The Logical Meeting Point of Multiset Rewriting and Process Algebra

Iliano Cervesato

Carnegie Mellon University - Qatar

iliano@cmu.edu

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
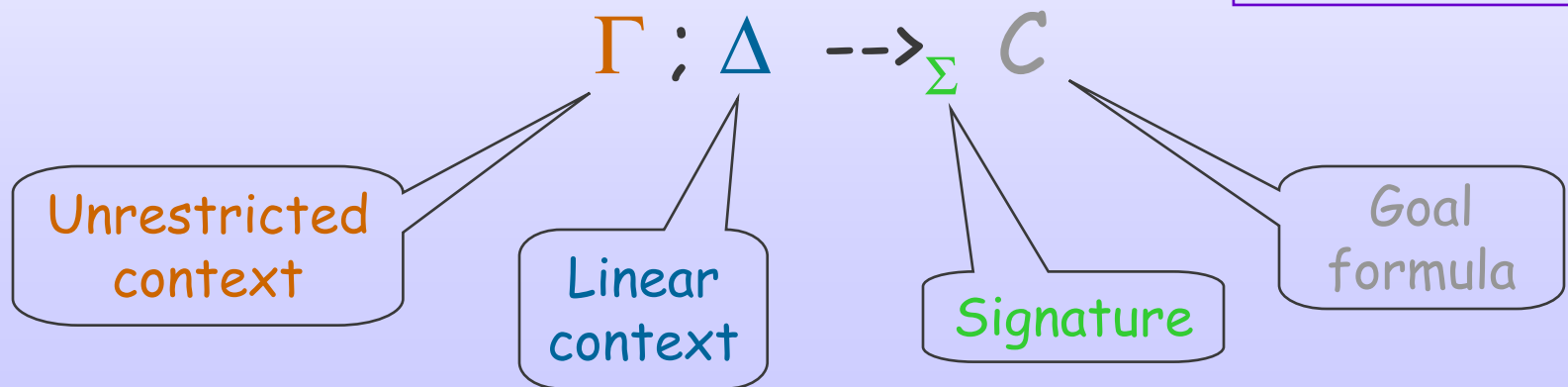 - ➤ Effective protocol specification language

Linear Logic

- Formulas

$$A, B ::= a \mid 1 \mid A \otimes B \mid A \multimap B \mid !A$$
$$\mid \top \mid A \& B \mid \forall x. A \mid \exists x. A$$

- LV sequents

> Constructor: ","
> Empty: "•"

$$\Gamma \,;\, \Delta \; \texttt{-->}_\Sigma \; C$$

Unrestricted context

Linear context

Signature

Goal formula

Some LV Rules

Left rules

$$\frac{\Gamma; \Delta, A, B \dashrightarrow_\Sigma C}{\Gamma; \Delta, A \otimes B \dashrightarrow_\Sigma C}$$

$$\frac{\Gamma; \Delta' \dashrightarrow_\Sigma A \quad \Gamma; \Delta, B \dashrightarrow_\Sigma C}{\Gamma; \Delta, \Delta', A \multimap B \dashrightarrow_\Sigma C}$$

$$\frac{\Sigma \vdash t \quad \Gamma; \Delta, [t/x]A \dashrightarrow_\Sigma C}{\Gamma; \Delta, \forall x.A \dashrightarrow_\Sigma C}$$

$$\frac{\Gamma; \Delta, A \dashrightarrow_{\Sigma,x} C}{\Gamma; \Delta, \exists x.A \dashrightarrow_\Sigma C}$$

$$\frac{\Gamma, A; \Delta \dashrightarrow_\Sigma C}{\Gamma; \Delta, !A \dashrightarrow_\Sigma C}$$

...

