The Maude-NRL Protocol Analyzer
Lecture 1: Introduction to Maude-NPA

Catherine Meadows

Naval Research Laboratory, Washington, DC 20375
catherine.meadows@nrl.navy.mil

Formal Methods for the Science of Security Summer School,
University of Illinois, July 22-26, 2013
Purpose of These Lectures

- Introduce you to a particular tool for symbolic cryptographic protocol analysis, Maude-NPA
  - Tool for automatic analysis of crypto protocols that takes into account equational theories of crypto operators
  - Based on unification and rewrite rules
- On the way, point out connections between research on the tool and open problems in crypto protocol analysis, rewriting logic, and unification
- Introduce you to two emerging related topics of research
  - Asymmetric Unification
  - Symbolic Indistinguishability
Outline

1. Approach

2. Introduction to Rewriting Logic and Unification

3. How Maude-NPA Works
   - Specifying Protocols and States in Maude-NPA
   - Backwards Narrowing and Rewrite Semantics
   - Sequential Composition in Maude-NPA
Example: Diffie-Hellman Without Authentication

1. $A \rightarrow B : g^{N_A}$
2. $B \rightarrow A : g^{N_B}$
3. $A$ and $B$ compute $g^{N_A \times N_B} = g^{N_B \times N_A}$

Well-known attack

1. $A \rightarrow I_B : g^{N_A}$
2. $I_A \rightarrow B : g^{N_I}$
3. $B \rightarrow I_A : g^{N_B}$
4. $I_B \rightarrow A : g^{N_I}$

- $A$ thinks she shares $g^{N_I \times N_A}$ with $B$, but she shares it with $I$
- $B$ thinks he shares $g^{N_I \times N_B}$ with $A$, but he shares it with $I$
- Commutative properties of $\times$ and fact that $(G^X)^Y = G^{X \times Y}$ crucial to understanding both the protocol and the attack
"Dolev-Yao" Model for Automated Cryptographic Protocol Analysis

- Start with a signature, giving a set of function symbols and variables
- For each role, give a program describing how a principal executing that role sends and receives messages
- Give a set of inference rules describing the deductions an intruder can make
  - E.g. if intruder knows $K$ and $e(K, M)$, can deduce $M$
- Assume that all messages go through intruder who can
  - Stop or redirect messages
  - Alter messages
  - Create new messages from already sent messages using inference rules
- This problem well understood since about early 2000’s
What We Know About Dolev-Yao

Important notion: the "session"
- A session is a single execution of a role in the protocol by a legitimate principal
- Legitimate execution of unauthenticated DH and attack both involved two sessions
- One for initiator and one for responder

Known results
- Secrecy undecidable in standard Dolev-Yao model (Cervesato et al., 1999)
- Secrecy NP-complete in standard DY model if number of sessions are bounded (bounded session model) (Rusinowitch and Turuani, 2001)
- Similar results for authentication: both secrecy and authentication can be expressed in terms of reachability of states
Beyond the Free Algebra

- Crypto protocol analysis with the standard free algebra model (Dolev-Yao) well understood.
- But, not adequate to deal with protocols that rely upon algebraic properties of cryptosystems
  1. Cancellation properties, encryption-decryption
  2. Abelian groups
  3. Diffie-Hellman (exponentiation, Abelian group properties)
  4. Homomorphic encryption (distributes over an operator with also has algebraic properties, e.g. Abelian group)
  5. Etc. ..,

- In many cases, a protocol uses some combination of these
Approach

State of the Art When We Started this Research (mid 2000’s)

- Free algebra model well understood
- Decidability results were beginning to appear for other theories
  - Similar to results for free algebras
- Tools were beginning to appear that handled different theories
- Next step needed
  - Approach equational theories in a systematic way
  - Support combination of theories to the greatest extent possible
Goal of Maude-NPA

Provide **tool** that

- can be used to reason about protocols with different **algebraic properties** in the **unbounded** session model
- supports **combinations** of algebraic properties to the greatest degree possible
Our approach

