MSR 3:
The Logical Meeting Point of Multiset
Rewriting and Process Algebra

Iliano Cervesato
iliano@itd.nrl.navy.mil
ITT Industries, inc @ NRL Washington, DC
http://www.cs.stanford.edu/~iliano

CS Department, UMBC
February 27-28, 2003

Iliano Cervesato
iliano@itd.nrl.navy.mil

ITT Industries, inc @ NRL Washington, DC
http://theory.stanford.edu/~iliano

Protocol eXchange Seminar, UMBC
May 27-28, 2004
NSPK in MSR 3

∀A: princ.
{∃L: princ × ∑B: princ.pubK B × nonce → mset.

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

Interpretation of L

- Rule invocation
  - Implementation detail
  - Control flow
- Local state of role
  - Explicit view
  - Important for DOS

MSR 2 spec.
NSPK in MSR 3

∀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

MSR 3: One Year Later

A → B: \{n_A, A\}_{k_B}
B → A: \{n_A, n_B\}_{k_A}
A → B: \{n_B\}_{k_B}

Not an MSR 2 spec.
Looks Familiar?

Process calculus

∀A: princ.
∀B: princ. ∀k_B: pubK B.
∀k_A: pubK A. ∀k_A ': prvK k_A. ∀n_B: nonce.

∀n_A: nonce.
net ({n_A, A}_{k_B}).
net \langle {n_A, n_B}_{k_A} \rangle.
net ({n_B}_{k_B}). 0

Parametric strand

Alice (A, B, N_A, N_B):
N_A Fresh, \pi_A (A, B)

{N_A, A}_{k_B} \rightarrow
\downarrow
\downarrow
\downarrow
\downarrow
\rightarrow
{N_A, N_B}_{k_A}
{N_B}_{k_B}
What is MSR 3?

A new language for security protocols

- Supports
  - State transition specs
    - Conservative over MSR 2
  - Process algebraic specs

- Rewriting re-interpretation of logic
  - Rich composable set of connectives

- Universal connector
More than the Sum of its Parts

Process- and transition-based specs. in the same language

- Choose the paradigm
  - User’s preference
  - Highlight characteristics of interest
  - Support various verification techniques (FW)

- Mix and match styles
  - Within a spec.
  - Within a protocol
  - Within a role
What is in MSR 3?

- **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

\[ A ::= \begin{align*}
    & a & \text{atomic object} \\
    | & 1 & \text{empty} \\
    | & A \otimes B & [A, B] \text{formation} \\
    | & A \rightarrow B & [A \rightarrow B] \text{rewrite} \\
    | & T & \text{no-op} \\
    | & A \& B & [A \parallel B] \text{choice} \\
    | & \forall x. A & \text{instantiation} \\
    | & \exists x. A & \text{generation} \\
    | & ! A & \text{replication}
\end{align*} \]

Generalizes FO multiset rewriting (MSR 1-2)

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

• States

\[ \Sigma ; \; \Gamma \; ; \; \Delta \]
\[ \Sigma \; ; \; \Delta \]

- \( \Sigma \) is a list
- \( \Gamma \) and \( \Delta \) are commutative monoids

• Transitions

\[ \Sigma; \; \Gamma; \; \Delta \rightarrow \Sigma'; \; \Gamma'; \; \Delta' \]
\[ \Sigma; \; \Gamma; \; \Delta \rightarrow^* \Sigma'; \; \Delta' \]

- \( \rightarrow^* \) for reflexive and transitive closure

- Constructor: “,”
- Empty: “•”
Transition Semantics

\[ \sigma \quad \Sigma ; \Gamma ; (\Delta, A, A \rightarrow B) \rightarrow \Sigma ; \Gamma ; (\Delta, B) \]
\[ \top \quad \text{(no rule)} \]
\[ \& \quad \Sigma ; \Gamma ; (\Delta, A_1 \& A_2) \rightarrow \Sigma ; \Gamma ; (\Delta, A_i) \]
\[ \forall \quad \Sigma ; \Gamma ; (\Delta, \forall x. A) \rightarrow \Sigma ; \Gamma ; (\Delta, [\tau/x]A) \]
\[ \text{if } \Sigma \vdash \tau \]
\[ \exists \quad \Sigma ; \Gamma ; (\Delta, \exists x. A) \rightarrow (\Sigma, x) ; \Gamma ; (\Delta, A) \]
\[ ! \quad \Sigma ; \Gamma ; (\Delta, !A) \rightarrow \Sigma ; (\Gamma, A) ; \Delta \]
\[ \Sigma ; (\Gamma, A) ; \Delta \rightarrow \Sigma ; (\Gamma, A) ; (\Delta, A) \]