Structural rules

$$\frac{}{\Gamma; A \dashrightarrow_\Sigma A}$$

$$\frac{\Gamma, A; \Delta, A \dashrightarrow_\Sigma C}{\Gamma, A; \Delta \dashrightarrow_\Sigma C}$$

Cut rules

$$\frac{\Gamma; \Delta' \dashrightarrow_\Sigma A \quad \Gamma; \Delta, A \dashrightarrow_\Sigma C}{\Gamma; \Delta, \Delta' \dashrightarrow_\Sigma C}$$

$$\frac{\Gamma; \bullet \dashrightarrow_\Sigma A \quad \Gamma, A; \Delta \dashrightarrow_\Sigma C}{\Gamma; \Delta \dashrightarrow_\Sigma C}$$

Right rules

...

- logic
- system ω
- rewriting
- processes
- security
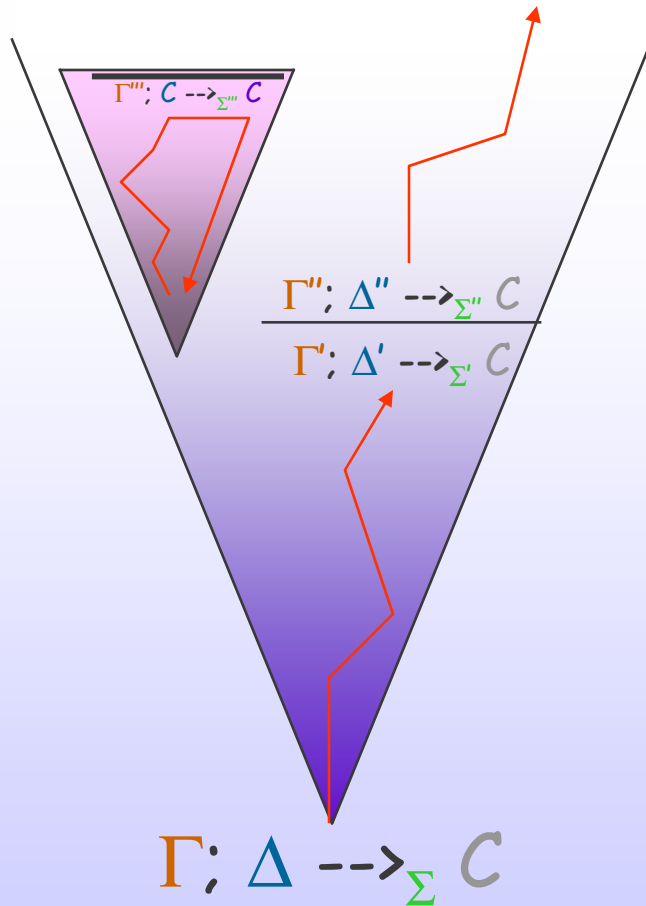
Logical Derivations

$$\Gamma'''; C \dashrightarrow_{\Sigma'''} C$$

$$\frac{\Gamma''; \Delta'' \dashrightarrow_{\Sigma''} C}{\Gamma'; \Delta' \dashrightarrow_{\Sigma'} C}$$

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

- logic
- system ω
- rewriting
- processes
- security
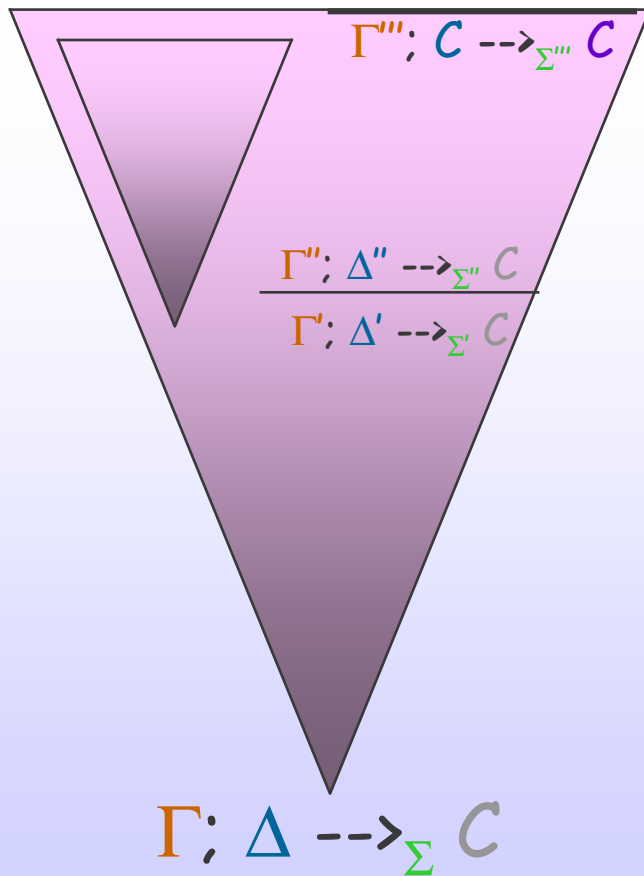
- **Proof of C from Δ and Γ**
  - ➤ Emphasis on C
    - C is input
- **Finite**
  - ➤ Closed