- Use rewriting logic as general theoretical framework
  - crypto protocols are specified using rewrite rules
  - algebraic identities as equational theories
- Use narrowing modulo equational theories as a symbolic reachability analysis method
- Combine with state reduction techniques of Maude-NPA’s ancestor, the NRL Protocol Analyzer (grammars, optimizations, etc.)
- Implement in Maude programming environment
  - Rewriting logic gives us theoretical framework and understanding
  - Maude implementation gives us tool support
Maude-NPA

- A tool to **find** or **prove the absence** of attacks using **backwards search**
- Analyzes **infinite state systems**
  - Active intruder
  - **No abstraction or approximation** of nonces
  - Unbounded number of sessions
- **Intruder** and **honest** protocol transitions represented using strand space model.
- So far supports a number of equational theories: cancellation (e.g. encryption-decryption), AC, exclusive-or, Diffie-Hellman, bounded associativity, homomorphic encryption over a free theory, various combinations, working on including more
Outline

1. Approach

2. Introduction to Rewriting Logic and Unification

3. How Maude-NPA Works
   - Specifying Protocols and States in Maude-NPA
   - Backwards Narrowing and Rewrite Semantics
   - Sequential Composition in Maude-NPA
A Little Background on Unification

- Given a signature $\Sigma$ and an equational theory $E$, and two terms $s$ and $t$ built from $\Sigma$:
- A unifier of $s =_E t$ is a substitution $\sigma$ to the variables in $s$ and $t$ s.t. $\sigma s$ can be transformed into $\sigma t$ by applying equations from $E$ to $\sigma s$ and its subterms.
- Example: $\Sigma = \{d/2, e/2, m/0, k/0\}$, $E = \{d(K, e(K, X)) = X\}$. The substitution $\sigma = \{Z \mapsto e(T, Y)\}$ is a unifier of $d(T, Z)$ and $Y$.
- The set of most general unifiers of $s =_E t$ is the set $\Gamma$ s.t. any unifier $\sigma$ is of the form $\rho \tau$ for some $\rho$, and some $\tau$ in $\Gamma$.
- Example: $\{Z \mapsto e(T, Y), Y \mapsto d(T, Z)\}$ mgu’s of $d(T, Z)$ and $Y$.
- Depending on the theory, can have:
  - at most one mgu (empty theory)
  - a finite number (AC)
  - an infinite number (associativity)
- There are unification problems that are undecidable.
A rewrite theory $\mathcal{R}$ is a triple $\mathcal{R} = (\Sigma, R, E)$, with:

- $\Sigma$ a signature
- $(\Sigma, R)$ a set of rewrite rules of the form $t \rightarrow s$
  e.g. $rcv\ e(K_A, N_A; X) \rightarrow snd\ e(K_B, X)$
- $(\Sigma, E)$ a set of equations of the form $t = s$
  e.g. $d(K, e(K, Y)) = Y$

Intuitively, $\mathcal{R}$ specifies a concurrent system, whose states are elements of the initial algebra $T_{\Sigma/E}$ specified by $(\Sigma, E)$, and whose concurrent transitions are specified by the rules $R$. Narrowing gives us the rules for executing transitions concurrently.
Narrowing and Backwards Narrowing

Narrowing: $t \rightsquigarrow_{\sigma,R,E} s$ if there is
- a non-variable position $p \in Pos(t)$;
- a rule $l \rightarrow r \in R$;
- a unifier $\sigma$ (modulo $E$) of $t|p =_E ?l$ such that $s = \sigma(t[r]_p)$.

