State Space Reduction in the Maude-NRL Protocol Analyzer

Santiago Escobar\textsuperscript{a}, Catherine Meadows\textsuperscript{b}, José Meseguer\textsuperscript{c}

\textsuperscript{a} DSIC-ELP, Universidad Politécnica de Valencia, Valencia, Spain
\textsuperscript{b} Naval Research Laboratory, Washington, DC, USA
\textsuperscript{c} University of Illinois at Urbana-Champaign, Urbana, IL, USA

Abstract

The Maude-NRL Protocol Analyzer (Maude-NPA) is a tool and inference system for reasoning about the security of cryptographic protocols in which the cryptosystems satisfy different equational properties. It both extends and provides a formal framework for the original NRL Protocol Analyzer, which supported equational reasoning in a more limited way. Maude-NPA supports a wide variety of algebraic properties that includes many crypto-systems of interest such as, for example, one-time pads and Diffie-Hellman. Maude-NPA, like the original NPA, looks for attacks by searching backwards from an insecure attack state, and assumes an unbounded number of sessions. Because of the unbounded number of sessions and the support for different equational theories, it is necessary to develop ways of reducing the search space and avoiding infinite search paths. In order for the techniques to prove useful, they need not only to speed up the search, but should not violate completeness, so that failure to find attacks still guarantees security. In this paper we describe some state space reduction techniques that we have implemented in Maude-NPA. We also provide completeness proofs, and experimental evaluations of their effect on the performance of Maude-NPA.

1. Introduction

The Maude-NPA \cite{4,7} is a tool and inference system for reasoning about the security of cryptographic protocols in which the cryptosystems satisfy different equational properties. The tool handles searches in the unbounded session model, and thus can be used to provide proofs of security as well as to search for attacks. It is the next generation of the NRL Protocol Analyzer \cite{15}, a tool that supported limited equational reasoning and was successfully applied to the analysis of many different protocols. In Maude-NPA we improve on the original NPA in three ways. First of all, unlike NPA, which required considerable interaction with the user, Maude-NPA is completely automated (see \cite{7}). Secondly, its inference system has a formal basis in terms of rewriting logic and narrowing, which allows us to provide proofs of soundness and completeness (see \cite{4}). Finally,
the tool’s inference system supports reasoning modulo the algebraic properties of cryptographic and other functions (see \[5, 3, 13\]). Such algebraic properties are expressed as equational theories $E = E' \cup Ax$ whose equations $E'$ are confluent, coherent, and terminating rewrite rules modulo equational axioms $Ax$ such as commutativity ($C$), associativity-commutativity ($AC$), or associativity-commutativity plus identity ($ACU$) of some function symbols. The Maude-NPA has then both dedicated and generic methods for solving unification problems in such theories $E = E' \cup Ax$ \[10, 11, 12\], which under appropriate checkable conditions \[9\] yield finitary unification algorithms.

Since Maude-NPA allows reasoning in the unbounded session model, and because it allows reasoning about different equational theories (which typically generate many more solutions to unification problems than syntactic unification, leading to bigger state spaces), it is necessary to find ways of pruning the search space in order to prevent infinite or overwhelmingly large search spaces. One technique for preventing infinite searches is the generation of formal grammars describing terms unreachable by the intruder (see \[15, 4\] and Section \[4.1\]). However, grammars do not prune out all infinite searches, since unbounded session security is undecidable, and there is a need for other techniques. Moreover, even when a search space is finite it may still be necessary to reduce it to a manageable size, and state space reduction techniques for doing that will be necessary. In this paper we describe some of the major state space reduction techniques that we have implemented in Maude-NPA, and provide completeness proofs and experimental evaluations demonstrating an average state-space size reduction of 99% (i.e., the average size of the reduced state space is 1% of that of the original one) in the examples we have evaluated. Furthermore, we show our combined techniques effective in obtaining a finite state space for all protocols in our experiments.

The optimizations we describe in this paper were designed specifically for Maude-NPA, and work within the context of Maude-NPA search techniques. However, although different tools use different models and search algorithms, they all have a commonality in their syntax and semantics that means that, with some adaptations, optimization techniques developed for one tool or type of tools can be applied to different tools as well. Indeed, we have already seen such common techniques arise, for example the technique of giving priority to input or output messages respectively when backwards or forwards search is used (used by us and by Shmatikov and Stern in \[20\]) and the use of the lazy intruder (used by us and, in a different form, by the On-the-Fly Model Checker \[1\]). One of our motivations of publishing our work on optimizations is to encourage the further interaction and adaptation of the techniques for use in different tools.

The rest of the paper is organized as follows. After some preliminaries in Section \[2\], we describe in Section \[3\] the model of computation used by the Maude-NPA. In Section \[4\] we describe the various state space reduction techniques that have been introduced to control state explosion, and give proofs of their completeness as well as showing their relations to other optimization techniques in the literature. We first briefly describe how automatically generated grammars provide the main reduction that cuts down the search space. Then, we describe
how we obtain a second important state-space reduction by reducing the number of logical variables present in a state. The additional state space reduction techniques presented in this paper are: (i) giving priority to input messages in strands, (ii) early detection of inconsistent states (that will never reach an initial state), (iii) a relation of transition subsumption (to discard transitions and states already being processed in another part of the search space), and (iv) the super-lazy intruder (to delay the generation of substitution instances as much as possible). In Section 5 we describe our experimental evaluation of these state-space reduction techniques. In Section 6 we describe future work and conclude the paper. This is an extended and improved version of [6], including proofs of all the results, a refinement of the interaction between the transition subsumption and the super-lazy intruder (Section 4.7.2), more examples and explanations, as well as more benchmarked protocols.

2. Background on Term Rewriting

We follow the classical notation and terminology from [20] for term rewriting and from [16] [17] for rewriting logic and order-sorted notions. We assume an order-sorted signature $\Sigma$ with a finite poset of sorts $(S, \leq)$ and a finite number of function symbols. We assume an $S$-sorted family $\mathcal{X} = \{\mathcal{X}_s\}_{s \in S}$ of mutually disjoint variable sets with each $\mathcal{X}_s$ countably infinite. $T_\Sigma(\mathcal{X})_s$ denotes the set of terms of sort $s$, and $T_{\Sigma,s}$ the set of ground terms of sort $s$. We write $T_\Sigma(\mathcal{X})$ and $T_\Sigma$ for the corresponding term algebras. We write $\text{Var}(t)$ for the set of variables present in a term $t$. The set of positions of a term $t$ is written $\text{Pos}(t)$, and the set of non-variable positions $\text{Pos}_\Sigma(t)$. The subterm of $t$ at position $p$ is $t[p]$, and $t[u]_p$ is the result of replacing $t[p]$ by $u$ in $t$. A substitution $\sigma$ is a sort-preserving mapping from a finite subset of $\mathcal{X}$, written $\text{Dom}(\sigma)$, to $T_\Sigma(\mathcal{X})$. The set of variables introduced by $\sigma$ is $\text{Ran}(\sigma)$. The identity substitution is $\text{id}$. Substitutions are homomorphically extended to $T_\Sigma(\mathcal{X})$. The restriction of $\sigma$ to a set of variables $V$ is $\sigma|_V$. The composition of two substitutions is $(\sigma \circ \theta)(X) = \theta(\sigma(X))$ for $X \in \mathcal{X}$.

A $\Sigma$-equation is an unoriented pair $t = t'$, where $t \in T_\Sigma(\mathcal{X})_s$, $t' \in T_\Sigma(\mathcal{X})_{s'}$, and $s$ and $s'$ are sorts in the same connected component of the poset $(S, \leq)$. Given a set $E$ of $\Sigma$-equations, order-sorted equational logic induces a congruence relation $=_E$ on terms $t, t' \in T_\Sigma(\mathcal{X})$ (see [17]). Throughout this paper we assume that $T_{\Sigma,S} \neq \emptyset$ for every sort $s$. We denote the $E$-equivalence class of a term $t \in T_\Sigma(\mathcal{X})$ as $[t]_E$ and the $E$-equivalence classes of all terms $T_\Sigma(\mathcal{X})$ and $T_{\Sigma,s}$ as $T_{\Sigma,E}(\mathcal{X})$ and $T_{\Sigma,s,E}(\mathcal{X})$, respectively.

For a set $E$ of $\Sigma$-equations, an $E$-unifier for a $\Sigma$-equation $t = t'$ is a substitution $\sigma$ s.t. $\sigma(t) = _E \sigma(t')$. A complete set of $E$-unifiers of an equation $t = t'$ is written $\text{CSU}_E(t = t')$. We say $\text{CSU}_E(t = t')$ is finitary if it contains a finite number of $E$-unifiers. $\text{CSU}(t = t')$ denotes a complete set of syntactic order-sorted unifiers between terms $t$ and $t'$, i.e., without any equational property.

A rewrite rule is an oriented pair $l \rightarrow r$, where $l \not\in \mathcal{X}$ and $l,r \in T_\Sigma(\mathcal{X})_s$ for some sort $s \in S$. An (unconditional) order-sorted rewrite theory is a triple
\( \mathcal{R} = (\Sigma, E, R) \) with \( \Sigma \) an order-sorted signature, \( E \) a set of \( \Sigma \)-equations, and \( R \) a set of rewrite rules. A topmost rewrite theory \( (\Sigma, E, R) \) is a rewrite theory s.t. for each \( l \to r \in R \), \( l, r \in T_\Sigma(\mathcal{X})_{\text{State}} \) for a top sort \( \text{State} \), \( r \notin \mathcal{X} \), and no operator in \( \Sigma \) has \( \text{State} \) as an argument sort.

The rewriting relation \( \to_R \) on \( T_\Sigma(\mathcal{X}) \) is \( t \xrightarrow{p}_R t' \) (or \( \to_R \)) if \( p \in \text{Pos}_\Sigma(t) \), \( l \to r \in R \), \( t|_p = \sigma(l) \), and \( t' = t[\sigma(r)]_p \) for some \( \sigma \). The relation \( \to_{R/E} \) on \( T_\Sigma(\mathcal{X}) \) is \( \equiv_E \to_R \equiv_E \), i.e., \( t \to_{R/E} s \) iff \( \exists u_1, u_2 \in T_\Sigma(\mathcal{X}) \) s.t. \( t =_E u_1 \), \( u_1 \to_{R/E} u_2 \), and \( u_2 =_E s \). Note that \( \to_{R/E} \) on \( T_\Sigma(\mathcal{X}) \) induces a relation \( \to_{R/E} \) on \( T_{\Sigma/E}(\mathcal{X}) \) by \( [t]_E \to_{R/E}[t']_E \) iff \( t \to_{R/E} t' \).

When \( \mathcal{R} = (\Sigma, E, R) \) is a topmost rewrite theory, we can safely restrict ourselves to the general rewriting relation \( \to_{R/E} \) on \( T_\Sigma(\mathcal{X}) \), where the rewriting relation \( \to_{R/E} \) on \( T_\Sigma(\mathcal{X}) \) is \( t \xrightarrow{p}_{R/E} t' \) (or \( \to_{R/E} \)) if \( p \in \text{Pos}_\Sigma(t) \), \( l \to r \in R \), \( t|_p =_E \sigma(l) \), and \( t' = t[\sigma(r)]_p \) for some \( \sigma \). Note that \( \to_{R/E} \) on \( T_\Sigma(\mathcal{X}) \) induces a relation \( \to_{R/E} \) on \( T_{\Sigma/E}(\mathcal{X}) \) by \( [t]_E \to_{R/E}[t']_E \) iff \( t \to_{R/E} w \) and \( w =_E t' \). We say that a term \( t \) is \( R, E \)-irreducible if there is no term \( t' \) such that \( t \to_{R/E} t' \); this is extended to substitutions in the obvious way.

The narrowing relation \( \rightsquigarrow_R \) on \( T_\Sigma(\mathcal{X}) \) is \( t \xrightarrow{p}_{\sigma,R} t' \) (or \( \rightsquigarrow_{\sigma,R} \)) if \( p \in \text{Pos}_\Sigma(t) \), \( l \to r \in R \), \( t|_p =_E \sigma(l) \), and \( t' = t[\sigma(r)]_p \) for some \( \sigma \). Assuming that \( E \) has a finitary and complete unification algorithm, the narrowing relation \( \rightsquigarrow_{R,E} \) on \( T_\Sigma(\mathcal{X}) \) is \( t \xrightarrow{p}_{\sigma,R,E} t' \) (or \( \rightsquigarrow_{\sigma,R,E} \)) if \( p \in \text{Pos}_\Sigma(t) \), \( l \to r \in R \), \( \sigma \in \text{CSU}_E(t|_p = l) \), and \( t' = \sigma(t[r]_p) \).

The use of topmost rewrite theories is entirely natural for communication protocols, since all state transitions can be viewed as changes of the global distributed state. It also provides several advantages (see \cite{21}): (i) as pointed out above the relation \( \to_{R,E} \) achieves the same effect as the relation \( \to_{R/E} \), and (ii) we obtain a completeness result between narrowing \( \rightsquigarrow_{R,E} \) and rewriting \( \to_{R/E} \).

**Theorem 1 (Topmost Completeness).** \cite{21} Let \( \mathcal{R} = (\Sigma, E, R) \) be a topmost rewrite theory, \( t, t' \in T_\Sigma(\mathcal{X}) \), and let \( \sigma \) be a substitution such that \( \sigma(t) \to_{R,E} t' \). Then, there are substitutions \( \theta, \tau \) and a term \( t'' \) such that \( t \rightsquigarrow_{R,E} t'' \), \( \sigma(t) =_E \tau(\theta(t)) \), and \( t'' =_E \tau(t'') \).

In this paper, we consider only equational theories \( E = E' \cup Ax \) such that the rewrite rules \( E' \) are confluent, coherent, and terminating modulo axioms \( Ax \) such as commutativity \( (C) \), associativity-commutativity \( (AC) \), or associativity-commutativity plus identity \( (ACU) \) of some function symbols. We also require axioms \( Ax \) to be regular, i.e., for each equation \( l = r \in Ax \), \( \text{Var}(l) = \text{Var}(r) \). Note that axioms such as commutativity \( (C) \), associativity-commutativity \( (AC) \), or associativity-commutativity plus identity \( (ACU) \) are regular. The Maude-NPA has then both dedicated and generic methods for solving unification problems in such theories \( E' \cup Ax \) \cite{10,11,12}. 
3. Maude-NPA’s Execution Model

Given a protocol $\mathcal{P}$, we first explain how its states are modeled algebraically. The key idea is to model protocol states as elements of an initial algebra $T_{\Sigma P/E P}$, where $\Sigma P$ is the signature defining the sorts and function symbols for the cryptographic functions and for all the state constructor symbols, and $E P$ is a set of equations specifying the algebraic properties of the cryptographic functions and the state constructors. Therefore, a state is an $E P$-equivalence class $[t] \in T_{\Sigma P/E P}$ with $t$ a ground $\Sigma P$-term. However, since the number of states $T_{\Sigma P/E P}$ is in general infinite, rather than exploring concrete protocol states $[t] \in T_{\Sigma P/E P}$, we explore symbolic state patterns $[t(x_1, \ldots, x_n)] \in T_{\Sigma P/E P}(X)$ on the free $(\Sigma P, E P)$-algebra over a set of variables $X$. In this way, a state pattern $[t(x_1, \ldots, x_n)]$ represents not a single concrete state but a possibly infinite set of such states, namely all the instances of the pattern $[t(x_1, \ldots, x_n)]$ where the variables $x_1, \ldots, x_n$ have been instantiated by concrete ground terms.

In the Maude-NPA [4, 7], a state in the protocol execution is a term $t$ of sort State, $t \in T_{\Sigma P/E P}(X)_{\text{State}}$. A state is then a multiset built by an associative and commutative union operator $\cup_\ast$ with identity operator $\emptyset$. Each element in the multiset is either a strand or the intruder’s knowledge at that state, both explained below.

