# State Space Reduction in the Maude-NRL Protocol Analyzer

Santiago Escobar[a], Catherine Meadows[b], José Meseguer[c], Sonia Santiago[d]

[a] *DSIC-ELP, Universidad Politécnica de Valencia, Valencia, Spain*
[b] *Naval Research Laboratory, Washington, DC, USA*
[c] *University of Illinois at Urbana-Champaign, Urbana, IL, USA*
[d] *DSIC-ELP, Universidad Politécnica de Valencia, Valencia, Spain*

**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 [11, 14] 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 [23], 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 [14]). 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 [11]). Finally, the tool's inference system supports reasoning modulo the algebraic properties of cryptographic and other functions (see [12, 10, 28]). Such algebraic properties are expressed as equational theories $E = E' \uplus Ax$ 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 $E = E' \uplus Ax$ [17, 18, 19], which under appropriate checkable conditions [16] 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 [23, 11] and Section 5.1). 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 95% (i.e., the average size of the reduced state space is 5% 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, whereas the state space will be infinite without our optimizations.

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 [29]), the use of lazy evaluation techniques (used in constraint-evaluation based approaches, and by us in a somewhat different form), and the identification of premature use of nonces (used by us and Scyther[8]). 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 give an overview of the various state space re-

duction techniques that have been introduced to control state explosion. In Sections 5, 6, and 7 we describe the state space reduction techniques and give proofs of their completeness as well as showing their relations to other optimization techniques in the literature. In Section 5, we first briefly describe how automatically generated grammars provide the main reduction that cuts down the search space. In this section, we also describe the early detection of inconsistent states (that will never reach an initial state). In Section 6, we obtain a second important state-space reduction by detecting redundant states using several optimizations: (i) reducing the number of logical variables present in a state, (ii) giving priority to input messages in strands, and (iii) a relation of transition subsumption (to discard transitions and states already being processed in another part of the search space). In Section 7, we obtain a third important state-space reduction by defining the super-lazy intruder, which delays the generation of substitution instances as much as possible. In Section 8 we describe our experimental evaluation of these state-space reduction techniques. In Section 9, we provide a detailed comparison of our state-space reduction techniques with related work. In Section 10 we describe future work and conclude the paper.

This is an extended and improved version of [13], including proofs of all the results, a refinement of the interaction between the transition subsumption and the super-lazy intruder (Section 7.4), more examples and explanations, as well as more benchmarked protocols.

## 2. Background on Term Rewriting

We follow the classical notation and terminology from [30] for term rewriting and from [24, 25] for rewriting logic and order-sorted notions. We assume an *order-sorted signature* $\Sigma$ with a finite poset of sorts $(\mathsf{S}, \leq)$ and a finite number of function symbols. We assume an $\mathsf{S}$-sorted family $\mathcal{X} = \{\mathcal{X}_\mathsf{s}\}_{\mathsf{s}\in\mathsf{S}}$ of mutually disjoint variable sets with each $\mathcal{X}_\mathsf{s}$ countably infinite. $\mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ denotes the set of terms of sort $\mathsf{s}$, and $\mathcal{T}_{\Sigma,\mathsf{s}}$ the set of ground terms of sort $\mathsf{s}$. We write $\mathcal{T}_\Sigma(\mathcal{X})$ and $\mathcal{T}_\Sigma$ for the corresponding term algebras. We write $\mathcal{V}ar(t)$ for the set of variables present in a term $t$. The set of positions of a term $t$ is written $Pos(t)$, and the set of non-variable positions $Pos_\Sigma(t)$. 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 $\mathcal{D}om(\sigma)$, to $\mathcal{T}_\Sigma(\mathcal{X})$. The set of variables introduced by $\sigma$ is $\mathcal{R}an(\sigma)$. The identity substitution is *id*. Substitutions are homomorphically extended to $\mathcal{T}_\Sigma(\mathcal{X})$. 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 $t \in \mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$, $t' \in \mathcal{T}_\Sigma(\mathcal{X})_{\mathsf{s}'}$, and $s$ and $s'$ are sorts in the same connected component of the poset $(\mathsf{S}, \leq)$. Given a set $E$ of $\Sigma$-equations, order-sorted equational logic induces a congruence relation $=_E$ on terms $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})$ (see [25]). Throughout this paper we assume that $\mathcal{T}_{\Sigma,\mathsf{s}} \neq \emptyset$ for every sort $\mathsf{s}$. We denote the $E$-equivalence class of a term

$t \in \mathcal{T}_\Sigma(\mathcal{X})$ as $[t]_E$ and the $E$-equivalence classes of all terms $\mathcal{T}_\Sigma(\mathcal{X})$ and $\mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ as $\mathcal{T}_{\Sigma/E}(\mathcal{X})$ and $\mathcal{T}_{\Sigma/E}(\mathcal{X})_\mathsf{s}$, 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 $CSU_E(t = t')$[1]. We say $CSU_E(t = t')$ is *finitary* if it contains a finite number of $E$-unifiers. $CSU(t = t')$ 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 $l \to r$, where $l \notin \mathcal{X}$ and $l, r \in \mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ for some sort $\mathsf{s} \in \mathsf{S}$. 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 $l \to r \in R$, $l, r \in \mathcal{T}_\Sigma(\mathcal{X})_{\mathsf{State}}$ for a top sort $\mathsf{State}$, $r \notin \mathcal{X}$, and no operator in $\Sigma$ has $\mathsf{State}$ as an argument sort.

The rewriting relation $\to_R$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is $t \overset{p}{\to}_R t'$ (or $\to_R$) if $p \in Pos_\Sigma(t)$, $l \to r \in R$, $t|_p = \sigma(l)$, and $t' = t[\sigma(r)]_p$ for some $\sigma$. The relation $\to_{R/E}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is $=_E ; \to_R ; =_E$, i.e., $t \to_{R/E} s$ iff $\exists u_1, u_2 \in \mathcal{T}_\Sigma(\mathcal{X})$ s.t. $t =_E u_1$, $u_1 \to_R u_2$, and $u_2 =_E s$. Note that $\to_{R/E}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ induces a relation $\to_{R/E}$ on $\mathcal{T}_{\Sigma/E}(\mathcal{X})$ 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 $\mathcal{T}_\Sigma(\mathcal{X})$, where the rewriting relation $\to_{R,E}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is $t \overset{p}{\to}_{R,E} t'$ (or $\to_{R,E}$) if $p \in Pos_\Sigma(t)$, $l \to r \in R$, $t|_p =_E \sigma(l)$, and $t' = t[\sigma(r)]_p$ for some $\sigma$. Note that $\to_{R,E}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ induces a relation $\to_{R,E}$ on $\mathcal{T}_{\Sigma/E}(\mathcal{X})$ by $[t]_E \to_{R,E} [t']_E$ iff $\exists w \in \mathcal{T}_\Sigma(\mathcal{X})$ s.t. $t \to_{R,E} w$ 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 $\mathcal{T}_\Sigma(\mathcal{X})$ is $t \overset{p}{\rightsquigarrow}_{\sigma,R} t'$ (or $\rightsquigarrow_{\sigma,R}$, $\rightsquigarrow_R$) if $p \in Pos_\Sigma(t)$, $l \to r \in R$, $\sigma \in CSU(t|_p = l)$, and $t' = \sigma(t[r]_p)$. Assuming that $E$ has a finitary and complete unification algorithm, the narrowing relation $\rightsquigarrow_{R,E}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is $t \overset{p}{\rightsquigarrow}_{\sigma,R,E} t'$ (or $\rightsquigarrow_{\sigma,R,E}$, $\rightsquigarrow_{R,E}$) if $p \in Pos_\Sigma(t)$, $l \to r \in R$, $\sigma \in CSU_E(t|_p = l)$, 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 [31]): (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).** [31] *Let $\mathcal{R} = (\Sigma, E, R)$ be a topmost rewrite theory, $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})$, 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 $t \rightsquigarrow^*_{\theta,R,E} t''$, $\sigma(t) =_E \tau(\theta(t))$, and $t' =_E \tau(t'')$.*

[1] We abuse notation here and write $CSU_E(t = t')$ in functional form although it is possible for a $\Sigma$-equation to have more than one complete set of $E$-unifiers.

In this paper, we consider only equational theories $E = E' \uplus Ax$ 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 $l = r \in Ax$, $\mathcal{V}ar(l) = \mathcal{V}ar(r)$. 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 $E' \uplus Ax$ [17, 18, 19].

## 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 $T_{\Sigma_\mathcal{P}/E_\mathcal{P}}$, where $\Sigma_\mathcal{P}$ is the signature defining the sorts and function symbols for the cryptographic functions and for all the state constructor symbols, and $E_\mathcal{P}$ is a set of equations specifying the *algebraic properties* of the cryptographic functions and the state constructors. Therefore, a state is an $E_\mathcal{P}$-equivalence class $[t] \in T_{\Sigma_\mathcal{P}/E_\mathcal{P}}$ with $t$ a ground $\Sigma_\mathcal{P}$-term. However, since the number of states $T_{\Sigma_\mathcal{P}/E_\mathcal{P}}$ is in general infinite, rather than exploring concrete protocol states $[t] \in T_{\Sigma_\mathcal{P}/E_\mathcal{P}}$ we explore *symbolic state patterns* $[t(x_1, \ldots, x_n)] \in T_{\Sigma_\mathcal{P}/E_\mathcal{P}}(\mathcal{X})$ on the free $(\Sigma_\mathcal{P}, E_\mathcal{P})$-algebra over a set of variables $\mathcal{X}$. 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 [11, 14], a *state* in the protocol execution is a term $t$ of sort $\mathsf{State}$, $t \in T_{\Sigma_\mathcal{P}/E_\mathcal{P}}(X)_{\mathsf{State}}$. A state is then a multiset built by an associative and commutative union operator $\_\&\_$ with identity operator $\emptyset$. Each element in the multiset is either a strand or the intruder's knowledge at that state, both explained below.[2]

The *intruder's knowledge* is represented as a multiset of facts unioned together with an associative and commutative union operator $\_,\_$ 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., $m \notin \mathcal{I}$).

A *strand* [20] 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 $m^\pm$ to denote $m^+$ or $m^-$, indistinctively. We often write $+(m)$ and $-(m)$ instead of $m^+$ and $m^-$, respectively. A strand is then a list $[msg_1^\pm, msg_2^\pm, msg_3^\pm,$

[2] We note that, as we shall see, most elements of a state (e.g. terms in the intruder knowledge, intruder strands, and most strands belonging to honest principals), cannot have multiplicity greater than one. However, this is because those states that violate this constraint are rejected as unreachable, not because the two elements are identified.

$\ldots,\ msg_{k-1}^{\pm},\ msg_k^{\pm}]$ 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., $msg_i \in T_{\Sigma_{\mathcal{P}}/E_{\mathcal{P}}}(X)_{\mathsf{Msg}}$. In Maude-NPA, strands evolve over time as the send and receive actions take place, and thus we use the symbol | 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 $\mathcal{S}_{\mathcal{P}}$ for the set of strands in the specification of the protocol $\mathcal{P}$, including the strands that describe the intruder's behavior.

The definition of the sorts used in a protocol definition is to a large part up to the user, but there are special sorts that must obey certain restrictions. We list these below.

1. Maude-NPA defines all protocol states as terms of sort State. It is an internal sort and the protocol specifier cannot add sorts as subsorts or supersorts of State and cannot create new operators of sort State.
2. Maude-NPA uses a special maximal sort Msg of messages that allows the protocol specifier to describe other sorts as subsorts of the maximal sort Msg.
3. The sort Public is a subsort of Msg to define data that is publicly available to all participants, e.g. principal names.
4. The specifier can make use of another special sort Fresh in the protocol-specific signature $\Sigma$. Terms of sort Fresh are used as arguments of terms that are intended to represent unguessable values, such as keys or nonces. The meaning of a variable of sort Fresh is that it will never be instantiated by an $E$-unifier generated during the protocol analysis. This ensures that if two nonces are represented using different variables of sort Fresh, they will never be identified and no approximation for nonces is necessary. We make explicit the Fresh variables $r_1, \ldots, r_k (k \geq 0)$ generated by a strand by writing $:: r_1, \ldots, r_k :: [msg_1^{\pm}, \ldots, msg_n^{\pm}]$, where each $r_i$ appears first in an output message $msg_{j_i}^{+}$ and can later be used in any input and output message of $msg_{j_i+1}^{\pm}, \ldots, msg_n^{\pm}$. Fresh variables generated by a strand are unique to that strand. We will thus often use them to identify a particular strand; we say that a strand is *indexed* by its fresh variables.
   Likewise, the specification of function symbols are at the discretion of the user. The only restrictions are that terms intended to be unpredictable should have a term of sort Fresh as an argument, e.g. $n(A, r)$ for a nonce generated by $A$, where $r$ is a variable of sort Fresh.

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 $(M)^X$. Symmetric encryption of message $M$ using the key $K$ is denoted by $\{M\}_K$. The protocol description is as follows.*

1. $A \hookrightarrow B : \{A \ ; \ B \ ; \ g^{N_A}\}$
   *Alice sends her name, Bob's name, and an exponentiation of a new nonce $N_A$ created by her to Bob.*
2. $B \hookrightarrow A : \{B \ ; \ A \ ; \ g^{N_B}\}$
   *Bob sends his name, Alice's name, and an exponentiation of a new nonce $N_B$ created by him to Alice.*
3. $A \hookrightarrow B : \{secret\}_{g^{N_A N_B}}$

   *Bob receives $g^{N_A}$ and he raises it to $N_B$ to obtain the key $g^{N_A N_B}$. He sends a secret to Alice encrypted using the key. Likewise, when Alice receives $g^{N_B}$, she raises it to $N_A$, to obtain the key $g^{N_B N_A}$. We assume that exponentiation satisfies the equation $g^{N_A N_B} = g^{N_A * N_B}$ and that the product operation $\_*\_$ is associative and commutative, so that*

$$g^{N_B N_A} = g^{N_A N_B} = g^{N_B * N_A}$$

   *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* Fresh*. Concatenation of two messages, e.g., $N_A$ and $N_B$, is denoted by the operator $\_;\_$, e.g., $n(A, r) \ ; \ n(B, r')$. 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 $sec(A, r)$, where $r$ is a unique variable of sort* Fresh*. The protocol-specific signature $\Sigma$ contains the following subsort relations* $(\mathsf{Name}, \mathsf{Nonce}, \mathsf{Secret}, \mathsf{Enc}, \mathsf{Exp} < \mathsf{Msg})$ *and* $(\mathsf{Gen}, \mathsf{Exp} < \mathsf{GenvExp})$*, and the following operators:*

$$\begin{array}{cc}
a, b, i : \rightarrow \mathsf{Name} & g :\rightarrow \mathsf{Gen} \\
n : \mathsf{Name} \times \mathsf{Fresh} \rightarrow \mathsf{Nonce} & sec : \mathsf{Name} \times \mathsf{Fresh} \rightarrow \mathsf{Secret} \\
_;_ : \mathsf{Msg} \times \mathsf{Msg} \rightarrow \mathsf{Msg} & e, d : \mathsf{GenvExp} \times \mathsf{Msg} \rightarrow \mathsf{Enc} \\
exp : \mathsf{GenvExp} \times \mathsf{Nonce} \rightarrow \mathsf{Exp} & _*_ : \mathsf{Nonce} \times \mathsf{Nonce} \rightarrow \mathsf{Nonce}
\end{array}$$

*In the following we will use letters $A, B$ for variables of sort* Name*, letters $r, r', r''$ for variables of sort* Fresh*, letters $M, M_1, M_2$ for variables of sort* Msg*, letters $E, E'$ for variables of sort* GenvExp*, and letters $N, N'$ for variables of sort* Nonce*.*

*The encryption/decryption cancellation properties are described using the equations*

$$e(E, d(E, M)) = M \text{ and } d(E, e(E, M)) = M$$

*in* $E_{\mathcal{P}}$. *The key algebraic property of exponentiation,* $x^{y^z} = x^{y*z}$, *is described using the equation*

$$exp(exp(G, N), N') = exp(G, N * N')$$

*in* $E_{\mathcal{P}}$ *(where variable* $G$ *is of sort* Gen *instead of the more general sort* GenvExp *in order to provide a finitary narrowing-based unification procedure modulo* $E_{\mathcal{P}}$, *see [10] for details on this concrete equational theory). We also include the fact that* $*$ *is associative-commutative. 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:*

$$:: r, r' :: [\ (A; B; exp(g, n(A, r)))^+,\ (B; A; E)^-,\ (e(exp(E, n(A, r)), sec(A, r')))^+]$$

$$:: r'' :: [\ (A; B; E')^-,\ (B; A; exp(g, n(B, r'')))^+,\ (e(exp(E', n(B, r'')), SR)^-]$$

*The following strands describe the intruder abilities according to the Dolev-Yao attacker's capabilities [9].*

- $[M_1^-, M_2^-, (M_1; M_2)^+]$ *Concatenation*
- $[(M_1; M_2)^-, M_1^+]$ *Left-deconcatenation*
- $[(M_1; M_2)^-, M_2^+]$ *Right-deconcatenation*
- $[\ E^-, M^-, e(E, M)^+\ ]$ *Encryption*
- $[\ E^-, M^-, d(E, M)^+\ ]$ *Decryption*
- $[\ M_1^-, M_2^-, (M_1 * M_2)^+\ ]$ *Multiplication*
- $[\ M_1^-, M_2^-, exp(M_1, M_2)^+\ ]$ *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.,* $[E^-, e(E, M)^-, M^+] =_{E_{\mathcal{P}}} [E^-, e(E, M)^-, d(E, e(E, M))^+]$.