Example:
- $R = \{ \text{rcv } X \rightarrow \text{snd } d(k, X) \}$, $E = \{ d(K, e(K, Y)) = Y \}$
- $\text{rcv } e(k, t) \rightsquigarrow_{\{X/e(k, t)\},R,E} \text{snd } d(k, e(k, t)) =_E t$

Backwards Narrowing: narrowing with rewrite rules reversed
A Warning About Narrowing

- Full narrowing (narrowing in every possible non-variable location) is often inefficient and even nonterminating.
- We need to construct our rewrite systems so that efficient narrowing strategies can be chosen.
- Maude-NPA has led to some major advances in this area.
Narrowing can be used as a general deductive procedure for solving reachability problems of the form

\[(\exists \vec{x}) \ t_1(\vec{x}) \rightarrow^* t_1'(\vec{x}) \land \ldots \land t_n(\vec{x}) \rightarrow^* t_n'(\vec{x})\]

in a given rewrite theory, where the terms \(t_i\) and \(t_i'\) denote sets of states.

- The terms \(t_i\) and \(t_i'\) denote sets of states.
- For what substitutions \(\sigma\) are \(t_i(\sigma \vec{x})\) reachable from \(t_i'(\sigma \vec{x})\)
- No finiteness assumptions about the state space.
- Maude-NPA rewrite system supports topmost narrowing for state reachability analysis
  - Rewrite rules apply to topmost position only, so narrowing steps only need to be applied to entire state
  - Topmost narrowing complete
$E$-Unification

- In order to apply narrowing to search, need an $E$ unification algorithm
- Two approaches:
  - 1. Built-in unification algorithms for each theory
  - 2. Hybrid approach with $E = R \cup \Delta$
- Hybrid Approach
  - $\Delta$ has built-in unification algorithm
  - $R$ confluent, terminating, and coherent rules modulo $\Delta$
    - Confluent: Always reach same normal form modulo $B$, no matter in which order you apply rewrite rules
    - Terminating: Sequence of rewrite rules is finite
    - Coherent: Technical condition on equations needed to make narrowing on representatives of $\Delta$-equivalence classes complete
    - Non-coherent equational theories can often be made coherent by adding extra (redundant) equations
- Lets us use $R, \Delta$ narrowing as a general method for $E$-unification
How $R, \Delta$ Narrowing Works for Unification

1. Start with a problem $s = ?t$ and $U = \emptyset$.
2. Find a set of $\Delta$-mgu’s of $s = ?t$ add these to $U$.
3. For each non-variable position $p$ in $s = ?t$ and each rewrite rule $r \rightarrow \ell \in R$, find a set of $\Delta$-mgu’s of $s = ?t|_p$ and $r$
4. For each unifier $\sigma$ of $s = ?t|_p$ and $r$ found in Step 3, create the problem $\sigma(s = ?t|_p)[\ell]$
5. Solve $\sigma(s = ?t|_p)[\ell]$ using Steps 1 through 4, for each unifier $\tau$ found, add $\tau \sigma$ to $U$. 
Example

- $\Sigma = \{e/2, d/3\}$, $R = \{d(K, e(K, X)) \rightarrow X\}$, $\Delta = \emptyset$
- $d(W, Z) = ? Y$
- First solution $Y / d(W, Z)$
- Second Solution: $d(W, Z)$ unifies with $d(K, e(K, X))$ via $\sigma = W / e(K, X)$
- New problem is $X = ? Y$, solution is $\tau = X / Y$, given unifier $\tau \sigma = W / e(K, Y)$
- Only variable positions left, so we are done with two unifiers
- We were lucky: narrowing often doesn’t terminate!
- Will later discuss theories for which narrowing can terminate and strategies for achieving termination and soundness
Outline

1. Approach

2. Introduction to Rewriting Logic and Unification

3. How Maude-NPA Works
   - Specifying Protocols and States in Maude-NPA
   - Backwards Narrowing and Rewrite Semantics
   - Sequential Composition in Maude-NPA
Uses Strand Space Notation

- Strand spaces: popular model introduced by Thayer, Herzog, and Guttman
- Each local execution, or session of an honest principal represented by sequence of positive and negative terms called a strand.
  - Terms made up of variables and function symbols
  - Negative term stand for received message, positive terms stand for sent messages
  - Example:
    
    \[
    + (\text{pke}(B, N_A; A)), \quad -(\text{pke}(A, N_A; N_B)), \quad + (\text{pke}(B, N_B))
    \]
- Each intruder computation also represented by strand
  - Example: \[-(X), + (\text{pke}(A, X))\]
Basic Structure of Maude-NPA

