The Maude-NRL Protocol Analyzer Lecture 2: Controlling the Search Space and Asymmetric Unification

Catherine Meadows

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

Fifth Summer School on Formal Techniques, Menlo College, Atherton, CA, May 17-22, 2015

Outline

Outline

1. Controlling the Search Space
   - Learn-Only-Once and Grammars
   - Other Ways of Reducing the Search Space
2. Asymmetric Unification
   - Background and Motivation
   - A New Unification Paradigm: Asymmetric Unification

How Maude-NPA Controls the Search Space

- Left to itself, Maude-NPA will search forever
- Must use techniques for ruling out redundant or provably unreachable states to obtain finite search space
- We have developed a number of different techniques for doing this:
  - Intruder learns only once
  - Grammars
  - Subsumption
  - Super-Lazy Intruder
- We will cover these in this lecture

Important Assumptions

- Equational theory is of the form $E = R \uplus \Delta$
- R is a set of rewrite rules and Δ is regular
- In any states produced by Maude-NPA $t{\in}\mathcal{I}$, $t{\notin}\mathcal{I}$, and negative terms are R-irreducible
- Furthermore, no substitutions produced by further search will make these terms reducible
    - Reason: many of the checks made by Maude-NPA for state space reduction rely on the presence of particular sub terms
    - Allowing these sub terms to vanish because of rewrite rules or further substitutions will invalidate the checks
    - We will show how these assumptions are guaranteed in the lecture on the asymmetric unification

Outline

Two basic restrictions of the search space

Powerful tools:

1. Learn-only-once: any terms the intruder will learn in the future can't already be known
2. Grammars describing unreachable states: the intruder learns a term in the language described by the grammar only if he/she knew another term in the language in a past state

Motivating Example

- Consider protocol with:
  - Two operators
    - $e(K, X)$ stands for encryption of message X with key K
    - $d(K, X)$ stands for decryption of message X with key K
  - Two regular strands: (Dolev-Yao):
    - $[-(X), +(d(k, X))]$
    - $[+(e(k, r))]$

    Two Intruder strands
    - $[-(K), -(X), +(d(K, X))]$
    - $[-(K), -(X), +(e(K, X))]$
  - One equation
    - $d(K, e(K, X)) = X$

A Partial (Backwards) Search Tree

$$
\begin{tikzcd}
 & t \arrow[d] & & \\
 & \{e(k,t)\} \arrow[dl] \arrow[dr] & & \\
\{k,t\} \arrow[d] & & \{e(k,e(k,t))\} \arrow[dl] \arrow[dr] & \\
\text{stop} & \{e(k,t),k\} \arrow[d] & & \{e(k,e(k,e(k,t)))\} \arrow[d] \\
 & \text{stop} & & \cdots
\end{tikzcd}
$$

Powerful tools:

(1) Learn-only-once: terms the intruder will learn in the future and doesn't know in he past.

(2) Unreachable states: the intruder learns a term in a family only if he/she knew another term in that family in a past state

(1) Learn-Only-Once Restriction

- Suppose in looking for a term t, you find a state where the intruder knows the same t, then cut the search space

$$
\begin{tikzcd}
& t \arrow[d] \\
& \{e(k,t)\} \arrow[dl] \\
\{k,t\} \arrow[d] & \\
\text{stop} &
\end{tikzcd}
$$

- Can tell if intruder has not learned X by seeing if intruder will learn X in the future

(2) Grammars characterizing unreachable states

$$
\begin{array}{lll}
Z \npreceq r & & \\
\downarrow & & \\
\{e(K,Z)\} & & \\
\quad\searrow & & \\
 & \{e(K,e(K,Z))\} & \\
 & \quad\searrow & \\
 & & \{e(K,e(K,e(K,Z)))\} \\
 & & \downarrow \\
 & & \ldots
\end{array}
$$

- Discover Grammars providing infinite set of terms intruder can't learn.
  1. $t \in L$
  2. $Z \in L \mapsto e(Y,Z) \in L$

  1. $Z \notin \mathcal{I},\ Z \npreceq r \mapsto e(A,Z) \in L$ ($Z \npreceq r$ means Z not subsumed by r)
  2. $Z \in L \mapsto e(Y,Z) \in L$
- If the intruder learns a term in the language, then he/she must have learned another term in a state in the past.