*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 $R_{\mathcal{P}}$ of *rewrite rules*, so that the rewrite theory $(\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, R_{\mathcal{P}})$ characterizes the behavior of protocol $\mathcal{P}$ modulo the equations $E_{\mathcal{P}}$. In the case where new strands are not introduced into the state, the corresponding rewrite rules in $R_{\mathcal{P}}$ are as follows[3], where $L, L_1, L_2$ denote lists of input and output messages $(+m,-m)$, $IK, IK'$ denote sets of intruder facts ($m{\in}\mathcal{I}$,$m{\notin}\mathcal{I}$), and $SS, SS'$ denote sets of strands:

$$[L \mid M^-, L'] \;\&\; SS \;\&\; (M{\in}\mathcal{I}, IK) \rightarrow [L, M^- \mid L'] \;\&\; SS \;\&\; (M{\in}\mathcal{I}, IK) \quad (1)$$

$$[L \mid M^+, L'] \;\&\; SS \;\&\; IK \rightarrow [L, M^+ \mid L'] \;\&\; SS \;\&\; IK \quad (2)$$

$$[L \mid M^+, L'] \;\&\; SS \;\&\; (M{\notin}\mathcal{I}, IK) \rightarrow [L, M^+ \mid L'] \;\&\; SS \;\&\; (M{\in}\mathcal{I}, IK) \quad (3)$$

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". This enables a very important restriction of the tool, expressed by saying that the intruder *learns* a term *only once* [11]: if the intruder needs to use a term twice, then he must learn it the first time it is needed; if he learns a term in a state $St_1$ and needs to learn it again in a previous state $St_2$, found later during the backwards search, then $St_2$ will be discarded as unreachable[4]. Note that Rules (1)–(3) are *generic*: they belong to $\mathcal{R}_{\mathcal{P}}$ for *any* protocol $\mathcal{P}$.

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 the set $\mathcal{S}_{\mathcal{P}}$ of strands associated to the protocol $\mathcal{P}$) as follows:

[3] To simplify the exposition, we omit the fresh variables at the beginning of each strand in a rewrite rule.

[4] Note that because Maude-NPA finds all possible paths, it will also find the path in which the term is learned later in the backwards search, deleting $St_2$ does not lead to incompleteness.

$$\text{for each } [\ l_1,\ u^+,\ l_2\ ] \in \mathcal{S}_\mathcal{P} : [\ l_1 \mid u^+, l_2\ ]\ \&\ SS\ \&\ (u \notin \mathcal{I}, IK) \rightarrow SS\ \&\ (u \in \mathcal{I}, IK) \quad (4)$$

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 ($m \in \mathcal{I}$,$m \notin \mathcal{I}$), 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:

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

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 $\mathcal{P}$, the set $R_\mathcal{P}$ of rewrite rules obtained from the protocol strands $\mathcal{S}_\mathcal{P}$ that are used for backwards narrowing reachability analysis *modulo* the equational properties $E_\mathcal{P}$ is $R_\mathcal{P} = \{(1), (2), (3)\} \cup (4)$. 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 $E = E' \uplus Ax$ 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 $R_\mathcal{P}$ (see [11]). Namely, for any term $m \in \mathcal{I}$ (resp. term $m^-$), we discard any substitution $\sigma$ that makes $\sigma(m) \in \mathcal{I}$ (resp. $(\sigma(m))^-$) being $E'$,$Ax$-irreducible. This is because many of our optimization techniques rely on the assumption that terms have a unique normal form modulo an 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 7.4, 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 $R_\mathcal{P}^{-1}$, where $v \longrightarrow u$ is in $R_\mathcal{P}^{-1}$ iff $u \longrightarrow v$ is in $R_\mathcal{P}$. Reachability analysis can be performed *symbolically*, not on concrete states but on symbolic state patterns $[t(x_1, \ldots, x_n)]_{E_\mathcal{P}}$ by means of *narrowing modulo* $E_\mathcal{P}$ (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 $u \in \mathcal{I}$ 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:*

$$:: r' :: [\,(A; B; E')^-, (B; A; exp(g, n(B, r')))^+, (e(exp(E', n(B, r')), sec(a, r'')))^- \mid nil\,] \;\&\; SS \;\&\; (sec(a, r'') \in \mathcal{I}, IK) \qquad (\dagger)$$

*We expressed the attack state in terms of a secrecy violation, but we could also have specified it in terms of an authentication violation. A more thorough discussion of how attack states are specified in Maude-NPA is given in [14].*

*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:*

$[nil \mid exp(g, n(a, r)))^-, N_1^-, exp(g, N_1 * n(a, r))^+] \;\&$
$[nil \mid exp(g, N_1 * n(a, r))^-, e(exp(g, N_1 * n(a, r)), sec(a, r'')^-, sec(a, r'')^+] \;\&$
$[nil \mid exp(g, n(b, r')))^-, N_2^-, exp(g, N_2 * n(b, r'))^+] \;\&$
$[nil \mid exp(g, N_2 * n(b, r'))^-, sec(a, r'')^-, e(exp(g, N_2 * n(b, r')), sec(a, r''))^+] \;\&$
$[nil \mid (b; a; exp(g, n(b, r')))^-, (b; exp(g, n(b, r')))^+] \;\&$
$[nil \mid (b; exp(g, n(b, r')))^-, exp(g, n(b, r'))^+] \;\&$
$[nil \mid (a; B'; exp(g, n(a, r)))^-, (B'; exp(g, n(a, r)))^+] \;\&$
$[nil \mid (B'; exp(g, n(a, r)))^-, exp(g, n(a, r))^+] \;\&$
$:: r' ::$
$[nil \,|\, (a; b; exp(g, N_2))^-, (b; a; exp(g, n(b, r')))^+, e(exp(g, N_2 * n(b, r')), sec(a, r''))^-] \;\&$
$:: r'', r ::$
$[nil \mid (a; B'; exp(g, n(a, r)))^+, (B'; a; exp(g, N_1))^-, e(exp(g, N_1 * n(a, r)), sec(a, r''))^+]$

*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:*

$1.(a; b; exp(g, N_2))^-$
$2.(b; a; exp(g, n(b, r')))^+$
$3.(b; a; exp(g, n(b, r')))^-$
$4.(a; exp(g, n(b, r')))^+$
$5.(a; exp(g, n(b, r')))^-$
$6.(exp(g, n(b, r')))^+$
$7.(exp(g, n(b, r')))^-$
$8.N_2^-$
$9.exp(g, N_2 * n(b, r'))^+$
$10.(a; B'; exp(g, n(a, r)))^+$
$11.(a; B'; exp(g, n(a, r)))^-$
$12.(B'; exp(g, n(a, r)))^+$
$13.(B'; exp(g, n(a, r)))^-$
$14.(exp(g, n(a, r)))^+$
$15.(exp(g, n(a, r)))^-$
$16.N_1^-$
$17.exp(g, N_1 * n(a, r))^+$
$18.(B'; a; exp(g, N_1))^-$
$19.e(exp(g, N_1 * n(a, r)), sec(a, r''))^+$
$20.e(exp(g, N_1 * n(a, r)), sec(a, r''))^-$
$21.exp(g, N_1 * n(a, r))^-$
$22.sec(a, r'')^+$
$23.exp(g, N_2 * n(b, r'))^-$
$24.sec(a, r'')^-$
$25.e(exp(g, N_2 * n(b, r'), sec(a, r''))^+$
$26.e(exp(g, N_2 * n(b, r')).sec(a, r''))^-$

*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 $exp(g, N_2 * n(b, r'))$ 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 12) through 17) describe the intruder constructing the key $exp(g, N_1 * n(a, r))$ 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 generable by the intruder, and so not searched for; namely, intruder strands generating variable $N_1$, variable $N_2$, term $(a; b; exp(g, N_2))$, and term $(a; B'; exp(g, N_1))$. Variables $N_1$ and $N_2$ can be filled in with any nonce, for instance nonces generated by the intruder, such as $N_2 = n(i, r''')$ and $N_1 = n(i, r'''')$ in the following way:*

$$:: r''' :: [nil \mid (n(i, r'''))^+] \;\&\; :: r'''' :: [nil \mid (n(i, r''''))^+]$$

*Also, note that nonces $N_2$ and $N_1$ are used by the intruder to generate messages $(a; b; exp(g, N_2))$ and $(a; B'; exp(g, N_1))$ in the following way:*

$[nil \mid (a)^+]$ & $[nil \mid (b)^+]$ & $[nil \mid (B')^+]$ &
$[nil \mid (g)^+]$ & $[nil \mid (g)^-, N_2^-, exp(g, N_2)^+]$ & $[nil \mid (g)^-, N_1^-, exp(g, N_1)^+]$ &
$[nil \mid (a)^-, (b)^-, (a; b)^+]$ & $[nil \mid (a; b)^-, (exp(g, N_2))^-, (a; b; exp(g, N_2))^+]$ &
$[nil \mid (B')^-, (a)^-, (B'; a)^+]$ & $[nil \mid (B'; a)^-, (exp(g, N_1))^-, (B'; a; exp(g, N_1))^+]$

*Finally, note that principal a believes she is talking to some $B'$ instead of b. This is because substituting b's name is not necessary for the intruder to produce the attack. Any arbitrary name will do.*

## 4. Overview of State Space Reduction Techniques

In this section we present Maude-NPA's state space reduction techniques. Maude-NPA's reduction techniques are applied when a state is generated in the backwards narrowing search. A number of tests are applied. If a state is identified as *unproductive*, that is, removal of that state does not affect reachability of the final state one way or the other, it is also removed. Techniques for removing unproductive states, which are later refined into unreachable and redundant states, are described in Sections 5 and 6. A state can also be identified as potentially leading to a state space explosion, in which case it may be delayed. Such a delay will be complete, but not necessarily sound; in this case, the delay may have to be reversed as the narrowing tree is generated in order to maintain soundness. We have developed one technique that falls into this class: the super-lazy intruder, which is described in Section 7.

In the remainder of this paper, we make use of a very general completeness result satisfied by Maude-NPA.

**Theorem 2 (Completeness).** [11] *Given a topmost rewrite theory* $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, R_{\mathcal{P}})$ *representing protocol* $\mathcal{P}$*, and a non-initial state* $St$ *(with logical variables), if there is a substitution* $\sigma$ *and an initial state* $St_{ini}$ *such that* $\sigma(St) \rightarrow^*_{R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_{ini}$*, then there are substitutions* $\sigma', \rho$ *and an initial state* $St'_{ini}$ *such that* $St \rightsquigarrow^*_{\sigma', R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St'_{ini}$*,* $\sigma =_{E_{\mathcal{P}}} \sigma' \circ \rho$*, and* $St_{ini} =_{E_{\mathcal{P}}} \rho(St'_{ini})$*.*

We have developed means of detecting two kinds of unproductive states: *unreachable* and *redundant* states. These are defined below.

**Definition 1 (Unreachable States).** *Given a topmost rewrite theory* $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, R_{\mathcal{P}})$ *representing protocol* $\mathcal{P}$*, a non-initial state* $St$ *(with logical variables) is* unreachable *if there is no sequence* $St \rightsquigarrow^*_{\sigma, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_{ini}$ *leading to an initial state* $St_{ini}$*.*

**Definition 2 (Redundant States).** *Given a topmost rewrite theory* $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, R_{\mathcal{P}})$ *representing protocol* $\mathcal{P}$ *and a non-initial state* $St$ *(with logical variables), a backwards narrowing step* $St \rightsquigarrow_{\sigma_1, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_1$ *such that* $St_1$ *is a non-initial state 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.,* $St_1 \rightsquigarrow^*_{\theta_1, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_{ini1}$*, there are states* $St_2 \neq_{E_{\mathcal{P}}} St_1$ *and* $St_{ini2}$*, a narrowing step* $St \rightsquigarrow_{\sigma_2, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_2$*, a narrowing sequence* $St_2 \rightsquigarrow^*_{\theta_2, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_{ini2}$*, and a substitution* $\rho$ *such that* $\sigma_1 \circ \theta_1 =_{E_{\mathcal{P}}} \sigma_2 \circ \theta_2 \circ \rho$ *and* $St_{ini1} =_{E_{\mathcal{P}}} \rho(St_{ini2})$*.*

There are three reasons for wanting to detect and remove unproductive states. 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.

Our discussion of Maude-NPA's optimization techniques is organized as follows. In Section 5 we present Maude-NPA's techniques for identifying unreachable states. In Section 6 we present Maude-NPA's techniques for identifying redundant states. Finally, in Section 7, we present the super-lazy intruder technique for delaying states that may lead to state explosion and restoring them to their unmodified version when necessary to maintain soundness.

## 5. Identifying Unreachable States

In this section we describe the various techniques Maude-NPA uses to identify unreachable states. These techniques have been adapted from those used by its ancestor, NPA.

There are two ways in which Maude-NPA identifies unreachable states. One is the use of inductive techniques to define grammars that characterize terms that can never be learned by the intruder. This has been described in detail in previous work, e.g. [11] and [22], so we give only a brief overview here in Section 5.1. The other is the identification of states that describe impossible events, e.g. states in which an intruder learns a term containing a nonce that has not yet been created. This is described in Section 5.2.

### *5.1. 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 $t{\in}\mathcal{I}$ of the intruder's knowledge of a state $St$ that will never be transformed into a negative fact $\theta(t){\notin}\mathcal{I}$ 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 [22, 11] 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.

