The Logical Meeting Point of Multiset Rewriting and Process Algebra

Iliano Cervesato
Carnegie Mellon University - Qatar
iliano@cmu.edu

Mathematics Department, UPenn

December 11, 2006
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

\[ A, B ::= a \mid 1 \mid A \otimes B \mid A \rightarrow^0 B \mid !A \]
\[ \mid T \mid A \& B \mid \forall x. A \mid \exists x. A \]

• LV sequents

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

Unrestricted context

Linear context

Signature

Goal formula

- Constructor: “,”
- Empty: “•”

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

Left rules

\[
\frac{\Gamma; \Delta, A, B \rightarrow_{\Sigma} C}{\Gamma; \Delta, A \otimes B \rightarrow_{\Sigma} C}
\]

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

\[
\Sigma \vdash \top \quad \frac{\Gamma; \Delta, [\top/x]A \rightarrow_{\Sigma} C}{\Gamma; \Delta, \forall x. A \rightarrow_{\Sigma} C}
\]

\[
\frac{\Gamma; \Delta, A \rightarrow_{\Sigma, x} C}{\Gamma; \Delta, \exists x. A \rightarrow_{\Sigma} C}
\]

\[
\frac{\Gamma, A; \Delta \rightarrow_{\Sigma} C}{\Gamma; \Delta, !A \rightarrow_{\Sigma} C}
\]

Right rules

\[
\Gamma; \Delta \rightarrow_{\Sigma} C
\]

Structural rules

\[
\Gamma; A \rightarrow_{\Sigma} A
\]

\[
\Gamma, A; \Delta, A \rightarrow_{\Sigma} C
\]

\[
\Gamma, A; \Delta \rightarrow_{\Sigma} C
\]

Cut rules

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

\[
\frac{\Gamma; \bullet \rightarrow_{\Sigma} A \quad \Gamma, A; \Delta \rightarrow_{\Sigma} C}{\Gamma; \Delta \rightarrow_{\Sigma} C}
\]
 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

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

- Constructor: "\
- Empty: "¬"
Interpreting Unary Rules

\[
\begin{align*}
\Gamma; \Delta, A, B & \rightarrow^\Sigma C \\
\Gamma; \Delta, A \otimes B & \rightarrow^\Sigma C \\
\Sigma; \Gamma; (\Delta, [t/x]A) & \rightarrow \Sigma; \Gamma; (\Delta, [t/x]A) \\
\Gamma; \Delta, \forall x.A & \rightarrow^\Sigma C \\
\Sigma; \Gamma; (\Delta, \forall x. A) & \rightarrow \Sigma; \Gamma; (\Delta, [t/x]A) \\
\Gamma; \Delta, A & \rightarrow^\Sigma x C \\
\Gamma; \Delta, \exists x.A & \rightarrow^\Sigma C \\
\Gamma; A; \Delta & \rightarrow^\Sigma C \\
\Gamma; \Delta, !A & \rightarrow^\Sigma C \\
\end{align*}
\]
Binary Rules and Axiom

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

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

• Observation states
  \[ \Sigma ; \Delta \]
  - In \( \Delta \), we identify
    - \( , \) with \( \otimes \)
    - \( \cdot \) 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
  \[ \Sigma; \Gamma; \Delta \rightarrow^* \Sigma'; \Delta' \]
Induced Structural Equivalences

Monoidal laws

- $A \otimes B = B \otimes A$
- $A \otimes 1 = A$
- $(A \otimes B) \otimes C = A \otimes (B \otimes C)$

Mobility laws

- $\exists x. \exists y. \Delta = \exists y. \exists x. \Delta$
- $\exists x. \bullet = \bullet$
- $\exists x. (\Delta, \Delta') = \Delta, \exists x. \Delta'$
  if $x \not\in FV(\Delta)$