- Uses modified strand space model
- Each local execution and each intruder action represented by a strand, plus a marker denoting the current state
  - Searches backwards through strands from final state
  - Set of rewrite rules governs how search is conducted
  - Sensitive to past and future
- Grammars used to prevent infinite loops
- Learn-only-once rule says intruder can learn term only once
- When an intruder learns term in a backwards search, tool keeps track of this and doesn’t allow intruder to learn it again
- Other optimization techniques used to reduce other infinite behavior and to cut down size of search space
Maude-NPA’s use of backwards search means we have incomplete picture of what intruder learned in past. But we need the concrete moment when the intruder learns something:

- **Notion of the present**
  - What the intruder knows in the present (i.e., $t \in I$)
  - Where the honest principals are in the present (strands)

- **Notion of the future**
  - What terms the intruder will learn in the future (i.e., $t \notin I$)
How Protocols Are Specified in Maude-NPA

- Represent protocols and intruder actions using strands
- Terms in strands obey an equational theory specified by the user
- Terms in strands of different sorts, mostly defined by user
- Special sort Fresh
  - Terms of sort *Fresh* are not unifiable with each other (used by *nonces*)
  - Strand annotated with fresh terms generated by the strand

\[
:: r :: [+(pke(B, n(A, r); A)), -(pke(A, n(A, r); NB)), +(pke(B, NB))]\]
A state is a set of strands plus the intruder knowledge (i.e., a set of terms)

1. Each strand is divided into past and future
   \[ [m_1^\pm, \ldots, m_i^\pm \mid m_{i+1}^\pm, \ldots, m_k^\pm] \]

2. Initial strand \([\text{nil} \mid m_1^\pm, \ldots, m_k^\pm]\), final strand
   \([m_1^\pm, \ldots, m_k^\pm \mid \text{nil}]\)

3. The intruder knowledge contains terms \(m \notin \mathcal{I}\) and \(m \in \mathcal{I}\)
   \[ \{t_1 \notin \mathcal{I}, \ldots, t_n \notin \mathcal{I}, s_1 \in \mathcal{I}, \ldots, s_m \in \mathcal{I}\} \]

4. Initial intruder knowledge \(\{t_1 \notin \mathcal{I}, \ldots, t_n \notin \mathcal{I}\}\),
   final intruder knowledge \(\{s_1 \in \mathcal{I}, \ldots, s_m \in \mathcal{I}\}\)
Outline

1. Approach

2. Introduction to Rewriting Logic and Unification

3. How Maude-NPA Works
   - Specifying Protocols and States in Maude-NPA
   - Backwards Narrowing and Rewrite Semantics
   - Sequential Composition in Maude-NPA
Protocol Rules and Their Execution With Strands
Already in State

To execute a protocol $\mathcal{P}$ associate to it a rewrite theory on sets of strands as follows. Let $\mathcal{I}$ informally denote the set of terms known by the intruder, and $K$ the facts known or unknown by the intruder.

1. $\left[ L \mid M^-, L' \right] \& S \& \{ M \in \mathcal{I}, K \} \rightarrow \left[ L, M^- \mid L' \right] \& S \& \{ M \in \mathcal{I}, K \}$
   Moves input messages into the past

2. $\left[ L \mid M^+, L' \right] \& S \& \{ K \} \rightarrow \left[ L, M^+ \mid L' \right] \& S \& \{ K \}$
   Moves output message that are not read into the past

3. $\left[ L \mid M^+, L' \right] \& S \& \{ M \notin \mathcal{I}, K \} \rightarrow \left[ L, M^+ \mid L' \right] \& S \& \{ M \notin \mathcal{I}, K \}$
   Joins output message with term in intruder knowledge.

