Logical Foundations of Multiset Rewriting

Iliano Cervesato
iti@itd.nrl.navy.mil

ITT Industries, inc @ NRL Washington, DC

http://www.cs.stanford.edu/~iliano
Outline

• Motivations
• Propositional multiset rewriting
   Interpretation in linear logic
   Interpretation as linear logic
• Logical extension
   First-order multiset rewriting
   \( \omega \)-multisets
• Applications
   Specification of security protocols
   A bridge to process algebra
Motivations

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

• Shallow relations to logic
  - Simple encodings
  - No deep insight
This Work

• Show that multiset rewriting has deeper relations to logic
  ➢ Interpretation as logic, rather than
  ➢ Interpretation in logic

• Explain and rationalize extensions

• Better specification languages

• Bridge to process algebra
Multiset Rewriting

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

  ➢ Commutative monoid
    - "\" is operation
    - "\•" is identity
      ("\," is commutative, associative, with "\•" as unit)

- **Rewrite rule**:
  
  \[ a \rightarrow b \]

  ➢ Monoidal rewriting
Semantics of Multiset Rewriting

- **Base step:** \( s \rightarrow_R s' \)

- **Reachability**
  - \( s_0 \rightarrow^*_R s_n \)
  - Iteration of \( \rightarrow \)
  - R&T closure of \( \rightarrow \)

- **Infinity**
  - \( s_0 \rightarrow^*_R \)
  - Limit of \( _- \rightarrow^*_R _- \)
Linear Logic

Logic with formulas as resources

• Formulas
  \[ A ::= a \mid A \otimes A \mid 1 \mid A \multimap A \mid \ldots \]

• Judgment (DILL / LV sequent)
  \[ \Gamma; \Delta \rightarrow A \]

  Unrestricted context
  - subject to exchange, weakening and contraction
  - behaves like context in traditional logic

  Linear context
  - subject to exchange only
Some Rules

\[
\begin{align*}
\Gamma; \Delta, A, B & \rightarrow C \\
\Gamma; \Delta, A \otimes B & \rightarrow C \\
\Gamma; \Delta & \rightarrow C \\
\Gamma; \Delta, 1 & \rightarrow C \\
\Gamma; \Delta_1 & \rightarrow A \\
\Gamma; \Delta_2 & \rightarrow B \\
\Gamma; \Delta_1, \Delta_2 & \rightarrow A \otimes B \\
\Gamma; \Delta_1 & \rightarrow A \\
\Gamma; \Delta_2 & \rightarrow B \\
\Gamma; \Delta_1, \Delta_2, A & \rightarrow B \\
\Gamma; \Delta & \rightarrow A \\
\Gamma; A; \Delta, A & \rightarrow C \\
\Gamma; A; \Delta & \rightarrow C \\
\Gamma; A & \rightarrow A
\end{align*}
\]
LL Interpretation of MSR

• Several possibilities
  ➢ "Conjunctive" encoding

• Objective
  \[ R ; s_0 \rightarrow^* s_n \]
  \[ \Gamma ; \Delta \rightarrow A \]

➢ Reachability mapped to derivability
Encoding

- $R$
  - $\to \Rightarrow \text{False}$

- $s_0$
  - $\, \Rightarrow \,$
  - $\bullet \Rightarrow \bullet$
  - ... or like $s_n$

- $s_n$
  - $\, \Rightarrow \times$
  - $\bullet \Rightarrow 1$
Encoding

- **R**
  - $[a \rightarrow b] = [a] \circ [b]$

- **$s_0$**
  - $[[a]] = a$
  - $[[\cdot]] = \cdot$
  - $[[a, b]] = [[a]], [[b]]$ or $[a, b]$

Well defined because
- $(\Delta s, \cdot, \cdot)$ is a commutative monoid
- $(As, \otimes, 1)$ is a commutative monoid

- **$s_n$**
  - $[a] = a$
  - $[[\cdot]] = 1$
  - $[a, b] = [a] \otimes [b]$
Property

\[ s_0 \rightarrow_R^* s_n \text{  iff  } [R]; [[s_0]] \rightarrow [s_n] \]

- For appropriate inverse encodings

\[ \Gamma; A \rightarrow B \text{  iff  } [A] \rightarrow^{*_{[\Gamma]}} [B] \]

Encoding of MSR in LL
End of the Story?

- Yes, NO!