Grammar Generation Is Automated

- Start with initial grammar, giving a single term known by the intruder, along with conditions on the term, such as some sub term not yet known by the intruder
  - Maude-NPA uses function symbol definitions in protocol spec as source for initial grammars
  - User can define own initial grammars if desired, either in addition to or in place of Maude-NPA grammars
- Maude-NPA finds the terms the intruder needed to know to generate these terms
- Checks if new terms are also in the language defined by the grammar
- If not, uses a set of heuristics to add new grammar rules
- If no heuristic applies, adds an exception to the grammar rule
- Repeats this process until it reaches a fixed point
- In cases Maude-NPA fails to generate a grammar, it provides the reasons for its failure

Status of Grammars

- Grammar generation heuristics little changed from original NRL Protocol Analyzer
- Works well on most theories we've tried
- Main exceptions are exclusive-or and Abelian groups: presence of inverses causes unexpected behavior
    - Grammar generation heuristics rely on assumptions about term on LHS of grammar rule being sub term of RHS of grammar rule
    - Not satisfied by grammars produced by these theories
    - Have a partial work-around for exclusive-or
- Currently planning to rethink grammars in order to address this

Outline

Other Ways of Reducing Search Space

- Grammars can reduce infinite to finite, but may still need to cut search space size for efficiency purposes
  - In some cases, grammars alone not enough to reduce infinite to finite, and we need other techniques as well
- We have developed a number of different techniques, and we describe them now
  - Execute Rule 1 First
  - Subsumption Partial Order Reduction
  - Use Power of Strands to See Into Past and Future
  - Super-Lazy Intruder

Execute Rule 1 First

- If there is a strand of the form $[\ l_1, u^- \mid l_2\ ]$ present, execute the rule replacing it by $[\ l_1 \mid u^-, l_2\ ]\ , u \in \mathcal{I}$ first
- If there are several fix an order and execute them all first, in that order
- Removes extra step introduced by converting negative terms to intruder terms
- Implementing this doubled the speed of the tool
    - Not surprising, because replaced two steps by one

Subsumption Partial Order Reduction

- Partial order reduction standard idea in model checking, used in a lot of protocol analysis tools, too
  - Identify when reachability of state S_1 implies reachability of S_2 and remove S_1
  - In Maude-NPA, this happens, roughly, when $S_2 \subseteq S =_B \sigma S_1$ for some substitution σ
  - Can then eliminate S_1

Using the Power of Strands

- Strands allow you to see the past and the future of a local execution
- Helpful since Maude-NPA very sensitive to the past and future
- Things we've done so far
  - If a term $x \notin \mathcal{I}$ and a strand $[\ l_1, -(x), l_2 \mid l_3\ ]$ both appear in a state, then the state is unreachable
    - Reaching it would require violation of intruder-learns-once
  - Let f and g be two terms containing $n(A, r)$. If
    - $f \in \mathcal{I}$ appears in a state, and;
    - $[\ l_1 \mid l_2, +(g), l_3,\ ]$ also appears, with strand identifier containing r and no $n(A, r)$ term in l_1;

    Then reaching the state requires the intruder to learn a nonce before it is generated and thus is unreachable.

Super-Lazy Intruder

- Based on an idea of David Basin, plus a trick used by the old NPA
- If a term $X \in \mathcal{I}$ appears in a state, where X is a variable, we assume that the intruder can easily find x, and so safe to drop it
- Super-lazy intruder: drop terms made out of variable terms, e.g. X;Y and e(K,Y)
- Need to revive variable terms if they later become instantiated
- Solution: keep the term, and state it appears in, around as a "ghost"
  - Revive the ghost, replacing current state by ghost term and ghost state, but with current substitutions to variables if any variable subterm becomes instantiated

