Event-based Methods for Security Protocols

Federico Crazzolara
C&C Laboratories, NEC Europe

(joint work with G. Winskel while at BRICS)

DIMACS, July 8, 2003

Road map

1) Security Protocol Language (SPL)

- Transition vs. event-based semantics

2) Relation between models (finite behaviours)

- SPL & Basic Nets, Event Structures, Inductive Rules
- SPL & Strand Spaces
- Strand Spaces & Event Structures

High level, special purpose language

Program, verify & compile

- program: concise & precise protocol description
- formal semantics that supports protocol verification

  => reduce gap between protocol & model

- compile verified program

  => correct protocol code

Security Protocol Language (SPL)

- asynchronous, process oriented language
- abstracts concrete network with a tuple space

- messages: **v | k | M,M' | $\{M\}_k$ | ψ**
- prefixing:
  - new-name generation & send: **out new(x) M . p**
  - input with pattern matching: **in pat(ψ) N . p**
- parallel composition of processes: $\|_{i \in I}\ \mathbf{p_i}$

ISO mutual authentication in SPL

$$(1)\quad A \rightarrow B \;:\; n$$

$$(2)\quad B \rightarrow A \;:\; \{n, m, B, K\}_{Key(A,B)}$$

$$(3)\quad A \rightarrow B \;:\; \{n, m\}_{Key(A,B)}$$

$$Resp(B, A) \stackrel{def}{=} \text{in pat}(x)\ x\ .\ \text{out new}(y,z)\ \{x, y, B, Key(z)\}_{Key(A,B)}\ .\ \text{in}\ \{x, y\}_{Key(A,B)}$$

$$RESP \stackrel{def}{=} \Big\|_{B \in Agents} \Big\|_{A \in Agents} !\ Resp(B,A)$$

$$ISO \stackrel{def}{=} \Big\|_{p \in \{INIT, RESP, \mathbf{SPY}\}} p$$

Transition Semantics

- output: provided $n \notin s$

$$\langle \text{out new}(x)\ M\ .\ p, s, t\rangle \xrightarrow{\text{out new}(n)\ M[n/x]} \langle p[n/x], s \cup \{n\}, t \cup \{M[n/x]\}\rangle$$

- input: provided $M[N/\psi] \in t$

$$\langle \text{in pat}(\psi)\ M\ .\ p, s, t\rangle \xrightarrow{\text{in } M[N/\psi]} \langle p[N/\psi], s, t\rangle$$

- parallel composition:

$$\frac{\langle p_j, s, t\rangle \xrightarrow{\alpha} \langle p'_j, s', t'\rangle}{\langle \|_i\ p_j, s, t\rangle \xrightarrow{j:\alpha} \langle \|_i\ p'_j, s', t'\rangle}$$

where p'_i is p'_j for $i=j$, else p_i

Transitions & security properties

Secrecy of session key: For all runs where

$$\langle p_j, s_j, t_j \rangle \xrightarrow{\text{resp:B,A:i:out new(m,b) } \{n, m, B, Key(b)\}_{Key(A,B)}} \langle p_{j+1}, s_{j+1}, t_{j+1} \rangle$$

$$Key(A,B) \not\in t_0 \Rightarrow \forall \text{ stage } w \,.\, Key(b) \not\in t_w$$

Possible proof strategy:

- assume does not hold => exists earliest violating action
- derive contradiction from causally preceding events ?

Transition semantics masks local dependencies !

Petri nets with persistence

Def: Petri net with persistent conditions consists of

- B set of conditions,
- $P \subseteq B$ persistent conditions,
- E set of events,
- pre,post: $E \rightarrow \mathrm{Pow}(B)$ pre and postcondition maps.

Def: Token game:

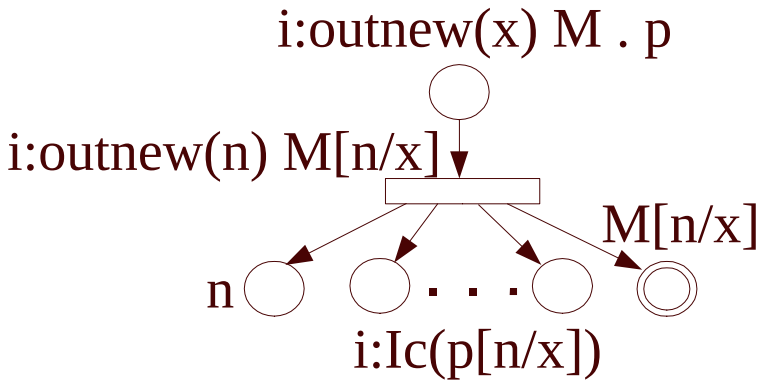
$$\boldsymbol{M} \xrightarrow{e} \boldsymbol{M'} \quad \text{iff } {}^\bullet e \subseteq \boldsymbol{M} \;\&\; (\boldsymbol{M} \setminus ({}^\bullet e \cup P)) \cap e^\bullet = \emptyset$$