A strand $\llbracket t \rrbracket$ represents the sequence of messages sent and received by a principal executing the protocol or by the intruder. A principal sending (resp., receiving) a message $msg$ is represented by $msg^+$ (resp. $msg^-$). We write $m^+\mid m^-$ to denote $m^+$ or $m^-$, indistinctively. We often write $+(m)$ and $-(m)$ instead of $m^+$ and $m^-$, respectively. A strand is then a list $[msg_1^\pm, msg_2^\pm, msg_3^\pm, \ldots, msg_{k-1}^\pm, msg_k^\pm]$ describing the sequence of send and receive actions of a principal role in a protocol, where each $msg_i$ is a term of a special sort $Msg$ described below, i.e., $msg_i \in T_{\Sigma P/E P}(X)_{\text{Msg}}$. In Maude-NPA, strands evolve over time as the send and receive actions take place, and thus we use the symbol $\mid$ to divide past and future in a strand, i.e., $[nil, msg_1^\pm, \ldots, msg_{j-1}^\pm \mid msg_j^\pm, msg_{j+1}^\pm, \ldots, msg_k^\pm, nil]$ where $msg_1^\pm, \ldots, msg_{j-1}^\pm$ are the past messages, and $msg_j^\pm, msg_{j+1}^\pm, \ldots, msg_k^\pm$ are the future messages ($msg_j^\pm$ is the immediate future message). The nils are present so that the bar may be placed at the beginning or end of the strand if necessary. A strand $[msg_1^\pm, \ldots, msg_k^\pm]$ is a shorthand for $[nil \mid msg_1^\pm, \ldots, msg_k^\pm, nil]$. We often remove the nils for clarity, except when there is nothing else between the vertical bar and the beginning or end of a strand. We write $\mathcal{S}_P$ for the set of strands in the specification of the protocol $\mathcal{P}$, including the strands that describe the intruder’s behavior.

The intruder’s knowledge is represented as a multiset of facts unioned together with an associative and commutative union operator $\cup_\ast$ with identity operator $\emptyset$. There are two kinds of intruder facts: positive knowledge facts (the intruder knows message expression $m$, i.e., $m \in \mathcal{I}$), and negative knowledge facts (the intruder does not yet know $m$ but will know it in a future state, i.e., $m \not\in \mathcal{I}$).

Maude-NPA uses a special sort $Msg$ of messages that allows the protocol specifier to describe other sorts as subsorts of the top sort $Msg$. The specifier can make use of another special sort $\text{Fresh}$ in the protocol-specific signature $\Sigma$ for
representing fresh unguessable values, e.g., nonces. The meaning of a variable of sort \texttt{Fresh} is that it will never be instantiated by an \texttt{E}-unifier generated during the protocol analysis. This ensures that if two nonces are represented using different variables of sort \texttt{Fresh}, they will never be identified and no approximation for nonces is necessary. We make explicit the \texttt{Fresh} variables \(r_1, \ldots, r_k \ (k \geq 0)\) generated by a strand by writing :: \(r_1, \ldots, r_k :: [msg_{i_1}^+, \ldots, msg_{i_j}^+]\), where each \(r_i\) appears first in an output message \(msg_{i_j}^+\) and can later be used in any input and output message of \(msg_{i_j}^+, \ldots, msg_{i_j}^+\). Fresh variables generated by a strand are unique to that strand.

Let us introduce the well-known Diffie-Hellman protocol as a motivating example.

**Example 1.** The Diffie-Hellman protocol uses exponentiation to share a secret between two parties, Alice and Bob. There is a public constant, denoted by \(g\), which will be the base of the exponentiations. We represent the product of exponents by using the symbol \(*\). Nonces are represented by \(N_x\), denoting a nonce created by principal \(X\). Raising message \(M\) to the power of exponent \(X\) is denoted by \((M)^X\). Encryption of message \(M\) using the key \(K\) is denoted by \(\{M\}_K\). The protocol description is as follows.

1. \(A \leftrightarrow B : \{A; B; g^{N_A}\}\)
   Alice sends her name, Bob’s name, and an exponentiation of a new nonce \(N_A\) created by her to Bob.

2. \(B \leftrightarrow A : \{A; B; g^{N_B}\}\)
   Bob sends his name, Alice’s name, and an exponentiation of a new nonce \(N_B\) created by him to Alice.

3. \(A \leftrightarrow B : \text{secret} \cdot g^{N_A \cdot N_B}\)
   Bob receives \(g^{N_A}\) and he raises it to the \(N_B\) to obtain the key \(g^{N_A \cdot N_B}\). He sends a secret to Alice encrypted using the key. Likewise, when Alice receives \(g^{N_B}\), she raises it to the \(N_A\), to obtain the key \(g^{N_B \cdot N_A}\). We assume that exponentiation satisfies the equation \(g^{N_A \cdot N_B} = g^{N_B \cdot N_A}\) and that the product operation \(*\) is associative and commutative, so that

\[ g^{N_B \cdot N_A} = g^{N_A \cdot N_B} = g^{N_B \cdot N_A} \]

and therefore both Alice and Bob share the same key.

In the Maude-NPA’s formalization of the protocol, we explicitly specify the signature \(\Sigma\) describing the sorts and operations for messages, nonces, etc. A nonce \(N_A\) is denoted by \(n(A, r)\), where \(r\) is a unique variable of sort \texttt{Fresh}. Concatenation of two messages, e.g., \(N_A\) and \(N_B\), is denoted by the operator \(\_\cdot\_\), e.g., \(n(A, r) \cdot n(B, r')\). Encryption of a message \(M\) is denoted by \(e(A, M)\), e.g., \(\{N_B\}_{K_B}\) is denoted by \(e(K_B, n(B, r'))\). Decryption is similarly denoted by \(d(A, M)\). Raising a message \(M\) to the power of an exponent \(E\) (i.e., \(M^E\)) is denoted by \(\exp(M, E)\), e.g., \(g^{N_B}\) is denoted by \(\exp(g, n(B, r'))\). Associative-commutative multiplication of nonces is denoted by \(*\)\_\_\_. A secret generated by a principal is denoted by \(\text{sec}(A, r)\), where \(r\) is a unique variable
of sort \textit{Fresh}. The protocol-specific signature $\Sigma$ contains the following subsort relations \((\text{Name}, \text{Nonce}, \text{Secret}, \text{Enc}, \text{Exp} < \text{Msg})\) and \((\text{Gen}, \text{Exp} < \text{GenvExp})\) and the following operators:

$$
\begin{align*}
\text{a, b, i} &: \rightarrow \text{Name} \\
\text{n} &: \text{Name} \times \text{Fresh} \rightarrow \text{Nonce} \\
\cdot : & : \text{Msg} \times \text{Msg} \rightarrow \text{Msg} \\
\text{exp} &: \text{GenvExp} \times \text{Nonce} \rightarrow \text{Exp} \\
\text{g} &: \rightarrow \text{Gen} \\
\text{sec} &: \text{Name} \times \text{Fresh} \rightarrow \text{Secret} \\
\text{e, d} &: \text{Key} \times \text{Msg} \rightarrow \text{Enc} \\
\text{exp} &: \text{GenvExp} \times \text{Nonce} \rightarrow \text{Exp} \\
\text{\_\_\_} &: \text{Nonce} \times \text{Nonce} \rightarrow \text{Nonce}
\end{align*}
$$

In the following we will use letters $A, B$ for variables of sort $\text{Name}$, letters $r, r', r''$ for variables of sort $\text{Fresh}$, and letters $M, M_1, M_2, Z$ for variables of sort $\text{Msg}$; whereas letters $X, Y$ will also represent variables, but their sort will depend on the concrete position in a term. The encryption/decryption cancellation properties are described using the equations

$$
e(X, d(X, Z)) = Z \text{ and } d(X, e(X, Z)) = Z$$

in $E_P$. The key algebraic property of exponentiation, $z^x y = z^{xy}$, is described using the equation

$$
\text{exp}(\text{exp}(W, Y), Z) = \text{exp}(W, Y \ast Z)
$$

in $E_P$ (where $W$ is of sort $\text{Gen}$ instead of the more general sort $\text{GenvExp}$ in order to provide a finitary narrowing-based unification procedure modulo $E_P$, see \cite{3} for details on this concrete equational theory). Although multiplication modulo a prime number has a unit and inverses, we have only included the algebraic properties that are necessary for Diffie-Hellman to work. The two strands $\mathcal{P}$ associated to the protocol roles, Alice and Bob, shown above are:

$$
:: r, r' :: [ (A; B; \text{exp}(g, n(A, r)))^+, (B; A; X)^-, (e(\text{exp}(X, n(A, r)), \text{sec}(A, r')))^+] \\
:: r'' :: [ (A; B; Y)^-, (B; A; \text{exp}(g, n(B, r'')))^+, (e(\text{exp}(Y, n(B, r'')), \text{SR})^-]^-
$$

The following strands describe the intruder abilities according to the Dolev-Yao attacker’s capabilities \cite{2}.

- \([M_1^-, M_2^-, (M_1 \ast M_2)^+]\] Concatenation
- \([(M_1; M_2)^-, M_1^+]\] Left-deconcatenation
- \([(M_1; M_2)^-, M_2^+]\] Right-deconcatenation
- \([ K^-, M^-, e(K, M)^+]\] Encryption
- \([ K^-, M^-, d(K, M)^+]\] Decryption
- \([ M_1^-, M_2^-, (M_1 \ast M_2)^+]\] Multiplication
- \([ M_1^-, M_2^-, \text{exp}(M_1, M_2)^+]\] Exponentiation
• \([ g^+ ] \) Generator

• \([ A^+ ] \) All names are public

• :: \( r'' : [ n(i, r''')^+ ] \) Generation of intruder nonces

Note that the intruder cannot extract information from either an exponentiation or a product of exponents, but can only compose them. Also, the intruder cannot extract information directly from an encryption but it can indirectly by using a decryption and the cancellation of encryption and decryption, which is an algebraic property, i.e., \([ K^-, e(K, M)^-, M^+] =_E_p [ K^-, e(K, M)^-, d(K, e(K, m))^+] \].

3.1. Backwards Reachability Analysis

Our protocol analysis methodology is then based on the idea of backwards reachability analysis, where we begin with one or more state patterns corresponding to attack states, and want to prove or disprove that they are unreachable from the set of initial protocol states. In order to perform such a reachability analysis we must describe how states change as a consequence of principals performing protocol steps and of intruder actions. This can be done by describing such state changes by means of a set \( R_P \) of rewrite rules, so that the rewrite theory \((\Sigma_P, E_P, R_P)\) characterizes the behavior of protocol \( P \) modulo the equations \( E_P \). In the case where new strands are not introduced into the state, the corresponding rewrite rules in \( R_P \) are as follows:¹ where \( L, L_1, L_2 \) denote lists of input and output messages (+m,−m), \( IK, IK' \) denote sets of intruder facts (\( m \in \mathcal{I}, m/\in \mathcal{I} \)), and \( SS, SS' \) denote sets of strands:

\[
\begin{align*}
[L | M^-, L'] & \& SS \& (M \in \mathcal{I}, IK) \rightarrow [L, M^- | L'] \& SS \& (M \in \mathcal{I}, IK) \quad (1) \\
[L | M^+, L'] & \& SS \& IK \rightarrow [L, M^+ | L'] \& SS \& IK \quad (2) \\
[L | M^+, L'] & \& SS \& (M \notin \mathcal{I}, IK) \rightarrow [L, M^+ | L'] \& SS \& (M \notin \mathcal{I}, IK) \quad (3)
\end{align*}
\]

In a forward execution of the protocol strands, Rule (1) describes a message reception event in which an input message is received from the intruder; the intruder’s knowledge acts in fact as the only channel through which all communication takes place. Rule (2) describes a message send in which the intruder’s knowledge is not increased; it is irrelevant where the message goes. Rule (3) describes the alternative case of a send event such that the intruder’s knowledge is positively increased. Note that Rule (3) makes explicit when the intruder learned a message \( M \), which was recorded in the previous state by the negative fact \( M \notin \mathcal{I} \). A fact \( M \notin \mathcal{I} \) can be paraphrased as: “the intruder does not yet know \( M \), but will learn it in the future”¹. This enables a very important restriction of the tool, expressed by saying that the intruder learns a term only once: if the intruder needs to use a term twice, then he must learn it the first time.

¹To simplify the exposition, we omit the fresh variables at the beginning of each strand in a rewrite rule.
it is needed; if he learns a term and needs to learn it again in a previous state, found later during the backwards search, then the state will be discarded as unreachable. Note that Rules (1)–(3) are generic: they belong to \( R_P \) for any protocol \( P \).

It is also the case that when we are performing a backwards search, only the strands that we are searching for are listed explicitly: extra strands necessary to reach an initial state are dynamically added to the state by explicit introduction through protocol-specific rewrite rules (one for each output message \( u^+ \) in an honest or intruder strand in \( S_P \)) as follows:

\[
\text{for each } [l_1, u^+, l_2] \in S_P : [l_1 | u^+, l_2] & SS & (u \notin I, IK) \rightarrow SS & (u \in I, IK) \quad (4)
\]

where \( u \) denotes a message, \( l_1, l_2 \) denote lists of input and output messages \((+m, -m)\), \( IK \) denotes a set of intruder facts \((m \in I, m/\notin I)\), and \( SS \) denotes a set of strands. For example, intruder concatenation of two learned messages, as well as the learning of such a concatenation by the intruder, is described as follows:

\[
[M_1^-, M_2^+] \mid (M_1; M_2)^+ \] \& SS \& ((M_1; M_2) \notin I, IK) \rightarrow SS \& ((M_1; M_2) \in I, IK)
\]

This rewrite rule can be understood, in a backwards search, as “in the current state the intruder is able to learn a message that matches the pattern \( M_1; M_2 \) if he is able to learn message \( M_1 \) and message \( M_2 \) in prior states”. In summary, for a protocol \( P \), the set \( R_P \) of rewrite rules obtained from the protocol strands \( S_P \) that are used for backwards narrowing reachability analysis modulo the equational properties \( E_P \) is \( R_P = \{[1], [2], [3]\} \cup [4] \). These rewrite rules give the basic execution model of Maude-NPA. However, as we shall see, it will later be necessary to modify them in order to optimize the search. In later sections of this paper we will show how these rules can be modified to optimize the search while still maintaining completeness.

On the other hand, the assumption that algebraic properties are expressed as equational theories \( E = E' \cup Ax \) whose equations \( E' \) are confluent, coherent, and terminating rewrite rules modulo regular equational axioms \( Ax \) such as commutativity (\( C \)), associativity-commutativity (\( AC \)), or associativity-commutativity plus identity (\( ACU \)) of some function symbols, implies some extra conditions on the rewrite theory \( R_P \) (see [4]). Namely, for any term \( m \in I \) (resp. term \( m^- \)) and any \( E', Ax \)-irreducible substitution \( \sigma, \sigma(m) \in I \) (resp. \( (\sigma(m))^- \)) must be \( E', Ax \)-irreducible. This is because many of our optimization techniques rely on the assumption that terms have a unique normal form modulo a regular equational theory, and achieve their results by reasoning about the normal forms of terms.

Finally, states have, in practice, another component containing the actual message exchange sequence between principal or intruder strands (i.e., all the expressions \( m^\pm \) exchanged between the honest and intruder strands). We do not make use of the message exchange sequence until Section 4.7.2, so we delay its introduction until there.