Maude-NPA generates grammars completely automatically, inferring initial grammars from the protocol specification, and using built-in inference rules to generated new grammars.

As an example of how grammars work, consider again the attack pattern (†) in Example 2. This pattern contains the intruder knowledge fact $sec(a, r"){\in}\mathcal{I}$.

If we run Maude-NPA without any optimizations, it will generate a state containing the facts $(M; sec(a, r"))\in\mathcal{I}$ and $sec(a, r")\notin\mathcal{I}$, then a state containing the facts $(M'; M; sec(a, r"))\in\mathcal{I}$, $(M; sec(a, r"))\notin\mathcal{I}$, and $sec(a, r")\notin\mathcal{I}$, and so on, producing an infinite sequence of states. We can describe the sequence beginning with the state $(M; sec(a, r"))\in\mathcal{I}$ and $sec(a, r")\notin\mathcal{I}$ using the following grammar:

```
grl M inL => (M' ; M) inL . ;
grl M notInI,  => (M' ; M) inL .)
```

where the first production describes the concatenation of two terms, the second of which is in the language $L$, and the second production gives the concatenation of two terms, the second of which is not yet known by the intruder.

We now want to see if all members of the language characterized by this grammar are unlearnable by the intruder. In order to do this, we attempt to show that, if the intruder learns a member of the language, it must have already known a member of the language. This is done by giving each production of the grammar to Maude-NPA as a goal, and using it to determine that in each preceding state the intruder knows a member of the language. Thus, if we give the state $(M'; M)\in\mathcal{I}$ to Maude-NPA, keeping in mind that the $M$ is a member of $L$, then we can see that one of the preceding states it finds contains $M\in\mathcal{I}, M'\in\mathcal{I}$. Since $M$ is a member of $L$, this state requires that the intruder knows a member of the language.

It is unlikely that initially all preceding states will require that the intruder knows a member of the language. Whenever that is the case, Maude-NPA employs heuristics to add or modify a production (by adding constraints of the form `M notLeq Pattern`) so that some term known by the intruder is in the language defined by the new grammar. This process is iterated until it either reaches a fixed point, or no more heuristics can be applied. These heuristics and a proof of correctness are given in [11].

For example, the initial grammar described above terminates in the following:

```
grl M inL => e(E, M) inL . ;
grl M inL => d(E, M) inL . ;
grl M inL => (M ; M') inL . ;
grl M inL => (M' ; M) inL . ;
grl M notInI,
    M notLeq exp(g, n(A, r)),
    M notLeq B ; exp(g, n(A, r')) => (M' ; M) inL .)
```

where all the productions and exceptions refer to normal forms of messages w.r.t. the equational theory $E_{\mathcal{P}}$.

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 $M\notin\mathcal{I}$ appears explicitly in the intruder's knowledge

of the current state being checked for unreachability (described by constraints of the form `M notInI`). Moreover, any term of any of the following normal forms: $e(E, M)$, $d(E, M)$, $(M'; M)$, or $(M; M')$ cannot be learned by the intruder if subterm $M$ a member of the language described by the above grammar.

The interested reader can determine that the term we were originally interested in, i.e., the term $(M; sec(a, r"))$, where $sec(a, r'')$ is not yet known by the intruder, is indeed a member of the language.

In order for Maude-NPA to generate grammars, it needs a set of initial grammars to start from. In NPA, users had to define their own initial grammars. Maude-NPA, however, generates initial grammars automatically. For any intruder strand of the form $[(M_1)^-, \ldots, (M_k)^-, (f(M_1, \ldots, M_k)^+]$, it generates $k+1$ initial grammars: an initial grammar with the production $f(M_1, \ldots, M_k) \in L$, and, for each negative term $M_i$, an initial grammar with the production $M_i \notin \mathcal{I} \Rightarrow f(M_1, \ldots, M_k) \in L$.

*5.2. Early Detection of Inconsistent States*

There are several types of states that are always unreachable or inconsistent. We give examples below.

**Example 3.** *Consider again the attack pattern* (†) *in Example 2. After a couple of backwards narrowing steps, the Maude-NPA finds the following state, where the intruder learns* $e(exp(E', n(B, r')), sec(a, r''))$ *by assuming she can learn* $exp(E', n(B, r'))$ *and* $sec(a, r'')$ *and combine them:*

$$
\begin{array}{l}
[\ nil \mid (exp(E', n(B, r')))^-, (sec(a, r''))^-, (e(exp(E', n(B, r')), sec(a, r'')))^+\ ]\ \& \\
:: r' :: \\
[\ (A; B; E')^-, (B; A; exp(g, n(B, r')))^+ \mid (e(exp(E', n(B, r')), sec(a, r')))^-\ ]\ \& \\
(sec(a, r'') \in \mathcal{I},\ exp(E', n(B, r')) \in \mathcal{I},\ e(exp(E', n(B, r')), sec(a, r'')) \notin \mathcal{I})
\end{array}
\qquad (\ddagger)
$$

*From this state, the intruder tries to learn* $sec(a, r'')$ *by assuming she can learn messages* $(e(exp(E', n(B, r')), sec(a, r')))$ *and* $exp(E', n(B, r'))$ *and combines them in a decryption:*

$$
\begin{array}{l}
[\ nil \mid (exp(E', n(B, r')))^-, (e(exp(E', n(B, r')), sec(a, r')))^-, (sec(a, r''))^+\ ]\ \& \\
[\ nil \mid (exp(E', n(B, r')))^-, (sec(a, r''))^-, (e(exp(E', n(B, r')), sec(a, r')))^+\ ]\ \& \\
:: r' :: \\
[\ (A; B; E')^-, (B; A; exp(g, n(B, r')))^+ \mid (e(exp(E', n(B, r')), sec(a, r')))^-\ ]\ \& \\
(sec(a, r'') \in \mathcal{I},\ exp(E', n(B, r')) \in \mathcal{I}, \\
e(exp(E', n(B, r')), sec(a, r'')) \in \mathcal{I},\ e(exp(E', n(B, r')), sec(a, r'')) \notin \mathcal{I})
\end{array}
$$

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

If 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:

**Proposition 1.** 1. *A state St containing two contradictory facts $t \in \mathcal{I}$ and $t \notin \mathcal{I}$ (modulo $E_{\mathcal{P}}$) 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 $[m_1^{\pm}, \ldots, t^{-}, \ldots, m_{j-1}^{\pm} \mid m_j^{\pm}, \ldots, m_k^{\pm}]$ (modulo $E_{\mathcal{P}}$).*
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., $s =:: r_1, \ldots, r, \ldots, r_k :: [m_1^{\pm}, \ldots, m_{j-1}^{\pm} \mid m_j^{\pm}, \ldots, m_k^{\pm}]$, cannot produce r, i.e., r is not a subterm of any output message in $m_1^{\pm}, \ldots, m_{j-1}^{\pm}$.*
4. *A state St containing a strand of the form $s = [m_1^{\pm}, \ldots, t^{-}, \ldots, m_{j-1}^{\pm} \mid m_j^{\pm}, \ldots, m_k^{\pm}]$ for some term t such that t contains a fresh variable r and the strand in St indexed by r cannot produce r.*

*Proof.* The proofs of unreachability of each case are given below.

1. After backwards narrowing, this will result in a state that violates the "intruder-learns-only-once" rule.
2. This state will become a case of 1 after backwards narrowing.
3. In order for $t$ to be found by the intruder, some other strand besides the strand in $St$ indexed by $r$ would need to produce it. But that strand would also need to be indexed by $r$, which contradicts the unique origin of fresh values.
4. We first note that any backward narrowing step will leave strand $s$ still unable to produce $r$. Moreover, eventually, a backwards narrowing step must result in the addition of $t \in \mathcal{I}$. Thus this state becomes a case of 3 after backwards narrowing.

## 6. Redundant States

In this section we describe how Maude-NPA identifies and removes redundant states.

### *6.1. Limiting Dynamic Introduction of New Strands*

As pointed out in Section 3.1, rules of type (4) are intended to be the only ones that introduce new strands. Rules of type (1), (2), and (3) are not intended for such an introduction. However, unless they are modified, they will introduce new strands, but in an unproductive way. That is, 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 $X{\in}\mathcal{I}$, 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)*:*

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

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

$$SS \;\&\; IK \overset{\sigma}{\rightsquigarrow}_{R,E} SS' \,\&\, [L \mid M^-, L'] \,\&\, (M{\in}\mathcal{I}, IK')$$

*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:

$$SS \,\&\, [L \mid M^-, L'] \,\&\, (M{\in}\mathcal{I}, IK) \rightarrow SS \,\&\, [L, M^- \mid L'] \,\&\, IK \qquad (5)$$

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.

Since this optimization is achieved by putting restrictions on attack patterns and rewrite rules, the soundness proof is trivial and thus omitted. However, a proof of completeness is still needed. The set of rewrite rules actually used for backwards narrowing is $\overline{R_{\mathcal{P}}} = \{(5), (2), (3)\} \cup (4)$. The following result ensures that $R_{\mathcal{P}}$ and $\overline{R_{\mathcal{P}}}$ compute similar initial states by backwards reachability analysis.

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

**Proposition 2.** *Let $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, R_{\mathcal{P}})$ be a topmost rewrite theory representing protocol $\mathcal{P}$ and let $\overline{R_{\mathcal{P}}}$ be defined as above. Let $St = ss \,\&\, SS \,\&\, (ik, IK)$. Let $ss = \{s_1, \ldots, s_n\}$ be a multiset of strands, $ik = \{k_1, \ldots, k_m\}$ be a set of intruder facts, SS is a variable denoting a set of strands, and IK is a variable denoting the intruder knowledge. Let $St' = ss \,\&\, ik$. If there is an initial state $St_{ini}$ and a substitution $\sigma$ such that $St \leadsto^*_{\sigma, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_{ini}$, then there is an initial state $St'_{ini}$ and two substitutions $\sigma'$, $\rho$ such that $St' \leadsto^*_{\sigma', \overline{R_{\mathcal{P}}}^{-1}, E_{\mathcal{P}}} St'_{ini}$, $\sigma =_{E_{\mathcal{P}}} \sigma' \circ \rho$, and $\rho(St'_{ini}) \subseteq St_{ini}$.*

*Proof.* We obtain the narrowing sequence 1) $St' \leadsto^*_{\sigma', \overline{R_{\mathcal{P}}}^{-1}, E_{\mathcal{P}}} St'_{ini}$ from 2) $St \leadsto^*_{\sigma, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_{ini}$ by removing $\sigma SS$ and $\sigma IK$ from every state in 2) and then deleting the transitions that become trivial. We then check that this results in a sequence resulting from application of the rules $\overline{R_{\mathcal{P}}}$. This is straightforward, except that we need to check that any conditions required by the new Rule (5) are fulfilled. The only such condition is that the intruder's knowledge be a set of intruder facts without repeated elements, i.e., the union operator $\_,\_$ is $ACUI$ (associative-commutative-identity-idempotent). This follows directly from the restriction in [11] that the intruder learns a term only once. □

*6.2. 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 major 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* (†) *in Example 2. Since the strand in the attack pattern has the input message $(e(exp(E', n(B, r')), sec(a, r'')))^-$ but also has the intruder challenge $sec(a, r'') \in \mathcal{I}$, 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 thus omitted.

Similar strategies have been employed by other tools in forward searches. For example, in [29], 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.

*6.3. Subsumption Partial Order Reduction*

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 [15], 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 6.** *Consider again the attack pattern* (†) *in Example 2. After a couple of backwards narrowing steps, the Maude-NPA finds the state* (‡) *of Example 3:*

$$
\begin{array}{l}
[\ nil \mid exp(E', n(B, r'))^{-}, sec(a, r'')^{-}, (e(exp(E', n(B, r')), sec(a, r')))^{+}\ ]\ \& \\
::\ r'\ :: \\
[\ (A; B; E')^{-}, (B; A; exp(g, n(B, r')))^{+} \mid (e(exp(E', n(B, r')), sec(a, r')))^{-}\ ]\ \& \\
(sec(a, r'') \in \mathcal{I},\ exp(E', n(B, r')) \in \mathcal{I},\ e(exp(E', n(B, r')), sec(a, r'')) \notin \mathcal{I})
\end{array}
$$

*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 $exp(G, N)$ for $G$ a generator –indeed the constant $g$— and $N$ a nonce variable:*

$$
\begin{array}{l}
[\ nil \mid exp(G, n(B, r'))^{-}, N^{-}, exp(G, N * n(B, r'))^{+}\ ]\ \& \\
[\ nil \mid exp(G, N * n(B, r'))^{-}, sec(a, r'')^{-}, (e(exp(G, N * n(B, r')), sec(a, r')))^{+}\ ]\ \& \\
::\ r'\ ::\ [\ (A; B; exp(G, N))^{-}, (B; A; exp(g, n(B, r')))^{+} \\
\qquad\qquad\qquad\qquad\qquad\qquad \mid (e(exp(G, N * n(B, r')), sec(a, r')))^{-}\ ]\ \& \\
(sec(a, r'') \in \mathcal{I},\ exp(G, n(B, r')) \in \mathcal{I},\ N \in \mathcal{I}, \\
exp(G, N * n(B, r')) \notin \mathcal{I},\ e(exp(G, N * n(B, r')), sec(a, r'')) \notin \mathcal{I})
\end{array}
$$

*However, the unreachability of the second state is implied (modulo $E_{\mathcal{P}}$) by the unreachability of the first state; unreachability in the sense of Definition 1. Intuitively, the challenges present in the first state that are relevant for backwards reachability are included in the second state, namely, the challenges $sec(a, r'') \in \mathcal{I}$ and $exp(E', n(B, r')) \in \mathcal{I}$. 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:*

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

*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.

**Definition 4.** *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 $St_1$ and $St_2$, we write $St_1 \triangleright St_2$ (or $St_2 \triangleleft St_1$) if each intruder fact of the form $t \in \mathcal{I}$ in $St_1$ appears in $St_2$ (modulo $E_{\mathcal{P}}$) and each non-initial strand in $St_1$ appears in $St_2$ (modulo $E_{\mathcal{P}}$ and with the vertical bar at the same position).*

This is similar to the relation $\subseteq$ given in Definition 3 in Section 6.1, except for the condition that the two states be non-initial. This condition is made because, otherwise, an initial state will imply any other state, erroneously making the search space finite after an initial state has been found.

We define the relation $St_1 \blacktriangleright St_2$ 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 ($\mathcal{P}$-subsumption relation).** *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 $St_1 \blacktriangleright St_2$ (or $St_2 \blacktriangleleft St_1$) and say that $St_2$ is* $\mathcal{P}$-subsumed *by $St_1$ if there is a substitution $\theta$ s.t. $\theta(St_1) \triangleright St_2$.*

We now show that, if $St_1 \blacktriangleright St_2$, then $St_2$ can be discarded without sacrificing completeness. We do this by showing how every path from $St_2$ to an initial state can be used to construct a path from $St_1$ to an initial state, so that unreachability of $St_1$ implies unreachability of $St_2$.

We first show that if $St_1 \blacktriangleright St_2$, i.e. there is a $\theta$ such that $\theta St_1 \triangleright St_2$, and a narrowing step $St_2 \rightsquigarrow_{\sigma_2, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_2'$ for some state $St_2'$, then either $\sigma_2(\theta(St_1)) \triangleright St_2'$ (and so $St_1 \blacktriangleright St_2'$), or there is a narrowing step $St_1 \rightsquigarrow_{\sigma_1, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_1'$ for some state $St_1'$ that is either initial or for which there is a substitution $\rho$ such that $\rho St_1' \triangleright St_2'$ (and so $St_1' \blacktriangleright St_2'$). Once we have proven these results, we can

then use induction to show that, if there is a path from $St_2$ to an initial state, there is a path from $St_1$ to an initial states, and we are done.