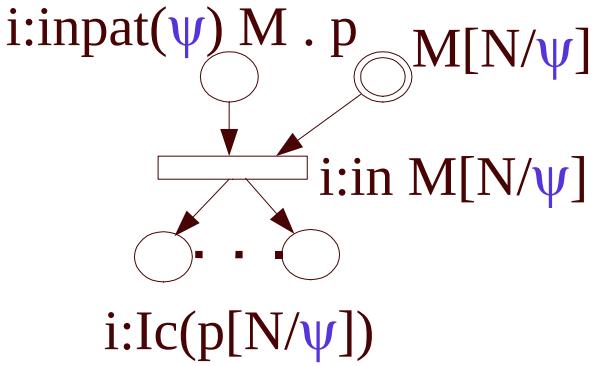
where $\boldsymbol{M'} = (\boldsymbol{M} \setminus {}^\bullet e) \cup e^\bullet \cup (\boldsymbol{M} \cap P)$

Event Semantics

SPL Petri net

- conditions **C** ∪ **N** ∪ **O**
  - **C**: control
  - **N**: names
  - **O**: output (persistent)
- events (with pre- and postcondition maps)



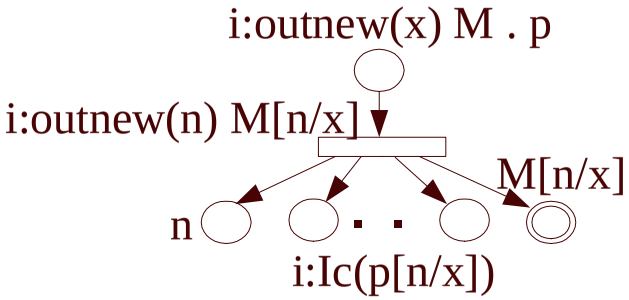


- events can carry indices to identify component

Net of an SPL process

Ev(out new(x) M. p) =

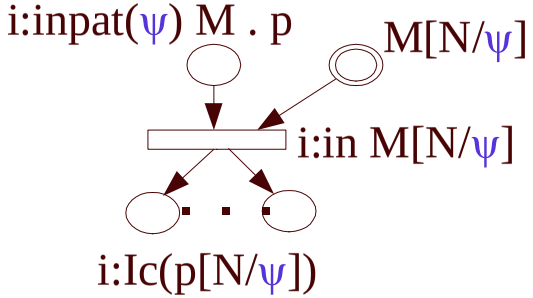
$\cup_n$ Ev(p[n/x]) $\cup$ {



| n names }

Ev (in pat (ψ) M. p) =

$\cup_M$ Ev([M/ψ]) $\cup$ {



| M messages}

Ev ($\|_{i\in I}$ p_i) = $\cup_{i\in I}$ i: Ev(p_i)

Relating transition and event semantics

- *Th*: If $\langle p, s, t\rangle \xrightarrow{\alpha} \langle p', s', t'\rangle$

  then $Ic(p) \cup s \cup t \xrightarrow{e} Ic(p') \cup s' \cup t'$

  for some event e with $act(e) = \alpha$.

- *Th*: If $Ic(p) \cup s \cup t \xrightarrow{e} \boldsymbol{M}$

  then $\langle p, s, t\rangle \xrightarrow{act(e)} \langle p', s', t'\rangle$ and $\boldsymbol{M} = Ic(p') \cup s' \cup t'$

  for some closed process term p', names s' and messages t'.

Protocol verification – proof strategy

Use event-based semantics of SPL:

- formalize security property **P** in terms of events (as safety property),
- assume the run contains event violating **P** (take first such event),
- use dependencies among events & derive contradiction (case analysis on the events of a protocol).

