MSR 3.0:

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

Brief History of MSR

- **MSR 1** [CSFW'99]
  - To formalize security protocols specification
  - First-order multiset rewriting with ∃
    - Undecidability of security protocol verification
    - Comparison with Strand Spaces
- **MSR 2** [MMM'01]
  - Add typing infrastructure, liberalize syntax
    - Specification of Kerberos V
    - Completeness of Dolev-Yao attacker
    - Subsorting view of type-flaw attacks
    - Implementation (undergoing)
    - Comparison with Process Algebra

MSR 3

- From multisets to ω-multisets
  - Embeds multiset rewriting
    - MSR 1, 2
    - Paulson's inductive traces
    - Tool-specific languages
      - NRL Protocol Analyzer
      - Murφ, …
  - Encompasses Process Algebra
    - Strand spaces
    - Crypto-SPA
    - Spi-calculus
  - Founded on logic

Indirect contributors

- Fabio Martinelli
- Dale Miller
- Andre Scedrov
- Frank Pfenning

ω-Multisets
MSR 2
Protocols
Repr. gap

What is in MSR 3 ?

- Instance of ω-multisets for cryptographic protocol specification
- Security-relevant signature
  - Network
  - Encryption, …

  } From MSR 1
- Typing infrastructure
  - Dependent types
  - Subsorting
- Data Access Specification (DAS)

  } From MSR 2
- Module system
- Equations

  } From MSR 2 implementation

ω-Multisets

Specification language for concurrent systems

- Crossroad of
  - State transition languages
    - Petri nets, multiset rewriting, …
  - Process calculi
    - CCS, π-calculus, …
  - (Linear) logic
- Benefits
  - Analysis methods from logic and type theory
  - Common ground for comparing
    - Multiset rewriting
    - Process algebra
  - Allows multiple styles of specification
    - Unified approach

Syntax

a	::=	P	atomic object
	|	•	empty
	|	a, b	formation
	|	$a \to b$	rewrite
	|	$\forall x.\ a$	instantiation
	|	$\exists x.\ a$	generation

Generalizes FO multiset rewriting (MSR 1-2)

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

Judgments

- Base step

$$\Sigma ; s \rightarrow_R \Sigma' ; s'$$

- Finite iteration

$$\Sigma ; s \rightarrow^*_R \Sigma' ; s'$$

  - Reflexive and transitive closure of →
  - Useful for reachability analysis

Operational Semantics

$\rightarrow$	$\Sigma ; (s, a, a \rightarrow b)$	$\rightarrow_R$	$\Sigma ; (s, b)$
$\forall$	$\Sigma ; (s, \forall x.\, a)$ if $\Sigma \vdash t$	$\rightarrow_R$	$\Sigma ; (s, [t/x]a)$
$\exists$	$\Sigma ; (s, \exists x.\, a)$	$\rightarrow_R$	$(\Sigma, x) ; (s, a)$
	$\Sigma ; s$	$\rightarrow_{R,a}$	$\Sigma ; (s, a)$

$$\Sigma ; s \rightarrow^*_R \Sigma ; s$$

$$\Sigma ; s \rightarrow^*_R \Sigma'' ; s'' \quad \text{if } \Sigma ; s \rightarrow_R \Sigma' ; s' \text{ and } \Sigma' ; s' \rightarrow^*_R \Sigma'' ; s''$$