\[ \Sigma ; \Gamma ; \Delta \Rightarrow^* \Sigma ; \Delta \]
\[ \Sigma ; \Gamma ; \Delta \Rightarrow^* \Sigma'' ; \Delta'' \]
\[ \text{if } \Sigma ; \Gamma ; \Delta \Rightarrow \Sigma' ; \Gamma' ; \Delta' \text{ and } \Sigma' ; \Gamma' ; \Delta' \Rightarrow^* \Sigma'' ; \Delta'' \]
Linear Logic

- **Formulas**
  \[ A, B ::= a \mid 1 \mid A \otimes B \mid A \multimap B \mid ! A \mid T \mid A \& B \mid \forall x. A \mid \exists x. A \]

- **LV sequents**
  \( \Gamma ; \Delta \quad \rightarrow \quad \Sigma \quad \Rightarrow \quad \mathcal{C} \)

- Unrestricted context
- Linear context
- Signature
- Goal formula

- **Constructor:** “,”
- **Empty:** “•”
Logical Derivations

- Proof of $C$ from $\Delta$ and $\Gamma$
  - Emphasis on $C$
    - $C$ is input
- Finite
  - Closed
- Rules shown
  - Major premise
    - Preserves $C$
  - Minor premise
    - Starts subderivation
A Rewriting Re-Interpretation

- **Transition**
  - From conclusion
  - To major premise
  - Emphasis on $\Gamma$, $\Delta$, and $\Sigma$
  - $C$ is output, at best
    - Does not change

- **Possibly infinite**
  - Open

- **Minor premise**
  - Auxiliary rewrite chain
    - Finite
  - Topped with axiom
Interpreting Unary Rules

\[
\begin{align*}
\frac{\Gamma; \Delta, A, B \rightarrow_{\Sigma} C}{\Gamma; \Delta, A \otimes B \rightarrow_{\Sigma} C} & \quad \text{\( \Sigma; \Gamma; (\Delta, A \otimes B) \rightarrow \Sigma; \Gamma; (\Delta, A, B) \)} \\
\frac{\Sigma |- \top \quad \frac{\Gamma; \Delta, [t/x]A \rightarrow_{\Sigma} C}{\Gamma; \Delta, \forall x. A \rightarrow_{\Sigma} C}}{\Sigma; \Gamma; (\Delta, \forall x. A) \rightarrow \Sigma; \Gamma; (\Delta, [t/x]A)} & \quad \text{\( \text{if } \Sigma |- \top \)} \\
\frac{\Gamma; \Delta, A \rightarrow_{\Sigma} C}{\Gamma; \Delta, \exists x. A \rightarrow_{\Sigma} C} & \quad \text{\( \Sigma; \Gamma; (\Delta, \exists x. A) \rightarrow (\Sigma, x); \Gamma; (\Delta, A) \)} \\
\frac{\Gamma, A; \Delta \rightarrow_{\Sigma} C}{\Gamma; \Delta, !A \rightarrow_{\Sigma} C} & \quad \text{\( \Sigma; \Gamma; (\Delta, !A) \rightarrow \Sigma; (\Gamma, A); \Delta \)}
\end{align*}
\]
Binary Rules and Axiom

- Minor premise
  - Auxiliary rewrite chain

- Top of tree
  - Focus shifts to RHS
    - Axiom rule
  - Observation

\[ \Gamma; \Delta' \rightarrow_{\Sigma} A \quad \Gamma; \Delta, B \rightarrow_{\Sigma} C \]
\[ \Gamma; \Delta, \Delta', A \rightarrow_{\Sigma} B \rightarrow_{\Sigma} C \]
Observations

