The Logical Meeting Point of Multiset Rewriting and Process Algebra

Iliano Cervesato `iliano@math.tulane.edu`

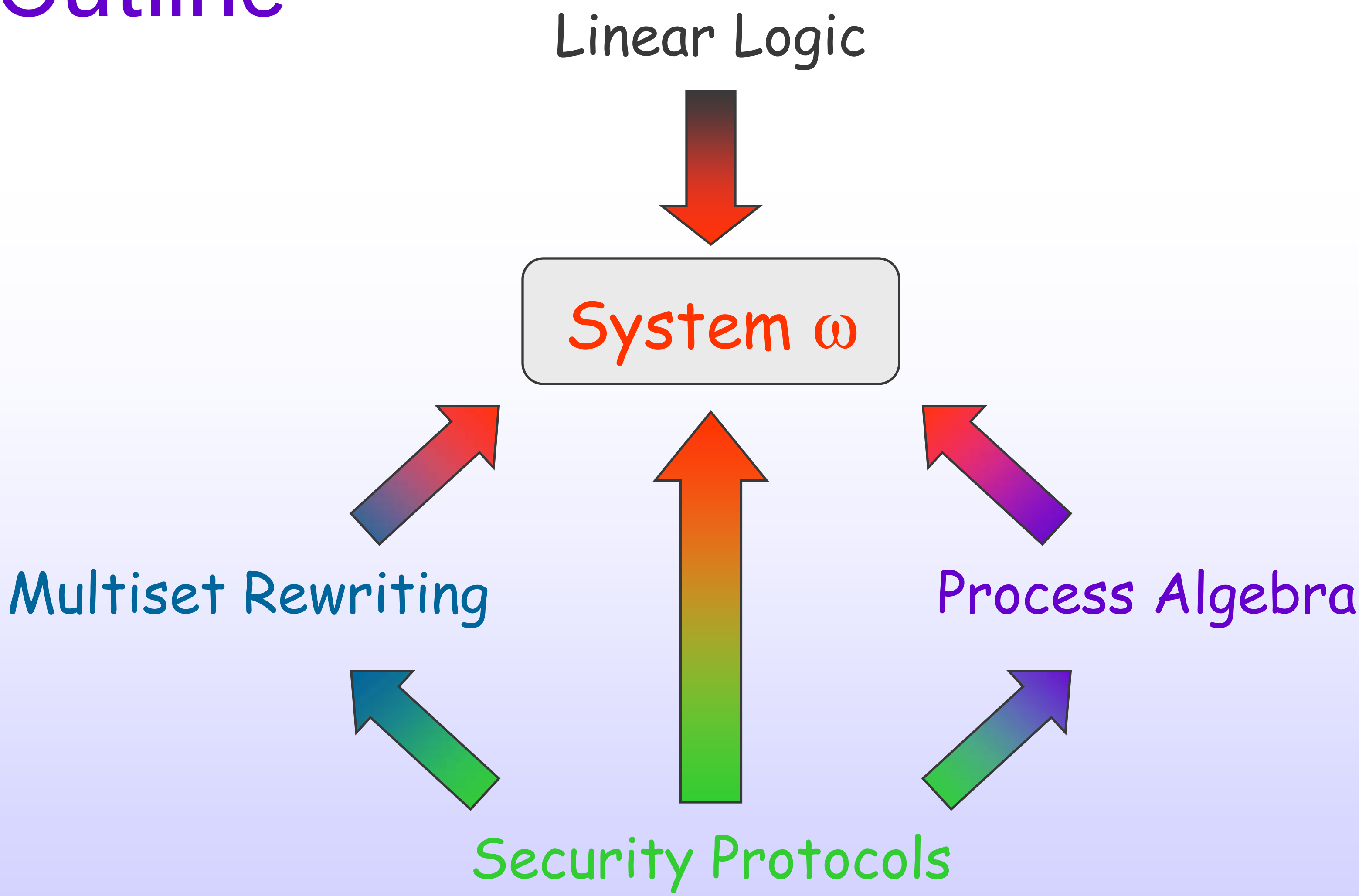
ITT Industries, Inc -> Tulane University

