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
Motivations

- **Security protocol specifications**
  - Transition-based
  - Process-based
  - Different languages and techniques
  - Ad-hoc translations

- **Attempt at a unified approach**
  - Rewriting re-interpretation of logic
    - Open derivations
    - Left rule semantics
  - Foundation of multiset rewriting
  - Bridge to process algebra
  - Effective protocol specification language
Linear Logic

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

- **LV sequents**

  \[ \Gamma ; \Delta \rightarrow^\Sigma C \]

  - Constructor: ","
  - Empty: "·"
Some LV Rules

**Left rules**

\[ \Gamma; \Delta, A, B \rightarrow_{\Sigma} C \]
\[ \Gamma; \Delta, A \otimes B \rightarrow_{\Sigma} C \]
\[ \Gamma; \Delta', \Delta \rightarrow_{\Sigma} C \]
\[ \Gamma; \Delta, \Delta', A \rightarrow x B \rightarrow_{\Sigma} C \]
\[ \Sigma |- \dag \Gamma; \Delta, [\dag/x] A \rightarrow_{\Sigma} C \]
\[ \Gamma; \Delta, \forall x. A \rightarrow_{\Sigma} C \]
\[ \Gamma; \Delta, A \rightarrow_{\Sigma,x} C \]
\[ \Gamma; \Delta, \exists x. A \rightarrow_{\Sigma} C \]
\[ \Gamma, A; \Delta \rightarrow_{\Sigma} C \]
\[ \Gamma; \Delta, \neg A \rightarrow_{\Sigma} C \]

**Right rules**

...
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

$\Gamma; \Delta \rightarrow_{\Sigma} C$

$I. Cervesato$: The Logical Meeting Point of MSR and PA
A Rewriting Re-Interpretation

- Logic
- System \( \omega \)
- Rewriting
- Processes
- Security

I. Cervesato: The Logical Meeting Point of MSR and PA

• 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
State and Transitions

- **States**
  - $\Sigma; \Gamma; \Delta$
  - $\Sigma$ is a list
  - $\Gamma$ and $\Delta$ are commutative monoids
  - No $C$
    - Does not change

- **Transitions**
  - $\Sigma; \Gamma; \Delta \rightarrow \Sigma'; \Gamma'; \Delta'$
  - $\rightarrow^*$ for reflexive and transitive closure
Interpreting Unary Rules

\[
\begin{align*}
\Gamma; \Delta, A, B & \rightarrow^\Sigma C \\
\Gamma; \Delta, A \otimes B & \rightarrow^\Sigma C
\end{align*}
\]

\[
\begin{align*}
\Sigma; \Gamma; (\Delta, A \otimes B ) & \rightarrow \Sigma; \Gamma; (\Delta, A, B) \\
\Sigma; \Gamma; (\Delta, \forall x. A) & \rightarrow \Sigma; \Gamma; (\Delta, [t/x]A) \\
& \text{if } \Sigma |- t
\end{align*}
\]

\[
\begin{align*}
\Gamma; \Delta, A & \rightarrow^\Sigma x C \\
\Gamma; \Delta, \exists x. A & \rightarrow^\Sigma C
\end{align*}
\]

\[
\begin{align*}
\Sigma; \Gamma; (\Delta, \exists x. A) & \rightarrow (\Sigma, x); \Gamma; (\Delta, A) \\
\Sigma; \Gamma; (\Delta, !A) & \rightarrow \Sigma; (\Gamma, A); \Delta
\end{align*}
\]

\ldots

\ldots
Binary Rules and Axiom

- Minor premise
  - Auxiliary rewrite chain
- Top of tree
  - Focus shifts to RHS
    - Axiom rule
    - Observation

$\Gamma; \Delta' \rightarrow_{\Sigma} A \quad \Gamma; \Delta, B \rightarrow_{\Sigma} C$

$\Gamma; \Delta, \Delta', A \circ B \rightarrow_{\Sigma} C$
Observations

- **Observation states**

  \[ \Sigma \ ; \ \Delta \]

  - In \( \Delta \), we identify
    - \( \otimes \) with \( \times \)
    - \( \bullet \) 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

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

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

...
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

\[
\bullet ; \bullet ; a \leftarrow o b, b \leftarrow o c \quad \rightarrow^* \quad \bullet ; a \leftarrow o c
\]
System $\omega$

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

- 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
  \[ a ::= \bullet \mid a, a \]
  - 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
First-Order Multiset Rewriting

- Multiset elements are FO atomic formulas
- Rules have the form
  \[ \forall x_1 \ldots x_n. \ a(x) \rightarrow \exists y_1 \ldots y_k. \ b(x,y) \]
- Semantics

\[
\Sigma ; a(t), s \rightarrow_R (a(x) \rightarrow \exists y. b(x,y)) \quad \Sigma, y ; b(t,y), s
\]
  if \( \Sigma |- t \)

- Several encodings into linear logic
  - [Martí-Oliet, Meseguer, 91]
ω-Multisets vs. Multiset Rewriting