Experimental Results 1

Protocol	none					Grammars					%
NSPK	5	19	136	642	4021	4	12	49	185	758	81
NSL	5	19	136	642	4019	4	12	50	190	804	79
SecReT06	1	6	22	119	346	1	2	6	15	36	89
SecReT07	6	20	140	635	4854	6	17	111	493	3823	21
DH	1	14	38	151	816	1	6	14	37	105	87

Protocol	none					Input First					%
NSPK	5	19	136	642	4021	11	123	1669	26432	N/A	0
NSL	5	19	136	642	4019	11	123	1666	26291	N/A	0
SecReT06	1	6	22	119	346	11	133	1977	32098	N/A	0
SecReT07	6	20	140	635	4854	11	127	3402	N/A	N/A	0
DH	1	14	38	151	816	14	135	1991	44157	N/A	0

Protocol	none					Inconsistency					%
NSPK	5	19	136	642	4021	5	18	95	310	650	83
NSL	5	19	136	642	4019	5	18	95	310	650	83
SecReT06	1	6	22	119	346	1	6	22	114	326	5
SecReT07	6	20	140	635	4854	6	18	107	439	3335	31
DH	1	14	38	151	816	1	12	12	56	128	84

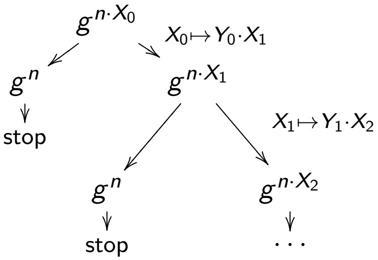
Experimental Results 2

Protocol	none					Transition Subsumption					%
NSPK	5	19	136	642	4021	5	15	61	107	237	94
NSL	5	19	136	642	4019	5	15	61	107	237	94
SecReT06	1	6	22	119	346	1	6	15	39	78	77
SecReT07	6	20	140	635	4854	6	15	61	165	506	89
DH	1	14	38	151	816	1	14	26	102	291	64

Protocol	none					Super-lazy Intruder					%
NSPK	5	19	136	642	4021	5	19	136	641	3951	1
NSL	5	19	136	642	4019	5	19	136	641	3949	2
SecReT06	1	6	22	119	346	1	6	22	119	340	2
SecReT07	6	20	140	635	4854	6	16	44	134	424	91
DH	1	14	38	151	816	1	14	38	138	525	35

Protocol	none					All optimizations					%
NSPK	5	19	136	642	4021	4	6	4	2	1	99
NSL	5	19	136	642	4019	4	7	6	2	0	99
SecReT06	1	6	22	119	346	2	3	2	-	-	99
SecReT07	6	20	140	635	4854	5	1	1	1	-	99
DH	1	14	38	151	816	4	6	10	9	12	99

Infinite Behavior We Aren't Able to Prevent



- Different from rewrite-rule based grammar behavior, because infinite behavior results from substitution
- Root term grows larger instead of leaf terms
- Behavior becomes more common as theories grow more complex
- Currently we can just cut off branch after a certain point
- But, would like a method that guarantees completeness

References

- Escobar, Santiago, Catherine Meadows, and José Meseguer. .A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties." Theoretical Computer Science 367.1 (2006): 162-202. Most thorough discussion of grammars
- Santiago Escobar, Catherine Meadows, José Meseguer, Sonia Santiago: State space reduction in the Maude-NRL Protocol Analyzer. Inf. Comput. 238: 157-186 (2014) Describes all other state space reduction techniques in this talk.

Outline

1. Controlling the Search Space
   - Learn-Only-Once and Grammars
   - Other Ways of Reducing the Search Space
2. Asymmetric Unification
   - Background and Motivation
   - A New Unification Paradigm: Asymmetric Unification

Outline

The Problem

- Protocol analysis tools often depend on syntactic properties of terms that fail to be invariant under equational theories
- Check for nonces appearing as subterms
  - Logical systems: CPSA, PCL, PDL, to determine which actions should precede others
  - Maude-NPA: to rule out unreachable states
- Depth of terms
  - ProVerif : option to ensure termination
- Syntactic pattern matching
  - Maude-NPA : to rule out infinite search paths
- In this lecture I'll describe how we're dealing with this problem in Maude-NPA, and how we think our techniques could be applied to other unification tools

