Maude-NPA: Status and Demonstration

Catherine Meadows, Naval Research Laboratory (USA)
José Meseguer, Joe Hendrix,University of Illinois at Urbana-Champaign (USA)
Santiago Escobar, Universidad Politécnica de Valencia (Spain)

PROTOCOL EXCHANGE, OCTOBER 26, 2007

Goal

- Crypto protocol analysis with the standard free algebra model (Dolev-Yao) well understood.
- Extend standard free algebra model of crypto protocol analysis to deal with algebraic properties
  1. Encryption-decryption,
  2. Diffie Hellman,
  3. Exclusive-or, etc.
- Provide tool that can be used to reason about protocols with these algebraic properties in the unbounded session model

Our approach

- Use rewriting logic as general theoretical framework
  - crypto protocols are specified as rewrite rules
  - algebraic identities as equational properties
- Use narrowing modulo equational theories as a symbolic reachability analysis method
- Combine with state reduction techniques of NPA (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.
- Different algebraic theories included
- Uses induction techniques defined in terms of formal languages to cut down search space
- Uses optimization techniques to improve performance: only input messages, partial order, information from strand space model,lazy intruder, etc.

Our Plans

(1) Formalizing techniques in rewriting logic (DONE)

- Initial version of Maude-NPA tool:
  - Grammar Generation
  - Backwards Narrowing Reachability Analysis
  - Soundness and completeness theorems
- Paper published in TCS special issue on ARSPA

(2) Include state reduction techniques present in NPA and new (CURRENT)

- Optimizations for state reduction.

(3) Extend model to different types of equational theories (CURRENT)

- Bounded Associativity (SecReT 2006)
- Diffie-Hellman Exponentiation (SecReT 2007)
- AC version of Maude-NPA implemented via CiME tool and ported to Maude with AC Unification (DONE)
- Extend grammars and state reduction techniques to equational theories (CURRENT)

Progress Since Last PX Meeting

① Unification Issues

- Ported Maude NPA to Steven Eker's Maude with AC unification
- Addressing nontermination of narrowing modulo AC with finite variant property

② Language Issues

- New specification and query language that human beings can use
- Approaches to Automatic Generation of Seedterms

③ Search Termination Issues

- Folding
- Grammars

④ Demos

- NSPK with "Never found any"
- Diffie-Hellman

A Little Background on Unification

- Given a signature Σ and an equational theory E, and two terms s and t built from Σ:
- A unifier of s and t is a substitution σ to the variables in s and t such that σs can be transformed into σt by applying equations from E to s and t and their subterms
- Example: $\Sigma = \{d/2, e/2, m/0, k/0\}, E = \{d(K, e(K, X) = X\}$. The substitution $\sigma = \{X/e(K, Y)\}$ is a unifier of $d(K, X) and Y$.
- The set of most general unifiers of s and t is the set Γ such that any unifier σ is of the form $\rho\tau$ for some ρ, and some τ in Γ.
- Example, $\{X/e(K, Y), Y/d(K, X)\}$ is the set of mgu's of $e(K, X)$ and Y.
- Given the theory, can have:
  - at most one mgu (empty theory)
  - a finite number (AC)
  - an infinite number (associativity)
- Problem in general undecidable, so different algorithms devised for different theories

Narrowing

Let σ be a substitution, R a set of rewrite rules and E and equational thoery
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 σ (modulo E) such that $\sigma(t|_p) =_E \sigma(l)$, and $s = \sigma(t[r]_p)$.

Example:

- $R = \{\ X \rightarrow d(k, X)\ \}$
- $E = \{\ d(K, e(K, Y)) = Y\ \}$
- $e(k,\ t) \quad \rightsquigarrow_{\emptyset,R,E} \quad d(k, e(k, t)) =_E t$

E-Unification and Narrowing

- Maude-NPA based on unification modulo equational theory defining the behavior of different operations used
- Two possible approaches:
  1. Built-in unification algorithms for each theory and combination of theories.
  2. Hybrid approach with Δ and B
     - B is built-in unification algorithm
     - Δ confluent and terminating rules modulo B
       - Confluent: Always reach same normal form, no matter in which order you apply rewrite rules
       - Terminating: Sequence of rewrite rules is finite
     - Implement unification via narrowing with Δ modulo B.
     - More readily extensible to different theories.
- Our Approach
  - Let B be the empty theory or AC
  - Old and new approaches
    - Old: Unification modulo B performed via calls to CiME unification tool
    - New: Unification module B provided by Maude
  - In both cases, narrowing with Δ performed at Maude meta-level

Order-sorted Unification

- In 2007, order-sorted unification and AC unification implemented in Maude by Steven Eker
- MaudeNPA ported to new AC version of Maude this summer
- Running time of Maude with CIME AC vs. AC Maude is approximately 1.3 on our benchmarks
  - External calls to CIME no longer necessary
  - Conversion from unsorted to sorted unification no longer necessary
  - True even for non-AC theories because because state is a set of strands, set obeys ACU
- We expect the ratio to go up as the number of AC operations increases.