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

Outline



http://theory.stanford.edu/~iliano/forthcoming.html

Linear Logic

- logic
- system ω
- rewriting
- processes
- security

- 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: "•"

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

Unrestricted context

Linear context

Signature

Goal formula

Some LV Rules

- logic
- system ω
- rewriting
- processes
- security

Left rules

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

$$\frac{\Gamma; \Delta' \longrightarrow_\Sigma A \qquad \Gamma; \Delta, B \longrightarrow_\Sigma C}{\Gamma; \Delta, \Delta', A \multimap B \longrightarrow_\Sigma C}$$

$$\frac{\Sigma \mid\!- t \qquad \Gamma; \Delta, [t/x]A \longrightarrow_\Sigma C}{\Gamma; \Delta, \forall x.A \longrightarrow_\Sigma C}$$

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

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

…

Structural rules

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

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

Cut rules

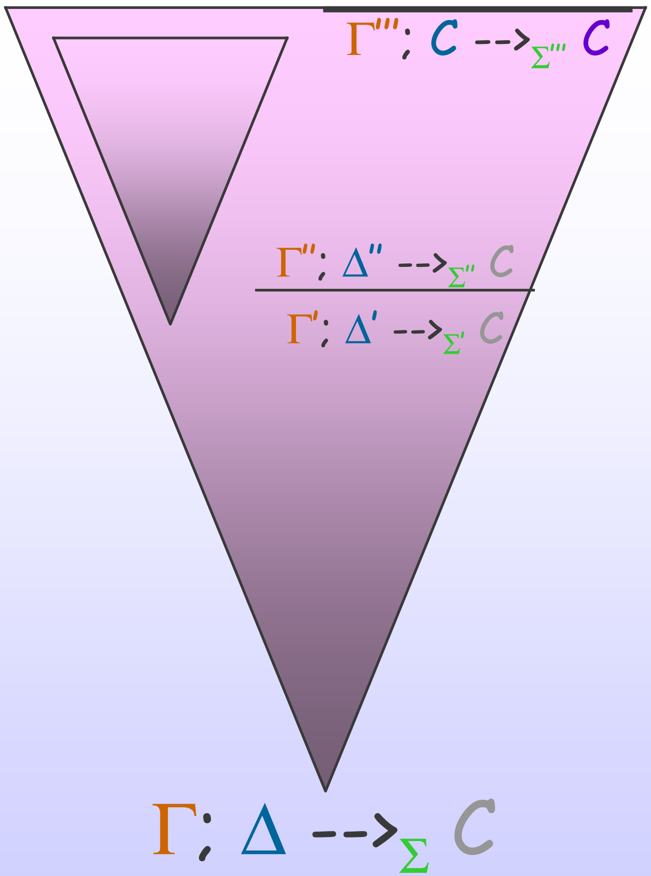
$$\frac{\Gamma; \Delta' \longrightarrow_\Sigma A \qquad \Gamma; \Delta, A \longrightarrow_\Sigma C}{\Gamma; \Delta, \Delta' \longrightarrow_\Sigma C}$$

$$\frac{\Gamma; \bullet \longrightarrow_\Sigma A \qquad \Gamma, A; \Delta \longrightarrow_\Sigma C}{\Gamma; \Delta \longrightarrow_\Sigma C}$$

