using MAFSMs

As described in Section 3, a MAFSM can be constructed out of the following elements: the structure of the views; the atomic propositions of each view and how they vary over time; the choice of explicit belief atoms of each view and how they vary over time; and the specification of the initial states for the FSMs in each view. In this section we present a MAFSM-based model of the Andrew protocol, thus providing, in turn, all these elements. For lack of space, we give only a partial description, emphasizing those elements of the model which are relevant to illustrate our approach.

In the MAFSM modeling the Andrew protocol there is only one FSM per view. Indeed, the processes associated to the all the views can be completely

[1] A state s of a FSM is said to be reachable if there is a path leading from an initial state of the FSM to state s.

[2] Item 3 gives to belief operators the same strength as modal $K(m)$, where m is the number of principals. In particular, we have that if $\Gamma \supset \phi$ is a theorem in a view then $B_i\Gamma \supset B_i\phi$ is a theorem in the (appropriate) view above, where $B_i\Gamma$ is the set $\{B_i\phi \mid \phi \in \Gamma\}$.

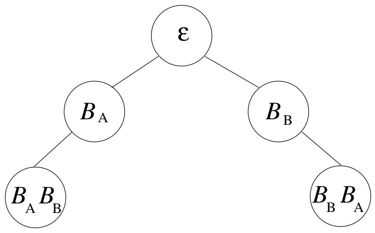

Fig. 3. The set of views of the Andrew protocol (Section 2)

specified, starting from the protocol specification given in Section 2. In the presentation below, we give the FSM specifications using the input language of the model checker NuSMV [4].

The structure of the views. The Andrew protocol involves two principals, A and B. Therefore, we have $I = \{A, B\}$. We model each principal as having beliefs about the other. Since, for the sake of the example, we do not need to model beliefs that a principal has about itself, we will need only to consider, besides the external view ϵ, the views of A and B, the view A has about B and the view B has about A. Therefore, $B^n = \{\epsilon, \mathrm{B}_A, \mathrm{B}_B, \mathrm{B}_A\mathrm{B}_B, \mathrm{B}_B\mathrm{B}_A\}$. Figure 3 shows the resulting situation. ϵ (the external observer) is modeled as a process which "sees" all the messages sent and received by the principals. B_A and $\mathrm{B}_B\mathrm{B}_A$ model the behavior of principal A, while views B_B and $\mathrm{B}_A\mathrm{B}_B$ model the behavior of principal B.

The set of atomic propositions P_α. In each view, we need to model a principal sending a message to another principal and/or receiving a message, as well as the properties the principal attributes to the (sub)messages it receives or sends. In particular, a (sub)message can be fresh and a key can be a good key for secret communication between the two principals. Each view has its own set of atomic propositions, reflecting the atomic properties of interest about its associated process. Since B_A and $\mathrm{B}_B\mathrm{B}_A$ model the same principal, we associate to both of them the same set of atomic propositions. Similarly for the views B_B and $\mathrm{B}_A\mathrm{B}_B$. For what concerns the views B_A and $\mathrm{B}_A\mathrm{B}_B$ in the Andrew protocol, we can consider the following set of atomic propositions:

$$P_{\mathrm{B}_A} = \left\{ \begin{array}{l} \mathit{send}_B\ N_a\text{-}K_{ab}, \\ \mathit{rec}\ N_a\text{-}N_b\text{-}K_{ab}, \\ \mathit{send}_B\ N_b\text{-}K_{ab}, \\ \mathit{rec}\ K'_{ab}\text{-}N'_b\text{-}K_{ab}, \\ \mathit{fresh}\ N_a, \\ \mathit{fresh}\ K'_{ab}\text{-}N'_b\text{-}K_{ab}, \\ \mathit{shk}\ K'_{ab} \\ \ldots \end{array} \right\} \qquad P_{\mathrm{B}_A\mathrm{B}_B} = \left\{ \begin{array}{l} \mathit{rec}\ N_a\text{-}K_{ab}, \\ \mathit{send}_A\ N_a\text{-}N_b\text{-}K_{ab}, \\ \mathit{send}_A\ K'_{ab}\text{-}N'_b\text{-}K_{ab}, \\ \mathit{fresh}\ N_b, \\ \mathit{fresh}\ K'_{ab}\text{-}N'_b\text{-}K_{ab}, \\ \mathit{shk}\ K'_{ab} \\ \ldots \end{array} \right\}$$