For backwards execution, just reverse.
If we want an unbounded number of strands, need some way of introducing new strands in the backwards search.

Specialize rule r3 using each strand \([ l_1, u^+, l_2 ]\) of the protocol \(\mathcal{P}\):

\[
[ l_1 | u^+] \& S \& \{ u \notin I, K \} \rightarrow \{ u \in I, K \} \& S
\]

Gives us a natural way of switching between bounded and unbounded sessions.

- Put a bound on the number of times specialized rule r3 could be invoked with non-intruder strands.
Reachability Analysis

- Backwards narrowing protocol execution defines a backwards reachability relation $St \rightsquigarrow_p^* St'$
- In initial step, prove lemmas that identify certain states unreachable
- Specify a state describing the attack state, including a set of final strands plus terms $m \notin I$ and $m \in I$
- Execute the protocol backwards to an initial state, if possible
- For each intermediate state found, check if it has been proved unreachable and discard if it is
Outline

1. Approach

2. Introduction to Rewriting Logic and Unification

3. How Maude-NPA Works
   - Specifying Protocols and States in Maude-NPA
   - Backwards Narrowing and Rewrite Semantics
   - Sequential Composition in Maude-NPA
Introduction

- Crypto protocols don’t exist in isolation, but often rely upon one another

- Protocols that work correctly in one environment may fail when they are composed with new protocols in new environments
  - The properties they guarantee are not quite appropriate for the new environment
  - The composition itself is mishandled

- Research has concentrated on parallel composition, but sequential composition is where most of the problems lie

- The problem is in providing a specification and verification environment that supports sequential composition
Motivating examples

One-parent, one-child protocol composition

- The parent protocol can have only one child instance
- Example: NSL with Distance Bounding (DB)
  - NSL is used to agree on $N_A$
  - DB reveals $N_A$, so it cannot be used with the same $N_A$ more than once
Motivating examples

One-parent, one-child protocol composition
- The parent protocol can have only one child instance
- Example: NSL with Distance Bounding (DB)
  - NSL is used to agree on $N_A$
  - DB reveals $N_A$, so it cannot be used with the same $N_A$ more than once

One-parent, many-children protocol composition
- The parent protocol has an arbitrary number of child instances
- Example: NSL with Key Distribution
  - The parent protocol generates a master key
  - The child protocol uses the master key and generates a session key
Motivating examples: NSL-DB

One-parent, one-child: NSL with Distance Bounding (DB)(*)

Alice claims that she is a certain distance $\delta_{AB}$ from Bob, and Bob wants to check this

- Needham-Schroeder-Lowe Public Key Protocol (NSL)
  1. $A \rightarrow B : \text{pke}(B, N_A; A)$
  2. $B \rightarrow A : \text{pke}(A, N_A; N_B; B)$
  3. $A \rightarrow B : \text{pke}(B, N_B)$

- At the end, $A$ and $B$ know that they share two secrets, $N_A$ and $N_B$. They will use $N_A$ for distance bounding (DB)
  4. $B \rightarrow A : N'_B$
  5. $A \rightarrow B : N_A \oplus N'_B$

- Bob checks time it takes for round trip, and uses it to put upper bound on distance $\delta_{AB}$ of Alice

The Maude-NRL Protocol Analyzer Lecture 1: Introduction to Maude-NPA

How Maude-NPA Works

Sequential Composition in Maude-NPA

Attack on NSL-DB

Bob concludes: $N_A, N_B$ shared with $I$, and $I$ is distance $\delta_{AB}$ from him.
What happened?

