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

Università di Venezia

March 8, 2004
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
  \[ A, B ::= a \mid 1 \mid A \otimes B \mid A \rightarrow B \mid ! A \mid T \mid A \& B \mid \forall x. A \mid \exists x. A \]

- LV sequents
  \[ \Gamma ; \Delta \rightarrow_{\Sigma} C \]

- Unrestricted context
- Linear context
- Signature
- Goal formula

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

Left rules

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

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

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

\[
\Sigma |- t \quad \Gamma; \Delta, [t/x]A \rightarrow_{\Sigma} C
\]

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

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

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

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

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

Structural rules

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

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

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

Cut rules

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

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

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

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

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
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 \( \mathcal{C} \)
    - Does not change

- **Transitions**
  - \( \Sigma; \Gamma; \Delta \rightarrow \Sigma'; \Gamma'; \Delta' \)
  - \( \rightarrow^* \) for reflexive and transitive closure

- Constructor: “,”
- Empty: “.”

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

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

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

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

...
Binary Rules and Axiom

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

\[ \Gamma; A \rightarrow_{\Sigma} A \]
\[ \Gamma; \Delta' \rightarrow_{\Sigma} A \]
\[ \Gamma; \Delta, B \rightarrow_{\Sigma} C \]
\[ \Gamma; \Delta, \Delta', A \rightarrow_{oB} \rightarrow_{\Sigma} C \]
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**
  \[ \Sigma; \Gamma; \Delta \Rightarrow^* \Sigma'; \Delta' \]
## 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 \( x \notin \text{FV}(\Delta) \)

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

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

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

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

if \( \Sigma; \Gamma; \Delta \rightarrow \Sigma' \); \( \Gamma'; \Delta' \)

and \( \Sigma'; \Gamma'; \Delta' \rightarrow^{*} \Sigma''; \Delta'' \)

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

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

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

\[ \Sigma; \Gamma; (\Delta, \Delta', A \rightarrow B) \rightarrow \Sigma; \Gamma; (\Delta, B) \]

if \( \Sigma; \Gamma; \Delta' \rightarrow^{*} \Sigma; \Delta \)

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

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

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

\[ \Sigma; \Gamma; (\Delta, \Delta', \Delta' \rightarrow \Sigma; \Gamma; (A, \Delta) \]

if \( \Sigma; \Gamma; \Delta' \rightarrow^{*} \Sigma; \Delta \)

...
Formal Correspondence

• Soundness

If $\Sigma ; \Gamma ; \Delta \rightarrow^* \Sigma, \Sigma'; \Delta'$ then $\Gamma ; \Delta \rightarrow_\Sigma \exists \Sigma'. \otimes \Delta'$

• Completeness?

➢ No! We have only crippled right rules

- $\bullet ; \bullet ; a \rightarrow o b, b \rightarrow o c \rightarrow^{+*} \bullet ; a \rightarrow o c$
**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 $\rightarrow^*$
Discussion

- **Other connectives?**
  - $\oplus$, 0, $\emptyset$, $\bot$
    - Odd rewrite properties
  - $\otimes$, $(\_\_\_)$
    - 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: \(--o, &, T\) (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
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
    - Competitor: 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
  \[ \forall x_1...x_n. \quad a(x) \rightarrow \exists y_1...y_k. \quad b(x,y) \]
- Semantics

\[ \Sigma ; a(t), s \rightarrow_R (a(x) \rightarrow \exists y. \quad b(x,y)) \quad \Sigma, y ; b(t,y), s \quad \text{if } \Sigma \vdash t \]

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

- **MSR 1 is an instance of ω-multisets**
  - Uses only $\otimes$, $1$, $\forall$, $\exists$, and $\lnot\rho$
  - $\lnot\rho$ never nested, always persistent
  
  $$
  \Sigma; \frac{s}{R} \Sigma'; \frac{s'}{*} 
  $$
  
  iff
  $$
  \Sigma; "R"; "s" \rightarrow^* \Sigma'; "s'" 
  $$

- **Interpretation of MSR as linear logic**
  - Logical explanation of multiset rewriting
    - MSR is logic
    - Guideline to design rewrite systems
\(\omega\)-Rewriting

\[
A, B ::= a \quad \text{atomic object}
\]

\[
| 1 \quad \text{empty}
\]

\[
| A \otimes B \quad \text{formation}
\]

\[
| A \longrightarrow B \quad \text{rewrite}
\]

\[
| T \quad \text{no-op}
\]

\[
| A \& B \quad \text{choice}
\]

\[
| \forall x. A \quad \text{instantiation}
\]

\[
| \exists x. A \quad \text{generation}
\]

\[
| ! A \quad \text{replication}
\]
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 $x \notin FN(P)$
    - $\!P \equiv \!P || P$
  
  - **Reaction law**
    - $x<y> || x(z). P || Q \Rightarrow [y/z]P || Q$
\( \pi \)-calculus in \( \omega \)-Multisets

- 0 \( \iff \) 1
- \( || \) \( \iff \) \( \otimes \)
- \( \nu \) \( \iff \) \( \exists \)
- \( ! \) \( \iff \) !
- x(y). P \( \iff \) \( \forall y. ch(x,y) \rightarrow \sigma \) "P"
- x<\( y > \) \( \iff \) ch(x,y)

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

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

- If \( P \Rightarrow^* Q \)
  
  then \( \bullet; \bullet; \, \text{“} P \rightarrow^* \Sigma; \Gamma; \Delta \)
  
  where \( \text{“} Q \text{”} = \exists \Sigma. !\Gamma \otimes \Delta \quad \text{mod} \quad !A = !A \otimes A \)

- Note: with \( !P \rightarrow !P \parallel P \) as a transition
  
  - If \( P \Rightarrow^* Q \)
    
    then \( \bullet; \bullet; \, \text{“} P \rightarrow^* \Sigma; \Gamma; \Delta \)
    
    where \( \text{“} Q \text{”} = \exists \Sigma. !\Gamma \otimes \Delta \)
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
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
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
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.
{ ∀B: princ. ∃L: 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. ∀kB: pubK B.

• → ∃nA: nonce.

net ({nA, A}kB),

(∀kA: pubK A. ∀kA': prvK kA. ∀nB: nonce.

net ({nA, nB}kA) → net ({nB}kB))

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

A → B: {nA, A}kB
B → A: {nA, nB}kA
A → B: {nB}kB
NSPK in Process Algebra

∀A: princ.
∀B: princ. ∀kB: pubK B.
∀kA: pubK A. ∀kA': prvK kA. ∀nB: nonce.

∀nA: nonce.

\[ \text{net}\left(\{n_A, A\}_{kB}\right). \]
\[ \text{net}\left(\{n_A, n_B\}_{kA}\right). \]
\[ \text{net}\left(\{n_B\}_{kB}\right). 0 \]

Same structure!

- Not a coincidence
- MSR 3 very close to Process Algebra
  - ω-multiset encodings of π-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 vs. process 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