The following results provide the appropriate connection between $\mathcal{P}$-subsumption and narrowing transitions. First, we make use of the following lemma, whose proof follows directly from the definition of $\triangleright$.

**Lemma 1.** *Suppose that we have 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 $St_1, St_2$ and substitution $\theta$ such that $\theta(St_1) \triangleright St_2$, i.e., $St_1 \blacktriangleright St_2$. Suppose furthermore that there is a narrowing step $St_2 \leadsto_{\sigma_2, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_2'$ where $St_2'$ is non-initial. Then $\sigma_2(\theta(St_1)) \not\triangleright St_2'$, if and only if (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 $E_\mathcal{P}$), or (b) there is a non-initial strand in $\sigma_2(\theta(St_1))$ that does not appear in $St_2'$ (modulo $E_\mathcal{P}$). In particular, if conditions a) and b) are satisfied, then $St_1 \blacktriangleright St_2'$.*

Suppose now that $\sigma_2(\theta(St_1)) \not\triangleright St_2'$. We consider both cases of Lemma 1 separately: either an expression $t{\in}\mathcal{I}$ in $St_2'$ or a non-initial strand in $\sigma_2(\theta(St_1))$, not appearing in $St_2'$. First, the case where an expression $t{\in}\mathcal{I}$ in $\sigma_2(\theta(St_1))$ does not appear in $St_2'$.

**Lemma 2.** *Suppose that we have 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 (i) there is a substitution $\theta$ s.t. $\theta(St_1) \triangleright St_2$, i.e., $St_1 \blacktriangleright St_2'$, (ii) there is a narrowing step $St_2 \leadsto_{\sigma_2, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_2'$, and (iii) 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 $E_\mathcal{P}$), then (a) $t{\notin}\mathcal{I}$ does appear in $St_2'$ (modulo $E_\mathcal{P}$) and (b) there is a state $St_1'$ and a substitution $\sigma_1$ such that $St_1 \leadsto_{\sigma_1, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_1'$ and either $St_1'$ is an initial state or there is a substitution $\rho$ s.t. $\rho(St_1') \triangleright St_2'$, i.e., $St_1' \blacktriangleright St_2'$,*

*Proof.* We prove the result by considering the different rules applicable to $St_2$ (remember that in $\mathcal{R}$, rewriting and narrowing steps always happen at the top position). Note that property (a) is immediate because rules in $R_\mathcal{P}$ do not remove expressions of the form $m{\in}\mathcal{I}$. Note also that if $t{\in}\mathcal{I}$ does appear in $St_2$ (modulo $E_\mathcal{P}$) and $t{\notin}\mathcal{I}$ does appear in $St_2'$ (modulo $E_\mathcal{P}$), then only Rule (3) or rules of type (4) have been applied to $St_2$ as follows:

- Reversed version of Rule (3), i.e., $St_2 \leadsto_{\sigma_2, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_2'$ using the following rule

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

  Recall that there is an intruder fact in $\sigma_2(\theta(St_1))$ of the form $t{\in}\mathcal{I}$ for $t$ a message term that does not appear in $St_2'$ (modulo $E_\mathcal{P}$) and $t =_{E_\mathcal{P}} \sigma_2(M)$. Thus, $\sigma_2(M){\in}\mathcal{I}$ does appear in $\sigma_2(\theta(St_1))$ (modulo $E_\mathcal{P}$). Here we have several cases:

– If the strand $\sigma_2([L, M^+ \mid L'])$ appears in $\sigma_2(\theta(St_1))$, then the very same narrowing step can be performed on $St_1$, i.e., there exist $\sigma_1, \rho$ such that $St_1 \rightsquigarrow_{\sigma_1, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_1'$ with the same rule and $\theta \circ \sigma_2 =_{E_{\mathcal{P}}} \sigma_1 \circ \rho$. Thus, either $St_1'$ is an initial state or $\rho(St_1') \triangleright St_2'$, since: (i) each positive intruder fact in $\sigma_2(\theta(St_1))$ of the form $u{\in}\mathcal{I}$ for $u$ a message term, except $\sigma_2(M){\in}\mathcal{I}$, appears in $\rho(St_1')$ (modulo $E_{\mathcal{P}}$), (ii) $\sigma_2(M){\notin}\mathcal{I}$ appears in $\rho(St_1')$ (modulo $E_{\mathcal{P}}$), (iii) each non-initial strand in $\sigma_2(\theta(St_1))$, except $\sigma_2([L, M^+ \mid L'])$, has not been modified and appears in $\rho(St_1')$ as well (modulo $E_{\mathcal{P}}$), and (iv) for $\sigma_2([L, M^+ \mid L'])$ in $\sigma_2(\theta(St_1))$, $\rho'([L \mid M^+, L'])$ appears in $\rho(St_1')$ and in $St_2'$.
– If the strand $\sigma_2([L_m, M^+ \mid L'])$ does not appear in $\sigma_2(\theta(St_1))$, then the strand $\sigma_2([L, M^+ \mid L'])$ corresponds to a strand $\mathcal{S}_{\mathcal{P}}$ 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 $\sigma_2([L, M^+ \mid L'])$. Note that it cannot correspond to a strand included originally in the attack pattern, because we assume that $St_1$ and $St_2$ are states generated by backwards narrowing from the same attack state and then both $St_1$ and $St_2$ should have the strand. Therefore, since the strand $\sigma_2([L, M^+ \mid L'])$ corresponds to a strand in $\mathcal{S}_{\mathcal{P}}$ and the set (4) contains a rewrite rule for each strand of the form $[\ l_1,\ u^+,\ l_2\ ]$ in $\mathcal{S}_{\mathcal{P}}$, 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 $St_1 \rightsquigarrow_{\sigma_1, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_1'$ using the rule $\alpha$ and $\theta{\circ}\sigma_2 =_{E_{\mathcal{P}}} \sigma_1{\circ}\rho$. Thus, either $St_1'$ is an initial state or $\rho(St_1'){\triangleright}St_2'$, since: (i) each positive intruder fact in $\sigma_2(\theta(St_1))$ of the form $u{\in}\mathcal{I}$ for $u$ a message term, except $\sigma_2(M){\in}\mathcal{I}$, appears in $\rho(St_1')$ (modulo $E_{\mathcal{P}}$), (ii) $\sigma_2(M){\notin}\mathcal{I}$ appears in $\rho(St_1')$ (modulo $E_{\mathcal{P}}$), (iii) each non-initial strand in $\sigma_2(\theta(St_1))$ has not been modified and appears in $\rho(St_1')$ as well (modulo $E_{\mathcal{P}}$), and (iv) $\sigma_2([\ l_1 \mid u^+,\ l_2\ ])$ appears in $\rho(St_1')$ and in $St_2'$.

- Rules in (4), i.e., $St_2 \rightsquigarrow_{\sigma_2, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_2'$ 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 $\sigma_2(\theta(St_1))$ of the form $t{\in}\mathcal{I}$ for $t$ a message term that does not appear in $St_2'$ (modulo $E_{\mathcal{P}}$) and $t =_{E_{\mathcal{P}}} \sigma_2(u)$, where $u$ is the message term used by the rewrite rule. Thus, $\sigma_2(u){\in}\mathcal{I}$ does appear in $\sigma_2(\theta(St_1))$ (modulo $E_{\mathcal{P}}$). That is, the same narrowing step is available from $\sigma_2(\theta(St_1))$ and there exist $\sigma_1, \rho$ such that $St_1 \rightsquigarrow_{\sigma_1, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_1'$ with the same rule and $\theta \circ \sigma_2 =_{E_{\mathcal{P}}} \sigma_1 \circ \rho$. Thus, either $St_1'$ is an initial state or $\rho(St_1') \triangleright St_2'$.

This concludes the proof. □

Second, we examine the case in which a non-initial strand in $St_2'$ does not appear in $t\sigma_2(\theta(St_1))$.

**Lemma 3.** *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 (i) there is a substitution $\theta$ s.t. $\theta(St_1) \triangleright St_2$, (ii) there is a narrowing step $St_2 \leadsto_{\sigma_2, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_2'$, and (iii) there is a non-initial strand $[m_1^\pm, \ldots, m_i^\pm \mid m_{i+1}^\pm, \ldots, m_n^\pm]$ in $\sigma_2(\theta(St_1))$ that does not appear in $St_2'$ (modulo $E_\mathcal{P}$), then (a) $\sigma_2|_{\mathcal{V}ar(St_2)} = id$, (b) $[m_1^\pm, \ldots, m_{i-1}^\pm \mid m_i^\pm, \ldots, m_n^\pm]$ does appear in $St_2'$ (modulo $E_\mathcal{P}$) and (c) there is a state $St_1'$ such that $St_1 \leadsto_{id, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_1'$ and either $St_1'$ is an initial state or $St_1' \triangleright St_2'$.*

*Proof.* We prove the result by considering the different rules applicable to $St_2$ (remember that in $\mathcal{R}$, rewriting and narrowing steps always happen at the top position). Note that property (a) is immediate because rules in $R_\mathcal{P}$ do not remove strands, only move the vertical bar to the left of the sequences of messages in the strands. Note also that if $[m_1^\pm, \ldots, m_i^\pm \mid m_{i+1}^\pm, \ldots, m_n^\pm]$ appears in $\sigma_2(\theta(St_1))$ and $[m_1^\pm, \ldots, m_{i-1}^\pm \mid m_i^\pm, \ldots, m_n^\pm]$ appears in $St_2'$, then only Rule (2) or Rule (5) have been applied to $St_2$ as follows:

- Reversed version of Rule (2), i.e., $St_2 \leadsto_{\sigma_2, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_2'$ using the following rule
$$[L, M^+ \mid L'] \,\&\, SS \,\&\, IK \to [L \mid M^+, L'] \,\&\, SS \,\&\, IK.$$
- Reversed version of Rule (5), i.e., $St_2 \leadsto_{\sigma_2, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_2'$ using the following rule
$$[L, M^- \mid L'] \,\&\, SS \,\&\, IK \to [L \mid M^-, L'] \,\&\, SS \,\&\, (M \in \mathcal{I}, IK).$$

However, note that $\sigma_2|_{\mathcal{V}ar(St_2)} = id$ in both possible rewrite steps. Then, there is a state $St_1'$ such that $St_1 \leadsto_{id, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_1'$ with the same rule and it is straightforward that either $St_1'$ is an initial state or $St_1' \blacktriangleright St_2'$, since only the vertical bar has been moved. □

Now we can formally define the relation between $\mathcal{P}$-subsumption and one narrowing step. In the following, $\leadsto^{\{0,1\}}_{\sigma, R_\mathcal{P}^{-1}, E_\mathcal{P}}$ denotes zero or one narrowing step.

**Lemma 4.** *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 $St_1 \blacktriangleright St_2$ and $St_2 \leadsto_{\sigma_2, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_2'$, then there is a state $St_1'$ and a substitution $\sigma_1$ such that $St_1 \leadsto^{\{0,1\}}_{\sigma_1, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_1'$ and either $St_1'$ is an initial state or $St_1' \blacktriangleright St_2'$.*

*Proof.* Since $St_1 \blacktriangleright St_2$, there is a substitution $\theta$ s.t. $\theta(St_1) \triangleright St_2$. If each intruder fact of the form $t \in \mathcal{I}$ in $\sigma_2(\theta(St_1))$ appears in $St_2'$ (modulo $E_\mathcal{P}$) and each non-initial strand in $\sigma_2(\theta(St_1))$ appears in $St_2'$ (modulo $E_\mathcal{P}$), then, by Lemma 1, $\sigma_2(\theta(St_1)) \triangleright St_2'$, i.e., $St_1 \blacktriangleright St_2'$. Otherwise, Lemma **??** states that 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 $E_\mathcal{P}$), or (b) there is a non-initial strand in $\sigma_2(\theta(St_1))$

that does not appear in $St_2'$ (modulo $E_{\mathcal{P}}$). For case (a), by Lemma 2, there is a state $St_1'$ and a substitution $\sigma_1$ such that $St_1 \leadsto_{\sigma_1, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_1'$ and either $St_1'$ is an initial state or there is a substitution $\rho$ s.t. $\rho(St_1') \triangleright St_2'$. For case (b), by Lemma 3, $\sigma_2|_{\mathcal{V}ar(St_2)} = id$, and there is a state $St_1'$ such that $St_1 \leadsto_{id, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_1'$ and either $St_1'$ is an initial state or $St_1' \triangleright St_2'$, i.e., $St_1' \blacktriangleright St_2'$. □

Preservation of reachability follows from the following main theorem. Note that the relation $\blacktriangleright$ is applicable only to non-initial states, whereas the relation $\subseteq_{E_{\mathcal{P}}}$ of Definition 3 is applicable to both initial and non-initial states.

**Theorem 3.** *Given a topmost rewrite theory $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, R_{\mathcal{P}})$ representing protocol $\mathcal{P}$ and two states $St_1, St_2$. If $St_1 \blacktriangleright St_2$, $St_2^{ini}$ is an initial state, and $St_2 \leadsto^*_{\sigma_2, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_2^{ini}$, then there is an initial state $St_1^{ini}$ and substitutions $\sigma_1$ and $\theta$ such that $St_1 \leadsto^*_{\sigma_1, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_1^{ini}$, and $\theta(St_1^{ini}) \subseteq_{E_{\mathcal{P}}} St_2^{ini}$.*

*Proof.* Consider $St_2 = U_0$, $St_2^{ini} = U_n$, $\sigma_2 = \rho_1 \cdots \rho_n$, and $U_0 \leadsto^n_{\rho_i, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} U_n$. Note that $n \neq 0$, since $St_2$ cannot be an initial state because $St_1 \blacktriangleright St_2$ implies that both $St_1$ and $St_2$ are not initial states. Then, by Lemma 4, there is $j \leq n$ such that for each $i < j$, $U_{i-1} \leadsto_{\rho_i, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} U_i$ and there is a step $U_{i-1}' \leadsto_{\rho_i', R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} U_i'$ s.t. $U_i' \blacktriangleright U_i$. Note that $U_j'$ is an initial state and there is a substitution $\theta$ s.t. $\theta(U_j') \subseteq_{E_{\mathcal{P}}} U_j \subseteq_{E_{\mathcal{P}}} U_n$. □

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

## 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,[5] 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, after the name *lazy intruder* coined by Basin et al. [3] to describe another optimization technique that involves delaying instantiation of variables.

To see how super-lazy terms arise, we consider the following example.

[5] 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.

**Example 7.** *Consider again the attack pattern* (†) *in Example 2. After a couple of backwards narrowing steps, the Maude-NPA finds the following state that describes how the intruder can learn* $sec(a, r'')$ *by assuming he can learn a message* $e(K, sec(a, r''))$ *and the key* $K$*:*

$$\begin{array}{l} [\ nil \mid K^-,\ e(K, sec(a, r'')))^-,\ sec(a, r'')^+\ ]\& \\ :: r' :: \\ [\ (A; B; E')^-, (B; A; exp(g, n(B, r')))^+ \mid (e(exp(E', n(B, r')), sec(a, r'')))^-\ ]\& \\ (e(exp(E', n(B, r')), sec(a, r'')) \in \mathcal{I},\ K \in \mathcal{I},\ e(K, sec(a, r'')) \in \mathcal{I},\ sec(a, r'') \notin \mathcal{I}) \end{array} \quad (\natural)$$

*Here variable* $K$ *is a super-lazy term. The intruder can find it by instantiating it by any term of sort key from its initial knowledge, so we drop* $K \in \mathcal{I}$ *from the state description.*

Dropping super-lazy terms is complete by Theorem 3, but if we drop them permanently we lose soundness. If the variables used in creating those terms appear elsewhere in the state, they may become instantiated as the backwards search continues. In that case, the super-lazy terms that were deleted may no longer be trivial to find. This may result in the construction of narrowing sequences from the state that has the super-lazy terms removed to an initial state, that does not correspond to any narrowing sequence that could be obtained if the terms had been retained.