- **Rules shown**
  - ➤ Major premise
    - Preserves C
  - ➤ Minor premise
    - Starts subderivation

A Rewriting Re-Interpretation

- **Transition**
  - From conclusion
  - To major premise
  - Emphasis on Γ, Δ and Σ
  - C is output, at best
    - Does not change
- **Possibly infinite**
  - Open

- **Minor premise**
  - Auxiliary rewrite chain
    - Finite
  - Topped with axiom

- logic
- **system ω**
- rewriting
- processes
- security

State and Transitions

- **States**

$$\Sigma \quad ; \quad \Gamma \quad ; \quad \Delta$$

  - Σ is a list
  - Γ and Δ are commutative monoids
  - No C
    - Does not change

> - Constructor: ","
> - Empty: "●"

- **Transitions**

$$\Sigma; \Gamma; \Delta \;\rightarrow\; \Sigma'; \Gamma'; \Delta'$$

  - $\rightarrow^*$ for reflexive and transitive closure

- logic
- system ω
- rewriting
- processes
- security

Interpreting Unary Rules

$$\frac{\Gamma; \Delta, A, B \dashrightarrow_\Sigma C}{\Gamma; \Delta, A \otimes B \dashrightarrow_\Sigma C}$$

$$\Sigma; \Gamma; (\Delta, A \otimes B) \rightarrow \Sigma; \Gamma; (\Delta, A, B)$$

$$\frac{\Sigma \vdash t \quad \Gamma; \Delta, [t/x]A \dashrightarrow_\Sigma C}{\Gamma; \Delta, \forall x.A \dashrightarrow_\Sigma C}$$

$$\Sigma; \Gamma; (\Delta, \forall x. A) \rightarrow \Sigma; \Gamma; (\Delta, [t/x]A)$$
$$\text{if } \Sigma \vdash t$$

$$\frac{\Gamma; \Delta, A \dashrightarrow_{\Sigma,x} C}{\Gamma; \Delta, \exists x.A \dashrightarrow_\Sigma C}$$

$$\Sigma; \Gamma; (\Delta, \exists x. A) \rightarrow (\Sigma, x); \Gamma; (\Delta, A)$$

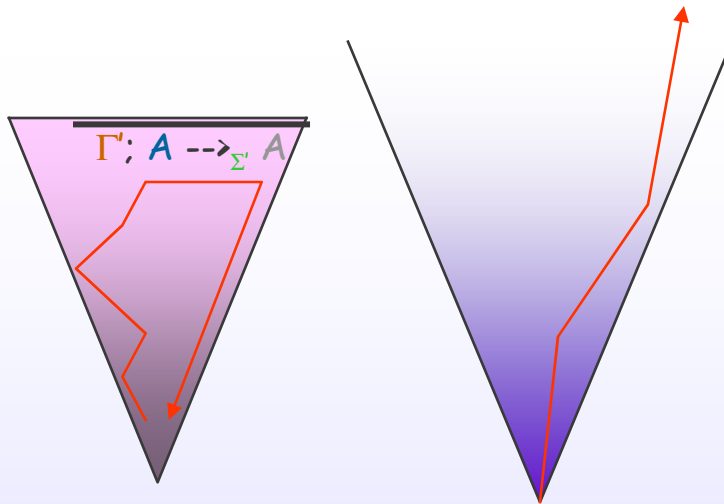$$\frac{\Gamma, A; \Delta \dashrightarrow_\Sigma C}{\Gamma; \Delta, !A \dashrightarrow_\Sigma C}$$

$$\Sigma; \Gamma; (\Delta, !A) \rightarrow \Sigma; (\Gamma, A); \Delta$$

...   ...