An Example

- Start with exclusive-or $\oplus$
  - $\oplus$ is AC, with additional equations $x \oplus 0 = x$ and $x \oplus x = 0$.
- Consider the following protocol
  1. $A \to B : pke(B, N_A)$
  2. $B \to A : N_B \oplus N_A$
- A checks that the message she receives is $Z \oplus N_A$ for some Z
  - How it works in Maude-NPA
  - `::r::[nil, +(pke(B,n(A,r))),-(Z [+] n(A,r)), nil ]`
  - Unify `Z [+] n(A,r) with some term in the intruder's knowledge`
- So, what if $Z = Y \oplus N_A$?

How we handle this in Maude-NPA

- Express equational theory as $R = \{X \oplus 0 \to X, X \oplus X \to 0, X \oplus X \oplus X \to X\} \uplus \Delta = \mathrm{AC}$
    - nonce containment invariant under AC
    - R is a set of rewrite rules convergent and terminating wrt AC
- Find all the possible reduced forms of `Z [+] n(A,r)` wrt R
    - There are two:
        - `< Z [+] n(A,r), id >`
        - `< Y, Z / Y [+] n(A,r) >`
- One strand for each reduced form
    - `::r::[nil, +(pke(B,n(A,r))),-(Z [+] n(A,r)), nil ]`
    - `::r::[nil, +(pke(B,n(A,r))),-(Y), nil ]`
- Include constraints that negative terms in strands are irreducible wrt R
- Any further substitution made in the search process must obey these constraints

Three Things we need to make this work

1. Characterize theories in which every term has a finite number of reduced forms
   - We understand this: this is equivalent to the finite variant property
2. Unification algorithms giving a set of mgu's Σ $x =?y$ such that for all $\sigma \in \Sigma$, σy is irreducible
   - We call this *asymmetric unification*
   - *Variant narrowing* has this property, we are looking for more efficient algorithms
3. Combine these into a sound and complete search strategy

Using Variant Narrowing to Satisfy Irreducibility Constraints in Protocol Analysis

- Recap from previous lecture
- For each strand St in a specification, compute a most general set of variants V of the the negative terms
- For each variant (σ, t) create a new strand σSt
- Each time a positive term s is unified what a term t in the intruder knowledge, user asymmetric unification to find a complete set of unifiers of s and t that leave t irreducible, as well as all negative terms already present in the state

Outline

2. Asymmetric Unification
   - Background and Motivation
   - A New Unification Paradigm: Asymmetric Unification

Asymmetric Unification

- Let $(\Sigma, R \uplus \Delta)$ be a an equational theory, where R is a set of rewrite rules confluent, terminating and coherent wrt Δ.
- A solution to an asymmetric unification problem $s_1 =_{\downarrow} t_1 \wedge \ldots \wedge s_k =_{\downarrow} t_k$ is a substitution σ such that
  1. For each i, $\sigma s_i =_{\Delta} \sigma t_i$
  2. For each i, σt_i is irreducible
- A set Θ is a most general set of asymmetric unifiers of P if for any asymmetric unifier there σ there is a $\theta \in \Theta$ such that $\sigma =_{\Delta} \tau\theta$ for some τ.

Some Examples from XOR

1. $c =?X \oplus Y$, S-unifiable, but not A-unifiable
2. $a + b =?X \oplus Y$
   - $\sigma = [X \mapsto Y \oplus a \oplus b]$ is a most general S-unifier, but not an A-unifier
   - $[X \mapsto a, Y \mapsto b], [X \mapsto b, Y \mapsto a]$ is a set of most general A-unifiers
3. $X =?Y \oplus X$
   - $[Y \mapsto X \oplus Z]$ an a most general S-unifier but not an A-unifier
   - $[X \mapsto Y \oplus Z]$ is equivalent, but is an A-unifier
4. $Z =?X_1 \oplus X_2, Z =?Y_1 \oplus Y_2$
   - $\sigma = [X_1 \mapsto Z \oplus X_2, Y_1 \mapsto X \oplus Y_2]$ is a most-general S-unifier, but not an A-unifier
   - $\sigma = [Z \mapsto X_1 \oplus V \oplus Y_2, X_2 \oplus V \oplus Y_2, Y_1 \oplus X_1 \oplus V]$ is equivalent and an A-unifier