**Example 8.** *Consider the state described in Example 7. Affter some more backwards narrowing steps, the tool unifies message* $e(K, sec(a, r'')))$ *with an output message* $e(exp(\overline{X}, n(\overline{A}, \overline{r})), sec(\overline{A}, \overline{r_2}))$ *of an explicitly added Bob's strand of the form*

$$\begin{array}{l} :: \overline{r_1}, \overline{r_2} :: \\ [\ (\overline{A}; \overline{B}; exp(g, n(\overline{A}, \overline{r_1})))^+,\ (\overline{B}; \overline{A}; \overline{X})^-,\ (e(exp(\overline{X}, n(\overline{A}, \overline{r})), sec(\overline{A}, \overline{r_2})))^+] \end{array}$$

*thus getting an instantiation for the super-lazy term* $K$*, namely* $\{K \mapsto exp(\overline{X}, n(\overline{A}, \overline{r}))\}$*. Now the intruder can no longer construct* $K$ *out of terms in its initial knowledge, because* $n(\overline{A}, \overline{r})$ *is not in its initial knowledge.*

Since we intend Maude-NPA to be both sound and complete, we elect to remove super-lazy terms only temporarily. When super-lazy terms are deleted from a state, a copy of the original state known as a *ghost state* is retained. The variables in the super-lazy terms are monitored to determine whether or not they become instantiated during a narrowing step. If that is a case, the state resulting from this narrowing step is deleted and replaced with the ghost state with the super-lazy terms instantiated.

The operation of resuscitating the ghost state is complex; in particular care must be taken to avoid interaction with subsumption partial order reduction. Removing super-lazy terms from a state affects its status in the subsumption

partial order, and it is again affected when a ghost is resuscitated. The result is that, if we wish to allow states with ghosts to participate in the subsumption partial order reduction, we must proceed very carefully. In particular, if we apply the subsumption partial order reduction indiscriminately, a resuscitated ghost state will be dominated in the partial order by the ancestor that introduced the ghost, and so will be removed. Thus, we have implemented a procedure for identifying the ancestor of a resuscitated ghost state when checking the subsumption partial order. See Sections 7.4 and 7.5.

The remainder of this section is organized as follows. In Section 7.1 we give a formal definition of super-lazy terms. In Section 7.2 we describe the procedure of creating and resuscitating ghost states. In Section 7.3 we describe an optimization of the super-lazy intruder that allows one to identify cases in which super-lazy terms will never be further instantiated, and thus can be removed without creating a ghost state. Finally, in Section 7.4 we describe how harmful interaction between the subsumption partial order and the super-lazy intruder is handled.

*7.1. Definition of Super-Lazy Terms*

The set $\mathcal{L}(St)$ of super-lazy terms w.r.t. a state $St$ is inductively generated as a subset $\mathcal{L}(St) \subseteq \mathcal{T}_\Omega(Y \cup IK_0)$ 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 $\mathcal{L}(St)$ the set $IK^{\not\in}(St)$ of terms that the intruder does not know and all its possible combinations with symbols in $\Omega$.

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

$$\Omega = \{f : \mathsf{s}_1 \cdots \mathsf{s}_\mathsf{n} \to \mathsf{s} \mid [(X_1{:}\mathsf{s}_1)^-, \ldots, (X_k{:}\mathsf{s}_\mathsf{k})^-, (f(X_1{:}\mathsf{s}_1, \ldots, X_k{:}\mathsf{s}_\mathsf{k}))^+] \in \mathcal{S}_\mathcal{P}\}.$$

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

1. *$IK_0 \subseteq \mathcal{L}(St)$,*
2. *$\mathcal{V}ar(St) - IK^{\not\in}(St) \subseteq \mathcal{L}(St)$,*
3. *for each $f : \mathsf{s}_1 \cdots \mathsf{s}_\mathsf{n} \to \mathsf{s} \in \Omega$ and for all $t_1{:}\mathsf{s}_1, \ldots, t_k{:}\mathsf{s}_\mathsf{k} \in \mathcal{L}(St)$, if $f(t_1{:}\mathsf{s}_1, \ldots, t_k{:}\mathsf{s}_\mathsf{k}) \not\in IK^{\not\in}(St)$, then $f(t_1{:}\mathsf{s}_1, \ldots, t_k{:}\mathsf{s}_\mathsf{k}) \in \mathcal{L}(St)$.*

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 public operations $e$ and $\_;\_$ available to the intruder.

*7.2. The Super-Lazy Intruder and Ghost States*

Let us first briefly explain how the ghost state mechanism works before formally describing it. A *ghost state* is a state extended to allow expressions of the form 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 $t{\in}\mathcal{I}$ in the intruder's knowledge, we replace the intruder fact $t{\in}\mathcal{I}$ in $St$ by $ghost(t)$ and keep the ghost version of $St$ in the history of states used by the transition subsumption of Section 6.3. For instance, the state ($\natural$) of Example 7 with a super-lazy intruder term $K$ would be represented as follows, where we have just replaced $K{\in}\mathcal{I}$ by $ghost(K)$:

$$
\begin{array}{l}
[\ nil \mid K^-,\ e(K, sec(a, r''))^-,\ sec(a, r'')^+\ ]\& \\
::\ r' :: [\ (A; B; E')^-, (B; A; exp(g, n(B, r')))^+ \mid (e(exp(E', n(B, r')), sec(a, r''))^-\ ]\& \\
(\mathrm{ghost}(K),\ e(exp(E', n(B, r')), sec(a, r''))\in\mathcal{I},\ e(K, sec(a, r''))\in\mathcal{I},\ sec(a, r'')\notin\mathcal{I})
\end{array}
$$

Suppose that later in the search tree we find a descendant $St'$ in which $ghost(u)$ has been instantiated to $ghost(t)$ where $t$ is not a super lazy intruder term. For the intruder to learn such a term $t$, if may be necessary for certain actions to occur before $St'$ was produced. That is, we "roll back" and replace the current state $St'$, containing expression $ghost(t)$, by an instantiated version of its ancestor state $St$, namely $\theta(St)$, where $t =_{E_\mathcal{P}} \theta(u)$. This is explained in detail in Definition 11 below.

A complication is introduced if the substitution $\theta$ binding variables in $u$ includes variables of sort Fresh. These must have been introduced by strands indexed by these fresh variables. If the strand indexed by a fresh variable in $t$ already appears in $St$, then there is no problem. However, if the strand was introduced later in the backwards narrowing process, and we do not include them in $St$, then this will result in difficulties. If such a strand is not in the reactivated version of $St$, it will not be re-introduced in the backwards narrowing search, because the fresh variables in newly introduced strands are non-unifiable with any of the fresh variables already present. 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 $ghost(t)$ 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 Alices's strand (after making the appropriate substitutions) with the vertical bar at the rightmost position because it is the strand generating the* Fresh *variable* $r''$*:*

$$
\begin{array}{l}
[\ nil \mid exp(\overline{E}, n(a, \overline{r_1}))^-,\ e(exp(\overline{E}, n(a, \overline{r_1}), sec(a, \overline{r_2})))^-,\ sec(a, \overline{r_2})^+\ ]\& \\
:: \overline{r_1}, \overline{r_2} :: \\
[\ (a; B'; exp(g, n(a, \overline{r_1}))^+,\ (B'; a; E)^-,\ (e(exp(E, n(a, \overline{r_1})), sec(a, \overline{r_2})))^+ \mid nil\ ]\& \\
:: r' :: [\ (A; B; E')^-, (B; A; exp(g, n(B, r')))^+ \mid (e(exp(E', n(B, r')), sec(a, \overline{r_2})))^-\ ]\& \\
(e(exp(E', n(B, r')), sec(a, \overline{r_2}))\in\mathcal{I},\ exp(\overline{E}, n(a, \overline{r_1}))\in\mathcal{I}, \\
e(exp(\overline{E}, n(a, \overline{r_1})), sec(a, \overline{r_2})))\in\mathcal{I},\ sec(a, \overline{r_2})\notin\mathcal{I})
\end{array}
$$

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:

$$\{\ [\, l_1 \,|\, u^+ \,] \,\&\, \{u\notin\mathcal{I}, K\} \rightarrow \{u\in\mathcal{I}, K\}\ \mid\ [\, l_1,\, u^+,\, l_2\,] \in \mathcal{S}_\mathcal{P}\} \tag{6}$$

Note that rules of type (4) introduce strands $[\, l_1 \mid u^+, l_2\,]$, whereas here rules of type (6) introduce strands $[\, l_1 \mid u^+\,]$. 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 $\widetilde{R_\mathcal{P}} = \{(5), (2), (3)\} \cup (6)$.

First, we define a new relation $\sqsubseteq_{E_\mathcal{P}}$ between states, which is similar to $\subseteq_{E_\mathcal{P}}$ of Definition 3 but considers partial strands.

**Definition 7 (Partial Inclusion).** *Given two states* $St_1, St_2$*, we abuse notation and write* $St_1 \sqsubseteq_{E_\mathcal{P}} St_2$ *to denote that every intruder fact in* $St_1$ *appears in* $St_2$ *(modulo* $E_\mathcal{P}$*) and that every strand* $[m_1^\pm, \ldots, m_k^\pm]$ *in* $St_1$*, either appears in* $St_2$ *(modulo* $E_\mathcal{P}$*) or there is* $i \in \{1, \ldots, k\}$ *s.t.* $m_i^\pm = m_i^+$ *and* $[m_1^\pm, \ldots, m_i^+]$ *appears in* $St_2$ *(modulo* $E_\mathcal{P}$*).*

The following result ensures that if a state is reachable via backwards reachability analysis using $R_\mathcal{P}$, then it is also reachable using $\widetilde{R_\mathcal{P}}$. Its proof is straightforward, and we omit it.

**Proposition 3.** *Let* $\mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_\mathcal{P}, R_\mathcal{P})$ *be a topmost rewrite theory representing protocol* $\mathcal{P}$*. Let* $St = ss \,\&\, SS \,\&\, (ik, IK)$ *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* $St \rightsquigarrow^*_{\sigma, R_\mathcal{P}^{-1}, E_\mathcal{P}} St_{ini}$*, then there is an initial state* $St'_{ini}$ *and two substitutions* $\sigma'$*,* $\rho$ *such that* $St \rightsquigarrow^*_{\sigma', \widetilde{R_\mathcal{P}}^{-1}, E_\mathcal{P}} St'_{ini}$*,* $\sigma =_{E_\mathcal{P}} \sigma' \circ \rho$*, and* $\rho(St'_{ini}) \sqsubseteq_{E_\mathcal{P}} St_{ini}$*.*

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

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

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

**Definition 9 (Strand Reset).** *Given a strand* $s$ *of the form* $:: r_1, \ldots, r_k :: [m_1^{\pm}, \ldots \mid \ldots, m_n^{\pm}]$*, when we want to move the bar to the rightmost position (denoting a final strand), we write* $s{\gg} =:: r_1, \ldots, r_k :: [m_1^{\pm}, \ldots, m_n^{\pm} \mid nil]$*.*

**Definition 10 (Fresh Generating Strands).** *Given a state* $St$ *containing an intruder fact* $ghost(t)$ *for some term* $t$ *with variables, we define the set of strands associated to* $t$*, denoted* $\mathrm{strands}_{St}(t)$*, as follows:*

- *for each strand* $s$ *in* $St$ *of the form* $:: r_1, \ldots, r_k :: [m_1^{\pm}, \ldots \mid \ldots, m_n^{\pm}]$*, if there is* $i \in \{1, \ldots, k\}$ *s.t.* $r_i \in \mathcal{V}ar(t)$*, then* $s{\gg}$ *is included into* $\mathrm{strands}_{St}(t)$*; or*
- *for each strand* $s$ *in* $St$ *of the form* $:: r_1, \ldots, r_k :: [m_1^{\pm}, \ldots \mid \ldots, m_n^{\pm}]$*, if there is another strand* $s'$ *of the form* $:: r'_1, \ldots, r'_{k'} :: [w_1^{\pm}, \ldots \mid \ldots, w_{n'}^{\pm}]$ *in* $\mathrm{strands}_{St}(t)$*, and there are* $i \in \{1, \ldots, k\}$ *and* $j \in \{1, \ldots, n'\}$ *s.t.* $r_i \in \mathcal{V}ar(w_j)$*, then* $s{\gg}$ *is included into* $\mathrm{strands}_{St}(t)$*.*

Now, we formally define how to resuscitate a state.

**Definition 11 (Resuscitation).** *Given a topmost rewrite theory* $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, \widetilde{R_{\mathcal{P}}})$ *representing protocol* $\mathcal{P}$ *and a state* $St$ *containing an intruder fact* $t{\in}\mathcal{I}$ *such that* $t$ *is a super-lazy term, i.e.,* $St = ss \,\&\, (t{\in}\mathcal{I}, ik)$ *where* $ss$ *is a term denoting a set of strands and* $ik$ *is a term denoting the rest of the intruder knowledge. Let* $\overline{St}$ *be the ghost version of* $St$*. Let* $St'$ *be a state such that* $\overline{St} \rightsquigarrow^*_{\sigma, \widetilde{R_{\mathcal{P}}}^{-1}, E_{\mathcal{P}}} St'$ *and* $\sigma(t)$ *is not a super-lazy term. Let* $\sigma_t = \sigma|_{\mathcal{V}ar(t)}$*. The* reactivated *(or* resuscitated*) version of* $St$ *w.r.t. state* $St'$ *and substitution* $\sigma_t$ *is defined as* $\overline{St} = \sigma_t(ss) \,\&\, \sigma_t((t{\in}\mathcal{I}, ik)) \,\&\, \mathrm{strands}_{St'}(\sigma_t(t))$*.*

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

**Theorem 4.** *Given a topmost rewrite theory* $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, \widetilde{R_{\mathcal{P}}})$ *representing protocol* $\mathcal{P}$ *and a state* $St$ *containing an intruder fact* $t{\in}\mathcal{I}$ *such that* $t$ *is a super-lazy term, if there exist an initial state* $St_{ini}$ *and substitution* $\theta$ *such that* $St \rightsquigarrow^*_{\theta, \widetilde{R_{\mathcal{P}}}^{-1}, E_{\mathcal{P}}} St_{ini}$*, then (i) there exist a state* $St'$ *and substitutions* $\tau, \tau'$ *such that* $St \rightsquigarrow^*_{\tau, \widetilde{R_{\mathcal{P}}}^{-1}, E_{\mathcal{P}}} St'$*,* $\theta =_{E_{\mathcal{P}}} \tau \circ \tau'$*, and* $\tau(t)$ *is not a super-lazy term, and (ii) there exist a reactivated version* $\widehat{St}$ *of* $St$ *w.r.t.* $St'$ *and* $\tau$*, an initial state* $St'_{ini}$*, and substitutions* $\theta'$*,* $\rho$ *such that* $\widehat{St} \rightsquigarrow^*_{\theta', \widetilde{R_{\mathcal{P}}}^{-1}, E_{\mathcal{P}}} St'_{ini}$*,* $\theta =_{E_{\mathcal{P}}} \theta' \circ \rho$*, and* $\rho(St'_{ini}) \subseteq_{E_{\mathcal{P}}} St_{ini}$*.*