- logic
- system ω
- rewriting
- processes
- security

Binary Rules and Axiom

$$\Gamma';\ A \dashrightarrow_{\Sigma'} A$$

$$\dfrac{\Gamma;\ \Delta' \dashrightarrow_\Sigma A \quad \Gamma;\ \Delta,\ B \dashrightarrow_\Sigma C}{\Gamma;\ \Delta,\ \Delta',\ A \!\!-\!\!\circ B \dashrightarrow_\Sigma C}$$

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

Observations

$$\Gamma, \Gamma'; \ A' \ \text{-->}_{\Sigma, \Sigma'} \ A'$$

$$\Gamma; \ \Delta \ \text{-->}_\Sigma \ \underbrace{\exists \Sigma'. \ A'}_{A}$$

- **Observation states**

$$\Sigma \ ; \ \Delta$$

  - ➤ In Δ, we identify
    - , with $\otimes$
    - • with **1**

    Categorical semantics

    $$\Delta = \otimes \Delta$$

  - ➤ Identified with $\exists x_1. \ ... \ \exists x_n. \ \Delta$
    - For $\Sigma = x_1, \ ..., \ x_n$

    De Bruijn's telescopes

    $$\Sigma; \ \Delta = \exists \Sigma. \ \otimes \Delta$$
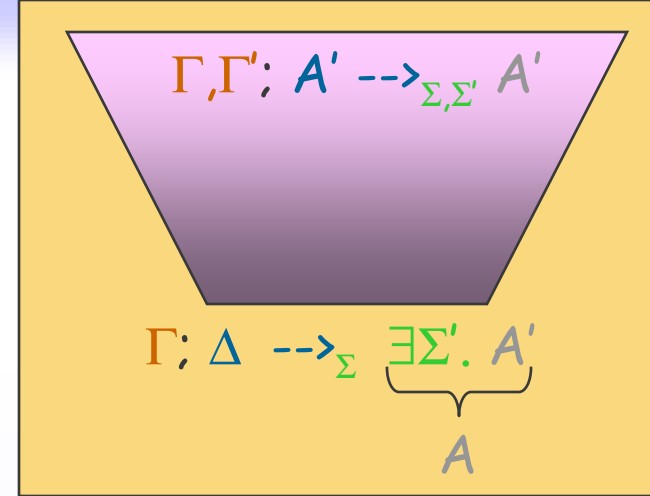
- **Observation transitions**

$$\Sigma; \ \Gamma; \ \Delta \ \blacktriangleright^* \ \Sigma'; \ \Delta'$$

- logic
- system ω
- rewriting
- processes
- security

Induced Structural Equivalences

Monoidal laws

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

Mobility laws

- $\exists x.\ \exists y.\ \Delta \quad = \exists y.\ \exists x.\ \Delta$
- $\exists x.\ \bullet \qquad\quad = \bullet$
- $\exists x.\ (\Delta, \Delta') = \Delta,\ \exists x.\ \Delta'$

  if $x \notin FV(\Delta)$

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

Interpreting Binary Rules

$$\overline{\Gamma;\ A \dashrightarrow_\Sigma A}$$

$$\Sigma;\ \Gamma;\ \Delta \blacktriangleright^* \Sigma;\ \Delta$$

$$\Sigma;\ \Gamma;\ \Delta \blacktriangleright^* \Sigma'';\ \Delta''$$
$$\text{if} \quad \Sigma;\ \Gamma;\ \Delta \rightarrow \Sigma';\ \Gamma';\ \Delta'$$
$$\text{and}\ \ \Sigma';\ \Gamma';\ \Delta' \blacktriangleright^* \ \Sigma'';\ \Delta''$$