- NSL guarantees origin of responder nonce only when responder is honest.
- If responder dishonest, Bob could have got the nonce from someone else.
- What a distance bounding protocol needs is the following:
  - If sender of authenticated response is honest, then sender of rapid response is the same individual.
  - If sender of rapid response is honest, then sender of authenticated response is the same individual.
- One solution: alter rapid response so that composition works.
Fixing the NSL-DB protocol

1. Needham-Schroeder-Lowe Public Key Protocol (NSL)
   1. $A \rightarrow B : pke(B, N_A; A)$
   2. $B \rightarrow A : pke(A, N_A; N_B; B)$
   3. $A \rightarrow B : pke(B, N_B)$

2. Distance bounding using $N_A$
   4. $B \rightarrow A : N'_B$
   5. $A \rightarrow B : h(A, N_A) \oplus N'_B$

- Alice hashes her nonce with her identity before responding
- If the sender of the rapid response is honest, he will hash with his own identity.
Motivating examples: NSL-KD

One-parent, many-children: NSL with Key Distribution (KD)

- Needham-Schroeder-Lowe Public Key Protocol (NSL)
  1. $A \rightarrow B : \text{pke}(B, N_A; A)$
  2. $B \rightarrow A : \text{pke}(A, N_A; N_B; B)$
  3. $A \rightarrow B : \text{pke}(B, N_B)$

- $N_A$ and $N_B$ will be used for key distribution

- The initiator of the session key protocol can be the child of either the initiator or responder of the NSL protocol
  4. $A \rightarrow B : \left\{ \text{Sk}_A \right\} h(N_A, N_B)$
  5. $B \rightarrow A : \left\{ \text{Sk}_B, N'_B \right\} h(N_A, N_B)$
  6. $A \rightarrow B : \left\{ N'_B \right\} h(N_A, N_B)$
  4. $B \rightarrow A : \left\{ \text{Sk}_B \right\} h(N_A, N_B)$
  5. $A \rightarrow B : \left\{ \text{Sk}_B, N'_A \right\} h(N_A, N_B)$
  6. $B \rightarrow A : \left\{ N'_A \right\} h(N_A, N_B)$
Strand Annotations

1. Separate strands for parent and child
2. Annotate strands with input and output parameters
3. Parameters tell you
   - Roles of parent and child
   - Whether composition is 1-1 or 1-many
   - Relevant input and output parameters
4. Composition is performed via unification of output parameters of parent strand with input parameters of child strand
Example: NSL-Distance Bounding Protocol

- **NSL Initiator**
  \[ r :: \{ \text{init} \rightarrow \text{dbresp} ;; 1-1 ;; A,B,n(A,r) \} . \]

- **NSL Responder**
  \[ r :: \{ \text{resp} \rightarrow \text{dbinit} ;; 1-1 ;; A,B,NA \} . \]

- **DB Initiator**
  \[ r :: \{ \text{resp} \rightarrow \text{dbinit} ;; 1-1 ;; A,B,NA \} ,
  +n(B,r), -(NA \ast n(B,r)), \text{nil} \].

- **DB Initiator**
  \[ \text{nil} :: \{ \text{init} \rightarrow \text{dbresp} ;; 1-1 ;; A,B,NA \} ,
  -(NB'), +\text{(NB'} \ast NA\text{)}, \text{nil} \].
Use of Null Strands

- Null strands used in case a parent can have more than one type of child, or vice versa.
- Example: Needham-Schroeder-Lowe with Key Distribution.
- Parent strand can have a child strand that is an initiator or a responder of the child protocol.
- Give the initiator parent output parameters of the form \{ \text{init} \rightarrow \text{initnull} ;; \text{1-many} ;; \text{A,B,n(A,r),NB} \}
- Have two initnull strands:
  - :: :: [\text{nil}, \{\text{init} \rightarrow \text{initnull} :: \text{1-many} :: \text{A,B,n(A,r),NB} \}, \{\text{initnull} \rightarrow \text{initchild} :: \text{1-1} :: \text{A,B,n(A,r),NB} \}, \text{nil} ]
  - :: :: [\text{nil}, \{\text{init} \rightarrow \text{initnull} :: \text{1-many} :: \text{A,B,n(A,r),NB} \}, \{\text{initnull} \rightarrow \text{respchild} :: \text{1-1} :: \text{A,B,n(A,r),NB} \}, \text{nil} ]
- Construct similar null strands for responder parent.
for each one-to-one composition with strand definitions \([\{\overrightarrow{I_a}\}, \overrightarrow{a}, \{\overrightarrow{O_a}\}]\) and \([\{\overrightarrow{I_b}\}, \overrightarrow{b}, \{\overrightarrow{O_b}\}\]
and unifiers \(\sigma_a, \sigma_{ab}\) s.t. \(\overrightarrow{O_a} = \overrightarrow{E_P} \sigma_a(\overrightarrow{O})\) and \(\sigma_a(\overrightarrow{l}) = \overrightarrow{E_P} \sigma_{ab}(\overrightarrow{l_b})\), add:

\[
\begin{align*}
SS & \& [\overrightarrow{a} \mid \{\overrightarrow{O_a}\}] \& [nil \mid \{\sigma_{ab}(\overrightarrow{l_b})\}, \sigma_{ab}(\overrightarrow{b})]\& IK \\
\rightarrow SS & \& [\overrightarrow{a}, \{\overrightarrow{O_a}\} \mid nil] \& [\{\sigma_{ab}(\overrightarrow{l_b})\} \mid \sigma_{ab}(\overrightarrow{b})]\& IK
\end{align*}
\]

Case in which parent already present in right-hand state

\[
\begin{align*}
SS & \& [\overrightarrow{a} \mid \{\overrightarrow{O_a}\}] \& [nil \mid \{\sigma_{ab}(\overrightarrow{l_b})\}, \sigma_{ab}(\overrightarrow{b})]\& IK \\
\rightarrow SS & \& [\{\sigma_{ab}(\overrightarrow{l_b})\} \mid \sigma_{ab}(\overrightarrow{b})]\& IK
\end{align*}
\]
Case in which parent not already present in right-hand state
Model for One-to-Many Composition

For each one-to-many composition with strand definitions \([\{\vec{I}_a\}, \vec{a}, \{\vec{O}_a\}]\) and \([\{\vec{I}_b\}, \vec{b}, \{\vec{O}_b\}]\) and unifiers \(\sigma_a, \sigma_{ab}\) s.t. \(\vec{O}_a =_{\mathbb{E}_P} \sigma_a(\vec{O})\) and \(\sigma_a(\vec{I}) =_{\mathbb{E}_P} \sigma_{ab}(\vec{I}_b)\), add to the previous rules:

\[
SS \& [\vec{a} \mid \{\vec{O}_a\}] \& [\text{nil} \mid \{\sigma_{ab}(\vec{I}_b)\}, \sigma_{ab}(\vec{b})] \& IK \rightarrow SS \& [\vec{a} \mid \{\vec{O}_a\}] \& [\{\sigma_{ab}(\vec{I}_b)\} \mid \sigma_{ab}(\vec{b})] \& IK \quad (3)
\]

Composition leaving parent available to compose with more children

- Rule 3 describes the interim transitions of one-to-many composition
- Rules 1 and 2 describe the final transition
What We Have

- Sequential composition of protocols supported in Maude-NPA
- Syntax and operational semantics extends in a natural way
- Have applied Maude-NPA to protocols described in this lecture (using a slightly different syntax)
- Have also applied it to group protocols, showing how it can be used to specify group protocols for arbitrarily large groups
Dolev-Yao References


Maude-NPA References

- Maude-NPA 2.0 and relevant papers available at
  http://maude.cs.uiuc.edu/tools/Maude-NPA/


- S. Escobar, C. Meadows, J. Meseguer, S. Santiago. Sequential Protocol Composition in Maude-NPA. In Proc. of European Symposium on Research in Computer Security (ESORICS 2010), LNCS 6345, pages 303-318. 2010. Technical report DSIC-II/06/10, Departamento de Sistemas Informaticos y Computacion, Universidad Politcnica de Valencia, 2010. [Note: this describes an earlier implementation of composition than is presented in this talk. The semantics is still the same, however.]