Right rules

…

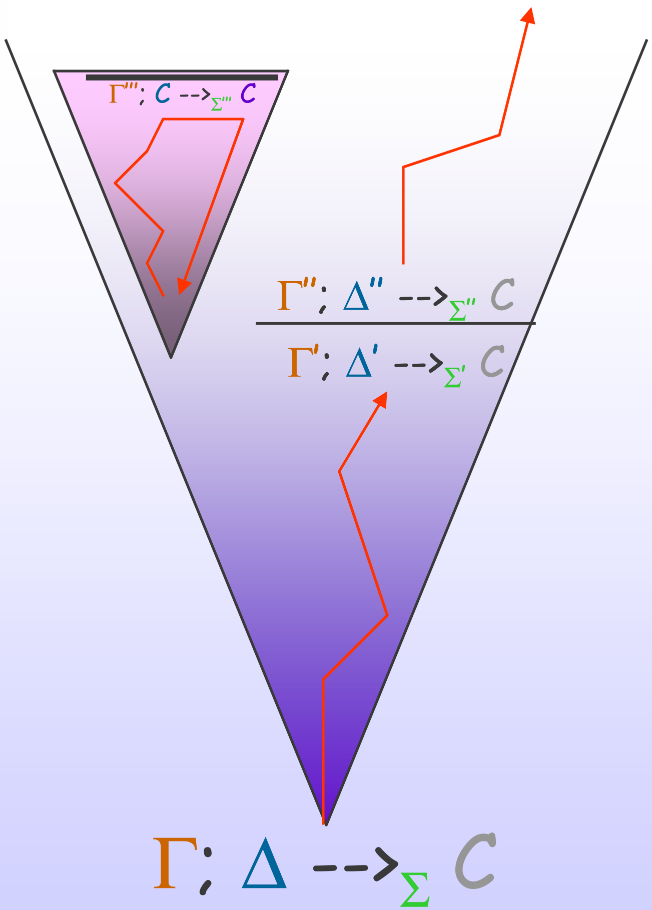
Logical Derivations



- 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

- logic
- system ω
- rewriting
- processes
- security

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

State and Transitions

- States

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

  - 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

Interpreting Unary Rules

- logic
- system ω
- rewriting
- processes
- security

$$\frac{\Gamma;\ \Delta, A, B \dashrightarrow_{\Sigma} C}{\Gamma;\ \Delta, A \otimes B \dashrightarrow_{\Sigma} C} \qquad \Sigma;\ \Gamma;\ (\Delta, A \otimes B) \rightarrow \Sigma;\ \Gamma;\ (\Delta, A, B)$$

$$\frac{\boxed{\Sigma \mid\!- t} \quad \Gamma;\ \Delta, [t/x]A \dashrightarrow_{\Sigma} C}{\Gamma;\ \Delta, \forall x.A \dashrightarrow_{\Sigma} C} \qquad \Sigma;\ \Gamma;\ (\Delta, \forall x.\ A) \rightarrow \Sigma;\ \Gamma;\ (\Delta, [t/x]A)$$

$$\boxed{\text{if } \Sigma \mid\!- t}$$

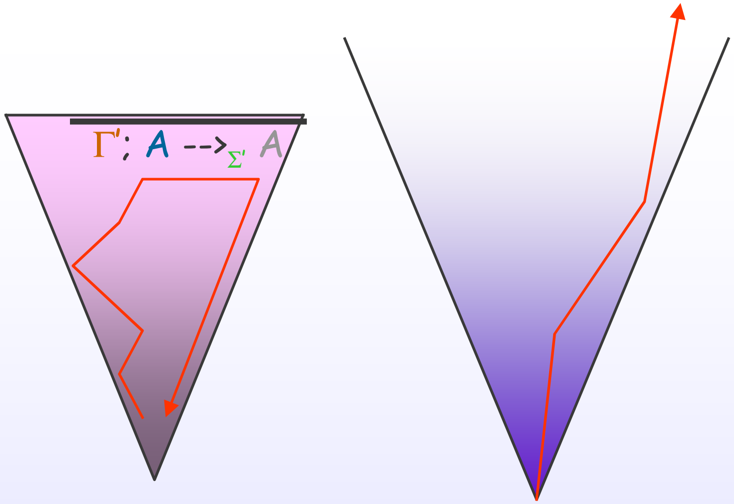
$$\frac{\Gamma;\ \Delta, A \dashrightarrow_{\Sigma, x} C}{\Gamma;\ \Delta, \exists x.A \dashrightarrow_{\Sigma} C} \qquad \Sigma;\ \Gamma;\ (\Delta, \exists x.\ A) \rightarrow (\Sigma, x);\ \Gamma;\ (\Delta, A)$$

$$\frac{\Gamma, A;\ \Delta \dashrightarrow_{\Sigma} C}{\Gamma;\ \Delta, !A \dashrightarrow_{\Sigma} C} \qquad \Sigma;\ \Gamma;\ (\Delta, !A) \rightarrow \Sigma;\ (\Gamma, A);\ \Delta$$

... ...