Using Variant Narrowing to Find Most General Set of Asymmetric Unifiers

- Let $(\Sigma, R \uplus \Delta)$ be a an equational theory, where R is a set of rewrite rules confluent, terminating and coherent wrt Δ.
- Furthermore, assume that $(R \uplus \Delta)$ has the finite variant property
- Given a problem $s =_{\downarrow} t$
  1. Use variant narrowing to find a set of most general variants V of s.
  2. Discard any $(\sigma, \sigma s \downarrow) \in V$ such that σt is reducible
  3. For each remaining $(\sigma, \sigma s \downarrow)$ find set of mgu's of $\sigma s \downarrow =_E ? \sigma t$.
  4. Discard any unifier θ such that $\theta \sigma t$ is reducible
  5. Remaining set is a set of most general asymmetric unifiers
- Can we do better (e.g. faster)?

Next Step: Asymmetric Unification (AU) as a Problem in its Own Right

- As far as we can tell, no-one has studied this before
- Narrowing only algorithm we know of that can achieve this
- What we do have found so far
  - AU at least as hard as symmetric unification (SU)
    - Any SU problem $s =?t$ can be turned into AU problem $s =?X, t =?X$.
  - AU *strictly harder* than SU - XOR without any other symbols is in P for SU but NP-complete for AS
  - SU can be unitary while AU is not (XOR)
  - There exist theories for which SU is decidable but AU is not
- We are working on a general approach for converting equational unification algorithms to asymmetric unification algorithms
- Have applied it to unification in XOR theory (ACU + cancellation)

Outline for a General Procedure

- Start with a decomposition $R \uplus Delta$ and a unification algorithm
- GIven a problem $x =?y$, find a complete set of unifiers Σ using the symmetric algorithm
- For each $\sigma \in \Sigma$
  1. If σ is an A-unifier, keep it
  2. If not, see if there is an equivalent A-unifier σ' and if so, replace σ with σ'
  3. If not, apply 1) and 2) to more completely instantiated versions σ and replace σ' with those
  4. If none of those work, discard σ
- Application to exclusive-or given in Ertabur et al. 2012 and 2013.
- Papers also includes experimental results

Experimental Results: Unification

Unif. Problem	T. A-V	# A-V	T. D-A	# D-A	% T.	% #
$NS_1 \oplus NS_2 =_{\downarrow} NS_3 \oplus N_A$	153	12	153	1	0	91
$NS_1 \oplus N_A =_{\downarrow} NS_2 \oplus NS_3$	137	5	121	1	11	80
$NS_1 \oplus NS_2 =_{\downarrow} NS_3 \oplus NS_4 \oplus NS_5$	286	54	116	1	59	98
$NS_1 \oplus NS_2 =_{\downarrow} NS_3 \oplus NS_4 \oplus N_A$	159	36	115	1	27	97
$NS_1 \oplus NS_2 =_{\downarrow} N_A$	127	4	114	1	10	75
$NS_1 \oplus NS_2 =_{\downarrow} null$	128	1	105	1	17	0
$NS_1 \oplus NS_2 =_{\downarrow} null \oplus NS_3$	130	7	105	1	20	85
$M_1 \oplus M_2 =_{\downarrow} M_3 \oplus pair(V_1, M_4)$	51	12	44	1	13	91
$pair(V, rc4(V_1, kAB) \oplus ([N_A, c(N_A)]))$ $=_{\downarrow} pair(V_1, M_1)$	30	1	29	1	3	0
$M_1 \oplus M_2 =_{\downarrow} M_3 \oplus V_1$	33	12	32	1	3	91
$M_1 \oplus M_2 =_{\downarrow} M_3 \oplus ([N_1, c(N_2)])$	34	12	30	1	11	91
$M_1 \oplus M_2 =_{\downarrow} M_3 \oplus pair(V_1, pair(V_2, M_4))$	36	12	30	1	16	91