- Observation states
  \[ \Sigma ; \Delta \]
  - In \( \Delta \), we identify
    - with \( \otimes \)
    - with 1
  
  Categorical semantics
  - Identified with \( \exists x_1. \ldots \exists x_n. \Delta \)
  - For \( \Sigma = x_1, \ldots, x_n \)
  
  De Bruijn's telescopes

- Observation transitions
  \[ \Sigma; \Gamma; \Delta \rightarrow^* \Sigma'; \Delta' \]
Interpreting Binary Rules

\[
\begin{align*}
\Gamma; A \rightarrow_{\Sigma} A & \quad \Sigma; \Gamma; \Delta \rightarrow^{*} \Sigma; \Delta \\
\Sigma; \Gamma; \Delta \rightarrow^{*} \Sigma''; \Delta'' & \\
& \quad \text{if} \quad \Sigma; \Gamma; \Delta \rightarrow \Sigma'; \Gamma'; \Delta' \\
& \quad \text{and} \quad \Sigma'; \Gamma'; \Delta' \rightarrow^{*} \Sigma''; \Delta''
\end{align*}
\]

\[
\begin{align*}
\Gamma; \Delta' \rightarrow_{\Sigma} A & \quad \Gamma; \Delta, B \rightarrow_{\Sigma} C \\
\Gamma; \Delta, \Delta', A \rightarrow_{o} B \rightarrow_{\Sigma} C & \\
\Gamma; \Delta, \Delta' \rightarrow_{\Sigma} C & \quad \Sigma; \Gamma; (\Delta, \Delta', A \rightarrow_{o} B) \rightarrow \Sigma; \Gamma; (\Delta, B) \\
& \quad \text{if} \quad \Sigma; \Gamma; \Delta' \rightarrow^{*} \Sigma; A
\end{align*}
\]

\[
\begin{align*}
\Gamma; \Delta' \rightarrow_{\Sigma} A & \quad \Gamma; \Delta, A \rightarrow_{\Sigma} C \\
\Gamma; \Delta, \Delta' \rightarrow_{\Sigma} C & \quad \Sigma; \Gamma; (\Delta, \Delta') \rightarrow \Sigma; \Gamma; (A, \Delta) \\
& \quad \text{if} \quad \Sigma; \Gamma; \Delta' \rightarrow^{*} \Sigma; A
\end{align*}
\]

...
Formal Correspondence

• Soundness

\[
\text{If } \Sigma ; \Gamma ; \Delta \rightarrow^* \Sigma, \Sigma'; \Delta' \text{ then } \Gamma ; \Delta \rightarrow_{\Sigma} \exists \Sigma'. \otimes \Delta'
\]

• Completeness?

➢ No! We have only crippled right rules

\[
\bullet ; \bullet ; a \rightarrow o b, \ b \rightarrow o c \quad \text{and} \quad \bullet ; a \rightarrow o c
\]
System $\omega$

- With cut, rule for $\rightarrow o$ can be simplified to $\Sigma; \Gamma; (\Delta, A, A \rightarrow o B) \rightarrow \Sigma; \Gamma; (\Delta, B)$

- Cut elimination holds
  = in-lining of auxiliary rewrite chains
  - But ...
    - Careful with extra signature symbols
    - Careful with extra persistent objects

- No rule for $\rightarrow$ needs a premise
  - $\rightarrow$ does not depend on $\Rightarrow^*$
Multiset Rewriting

- **Multiset**: set with repetitions allowed
  \[ a ::= \bullet | a, a \]
  - Commutative monoid

- Multiset rewriting (a.k.a. *Petri nets*)
  - Rewriting within the monoid
  - Fundamental model of distributed computing
    - Alternative: Process Algebras
  - Basis for security protocol spec. languages
    - MSR family
    - ... several others
  - Many extensions, more or less ad hoc
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 \(I\)
- ...

**Fully definable**

*MSR 3: One Year Later*
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

\[ \tau <: \tau' \]

- 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
    - $\text{Dec}(k, \text{Enc}(k, M)) = M$
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

State ↔ Process translation done once and for all in MSR 3
Summary

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

- **$\omega$-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