where the variables of the form $\mathit{rec}\, M$ or $\mathit{send}_B\, M$ (where M is a message of the Andrew protocol) are called *message variables* and represent the act of receiving M, and sending M to B, respectively. For instance, $rec\, N_a\_N_b\_K_{ab}$ and $send_B\, N_a\_K_{ab}$ in view $_{B_A}$ represent A receiving $\{N_a, N_b\}_{K_{ab}}$ (message 2 in the Andrew protocol), and A sending $\{N_a\}_{K_{ab}}$ to B (message 3 in the Andrew protocol), respectively. Variables of the form *fresh* M or *shk* M are called *freshness variables*, and express freshness properties of (sub)messages. For instance, *fresh* N_a in view $_{B_A}$ means that N_a is a fresh (sub)message, while *shk* K'_{ab} expresses the fact that K'_{ab} is a good shared key between A and B. For what concerns view ϵ, we set:

$$P_\epsilon = \left\{ \begin{array}{l} send_{B,A}\, K'_{ab}\_N'_b\_K_{ab}, \\ rec_A\, K'_{ab}\_N'_b\_K_{ab}, \\ \ldots \end{array} \right\}$$

where the variables for the sent messages (e.g., $send_{B,A}\, K'_{ab}\_N'_b\_K_{ab}$) are labeled (in subscript) with both the sender (B) and the intended receiver (A), while those of received messages (e.g., $rec_A\, K'_{ab}\_N'_b\_K_{ab}$) are labeled only with the actual receiver (A). With respect to the other views, the additional subscripts for both *send* and *rec* reflect the fact that ϵ knows who sends a message to whom and who receives what message.

Evolution of message variables. To specify the evolution of variables in a view we use the *next()* operator of the NuSMV input language. The *next* operator allows us to specify the next value of a variable in each state, possibly depending on its value in the current state. Since all variables are of type boolean, the possible values are T (for true) and F (for false). We report below the definitions of the next state value for some message variables in the language of view $_{B_A}$, modeling the behavior of (the beliefs of) A. [3]

```
B_A
1  next(send_B N_a_K_ab) := case
                              !send_B N_a_K_ab : {T,F};
                              1 : send_B N_a_K_ab;
                            esac;
2  next(rec K'_ab_N'_b_K_ab) := case
                              send_B N_b_K_ab & !rec K'_ab_N'_b_K_ab: {T,F};
                              1 : rec K'_ab_N'_b_K_ab;
                            esac;
```

Statement 1 contains a case statement, whose first clause ($!send_B\, N_a\_K_{ab}$: $\{T, F\}$) contains a precondition on the left-hand side and the next value on the right-hand side. The precondition is the negation of a message variable and is true when $send_B\, N_a\_K_{ab}$ is false, that is if message $N_a\_K_{ab}$ has not been sent to B yet . The next value is a set of values ($\{T, F\}$). This intuitively means that the first message of the Andrew protocol ($\{N_a\}_{K_{ab}}$) may or may not be sent (i.e., $send_B\, N_a\_K_{ab}$ may nondeterministically take value T or F) in the next state, if it has not been sent yet in the current state. The second item is the "default" clause, and it is taken if the precondition in the first clause does

[3] The label $_{B_A}$ at the top of a block of statements means that the block belongs to the specification of view $_{B_A}$, and similarly for the other views.

not apply. The result of this clause is that $\mathit{send}_B\, N_a\_K_{ab}$ keeps, in the next state, its current value. In Statement 2, the precondition of the first clause (i.e., $\mathit{send}_B\, N_b\_K_{ab}$ & $!\mathit{rec}\, K'_{ab}\_N'_b\_K_{ab}$) is a conjunction of two boolean expressions. The first expression means that $\{N_b\}_{K_{ab}}$ has been already sent to B, and the second means that $\{K'_{ab}, N'_b\}_{K_{ab}}$ has not been received yet. The next value again is the nondeterministic choice between values T and F (the message is received or not). The messages in each session of the Andrew protocol are supposed to be sent and received following the order reported in Section 2. Therefore, for each message variable, the preconditions in the next statement checks whether the previous message (and therefore all the previous messages) involving the current principal (A in the case of view B_A), has been already conveyed or not. The default clause is similar to that of Statement 1. The specification of the evolution of the other message variables in the other views is specified in a similar way.