- logic
- system ω
- rewriting
- processes
- security

$$\frac{\Gamma;\ \Delta' \dashrightarrow_\Sigma A \quad \Gamma;\ \Delta, B \dashrightarrow_\Sigma C}{\Gamma;\ \Delta, \Delta',\ A \multimap B \dashrightarrow_\Sigma C}$$

$$\Sigma;\ \Gamma;\ (\Delta, \Delta',\ A \multimap B) \rightarrow \Sigma;\ \Gamma;\ (\Delta, B)$$
$$\text{if}\ \Sigma;\ \Gamma;\ \Delta' \blacktriangleright^* \ \Sigma;\ A$$

$$\frac{\Gamma;\ \Delta' \dashrightarrow_\Sigma A \quad \Gamma;\ \Delta, A \dashrightarrow_\Sigma C}{\Gamma;\ \Delta, \Delta' \dashrightarrow_\Sigma C}$$

$$\Sigma;\ \Gamma;\ \Delta, \Delta' \rightarrow \Sigma;\ \Gamma;\ (A, \Delta)$$
$$\text{if}\ \Sigma;\ \Gamma;\ \Delta' \blacktriangleright^* \Sigma;\ A$$

...   ...

Formal Correspondence

- Soundness

If $\quad \Sigma \, ; \, \Gamma \, ; \, \Delta \; \twoheadrightarrow^* \; \Sigma,\Sigma'; \, \Delta'$

then $\quad \Gamma \, ; \, \Delta \; \dashrightarrow_\Sigma \; \exists\Sigma'. \; \otimes \; \Delta'$

- Completeness?

➢No! We have only crippled right rules

$\bullet \, ; \, \bullet \, ; \, a \multimap b, \, b \multimap c \; \cancel{\twoheadrightarrow}^* \; \bullet \, ; \, a \multimap c$

System ω

- With cut, rule for $\multimap$ can be simplified to

$$\Sigma;\ \Gamma;\ (\Delta,\ A,\ A \multimap B) \rightarrow \Sigma;\ \Gamma;\ (\Delta,\ B)$$

- Cut elimination holds

  = in-lining of auxiliary rewrite chains

  ➢ But …

    - Careful with extra signature symbols
    - Careful with extra persistent objects

- No rule for → needs a premise

  ➢ → does not depend on ➔*

Summary – Syntax

$$
\begin{array}{llll}
A, B & ::= & a & \text{atomic object} \\
& | & 1 & \text{empty} \\
& | & A \otimes B & \text{formation} \\
& | & A \multimap B & \text{rewrite} \\
& | & \top & \text{no-op} \\
& | & A \mathbin{\&} B & \text{choice} \\
& | & \forall x.\, A & \text{instantiation} \\
& | & \exists x.\, A & \text{generation} \\
& | & !\, A & \text{replication}
\end{array}
$$