- Logical bi-equivalences
  - Require limited right rules
- Express structure of context / binders
- Expand rewrite opportunities
Interpreting Binary Rules

\[ \frac{\Gamma; A \longrightarrow_{\Sigma} A}{\Sigma; \Gamma; \Delta \rightarrow^{*} \Sigma; \Delta} \]

\[ \frac{\Sigma; \Gamma; \Delta \rightarrow^{*} \Sigma''; \Delta''}{\text{if } \Sigma; \Gamma; \Delta \rightarrow \Sigma'; \Gamma'; \Delta'} \]

and \[ \Sigma'; \Gamma'; \Delta' \rightarrow^{*} \Sigma''; \Delta'' \]

\[ \frac{\Gamma; \Delta' \longrightarrow_{\Sigma} A; \Gamma; \Delta, B \longrightarrow_{\Sigma} C}{\Gamma; \Delta, \Delta', A \longrightarrow_{oB} \longrightarrow_{\Sigma} C} \]

\[ \frac{\Sigma; \Gamma; (\Delta, \Delta', A \longrightarrow_{oB} B) \rightarrow \Sigma; \Gamma; (\Delta, B)}{\text{if } \Sigma; \Gamma; \Delta' \rightarrow^{*} \Sigma; A} \]

\[ \frac{\Gamma; \Delta' \longrightarrow_{\Sigma} A; \Gamma; \Delta, A \longrightarrow_{\Sigma} C}{\Gamma; \Delta, \Delta' \longrightarrow_{\Sigma} C} \]

\[ \frac{\Sigma; \Gamma; \Delta, \Delta' \rightarrow \Sigma; \Gamma; (A, \Delta)}{\text{if } \Sigma; \Gamma; \Delta' \rightarrow^{*} \Sigma; A} \]

...
Formal Correspondence

• Soundness

If \( \Sigma ; \Gamma ; \Delta \rightarrow^* \Sigma, \Sigma'; \Delta' \)
then \( \Gamma ; \Delta \rightarrow \exists \Sigma'. \bigotimes \Delta' \)

• Completeness?

\( \text{No! \; We have only crippled right rules} \)

\( ; ; a \rightarrow o \; b, \; b \rightarrow o \; c \) \( \quad \) \( ; a \rightarrow o \; c \)
System $\omega$

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

- 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 $\rightarrow^*$
Summary – Syntax

\[ A, B ::= a \quad \text{atomic object} \]
\[ \mid 1 \quad \text{empty} \]
\[ \mid A \otimes B \quad \text{formation} \]
\[ \mid A \rightarrow B \quad \text{rewrite} \]
\[ \mid T \quad \text{no-op} \]
\[ \mid A \& B \quad \text{choice} \]
\[ \mid \forall x. A \quad \text{instantiation} \]
\[ \mid \exists x. A \quad \text{generation} \]
\[ \mid ! A \quad \text{replication} \]
Summary – Semantics

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>$\Sigma ; \Gamma ; (\Delta, A, A \rightarrow B) \rightarrow \Sigma ; \Gamma ; (\Delta, B)$</td>
</tr>
<tr>
<td>&amp;</td>
<td>$\Sigma ; \Gamma ; (\Delta, A_1 &amp; A_2) \rightarrow \Sigma ; \Gamma ; (\Delta, A_i)$</td>
</tr>
<tr>
<td>∨</td>
<td>$\Sigma ; \Gamma ; (\Delta, \forall x. A) \rightarrow \Sigma ; \Gamma ; (\Delta, [t/x]A)$ if $\Sigma \vdash t$</td>
</tr>
<tr>
<td>!</td>
<td>$\Sigma ; \Gamma ; (\Delta, !A) \rightarrow \Sigma ; (\Gamma, A) ; \Delta$</td>
</tr>
<tr>
<td>1</td>
<td>$\Sigma ; \Gamma ; (\Delta, 1) \rightarrow \Sigma ; \Gamma ; \Delta$</td>
</tr>
<tr>
<td>⊗</td>
<td>$\Sigma ; \Gamma ; (\Delta, A \otimes B) \rightarrow \Sigma ; \Gamma ; (\Delta, A, B)$</td>
</tr>
<tr>
<td>∃</td>
<td>$\Sigma ; \Gamma ; (\Delta, \exists x. A) \rightarrow (\Sigma, x) ; \Gamma ; (\Delta, A)$</td>
</tr>
</tbody>
</table>
Discussion

- Other connectives?
  - ⊕, 0, ⊥
    - Odd rewrite properties
  - ?, (_)⊥
    - Not yet explored

- Other presentations?

- Other logics?