When Narrowing Doesn't Terminate: Finite Variant Property

- Problem: Even typed narrowing mod AC is nonterminating for several theories of interest
  - Example: exclusive-or
- Investigating the finite variant property of Comon and Delaune
- Finite variant property: have an algorithm that computes, from any term t, a finite set of subsitutions $\sigma_1, \cdots, \sigma_n$ such that

$$\{t\sigma \downarrow \mathcal{R} \mid \sigma \in \Sigma\} = \bigcup_{i=1}^{n} \{t\sigma_i\theta \mid \theta \in \Sigma\}$$

- Finite variant property means that can compute a bound on the number of narrowing steps necessary to get a complete solution
- Even better, don't even need to narrow, just compute all finite variants, try to unification modulo AC using each one
  - Already have something like this for parts of MaudeNPA that require term to be in reduced form
- Allows us to maintain hybrid approach while expanding scope of theory
- Have begun experimental implementation, already have implementation of ACU

Progress Since Last PX Meeting

① Unification Issues

- Ported Maude NPA to Steven Eker's Maude with AC unification
- Addressing nontermination of narrowing modulo AC with finite variant property

② Language Issues

- New specification and query language that human beings can use
- Approaches to Automatic Generation of Seedterms

③ Search Termination Issues

- Folding
- Grammars

④ Demos

- NSPK with "Never found any"
- Diffie-Hellman

Parts of a MaudeNPA Specification

- Information About Message Components
- Strand Specifications
- Attack States
- Seed Terms

Sorts

```
-------------------------------------------------
--- We modify only the relevant MAUDE-NPA modules
-------------------------------------------------

fmod PROTOCOL-EXAMPLE-SYMBOLS is
  --- Importing sorts Msg, Fresh, Public, and GhostData
  protecting DEFINITION-PROTOCOL-RULES .

  ---------------------------------------------------------
  --- Overwrite this module with the syntax of your protocol
  --- Notes:
  --- * Sort Msg and Fresh are special and imported
  --- * Every sort must be a subsort of Msg
  --- * No sort can be a supersort of Msg
  ---------------------------------------------------------
--- Sort Information
  sorts Name Nonce Key Enc .
  subsort Name Nonce Enc Key < Msg .
  subsort Name < Key .
  subsort Name < Public .
```

Operations

```
  --- Encoding operators for public/private encryption
  op pk : Key Msg -> Enc [frozen] .
  op sk : Key Msg -> Enc [frozen] .

  --- Nonce operator
  op n : Name Fresh -> Nonce [frozen] .

  --- Principals
  op a : -> Name . --- Alice
  op b : -> Name . --- Bob
  op i : -> Name . --- Intruder

 --- ConcatenationBa operator
  op _;_ : Msg  Msg  -> Msg [gather (e E) frozen] .

endfm
```

Algebraic Theory

```
fmod PROTOCOL-EXAMPLE-ALGEBRAIC is
  protecting PROTOCOL-EXAMPLE-SYMBOLS .


  var Z : Msg .
  var Ke : Key .

  *** Encryption/Decryption Cancellation
  eq pk(Ke,sk(Ke,Z)) = Z [nonexec] .
  eq sk(Ke,pk(Ke,Z)) = Z [nonexec] .

endfm
```

Intruder Strands

```
fmod USER-INPUT is
  protecting PROTOCOL-EXAMPLE-SYMBOLS .
  protecting DEFINITION-PROTOCOL-RULES .
  protecting DEFINITION-CONSTRAINTS-INPUT .

var Ke : Key .
  vars X Y Z : Msg .
  vars r r' : Fresh .
  vars A B : Name .
  vars N N1 N2 : Nonce .

  eq STRANDS-DOLEVYAO
   = :: nil :: [ nil | -(X), -(Y), +(X ; Y), nil ] &
     :: nil :: [ nil | -(X ; Y), +(X), nil ] &
     :: nil :: [ nil | -(X ; Y), +(Y), nil ] &
     :: nil :: [ nil | -(X), +(sk(i,X)), nil ] &
     :: nil :: [ nil | -(X), +(pk(Ke,X)), nil ] &
     :: nil :: [ nil | +(A), nil ]
  [nonexec] .
```

Protocol Strands

```
eq STRANDS-PROTOCOL
 = :: r ::
   [ nil | +(pk(B,A ; n(A,r))), -(pk(A,n(A,r) ; N)), +(pk(B, N)), nil ] &
   :: r ::
   [ nil | -(pk(B,A ; N)), +(pk(A, N ; n(B,r))), -(pk(B,n(B,r))), nil ]
[nonexec] .
```

Attack States

```
eq ATTACK-STATE(2)
= :: r ::
  [ nil, -(pk(b,a ; N)), +(pk(a, N ; n(b,r))), -(pk(b,n(b,r))) | nil ]
  || n(b,r) inI, empty
  || nil
  || nil
  butNeverFoundAny
  :: r' ::  [nil | +(pk(b,a ; N)), -(pk(a, N ; n(b,r))), +(pk(b,n(b,r))), nil ]
          & :: nil ::  [nil | -(N1 ; N2), +(pk(B, N1 ; N2)), nil] & S:StrandSet
  || K:IntruderKnowledge
  || M:SMsgList
  || G:GhostList
[nonexec] .
```