The way to analyze backwards reachability is then relatively easy, namely, to run the protocol “in reverse.” This can be achieved by using the set of rules
$R_p^{-1}$, where $v \rightarrow u$ is in $R_p^{-1}$ iff $u \rightarrow v$ is in $R_p$. Reachability analysis can be performed symbolically, not on concrete states but on symbolic state patterns $[t(x_1, \ldots, x_n)]_{E_p}$, by means of narrowing modulo $E_p$ (see Section 2). We call *attack patterns* those states patterns (i.e., terms with logical variables) used to start the narrowing-based backwards reachability analysis. An *initial state* is a state where all strands have their vertical bar at the beginning and there is no positive fact of the form $u \in I$ for a message term $u$ in the intruder’s knowledge. If no initial state is found during the backwards reachability analysis from an attack pattern, the protocol has been proved secure for that attack pattern with respect to the assumed intruder capabilities and the algebraic properties. If an initial state is found, then we conclude that the attack pattern is possible and a concrete attack can be inferred from the exchange sequence stored in the initial state. Note that an initial state may be generic, in the sense of having logical variables for those elements that are not relevant for the attack.

**Example 2.** (Example [1], continued) The attack pattern that we are looking for is one in which Bob completes the protocol and the intruder is able to learn the secret. The attack state pattern to be given as input to Maude-NPA is:

\[
:: r' :: [(A; B; Y)\vdash 1, (B; A; \exp(g, n(B, r')))\vdash 1, (\exp(Y, n(B, r')), \sec(a, r''))\vdash 1, \nil] \\
& SS & (\sec(a, r'')\in I, \ IK)
\]

Using the above attack pattern Maude-NPA is able to find an initial state of the protocol, showing that the attack state is possible. Note that this initial state is generalized to two sessions in parallel: one session where Alice (i.e., principal named $a$) is talking to another principal $B'$ — in this session the intruder gets a nonce $n(a, r)$ originated from $a$ — and another session where Bob (i.e., principal named $b$) is trying to talk to Alice. If we instantiate $B'$ to be $b$, then one session is enough, although the tool returns the most general attack. The strands associated to the initial state found by the backwards search are as follows:

\[
\begin{align*}
\nil \mid \exp(g, n(a, r)))\vdash 1, Z\vdash 1, \exp(g, Z \ast n(a, r))\vdash 1] & & \\
\nil \mid \exp(g, Z \ast n(a, r))\vdash 1, \exp(g, n(b, r'))\vdash 1, \exp(g, W \ast n(b, r'))\vdash 1] & & \\
\nil \mid \exp(g, n(b, r'))\vdash 1, \sec(a, r'')\vdash 1, \exp(g, W \ast n(b, r'))\vdash 1] & & \\
\nil \mid \exp(g, W \ast n(b, r'))\vdash 1, \exp(g, n(b, r'))\vdash 1, \sec(a, r'')\vdash 1] & & \\
\nil \mid \exp(g, n(b, r'))\vdash 1, \exp(g, n(b, r'))\vdash 1] & & \\
\nil \mid \exp(g, n(b, r'))\vdash 1, \exp(g, n(b, r'))\vdash 1] & & \\
\nil \mid \exp(g, n(b, r'))\vdash 1, \exp(g, n(b, r'))\vdash 1] & & \\
\nil \mid (B'; \exp(g, n(a, r)))\vdash 1, \exp(g, n(a, r))\vdash 1] & & \\
:: r' :: \\
\nil \mid (a; b; \exp(g, W))\vdash 1, (a; b; \exp(g, n(b, r'))\vdash 1, \exp(g, W \ast n(b, r'))\vdash 1, \sec(a, r'')\vdash 1] & & \\
:: r'' :: \\
\nil \mid (a; B'; \exp(g, n(a, r)))\vdash 1, (a; B'; \exp(g, Z))\vdash 1, \exp(g, Z \ast n(a, r))\vdash 1, \sec(a, r'')\vdash 1] & &
\end{align*}
\]
Note that the last two strands, generating fresh variables \( r, r', r'' \), are protocol strands and the others are intruder strands.

The concrete message exchange sequence obtained by the reachability analysis is the following:

\[
\begin{align*}
&1. (a; b; \exp(g, W))^- \\
&2. (a; b; \exp(g, n(b, r')))^+ \\
&3. (a; b; \exp(g, n(b, r')))^- \\
&4. (b; \exp(g, n(b, r')))^+ \\
&5. (b; \exp(g, n(b, r')))^- \\
&6. (\exp(g, n(b, r')))^+ \\
&7. (\exp(g, n(b, r')))^- \\
&8. W^- \\
&9. \exp(g, W * n(b, r'))^+ \\
&10. (a; B'; \exp(g, n(a, r)))^+ \\
&11. (a; B'; \exp(g, n(a, r)))^- \\
&12. (B'; \exp(g, n(a, r)))^+ \\
&13. (B'; \exp(g, n(a, r)))^- \\
&14. (\exp(g, n(a, r)))^+ \\
&15. (\exp(g, n(a, r)))^- \\
&16. Z^- \\
&17. \exp(g, Z * n(a, r))^+ \\
&18. (a; B'; \exp(g, Z))^- \\
&19. e(\exp(g, Z * n(a, r)), sec(a, r''))^+ \\
&20. e(\exp(g, Z * n(a, r)), sec(a, r''))^- \\
&21. \exp(g, Z * n(a, r))^- \\
&22. sec(a, r'')^+ \\
&23. \exp(g, W * n(b, r'))^- \\
&24. sec(a, r'')^- \\
&25. e(\exp(g, W * n(b, r'), sec(a, r''))^+ \\
&26. e(\exp(g, W * n(b, r')).sec(a, r''))^- \\
\end{align*}
\]

Step 1) describes Bob (i.e., principal named b) receiving an initiating message from the intruder impersonating Alice. Step 2) describes Bob sending the response, and Step 3) describes the intruder receiving it. Steps 4) through 9) describe the intruder computing the key \( \exp(g, W * n(b, r')) \) she will use to communicate with Bob. Step 10) describes Alice initiating the protocol with a principal \( B' \). Step 11) describes the intruder receiving it, and steps 11) through 17) describe the intruder constructing the key \( \exp(g, Z * n(a, r)) \) she will use to communicate with Alice. Steps 18) and 19) describe Alice receiving the response from the intruder impersonating \( B' \) and Alice sending the encrypted message. Steps 20) through 22) describe the intruder decrypting the message to get the secret. In steps 23) through 25) the intruder re-encrypts the secret with the key she shares with Bob and sends it, and in Step 26) Bob receives the message.

Note that there are some intruder strands missing in the initial state because certain terms are assumed to be trivially generateable by the intruder, and so not searched for; namely, intruder strands generating variable \( Z \), variable \( W \), term \( (a; b; \exp(g, W)) \), and term \( (a; B'; \exp(g, Z)) \). Variables \( Z \) and \( W \) can be filled in with any nonce, for instance nonces generated by the intruder, such as \( W = n(i, r''') \) and \( Z = n(i, r''''') \) in the following way:

\[
:: r''' :: [\nil | (n(i, r'''))]^+ & :: r''''' :: [\nil | (n(i, r'''''))]^+
\]

Also, note that nonces \( W \) and \( Z \) are used by the intruder to generate messages \((a; b; \exp(g, W))\) and \((a; B'; \exp(g, Z))\) in the following way:

\[
[\nil | (a)^+] & [\nil | (b)^+] & [\nil | (B')^+] & \\
[\nil | (g)^+] & [\nil | (g)^-, W^-, \exp(g, W)^+] & [\nil | (g)^-, Z^-, \exp(g, Z)^+] & \\
[\nil | (a)^-, (b)^-, (a; b)^+] & [\nil | (a; b)^-, (\exp(g, W))^-, (a; b; \exp(g, W))^+] & \\
[\nil | (a)^-, (B')^-, (a; B')^+] & [\nil | (a; B')^-, (\exp(g, Z))^-, (a; B'; \exp(g, Z))^+]
\]
4. State Space Reduction Techniques

In this section we present Maude-NPA’s state space reduction techniques. Before presenting them, we formally identify two classes of states that can be safely removed: unreachable and redundant states. We begin the presentation with the notion of grammars, and its associated state space reduction technique, which is the oldest Maude-NPA technique and does much to identify and remove non-terminating search paths. In many cases (although not all) this is enough to turn an infinite search space into a finite one. We then describe a number of simple techniques which remove states that can be shown to be unreachable, thus saving the cost of searching for them. We conclude by describing two powerful techniques for eliminating redundant states: subsumption partial order reduction and the super-lazy intruder, and we prove their completeness.

First, the Maude-NPA satisfies a very general completeness result.

Theorem 2 (Completeness). [4] Given a topmost rewrite theory \( R_P = (\Sigma_P, E_P, R_P) \) representing protocol \( P \), and a non-initial state \( St \) (with logical variables), if there is a substitution \( \sigma \) and an initial state \( St_{ini} \) such that \( \sigma(St) \xrightarrow{R_P^{-1}, E_P} St_{ini} \), then there are substitutions \( \sigma', \rho \) and an initial state \( St'_{ini} \) such that \( St \xrightarrow{\sigma', R_P^{-1}, E_P} St'_{ini} \), \( \sigma = E_P \sigma' \circ \rho \), and \( St_{ini} = E_P \rho(St'_{ini}) \).

Our optimizations are able to identify two kinds of unproductive states: unreachable and redundant states.

Definition 1 (Unreachable States). Given a topmost rewrite theory \( R_P = (\Sigma_P, E_P, R_P) \) representing protocol \( P \), a state \( St \) (with logical variables) is unreachable if there is no sequence \( St \xrightarrow{\sigma, R_P^{-1}, E_P} St_{ini} \) leading to an initial state \( St_{ini} \).

Definition 2 (Redundant States). Given a topmost rewrite theory \( R_P = (\Sigma_P, E_P, R_P) \) representing protocol \( P \) and a state \( St \) (with logical variables), a backwards narrowing step \( St \xrightarrow{\sigma, R_P^{-1}, E_P} St_{ini} \) is called redundant (or just state \( St_1 \) is identified as redundant) if for any initial state \( St_{ini1} \) reachable from \( St_1 \), i.e., \( St_1 \xrightarrow{\theta_1, R_P^{-1}, E_P} St_{ini1} \), there are states \( St_2 \) and \( St_{ini2} \), a narrowing step \( St \xrightarrow{\theta_2, R_P^{-1}, E_P} St_2 \), a narrowing sequence \( St_2 \xrightarrow{\theta_2, R_P^{-1}, E_P} St_{ini2} \), and a substitution \( \rho \) such that \( \sigma_1 \circ \theta_1 = E_P \sigma_2 \circ \theta_2 \circ \rho \) and \( St_{ini1} = E_P \rho(St_{ini2}) \).

There are three reasons for wanting to detect unproductive backwards narrowing reachability steps. One is to reduce, if possible, the initially infinite search space to a finite one, as it is sometimes possible to do with the use of grammars, by removing unreachable states. Another is to reduce the size of a (possibly finite) search space by eliminating unreachable states early, i.e., before they are eliminated by exhaustive search. This elimination of unreachable states can have an effect far beyond eliminating a single node in the search space, since a single unreachable state may appear multiple times and/or have multiple descendants. Finally, if there are several steps leading to the same initial state,
as for redundant states, then it is also possible to use various partial order reduction techniques that can further shrink the number of states that need to be explored.

4.1. Grammars

The Maude-NPA's ability to reason effectively about a protocol's algebraic properties is a result of its combination of symbolic reachability analysis using narrowing modulo equational properties (see Section 2), together with its grammar-based techniques for reducing the size of the search space. The key idea of grammars is to detect terms \( t \) in positive facts \( t \in I \) of the intruder's knowledge of a state \( St \) that will never be transformed into a negative fact \( \theta(t) \notin I \) in any initial state \( St' \) backwards reachable from \( St \). This means that \( St \) can never reach an initial state and therefore it can be safely discarded. Here we briefly explain how grammars work as a state space reduction technique and refer the reader to [14, 4] for further details.

Automatically generated grammars \( \langle G_1, \ldots, G_m \rangle \) represent unreachability information (or co-invariants), i.e., typically infinite sets of states unreachable from an initial state. These automatically generated grammars are very important in our framework, since in the best case they can reduce the infinite search space to a finite one, or, at least, can drastically reduce the search space.

Example 3. Consider again the attack pattern \( \{\} \) in Example 2. After a couple of backwards narrowing steps, the Maude-NPA finds the following state:

\[
\begin{align*}
[ \text{nil} | (M; \text{sec}(a,r''))^- , (\text{sec}(a,r''))^+ ] \& \\
:: r' : [(A;B;Y)^-, (B;A;\text{exp}(g,n(B,r')))^+ | (e(\text{exp}(Y,n(B,r')), \text{sec}(a,r'')))^- ] \& \\
( (M;\text{sec}(a,r'')) \notin I, e(\text{exp}(Y,n(B,r')), \text{sec}(a,r''))) \notin I, \text{sec}(a,r'')/ \notin I )
\end{align*}
\]

which corresponds to the intruder obtaining (i.e., learning) the message \( \text{sec}(a,r'') \) from a bigger message \( (M;\text{sec}(a,r'')) \), although the contents of variable \( M \) have not yet been found by the backwards reachability analysis. This process of adding more and more intruder strands that look for terms \( (M';M;\text{sec}(a,r'')) \) \( (M'';M';M;\text{sec}(a,r'')) \), \ldots can go on forever. Note that if we carefully check the strands for the protocol, we can see that the honest strands either never produce a message with normal form "M;secret" or such a message is under a public key encryption (and thus the intruder cannot get the contents), so the previous state is clearly unreachable and can be discarded. The grammar, which is generated by Maude-NPA, capturing the previous state as unreachable, is as follows:

\[
\begin{align*}
grl M \text{ inL} &\Rightarrow e(K, M) \text{ inL}. \\
grl M \text{ inL} &\Rightarrow d(K, M) \text{ inL}. \\
grl M \text{ inL} &\Rightarrow (M ; M') \text{ inL}. \\
grl M \text{ inL} &\Rightarrow (M' ; M) \text{ inL}. \\
grl M \text{ notInI}, \\
&M \text{ notLeq exp}(g, n(A, r)), \\
&M \text{ notLeq B ; exp}(g, n(A, r')) \Rightarrow (M' ; M) \text{ inL}.
\end{align*}
\]
where all the productions and exceptions refer to normal forms of messages w.r.t. the equational theory $E_P$.

Intuitively, the last production rule in the grammar above says that any term with normal form $M'; M$ cannot be learned by the intruder if the subterm $M$ is different from $\exp(g, n(A, r))$ and $B: \exp(g, n(A, r'))$ (i.e., it does not match such patterns) and the constraint $M \notin I$ appears explicitly in the intruder’s knowledge of the current state being checked for unreachability. Moreover, any term of any of the following normal forms: $e(A, M)$, $d(A, M)$, $(M'; M)$, or $(M; M')$ cannot be learned by the intruder if subterm $M$ is also not learnable by the intruder.

4.2. Public data

The simplest optimization possible is one that can be provided explicitly by the user. When we are searching for some data that we know is easy to learn by the intruder, the tool can avoid this by assuming that such data is public. Such data is considered public by using a special sort $\text{Public}$ and a subsort definition, e.g. “$\text{subsort Name < Public}$”. That is, given a state $St$ that contains an expression $t \in I$ in the intruder’s knowledge where $t$ is of sort $\text{Public}$, we can remove the expression $t \in I$ from the intruder’s knowledge, since the backwards reachability steps taken care of such a $t \in I$ are necessary in order to lead to an initial state but their inclusion in the message sequence is superfluous. The completeness proof for this optimization is trivial and thus omitted.

4.3. Limiting Dynamic Introduction of New Strands

As pointed out in Section 3.1, rules of type (4) allow the dynamic introduction of new strands. However, new strands can also be introduced by unification of a state containing a variable $SS$ denoting a set of strands and one of the rules of (1), (2), and (3), where variables $L$ and $L'$ denoting lists of input/output messages will be introduced by instantiation of $SS$. The same can happen with new intruder facts of the form $X \in I$, where $X$ is a variable, by instantiation of a variable $IK$ denoting the rest of the intruder knowledge.