- From interpretation of MSR in logic to interpretation of MSR as logic

- Multiset rewriting semantics = left sequent rules

- First, a few rough edges to smooth
Context vs. Formulas (1)

- Either go against tradition of logic
  - *(As, \(\otimes\), 1)* is a congruence w.r.t. derivability

  ➢ Identify contexts and formulas
  - Whenever formula is expected
    - Turn \(\cdot\) into \(\otimes\)
    - Turn \(\cdot\) into 1
  - Consistent with categorical semantics of logic
  - Has to be done with extreme care
Context vs. Formulas (2)

• ... or go against tradition of rewriting
  ➢ Distinguish states and multisets
    ▪ state constructors: , and •
    ▪ mset constructors: ⊗ and 1
  ➢ Additional transition rules
    ▪ $s, a \otimes b \rightarrow_R s, a, b$
    ▪ $s, 1 \rightarrow_R s$

• This research is compatible with both
  ➢ We will lean towards (2)
Rewriting View of Derivations

- **Step up:**
  - Left rules
- **Step across:**
  - Axiom
- **Right rules not used**

\[
\begin{align*}
\Gamma''; \Delta'' &\rightarrow C \\
\Gamma'; \Delta' &\rightarrow C \\
\Gamma; \Delta &\rightarrow C
\end{align*}
\]
Rewriting Semantics as Left Rules

\[ s \rightarrow^*_R s \]

\[ s, a \otimes b \rightarrow_R s, a, b \]

\[ s, 1 \rightarrow_R s \]

\[ s, a \rightarrow_R (a \rightarrow b) s, b \]

\[ \Gamma; A \rightarrow A \]

\[ \Gamma; \Delta, A, B \rightarrow C \]

\[ \Gamma; \Delta, A \otimes B \rightarrow C \]

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

\[ \Gamma; \Delta, 1 \rightarrow C \]

\[ \Gamma, A \rightarrow^*_R B; \Delta, B \rightarrow C \]

\[ \Gamma, A \rightarrow^*_R B; A, \Delta \rightarrow C \]

Not quite, but not too far off

- Admissible rule

\[ \Gamma, A; \Delta, A \rightarrow C \]

\[ \Gamma, A; \Delta \rightarrow C \]

\[ \Gamma; \Delta_1 \rightarrow A \]

\[ \Gamma; \Delta_2, B \rightarrow C \]

\[ \Gamma; \Delta_1, \Delta_2, A \rightarrow B \rightarrow C \]
Questions

- **Can we make the correspondence precise?**
  - Yes

- **Does it extend to other connectives?**
  - Yes ... to a large extent

- **What are the implications?**
  - Logical explanation of multiset rewriting
    - Not just interpretation
    - Now MSR is logic
  - Guideline to design rewrite systems
    - Can we do this with other logics?
  - Derivations do not need to be finite
    - Goal is important only for reachability
First Proof of Concept

- **First-Order Multiset Rewriting (MSR 1.0)**
  - Multiset elements are F0 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 (\(\Rightarrow^*\))
    
    \[ \Sigma; \; a(t), \; s \rightarrow_{R, \; (a(x) \rightarrow \exists y. \; b(x,y))} \Sigma, y; \; b(t,y), \; s \]
    
    if \( \Sigma \models t \)
  - Encoding is simple extension of prop. case
Semantics from Left Rules

- Updated judgment forms
  - $\Sigma; s \rightarrow_{R} \Sigma; s$
  - $\Gamma; \Delta \rightarrow_{\Sigma} C$

- Semantics ($\rightarrow^{**}$)

| $\Sigma; s, \forall x.a \rightarrow_{R} \Sigma; s, [t/x]a$ | $\Gamma; \Delta, [t/x]A \rightarrow_{\Sigma} C$ if $\Sigma |- t$ |
|----------------------------------------------------------|---------------------------------------------------------------|
| $\Sigma; s, \exists x.a \rightarrow_{R} \Sigma,x; s, a$ | $\Gamma; \Delta, \forall x.A \rightarrow_{\Sigma} C$ |
|                                                         | $\Gamma; \Delta, \exists x.A \rightarrow_{\Sigma} C$ |
Comparing Semantics