Evolution of freshness variables. In any path of states of a view, freshness variables for (sub)messages originated by that principal always keep their initial values. In the Andrew protocol, this is the case for N_a in view B_A (and also $B_B B_A$), as expressed by the *next* statements below, and N_b in views B_B and $B_A B_B$.

```
B_A
3  next(fresh N_a) := fresh N_a;
```

Statement 3 simply says that *fresh* N_a keeps the current value in the next state. Similar statements are made also for *fresh* N_b, *fresh* N'_b and *shk* K'_{ab} (which are messages originated by B) in the views modeling B.

On the other hand, a principal can attain freshness of the messages it has not originated itself, only after it receives (possibly other) messages which contain them. Therefore, freshness variables of a message M not originated by a principal may change their value (e.g., from false to true) only when the principal has received a message containing M. After the last message of the session has been conveyed, the freshness variables of M keep their current value (no new information can be gained by the principal). Moreover, once it becomes true, a freshness variable remains stable. All of the above intuitions are specified as follows:

```
B_A
4  next(shk K'_ab) := case
                          !shk K'_ab & !rec K'_ab_N'_b_K_ab : {T,F};
                          1 : shk K'_ab ;
                        esac;
5  next(fresh K'_ab_N'_b_K_ab) := case
                          !fresh K'_ab_N'_b_K_ab & !rec K'_ab_N'_b_K_ab : {T,F};
                          1 : fresh K'_ab_N'_b_K_ab;
                        esac;
```

Statement 4 and 5 are very similar in form. As to statement 4, the precondition in the first clause of the case statement is a conjunction ($!\mathit{shk}\, K'_{ab}$ & $!\mathit{rec}\, K'_{ab}\_N'_b\_K_{ab}$) of two negations (meaning respectively that "K'_{ab} is not known to be a good shared key", and that the "$\{K'_{ab}, N'_b\}_{K_{ab}}$ has not been received yet"). If this condition is true, the nondeterministic choice on the left-hand side of the clause is taken. The "default" clause leaves the value of the variable in the next state unchanged. Statement 5 checks, in the first clause of the case statement, if the conjunction in the precondition (!*fresh* K'_{ab}_N'_b_K_{ab} & !*rec* $K'_a b$_N'_b_K_{ab}) is true ($\{K'_{ab}, N'_b\}_{K_{ab}}$ is not known to be fresh, and it has not been received yet), and in this case chooses nondeterministically the next value of the variable.

Clearly Statements 4 and 5 are very general and do not allow us to model appropriately the freshness of (sub)messages. Indeed, additional constraints are needed. Following the BAN logic approach, there are a number of properties of messages, which relate their freshness to that of their components. For instance, a principal can conclude that a message is fresh from the freshness of one of its components. This is the case for $\{N_a\}_{K_{ab}}$, which is known to be fresh whenever N_a is known to be.[4] NuSMV allows for specifying constraints on the admissible (reachable) states by means of invariants, which are boolean formulae that must hold in every reachable state. The following invariant statement captures a relevant constraint on some freshness variables:

^{B_A}
6 *INVAR (fresh* K'_{ab} | *fresh* N'_b*) & rec* K'_{ab}_N'_b_K_{ab} $\leftrightarrow$ *fresh* K'_{ab}_N'_b_K_{ab}

Invariant 6 is an equivalence ($\leftrightarrow$) whose left-hand side is a conjunction. The disjunction in the left conjunct (*fresh* K'_{ab} | *fresh* N'_b) means that K'_{ab} or N'_b is fresh; the second conjunct is meant to be true when the message $\{K'_{ab}, N'_b\}_{K_{ab}}$ has been received. Intuitively, it states that A can consider (the encrypted message) $\{K'_{ab}, N'_b\}_{K_{ab}}$ fresh (right-hand side of the equivalence) if and only if it has received the message (second conjunct on the left-hand side) and either of its components is fresh (first conjunct on the left-hand side). Similar invariants must be added for each message received by the principal.