Derived proof principles

- *Well foundedness*: in a protocol run

  at some stage $\neg P \Rightarrow \exists$ first stage s.t. $\neg P$

- *Freshness* of m in a run:

  at most one event s.t. $m \in e^n$

- *Precedence*:

  control:

  if $b \in {}^c e_i$ either $b \in Ic(p_0)$ or $\exists e_j, j<i$ s.t. $b \in e^c_j$

  output input:

  if $M \in {}^o e_i$ either $M \in t_0$ or $\exists e_j, j<i$ s.t. $M \in e^o_j$

Summary (I)

- Event based semantics of SPL

  => *non interleaving models* useful for security-protocol analysis.

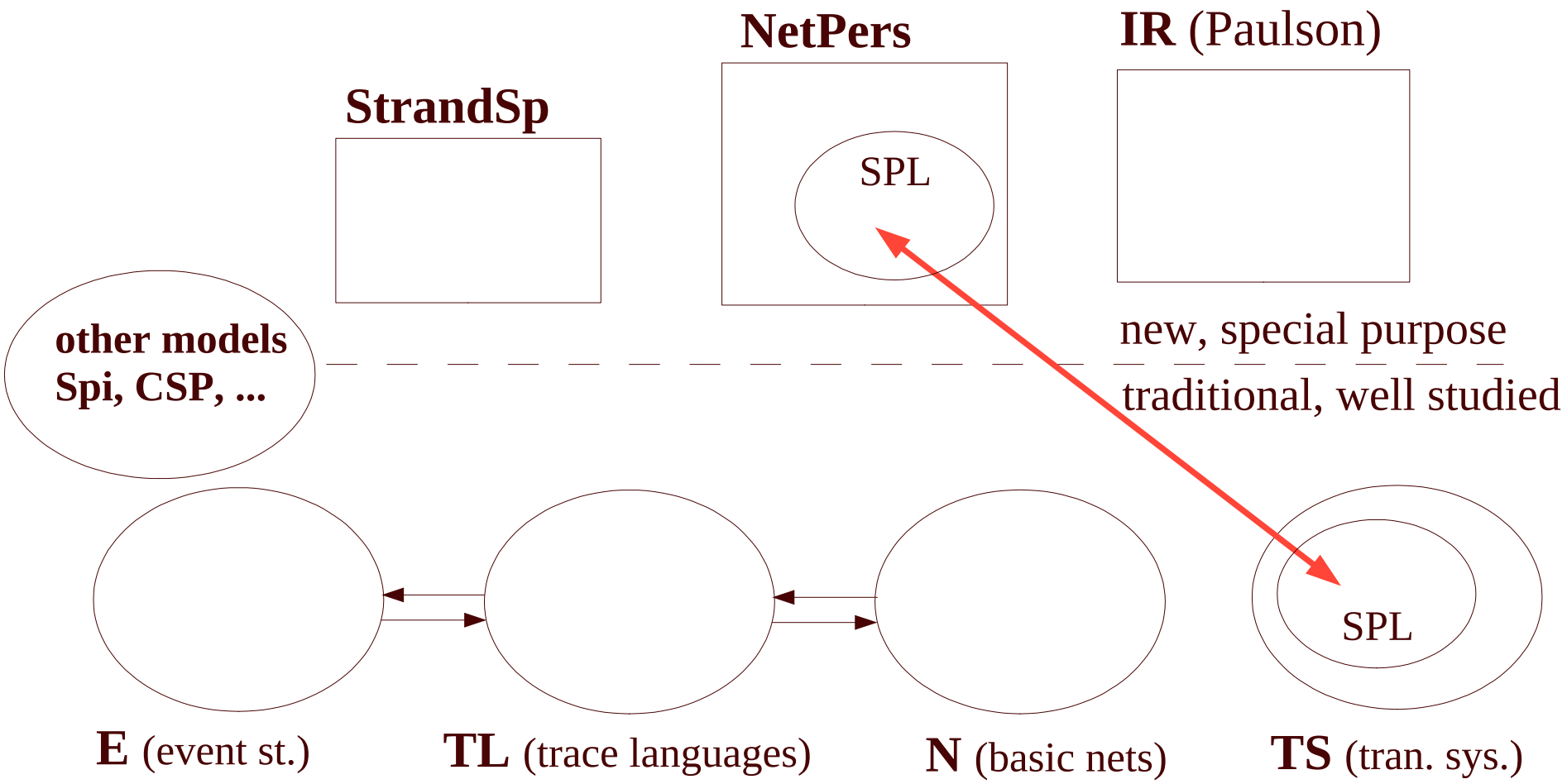
- Transition semantics of SPL *easy to implement*.

