MSR 3.0:
The Logical Meeting Point of Multiset Rewriting and Process Algebra

Iliano Cervesato
iliano@itd.nrl.navy.mil

ITT Industries, inc @ NRL Washington, DC

http://theory.stanford.edu/~iliano

ISSS 2003, Tokyo, Japan
November 4-6, 2003
Brief History of MSR

• **MSR 1**  [*CSFW’99*]
  - To formalize security protocols specification
  - First-order multiset rewriting with $\exists$
    - Undecidability of security protocol verification
    - *Comparison with Strand Spaces*

• **MSR 2**  [*MMM’01*]
  - Add typing infrastructure, liberalize syntax
    - Specification of Kerberos V
    - Completeness of Dolev-Yao attacker
    - Subsorting view of type-flaw attacks
    - Implementation (undergoing)
    - *Comparison with Process Algebra*
From multisets to $\omega$-multisets

- Embeds multiset rewriting
  - MSR 1, 2
  - Paulson’s inductive traces
  - Tool-specific languages
    - NRL Protocol Analyzer
    - Mur$\phi$, ...

- Encompasses Process Algebra
  - Strand spaces
  - Crypto-SPA
  - Spi-calculus

- Founded on logic

Indirect contributors
- Fabio Martinelli
- Dale Miller
- Andre Scedrov
- Frank Pfenning
What is in MSR 3?

- Instance of $\omega$-multisets for cryptographic protocol specification
- Security-relevant signature
  - Network
  - Encryption, ...
- Typing infrastructure
  - Dependent types
  - Subsorting
- Data Access Specification (DAS)
- Module system
- Equations

From MSR 1

From MSR 2

From MSR 2 implementation
ω-Multisets

Specification language for concurrent systems

• Crossroad of
  - State transition languages
    - Petri nets, multiset rewriting, ...
  - Process calculi
    - CCS, π-calculus, ...
  - (Linear) logic

• Benefits
  - Analysis methods from logic and type theory
  - Common ground for comparing
    - Multiset rewriting
    - Process algebra
  - Allows multiple styles of specification
    - Unified approach
Syntax

\[ a ::= P \quad \text{atomic object} \]
\[ \bullet \quad \text{empty} \]
\[ a, b \quad \text{formation} \]
\[ a \rightarrow b \quad \text{rewrite} \]
\[ \forall x. a \quad \text{instantiation} \]
\[ \exists x. a \quad \text{generation} \]

Generalizes FO multiset rewriting (MSR 1-2)

\[ \forall x_1 \ldots x_n. a(x) \rightarrow \exists y_1 \ldots y_k. b(x, y) \]
Judgments

• **Base step**
  \[ \Sigma ; s \rightarrow_R \Sigma' ; s' \]

• **Finite iteration**
  \[ \Sigma ; s \rightarrow^* R \Sigma' ; s' \]
  - Reflexive and transitive closure of \( \rightarrow \)
  - Useful for reachability analysis
## Operational Semantics

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>→</td>
<td>$\Sigma : (s, a, a \rightarrow b)$</td>
<td>$\rightarrow_R$</td>
<td>$\Sigma : (s, b)$</td>
<td></td>
</tr>
<tr>
<td>∀</td>
<td>$\Sigma : (s, \forall x. a)$</td>
<td>$\rightarrow_R$</td>
<td>$\Sigma : (s, [t/x]a)$</td>
<td>if $\Sigma</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>∃</td>
<td>$\Sigma : (s, \exists x. a)$</td>
<td>$\rightarrow_R$</td>
<td>$(\Sigma, x) : (s, a)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\Sigma : s$</td>
<td>$\rightarrow_{R,a}$</td>
<td>$\Sigma : (s, a)$</td>
<td></td>
</tr>
</tbody>
</table>

**ω-Multisets**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Sigma ; s \rightarrow^*_{R} \Sigma ; s$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\Sigma ; s \rightarrow^*_{R} \Sigma'' ; s''$</td>
<td>if $\Sigma ; s \rightarrow_{R} \Sigma' ; s'$</td>
<td>and $\Sigma' ; s' \rightarrow^*_{R} \Sigma'' ; s''$</td>
<td></td>
</tr>
</tbody>
</table>
Logical Foundations

- is left sequent rules of linear logic
  - $\bullet \equiv 1$, $\equiv \otimes \rightarrow \equiv \rightarrow 0$

- MSR 3 is logic
  - Guideline for extensions with new operators
    - Non-deterministic choice (+)
    - Replication (!)
    - ... more

- Related to Concurrent Logical Framework
ω-Multiset View of Derivations

- Step up: 
  - Left rules

- Step across: 
  - Axiom

- Right rules not used
The Atomic Objects of MSR 3

**Atomic terms**
- **Principals**: \( A \)
- **Keys**: \( K \)
- **Nonces**: \( N \)
- **Other**
  - Raw data, timestamp, ...