View $B_A B_B$ can be specified in a similar way. Some additional statements must be added, though, reflecting the role of principal B. Indeed, the Andrew protocol assumes that B is the key server (see Section 2). Therefore, B is supposed to generate and send a good shared key in message 4, whenever it believes that $\{K'_{ab}, N'_b\}_{K_{ab}}$ is fresh. Remember that we have a variable for the freshness of the last message of the protocol (namely, *fresh* K'_{ab}_N'_b_K_{ab}), and a variable for K'_{ab} being a good key (*shk* K'_{ab}). Then we can specify the above invariant as an implication:

$^{B_A B_B}$
7 *INVAR fresh* K'_{ab}_N'_b_K_{ab} $\rightarrow$ *shk* K'_{ab}

[4] For (sub)messages which are not originated by the principal, usually BAN-like logics substitute the notion of freshness with that of recentness. A message is then considered recently conveyed by another principal, if it is part of a fresh message encrypted with a secret key, shared with that principal. The difference between these two concepts is not relevant for the present paper.

Explicit belief atoms of a view. We need now 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 authentication protocols, principals can only gain information carried by the messages they receive. In BAN-like logics the freshness of the received messages is based on their form, and it is a basic property to be attained by a principal. We choose, therefore, the following belief atoms as explicit beliefs atoms: the beliefs about other principals having sent or received a given message, and the beliefs about the freshness of messages. In our case, we have:

$$Expl(B_A,\epsilon)=\left\{\begin{array}{l}B_A rec\ K'_{ab}\_N'_b\_K_{ab},\\ B_A fresh\ N_a\\ \ldots\end{array}\right\}\quad Expl(B_B,B_A)=\left\{\begin{array}{l}B_B send_A\ K'_{ab}\_N'_b\_K_{ab},\\ B_B fresh\ K'_{ab}\_N'_b\_K_{ab}\\ \ldots\end{array}\right\}$$

where, for instance, $B_A fresh\ N_a$ in ϵ intuitively means that A believes that N_a is a fresh nonce; while $B_B send_A\ K'_{ab}\_N'_b\_K_{ab}$ in B_A expresses the fact that A believes that B believes that it has sent $\{K'_{ab}, N'_b\}_{K_{ab}}$ to A. There are no explicit belief atoms in $B_B B_A$ and $B_A B_B$, as they have no beliefs atoms at all in their language. Indeed, the example we are considering does not need to model more than two nesting of the belief operators.

Evolution of explicit belief atoms. Variables representing explicit belief atoms are supposed to vary similarly to freshness variables. In particular, as long as there are still messages to be received by a principal of a view, explicit belief atoms of that view are free to change value from F to T. Once the last message of the protocol has been received, no new belief can be attained, and explicit belief atoms keep their value, thereafter. Again, once they become true, they remain stable. The following statement specifies how the value of $B_B send_A\ N_a\_N_b\_K_{ab}$ may vary in B_A:

```
B_A
8  next(B_B send_A N_a_N_b_K_ab) := case
                                   !B_B send_A N_a_N_b_K_ab & !rec K'_ab_N'_b_K_ab : {T,F};
                                    1 : B_B send_A N_a_N_b_K_ab;
                                   esac;
```

Very similar statements can be added for all the explicit beliefs in all views. Similarly to freshness variables, additional constraints, in the form of state invariants, on reachable states need to be added for explicit beliefs atoms. In particular we report below some relevant ones for the Andrew protocol. All these invariants can be seen as encodings of standard properties holding in BAN-like logics.

```
B_A
9  INVAR rec K'_ab_N'_b_K_ab → B_B send_A K'_ab_N'_b_K_ab
10 INVAR fresh K'_ab_N'_b_K_ab → B_B fresh K'_ab_N'_b_K_ab
```

Both invariants are implications. Intuitively, Invariant 9 states that if it receives the message $\{K'_{ab}, N'_b\}_{K_{ab}}$ (left-had side of the implication) then A ascribes to B the belief that B has sent that message to A (right-hand side of the implication).

Invariant 10 states that if it can conclude that $\{K'_{ab}, N'_b\}_{K_{ab}}$ is fresh, then A also ascribes to B the same belief.