- Relation between event-based & transition sem.

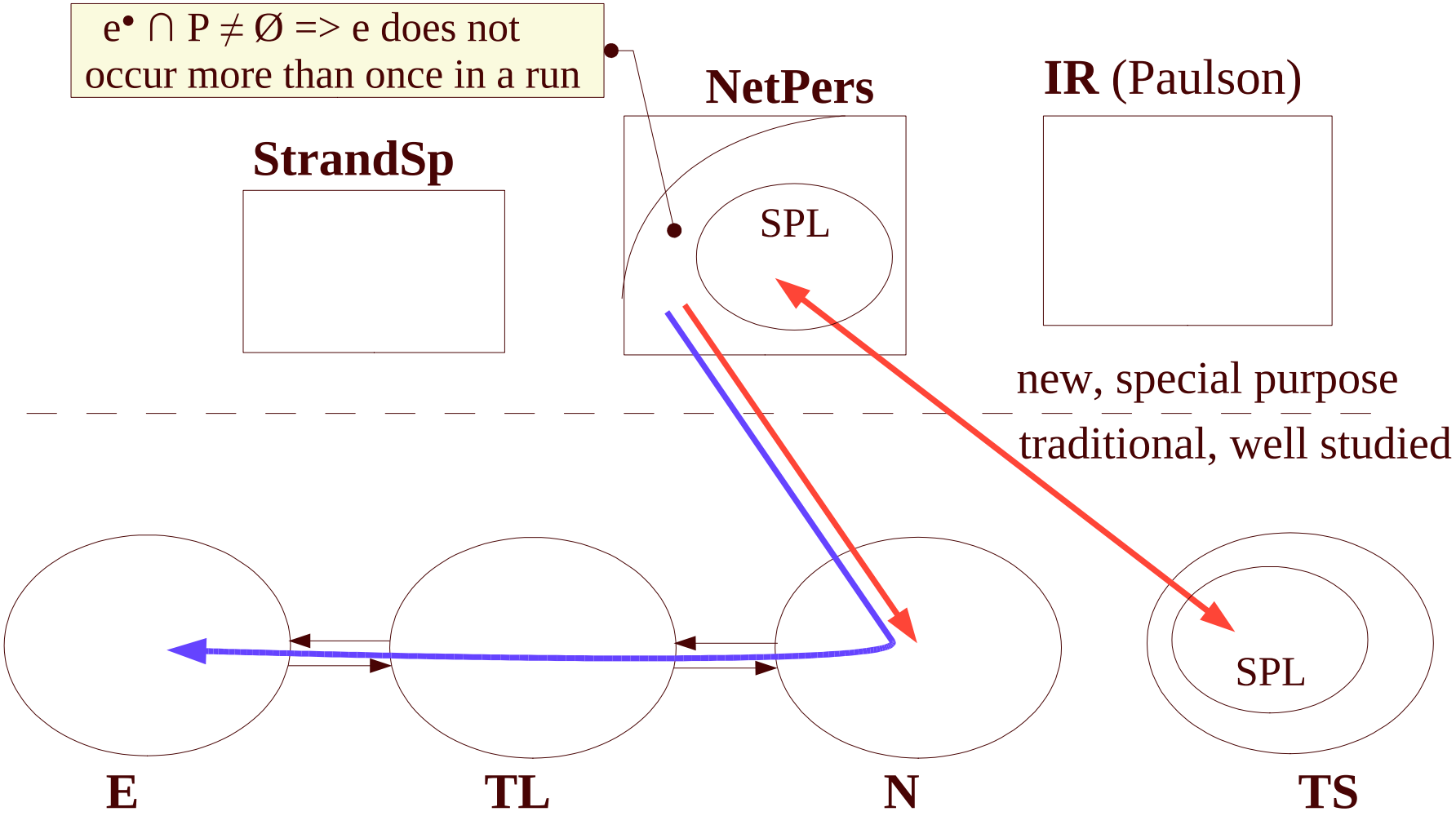
  + correct impl. of transition sem.

  => properties of *protocol model* are properties of *protocol implementation*.