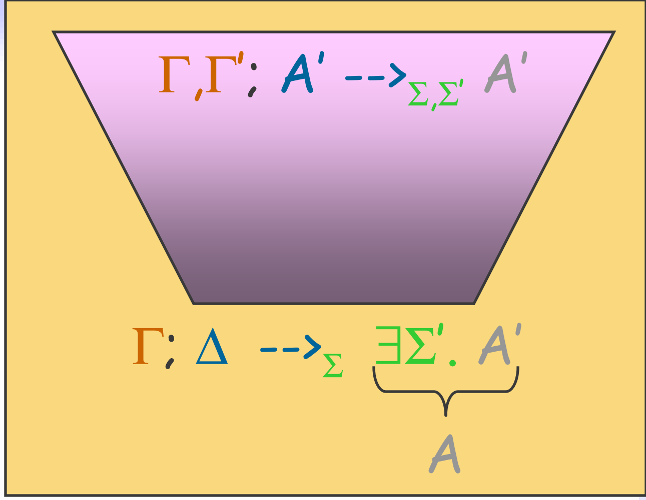
Binary Rules and Axiom

- logic
- system ω
- rewriting
- processes
- security



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

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



Observations

- Observation states

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

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

    Categorical semantics

$\Delta = \otimes\Delta$

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

    De Bruijn's telescopes

$\Sigma; \Delta = \exists\Sigma. \otimes\Delta$

- Observation transitions

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

Structural Equivalences

Monoidal laws

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

Mobility laws

- $\exists x.\ \exists y.\ \Delta \quad = \quad \exists y.\ \exists x.\ \Delta$
- $\exists x.\ \bullet \quad = \quad \bullet$
- $\exists x.\ (\Delta, \Delta') \quad = \quad \Delta, \exists x.\ \Delta'$ if $x \notin FV(\Delta)$

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

Interpreting Binary Rules

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

$$\Sigma; \Gamma; \Delta \rightarrow^{*} \Sigma; \Delta$$

$$\Sigma; \Gamma; \Delta \rightarrow^{*} \Sigma''; \Delta''$$

$$\text{if} \quad \Sigma; \Gamma; \Delta \rightarrow \Sigma'; \Gamma'; \Delta'$$

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

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

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

$$\boxed{\text{if} \quad \Sigma; \Gamma; \Delta' \rightarrow^{*} \Sigma; A}$$

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

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

$$\boxed{\text{if} \quad \Sigma; \Gamma; \Delta' \rightarrow^{*} \Sigma; A}$$

... ...

- logic
- system ω
- rewriting
- processes
- security

- logic
- system ω
- rewriting
- processes
- security

Formal Correspondence

- Soundness

  If $\Sigma \,;\, \Gamma \,;\, \Delta \;\rightarrow^{*}\; \Sigma, \Sigma' ;\, \Delta'$

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

- Completeness?
  - No! We have only crippled right rules

    $\bullet \,;\, \bullet \,;\, a \multimap b,\, b \multimap c \;\not\rightarrow^{*}\; \bullet \,;\, a \multimap c$

System ω

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

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

- Cut elimination holds

  = in-lining of auxiliary rewrite chains

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

- No rule for $\to$ needs a premise
  - $\to$ does not depend on $\to^*$

- logic
- system ω
- rewriting
- processes
- security

Discussion

- Other connectives?
  - $\oplus$, 0, $\wp$, $\perp$
    - 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