- Other forms of proof-as-computation?
  - Dual of logic programming
  - Similar to ACL [Kobayashi & Yonezawa, 93]

- Can logic benefit?
Type Theoretic Side

- Very close to **CLF**
  
  *Concurrent Logical Framework*
  
  - Linear type theory with
    - Dependent function types: \(\Pi\) (LF)
    - Asynchronous connectives: \(\rightarrow, \&, T\) (LLF)
    - Synchronous connectives: \(\otimes, 1, !, \exists\)
    - Monadic sandboxing
    - Concurrency equations
  
  - Faithful encoding of true concurrency
    - Petri nets, MSR 2 specs, \(\pi\)-calculus, concurrent ML

- Details of relation still unclear

---

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

- Multiset: set with repetitions allowed
  \[ a ::= • \mid a, a \]
  - 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 \( \Sigma \vdash t \)
- 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 ¬ο
  ▪ ¬ο never nested, always persistent

  ➢ \[ \Sigma; s \rightarrow_R \Sigma'; s' \]
  iff \[ \Sigma; \text{"R"}; \text{"s"} \rightarrow^* \Sigma'; \text{"s'"} \]

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

Another fundamental model of distributed computing

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

- **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 $x \notin FN(P)$
    - $!P \equiv !P || P$

  - **Reaction law**
    - $x<y> || x(z). P || Q \Rightarrow [y/z]P || Q$
\(\pi\)-calculus in \(\omega\)-Multisets

- \(0 \Leftrightarrow 1\)
- \(|| \Leftrightarrow \otimes||\)
- \(\nu \Leftrightarrow \exists\)
- \(! ! \Leftrightarrow ! !\)
- \(x(y). P \Leftrightarrow \forall y. ch(x,y) \rightarrow o \text{ "P"} \)
- \(x\langle y\rangle \Leftrightarrow ch(x,y)\)

**Reaction law**
- \(\Sigma; \Gamma; ch(x,y), \forall z. ch(x,z) \rightarrow o P, \Delta \rightarrow^2 \Sigma; \Gamma; [y/z]P, \Delta\)

**Structural equivalence**
- Monoidal congr. of \(||\) and \(0 \Leftrightarrow\) monoidal laws of \(\otimes\) and \(1\)
- Mobility congr. of \(\nu \Leftrightarrow\) mobility laws of \(\exists\)
- \(!P \equiv !P || P\)
  - Only \(\Rightarrow\) in \(\omega\)-multisets
  - Oversight in the \(\pi\)-calculus?

---

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

• If $P \Rightarrow^* Q$
  then $\cdot; \cdot; \text{"} P \text{"} \Rightarrow^* \Sigma; \Gamma; \Delta$

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 $\cdot; \cdot; \text{"} P \text{"} \Rightarrow^* \Sigma; \Gamma; \Delta$

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
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)
∀A: princ.
\{ ∃L: princ × ΣB: 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

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

Process-Based

∀A: princ.
∀B: princ. ∀k_B: pubK B.

• → ∃n_A: nonce.

net (\{n_A, A\}_kB),

(∀k_A: pubK A. ∀k_A': prvK k_A. ∀n_B: nonce.

net (\{n_A, n_B\}_KA) → net (\{n_B\}_kB))

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

A \rightarrow B: \{n_A, A\}_kB
B \rightarrow A: \{n_A, n_B\}_KA
A \rightarrow B: \{n_B\}_kB
**NSPK in Process Algebra**

\[ \forall A: \text{princ.} \]
\[ \forall B: \text{princ.} \forall k_B: \text{pubK } B. \]
\[ \forall k_A: \text{pubK } A. \forall k'_A: \text{prvK } k_A. \forall n_B: \text{nonce.} \]
\[ \forall n_A: \text{nonce.} \]
\[ \text{net } \{n_A, A\}_{k_B}. \]
\[ \text{net } \{n_A, n_B\}_k A. \]
\[ \text{net } \{n_B\}_{k_B}. \]

- **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, ...

- **Process-based languages**
  - Process Algebra
  - Strand spaces, spi-calculus, ...

  - Independent communicating threads
I. Cervesato: The Logical Meeting Point of MSR and PA

MSR 3 Bridges the Gap

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

State vs. process distance

MSR 3

Other distance

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

• $\omega$-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

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