- logic
- system ω
- **rewriting**
- processes
- security

Summary – Semantics

$\multimap$	$\Sigma \; ; \; \Gamma \; ; \; (\Delta, A, A \multimap B)$	$\rightarrow \Sigma \; ; \; \Gamma \; ; \; (\Delta, B)$
$\top$	(no rules)	
$\&$	$\Sigma \; ; \; \Gamma \; ; \; (\Delta, A_1 \& A_2)$	$\rightarrow \Sigma \; ; \; \Gamma \; ; \; (\Delta, A_i)$
$\forall$	$\Sigma \; ; \; \Gamma \; ; \; (\Delta, \forall x. A)$	$\rightarrow \Sigma \; ; \; \Gamma \; ; \; (\Delta, [t/x]A) \quad$ if $\Sigma \vdash t$
$!$	$\Sigma \; ; \; \Gamma \; ; \; (\Delta, !A)$	$\rightarrow \Sigma \; ; \; (\Gamma, A) \; ; \; \Delta$
	$\Sigma \; ; \; (\Gamma, A) \; ; \; \Delta$	$\rightarrow \Sigma \; ; \; (\Gamma, A) \; ; \; (\Delta, A)$
1	$\Sigma \; ; \; \Gamma \; ; \; (\Delta, 1)$	$\rightarrow \Sigma \; ; \; \Gamma \; ; \; \Delta$
$\otimes$	$\Sigma \; ; \; \Gamma \; ; \; (\Delta, A \otimes B)$	$\rightarrow \Sigma \; ; \; \Gamma \; ; \; (\Delta, A, B)$
$\exists$	$\Sigma \; ; \; \Gamma \; ; \; (\Delta, \exists x. A)$	$\rightarrow (\Sigma, x) \; ; \; \Gamma \; ; \; (\Delta, A)$

Discussion

- Other connectives?
  - ⊕, 0, ℘, ⊥
    - Odd rewrite properties
  - ?, (_)$^\perp$
    - 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: Π (LF)
        - Asynchronous connectives: $\multimap$, &, $\top$ (LLF)
        - Synchronous connectives: $\otimes$, 1, !, $\exists$
        - Monadic sandboxing
        - Concurrency equations
    ➢ Faithful encoding of true concurrency
        - Petri nets, MSR 2 specs, π-calculus, concurrent ML

- Details of relation still unclear

- logic
- system ω
- rewriting
- processes
- security

Multiset Rewriting

- Multiset: set with repetitions allowed

$$\underline{a} \ ::= \ \bullet \ | \ a, \underline{a}$$

  - 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 F0 atomic formulas
- Rules have the form

$$\forall x_1 \ldots x_n. \; \underline{a}(\boldsymbol{x}) \rightarrow \exists y_1 \ldots y_k. \; \underline{b}(\boldsymbol{x}, \boldsymbol{y})$$

- Semantics

$$\Sigma \, ; \, \underline{a}(\boldsymbol{t}), \, \underline{s} \; \rightarrow_{R, \, (\underline{a}(\boldsymbol{x}) \rightarrow \exists y. \, \underline{b}(\boldsymbol{x}, \boldsymbol{y}))} \; \Sigma, \boldsymbol{y} \, ; \, \underline{b}(\boldsymbol{t}, \boldsymbol{y}), \, \underline{s}$$

$$\text{if } \Sigma \vdash t$$

- 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 $\multimap$
  - $\multimap$ never nested, always persistent

  $\quad\quad\quad\quad \Sigma\ ;\ \underline{s}\ \rightarrow_R\ \Sigma'\ ;\ \underline{s}'$

  iff $\quad\quad \Sigma\ ;\ \text{"R"}\ ;\ \text{"}\underline{s}\text{"} \rightarrow^*\ \Sigma'\ ;\ \text{"}\underline{s}'\text{"}$

- 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

    $P ::= 0 \mid P||Q \mid \nu\, x.\, P \mid\, !P \mid x(y).P \mid \underline{x}\langle y\rangle$

- ## Semantics

    - ➤ **Structural equivalence**
        - Comm. monoidal congruence of || and 0
        - Binder mobility congruence of ν
            - $\nu\, x.\, \nu\, y.\, P \equiv \nu\, y.\, \nu\, x.\, P$
            - $0 \equiv \nu\, x.\, 0$
            - $P\,||\, \nu\, x.\, Q \equiv \nu\, x.\, (P\,||\,Q)$ if $x \notin FN(P)$
        - $!P \equiv\, !P\,||\,P$
    - ➤ **Reaction law**
        - $\underline{x}\langle y\rangle\,||\,x(z).\,P\,||\,Q \Rightarrow [y/z]P\,||\,Q$

π-calculus in ω-Multisets

- 0 ⇔ 1
- || ⇔ ⊗
- ν ⇔ ∃

- ! ⇔ !
- x(y). P ⇔ ∀y. ch(x,y) ⊸ "P"
- <u>x</u><y> ⇔ ch(x,y)