• MSR 1 is an instance of ω-multisets
  - Uses only ⊗, 1, ∀, ∃, and ⏞
  - ⏞ never nested, always persistent

  \[ \Sigma ; s \xrightarrow{R} \Sigma' ; s' \]
  iff \[ \Sigma ; "R" ; "s" \xrightarrow{*} \Sigma' ; "s'" \]

• Interpretation of MSR as linear logic
  ➢ Logical explanation of multiset rewriting
    ➢ MSR is logic
    ➢ Guideline to design rewrite systems
The Asynchronous $\pi$-Calculus

Another fundamental model of distributed computing

- **Language**
  
  $$P ::= 0 \mid P||Q \mid \nu x. P \mid !P \mid x(y).P \mid x<y>$$

- **Semantics**
  
  - **Structural equivalence**
    - Comm. monoidal congruence of $||$ and $0$
    - Binder mobility congruence of $\nu$
      - $\nu x. \nu y. P \equiv \nu y. \nu x. P$
      - $0 \equiv \nu x. 0$
      - $P || \nu x. Q \equiv \nu x. (P || Q)$ if $x \not\in \text{FN}(P)$
    - $!P \equiv !P || P$
  
  - **Reaction law**
    - $x<y> || x(z). P || Q \Rightarrow [y/z]P || Q$
Properties

• If $P \Rightarrow^* Q$
  then $\bullet; \bullet; \text{"P" } \Rightarrow^* \Sigma; \Gamma; \Delta$
  where "Q" = $\exists\Sigma. !\Gamma \otimes \Delta$ mod $!A = !A \otimes A$

➢ Note: with $!P \rightarrow !P || P$ as a transition
  ▪ If $P \Rightarrow^* Q$
    then $\bullet; \bullet; \text{"P" } \Rightarrow^* \Sigma; \Gamma; \Delta$
    where "Q" = $\exists\Sigma. !\Gamma \otimes \Delta$
ω-Multisets vs. Process Algebra

- Simple encoding of asynchronous π-calculus into ω-multisets
  - Doesn’t show that π-calculus is logic
  - Uses only a fraction of ω-multiset syntax
  - Inverse encoding?
    - As hard as going from multiset rewriting to π-calculus

- Other languages
  - Join calculus
  - Strand spaces
  - To do: Synchronous π-calculus
MSR 3

• Instance of $\omega$-multisets for cryptographic protocol specification
  - Security-relevant signature
  - Typing infrastructure
  - Modules, equations, ...

• 3rd generation
  - MSR 1: First-order multiset rewriting with $\exists$
    - Undecidability of protocol analysis
  - MSR 2: MSR 1 + typing
    - Actual specification language
    - More theoretical results
    - Implementation underway
Example

Needham-Schroeder public-key protocol

1. \( A \rightarrow B: \{n_A, A\}_{kB} \)
2. \( B \rightarrow A: \{n_A, n_B\}_{kA} \)
3. \( A \rightarrow B: \{n_B\}_{kB} \)

- 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)

I. Cervesato: The Logical Meeting Point of MSR and PA
\[ \forall A: \text{princ.} \]
\[ \{ \exists L: \text{princ} \times \sum B: \text{princ.pubK B} \times \text{nonce} \rightarrow \text{mset.} \} \]

\[ \forall B: \text{princ.} \forall k_B: \text{pubK B.} \]
\[ \rightarrow \exists n_A: \text{nonce.} \]
\[ \text{net} (\{n_A, A\}_{k_B}), \ L (A, B, k_B, n_A) \]

\[ \forall B: \text{princ.} \forall k_B: \text{pubK B.} \]
\[ \forall k_A: \text{pubK A.} \forall k_A': \text{prvK k_A}. \]
\[ \forall n_A: \text{nonce.} \forall n_B: \text{nonce.} \]
\[ \text{net} (\{n_A, n_B\}_{k_A}), \ L (A, B, k_B, n_A) \]
\[ \rightarrow \text{net} (\{n_B\}_{k_B}) \]

**Interpretation of L**

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

---

**State-Based**

**MSR 2 spec.**

\[ A \rightarrow B: \{n_A, A\}_{k_B} \]
\[ B \rightarrow A: \{n_A, n_B\}_{k_A} \]
\[ A \rightarrow B: \{n_B\}_{k_B} \]
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}

I. Cervesato: The Logical Meeting Point of MSR and PA
**NSPK in Process Algebra**

\[ \forall A: \text{princ.} \]
\[ \forall B: \text{princ.} \forall k_B: \text{pubK } B. \]
\[ \forall k_A: \text{pubK } A. \forall k_A': \text{prvK } k_A. \forall n_B: \text{nonce.} \]

\[ \forall n_A: \text{nonce.} \]
\[ \text{net} (\{n_A, A\}_{KB}). \]
\[ \text{net} <\{n_A, n_B\}_{kA}>. \]
\[ \text{net} (\{n_B\}_{KB}). 0 \]

**Same structure!**
- Not a coincidence
- MSR 3 very close to Process Algebra
  - \(\omega\)-multiset encodings of \(\pi\)-calculus and Join Calculus

- MSR 3 is promising 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

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

- **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

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