Example 4. Consider a state $St$ of the form $SS & IK$ where $SS$ denotes a set of strands and $IK$ denotes a set of facts in the intruder’s knowledge. Now, consider Rule (1):

$$SS' & [L \mid M^-, L'] \& (M \in I, IK') \rightarrow SS' & [L, M^- \mid L'] \& (M \in I, IK')$$

The following backwards narrowing step applying such a rule can be performed from $St = SS \& IK$ using the unifier $\sigma = \{SS \mapsto SS' & [L, M^- \mid L'], IK \mapsto (M \in I, IK')\}$

$$SS & IK \xrightarrow{\sigma_{RE}} SS' & [L \mid M^-, L'] \& (M \in I, IK')$$

but this backwards narrowing step is unproductive, since it is not guided by the information in the attack state. Indeed, the same rule can be applied again using variables $SS'$ and $IK'$ and this can be repeated many times.
In order to avoid a huge number of unproductive narrowing steps by useless instantiation, we allow the introduction of new strands and/or new intruder facts only by rule application instead of just by unification. For this, we do two things:

1. we remove any of the following variables from attack patterns: \( SS \) denoting a set of strands, \( IK \) denoting a set of intruder facts, and \( L, L' \) denoting a set of input/output messages; and
2. we replace Rule (1) by the following Rule (5), since we do no longer have a variable denoting a set of intruder facts that has to be instantiated:

\[
SS \& (L \mid M^-, L') \& (M \in \mathcal{I}, IK) \rightarrow SS \& (L, M^- \mid L') \& IK
\]

Note that in order to replace Rule (1) by Rule (5) we have to assume that the intruder’s knowledge is a set of intruder facts without repeated elements, i.e., the union operator \( \cup \) is \( ACUI \) (associative-commutative-identity-idempotent).

Furthermore, one might imagine that Rule (3) and rules of type (4) must also be modified in order to remove the \( M \in \mathcal{I} \) expression from the intruder’s knowledge of the right-hand side of each rule. However, this is not so, since, by keeping the expression \( M \in \mathcal{I} \), we force the backwards application of the rule only when there is indeed a message for the intruder to be learned. This provides some form of on-demand evaluation of the protocol.

The completeness proof for this optimization is trivial and thus omitted. However, since we have modified the set of rules used for backwards reachability, we prove that such modification has the same reachability capabilities. The set of rewrite rules actually used for backwards narrowing is \( \mathcal{R}_P \). The following result ensures that \( \mathcal{R}_P \) and \( \mathcal{R}_{\mathcal{P}} \) compute similar initial states by backwards reachability analysis. Its proof is straightforward.

**Definition 3 (Inclusion).** Given a topmost rewrite theory \( \mathcal{R}_P = (\Sigma_P, E_P, \mathcal{R}_P) \) representing protocol \( \mathcal{P} \), and two states \( St_1, St_2 \), we abuse notation and write \( St_1 \subseteq E_P St_2 \) to denote that every state element (i.e., strand or intruder fact) in \( St_1 \) appears in \( St_2 \) (modulo \( E_P \)).

**Proposition 1.** Let \( \mathcal{R}_P = (\Sigma_P, E_P, \mathcal{R}_P) \) be a topmost rewrite theory representing protocol \( \mathcal{P} \). Let \( St = SS \& (ik, IK) \) where \( SS \) is a variable for strands, \( ik \) is a term representing a set of intruder facts, \( SS \) is a variable for strands, and \( IK \) is a variable for intruder knowledge. Let \( St' = SS \& ik \). If there is an initial state \( St_{ini} \) and a substitution \( \sigma \) such that \( St \sim^*_{\sigma, E_P^{-1}, \mathcal{R}_P} St_{ini} \), then there is an initial state \( St'_{ini} \) and two substitutions \( \sigma' \), \( \rho \) such that \( St' \sim^*_{\sigma', E_P^{-1}, \mathcal{R}_P} St'_{ini} \), \( \sigma = E_P \sigma' \circ \rho \), and \( \rho(St'_{ini}) \subseteq E_P St_{ini} \).

### 4.4. Partial Order Reduction Giving Priority to Input Messages

The different rewrite rules on which the backwards narrowing search from an attack pattern is based are in general executed non-deterministically. This
is because the order of execution can make a difference as to what subsequent rules can be executed. For example, an intruder cannot receive a term until it is sent by somebody, and that send action within a strand may depend upon other receives in the past. There is one exception, Rule (5) (originally Rule (1)), which, in a backwards search, only moves a negative term appearing right before the bar into the intruder’s knowledge.

Example 5. For instance, consider the attack pattern \( \dagger \) in Example 2. Since the strand in the attack pattern has the input message \((e(exp(Y,n(B,r')),sec(a,r'')))^-\) but also has the intruder challenge \(\text{sec}(a,r'')\in I\), there are several possible backwards narrowing steps: some processing the intruder challenge, and Rule (5) processing the input message.

The execution of Rule (5) in a backwards search does not disable any other transitions; indeed, it only enables send transitions. Thus, it is safe to execute it at each stage before any other transition. For the same reason, if several applications of Rule (5) are possible, it is safe to execute them all at once before any other transition. Requiring all executions of Rule (5) to execute first thus eliminates interleavings of Rule (5) with send and receive transitions, which are equivalent to the case in which Rule (5) executes first. In practice, this typically cuts down in half the search space size. The completeness proof for this optimization is trivial and is thus omitted.

Similar strategies have been employed by other tools in forward searches. For example, in [19], a strategy is introduced that always executes send transitions first whenever they are enabled. Since a send transition does not depend on any other component of the state in order to take place, it can safely be executed first. The original NPA also used this strategy; it had a receive transition (similar to the input message in Maude-NPA) which had the effect of adding new terms to the intruder’s knowledge, and which always was executed before any other transition once it was enabled.

4.5. Early Detection of Inconsistent States

There are several types of states that are always unreachable or inconsistent.

Example 6. Consider again the attack pattern \( \dagger \) in Example 2. After a couple of backwards narrowing steps, the Maude-NPA finds the following state, where the intruder learns \( e(exp(Y,n(B,r')),sec(a,r'')) \) by assuming she can learn \( exp(Y,n(B,r')) \) and \( sec(a,r'') \) and combines them:

\[
[ \text{nil} | (exp(Y,n(B,r')))\text{^-},(sec(a,r''))\text{^-},(e(exp(Y,n(B,r')),sec(a,r'')))^+ ] \&
:: r' ::
[ (A;B;Y)^-, (B;A;exp(g,n(B,r')))^+ | (e(exp(Y,n(B,r')),sec(a,r'')))^- ] \&
\text{(sec(a,r''))\in I, exp(Y,n(B,r'))\in I, e(exp(Y,n(B,r')),sec(a,r''))\notin I) (\dagger) }
\]
From this state, the intruder tries to learn \( \text{sec}(a, r'') \) by assuming she can learn messages \( (e(exp(Y, n(B, r')), \text{sec}(a, r''))) \) and \( exp(Y, n(B, r')) \) and combines them in a decryption:

\[
[ \text{nil} \mid (exp(Y, n(B, r')))^{-}, \ (e(exp(Y, n(B, r')), \text{sec}(a, r'')))^{-}, \ (\text{sec}(a, r''))^{+} ] \&
[ \text{nil} \mid (exp(Y, n(B, r')))^{-}, \ (\text{sec}(a, r''))^{-}, \ (e(exp(Y, n(B, r')), \text{sec}(a, r'')))^{+} ] \& \\
:: r' :: \\
[ (A; B; Y)^{-}, (B; A; exp(g, n(B, r')))^{+} \mid (e(exp(Y, n(B, r')), \text{sec}(a, r'')))^{-} ] \& \\
(\text{sec}(a, r'')) \in \mathcal{I}, \ e(exp(Y, n(B, r'))), \text{sec}(a, r'')) \in \mathcal{I}, \ e(exp(Y, n(B, r')), \text{sec}(a, r'')) \notin \mathcal{I}
\]

But then this state is inconsistent, since we have both the challenge \( e(exp(Y, n(B, r')), \text{sec}(a, r'')) \in \mathcal{I} \) and the already learned message \( e(exp(Y, n(B, r')), \text{sec}(a, r'')) \notin \mathcal{I} \) at the same time, violating the learn-only-once condition in Maude-NPA.

If the Maude-NPA attempts to search beyond an inconsistent state, it will never find an initial state. For this reason, the Maude-NPA search strategy always marks the following types of states as unreachable, and does not search beyond them any further:

1. A state \( St \) containing two contradictory facts \( t \in \mathcal{I} \) and \( t \notin \mathcal{I} \) (modulo \( E_P \)) for a term \( t \).
2. A state \( St \) whose intruder’s knowledge contains the fact \( t \notin \mathcal{I} \) and a strand of the form \([m_1^+, \ldots, t^-, \ldots, m_{j-1}^+, \ldots, m_k^+] \) (modulo \( E_P \)).
3. A state \( St \) containing a fact \( t \in \mathcal{I} \) such that \( t \) contains a fresh variable \( r \) and the strand in \( St \) indexed by \( r \), i.e., \( :: r_1, \ldots, r, \ldots, r_k :: [m_1^+, \ldots, m_{j-1}^+, \ldots, m_j^-, \ldots, m_k^+] \), cannot produce \( r \), i.e., \( r \) is not a subterm of any output message in \( m_1^+, \ldots, m_{j-1}^+ \).
4. A state \( St \) containing a strand of the form \([m_1^+, \ldots, t^-, \ldots, m_{j-1}^+, \ldots, m_j^+, \ldots, m_k^+] \) for some term \( t \) such that \( t \) contains a fresh variable \( r \) and the strand in \( St \) indexed by \( r \) cannot produce \( r \).

Note that case 2 will become an instance of case 1 after some backwards narrowing steps, and the same happens with cases 4 and 3. The proof of inconsistency of cases 1 and 3 is straightforward.

4.6. Transition Subsumption

Partial order reduction (POR) techniques are common in state exploration. However, POR techniques for narrowing-based state exploration do not seem to have been explored in detail, although they may be extremely relevant and may afford greater reductions than in standard state exploration based on ground terms rather than on terms with logical variables. For instance, the simple concept of two states being equivalent modulo renaming of variables does not
apply to standard state exploration, whereas it does apply to narrowing-based state exploration. In [8], Escobar and Meseguer studied narrowing-based state exploration and POR techniques, which may transform an infinite-state system into a finite one. However, the Maude-NPA needs a dedicated POR technique applicable to its specific execution model.

Let us motivate this POR technique with an example before giving a more detailed explanation.

**Example 7.** Consider again the attack pattern (†) in Example [4]. After a couple of backwards narrowing steps, the Maude-NPA finds the state (‡) of Example [7]:

\[
[\text{nil} \mid \text{exp}(Y, n(B, r'))^-, \text{sec}(a, r'^-), (e(\text{exp}(Y, n(B, r')), \text{sec}(a, r'')))^+] \land \\
\text{r'} :: [\text{G}, \text{N}] \land \\
[\text{nil} \mid \text{exp}(G, n(B, r'))^-, \text{exp}(G, N \ast n(B, r'))^+] \land \\
\text{r'} :: [\text{A}; B; \text{exp}(g, n(B, r'))]^+ \land \\
(e(\text{exp}(Y, n(B, r')), \text{sec}(a, r'')))^-] \land \\
(\text{sec}(a, r'') \in \mathcal{I}, \text{exp}(Y, n(B, r')) \in \mathcal{I}, \text{exp}(Y, n(B, r')) \in \mathcal{I})
\]

However, the following state is also generated after a couple of narrowing steps from the attack pattern, where, thanks to the equational theory, variable \(Y\) is instantiated to \(\text{exp}(G, N)\) for \(G\) a generator –indeed the constant \(g\)— and \(N\) a nonce variable:

\[
[\text{nil} \mid \text{exp}(G, n(B, r'))^-, \text{exp}(G, N \ast n(B, r'))^+] \land \\
[\text{nil} \mid \text{exp}(G, N \ast n(B, r'))^-, \text{sec}(a, r'^'')^-, (e(\text{exp}(G, N \ast n(B, r')), \text{sec}(a, r'')))^+] \land \\
\text{r'} :: [\text{A}; B; \text{exp}(G, N)]^-, \text{exp}(g, n(B, r'))^+ \\
| (e(\text{exp}(G, N \ast n(B, r')), \text{sec}(a, r'')))^-] \land \\
(\text{sec}(a, r'') \in \mathcal{I}, \text{exp}(G, n(B, r')) \in \mathcal{I}, \text{exp}(G, N \ast n(B, r')) \in \mathcal{I})
\]

However, the unreachability of the second state is implied (modulo \(E_P\)) by the unreachability of the first state; unreachability in the sense of Definition [4]. Intuitively, the challenges present in the first state that are relevant for backwards reachability are included in the second state, namely, the challenges \(\text{sec}(a, r''') \in \mathcal{I}\) and \(\text{exp}(Y, n(B, r')) \in \mathcal{I}\). Indeed, the unreachability of the following “kernel” state implies the unreachability of both states, although this kernel state is never computed by the Maude-NPA:

\[
\text{r'} :: [\text{A}; B; Y^-, (B; A; \text{exp}(g, n(B, r')))^+ | (e(\text{exp}(Y, n(B, r')), \text{sec}(a, r'')))^-] \land \\
(\text{sec}(a, r'') \in \mathcal{I}, \text{exp}(Y, n(B, r')) \in \mathcal{I})
\]

Note that the converse is not true, i.e., the second state does not imply the first one, since it contains one more intruder item relevant for backwards reachability purposes, namely \(N \in \mathcal{I}\).
Let us now formalize this state space reduction and prove its completeness. First, an auxiliary relation $St_1 \triangleright St_2$ identifying whether $St_1$ is smaller than $St_2$ in terms of messages to be learned by the intruder.

**Definition 4.** Given a topmost rewrite theory $R_P = (\Sigma_P, E_P, R_P)$ representing protocol $P$, and two non-initial states $St_1$ and $St_2$, we write $St_1 \triangleright St_2$ (or $St_2 \triangleright St_1$) if each intruder fact of the form $t \in \mathcal{I}$ in $St_1$ appears in $St_2$ (modulo $E_P$) and each non-initial strand in $St_1$ appears in $St_2$ (modulo $E_P$ and with the vertical bar at the same position).

Then, we define the relation $St_1 \triangleright St_2$ which extends $St_1 \triangleright St_2$ to the case where $St_1$ is more general than $St_2$ w.r.t. variable instantiation.

**Definition 5 ($P$-subsumption relation).** Given a topmost rewrite theory $R_P = (\Sigma_P, E_P, R_P)$ representing protocol $P$ and two non-initial states $St_1, St_2$, we write $St_1 \triangleright St_2$ (or $St_2 \triangleright St_1$) and say that $St_2$ is $P$-subsumed by $St_1$ if there is a substitution $\theta$ s.t. $\theta(St_1) \triangleright St_2$.

Note that we restrict the relation $\triangleright$ to non-initial states because, otherwise, an initial state will imply any other state, erroneously making the search space finite after an initial state has been found.

The following results provide the appropriate connection between $P$-subsumption and narrowing transitions. First, we consider the simplest case where, given two non-initial states $St_1, St_2$ such that $St_1 \triangleright St_2$, a narrowing step on $St_2$, yielding state $St'_2$, does not affect the transition subsumption property $\triangleright$ and thus $St_1 \triangleright St'_2$. The proof is straightforward.