- Reaction law
  - ➢ Σ; Γ; ch(x,y), ∀z. ch(x,z) ⊸ P, Δ →² Σ; Γ; [y/z]P, Δ

- Structural equivalence
  - ➢ Monoidal congr. of || and 0 ⇔ monoidal laws of ⊗ and 1
  - ➢ Mobility congr. of ν ⇔ mobility laws of ∃
  - ➢ !P ≡ !P || P
    - ▪ Only ⇒ in ω-multisets
    - ▪ Oversight in the π-calculus?

Properties

- If P =>* Q

  then •; •; "P" →* Σ; Γ; Δ

  where "Q" = ∃Σ. !Γ ⊗ Δ   mod !A = !A⊗A

  > Note: with !P → !P || P as a transition
  >
  > - If P =>* Q
  >
  >   then •; •; "P" →* Σ; Γ; Δ
  >
  >   where "Q" = ∃Σ. !Γ ⊗ Δ

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 ω-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 $\rightarrow$
    - Continuation-passing style
    - As in process algebra
  - (Intermediate approaches)

State-Based

$A \rightarrow B$: $\{n_A, A\}_{kB}$
$B \rightarrow A$: $\{n_A, n_B\}_{kA}$
$A \rightarrow B$: $\{n_B\}_{kB}$

MSR 2 spec.

$\forall A$: princ.

$\{ \exists L$: princ $\times \Sigma B$:princ.pubK B $\times$ nonce $\rightarrow$ mset.

$\forall B$: princ. $\forall k_B$: pubK B.

$\quad \bullet$

$\rightarrow \exists n_A$: nonce.

$\quad$ net $(\{n_A, A\}_{kB})$, L (A, B, k_B, n_A)

$\forall B$: princ. $\forall k_B$: pubK B.

$\forall k_A$: pubK A. $\forall k_A'$: prvK k_A.

$\forall n_A$: nonce. $\forall n_B$: nonce.

$\quad$ net $(\{n_A, n_B\}_{kA})$, L (A, B, k_B, n_A)

$\rightarrow$ net $(\{n_B\}_{kB})$

$\}$

- logic
- system ω
- rewriting
- processes
- security

Interpretation of L

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

Process-Based

$A \rightarrow B: \{n_A, A\}_{kB}$
$B \rightarrow A: \{n_A, n_B\}_{kA}$
$A \rightarrow B: \{n_B\}_{kB}$

$\forall A$:princ.

$\forall B$: princ. $\forall k_B$: pubK B.

- $\rightarrow \quad \exists n_A$: nonce.

net $(\{n_A, A\}_{kB})$,

$(\forall k_A$: pubK A. $\forall k_A'$: prvK k_A. $\forall n_B$: nonce.

net $(\{n_A, n_B\}_{kA}) \rightarrow$ net $(\{n_B\}_{kB}))$

- Succinct
- Continuation-passing style
  - ➢ Rule asserts what to do next
  - ➢ Lexical control flow

- State is implicit
  - ➢ Abstract

- logic
- system ω
- rewriting
- processes
- security

NSPK in Process Algebra

$$A \to B: \{n_A, A\}_{kB}$$
$$B \to A: \{n_A, n_B\}_{kA}$$
$$A \to B: \{n_B\}_{kB}$$

$\forall A$:princ.

$\forall B$: princ. $\forall k_B$: pubK B.

$\forall k_A$: pubK A. $\forall k_A'$: prvK k_A. $\forall n_B$: nonce.

νn_A: nonce.

$\quad$ net $(\{n_A, A\}_{kB})$.

$\qquad$ <u>net</u> $<\{n_A, n_B\}_{kA}>$.

$\qquad\qquad$ net $(\{n_B\}_{kB})$. 0

Same structure !