- butNeverFoundAny can be used to define insecure state and to rule out actions leading to infinite paths

Seed Terms for Grammars

```
eq USER-GRAMMARS
   = (grl Y notInI => (X ; Y) inL . ! S1)
     |
     (grl X notInI => (X ; Y) inL . ! S1)
     |
     (grl Z notInI => pk(Ke,Z) inL . ! S1)
     |
     (grl Z notInI => sk(Ke,Z) inL . ! S1)
  [nonexec] .
```

Quick and Dirty Suggestions for Automating Seed Term Generation

- Give $X \in \mathcal{I}$ as a goal to MaudeNPA, where X is a variable
- Search back one step to find all terms T intruder must know to find X
- For each such T containing X as a proper subterm, generate a seedterm

```
grl X notInI => T inL . ! S1
```

Progress Since Last PX Meeting

① Unification Issues

- Ported Maude NPA to Steven Eker's Maude with AC unification
- Addressing nontermination of narrowing modulo AC with finite variant property

② Language Issues

- New specification and query language that human beings can use
- Approaches to Automatic Generation of Seedterms

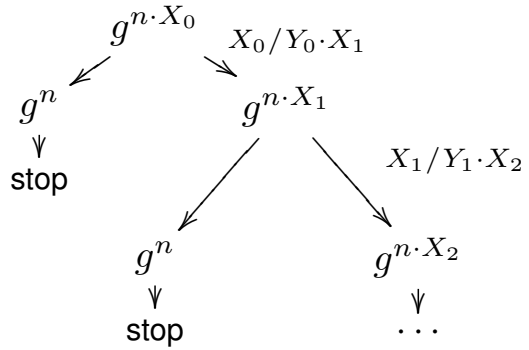
③ Search Termination Issues

- Folding
- Grammars

④ Demos

- NSPK with "Never found any"
- Diffie-Hellman

Infinite Behavior Not Captured by Grammars: 1



- Different from rewrite-rule based grammar behavior, because infinite behavior results from substitution
- Root term grows larger instead of leaf terms

Infinite Behavior Not Captured by Grammars: 2

- Consider the following protocol
  1. $A \hookrightarrow B : mess(A, r)$.
  2. $B \hookrightarrow A : mess(B, r')$.
- If we ask how intruder can find $mess(A_0, r_0)$ we run into the following:

$$\cdots, -mess(A_2, r_2), \ +mess(A_1, r_1), \quad -mess(A_1, r_1), \quad +(mess(A_0, r_0), \quad -mess(A_0, r_0)$$

- Possible to find much more complex examples
- Only a problem if have unbounded number of sessions

Investigating Folding As a Solution

- While searching, collapse any state encountered into previously seen state if some criterion satisfied
  - Examples: $=_E$, $\approx_E$, $\preceq_E$
- Basic idea is that reachability of old state implies reachability of new state, so no need for new state
- Original NPA did something very similar to this (although not as general), so we think this is likely to be successful
- Only drawback: Maude NPA will have to remember old states
- One possibility, only check k states back for some k
- Reference: Santiago Escobar, Jos Meseguer Symbolic Model Checking of Infinite-State Systems Using Narrowing In proceedings of 18th International Conference on Rewriting Techniques and Applications (RTA 2007), LNCS volume 4533, pages 153-168, 2007.

Grammar Termination

- Need to understand how grammars contribute to termination
- Question investigating now: to what degree can grammars contribute to termination in bounded session case
- It all depends on the exceptions:
  - `grl #0:Msg notInI, (#0:Msg notLeq pk( #2:Key, #3:Msg)) => pk( #7:Key, #0:Msg) inL .` leads to nontermination
  - `grl #0:Msg notInI, (#0:Msg notLeq #2:Name ; n(#2:Name, #3:Fresh) ) => pk( #7:Key, #0:Msg) inL` leads to termination
  - `grl #0:Msg notInI, (#0:Msg notLeq #2:Key, #3:Msg) => pk( #7:Key, #0:Msg) inL .` `grl #0:Msg notInI (#0:Msg notLeq n(#2:Name, #3:Fresh)) => #7:Msg, #0:Msg inL .` leads to termination
- Have defined the notion of a exception rank of a grammar, that can be used to compute the maximum size of a backwards search using intruder stands only
  - In above two cases, exception ranks were infinite, one, and two, respectively
- Exception rank gives a bound on the number of consecutive intruder strands that can be applied in a backwards search, also helps us to optimize search

Questions

- How do we guarantee finite exception rank?
- How do we generalize exception rank for AC grammars?

Progress Since Last PX Meeting

① Unification Issues

- Ported Maude NPA to Steven Eker's Maude with AC unification
- Addressing nontermination of narrowing modulo AC with finite variant property

② Language Issues

- New specification and query language that human beings can use
- Approaches to Automatic Generation of Seedterms

③ Search Termination Issues

- Folding
- Grammars

④ Demos

- NSPK with "Never found any"
- Diffie-Hellman