Logical Foundations

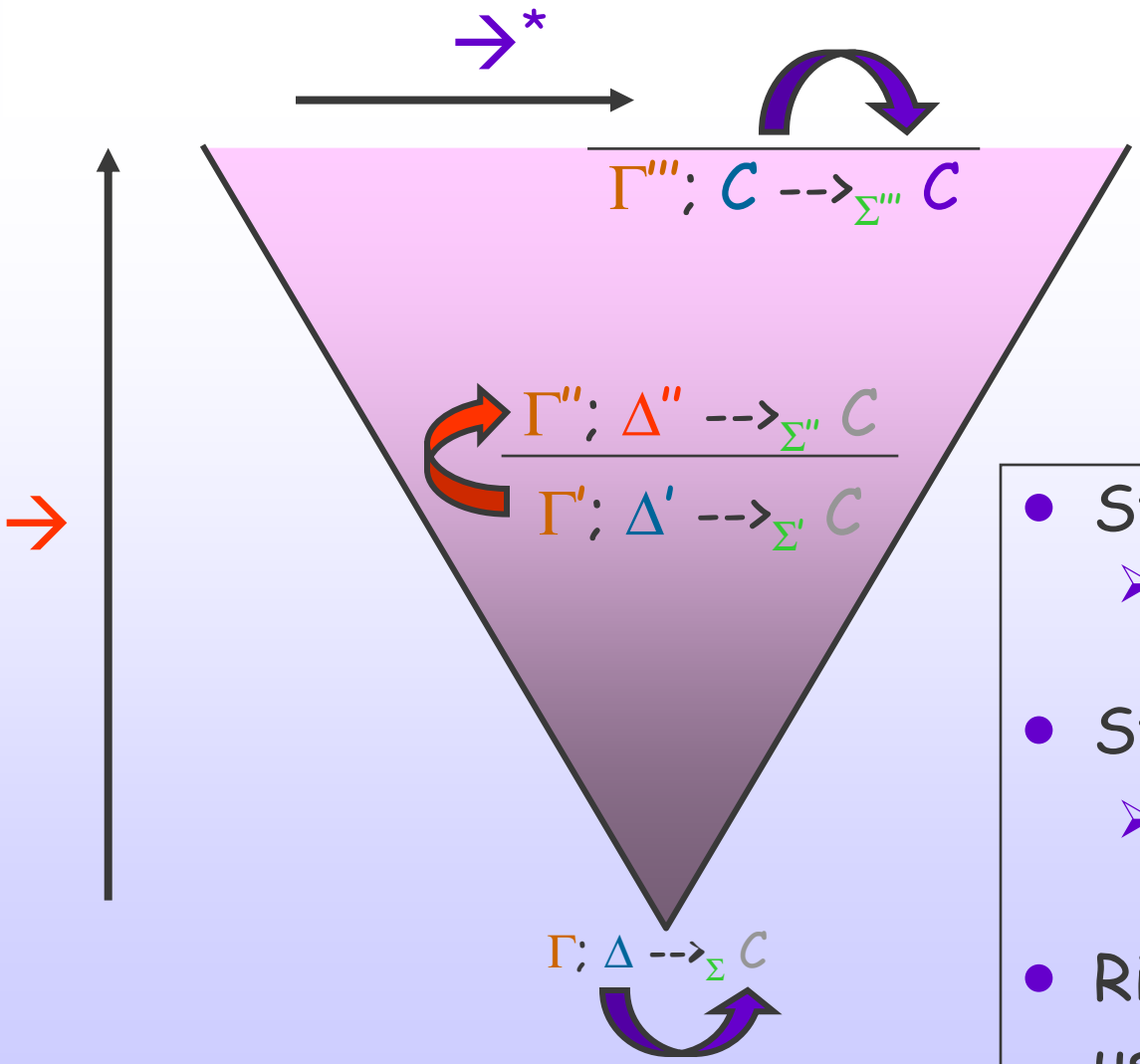
→ is left sequent rules of linear logic

- • ⇔ 1 , ⇔ ⊗ → ⇔ —o

- MSR 3 is logic
  - Guideline for extensions with new operators
    - Non-deterministic choice (+)
    - Replication (!)
    - … more
- Related to Concurrent Logical Framework

ω-Multiset View of Derivations

ω-Multisets
MSR 2
Protocols
Repr. gap



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

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

Subsorting

$$\tau <: \tau'$$

- Allows atomic terms in messages
- Definable
  - Non-transmittable terms
  - Sub-hierarchies
- Discriminant for type-flaw attacks

Data Access Specification

- Prevent illegitimate use of information
  - Protocol specification divided in roles
    - Owner = principal executing the role
  - A signing/encrypting with B's key
  - A accessing B's private data, …
- Simple static check
- Central meta-theoretic notion
  - Detailed specification of Dolev-Yao access model
- Gives meaning to Dolev-Yao intruder
- Current effort towards integration in type system
  - Definable
    - Possibility of going beyond Dolev-Yao model

Modules and Equations

- Modules
  - Bundle declarations with simple import/export interface
  - Keep specifications tidy
  - Reusable
- Equations

  (For free from underlying Maude engine)
  - Specify useful algebraic properties
    - Associativity of pairs
  - Allow to go beyond free-algebra model
    - Dec(k, Enc(k, M)) = M

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 $\times \sum B$:princ.pubK B $\times$ nonce $\rightarrow$ mset.

$\forall B$: princ. $\forall k_B$: pubK B.
•
$\rightarrow \exists n_A$: 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)
$\rightarrow$ net $(\{n_B\}_{kB})$

}

Interpretation of L

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

Process-Based

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

$\forall A$:princ.