*Proof.* The sequence from $St$ to $St_{ini}$ 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 $St'$ and substitutions $\tau$, $\tau'$ such that $\tau(t)$ is not a super-lazy term, $\theta = \tau \circ \tau'$, $St \rightsquigarrow^*_{\tau,\widetilde{R_\mathcal{P}}^{-1},E_\mathcal{P}} St' \rightsquigarrow^*_{\tau',\widetilde{R_\mathcal{P}}^{-1},E_\mathcal{P}} St_{ini}$, and the sequence $St \rightsquigarrow^*_{\tau,\widetilde{R_\mathcal{P}}^{-1},E_\mathcal{P}} St'$ can be viewed as $St = St_0 \rightsquigarrow_{\tau_1,\widetilde{R_\mathcal{P}}^{-1},E_\mathcal{P}} \cdots \rightsquigarrow_{\tau_k,\widetilde{R_\mathcal{P}}^{-1},E_\mathcal{P}} St_k = St'$ 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 $\overline{St}$ computing such substitution $\tau$. That is, there is a state $St''$ such that $\overline{St} \rightsquigarrow^*_{\tau,\widetilde{R_\mathcal{P}}^{-1},E_\mathcal{P}} St''$ and $St''$ differs from $St'$ (modulo $E_\mathcal{P}$-equivalence and variable renaming) only in that $\tau(t){\in}\mathcal{I}$ is replaced by $ghost(\tau(t))$. Let $\tau_t = \tau|_{\mathcal{V}ar(t)}$, there exists a substitution $\tau''$ s.t. $\tau =_{E_\mathcal{P}} \tau_t \circ \tau''$. Let $\widehat{St}$ 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 $\widehat{St} \rightsquigarrow^*_{\sigma,\widetilde{R_\mathcal{P}}^{-1},E_\mathcal{P}} St'_{ini}$, $\tau'' \circ \tau' =_{E_\mathcal{P}} \sigma \circ \rho$, and $\rho(St'_{ini}) =_{E_\mathcal{P}} St_{ini}$. □

*7.3. Optimizing 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 $\mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_\mathcal{P}, \widetilde{R_\mathcal{P}})$ representing protocol $\mathcal{P}$, and a state $St$ containing an intruder fact $t{\in}\mathcal{I}$ such that $t$ is a super-lazy term, if for each strand $[m_1^\pm,\ldots,m_{j-1}^\pm \mid m_j^\pm,\ldots,m_k^\pm]$ in $St$ and each $i \in \{1,\ldots,j-1\}$, $\mathcal{V}ar(t) \cap \mathcal{V}ar(m_i) = \emptyset$, and for each term $w{\in}\mathcal{I}$ in the intruder's knowledge, $\mathcal{V}ar(t) \cap \mathcal{V}ar(w) = \emptyset$, then, $t$ is called a* void super-lazy term.

**Proposition 4.** *Given a topmost rewrite theory $\mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_\mathcal{P}, \widetilde{R_\mathcal{P}})$ representing protocol $\mathcal{P}$ and a state $St$ containing an intruder fact $t{\in}\mathcal{I}$ such that $t$ is a void super-lazy term, let $\overline{St}$ 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 $St \rightsquigarrow^*_{\theta,\widetilde{R_\mathcal{P}}^{-1},E_\mathcal{P}} St_{ini}$, then there exist an initial state $St'_{ini}$ and substitutions $\sigma, \rho$ such that $\overline{St} \rightsquigarrow^*_{\sigma,\widetilde{R_\mathcal{P}}^{-1},E_\mathcal{P}} St'_{ini}$, $\theta =_{E_\mathcal{P}} \sigma \circ \rho$, and $\rho(St'_{ini}) \subseteq_{E_\mathcal{P}} St_{ini}$.*

*Proof.* Since $t$ is a super-lazy term, $St_{ini}$ contains a sequence of intruder strands of $\mathcal{S}_\mathcal{P}$ generating $t$. Let $\theta_t = \theta|_{\mathcal{V}ar(t)}$, there exists a substitution $\theta'$ s.t. $\theta =_{E_\mathcal{P}} \theta_t \circ \theta'$. Since $t$ is a void super-lazy term, there is a state $St''_{ini}$ such that $\theta'(\overline{St}) \rightarrow^*_{\widetilde{R_\mathcal{P}}^{-1},E_\mathcal{P}} St''_{ini}$. Then, by narrowing completeness, i.e., Theorem 1, there are an initial state $St'_{ini}$ and substitutions $\sigma, \rho$ such that $\overline{St} \rightsquigarrow^*_{\sigma,\widetilde{R_\mathcal{P}}^{-1},E_\mathcal{P}} St'_{ini}$, $\theta' =_{E_\mathcal{P}} \sigma \circ \rho$, and $\rho(St'_{ini}) \subseteq_{E_\mathcal{P}} St''_{ini}$. Finally, $St''_{ini} \subseteq_{E_\mathcal{P}} St_{ini}$, since $St_{ini}$ simply has the strands generating $t$ that $St''_{ini}$ does not contain. □

*7.4. Transition Subsumption and the Super-Lazy Intruder.*

Transition subsumption in the presence of the lazy intruder is computed for the most part as if the lazy intruder did not exist. That is, we define a partial order on states with ghosts that extends the $\blacktriangleright$ of Section 6.3:

**Definition 13.** *Let* $St_1 = ss_1 \,\&\, (ghost(t_1), \ldots, ghost(t_n), ik_1)$ *and let* $St_2 = ss_2 \,\&\, (ghost(t'_1), \ldots, ghost(t'_m), ik_2)$. *Let* $St'_1 = ss_1 \,\&\, (t_1{\in}\mathcal{I}, \ldots, t_n{\in}\mathcal{I}, ik_1)$ *and let* $St'_2 = ss_2 \,\&\, (t'_1{\in}\mathcal{I}, \ldots, t'_m{\in}\mathcal{I}, ik_2)$. *We say that* $St_1 \blacktriangleright_0 St_2$ *if* $St'_1 \blacktriangleright St'_2$.

However, we cannot use this definition without modification. If we do, then when a ghost state is reactivated, we see from Definition 13 that such a reactivated state will be $\mathcal{P}$-subsumed by the original state that raised the ghost expression. Therefore, we modify $\blacktriangleright_0$ by excluding resuscitated descendants as follows.

**Definition 14 (Resuscitated Descendant).** *Given a topmost rewrite theory* $\mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_\mathcal{P}, \widetilde{R_\mathcal{P}})$ *representing protocol* $\mathcal{P}$ *and two non-initial states* $St$ *and* $St'$ *such that* $St$ *contains an intruder fact* $t{\in}\mathcal{I}$ *and* $t$ *is a super-lazy term, we say* $St'$ *is a* resuscitated descendant *of* $St$, *written* $St \curvearrowright St'$, *if:*

1. *given the ghost version* $\overline{St}$ *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*

$$\overline{St} \rightsquigarrow_{\tau_1, \widetilde{R_\mathcal{P}}^{-1}, E_\mathcal{P}} St_1 \cdots St_{i-1} \rightsquigarrow_{\tau_i, \widetilde{R_\mathcal{P}}^{-1}, E_\mathcal{P}} St_i \cdots St_{k-1} \rightsquigarrow_{\tau_k, \widetilde{R_\mathcal{P}}^{-1}, E_\mathcal{P}} St_k,$$

   $\tau_1 \circ \cdots \circ \tau_j(t)$ *is a super-lazy term for* $1 \leq j \leq i-1$, *and* $\tau_1 \circ \cdots \circ \tau_i(t)$ *is not a super-lazy term, and*
2. *given the reactivated version* $\widetilde{St}$ *of* $St$ *w.r.t.* $St_i$ *and* $\tau = \tau_1 \circ \cdots \circ \tau_i$ *and* $\tau_t = \tau|_{\mathcal{V}ar(t)}$, *there exist a positive integer* $k$ *and substitutions* $\tau'_1, \ldots, \tau'_k$ *such that* $\tau_j = \tau_t \circ \tau'_j$ *for* $1 \leq j \leq k$, *states* $St'_1, \ldots, St'_k$, *and a narrowing sequence*

$$\widetilde{St} \rightsquigarrow_{\tau'_1, \widetilde{R_\mathcal{P}}^{-1}, E_\mathcal{P}} St'_1 \cdots St'_{k-1} \rightsquigarrow_{\tau'_k, \widetilde{R_\mathcal{P}}^{-1}, E_\mathcal{P}} St'_k$$

*then there is* $j \in \{1, \ldots, k\}$ *such that* $St' =_{E_\mathcal{P}} St'_j$.

**Proposition 5 ($\mathcal{P}$-subsumption relation I).** *Given a topmost rewrite theory* $\mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_\mathcal{P}, \widetilde{R_\mathcal{P}})$ *representing protocol* $\mathcal{P}$, *let* $\ltimes$ *be a partial order on states such that* $St \ltimes St'$ *implies that* $St \blacktriangleright_0 St'$ *and* $St \not\curvearrowright St'$. *Then the partial order reduction imposed by* $\ltimes$ *preserves soundness and completeness reachability.*

*7.5. Implementing Subsumption Partial Order Reduction in the Presence of the Super-Lazy Intruder*

In this section we describe how the subsumption partial order is actually implemented in Maude-NPA in the presence of the super-lazy intruder. Proposition 5 gives a simple way of doing this, but is not very efficient if implemented

in a straightforward way, since it requires extensive examination of the search tree, examining not only the two states being compared but the narrowing path between them. Instead, we use an approximation of the $\curvearrowright$ relation that can be computed directly via a syntactic check on the state information.

This section is very technical and depends heavily on details about Maude-NPA. The reader who is interested mainly in understanding in the basic principals of the super-lazy intruder, and is not concerned about how it is actually implemented in Maude-NPA, is encouraged to skip it. However, we believe that it is valuable because it not only documents what actually is implemented in the tool, but demonstrates how one can use approximation to maximize state space reduction while minimizing the amount of search tree examination required.

In order make the presentation easier to follow, we describe our approximation in terms of a series of approximations, each one of which is closer to the relation used in Proposition 5.

To begin with, we extend protocol states to include the actual message exchange sequence between principal or intruder strands and add a new expression $resuscitated(m)$ to indicate when a state has been resuscitated. This information, except for $resuscitated(m)$, was already included in Maude-NPA states, but its purpose had been to assist the user in reconstructing attacks, not in performing the search itself.

The set of rewrite rules is extended to compute the exchange sequence as follows, where $X$ is a variable denoting an exchange sequence:

$$
\begin{array}{l}
[L \mid M^-, L'] \;\&\; SS \;\&\; (M{\in}\mathcal{I}, IK) \;\&\; (M^-, X) \rightarrow [L, M^- \mid L'] \;\&\; SS \;\&\; (M{\in}\mathcal{I}, IK) \;\&\; X \\
[L \mid M^+, L'] \;\&\; SS \;\&\; IK \qquad\quad \&\; (M^+, X) \rightarrow [L, M^+ \mid L'] \;\&\; SS \;\&\; IK \;\&\; X \\
[L \mid M^+, L'] \;\&\; SS \;\&\; (M{\notin}\mathcal{I}, IK) \;\&\; (M^+, X) \rightarrow [L, M^+ \mid L'] \;\&\; SS \;\&\; (M{\in}\mathcal{I}, IK) \;\&\; X \\
\text{for each } [\, l_1,\ u^+,\ l_2 \,] \in \mathcal{S}_\mathcal{P} : [\, l_1 \mid u^+, l_2 \,] \;\&\; SS \;\&\; (u{\notin}\mathcal{I}, IK) \;\&\; (u^+, X) \rightarrow SS \;\&\; (u{\in}\mathcal{I}, IK) \;\&\; X
\end{array}
$$

Completeness reachability and soundness is clearly preserved for this set of rules and for the obvious extensions to $\overline{R_\mathcal{P}}$ and $\widetilde{R_\mathcal{P}}$. To give an example, the resuscitated state of Example 9 will be written as follows, where the resuscitated message is the first item in the exchange sequence:

$$
\begin{array}{l}
[\ nil \mid exp(E, n(a,r))^-,\ e(exp(E, n(a,r), sec(a,r'')))^-,\ sec(a,r'')^+\ ] \;\& \\
::\ r, r''\ :: \\
[\ (a; B'; exp(g, n(a,r)))^+,\ (B'; a; E)^-,\ (e(exp(E, n(a,r)), sec(a,r'')))^+ \mid nil\ ] \;\& \\
::\ r'\ ::\ [\ (A; B; E')^-, (B; A; exp(g, n(B,r')))^+ \mid (e(exp(E', n(B,r')), sec(a,r'')))^-\ ] \;\& \\
(e(exp(E', n(B,r')), sec(a,r''))\in\mathcal{I},\ exp(E, n(a,r))\in\mathcal{I}, \\
\ e(exp(E, n(a,r)), sec(a,r')))\in\mathcal{I},\ sec(a,r'')\notin\mathcal{I}) \;\& \\
(resuscitated(exp(E, n(a,r))),\ exp(E, n(a,r)))^-,\ e(exp(E, n(a,r)), sec(a,r')))^-, \\
\ (sec(a,r''))^+,\ (exp(E', n(b,r')))^-,\ (sec(a,r''))^-,\ (e(exp(E', n(b,r')), sec(a,r')))^+, \\
\ (e(exp(E', n(b,r')), sec(a,r')))^-)
\end{array}
$$

In [13], we provided a very simple rule for approximating Definition 14.

**Definition 15.** *Given a topmost rewrite theory $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, \widetilde{R_{\mathcal{P}}})$ representing protocol $\mathcal{P}$ and two non-initial states $St_1, St_2$, we write $St_1 \dashrightarrow St_2$ if either $St_1$ does not contain an expression* ghost$(m)$ *for a message term $m$ or $St_1$ does contain an expression* ghost$(m)$ *for a message term $m$ but $St_2$ does not contain the expression $resuscitated(m)$.*

The following result establishes that $\dashrightarrow$ is an approximation of $\curvearrowright$. The proof is straightforward.

**Lemma 5.** *Given a topmost rewrite theory $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, \widetilde{R_{\mathcal{P}}})$ representing protocol $\mathcal{P}$ and two non-initial states $St_1, St_2$, if $St_1 \curvearrowright St_2$, then $St_1 \dashrightarrow St_2$.*

Now, we can provide a better transition subsumption relation.

**Proposition 6 ($\mathcal{P}$-subsumption relation II).** *Given a topmost rewrite theory $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, \widetilde{R_{\mathcal{P}}})$ representing protocol $\mathcal{P}$, let $\ltimes_{II}$ be a partial order on states such that $St \ltimes_{II} St'$ implies that $St \blacktriangleright_0 St'$ and there is a substitution $\theta$ s.t. $\theta(St) \not\dashrightarrow St'$. Then the partial order reduction imposed by $\ltimes_{II}$ preserves completeness reachability.*

Reachability completeness is straightforward from Lemma 5 and Proposition 5, since $St_1 \not\dashrightarrow St_2$ implies $St_1 \not\curvearrowright St_2$.