**Lemma 1.** Given a topmost rewrite theory $R_P = (\Sigma_P, E_P, R_P)$ representing protocol $P$ and two non-initial states $St_1, St_2$. If (i) there is a substitution $\theta$ s.t. $\theta(St_1) \triangleright St_2$, i.e., $St_1 \triangleright St_2$, (ii) there is a narrowing step $St_2 \triangleright_{\sigma_2, R_P^{-1}, E_P} St'_2$, (iii) each intruder fact of the form $t \in \mathcal{I}$ in $\sigma_2(\theta(St_1))$ appears in $St'_2$ (modulo $E_P$) and (iv) each non-initial strand in $\sigma_2(\theta(St_1))$ appears in $St'_2$ (modulo $E_P$), then $\sigma_2(\theta(St_1)) \triangleright St'_2$, i.e., $St_1 \triangleright St'_2$.

Second, we consider what happens when, given two non-initial states $St_1, St_2$ such that $St_1 \triangleright St_2$, a narrowing step on $St_2$, yielding state $St'_2$, does affect the transition subsumption property $\triangleright$ and thus $St_1 \triangleright St'_2$. The proof is straightforward.

**Lemma 2.** Given a topmost rewrite theory $R_P = (\Sigma_P, E_P, R_P)$ representing protocol $P$ and two non-initial states $St_1, St_2$. If (i) there is a substitution $\theta$ s.t. $\theta(St_1) \triangleright St_2$, i.e., $St_1 \triangleright St_2$, (ii) there is a narrowing step $St_2 \triangleright_{\sigma_2, R_P^{-1}, E_P} St'_2$, and (iii) $\sigma_2(\theta(St_1)) \not\triangleright St'_2$, i.e., $St_1 \not\triangleright St'_2$, then either (a) there is an intruder fact of the form $t \in \mathcal{I}$ in $\sigma_2(\theta(St_1))$ that does not appear in $St'_2$ (modulo $E_P$), or (b) there is a non-initial strand in $\sigma_2(\theta(St_1))$ that does not appear in $St'_2$ (modulo $E_P$).
Now, we can consider both cases of Lemma 2 separately: either an expression \( t \in \mathcal{I} \) in \( \text{St}'_2 \) or a non-initial strand in \( \text{St}'_2 \), not appearing in the instantiated version of \( \text{St}_1 \). First, the case where an expression \( t \in \mathcal{I} \) in \( \text{St}'_2 \) does not appear in the instantiated version of \( \text{St}_1 \).

**Lemma 3.** Given a topmost rewrite theory \( R_P = (\Sigma_P, E_P, R_P) \) representing protocol \( P \) and two non-initial states \( \text{St}_1, \text{St}_2 \). If (i) there is a substitution \( \theta \) s.t. \( \theta(\text{St}_1) \triangleright \text{St}_2 \), i.e., \( \text{St}_1 \triangleright \text{St}'_2 \), (ii) there is a narrowing step \( \text{St}_2 \xrightarrow{\sigma_2,R_{\rho^{-1}},E_P} \text{St}'_2 \), and (iii) there is an intruder fact of the form \( t \in \mathcal{I} \) in \( \sigma_2(\theta(\text{St}_1)) \) that does not appear in \( \text{St}'_2 \) (modulo \( E_P \)), then (a) \( t \in \mathcal{I} \) does not appear in \( \text{St}'_2 \) (modulo \( E_P \)) and (b) there is a state \( \text{St}'_1 \) and a substitution \( \sigma_1 \) such that \( \text{St}_1 \xrightarrow{\sigma_1,R_{\rho^{-1}},E_P} \text{St}'_1 \) and either \( \text{St}'_1 \) is an initial state or there is a substitution \( \rho \) s.t. \( \rho(\text{St}_1) \triangleright \text{St}'_2 \), i.e., \( \text{St}'_1 \triangleright \text{St}'_2 \).

**Proof.** We prove the result by considering the different rules applicable to \( \text{St}_2 \) (remember that in \( R \), rewriting and narrowing steps always happen at the top position). Note that property (a) is immediate because rules in \( R_P \) do not remove expressions of the form \( m \in \mathcal{I} \). Note also that if \( t \in \mathcal{I} \) does appear in \( \text{St}_2 \) (modulo \( E_P \)) and \( t \notin \mathcal{I} \) does appear in \( \text{St}'_2 \) (modulo \( E_P \)), then only Rule (3) or rules of type (4) have been applied to \( \text{St}_2 \) as follows:

- Reversed version of Rule (3), i.e., \( \text{St}_2 \xrightarrow{\sigma_2,R_{\rho^{-1}},E_P} \text{St}'_2 \) using the following rule

\[
[L, M^+ | L'] \& SS \& (M \in \mathcal{I}, IK) \rightarrow [L, M^+, L'] \& SS \& (M \notin \mathcal{I}, IK).
\]

Recall that there is an intruder fact in \( \sigma_2(\theta(\text{St}_1)) \) of the form \( t \in \mathcal{I} \) for \( t \) a message term that does not appear in \( \text{St}'_2 \) (modulo \( E_P \)) and \( t = E_P \sigma_2(M) \). Thus, \( \sigma_2(M) \in \mathcal{I} \) does appear in \( \sigma_2(\theta(\text{St}_1)) \) (modulo \( E_P \)). Here we have several cases:

- If the strand \( \sigma_2([L, M^+ | L']) \) appears in \( \sigma_2(\theta(\text{St}_1)) \), then the very same narrowing step can be performed on \( \text{St}_1 \), i.e., there exist \( \sigma_1, \rho \) such that \( \text{St}_1 \xrightarrow{\sigma_1,R_{\rho^{-1}},E_P} \text{St}'_1 \) with the same rule and \( \sigma \circ \sigma_2 = E_P \sigma_1 \circ \rho \). Thus, either \( \text{St}'_1 \) is an initial state or \( \rho(\text{St}'_1) \triangleright \text{St}'_2 \), since:

  (i) each positive intruder fact in \( \sigma_2(\theta(\text{St}_1)) \) of the form \( u \in \mathcal{I} \) for \( u \) a message term, except \( \sigma_2(M) \in \mathcal{I} \), appears in \( \rho(\text{St}'_1) \) (modulo \( E_P \)), (ii) \( \sigma_2(M) \notin \mathcal{I} \) appears in \( \rho(\text{St}'_1) \) (modulo \( E_P \)), (iii) each non-initial strand in \( \sigma_2(\theta(\text{St}_1)) \), except \( \sigma_2([L, M^+ | L']) \), has not been modified and appears in \( \rho(\text{St}'_1) \) as well (modulo \( E_P \)), and (iv) for \( \sigma_2([L, M^+ | L']) \) in \( \sigma_2(\theta(\text{St}_1)) \), \( \rho'([L, M^+, L']) \) appears in \( \rho(\text{St}'_1) \) and in \( \text{St}_2 \).

- If the strand \( \sigma_2([Lm,M^+ | L']) \) does not appear in \( \sigma_2(\theta(\text{St}_1)) \), then the strand \( \sigma_2([L, M^+ | L']) \) corresponds to a strand \( S_P \) in the protocol specification that had been introduced via a rule of the set (4), where the strand’s bar was clearly more to the right than in \( \sigma_2([L, M^+ | L']) \). Note that it cannot correspond to a strand included
originally in the attack pattern, because we assume that \( S_{t1} \) and \( S_{t2} \) are states generated by backwards narrowing from the same attack state and then both \( S_{t1} \) and \( S_{t2} \) should have the strand. Therefore, since the strand \( \sigma_2(\{L, M^+ \mid L'\}) \) corresponds to a strand in \( S_P \) and the set \( \{4\} \) contains a rewrite rule for each strand of the form \([l_1, u^+, l_2]\) in \( S_P \), there must be a rule \( \alpha \) in \( \{4\} \) introducing a strand of the form \([l_1, u^+, l_2]\) and there must be substitutions \( \sigma_1, \rho \) such that \( S_{t1} \sim_{\sigma_1, \rho, E_P} S'_{t1} \) using the rule \( \alpha \) and \( \theta \circ \sigma_2 = E_P \circ \sigma_1 \circ \rho \). Thus, either \( S'_{t1} \) is an initial state or \( \rho(S'_{t1}) \triangleright S'_{t2} \), since: (i) each positive intruder fact in \( \sigma_2(\theta(S_{t1})) \) of the form \( u \in \mathcal{I} \) for \( u \) a message term, except \( \sigma_2(M) \notin \mathcal{I} \), appears in \( \rho(S'_{t1}) \) (modulo \( E_P \)), (ii) \( \sigma_2(M) \notin \mathcal{I} \) appears in \( \rho(S'_{t1}) \) (modulo \( E_P \)), (iii) each non-initial strand in \( \sigma_2(\theta(S_{t1})) \) has not been modified and appears in \( \rho(S'_{t1}) \) as well (modulo \( E_P \)), and (iv) \( \sigma_2([l_1 \mid u^+, l_2]) \) appears in \( \rho(S'_{t1}) \) and in \( S'_{t2} \).

- Rules in \( \{4\} \), i.e., \( S_{t2} \sim_{\sigma_2, \rho^{-1}, E_P} S'_{t2} \) using a rule of the form

\[
\{SS \& (u \in \mathcal{I}, IK) \rightarrow [l_1 \mid u^+, l_2] \& SS \& (u \notin \mathcal{I}, IK) \mid [l_1, u^+, l_2] \in \mathcal{P}\}.
\]

Recall that there is an intruder fact in \( \sigma_2(\theta(S_{t1})) \) of the form \( \mathcal{E} \in \mathcal{I} \) for \( \mathcal{E} \) a message term that does not appear in \( S'_{t2} \) (modulo \( E_P \)) and \( t = E_P \circ \sigma_2(u) \), where \( u \) is the message term used by the rewrite rule. Thus, \( \sigma_2(u) \notin \mathcal{I} \) does appear in \( \sigma_2(\theta(S_{t1})) \) (modulo \( E_P \)). That is, the same narrowing step is available from \( \sigma_2(\theta(S_{t1})) \) and there exist \( \sigma_1, \rho \) such that \( S_{t1} \sim_{\sigma_1, \rho, E_P} S'_{t1} \) with the same rule and \( \theta \circ \sigma_2 = E_P \circ \sigma_1 \circ \rho \). Thus, either \( S'_{t1} \) is an initial state or \( \rho(S'_{t1}) \triangleright S'_{t2} \).

This concludes the proof.

Second, the case where a non-initial strand in \( S'_{t2} \) does not appear in the instantiated version of \( S_{t1} \).

**Lemma 4.** Given a topmost rewrite theory \( R_P = (\Sigma_P, E_P, R_P) \) representing protocol \( \mathcal{P} \) and two non-initial states \( S_{t1}, S_{t2} \). If (i) there is a substitution \( \theta \) s.t. \( \theta(S_{t1}) \triangleright S_{t2} \), (ii) there is a narrowing step \( S_{t2} \sim_{\sigma_2, \rho^{-1}, E_P} S'_{t2} \), and (iii) there is a non-initial strand \([m_1^+, \ldots, m_i^+ \mid m_{i+1}^+, \ldots, m_{i+s}^+]\) in \( \sigma_2(\theta(S_{t1})) \) that does not appear in \( S'_{t2} \) (modulo \( E_P \)), then (a) \( \sigma_2|_{\text{Var}(S_{t2})} = \text{id} \), (b) \([m_1^+, \ldots, m_{i+s}^+] \mid m_{i+1}^+, \ldots, m_{i+s}^+]\) does appear in \( S'_{t2} \) (modulo \( E_P \)) and (c) there is a state \( S'_{t1} \) such that \( S_{t1} \sim_{\text{id}, \rho^{-1}, E_P} S'_{t1} \) and either \( S'_{t1} \) is an initial state or \( S'_{t1} \triangleright S'_{t2} \).

**Proof.** We prove the result by considering the different rules applicable to \( S_{t2} \) (remember that in \( R \), rewriting and narrowing steps always happen at the top position). Note that property (a) is immediate because rules in \( R_P \) do not remove strands, only move the vertical bar to the left of the sequences of messages in the strands. Note also that if \([m_1^+, \ldots, m_i^+ \mid m_{i+1}^+, \ldots, m_{i+s}^+]\) appears in \( \sigma_2(\theta(S_{t1})) \) and \([m_1^+, \ldots, m_{i+s}^+] \mid m_{i+1}^+, \ldots, m_{i+s}^+]\) appears in \( S'_{t2} \), then only Rule \( \{2\} \) or Rule \( \{5\} \) have been applied to \( S_{t2} \) as follows:
• Reversed version of Rule (2), i.e., \( S_t \sim^{\sigma_2, R_p^{-1}, E_p} S_{t'} \) using the following rule
\[
[L, M^+ \mid L'] \& SS \& IK \rightarrow [L \mid M^+, L'] \& SS \& IK.
\]
• Reversed version of Rule (5), i.e., \( S_t \sim^{\sigma_2, R_p^{-1}, E_p} S_{t'} \) using the following rule
\[
[L, M^- \mid L'] \& SS \& IK \rightarrow [L \mid M^-, L'] \& SS \& (M \in \mathcal{I}, IK).
\]

However, note that \( \sigma_2 |_{\text{var}(S_{t_2})} = id \) in both possible rewrite steps. Then, there is a state \( S_{t_1'} \) such that \( S_{t_1} \sim^{\sigma_{id, R_p^{-1}, E_p}} S_{t_1'} \) with the same rule and it is straightforward that either \( S_{t_1'} \) is an initial state or \( S_{t_1'} \triangleright S_{t_2} \), since only the vertical bar has been moved. \( \square \)

Now we can formally define the relation between \( \mathcal{P} \)-subsumption and one narrowing step. In the following, \( \sim^{(0,1)}_{\sigma, R_p^{-1}, E_p} \) denotes zero or one narrowing steps.

**Lemma 5.** Given a topmost rewrite theory \( \mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_p, R_\mathcal{P}) \) representing protocol \( \mathcal{P} \) and two non-initial states \( S_{t_1}, S_{t_2} \). If \( S_{t_1} \triangleright S_{t_2} \) and \( S_{t_2} \sim^{\sigma_2, R_p^{-1}, E_p} S_{t_2'} \), then there is a state \( S_{t_1}' \) and a substitution \( \sigma_1 \) such that \( S_{t_1} \sim^{(0,1)}_{\sigma_1, R_p^{-1}, E_p} S_{t_1}' \) and either \( S_{t_1}' \) is an initial state or \( S_{t_1}' \triangleright S_{t_2} \).

**Proof.** Since \( S_{t_1} \triangleright S_{t_2} \), there is a substitution \( \theta \) s.t. \( \theta(S_{t_1}) \triangleright S_{t_2} \). If each intruder fact of the form \( t \in \mathcal{I} \) in \( \sigma_2(\theta(S_{t_1})) \) appears in \( S_{t_2'} \) (modulo \( E_p \)) and each non-initial strand in \( \sigma_2(\theta(S_{t_1})) \) appears in \( S_{t_2'} \) (modulo \( E_p \)), then, by Lemma 1, \( \sigma_2(\theta(S_{t_1})) \triangleright S_{t_2'} \), i.e., \( S_{t_1} \triangleright S_{t_2'} \). Otherwise, Lemma 2 states that either (a) there is an intruder fact of the form \( t \in \mathcal{I} \) in \( \sigma_2(\theta(S_{t_1})) \) that does not occur in \( S_{t_2'} \) (modulo \( E_p \)), or (b) there is a non-initial strand in \( \sigma_2(\theta(S_{t_1})) \) that does not occur in \( S_{t_2'} \) (modulo \( E_p \)). For case (a), by Lemma 3, there is a state \( S_{t_1}' \) and a substitution \( \sigma_1 \) such that \( S_{t_1} \sim^{\sigma_1, R_p^{-1}, E_p} S_{t_1}' \) and either \( S_{t_1}' \) is an initial state or there is a substitution \( \rho \) s.t. \( \rho(S_{t_1}') \triangleright S_{t_2'} \). For case (b), by Lemma 4, \( \sigma_2 |_{\text{var}(S_{t_2})} = id \), and there is a state \( S_{t_1}' \) such that \( S_{t_1} \sim^{\sigma_{id, R_p^{-1}, E_p}} S_{t_1}' \) and either \( S_{t_1}' \) is an initial state or \( S_{t_1}' \triangleright S_{t_2'} \), i.e., \( S_{t_1}' \triangleright S_{t_2'} \). \( \square \)

Preservation of reachability follows from the following main theorem. Note that the relation \( \triangleright \) is applicable only to non-initial states, whereas the relation \( \subseteq_{E_p} \) of Definition 3 is applicable to both initial and non-initial states.