**Constructors**
- **Encryption**: \( \{\_\}_\_ \)
- **Pairing**: \( (_, _) \)
- **Other**
  - Signature, hash, MAC, ...

**Predicates**
- **Network**: \( net \)
- **Memory**: \( M_A \)
- **Intruder**: \( I \)
- **...**
Types

• Simple types
  - A : princ
  - n : nonce
  - m : msg, ...

• Dependent types
  - k : shK A B
  - K : pubK A
  - K' : privK K, ...

Fully definable

• Powerful abstraction mechanism
  - At various user-definable level
    - Finely tagged messages
    - Untyped: msg only

• Simplify specification and reasoning

• Automated type checking
Subsorting

\(\tau \prec \tau'\)

- Allows atomic terms in messages
- **Definable**
  - Non-transmittable terms
  - Sub-hierarchies
- Discriminant for type-flaw attacks
Data Access Specification

- Prevent illegitimate use of information
  - Protocol specification divided in roles
    - Owner = principal executing the role
      - A signing/encrypting with B’s key
      - A accessing B’s private data, ...

- Simple static check

- Central meta-theoretic notion
  - Detailed specification of Dolev-Yao access model

- Gives meaning to Dolev-Yao intruder

- Current effort towards integration in type system
  - Definable
    - Possibility of going beyond Dolev-Yao model
Modules and Equations

• Modules
  - Bundle declarations with simple import/export interface
  - Keep specifications tidy
  - Reusable

• Equations
  (For free from underlying Maude engine)
  - Specify useful algebraic properties
    - Associativity of pairs
  - Allow to go beyond free-algebra model
    - \( \text{Dec}(k, \text{Enc}(k, M)) = M \)
Example

Needham-Schroeder public-key protocol

1. \( A \to B: \{n_A, A\}_{KB} \)
2. \( B \to A: \{n_A, n_B\}_{KA} \)
3. \( A \to B: \{n_B\}_{KB} \)

- Can be expressed in several ways
  - State-based
    - Explicit local state
    - As in MSR 2
  - Process-based: embedded
    - Continuation-passing style
    - As in process algebra
  - (Intermediate approaches)
∀ A: princ.
{ ∃ L: princ × ∑ B: princ.pubK B × nonce → mset.

∀ B: princ. ∀ k_B: pubK B.
  •
  → ∃ n_A: nonce.
  net {{n_A, A}_kB}, L (A, B, k_B, n_A)

∀ B: princ. ∀ k_B: pubK B.
∀ k_A: pubK A. ∀ k_A': prvK k_A.
∀ n_A: nonce. ∀ n_B: nonce.
  net {{n_A, n_B}_kA}, L (A, B, k_B, n_A)
  → net {{n_B}_kB}

}
Process-Based

∀A:princ.
∀B: princ. ∀k_B: pubK B.

• → ∃n_A: nonce.
  net (\{n_A, A\}_k_B),

(∀k_A: pubK A. ∀k'_A: prvK k_A. ∀n_B: nonce.
  net (\{n_A, n_B\}_k_A) → net (\{n_B\}_k_B))

- Succinct
- Continuation-passing style
  - Rule asserts what to do next
  - Lexical control flow
- State is implicit
  - Abstract

A → B: \{n_A, A\}_k_B
B → A: \{n_A, n_B\}_k_A
A → B: \{n_B\}_k_B
NSPK in Process Algebra

∀A:princ.
∀B: princ. ∀kB: pubK B.
∀kA: pubK A. ∀kA': prvK kA. ∀nB: nonce.
∀nA: nonce.
net ({nA, A}{kB}).
net <{nA, nB}{kA}>
net ({nB}{kB}). 0

Same structure!
- Not a coincidence
- MSR 3 very close to Process Algebra
  - ω-multiset encodings of π-calculus
  - Ties to Join Calculus

• MSR 3 is ideal middle-ground for relating
  - State-based
  - Process-based
  representations of a problem
State-Based vs. Process-Based

- **State-based languages**
  - Multiset Rewriting
  - NRL Prot. Analyzer, CAPSL/CIL, Paulson’s approach, …
  - State transition semantics

- **Process-based languages**
  - Process Algebra
  - Strand spaces, spi-calculus, …
  - Independent communicating threads
MSR 3 Bridges the Gap

- Difficult to go from one to the other
  - Different paradigms

\[ \omega \text{-Multisets} \]

MSR 2

Protocols

Repr. gap

State vs. process distance

\[ \leftrightarrow \]

Other distance

\[ \text{State} \leftrightarrow \text{Process translation done once and for all in MSR 3} \]
Summary

- **MSR 3.0**
  - Language for security protocol specification
  - Succinct representations
    - Simpl specifications
    - Economy of reasoning
  - Bridge between
    - State-based representation
    - Process-based representation

- **ω-multisets**
  - Logical foundation of multiset rewriting
  - Relationship with process algebras
  - Unified logical view
    - Better understanding of where we are
    - Hint about where to go next