Initial states of a view. Finally, we have to set the initial states of the FSM in each view. They can be specified as a boolean formula which identifies all and only the admissible initial states of the FSM. Following the BAN logic tradition, we want to model a single session of the Andrew protocol and study what beliefs each principal can attain at the end of the session. A protocol session starts with no message sent or received. Thus, the process in each view starts with the value of all the message variables set to false. Since no message has been received yet, freshness variables of messages not originated by a principal are set to false as well. All the other variables can take nondeterministically the value T or F in the initial states. The following specification in view B_A formalises these intuitions for the Andrew protocol:

```
$B_A$
INIT !send$_B$ N$_a$_K$_{ab}$ &
    !rec N$_a$_N$_b$_K$_{ab}$ &
    !send$_B$ N$_b$_K$_{ab}$ &
    !rec K$'_{ab}$_N$'_b$_K$_{ab}$ &
    !fresh K$'_{ab}$_N'$_b$_K$_{ab}$ &
    !shk K$'_{ab}$
    ...
```

All the other views can be specified in a similar way.

5 Model Checking a MAFSM

The basic operation of a standard CTL model checking algorithm is to extend the labelling function of an FSM (which considers only propositional atoms) to all the sub-formulae 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 labelling function explicitly defines the truth value of all atoms. The problem is that in the FSMs of a MAFSM the labelling function is not defined on implicit belief atoms, whose truth value is therefore left undefined; and 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 2) tells us how to solve this problem.

The crucial observation is that $ArgExpl(B_i, \alpha, s)$, in Item 3 of Definition 2, is generated from the formulae in $Expl(B_i, \alpha)$ and the labelling functions of the FSMs in α; $ArgExpl(B_i, \alpha, s)$ is a finite set; and it only depends on the MAFSM specified (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 α. Then:

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

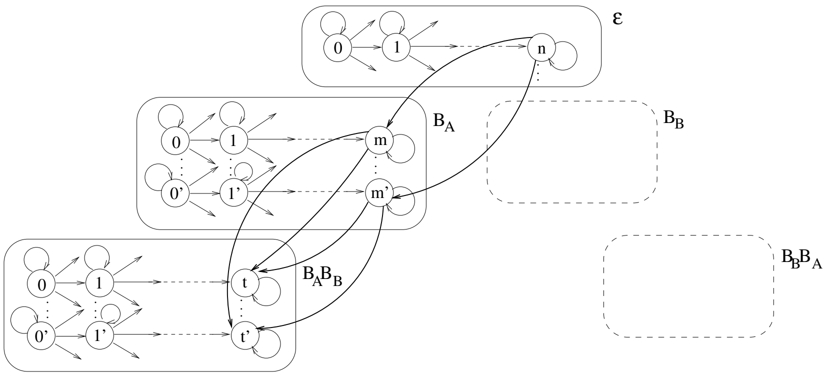

Fig. 4. Some FSMs for the Andrew Protocol

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 αB_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 αB_i, in which the process associated to view αB_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 αB_i, s' any reachable state of f', and s a state of a FSM of view α:

$$\langle f', s'\rangle \in C_{B_i}(\alpha, L(s) \cap Expl(B_i, \alpha)) \text{ iff } F, \alpha B_i, f', s' \models \bigwedge ArgExpl(B_i, \alpha, s)$$

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

$$F, \alpha, f, s \models B_i\phi \text{ iff for all } \langle f', s'\rangle \in C_{B_i}(\alpha, L(s) \cap Expl(B_i, \alpha)),\ F, \alpha B_i, f', s' \models \phi \tag{1}$$

where $B_i\phi$ is any implicit belief atom.

Figure 4 represents the underlying intuition on the MAFSM for the Andrew protocol. The relation $C_{B_i}(\alpha, L(s) \cap Expl(B_i, \alpha))$ is depicted as arrows connecting states of adjacent 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 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 4), satisfies ϕ. Given a state s' of a FSM f' of view αB_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−View(α, Γ) (see [2] for a complete description and a proof of correctness) 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. MAMC−View(α, Γ) considers in turn the belief operators B_i. For each of them, the set $ArgImpl(\mathrm{B}_i, \alpha, \Gamma)$, the set of all the formulae ϕ which are arguments of the implicit belief atoms $B_i\phi$ occurring in Γ, is computed.