**Theorem 3.** Given a topmost rewrite theory \( \mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_p, R_\mathcal{P}) \) representing protocol \( \mathcal{P} \) and two states \( S_{t_1}, S_{t_2} \). If \( S_{t_1} \triangleright S_{t_2}, S_{t_2}^{ini} \) is an initial state, and \( S_{t_2} \sim^{(0,1)}_{\sigma_2, R_p^{-1}, E_p} S_{t_2}^{ini} \), then there is an initial state \( S_{t_1}^{ini} \) and substitutions \( \sigma_1 \) and \( \theta \) such that \( S_{t_1} \sim^{*}_{\sigma_1, R_p^{-1}, E_p} S_{t_1}^{ini} \), \( \theta(S_{t_1}^{ini}) \subseteq_{E_p} S_{t_2}^{ini} \).
Proof. Consider $St_2 = U_0$, $St_2^{ini} = U_n$, $\sigma_2 = \rho_1 \cdots \rho_n$, and $U_0 \sim_{\rho_1, R_{\rho_1}^{-1}, E_P}^n U_n$. Note that $n \neq 0$, since $St_2$ cannot be an initial state because $St_1 \triangleright St_2$ implies that both $St_1$ and $St_2$ are not initial states. Then, by Lemma 5, there is $j \leq n$ such that for each $i < j$, $U_{i-1} \sim_{\rho_i, R_{\rho_i}^{-1}, E_P} U_i$ and there is a step $U_{i-1}^{j} \sim_{\rho_i, R_{\rho_i}^{-1}, E_P} U_j^{j}$ s.t. $U_j^{j} \triangleright U_i$. Note that $U_j^{j}$ is an initial state and there is a substitution $\theta$ s.t. $\theta(U_j^{j}) \subseteq_{E_P} U_j \subseteq_{E_P} U_n$. \hfill $\Box$

This POR technique is used as follows: we keep all the states of the backwards narrowing-based tree and compare each new leaf node of the tree with all the previous states in the tree. If a leaf node is $\mathcal{P}$-subsumed by a previously generated node in the tree, we discard such leaf node.

4.7. The Super-Lazy Intruder

Sometimes terms appear in the intruder's knowledge that are trivially learnable by the intruder. These include terms initially available to the intruder (such as names) and variables. In the case of variables, specially, the intruder can substitute any arbitrary term of the same sort as the variable, and so there is no need to try to determine all the ways in which the intruder can do this. For this reason it is safe, at least temporarily, to drop these terms from the state. We will refer to those terms as (super) lazy intruder terms.

Example 8. Consider again the attack pattern $(\dag)$ in Example 3. After a couple of backwards narrowing steps, the Maude-NPA finds the following state that considers how the intruder can learn $\sec(a, r^{''})$ by assuming he can learn a message $e(K, \sec(a, r^{''}))$ and the key $K$:

\[
\begin{align*}
[\ \text{nil} \mid K^-, e(K, \sec(a, r^{''}))^-, \sec(a, r^{''})^+] \& \\
\triangleright: r' : \\
\mid (A; B; Y)^-, (B; A; \exp(g, n(B, r')))^+ \mid (e(\exp(Y, n(B, r'))), \sec(a, r^{''}))^- \& \\
(e(\exp(Y, n(B, r')), \sec(a, r^{''})) \in \mathcal{I}, K \in \mathcal{I}, e(K, \sec(a, r^{''})) \in \mathcal{I}, \sec(a, r^{''}) \notin \mathcal{I})
\end{align*}
\]\n
Here variable $K$ is a super-lazy term and the tool wouldn’t search for values. The problem, of course, is that later on in the search the variable $K$ may become instantiated, in which case the term then becomes relevant to the search. Indeed, after some more backwards narrowing steps, the tool tries to unify message $e(K, \sec(a, r^{''}))$ with an output message $e(\exp(X, n(A, r)), \sec(A, r_2))$ of an explicitly added Bob’s strand of the form

\[
\triangleright: r_1, r_2 : \\
\mid (\mathcal{A}; B; \exp(g, n(A, r_1)))^+, (B; A; X)^-, (e(\exp(X, n(A, r)), \sec(A, r_2)))^+
\]\n
\footnote{This, of course, is subject to the assumption that the intruder can produce at least one term of that sort. But since the intruder is assumed to have access to the network and to all the operations available to an honest principal, this is a reasonable restriction to make.}
thus getting an instantiation for the super-lazy term $K$, namely 
\{K \mapsto \exp(X, n(A, r))\}.

Note that the tool might continue searching for an initial state when a super 
lazy term is properly instantiated, and this would not cause the tool to prove 
an insecure protocol to be secure. However, it would lead to an unacceptably 
large number of false attacks because the contents of variable $K$ are expected 
to be learned by the intruder too.

We take an approach similar to that of the lazy intruder of Basin et al. [1] 
and extend it to a more general case, that we call super-lazy terms. We note 
that this use of what we here call the super-lazy intruder was also present in 
the original NPA.

The set $L(St)$ of super-lazy terms w.r.t. a state $St$ is inductively generated 
as a subset $L(St) \subseteq T_0(Y \cup IK_0)$ where $IK_0$ is the basic set of terms known 
by the intruder at the beginning of a protocol execution, $Y$ is a subset of the 
variables of $St$, and $\Omega$ is the set of operations available to the intruder. The idea 
of super-lazy terms is that we also want to exclude from $L(St)$ the set $IK^\notin(St)$ 
of terms that the intruder does not know and all its possible combinations with 
symbols in $\Omega$.

**Definition 6 (Super-lazy terms).** Let $R_P = (\Sigma_P, E_P, R_P)$ be a topmost 
rewrite theory representing protocol $P$. Let $IK_0$ be the basic set of terms known 
by the intruder at the beginning of a protocol execution, defined as $IK_0 = \{t' \mid [t'] \in S_P, t' =_{E_P} t\}$. Let $\Omega$ be the set of operations available 
to the intruder, defined as

$$\Omega = \{f : s_1 \cdots s_n \rightarrow s \mid [(X_1:s_1)^-, \ldots, (X_k:s_k)^-, (f(X_1:s_1, \ldots, X_k:s_k))^+] \in S_P\}.$$ 

Let $St$ be a state (with logical variables). Let $IK^\notin(St)$ be the set of terms that 
the intruder does not know at state $St$, defined as $IK^\notin(St) = \{m' \mid (m^\notin) \in St, m' =_{E_P} m\}$. The set $L(St)$ of super-lazy terms w.r.t. $St$ (or simply super-lazy terms) is defined as

1. $IK_0 \subseteq L(St)$,
2. $\text{Var}(St) - IK^\notin(St) \subseteq L(St)$,
3. for each $f : s_1 \cdots s_n \rightarrow s \in \Omega$ and for all $t_1:s_1, \ldots, t_k:s_k \in L(St)$, if 
   $f(t_1:s_1, \ldots, t_k:s_k) \notin IK^\notin(St)$, then $f(t_1:s_1, \ldots, t_k:s_k) \in L(St)$.

The idea behind the super-lazy intruder is that, given a term made out of lazy 
intruder terms, such as “$a; e(K, Y)$”, where $a$ is a public name and $K$ and $Y$ are 
variables, the term “$a; e(K, Y)$” is also a (super) lazy intruder term by applying 
the operations $e$ and $\cdots$.

Let us first briefly explain how the (super) lazy intruder mechanism works 
before formally describing it. A ghost state is a state extended to allow expres-
sions of the form $\text{ghost}(m)$ in the intruder’s knowledge, where $m$ is a super-lazy 
term. When, during the backwards reachability analysis, we detect a state $St$ 
having a super lazy term $t$ in an expression $t\in I$ in the intruder’s knowledge, we
replace the intruder fact $t \in \mathcal{I}$ in $St$ by $\text{ghost}(t)$ and keep the ghost version of $St$ in the history of states used by the transition subsumption of Section 4.6. For instance, the state $(t)$ of Example 8 with a super-lazy intruder term $K$ would be represented as follows, where we have just replaced $K \in \mathcal{I}$ by $\text{ghost}(K)$:

\[
[\text{nil} \mid K^-, \; e(K, \sec(a, r'''))^-] \&
:: r' :: [\; (A; B; Y)^-, (B; A; \exp(g, n(B, r'))) + \; | \; (e(\exp(Y, n(B, r'))), \sec(a, r''')) - \; ] \&
(\text{ghost}(K), \; e(\exp(Y, n(B, r'))), \sec(a, r''')) \in \mathcal{I}, \; e(K, \sec(a, r''')) \in \mathcal{I}, \; \sec(a, r''') \notin \mathcal{I})
\]

If later in the search tree we detect a ghost state $St'$ containing an expression $\text{ghost}(t)$ such that $t$ is no longer a super lazy intruder term, then there is a state $St$ with an expression $\text{ghost}(u)$ that precedes $St'$ in the narrowing tree such that the message $u$ has been instantiated to $t$ in an appropriate way and we must reactivate such original state $St$. That is, we “roll back” and replace the current state $St'$, containing expression $\text{ghost}(t)$, by an instantiated version of state $St$, namely $\theta(St)$, where $t =_{E_\theta} \theta(u)$. This is explained in detail in Definition 11 below.

However, if the substitution $\theta$ binding variables in $u$ includes variables of sort Fresh, we have to keep them in the reactivated version of $St$, since they are unique in our model. Therefore, the strands indexed by these fresh variables must also be included in the “rolled back” state, even if they were not there originally. Moreover, they must have the bar at the place where it was when the strands were originally introduced. We show below how this is accomplished. Furthermore, if any of the strands thus introduced have other variables of sort Fresh as subterms, then the strands indexed by those variables must be included too, and so on. That is, when a state $St'$ properly instantiating a ghost expression $\text{ghost}(t)$ is found, the procedure of rolling back to the original state $St$ that gave rise to that ghost expression implies not only applying the bindings for the variables of $t$ to $St$, but also introducing in $St$ all the strands from $St'$ that produced fresh variables and that either appear in the variables of $t$ or are recursively connected with them.

**Example 9.** For instance, after the tool finds an instantiation for variable $K$, the tool rolls back to the state originating the super-lazy term $K$ as follows, where we have copied the explicitly added Bob’s strand with the vertical bar at the rightmost position because it is the strand generating the Fresh variable $r'''$:

\[
[\text{nil} \mid \exp(X, n(a, r))^-] \&
:: r, r'' :: [\; (a; B' ; \exp(g, n(a, r)) + \; | \; (B'; a; X)^- \; | \; (e(\exp(X, n(a, r)), \sec(a, r''')) + \; | \; \text{nil} \; ] \&
:: r' :: [\; (A; B; Y)^-, (B; A; \exp(g, n(B, r'))) + \; | \; (e(\exp(Y, n(B, r'))), \sec(a, r''')) - \; ] \&
(e(\exp(Y, n(B, r'))), \sec(a, r''')) \in \mathcal{I}, \; (e(\exp(X, n(a, r))) \in \mathcal{I}, \; \exp(X, n(a, r)) \in \mathcal{I}, \; \sec(a, r''') \notin \mathcal{I})
\]

25
In order for the super-lazy intruder mechanism to be able to tell where the bar was when a strand was introduced, we must modify the set of rules of type [4] introducing new strands:

\[
\{ \{ l_1 | u^+ \} \land \{ u \in I, K \} \rightarrow \{ u \in I, K \} \mid \{ l_1, u^+ \}, l_2 \in S_P \}\}
\]

Note that rules of type (4) introduce strands \([l_1 | u^+], l_2]\), whereas here rules of type (6) introduce strands \([l_1 | u^+]\). This slight modification makes it possible to safely move the position of the bar back to the place where the strand was introduced. However, now the strands added may be partial, since the whole sequence of actions performed by the principal is not directly recorded in the strand. Therefore, the set of rewrite rules used by narrowing in reverse are now \(\tilde{R}_P = \{[4], [6] \} \cup [4]\).

First, we define a new relation \(\sqsubseteq_{E_P}\) between states, which is similar to \(\subseteq_{E_P}\) of Definition 3 but considers partial strands.

**Definition 7 (Partial Inclusion).** Given two states \(St_1, St_2\), we abuse notation and write \(St_1 \sqsubseteq_{E_P} St_2\) to denote that every intruder fact in \(St_1\) appears in \(St_2\) (modulo \(E_P\)) and that every strand \([m^\pm_1, \ldots, m^\pm_k]\) in \(St_1\), either appears in \(St_2\) (modulo \(E_P\)) or there is \(i \in \{1, \ldots, k\}\) s.t. \(m^\pm_i = m^\pm_i + i\) and \([m^\pm_1, \ldots, m^\pm_i]\) appears in \(St_2\) (modulo \(E_P\)).

The following result ensures that if a state is reachable via backwards reachability analysis using \(R_P\), then it is also reachable using \(\tilde{R}_P\). Its proof is straightforward.

**Proposition 2.** Let \(R_P = (\Sigma_P, E_P, R_P)\) be a topmost rewrite theory representing protocol \(P\). Let \(St = ss \land SS \land (ik, IK)\) where \(ss\) is a term representing a set of strands, \(ik\) is a term representing a set of intruder facts, \(SS\) is a variable for strands, and \(IK\) is a variable for intruder knowledge. If there is an initial state \(St_{ini}\) and a substitution \(\sigma\) such that \(St \sim^*_{\sigma, R_P^{-1}, E_P} St_{ini}\), then there is an initial state \(St'_{ini}\) and two substitutions \(\sigma', \rho\) such that \(St \sim^*_{\sigma', \tilde{R}_P^{-1}, E_P} St'_{ini}\), \(\sigma = E_P \sigma' \circ \rho\), and \(\rho(St'_{ini}) \sqsubseteq_{E_P} St_{ini}\).

Now, we describe how to resuscitate a state. First, we formally define a ghost state.

**Definition 8 (Ghost State).** Given a topmost rewrite theory \(R_P = (\Sigma_P, E_P, R_P)\) representing protocol \(P\) and a state \(St\) containing an intruder fact \(t \in I\) such that \(t\) is a super-lazy term, we define the ghost version of \(St\), written \(\overline{St}\), by replacing \(t\) in \(St\) by \(\text{ghost}(t)\) in \(\overline{St}\).

Now, in order to resuscitate a state, we need to formally compute the strands that are generating \textit{Fresh} variables relevant to the instantiation found for the super-lazy term.

**Definition 9 (Strand Reset).** Given a strand \(s\) of the form \(:: r_1, \ldots, r_k :: [m^\pm_1, \ldots, m^\pm_n]\), when we want to move the bar to the rightmost position (denoting a final strand), we write \(s \gg =:: r_1, \ldots, r_k :: [m^\pm_1, \ldots, m^\pm_n] \mid \text{nil}\).
Definition 10 (Fresh Generating Strands). Given a state \( S_t \) containing an intruder fact \( \text{ghost}(t) \) for some term \( t \) with variables, we define the set of strands associated to \( t \), denoted \( \text{strands}_S(t) \), as follows:

- for each strand \( s \) in \( S_t \) of the form :: \( r_1, \ldots, r_k :: [m_1^+, \ldots, | \ldots, m_n^+] \), if there is \( i \in \{1, \ldots, k\} \) s.t. \( r_i \in \text{Var}(t) \), then \( s \) is included into \( \text{strands}_S(t) \); or

- for each strand \( s \) in \( S_t \) of the form :: \( r_1, \ldots, r_k :: [m_1^+, \ldots, | \ldots, m_n^+] \), if there is another strand \( s' \) of the form :: \( r'_1, \ldots, r'_k :: [w_1^+, \ldots, | \ldots, w_n^+] \) in \( \text{strands}_S(t) \), and there are \( i \in \{1, \ldots, k\} \) and \( j \in \{1, \ldots, n\} \) s.t. \( r_i \in \text{Var}(w_j) \), then \( s \) is included into \( \text{strands}_S(t) \).