Lemma

- If $a \xrightarrow{R}^* (b)$, then $a \xrightarrow{R}^{**} (b)$

- And vice versa
  
  - Careful with non-observable steps
Second Proof of Concept

- **Minimal $\omega$-multiset rewriting**
  - **Language**
    \[
    \omega ::= a \mid \bullet \mid \omega, \omega \mid \omega \rightarrow \omega
    \]
    - No distinction between atoms and formulas
  - **Semantics (v.1)**
    - $s, (a \rightarrow b), a \rightarrow s, b$
  - **Check against left rule for $\longrightarrow_o$**
    \[
    \Delta_1 \rightarrow A \quad \Delta_2, B \rightarrow C
    \]
    \[
    \Delta_1, \Delta_2, A \longrightarrow_o B \rightarrow C
    \]
  - **Semantics (v.2)**
    - $s_1, s_2, (a \rightarrow b) \rightarrow s_2, b$ if $s_1 \rightarrow^* a$
    - Step depends on reachability!
Comparing Semantics

• Lemma

\[ a \rightarrow^*_{v.1} (b) \iff a \rightarrow^*_{v.2} (b) \]

(⇒) Trivial by reflexivity
(⇐) Recursively turn every step
  - \( s_1, s_2, (a \rightarrow b) \rightarrow_{v.2} s_2, b \) if \( s_1 \rightarrow^*_{v.2} a \)
  into
  - \( s_1, s_2, (a \rightarrow b) \rightarrow^*_{v.1} a, s_2, (a \rightarrow b) \rightarrow_{v.1} s_2, b \)

• However

- Do all extensions support transformation?
  - Use \( v.1 \) when adequate, \( v.2 \) other times
- Seems to be an instance of cut elimination
  - (see later)
Adding Persistent Multisets

- **Language**
  \[ \omega ::= a \mid \bullet \mid \omega, \omega \mid \omega \rightarrow \omega \mid \forall x. \omega \mid \exists x. \omega \mid ! \omega \]

- **Judgment**
  \[ \Sigma; p; s \rightarrow \Sigma; p; s \]

- **Semantics from left rules**

<table>
<thead>
<tr>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Sigma; p; s, !a \rightarrow \Sigma; p, a; s )</td>
<td>( \Gamma, A; \Delta \rightarrow_{\Sigma} C )</td>
</tr>
<tr>
<td>( \Gamma; \Delta, !A \rightarrow_{\Sigma} C )</td>
<td></td>
</tr>
<tr>
<td>( \Sigma; p, a; s \rightarrow \Sigma; p, a; s, a )</td>
<td>( \Gamma, A; \Delta, A \rightarrow_{\Sigma} C )</td>
</tr>
<tr>
<td>( \Gamma, A; \Delta \rightarrow_{\Sigma} C )</td>
<td></td>
</tr>
</tbody>
</table>
A Word of Caution

\[ !(a \otimes b) \neq !a \otimes !b \]

- \( \otimes \) corresponds to "," in \( \Delta \), but not in \( \Gamma \)
  - Distinguish \( \otimes \) and "," in \( \omega \text{MSR} \)
  - Consider only sublanguages
  - Use different symbol ",," in \( p \)
    - \( p \) is multiset of multisets, not multiset
Additive Conjunction and Unit

- **Language**
  \[ \omega ::= \ldots | \omega \& \omega | T \]

- **Semantics from left rules**

<table>
<thead>
<tr>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \Sigma ; p : s, a_1 &amp; a_2 \rightarrow \Sigma ; p : s, a_i ]</td>
<td>[ \Gamma; \Delta, A_i \rightarrow_\Sigma C ]</td>
</tr>
<tr>
<td>Non-deterministic choice</td>
<td>[ \Gamma; \Delta, A_1 &amp; A_2 \rightarrow_\Sigma C ]</td>
</tr>
<tr>
<td>• Usually written +</td>
<td>(no left rule)</td>
</tr>
</tbody>
</table>

(no T-transition)

Absence of any choice
Additive Disjunction and Unit

• **Language**
  \[ \omega ::= \ldots \mid \omega \oplus \omega \mid 0 \]

• **Semantics from left rules**

\[ \Sigma ; p ; s, 0 \Rightarrow^* s_n \]

- Inconsistency?
- Forced reachability?

\[ \Gamma ; \Delta , 0 \rightarrow^*_\Sigma C \]
The case of $\oplus$

The 2 computations shall be synchronized
- If one “ends”, the other “ends” in the same way
  - Breakpoint, or final state
- If one diverges, the other shall diverge

Flavor of
- Confluence
- Bisimulation?
Multiplicative Disjunction and Unit

- **Language:**
  \[ \omega ::= \ldots \mid \omega \mathcal{O} \omega \mid \bot \]

- **Semantics from left rules**

\[
\Sigma ; p ; \bot \Rightarrow^* \bullet
\]