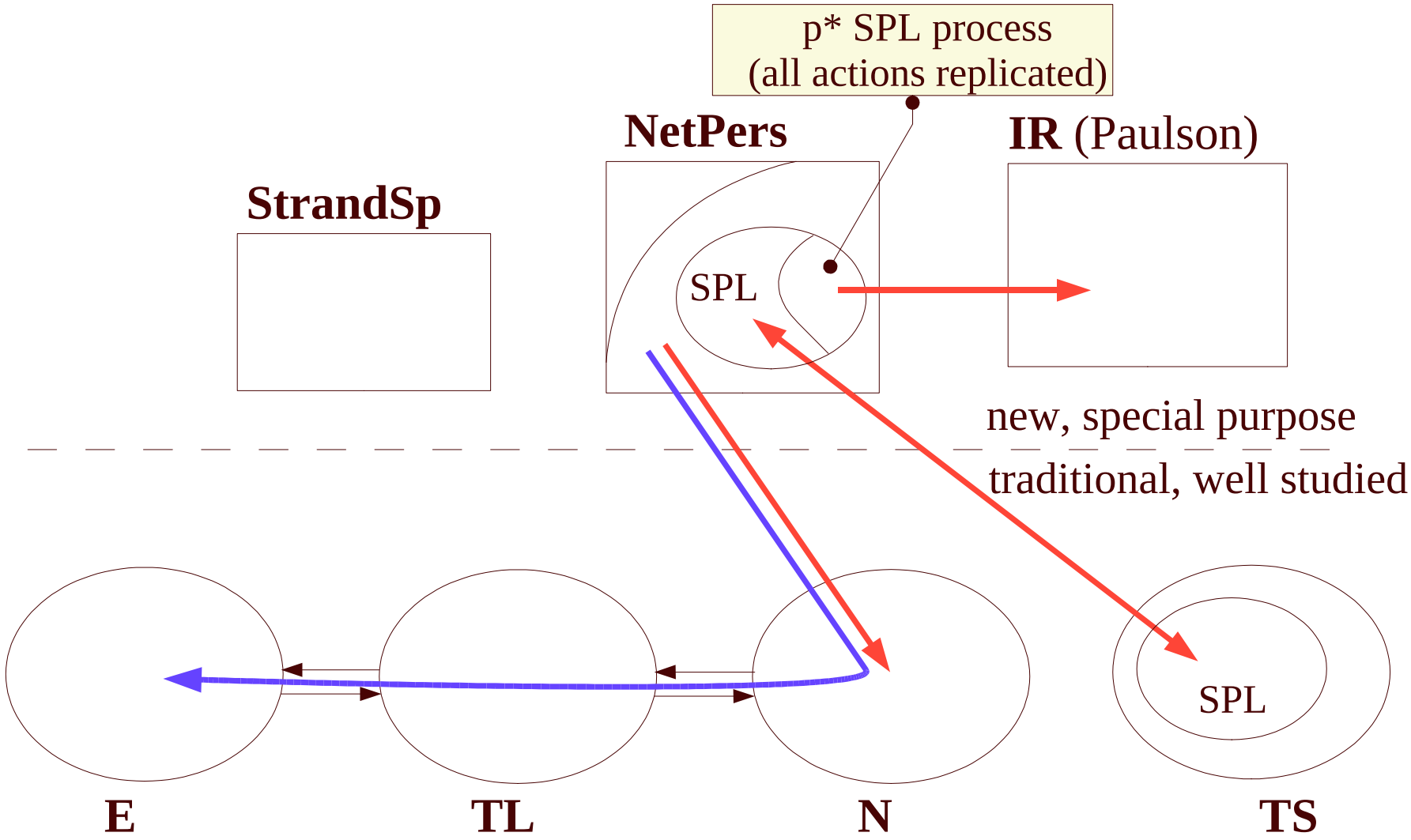
Relation between models
(relate finite behaviours)



SPL Nets, Trace Languages, Event Structures



SPL and Inductive Rules



Strand Spaces with conflict

Strand Spaces: $\langle \mathbf{s}_i \rangle_{i \in I}$

- only limited form of nondeterminism
- difficult to compose using traditional process op.

Extension: $\mathbf{(\langle s_i \rangle_{i \in I}, \#)}$

- $\# \in \mathbf{I \times I}$, symmetric & irreflexive (*conflict relation*)
- unique orig. on the bundles not on entire space

Compose Strand Spaces: $\mathbf{a.S}$, $\mathbf{S \parallel S'}$, $\mathbf{S+S'}$

(abbreviation $\|_{k \in \mathbb{N}} (\langle s_i \rangle_{i \in I}, \#) = !(\langle s_i \rangle_{i \in I}, \#)$)

Conflict relation is inessential

Def: $\approx$ binary, symmetric relation s.t. S $\approx$ S' iff
$\forall$b bundle of S => $\exists$b' bundle of S' s.t. b and b' are isomorphic graphs.