Phase B. MAMC−View(α, Γ) calls itself recursively on the view below (e.g., $\alpha\mathrm{B}_i$) and on the set $ArgImpl(\mathrm{B}_i, \alpha, \Gamma)$. 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 current view α to extend the labelling functions of the visited FSMs. This loop iteratively performs the following two phases:
Phase C.1. In this phase, all the states of the FSM f 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 labelling 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 $L(s) \cap Expl(\mathrm{B}_i, \alpha)$ is computed. Then, the algorithm computes the implicit belief atoms occurring in Γ which can be added to the labelling function of s. That is, 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 $C_{B_i}(\alpha, L(s) \cap Expl(\mathrm{B}_i, \alpha))$.

Phase C.2. This phase simply calls a standard CTL model checking 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 phase C.2 we can employ any model checker (in our case NuSMV) as a black box.

6 Model Checking the Andrew Protocol

In this section, we show how the model checking algorithm described in Section 5 works, trying to check a desired property of the Andrew protocol of Section 2. Let us consider the following property (from Section 2): A believes that B believes that K'_{ab} is a good shared key, any time it receives the last message of the Andrew protocol. This property can be written as the following MATL formula in view ϵ:

$$\epsilon : \mathsf{AG}\,(rec_A\, K'_{ab}\_N'_b\_K_{ab} \wedge B_A\, fresh\, N_a \rightarrow B_A\ B_B\ shk\, K'_{ab}) \tag{2}$$

where the (sub)formula $B_A\ B_B\ shk\, K'_{ab}$ is an implicit belief atom.
Phase A and B. We only have to consider the belief operator B_A. We construct the set $ArgImpl(\mathrm{B}_B, \epsilon, \Gamma) = \{B_B\, shk\, K'_{ab}\}$. Since $B_B\, shk\, K'_{ab}$ is an implicit belief atom of ϵ, MAMC−View$(\epsilon, \{\mathsf{AG}\,(rec_A\, K'_{ab}_N'_b_K_{ab} \wedge B_A\, fresh\, N_a \rightarrow B_A B_B\, shk\, K'_{ab})\})$ calls itself recursively on view B_A and $\{B_B\, shk\, K'_{ab}\}$ (calling MAMC−View$(\mathrm{B}_A, \{B_B\, shk\, K'_{ab}\})$). In view B_A, we need to consider the operator B_B and to compute $ArgImpl(\mathrm{B}_B, \mathrm{B}_A, \{B_B\, shk\, K'_{ab}\}) = \{shk\, K'_{ab}\}$.

MAMC−View($_{B_A}$, $\{B_B\, \mathit{shk}\, K'_{ab}\}$) descends to view $_{B_A B_B}$ (calling MAMC−View($_{B_A B_B}$, $\{\mathit{shk}\, K'_{ab}\}$)). Phase B is not performed in view $_{B_A B_B}$ as no implicit beliefs occur in the input formulae (namely, $\{\mathit{shk}\, K'_{ab}\}$).

Phases C.1 and C.2 at $_{B_A B_B}$. The only formula to check in this view is the atomic formula $\mathit{shk}\, K'_{ab}$. The FSM of this view already contains the information about its truth value in each state (see Section 4). Therefore, both phases end immediately.