Multiset Rewriting

- Multiset: set with repetitions allowed

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

  - 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

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

- Semantics

$$\boxed{\Sigma\ ;\ \underline{a}(\mathbf{t}),\ \underline{s} \ \rightarrow_{R,\ (\underline{a}(\mathbf{x}) \rightarrow \exists \mathbf{y}.\ \underline{b}(\mathbf{x},\mathbf{y}))}\ \Sigma,\mathbf{y}\ ;\ \underline{b}(\mathbf{t},\mathbf{y}),\ \underline{s} \qquad \text{if } \Sigma \mid\!- \mathbf{t}}$$

- Several encodings into linear logic
  - [Martí-Oliet, Meseguer, 91]

- logic
- system ω
- rewriting
- processes
- security

ω-Multisets vs. Multiset Rewriting

- MSR 1 is an <u>instance</u> of ω-multisets
  - Uses only $\otimes$, 1, $\forall$, $\exists$, and $\multimap$
  - $\multimap$ never nested, always persistent
  - ➢ $\Sigma ; \underline{s} \rightarrow_{R} \Sigma' ; \underline{s}'$

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

- Interpretation of MSR <u>as</u> linear logic
  - ➢ Logical explanation of multiset rewriting
    - MSR is logic
  - ➢ Guideline to design rewrite systems

- logic
- system ω
- rewriting
- processes
- security

ω-Rewriting

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

- logic
- system ω
- rewriting
- processes
- security

The Asynchronous π-Calculus

Another fundamental model of distributed computing

- Language

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

- 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

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

- Reaction law
  - $\Sigma;\ \Gamma;\ ch(x,y),\ \forall z.\ ch(x,z) \multimap P,\ \Delta \rightarrow^2 \Sigma;\ \Gamma;\ [y/z]P,\ \Delta$
- 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 ω-multisets
    - Oversight in the π-calculus?

- logic
- system ω
- rewriting
- processes
- security

Properties

- If $P \Rightarrow^* Q$
  then $\bullet; \bullet; \text{“}P\text{”} \rightarrow^* \Sigma; \Gamma; \Delta$
  where $\text{“}Q\text{”} = \exists\Sigma.\ !\Gamma \otimes \Delta$ mod $!A = !A \otimes A$
  - Note: with $!P \mid\mid P \rightarrow\ !P$ as a transition
    - If $P \Rightarrow^* Q$
      then $\bullet; \bullet; \text{“}P\text{”} \rightarrow^* \Sigma; \Gamma; \Delta$
      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 ω-multisets for cryptographic protocol specification
  - Security-relevant signature
  - Typing infrastructure
  - Modules, equations, …
- 3rd generation
  - MSR 1: First-order multiset rewriting with ∃
    - Undecidability of protocol analysis
  - MSR 2: MSR 1 + typing
    - Actual specification language
    - More theoretical results
    - Implementation underway

- logic
- system ω
- rewriting
- processes
- security

The Atomic Objects of MSR 3

- logic
- system ω
- rewriting
- processes
- security

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, …
- Subsorting

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

State-Based

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

- logic
- system ω
- rewriting
- processes
- security

MSR 2 spec.

$\forall A$: princ.

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

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

$\bullet$

$\to \exists n_A$: nonce.

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.

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

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

$\}$

Interpretation of L

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

Process-Based

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

$\forall A$:princ.

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

$\bullet \to \exists n_A$: nonce.

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

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

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

- logic
- system ω
- rewriting
- processes
- security

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

NSPK in Process Algebra

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

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

νn_A: nonce.
  net $(\{n_A, A\}_{kB})$.
    $\underline{\text{net}}\ \langle\{n_A, n_B\}_{kA}\rangle$.
      net $(\{n_B\}_{kB})$. 0

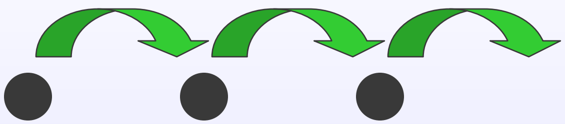
Same structure !
- Not a coincidence
- MSR 3 very close to Process Algebra
  - ω-multiset encodings of π-calculus and Join Calculus