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

- $\rightarrow$ $\exists n_A$: 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

NSPK in Process Algebra

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

ω-Multisets
MSR 2
Protocols
Repr. gap

$\forall A$: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 $(\{n_A, A\}_{kB})$.
net $\langle\{n_A, n_B\}_{kA}\rangle$.
net $(\{n_B\}_{kB})$. 0

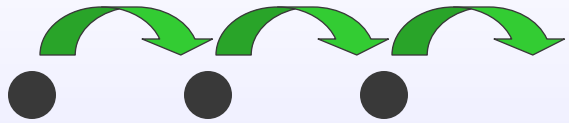
Same structure !
- Not a coincidence
- MSR 3 very close to Process Algebra
  - ω-multiset encodings of π-calculus
  - Ties to Join Calculus

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

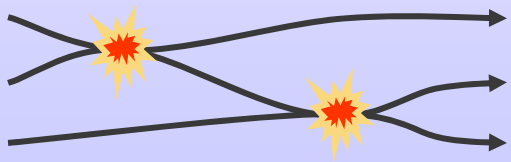
representations of a problem

State-Based vs. Process-Based

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

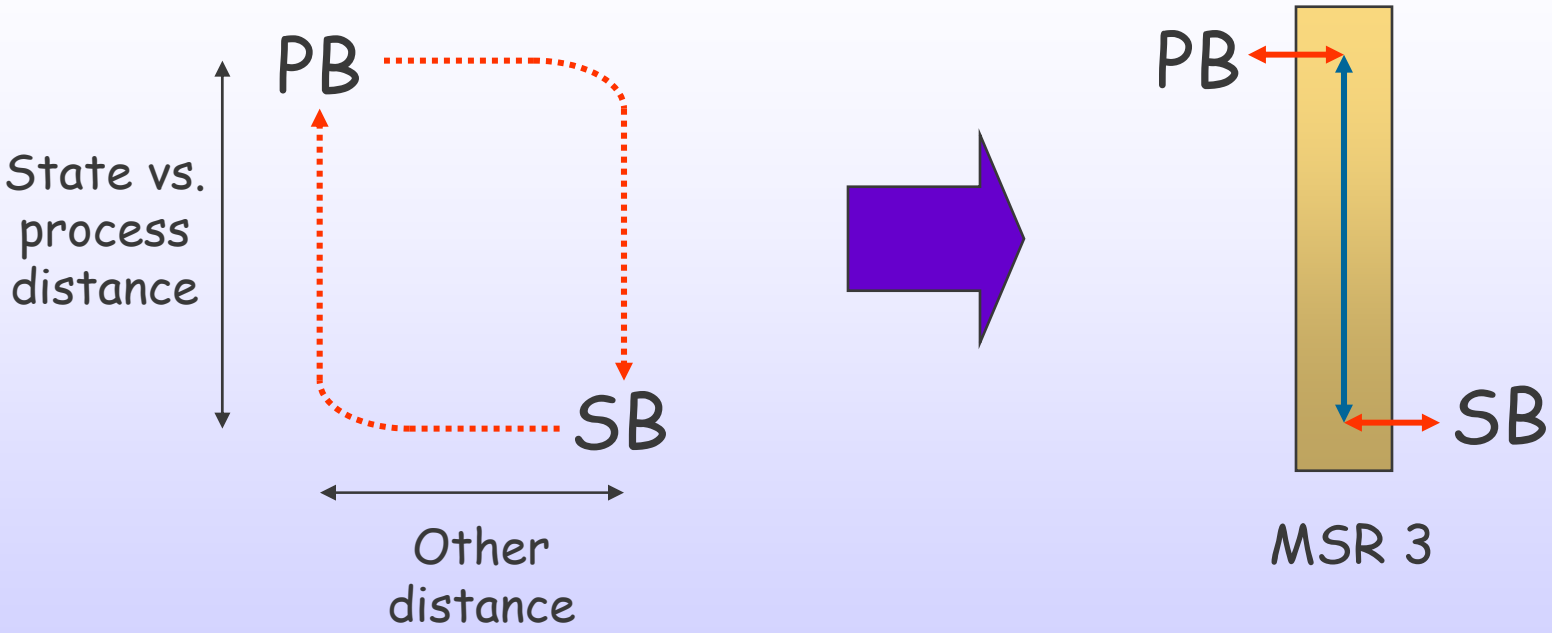


- Process-based languages
  - Process Algebra
  - Strand spaces, spi-calculus, …
  - ➢ Independent communicating threads



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

Summary

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