Phase C.1 at $_{B_A}$. The FSM f of this view is labeled with the implicit belief atom $B_B\, \mathit{shk}\, K'_{ab}$. For each reachable state s of f and each pair $\langle f', s' \rangle \in C_{B_B}(_{B_A}, L(s) \cap \mathit{Expl}(_{B_B}, _{B_A}))$, the intersection of $\mathit{ArgImpl}(_{B_B}, _{B_A}, \{B_B\, \mathit{shk}\, K'_{ab}\}) = \{\mathit{shk}\, K'_{ab}\}$ and $L(s')$ is computed. This gives either the empty set (meaning that $\mathit{shk}\, K'_{ab}$ is not true in s') or $\{\mathit{shk}\, K'_{ab}\}$ itself (meaning that $\mathit{shk}\, K'_{ab}$ is true in s'). The final step consists in adding to $L(s)$ the implicit belief atom $B_B\, \mathit{shk}\, K'_{ab}$, if every state of f', compatible with s, satisfies $\mathit{shk}\, K'_{ab}$. It turns out that the states of $_{B_A}$ satisfying that implicit belief are those which satisfy $\mathit{rec}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$ (A has received message 4) and $\mathit{fresh}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$ (message 4 is known by A to be fresh). All those states also satisfy the explicit belief atom $B_B\, \mathit{fresh}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$ (by Invariant 10), and are therefore compatible only with those states of view $_{B_A B_B}$ where $\mathit{fresh}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$ is true. Notice that there can be a reachable state of $_{B_A}$ where $\mathit{rec}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$ and $\mathit{fresh}\, N_a$ are both true but $\mathit{fresh}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$ is false. This is possible, as nothing in message 4 is recognisable by A as fresh. As a consequence, there is a reachable state of view $_{B_A}$, which does not satisfy $B_B\, \mathit{fresh}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$ either. Let the state m of view $_{B_A}$ in Figure 4 be one such state. Therefore, state m satisfies $\mathit{rec}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$ and $\mathit{fresh}\, N_a$ but not $B_B\, \mathit{shk}\, K'_{ab}$.

Phase C.2 at view $_{B_A}$. Once again, the formula to check is atomic (though an implicit belief atom). Therefore, this phase ends immediately.

Phase C.1 at ϵ. We have now to process the implicit belief $B_A\, B_B\, \mathit{shk}\, K'_{ab}$. It performs similar steps as in phase C.1 for view $_{B_A}$. It turns out that there is (at least) a reachable state in the FSM of ϵ which does not satisfy this implicit belief. Indeed, as we have pointed out above, there is a state of view $_{B_A}$ (state m in Figure 4) which does not satisfy $B_B\, \mathit{shk}\, K'_{ab}$ but satisfies both $\mathit{fresh}\, N_a$ and $\mathit{rec}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$. Let us assume that state n in ϵ (see Figure 4) satisfies $B_A\, \mathit{fresh}\, N_a$ and $\mathit{rec}_A\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$ but does not satisfy the explicit belief atom $B_A\, \mathit{fresh}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$. n is actually a reachable state of the FSM of ϵ. Since n satisfies $\mathit{rec}_A\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$, by an invariant of ϵ similar to Invariant 9, n also satisfies $B_A\, \mathit{rec}_A\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$. Since n does not satisfy the belief atom $B_A\, \mathit{fresh}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$, the explicit belief atoms true at n are not enough to rule out the compatibility with state m in $_{B_A}$ (see again Figure 4). Indeed, m does not satisfy $\mathit{fresh}\, K'_{ab}\text{-}N'_b\text{-}K_{ab}$, therefore, it belongs to the compatibility relation $C_{B_A}(\epsilon, L(n) \cap \mathit{Expl}(_{B_A}, \epsilon))$. State m, as we know, does not satisfy $B_B\, \mathit{shk}\, K'_{ab}$. As a consequence of (1), state n of ϵ does not satisfy $B_A\, B_B\, \mathit{shk}\, K'_{ab}$.

Phase C.2 at ϵ. Here the usual CTL model checking algorithm on the formula (2) is called. As expected, the final answer is negative since, from phase C.1 in ϵ, there are reachable states satisfying $rec_A\, K'_{ab}\_N'_b\_K_{ab} \wedge B_A\, fresh\, N_a$ but not $B_A\, B_B\, shk\, K'_{ab}$.

7 Conclusion

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 the notion of Multiagent Finite State Machine as an extension of the usual notion of Finite State Machine. Then, we have described a model checking algorithm which allows us to verify formulae containing belief (sub)formulae in a MAFSM. Finally, we have formalized within this framework a well known security protocol, the Andrew protocol.

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. 521
2. 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. Also IRST-Technical Report 9708-07, IRST, Trento, Italy. 531
3. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990. 519, 520, 521
4. 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*. 525
5. E. Clarke, O. Grumberg, and D. Long. Model Checking. In *Proceedings of the International Summer School on Deductive Program Design*, Marktoberdorf, Germany, 1994. 519, 520, 521
6. K.L. McMillan. Symbolic Model Checking. Kluwer Academic, 1993. 519