- T. A-V , # A-V = time and number of unifiers, resp. of variant unification
- T. D-A, # D-A = time and number of unifiers, resp. of asymmetric algorithm
- % T. and % # = improvement represented as percentage of A-V score

Experimental Results: Protocol Analysis

states/seconds	1 step	2 steps	3 steps	4 steps	5 steps
RP - Standard	2/0.08	5/0.16	13/0.86	49/3.09	267/17.41
RP - Asymmetric	1/0.03	45/1.08	114/2.26	1175/37.25	13906/4144.30
WEPP - Standard	5/0.09	9/0.42	26/1.27	106/5.80	503/ 34.76
WEPP - Asymmetric	4/0.05	9/0.12	26/0.64	257/144.65	2454/612.08
TMN - Standard	5/0.11	15/ 0.55	99/3.82	469/ 25.68	timeout
TMN - Asymmetric	4/0.06	24/0.53	174/3.63	1079/170.29	9737/1372.55

- Protocol analysis experiments with regular XOR unification algorithm vs using asymmetric XOR unification algorithm.
- A pair n/t means: $n =$ number of states, and $t =$ time in seconds

Asymmetric Unification Over Combinations of Theories : Strategy 1

- Suppose that $E_1 = (R_1 \uplus \Delta_1)$ and $E_2 = (R_2 \uplus \Delta_2)$ have asymmetric unification algorithms $\mathcal{A}_\infty, \mathcal{A}_\in$
- How do we find an asymmetric algorithm $\mathcal{A}$ for $E = ((R_1 \cup R_2) \uplus (\Delta_1 \cup \Delta_2))$?
- $(R_1 \uplus \Delta_1)$ and $(R_2 \uplus \Delta_2)$ are both finite variant decompositions, and a unification algorithm exists for $\Delta_1 \cup \Delta_2$
  - If $((R_1 \cup R_2) \uplus (\Delta_1 \Delta_2))$ is FVP, then can use variant narrowing
  - Although this is not decidable, tools and semi decision procedures exist

Asymmetric Unification Over Combinations of Theories: Strategy 2

- If both $\mathcal{A}_\infty$ and $\mathcal{A}_\in$ satisfy a condition known as *linear constant restriction* than a general combination procedure exists (Erbatur et al., 2014), based on Baader-Schwarz method (1996)
  - Linear constant restriction basically means algorithm still works with additional free constants added to theory
  - Needed for Baader-Schwarz result to
  - $\mathcal{A}$ highly nondeterministic, but may be able to adapt known optimization techniques
- Both strategies result in inefficient algorithms: can we do better?

Conclusion

- We've described a way of decomposing and attacking unification problems that we originally adopted for convenience in Maude-NPA
- We found that it also makes it much easier to apply state space reduction techniques that rely upon syntactic checking
- We believe that this has applications, not only to Maude-NPA, but to other tools and approaches as well
- With that in mind, we are starting to investigate this approach in a more general and systematic way

Asymmetric Unification References

- Serdar Erbatur, Santiago Escobar, Deepak Kapur, Zhiqiang Liu, Christopher Lynch, Catherine Meadows, José Meseguer, Paliath Narendran, Sonia Santiago, Ralf Sasse. Effective Symbolic Protocol Analysis via Equational Irreducibility Conditions.. In Proceedings of ESORICS 2012, Springer-Verlag, 2012.
- Serdar Erbatur, Santiago Escobar, Deepak Kapur, Zhiqiang Liu, Christopher Lynch, Catherine Meadows, José Meseguer, Paliath Narendran, Sonia Santiago, Ralf Sasse. Asymmetric unification: A new unification paradigm for cryptographic protocol analysis. In proceedings of CADE 2013, Springer-Verlag, 2013.
- Serdar Erbatur, Deepak Kapur, Andrew M. Marshall, Catherine Meadows, Paliath Narendran, Christophe Ringeissen: On Asymmetric Unification and the Combination Problem in Disjoint Theories. FoSSaCS 2014.