- ➤ Not a coincidence
- ➤ MSR 3 very close to Process Algebra
  - ▪ ω-multiset encodings of π-calculus and Join Calculus

- logic
- system ω
- rewriting
- processes
- security

- **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
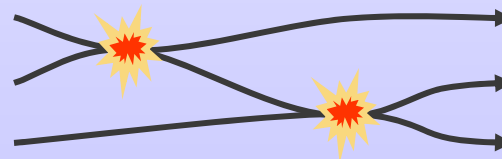 - NRL Prot. Analyzer, CAPSL/CIL, Paulson's approach, …
  - ➢ State transition semantics

- ## Process-based languages
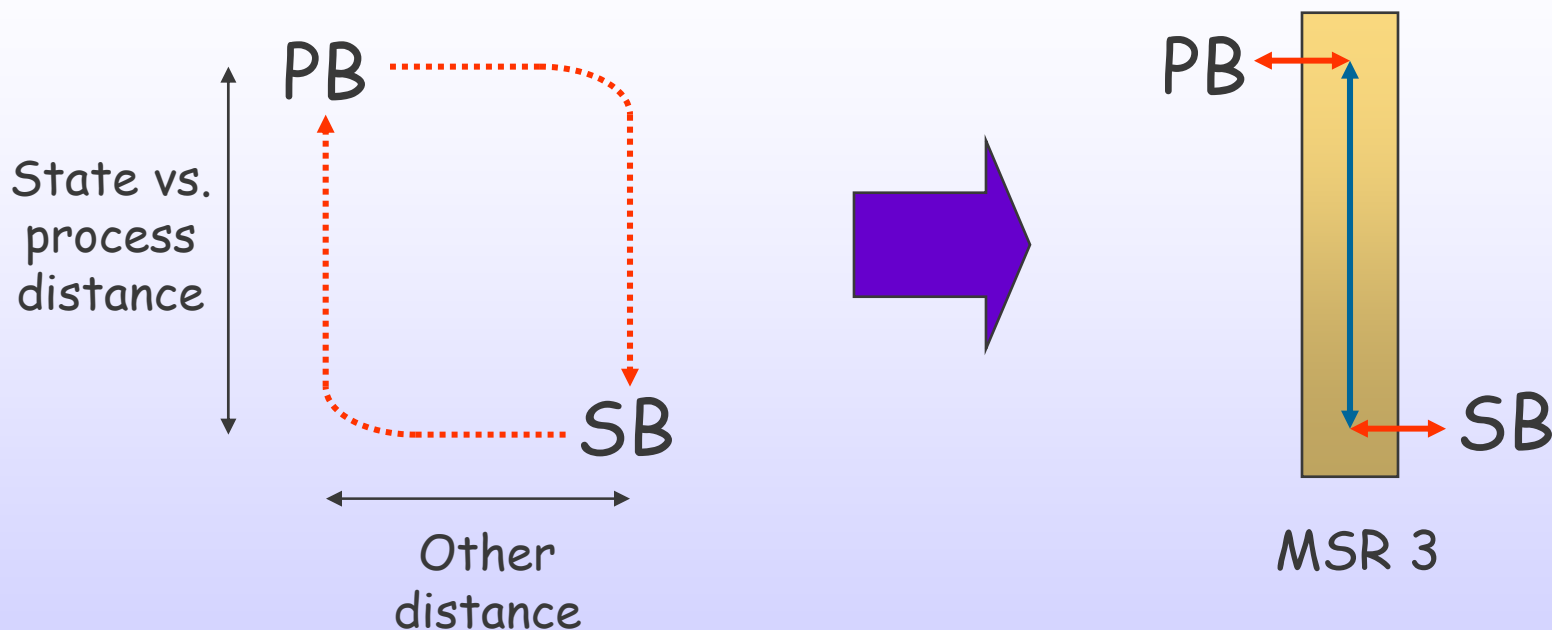 - Process Algebra
  - Strand spaces, spi-calculus, …
  - ➢ Independent communicating threads

MSR 3 Bridges the Gap

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

State vs. process distance

PB

SB

Other distance

➡

PB

SB

MSR 3

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