- MSR 3 is promising middle-ground for relating
  - State-based
  - Process-based

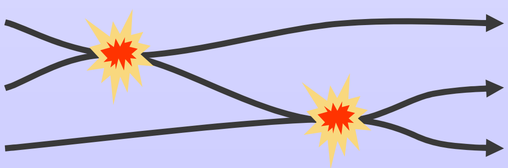
  representations of a problem

State-Based vs. Process-Based

- State-based languages
  - Multiset Rewriting
  - NRL Prot. Analyzer, CAPSL/CIL, Paulson's approach, …
  - ➢ State transition semantics

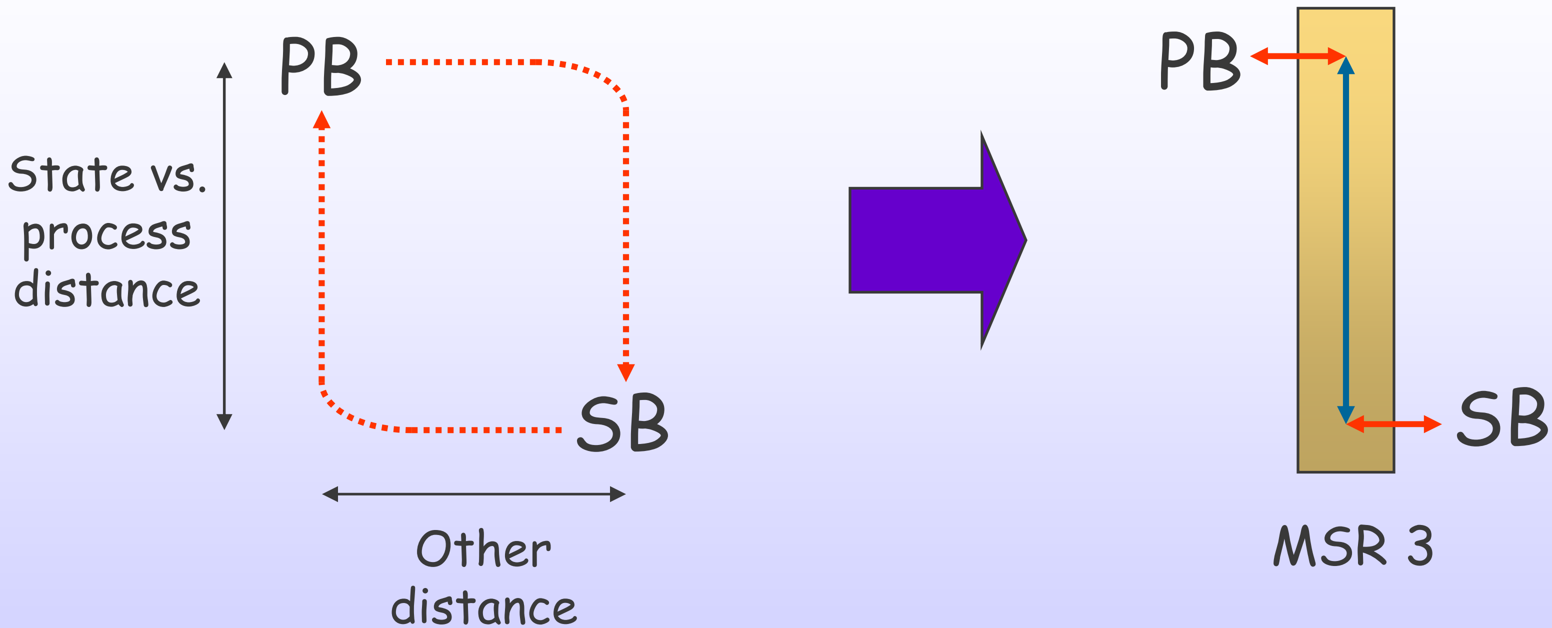


- Process-based languages
  - Process Algebra
  - Strand spaces, spi-calculus, …
  - ➢ Independent communicating threads



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

- logic
- system ω
- rewriting
- processes
- security

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