- Abort?
- Deadlock?
The Case of $\varnothing$

\[
\begin{align*}
\Gamma; \Delta_1, A & \rightarrow_\Sigma \Psi_1 \quad \Gamma; \Delta_2, B & \rightarrow_\Sigma \Psi_2 \\
\Gamma; \Delta_1, \Delta_2, A \varnothing B & \rightarrow_\Sigma \Psi_1, \Psi_2 \\
\end{align*}
\]

\[
\Sigma; p; s_1, a, s_2, b \rightarrow
\begin{cases}
\Sigma; p; s_1, a \\
\Sigma; p; s_2, b
\end{cases}
\]

- Start of completely independent computations involving $a$ and $b$
The Axiom Rule

$$\Sigma; p; a \rightarrow^* a$$

- Makes a reachability statement
- Turns $\rightarrow$ into $\rightarrow^* a$
The Cut Rules

\[ \Gamma; \Delta_1 \rightarrow_{\Sigma} A \quad \Gamma; \Delta_2, A \rightarrow_{\Sigma} C \]
\[ \Gamma; \Delta_1, \Delta_2 \rightarrow_{\Sigma} C \]
\[ \Gamma; \cdot \rightarrow_{\Sigma} A \quad \Gamma, A; \Delta \rightarrow_{\Sigma} C \]
\[ \Gamma; \Delta \rightarrow_{\Sigma} C \]

\begin{align*}
\Sigma; p; s_1, s_2 & \rightarrow \Sigma; p; a, s_2 & \text{if } \Sigma; p; s_2 & \rightarrow^* a \\
\Sigma; p; s & \rightarrow \Sigma; p, a; s & \text{if } \Sigma; p; \cdot & \rightarrow^* a
\end{align*}

- Compositionality laws

- Does cut elimination hold?

- Note
  - Not as deep as in Logic
    - No right rules
Summary: $\omega$-Multisets

$$\omega ::= a \quad \text{atomic object}$$

$$\bullet \quad \text{empty mset}$$

$$\omega, \omega \quad \text{mset formation}$$

$$\omega \rightarrow \omega \quad \text{mset rewrite}$$

$$T \quad \text{no-op}$$

$$\omega + \omega \quad \text{choice}$$

$$! \omega \quad \text{replication}$$

$$\forall x. \omega \quad \text{instantiation}$$

$$\exists x. \omega \quad \text{generation}$$

$$0 \quad \omega \oplus \omega \quad \bot \quad \omega \not\in \omega$$

$$? \omega \quad \omega^\perp$$

???
### Summary: ω-Multisets Semantics

- \( \Sigma ; p ; (s, 1) \rightarrow \Sigma ; p ; s \)
- \( \Sigma ; p ; (s, a \otimes b) \rightarrow \Sigma ; p ; (s, a, b) \)
- \( \Sigma ; p ; (s, a, a \rightarrow b) \rightarrow \Sigma ; p ; (s, b) \)
- \( \Theta \) (no rule)

& \( \Sigma ; p ; (s, a_1 \& a_2) \rightarrow \Sigma ; p ; (s, a_i) \)

! \( \Sigma ; p ; (s, !a) \rightarrow \Sigma ; (p, a) ; s \)

∀ \( \Sigma ; p ; (s, \forall x. a) \rightarrow \Sigma ; p ; (s, [t/x]a) \)

∃ \( \Sigma ; p ; (s, \exists x. a) \rightarrow (\Sigma, x) ; p ; (s, a) \)

\( \Sigma ; (p, a) ; s \rightarrow \Sigma ; (p, a) ; (s, a) \)
Applications to Security

- MSR: family of security protocol specification languages
  - MSR 1: first-order multiset rewriting
  - MSR 2: MSR 1 + dependent types
  - MSR 3: $\omega$-multiset (+ dependent types)

- Unified logical view
  - Better understanding of where we are
  - Hint about where to go next
NSPK in MSR 2.0

∀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})}
NSPK in MSR 3

∀A: princ.
∀B: princ. ∀KB: pubK B.