Now, we formally define how to resuscitate a state.

Definition 11 (Resuscitation). Given a topmost rewrite theory \( R_P = (\Sigma_P, E_P, \tilde{R}_P) \) representing protocol \( P \) and a state \( S_t \) containing an intruder fact \( t \in I \) such that \( t \) is a super-lazy term, i.e., \( S_t = ss \&(t \in I, ik) \) where \( ss \) is a term denoting a set of strands and \( ik \) is a term denoting the rest of the intruder knowledge. Let \( \tilde{S}_t \) be the ghost version of \( S_t \). Let \( S'_t \) be a state such that \( \tilde{S}_t \sim_{\tau, \tilde{R}_P^{-1}, E_P}^* S'_t \) and \( \tau(t) \) is not a super-lazy term. Let \( \sigma_t = \sigma|_{\text{Var}(t)} \). The resuscitated (or resuscitated) version of \( S_t \) w.r.t. state \( S'_t \) and substitution \( \sigma_t \) is defined as \( \tilde{S}_t = \sigma_t(ss) \& \sigma_t(ik) \& \text{strands}_{S'_t}(\sigma_t) \).

Let us now prove the completeness of this state space reduction technique.

Theorem 4. Given a topmost rewrite theory \( R_P = (\Sigma_P, E_P, \tilde{R}_P) \) representing protocol \( P \) and a state \( S_t \) containing an intruder fact \( t \in I \) such that \( t \) is a super-lazy term, if there exist an initial state \( S_{ini} \) and substitution \( \theta \) such that \( S_t \sim_{\theta, \tilde{R}_P^{-1}, E_P}^* S_{ini} \), then (i) there exist a state \( S'_t \) and substitutions \( \tau, \tau' \) such that \( S_t \sim_{\tau, \tilde{R}_P^{-1}, E_P}^* S'_t \), \( \theta = E_P \tau' \circ \tau \), and \( \tau(t) \) is not a super-lazy term, and (ii) there exist a resuscitated version \( \tilde{S}_t \) of \( S_t \) w.r.t. state \( S_t \) and \( \tau, \) an initial state \( S_{ini}' \), and substitutions \( \theta', \rho \) such that \( \tilde{S}_t \sim_{\theta', \tilde{R}_P^{-1}, E_P}^* S_{ini}' \), \( \theta = E_P \theta' \circ \rho \), and \( \rho(S_{ini}') \subseteq E_P S_{ini} \).

Proof. The sequence from \( S_t \) to \( S_{ini} \) can be decomposed into two fragments, computing substitutions \( \tau, \tau' \), respectively, such that \( \tau \) is the smallest part of \( \theta \) that makes \( \tau(t) \) not a super-lazy term. That is, there is a state \( S'_t \) and substitutions \( \tau, \tau' \) such that \( \tau(t) \) is not a super-lazy term, \( \theta = \tau \circ \tau' \), \( S_t \sim_{\tau, \tilde{R}_P^{-1}, E_P}^* S'_t \), and the sequence \( \tilde{S}_t \sim_{\theta', \tilde{R}_P^{-1}, E_P}^* S'_t \) can be viewed as \( S_t = S_{t_0} \sim_{\tau_i, \tilde{R}_P^{-1}, E_P} \cdots \sim_{\tau_k, \tilde{R}_P^{-1}, E_P} S_{t_k} = S'_t \) such that for all \( i \in \{1, \ldots, k - 1\} \), \( \tau_i(t) \) is a super-lazy term. However, using the completeness results of narrowing, Theorem 1 there must be a narrowing sequence from \( \tilde{S}_t \) computing such substitution \( \tau \). That is, there is a state \( S''_t \) such that \( \tilde{S}_t \sim_{\tau, \tilde{R}_P^{-1}, E_P}^* S''_t \) and \( S''_t \) differs from \( S'_t \) (modulo \( E_P \)-equivalence and variable renaming) only in that \( \tau(t) \in I \) is replaced by \( \text{ghost}(\tau(t)) \). Let \( \tau_i = \tau|_{\text{Var}(t)} \),
there exists a substitution $\tau''$ s.t. $\tau =_{E_P} \tau_t \circ \tau''$. Let $\tilde{St}$ be the resuscitated version of $St$ w.r.t. state $St''$ and substitution $\tau_t$. Then, by narrowing completeness, i.e., Theorem 1, there exist a state $St'_{ini}$ and substitutions $\sigma, \rho$ such that $\tilde{St} \overset{\sigma, \rho}{\sim}^{* \sim}_E P \setminus _{E_P} St'_{ini}$, $\tau'' =_{E_P} \sigma \circ \rho$, and $\rho(St'_{ini}) =_{E_P} St_{ini}$. $\square$

4.7.1. Improving the Super-Lazy Intruder.

When we detect a state $St$ with a super lazy term $t$, we may want to analyze whether the variables of $t$ may be eventually instantiated or not before creating a ghost state. The following definition provides the key idea.

Definition 12 (Void Super-Lazy Term). Given a topmost rewrite theory $R_P = (\Sigma_P, E_P, R_P)$ representing protocol $P$, and a state $St$ containing an intruder fact $t \in I$ such that $t$ is a super-lazy term, if for each strand $[m_1^\pm, \ldots, m_{j-1}^\pm, m_j, \ldots, m_k^\pm]$ in $St$ and each $i \in \{1, \ldots, j - 1\}$, $\text{Var}(t) \cap \text{Var}(m_i) = \emptyset$, and for each term $w \in I$ in the intruder’s knowledge, $\text{Var}(t) \cap \text{Var}(w) = \emptyset$, then $t$ is called a void super-lazy term.

Proposition 3. Given a topmost rewrite theory $R_P = (\Sigma_P, E_P, R_P)$ representing protocol $P$ and a state $St$ containing an intruder fact $t \in I$ such that $t$ is a void super-lazy term, let $\tilde{St}$ be the ghost version of $St$ w.r.t. the void super-lazy term $t$. If there exist an initial state $St_{ini}$ and a substitution $\theta$ such that $St \overset{\theta, R_P}{\sim}^{* \sim}_E P St_{ini}$, then there exist an initial state $St'_{ini}$ and substitutions $\sigma, \rho$ such that $\tilde{St} \overset{\sigma, \rho}{\sim}^{* \sim}_E P \setminus _{E_P} St'_{ini}$, $\theta =_{E_P} \sigma \circ \rho$, and $\rho(St'_{ini}) \subseteq_{E_P} St_{ini}$.

Proof. Since $t$ is a super-lazy term, $St_{ini}$ contains a sequence of intruder strands of $S_P$ generating $t$. Let $\theta_t = \theta|_{\text{Var}(t)}$, there exists a substitution $\theta'$ s.t. $\theta =_{E_P} \theta_t \circ \theta'$. Since $t$ is a void super-lazy term, there is a state $St''_{ini}$ such that $\theta'(\tilde{St}) \overset{\theta', R_P}{\sim}^{* \sim}_E P \setminus _{E_P} St''_{ini}$. Then, by narrowing completeness, i.e., Theorem 1, there are an initial state $St'_{ini}$ and substitutions $\sigma, \rho$ such that $\tilde{St} \overset{\sigma, \rho}{\sim}^{* \sim}_E P \setminus _{E_P} St'_{ini}$, $\theta =_{E_P} \sigma \circ \rho$, and $\rho(St'_{ini}) \subseteq_{E_P} St''_{ini}$. Finally, $St''_{ini} \subseteq_{E_P} St_{ini}$, since $St_{ini}$ simply has the strands generating $t$ that $St''_{ini}$ does not contain. $\square$

4.7.2. Interaction with Transition Subsumption.

When a ghost state is reactivated, we see from the above definition that such a reactivated state will be $P$-subsumed by the original state that raised the ghost expression. Therefore, the transition subsumption relation $\triangleright$ of Section 4.6 has to be slightly modified to avoid checking a resuscitated state against its predecessor ghost state. Now, let us formally state this problem.

Definition 13 (Resuscitated Child). Given a topmost rewrite theory $R_P = (\Sigma_P, E_P, R_P)$ representing protocol $P$ and two non-initial states $St$ and $St'$ such that $St$ contains an intruder fact $t \in I$ and $t$ is a super-lazy term, we say $St'$ is a resuscitated child of $St$, written $St \curvearrowleft St'$, if:

28
1. given the ghost version $\bar{St}$ of $St$ w.r.t. the super-lazy term $t$, then there exist states $St_1, \ldots, St_k$, substitutions $\tau_1, \ldots, \tau_k$, and $i \in \{1, \ldots, k\}$ such that

$$\bar{St} \leadsto_{\tau_1, R_p^{-1}, E_p} St_1 \cdots St_{i-1} \leadsto_{\tau_i, R_p^{-1}, E_p} St_i \cdots St_{k-1} \leadsto_{\tau_k, R_p^{-1}, E_p} St_k,$$

$\tau_j(t)$ is a super-lazy term for $1 \leq j \leq i - 1$, and $\tau_i(t)$ is not a super-lazy term, and

2. given the reactivated version $\hat{St}$ of $St$ w.r.t. $St_i$ and $\tau = \tau_1 \circ \cdots \circ \tau_i$ and $\tau_i = \tau|_{\text{Var}(t)}$, there exist substitutions $\tau'_1, \ldots, \tau'_k$ such that $\tau_j = \tau_i \circ \tau'_j$ for $1 \leq j \leq k$, states $St'_1, \ldots, St'_k$, and a narrowing sequence

$$\hat{St} \leadsto_{\tau'_1, R_p^{-1}, E_p} St'_1 \cdots St'_{k-1} \leadsto_{\tau'_k, R_p^{-1}, E_p} St'_k,$$

3. then there is $j \in \{1, \ldots, k\}$ such that $St' =_{E_p} St'_j$.

**Proposition 4.** Given a topmost rewrite theory $R_P = (\Sigma_P, E_P, \bar{R}_P)$ representing protocol $P$ and two non-initial states $St$ and $St'$ such that $St$ contains an intruder fact $t \in I$ and $t$ is a super-lazy term, if $St \rhd St'$, then $St \blacktriangleright St'$ and reachability completeness is lost.

**Proof.** Since $\bar{St}$ is similar to $St$ but $t \in I$ has been replaced by $\text{ghost}(t)$, and $\hat{St}$ contains all the strands and positive intruder facts of $St$ but instantiated with $\tau|_{\text{Var}(t)}$, then for the sequences

$$\bar{St} \leadsto_{\tau_1, R_p^{-1}, E_p} St_1 \cdots St_{k-1} \leadsto_{\tau_k, R_p^{-1}, E_p} St_k$$

and

$$\hat{St} \leadsto_{\tau'_1, R_p^{-1}, E_p} St'_1 \cdots St'_{k-1} \leadsto_{\tau'_k, R_p^{-1}, E_p} St'_k$$

we have that $St_j \blacktriangleright St'_j$ for $j \in \{1, \ldots, k\}$, since $St'_j$ contains all the strands and positive intruder facts of $St_j$ but instantiated with $\tau|_{\text{Var}(t)}$. Reachability completeness is lost because if there is an initial state $St_{ini}$ and substitution $\tau'$ such that $St \leadsto^*_{\tau, R_p^{-1}, E_p} St' \leadsto^*_{\tau', R_p^{-1}, E_p} St_{ini}$, then, since $St$ is replaced by $\bar{St}$ during the backwards reachability analysis and later $\bar{St}$ is replaced by $\hat{St}$, when Maude-NPA finds that $St_j \blacktriangleright St'_j$, it removes $St'_j$ from the backwards reachability analysis, (possibly) leaving no successor of $St$ leading to $St_{ini}$. $\square$

The simplest way of ensuring whether or not $St_1 \rhd St_2$ is to examine the relative positions of $St_1$ and $St_2$ in the search tree as well as the narrowing steps between them in the form established by Definition 13. However, for reasons of efficiency, we want to keep examinations of the search tree to a minimum, and restrict ourselves as much as possible to looking at information in the state itself. Thus, we make use of information that is already in the state, the message sequence first mentioned in Section 3.1. We find, that after making minor modifications to this message sequence to take account of resuscitated ghosts, a
simple syntactic check on the sequence can provide a relation that approximates \( \sim \).

In order to formally identify when a resuscitated state must not be erroneously discarded by \( \triangleright \), we extend protocol states to have the actual message exchange sequence between principal or intruder strands and add a new expression \textit{resuscitated}(m) to indicate when a state has been resuscitated. The actual set of rewrite rules extended to compute the exchange sequence is as follows, where \( X \) is a variable denoting an exchange sequence:

\[
\begin{align*}
[L | M^-, L'] \& SS & \& (M \in I, IK) & \& (M^-, X) \rightarrow [L, M^- | L'] \& SS & \& (M \in I, IK) & \& X \\
[L | M^+, L'] \& SS & \& IK & \& (M^+, X) \rightarrow [L, M^+ | L'] \& SS & \& IK & \& X \\
[L | M^+, L'] \& SS & \& (M \notin I, IK) & \& (M^+, X) \rightarrow [L, M^+ | L'] \& SS & \& (M \in I, IK) & \& X
\end{align*}
\]

for each \( [l_1, u^+, l_2] \in S_P : [l_1 | u^+, l_2] \& SS \& (u \in I, IK) \& (u^+, X) \rightarrow SS \& (u \in I, IK) \& X \)

Completeness reachability is obviously preserved for this set of rules and for the obvious extensions to \( \overline{R_P} \) and \( \overline{R_P} \). For instance, the resuscitated state of Example 9 will be written as follows, where the resuscitated message is the first item in the exchange sequence:

\[
\begin{align*}
[\text{nil} | \text{exp}(X, n(a, r)))^- & , e(\text{exp}(X, n(a, r)), \text{sec}(a, r''))^- , \text{sec}(a, r')^+] \& \\
\text{:: } r , r' :: \\
\begin{align*}
& [a; B; \text{exp}(g, n(a, r)))^+ , (B'; a; X)^- , (e(\text{exp}(X, n(a, r)), \text{sec}(a, r''))^+ | \text{nil}] \& \\
\text{:: } r' :: [A; B; Y)^- , (B; A; \text{exp}(g, n(B, r')))^+ | (e(\text{exp}(Y, n(B, r'))), \text{sec}(a, r''))^+] & \text{&} \\
& (e(\text{exp}(Y, n(B, r'))), \text{sec}(a, r'')) \in I , \text{exp}(X, n(a, r)) \in I , \\
& e(\text{exp}(X, n(a, r)), \text{sec}(a, r'')) \in I , \text{sec}(a, r'') \notin I) \& \\
& (\text{resuscitated}(\text{exp}(X, n(a, r))), \text{exp}(X, n(a, r)))^- , e(\text{exp}(X, n(a, r)), \text{sec}(a, r''))^- , \\
& (\text{sec}(a, r''))^+ , (e(\text{exp}(Y, n(h, r'))), \text{sec}(a, r''))^- , (e(\text{exp}(Y, n(h, r'))), \text{sec}(a, r''))^+ , \\
& (e(\text{exp}(Y, n(h, r'))), \text{sec}(a, r''))^-)
\end{align*}
\]

In [6], we provided a very simple rule for approximating Definition 13.