Though this method solves the problem, since it is safe when a state $St'$ is discarded by $St \ltimes_{II} St'$, and $St \ltimes_{II} St'$ implies $St \ltimes St'$ but not vice versa, it disables almost completely benefits of the transition subsumption for those states after a resuscitation, since the relation $\ltimes$ is able to remove many more states than $\ltimes_{II}$ in that case. 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 $(L_1, M_1^-, L_2)$ and it has a ghost expression ghost$(M_1)$, whereas the exchange sequence of the resuscitated version is $(L_1, resuscitated(M_1), M_1^-, L_2)$.

**Definition 16.** *Given a topmost rewrite theory $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, \widetilde{R_{\mathcal{P}}})$ 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 $S_1 \twoheadrightarrow St_2$, 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 ghost$(M_1)$,*
2. *the exchange sequence of state $St_1$ is of the form*

$$(L_1, M_1^-, L_2)$$

3. *the exchange sequence of state $St_2$ is of the form*

$$(L_1', resuscitated(M_2), M_2^-, L_2'),$$

4. *and* $\rho(L_1, M_1^-, L_2) =_{E_\mathcal{P}} (L_1', M_2^-, L_2')$.

Relation $\twoheadrightarrow$ tries to approximate $\curvearrowright$ better than $\dashrightarrow$ but it fails, since it is an over approximation, as shown by the following result, rather than an under approximation, which is necessary for soundness. We need to go even further and restrict $\twoheadrightarrow$ to obtain a closer characterization of $\curvearrowright$.

**Lemma 6.** *Given a topmost rewrite theory* $\mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_\mathcal{P}, \widetilde{R_\mathcal{P}})$ *representing protocol* $\mathcal{P}$ *and two non-initial states* $St_1, St_2$*, if* $St_1 \twoheadrightarrow St_2$*, then* $St_1 \curvearrowright St_2$*.*

Relation $St_1 \twoheadrightarrow St_2$ 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. That is, descendants of the state that produced the ghost state that are instantiations of descendants $St_1$. Intuitively, in the following definition below, we compare the exchange sequences of the two states to see whether the exchange sequence of the first is $(L_1, L_2, M_1^-, L_3)$ and it has a ghost expression $\mathrm{ghost}(M_1)$, whereas the exchange sequence of the second is $(L_1, M_1^+, L_2, resuscitated(M_1), M_1^-, L_3)$. 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* $\mathcal{R}_\mathcal{P} = (\Sigma_\mathcal{P}, E_\mathcal{P}, \widetilde{R_\mathcal{P}})$ *representing protocol* $\mathcal{P}$ *and two non-initial states* $St_1, St_2$*, we say that* $St_2$ *is a* resuscitated version *of* $St_1$*, written* $S_1 \twoheadrightarrow^+ St_2$*, if either* $S_1 \twoheadrightarrow St_2$ *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* $ghost(M_1)$*,*
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*

$$(L_1', L_1'', M_2^+, L_2', resuscitated(M_2), M_2^-, L_3')$$

4. $\rho(L_2, M_1^-, L_3) =_{E_\mathcal{P}} (L_2', M_2^-, L_3')$
5. *and either*
   (a) *there is a subsequence* $L_1'''$ *of* $L_1'$ *such that* $\rho(L_1) =_{E_p} (L_1''')$ *or*
   (b) $St_1' \twoheadrightarrow^+ St_2'$ *where* $St_1'$ *is* $St_1$ *without the* $ghost(M_1)$ *expression and* $St_2'$ *is* $St_2$ *with the shorter exchange sequence* $(L_1', L_2', M_2^-, L_3')$*.*

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

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

Now, we can provide a better transition subsumption relation.

| Protocol | none | | | | | All optimizations | | | | | % |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Needham-Shroeder Public Key | 5 | 19 | 142 | 727 | 4904 | 4 | 6 | 4 | 2 | 1 | 99 |
| Needham-Shroeder Lowe's fix | 5 | 19 | 142 | 727 | 4902 | 4 | 7 | 6 | 2 | - | 81 |
| SecReT06 | 1 | 6 | 22 | 111 | 312 | 2 | 3 | 2 | - | - | 98 |
| SecReT07 | 8 | 24 | 212 | 902 | 8047 | 5 | 1 | 1 | 1 | - | 99 |
| Diffie-Hellman | 3 | 24 | 72 | 316 | 1884 | 4 | 6 | 10 | 9 | 12 | 98 |
| Homo-hpc | 1 | 8 | 22 | 100 | 533 | 2 | 2 | 1 | 1 | 1 | 98 |
| Homo-NSL | 4 | 15 | 92 | 418 | 2409 | 4 | 9 | 10 | 9 | 11 | 98 |
| Amended Needham-Shroeder | 1 | 8 | 24 | 121 | 781 | 2 | 4 | 9 | 20 | 41 | 91 |
| Carlsen's Secret Key Initiator Protocol | 5 | 19 | 145 | 764 | 5931 | 4 | 7 | 10 | 18 | 12 | 99 |
| Denning-Sacco | 6 | 16 | 65 | 357 | 1628 | 1 | 2 | 3 | 5 | 7 | 99 |
| ISO-5-Pass Authentication | 5 | 19 | 145 | 766 | 5982 | 5 | 5 | 13 | 19 | 20 | 99 |
| Kao Chow Repeated Authentication | 5 | 19 | 144 | 795 | 6059 | 4 | 7 | 9 | 11 | 5 | 99 |
| Kao Chow Repeated Authentication-Handshake Key | 1 | 7 | 22 | 99 | 555 | 2 | 2 | 6 | 12 | 18 | 94 |
| Kao Chow Repeated Authentication-Ticket | 5 | 19 | 139 | 715 | 5382 | 8 | 35 | 33 | 27 | 118 | 96 |
| NSL-ECB-homo | 5 | 18 | 116 | 532 | 3006 | 3 | 8 | 15 | 16 | 10 | 98 |
| NSL-XOR-modified | 1 | 10 | 42 | 442 | 6184 | 4 | 5 | 5 | 5 | 2 | 99 |
| Otway-Rees | 1 | 7 | 18 | 55 | 237 | 2 | 6 | 14 | 34 | 84 | 55 |
| Wide Mouthed Frog | 6 | 31 | 226 | 1492 | 11788 | 6 | 13 | 27 | 44 | 65 | 98 |
| Woo-Lam | 1 | 8 | 24 | 115 | 936 | 3 | 5 | 6 | 5 | 6 | 97 |

Table 1: Number of new states produced in each of 1,2,3,4 and 5 backwards narrowing steps comparing each optimization of Sections 5.1,5.2,6.2,6.3, and 7.

**Proposition 7 ($\mathcal{P}$-subsumption relation III).** *Given a topmost rewrite theory $\mathcal{R}_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, \widetilde{R_{\mathcal{P}}})$ representing protocol $\mathcal{P}$, let $\ltimes_{III}$ be a partial order on states such that $St \ltimes_{III} St'$ implies that $St \blacktriangleright_0 St'$ and $St \not\twoheadrightarrow^+ St'$. Then the partial order reduction imposed by $\ltimes_{III}$ preserves completeness reachability.*

Finally, reachability completeness is straightforward from Lemma 7 and Proposition 5, since $St_1 \not\twoheadrightarrow^+ St_2$ implies $St_1 \not\curvearrowright St_2$.

## 8. Experimental Evaluation

In order to measure the contribution the various optimization techniques made to the Maude-NPA protocol analysis, we ran Maude-NPA on several example protocols, with no reduction method in place, with a single reduction method in place, and with all the reduction methods in place. Note that Maude-NPA was never intended to run without these optimizations. Maude-NPA will never terminate unless at least the grammars or the subsumption are used and, thus, we also provide information on whether or not we were able to achieve termination, that is, if we reached a depth in which all states are initial. In our experiments, we use the results for the case with no optimizations as a baseline that allows us to compare the different optimization techniques with each other.

In Table 1, we summarize the experimental evaluation of the impact of the different state space reduction techniques for these example protocols searching up to depth 5. We also provide tables for each single reduction method. Table 3 analyzes the most powerful reduction method, grammars, of Section 5.1. Table 4 shows the comparison of the reduction method of Section 5.2. Table 5 analyzes the transition subsumption of Section 6.3, which is one of the most powerful reduction methods. And Table 6 analyzes the also powerful super-lazy intruder reduction method of Section 7.

| Protocol | Is State Space Finite? |
|---|---|
| Needham-Shroeder Public Key | Yes, by Grammars and Subsumption |
| Needham-Shroeder Lowe's fix | Yes, by Grammars and Subsumption |
| SecReT06 | Yes, by Subsumption or (Grammars and Lazy) |
| SecReT07 | Yes, by Subsumption and Lazy |
| Diffie-Hellman | Yes, by Grammars and Subsumption |
| Homo-hpc | Yes, by Grammars and Subsumption |
| Homo-NSL | Yes, by Grammars, Subsumption and Lazy |
| Amended Needham-Shroeder | No and no attack is found |
| Carlsen's Secret Key Initiator Protocol | No and no attack is found |
| Denning-Sacco | Yes, by Grammars, Subsumption and Lazy |
| ISO-5-Pass Authentication | No and no attack is found |
| Kao Chow Repeated Authentication | Yes, Grammars, Subsumption and Lazy |
| Kao Chow Repeated Authentication-Handshake Key | No and no attack is found |
| Kao Chow Repeated Authentication-Ticket | Yes, by Grammars, Subsumption and Lazy |
| NSL-ECB-homo | Yes, by Grammars and Subsumption |
| NSL-XOR-modified | Yes, by Grammars, Subsumption and Lazy |
| Otway-Rees | No, but an authentication attack is found |
| Wide Mouthed Frog | No, but a secrecy attack is found |
| Woo-Lam | Yes, Grammars and Subsumption |

Table 2: Finite state space achieved by reduction techniques

The experiments have been performed on a Linux machine with an Intel Xeon processor using Maude 2.7. All protocol specifications are included in the official Maude-NPA website[6]. Note that the label "-" means that the reachability analysis finished some levels before.

The overall percentage of state-space reduction for each protocol and the average of nearly 95% suggest that our combined techniques are remarkably effective (the reduced number of states is almost only 5% or less of the original number of states).

Table 2 shows whether the protocols have a finite state space or not. We show the different techniques yielding a finite space for each protocol, when it is the case. The use of grammars and the transition subsumption are clearly the most useful techniques in general. Note that grammars are insufficient to achieve termination for the SecReT07 example, while subsumption and the super lazy intruder are essential in this case.

## 9. Comparison with Related Work

As we have remarked earlier, optimization techniques are not always well documented in the literature. However, even so, there are a number of exceptions in which particular techniques have been well documented. In this section we describe some of these techniques and show how the Maude-NPA techniques related to them.

The grammar generation technique, with is taken from the one used in NPA with very little change, can be thought of as implementing something similar to a resolution technique, in which backwards narrowing steps are applied until saturation is achieved. This is probably closest in spirit to the use of resolution to

[6] Available at `http://maude.cs.uiuc.edu/tools/Maude-NPA`.

| Protocol | none | | | | | Grammars | | | | | % |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Needham-Shroeder Public Key | 5 | 19 | 142 | 727 | 4904 | 4 | 12 | 49 | 185 | 769 | 82 |
| Needham-Shroeder Lowe's fix | 5 | 19 | 142 | 727 | 4902 | 4 | 12 | 50 | 190 | 804 | 81 |
| SecReT06 | 1 | 6 | 22 | 111 | 312 | 1 | 2 | 6 | 15 | 36 | 86 |
| SecReT07 | 8 | 24 | 212 | 902 | 8047 | 7 | 21 | 181 | 747 | 6713 | 16 |
| Diffie-Hellman | 3 | 24 | 72 | 316 | 1884 | 3 | 12 | 30 | 80 | 233 | 84 |
| Homo-hpc | 1 | 8 | 22 | 100 | 533 | 1 | 2 | 4 | 10 | 23 | 93 |
| Homo-NSL | 4 | 15 | 92 | 418 | 2409 | 4 | 14 | 78 | 336 | 1671 | 28 |
| Amended Needham-Shroeder | 1 | 8 | 24 | 121 | 781 | 1 | 3 | 7 | 24 | 96 | 85 |
| Carlsen's Secret Key Initiator Protocol | 5 | 19 | 145 | 764 | 5931 | 4 | 13 | 62 | 265 | 1322 | 75 |
| Denning-Sacco | 6 | 16 | 65 | 357 | 1628 | 2 | 4 | 10 | 27 | 54 | 95 |
| ISO-5-Pass Authentication | 5 | 19 | 145 | 766 | 5982 | 4 | 14 | 73 | 322 | 1562 | 71 |
| Kao Chow Repeated Authentication | 5 | 19 | 144 | 795 | 6059 | 4 | 13 | 63 | 271 | 1336 | 75 |
| Kao Chow Repeated Authentication-Handshake Key | 1 | 7 | 22 | 99 | 555 | 1 | 5 | 13 | 40 | 152 | 69 |
| Kao Chow Repeated Authentication-Ticket | 5 | 19 | 139 | 715 | 5382 | 5 | 19 | 139 | 715 | 5382 | 0 |
| NSL-ECB-homo | 5 | 18 | 116 | 532 | 3006 | 2 | 6 | 21 | 75 | 295 | 89 |
| NSL-XOR-modified | 1 | 10 | 42 | 442 | 6184 | 1 | 8 | 34 | 353 | 5193 | 16 |
| Otway-Rees | 1 | 7 | 18 | 55 | 237 | 1 | 3 | 6 | 13 | 44 | 78 |
| Wide Mouthed Frog | 6 | 31 | 226 | 1492 | 11788 | 4 | 18 | 93 | 458 | 2357 | 78 |
| Woo-Lam | 1 | 8 | 24 | 115 | 936 | 1 | 3 | 7 | 23 | 135 | 84 |

Table 3: Number of new states produced in each of 1,2,3,4 and 5 backwards narrowing steps with and without the optimization of Section 5.1.

generate a search space, for example, in ProVerif [5]or SPASS [32]. In these tools actions of principals are represented as Horn clauses, and a form of resolution (resolution with free selection in ProVerif and ordered resolution in SPASS), until saturation is achieved. The application of the technique, of course, is very different, since ProVerif and SPASS use resolution to generate a search space, while in the generation of grammars the goal is to give a finite description of a set of words not learnable by the intruder. It is perhaps closer in intent to a heuristic used in the Scyther tool [7, 8] to recognize cyclical dependencies among terms sent in messages, although Scyther uses a technique that is closer to Maude-NPA's "intruder-learns-only-once" rule.

The partial order reduction giving priority to input messages is an old technique, used, as we have remarked, by NPA, and a similar reduction, giving priority to output messages, is used by Shmatikov and Stern [29] for forward search. It is also mentioned as a general technique in [2].

The use of heuristics to identify unreachable states is probably the least well documented of optimization techniques. However we note that [8] describes a mechanism used in the Scyther tool for identifying the case in which a nonce is used before it has been generated. NPA also had a mechanism for doing this, which was very similar to the one used in Maude-NPA. The main difference with NPA is that our use of the strand space model often allows us to identify these anomalous states before the event in question has actually been produced in the narrowing sequence.