→ ∃NA: nonce.

net (\{NA, A\}_KB),
(∀KA: pubK A. ∀KA': prvK K_A. ∀NB: nonce.
net (\{NA, NB\}_KA)
→ net (\{NB\}_KB))
• **Succinct representations**
  - Simpler specifications
  - Economy of reasoning

• **Logical foundations**

• **Bridge between**
  - State-based representation
  - Process-based representations
    - Logical foundation of process algebra?
MSR vs. Process Algebra

MSR
- NRL Prot. Analyzer, CAPSL/CIL, Paulson’s approach, ...

and Process Algebra
- Strand spaces, spi-calculus, other process-based lang.

operate in very different ways:

• State transitions

• Contact evolution
Representing Protocols

- **MSR 2**

  \[
  \begin{align*}
  n & \rightarrow a_1, n' \\
  n'', a_1 & \rightarrow a_2, n'' \\
  \ldots
  \end{align*}
  \]

  - \( a_i \) pass control/data to the next rule

- **PA**

  \[ n.n'.n''.n'''.\ldots.0 \]

  - Control is implicit

---

**NS: MSR rules for Alice**

\[
\begin{align*}
\pi_{A0}(A) & \rightarrow A_0(A), \pi_{A0}(A) \\
A_0(A), \pi_{A1}(B) & \rightarrow \exists N_A. A_0(A, B, N_A), N((N_A, A)_{KB}), \pi_{A1}(B) \\
A_0(A, B, N_A), N((N_A, N_B)_{KA}) & \rightarrow A_1(A, B, N_A, N_B) \\
A_0(A, B, N_A, N_B) & \rightarrow A_2(A, B, N_A, N_B, N((N_B)_{KB})) \\
\text{where} & \\
\pi_{A0}(A) & = Pr(A), \text{PrvK}(A, K_{A^{-1}}) \\
\pi_{A1}(B) & = Pr(B), \text{PubK}(B, K_B)
\end{align*}
\]

**NS: Parametric Strand for Alice**

\[
\begin{align*}
\text{Alice} (A, B, N_A, N_B) : & \rightarrow \{N_A, A\}_{KB} \\
N_A \text{ Fresh, } \pi_A (A, B) & \rightarrow \downarrow \\
& \text{where} \\
\pi(A, B) & = Pr(A), \text{PrvK}(A, K_{A^{-1}}), Pr(B), \text{PubK}(B, K_B) \\
& \rightarrow \{N_A, N_B\}_{KA} \rightarrow \{N_B\}_{KB}
\end{align*}
\]

Relating Strands and Multiset Rewriting for Security Protocols
Representing Protocols

- **MSR 2**
  \[
  \begin{align*}
  n \rightarrow & \ a_1, n' \\
  n'' \rightarrow & \ a_1 \rightarrow a_2, n'''
  \end{align*}
  \]

  - \(a_i\) pass control/data to the next rule

- **MSR 3**
  \[
  n \rightarrow n', (n'' \rightarrow n''' , (...))
  \]

  - Control is implicit

**NS: MSR rules for Alice**

- \(\pi_{A_0}(A) \rightarrow A_0(A), \pi_{A_0}(A)\)
- \(A_0(A), \pi_{A_1}(B) \rightarrow \exists N_A. A_1(A,B,N_A), N((N_A,A)_{KB}), \pi_{A_1}(B)\)
- \(A_0(A,B,N_A), N((N_A,N_B)_{KB}) \rightarrow A_2(A,B,N_A,N_B)\)
- \(A_0(A,B,N_A,N_B) \rightarrow A_3(A,B,N_A,N_B), N((N_B)_{KB})\)

  where \(\pi_{A_0}(A) = Pr(A), Prv_K(A, K_A^{-1})\)
  \(\pi_{A_1}(B) = Pr(B), Pub_K(B, K_B)\)

**NS: Parametric Strand for Alice**

- Alice \((A,B,N_A,N_B) : N_A\text{ Fresh}, \pi_A(A,B)\)
  \[
  \begin{align*}
  (N_A, A)_{KB} \longrightarrow \\
  (N_A, N_B)_{KA} \leftarrow
  \end{align*}
  \]

  where
  \(\pi(A,B) = Pr(A), Prv_K(A, K_A^{-1}), Pr(B), Pub_K(B, K_B)\)
ω-Multisets and Process Algebra

- **Similarities**
  - ω-Multisets behave like very general process algebra
    - π-calculus
    - Join calculus

- **Differences**
  - PA’s structural equivalences

- **Towards a logical foundation of Process Algebra?**
Encoding Distributed Algorithms

State vs. process distance

Other distance

State ↔ Process translation done once and forall

MSR 3
Conclusions

- Interpretation of multiset rewriting guided by left rules of linear logic
- Definition of $\omega$-multisets
- Hint at application in security protocol specification
  - MSR 3.0
- Possible relationship with process algebras