**Definition 14.** Given a topmost rewrite theory \( R_P = (\Sigma_P, E_P, \overline{R_P}) \) representing protocol \( P \) and two non-initial states \( St_1, St_2 \), we write \( St_1 \rightarrow St_2 \) if either \( St_1 \) does not contain an expression \( \text{ghost}(m) \) for a message term \( m \) or \( St_1 \) does contain an expression \( \text{ghost}(m) \) for a message term \( m \) but \( St_2 \) does not contain the expression \( \text{resuscitated}(m) \).

The following result establishes that \( \rightarrow \) is an approximation of \( \sim \). The proof is straightforward.

**Lemma 6.** Given a topmost rewrite theory \( R_P = (\Sigma_P, E_P, \overline{R_P}) \) representing protocol \( P \) and two non-initial states \( St_1, St_2 \), if \( St_1 \sim St_2 \), then \( St_1 \rightarrow St_2 \).

Now, we can provide a better transition subsumption relation.

30
Definition 15 ($\mathcal{P}$-subsumption relation II). Given a topmost rewrite theory $\mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_\mathcal{P}, R_\mathcal{P})$ representing protocol $\mathcal{P}$ and two non-initial states $St_1, St_2$, we write $St_1 \triangleright_{H} St_2$ and say that $St_2$ is $\mathcal{P}$-subsumed by $St_1$ if there is a substitution $\theta$ s.t. $\theta(St_1) \triangleright St_2$ and $\theta(St_1) \not\rightarrow St_2$.

Reachability completeness is straightforward from Lemma 9 and Proposition 4 since $St_1 \not\rightarrow St_2$ implies $St_1 \not\rightarrow St_2$.

Though this method solves the problem, it disables almost completely the transition subsumption for those states after a resuscitation, since $\rightarrow$ is a bad approximation of $\triangleright$. Here, we provide a more concise definition of the interaction between the transition subsumption and the super-lazy intruder reduction techniques.

We characterize those states after a resuscitation that are truly linked to the parent state. First, we identify those states that are directly resuscitated versions of a former state. Intuitively, by comparing the exchange sequences of the two states, we can see whether the exchange sequence of the former is $(L_1, L_2, M_1^-, L_3)$ and it has a ghost expression $\text{ghost}(M_1)$, whereas the exchange sequence of the resuscitated version is $(L_1, \text{resuscitated}(M_1), L_2, M_1^-, L_3)$.

Definition 16. Given a topmost rewrite theory $\mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_\mathcal{P}, R_\mathcal{P})$ representing protocol $\mathcal{P}$ and two non-initial states $St_1, St_2$, we say that $St_2$ is a direct resuscitated version of $St_1$, written $St_1 \rightarrow St_2$, if there are messages $M_1$ and $M_2$ and a substitution $\rho$ such that

1. state $St_1$ has a ghost of the form $\text{ghost}(M_1)$,
2. the exchange sequence of state $St_1$ is of the form $(L_1, L_2, M_1^-, L_3)$
3. the exchange sequence of state $St_2$ is of the form $(L_1', \text{resuscitated}(M_2), L_2', M_2^-, L_3')$,
4. and $\rho(L_1, L_2, M_1^-, L_3) =_{E_\mathcal{P}} (L_1', L_2', M_2^-, L_3')$.

Relation $\rightarrow$ is closer to $\triangleright$.

Lemma 7. Given a topmost rewrite theory $\mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_\mathcal{P}, R_\mathcal{P})$ representing protocol $\mathcal{P}$ and two non-initial states $St_1, St_2$, if $St_1 \rightarrow St_2$, then $St_1 \triangleright St_2$.

However, $St_1 \triangleright St_2$ does not imply $St_1 \rightarrow St_2$ and we have to go even further. Relation $St_1 \rightarrow St_2$ takes into account only whether $St_2$ is a resuscitated version of $St_1$, but does not consider what happens beyond the state that produced the instantiation that reactivated the ghost state. Intuitively, now we compare the exchange sequences of the two states to see whether the exchange sequence of the first is $(L_1, L_2, L_3, M_1^-, L_4)$ and it has a ghost expression $\text{ghost}(M_1)$, whereas the exchange sequence of the second is $(L_1, M_1^+, L_2, \text{resuscitated}(M_1), L_3, M_1^-, L_4)$. Indeed, a recursive definition can be given here that becomes extremely useful when several resuscitations have happened in a concrete state.
Definition 17. Given a topmost rewrite theory $R_P = (\Sigma_P, E_P, \tilde{R}_P)$ representing protocol $P$ and two non-initial states $St_1, St_2$, we say that $St_2$ is a resuscitated version of $St_1$, written $St_1 \rightarrow^+ St_2$, if $St_1 \rightarrow St_2$ or there are messages $M_1$ and $M_2$, a substitution $\rho$, and sequences $L'_1, L''_1$ such that:

1. state $St_1$ has a ghost of the form $\text{ghost}(M_1)$,
2. the exchange sequence of state $St_1$ is of the form
   $$(L_1, L_2, L_3, M_1^-, L_4)$$
3. the exchange sequence of state $St_2$ is of the form
   $$(L'_1, L''_1, M_2^+, L'_2, \text{resuscitated}(M_2), L'_3, M_2^-, L'_4)$$
4. $\rho(L_2, L_3, M_1^-, L_4) =_{E_P} (L'_2, L'_3, M_2^-, L'_4)$
5. $L''_1$ is the longest sequence such that each message $m^\pm$ in $L''_1$ has message $\rho(M_1)$ as a subterm
6. and either
   (a) $\rho(L_1) =_{E_P} L'_1$ or
   (b) $St'_1 \rightarrow^+ St'_2$ where $St'_1$ is $St_1$ without the ghost($M_1$) expression and $St'_2$ is $St_2$ with the shorter exchange sequence $(L'_1, L'_2, L'_3, M_2^-, L'_4)$.

The following result establishes that $\rightarrow^+$ is a better approximation of $\rightarrow$ than $\rightarrow\rightarrow$. The proof is straightforward.

Lemma 8. Given a topmost rewrite theory $R_P = (\Sigma_P, E_P, \tilde{R}_P)$ representing protocol $P$ and two non-initial states $St_1, St_2$, if $St_1 \rightarrow St_2$, then $St_1 \rightarrow^+ St_2$.

Now, we can provide a better transition subsumption relation.

Definition 18 ($P$-subsumption relation III). Given a topmost rewrite theory $R_P = (\Sigma_P, E_P, \tilde{R}_P)$ representing protocol $P$ and two non-initial states $St_1, St_2$, we write $St_1 \triangleright_{III} St_2$ and say that $St_2$ is $P$-subsumed by $St_1$ if there is a substitution $\theta$ s.t. $\theta(St_1) \triangleright St_2$ and $St_1 \not\rightarrow^+ St_2$.

Finally, reachability completeness is straightforward from Lemma 8 and Proposition 4, since $St_1 \not\rightarrow^+ St_2$ implies $St_1 \not\triangleright St_2$.

5. Experimental Evaluation

In Table 1, we summarize the experimental evaluation of the impact of the different state space reduction techniques for various example protocols searching up to depth 4. We measure several numerical values for the techniques: (i) number of states at each backwards narrowing step, and (ii) whether the state space is finite or not. The experiments have been performed on a MacBook with 2 Gb RAM using Maude 2.6. All protocol specifications are included in the official Maude-NPA distribution. The protocols are the following: (i) NSPK, (ii) ...
### Table 1: Number of states for 1, 2, 3, and 4 backwards narrowing steps comparing each optimization of Sections 4.1, 4.4, 4.5, 4.6, and 4.7.

| Protocol | Grammars | Input First | Transition Subsumption | Super-lazy Intruder | All optimizations |
|----------|----------|-------------|------------------------|---------------------|------------------|
| NSPK     | none     | none        | 5 18 95 310 650 83     | 5 18 95 310 641 1   | 5 19 136 642 4021 4 |
| NSL      | NSL      | none        | 5 18 95 310 650 83     | 5 18 95 310 641 2   | 5 19 136 642 4021 4 |
| SecReT06 | SecReT06 | none        | 5 18 95 310 650 83     | 5 18 95 310 641 3   | 5 19 136 642 4021 4 |
| SecReT07 | SecReT07 | none        | 5 18 95 310 650 83     | 5 18 95 310 641 4   | 5 19 136 642 4021 4 |
| DH       | DH       | none        | 5 18 95 310 650 83     | 5 18 95 310 641 5   | 5 19 136 642 4021 4 |

The overall percentage of state-space reduction for each protocol and an average (99%) suggest that our combined techniques are remarkably effective (the reduced number of states is on average only 1% or less of the original number of states). The state reduction achieved by consuming input messages first is difficult to analyze, since the reduction shown in Table 1 for this optimization (labelled as “Input First”) is 0. The reason is that it can reduce the number of
states in protocols that contain several input messages in the strands, as in the NSPK protocol, but in general it simply reduces the length of the narrowing sequences and therefore more states can be generated at an earlier depth of the narrowing tree compared to the case where the optimization is not used. Table 2 summarizes the different techniques yielding a finite space for each protocol. The use of grammars and the transition subsumption are clearly the most useful techniques in general. Indeed, all examples have a finite search space thanks to the combined use of the different state space reduction techniques. Note that grammars are insufficient to obtain a finite space for the SecReT07 example, while subsumption and the super lazy intruder are essential in this case.

6. Concluding Remarks

The Maude-NPA can analyze the security of cryptographic protocols, modulo given algebraic properties of the protocol’s cryptographic functions in executions with an unbounded number of sessions and with no approximations or data abstractions. In this full generality, protocol security properties are well-known to be undecidable. The Maude-NPA uses backwards narrowing-based search from a symbolic description of a set of attack states by means of patterns to try to reach an initial state of the protocol. If an attack state is reachable from an initial state, the Maude-NPA’s complete narrowing methods are guaranteed to prove it. But if the protocol is secure, the backwards search may be infinite and never terminate.

It is therefore very important, both for efficiency and to achieve full verification whenever possible when a protocol is secure, to use state-space reduction techniques that: (i) can drastically cut down the number of states to be explored; and (ii) have in practice a good chance to make the, generally infinite, search space finite without compromising the completeness of the analysis; that is, so that if a protocol is indeed secure, failure to find an attack in such a finite state space guarantees the protocol’s security for that attack relative to the assumptions about the intruder actions and the algebraic properties. We have presented a number of state-space reduction techniques used in combination by the Maude-NPA for exactly these purposes. We have given precise characterizations of these techniques and have shown that they preserve completeness, so that if no attack is found and the state space is finite, full verification of the given security property is achieved.

Using several representative examples we have also given an experimental evaluation of these techniques. Our experiments support the conclusion that,

| Protocol   | Finite State Space Achieved by:          |
|------------|------------------------------------------|
| NSPK       | Grammars and Subsumption                |
| NSL        | Grammars and Subsumption                |
| SecReT06   | Subsumption or (Grammars and Lazy)      |
| SecReT07   | Subsumption and Lazy                    |
| DH         | Grammars and Subsumption                |

Table 2: Finite state space achieved by reduction techniques
when used in combination, these techniques: (i) typically provide drastic state space reductions; and (ii) they can often yield a finite state space, so that whether the desired security property holds or not can in fact be decided automatically, in spite of the general undecidability of such problems.

Acknowledgements

We are very thankful to Sonia Santiago for her help on evaluating the different benchmarks.

References

[1] Basin, D., Mödersheim, S., Viganò, L., 2005. OFMC: A symbolic model checker for security protocols. International Journal of Information Security 4 (3), 181–208.

[2] Dolev, D., Yao, A., 1983. On the security of public key protocols. IEEE Transaction on Information Theory 29 (2), 198–208.

[3] Escobar, S., Hendrix, J., Meadows, C., Meseguer, J., 2007. Diffie-Hellman cryptographic reasoning in the Maude-NRL protocol analyzer. In: Proc. 2nd International Workshop on Security and Rewriting Techniques (SecReT 2007).

[4] Escobar, S., Meadows, C., Meseguer, J., 2006. A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties. Theor. Comput. Sci. 367 (1-2), 162–202.

[5] Escobar, S., Meadows, C., Meseguer, J., 2007. Equational cryptographic reasoning in the Maude-NRL protocol analyzer. In: Proc. 1st International Workshop on Security and Rewriting Techniques (SecReT 2006). ENTCS 171(4), Elsevier, pp. 23–36.

[6] Escobar, S., Meadows, C., Meseguer, J., 2008. State space reduction in the Maude-NRL Protocol Analyzer. In: Jajodia, S., López, J. (Eds.), Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings. Vol. 5283 of Lecture Notes in Computer Science. Springer, pp. 548–562.

[7] Escobar, S., Meadows, C., Meseguer, J., 2009. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In: Aldini, A., Barthe, G., Gorrieri, R. (Eds.), FOSAD 2008/2009 Tutorial Lectures. Vol. 5705 of LNCS. Springer, pp. 1–50.

[8] Escobar, S., Meseguer, J., 2007. Symbolic model checking of infinite-state systems using narrowing. In: Proceedings of the 18th International Conference on Rewriting Techniques and Applications (RTA’07). Vol. 4533 of Lecture Notes in Computer Science. pp. 153–168.
[9] Escobar, S., Meseguer, J., Sasse, R., 2008. Effectively checking the finite variant property. In: Voronkov, A. (Ed.), Rewriting Techniques and Applications, 19th International Conference, RTA 2008, Hagenberg, Austria, July 15-17, 2008, Proceedings. Vol. 5117 of Lecture Notes in Computer Science. Springer, pp. 79–93.

[10] Escobar, S., Meseguer, J., Sasse, R., 2008. Variant narrowing and equational unification. In: Rossu, G. (Ed.), Proc. 7th. Intl. Workshop on Rewriting Logic and its Applications. ENTCS to appear. Elsevier.

[11] Escobar, S., Meseguer, J., Sasse, R., 2010. Folding variant narrowing and optimal variant termination. In: ¨Olvezcsky, P. (Ed.), Rewriting Logic and its Applications, 8th International Workshop, WRLA 2010, Proceedings. Lecture Notes in Computer Science. Springer, to appear.

[12] Escobar, S., Sasse, R., Meseguer, J., 2011. Folding variant narrowing and optimal variant termination. J. Log. Algebr. Program. To appear.

[13] Fabrega, F. J. T., Herzog, J., Guttman, J., 1999. Strand Spaces: What Makes a Security Protocol Correct? Journal of Computer Security 7, 191–230.

[14] Meadows, C., 1996. Language generation and verification in the NRL protocol analyzer. In: Ninth IEEE Computer Security Foundations Workshop, March 10 - 12, 1996, Dromquinna Manor, Kenmare, County Kerry, Ireland. IEEE Computer Society, pp. 48–61.

[15] Meadows, C., 1996. The NRL protocol analyzer: An overview. Journal of logic programming 26 (2), 113–131.

[16] Meseguer, J., 1992. Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science 96 (1), 73–155.

[17] Meseguer, J., 1998. Membership algebra as a logical framework for equational specification. In: Parisi-Presicce, F. (Ed.), Proc. WADT’97. Springer LNCS 1376, pp. 18–61.

[18] Sasse, R., Escobar, S., Meadows, C., Meseguer, J., 2010. Protocol analysis modulo combination of theories: A case study in Maude-NPA. In: 6th International Workshop on Security and Trust Management — STM’10. LNCS. Springer-Verlag, to appear.

[19] Shmatikov, V., Stern, U., 1998. Efficient finite-state analysis for large security protocols. In: 11th Computer Security Foundations Workshop — CSFW-11. IEEE Computer Society Press.

[20] TeReSe (Ed.), 2003. Term Rewriting Systems. Cambridge University Press, Cambridge.
[21] Thati, P., Meseguer, J., 2007. Symbolic reachability analysis using narrowing and its application verification of cryptographic protocols. J. Higher-Order and Symbolic Computation 20 (1–2), 123–160.