Our super-lazy intruder, as the name suggests, has much in common with the concept of the *lazy intruder* used in constraint-based analysis. The lazy intruder, first proposed by Huima[7][21], and later expanded upon by a num-

[7]although not under that name, which was coined by Basin et al. in [4]

| Protocol | none | | | | | Inconsistency | | | | | % |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Needham-Shroeder Public Key | 5 | 19 | 142 | 727 | 4904 | 5 | 18 | 96 | 318 | 1663 | 63 |
| Needham-Shroeder Lowe's fix | 5 | 19 | 142 | 727 | 4902 | 5 | 18 | 97 | 318 | 1664 | 63 |
| SecReT06 | 1 | 6 | 22 | 111 | 312 | 1 | 6 | 22 | 107 | 290 | 5 |
| SecReT07 | 8 | 24 | 212 | 902 | 8047 | 8 | 22 | 178 | 697 | 6210 | 22 |
| Diffie-Hellman | 3 | 24 | 72 | 316 | 1884 | 3 | 21 | 27 | 132 | 342 | 77 |
| Homo-hpc | 1 | 8 | 22 | 100 | 533 | 1 | 8 | 22 | 91 | 427 | 17 |
| Homo-NSL | 4 | 15 | 92 | 418 | 2409 | 4 | 14 | 60 | 190 | 883 | 60 |
| Amended Needham-Shroeder | 1 | 8 | 24 | 121 | 781 | 1 | 7 | 8 | 59 | 162 | 74 |
| Carlsen's Secret Key Initiator Protocol | 5 | 19 | 145 | 764 | 5931 | 5 | 18 | 100 | 348 | 2182 | 61 |
| Denning-Sacco | 6 | 16 | 65 | 357 | 1628 | 5 | 5 | 29 | 74 | 439 | 73 |
| ISO-5-Pass Authentication | 5 | 19 | 145 | 766 | 5982 | 5 | 18 | 100 | 349 | 2206 | 61 |
| Kao Chow Repeated Authentication | 5 | 19 | 144 | 795 | 6059 | 5 | 18 | 99 | 362 | 2186 | 61 |
| Kao Chow Repeated Authentication-Handshake Key | 1 | 7 | 22 | 99 | 555 | 1 | 7 | 22 | 93 | 476 | 12 |
| Kao Chow Repeated Authentication-Ticket | 5 | 19 | 139 | 715 | 5382 | 5 | 19 | 135 | 677 | 4589 | 13 |
| NSL-ECB-homo | 5 | 18 | 116 | 532 | 3006 | 5 | 17 | 90 | 281 | 1291 | 54 |
| NSL-XOR-modified | 1 | 10 | 42 | 442 | 6184 | 1 | 7 | 10 | 143 | 1422 | 76 |
| Otway-Rees | 1 | 7 | 18 | 55 | 237 | 1 | 6 | 6 | 33 | 65 | 65 |
| Wide Mouthed Frog | 6 | 31 | 226 | 1492 | 11788 | 6 | 29 | 189 | 1183 | 8591 | 26 |
| Woo-Lam | 1 | 8 | 24 | 115 | 936 | 1 | 8 | 24 | 108 | 812 | 12 |

Table 4: Number of new states produced in each of 1,2,3,4 and 5 backwards narrowing steps with and without the optimization of Section 5.2.

ber of others, for example [1, 26, 4, 6], is a technique used in connection with constraint-based protocol analysis. It arises from the fact that the actual value of a certain part of a message may be irrelevant to the receiver, for example, when the receiver will simply pass it on without interpreting it. Determining all the ways an intruder could construct a message would lead to a needless state space explosion. Thus, instead the decision is postponed by replacing the "irrelevant" part of the message with a variable and recording, as a constraint, the information on which knowledge the intruder can used to generate the message. Because the relevant information is carried along with the variable, the solving of the constraint can be delayed until more information is known about how the lazy term will be used by a recipient of a message containing it.

For the super-lazy intruder, we also delay finding how to reach certain "super-lazy" terms in the intruder knowledge about which little is known, until one or more variables in the term is further instantiated. However, our search method differs from that done in constraint-based systems in that the evaluation of a constraint can take place at any point in a search, while in Maude-NPA a search for a term must be executed at the point in the search tree corresponding to the time the term was learned by the intruder. This means that if a search for a term is delayed until it has been further instantiated, the state must be "rolled back" to the point in the search tree at which the term was learned in the intruder.

The super-lazy intruder technique was also used by NPA, but the way in which ghost states were implemented and resuscitated was considerably different. First of all, ghost states were not automatically resuscitated when super-lazy variables were instantiated. It was left to the user to determine whether or not a particular attack path appeared to be valid. If not, the user could elect to resuscitate ghosts in the states leading to the path and redo the analysis.

| Protocol | none | | | | | Subsumption | | | | | % |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Needham-Shroeder Public Key | 5 | 19 | 142 | 727 | 4904 | 5 | 15 | 61 | 107 | 237 | 92 |
| Needham-Shroeder Lowe's fix | 5 | 19 | 142 | 727 | 4902 | 5 | 15 | 61 | 107 | 237 | 92 |
| SecReT06 | 1 | 6 | 22 | 111 | 312 | 1 | 6 | 15 | 31 | 40 | 79 |
| SecReT07 | 8 | 24 | 212 | 902 | 8047 | 6 | 15 | 61 | 165 | 506 | 91 |
| Diffie-Hellman | 3 | 24 | 72 | 316 | 1884 | 2 | 14 | 26 | 102 | 288 | 81 |
| Homo-hpc | 1 | 8 | 22 | 100 | 533 | 1 | 8 | 15 | 72 | 174 | 59 |
| Homo-NSL | 4 | 15 | 92 | 418 | 2409 | 4 | 12 | 41 | 68 | 181 | 89 |
| Amended Needham-Shroeder | 1 | 8 | 24 | 121 | 781 | 1 | 8 | 15 | 70 | 203 | 68 |
| Carlsen's Secret Key Initiator Protocol | 5 | 19 | 145 | 764 | 5931 | 5 | 15 | 69 | 192 | 825 | 83 |
| Denning-Sacco | 6 | 16 | 65 | 357 | 1628 | 6 | 11 | 44 | 102 | 393 | 73 |
| ISO-5-Pass Authentication | 5 | 19 | 145 | 766 | 5982 | 5 | 15 | 69 | 194 | 837 | 83 |
| Kao Chow Repeated Authentication | 5 | 19 | 144 | 795 | 6059 | 5 | 15 | 68 | 194 | 792 | 84 |
| Kao Chow Repeated Authentication-Handshake Key | 1 | 7 | 22 | 99 | 555 | 1 | 7 | 13 | 53 | 144 | 68 |
| Kao Chow Repeated Authentication-Ticket | 5 | 19 | 139 | 715 | 5382 | 5 | 15 | 69 | 204 | 858 | 81 |
| NSL-ECB-homo | 5 | 18 | 116 | 532 | 3006 | 5 | 14 | 55 | 129 | 410 | 83 |
| NSL-XOR-modified | 1 | 10 | 42 | 442 | 6184 | 1 | 8 | 16 | 71 | 152 | 96 |
| Otway-Rees | 1 | 7 | 18 | 55 | 237 | 1 | 7 | 12 | 43 | 104 | 47 |
| Wide Mouthed Frog | 6 | 31 | 226 | 1492 | 11788 | 6 | 20 | 76 | 212 | 721 | 92 |
| Woo-Lam | 1 | 8 | 24 | 115 | 936 | 1 | 8 | 15 | 67 | 246 | 68 |

Table 5: Number of new states produced in each of 1,2,3,4 and 5 backwards narrowing steps with and without the optimization of Section 6.3.

Resuscitating ghost states only at the user's discretion made for a smaller state space, but also made using the tool more difficult to use.

Secondly, unlike Maude-NPA, in which ghost terms are included in the subsumption partial order computation, the partial order in NPA was computed on states without their ghosts. That meant, that when a state $St$ was replaced by a resuscitated ghost, any state $St'$ that $St$ dominated in the subsumption partial order would have to be resuscitated too, since $St'$ might not be dominated by the resuscitated ghost. Since resuscitating ghost states was done only rarely in NPA, this was an acceptable cost. But in Maude-NPA, where ghosts are constantly being resuscitated, a more conservative approach is desirable.

Finally, subsumption partial order reduction is probably closest to a partial order reduction used in the OFMC model checker in connection with the *lazy intruder* technique. This technique, known as *constraint differentiation* [27] works by identifying overlaps between the constraints that arise in the application of the lazy intruder technique. If the constraints belonging to two different states overlap, then the constraints in the overlap are ultimately applied to only one of the states. The subsumption partial order can be thought of as taking a similar approach, in which substitutions are being compared instead of constraints, and instead of identifying overlaps, one identifies cases in which one state subsumes another, which for constraint differentiation would correspond to the case in which one set of constraints contains another. This use of differentiation instead of subsumption may be an interesting avenue to explore for future work in Maude-NPA optimization.

## 10. Concluding Remarks

The Maude-NPA can analyze the security of cryptographic protocols, modulo given algebraic properties of the protocol's cryptographic functions in exe-

| Protocol | none | | | | | Super-lazy Intruder | | | | | % |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Needham-Shroeder Public Key | 5 | 19 | 142 | 727 | 4904 | 5 | 19 | 142 | 726 | 4822 | 1 |
| Needham-Shroeder Lowe's fix | 5 | 19 | 142 | 727 | 4902 | 5 | 19 | 142 | 726 | 4820 | 1 |
| SecReT06 | 1 | 6 | 22 | 111 | 312 | 1 | 6 | 22 | 111 | 306 | 1 |
| SecReT07 | 8 | 24 | 212 | 902 | 8047 | 8 | 22 | 62 | 199 | 648 | 89 |
| Diffie-Hellman | 3 | 24 | 72 | 316 | 1884 | 3 | 24 | 72 | 297 | 1307 | 25 |
| Homo-hpc | 1 | 8 | 22 | 100 | 533 | 1 | 8 | 22 | 100 | 533 | 0 |
| Homo-NSL | 4 | 15 | 92 | 418 | 2409 | 4 | 15 | 92 | 417 | 2335 | 2 |
| Amended Needham-Shroeder | 1 | 8 | 24 | 121 | 781 | 1 | 8 | 24 | 95 | 573 | 25 |
| Carlsen's Secret Key Initiator Protocol | 5 | 19 | 145 | 764 | 5931 | 5 | 15 | 59 | 266 | 1291 | 76 |
| Denning-Sacco | 6 | 16 | 65 | 357 | 1628 | 6 | 16 | 63 | 237 | 1014 | 35 |
| ISO-5-Pass Authentication | 5 | 19 | 145 | 766 | 5982 | 5 | 15 | 59 | 268 | 1307 | 76 |
| Kao Chow Repeated Authentication | 5 | 19 | 144 | 795 | 6059 | 5 | 15 | 59 | 269 | 1321 | 76 |
| Kao Chow Repeated Authentication-Handshake Key | 1 | 7 | 22 | 99 | 555 | 1 | 1 | 1 | 6 | 13 | 96 |
| Kao Chow Repeated Authentication-Ticket | 5 | 19 | 139 | 715 | 5382 | 5 | 15 | 57 | 243 | 1115 | 77 |
| NSL-ECB-homo | 5 | 18 | 116 | 532 | 3006 | 5 | 18 | 116 | 531 | 2896 | 3 |
| NSL-XOR-modified | 1 | 10 | 42 | 442 | 6184 | 1 | 10 | 42 | 311 | 2077 | 63 |
| Otway-Rees | 1 | 7 | 18 | 55 | 237 | 1 | 7 | 18 | 49 | 193 | 15 |
| Wide Mouthed Frog | 6 | 31 | 226 | 1492 | 11788 | 6 | 23 | 113 | 663 | 3872 | 65 |
| Woo-Lam | 1 | 8 | 24 | 115 | 936 | 1 | 8 | 24 | 90 | 331 | 57 |

Table 6: Number of new states produced in each of 1,2,3,4, and 5 backwards narrowing steps with and without the optimization of Section 7.

cutions 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 theses techniques and have shown that they preserve soundness and completeness, so that 1) any attact that is found is valid, and 2) 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, when used in combination, these techniques: (i) typically provide drastic state space reductions, removing as much as 95 percent of the states that would otherwise by generated; 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 would like to thank Antonio Gonzalez for his help in providing several protocol specifications in Maude-NPA.

**References**

[1] Amadio, R., Lopez, V., 2000. On the reachability problem in cryptographic protocols. In: Concur '00. Springer, pp. 380–394.

[2] Basin, D., Cremers, C., Meadows, C., 2012. Model checking security protocols. To appear in Handbook of Model Checking, available at `http://people.inf.ethz.ch/cremersc/publications`.

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

[4] Basin, D. A., Mödersheim, S., Viganò, L., 2003. An on-the-fly model-checker for security protocol analysis. In: Snekkenes, E., Gollmann, D. (Eds.), ESORICS. Vol. 2808 of Lecture Notes in Computer Science. Springer, pp. 253–270.

[5] Blanchet, B., Jun. 2001. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: 14th IEEE Computer Security Foundations Workshop (CSFW-14). IEEE Computer Society, Cape Breton, Nova Scotia, Canada, pp. 82–96.

[6] Chevalier, Y., Küsters, R., Rusinowitch, M., Turuani, M., 2008. Complexity results for security protocols with diffie-hellman exponentiation and commuting public key encryption. ACM Trans. Comput. Log. 9 (4).

[7] Cremers, C., 2006. Scyther - semantics and verification of security protocols. Ph.D. thesis, Eindhoven University of Technology.

[8] Cremers, C. J. F., 2008. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In: Ning, P., Syverson, P. F., Jha, S. (Eds.), ACM Conference on Computer and Communications Security. ACM, pp. 119–128.

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

[10] 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).

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] Escobar, S., Meseguer, J., Sasse, R., 2009. Variant narrowing and equational unification. Electr. Notes Theor. Comput. Sci. 238 (3), 103–119.

[18] Escobar, S., Sasse, R., Meseguer, J., 2010. Folding variant narrowing and optimal variant termination. In: Ölveczky, P. C. (Ed.), Rewriting Logic and Its Applications - 8th International Workshop, WRLA 2010, Held as a Satellite Event of ETAPS 2010, Paphos, Cyprus, March 20-21, 2010, Revised Selected Papers. Vol. 6381 of Lecture Notes in Computer Science. Springer, pp. 52–68.

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

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

[21] Huima, A., 1999. Efficient infinite-state analysis of security protocols. In: Proc. FLOC99 Workshop on Formal Methods and Security Protocols (FMSP99).

[22] 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.

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

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

[25] 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.

[26] Millen, J. K., Shmatikov, V., 2001. Constraint solving for bounded-process cryptographic protocol analysis. In: Reiter, M. K., Samarati, P. (Eds.), ACM Conference on Computer and Communications Security. ACM, pp. 166–175.

[27] Mödersheim, S., Viganò, L., Basin, D. A., 2010. Constraint differentiation: Search-space reduction for the constraint-based analysis of security protocols. Journal of Computer Security 18 (4), 575–618.

[28] Sasse, R., Escobar, S., Meadows, C., Meseguer, J., 2011. Protocol analysis modulo combination of theories: A case study in maude-npa. In: Cuéllar, J., Lopez, J., Barthe, G., Pretschner, A. (Eds.), Security and Trust Management - 6th International Workshop, STM 2010, Athens, Greece, September 23-24, 2010, Revised Selected Papers. Vol. 6710 of Lecture Notes in Computer Science. Springer, pp. 163–178.

[29] 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.

[30] TeReSe (Ed.), 2003. Term Rewriting Systems. Cambridge University Press, Cambridge.

[31] 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.

[32] Weidenbach, C., 1999. Towards an automatic analysis of security protocols in first-order logic. In: Ganzinger, H. (Ed.), CADE. Vol. 1632 of Lecture Notes in Computer Science. Springer, pp. 314–328.