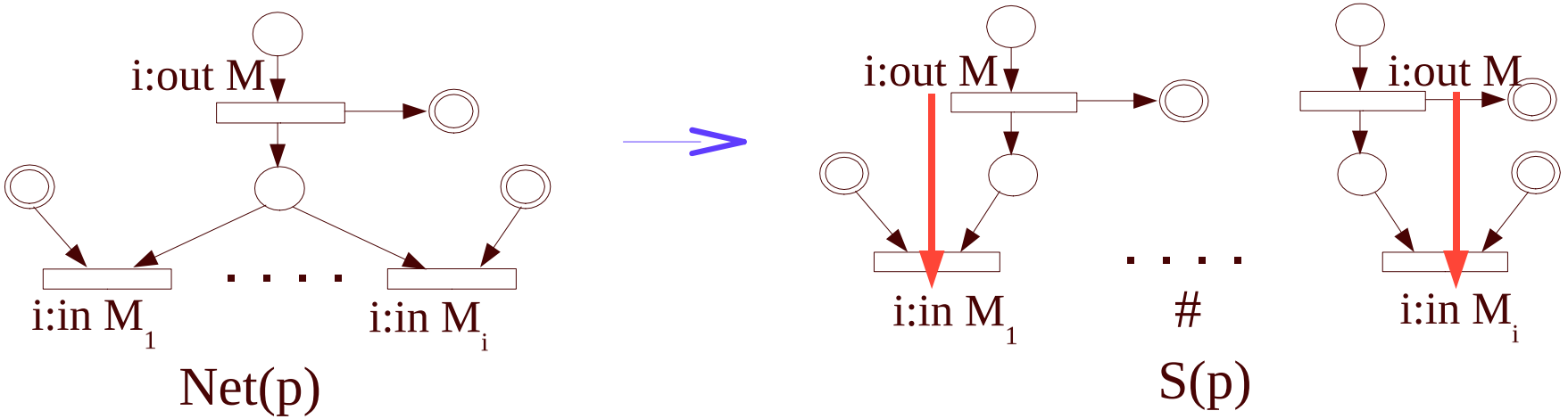
Th:

- b bundle of $!(\langle s_i \rangle_{i \in I}, \#)$ then b bundle of $!(\langle s_i \rangle_{i \in I}, \emptyset)$
- b bundle of $!(\langle s_i \rangle_{i \in I}, \emptyset)$ then $\exists$ re-indexing π s.t. π(b) bundle of $!(\langle s_i \rangle_{i \in I}, \#)$.

Cor: $\mathbf{!(\langle s_i \rangle_{i \in I}, \#) \approx !(\langle s_i \rangle_{i \in I}, \emptyset)}$

SPL and Strand Spaces

- max seq. in Ev(p) coinciding at control (p “par” process)



- *Th*: Seq. of events in Net(p) <=> lineariz. of bundle in S(p)
- if p is “!-par” process then **S(p) = !(<s_i>$_{i\in I}$,#) ≈ !(<s_i>$_{i\in I}$,∅)**

Prime Event Structures

- Prime Event Structure **(E,#,≤)**
  - binary conflict relation #, symmetric and irreflexive
  - {e' | e' ≤ e} finite
  - e#e'≤ e'' => e#e''

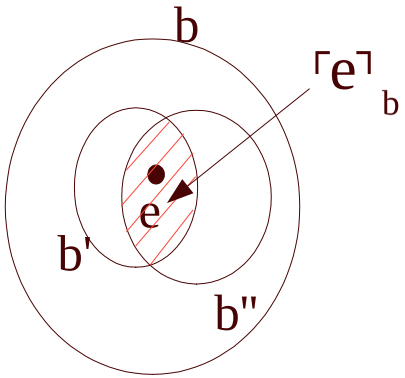
- configurations F(**E**) are x⊆**E** s.t.
  - x is conflict free
  - x is left closed (e' ≤ e ∈ x => e' ∈ x)

Strand Spaces and Event Structures

- bundles are graphs, i.e. sets of nodes and edges $(\mathbf{B}, \subseteq)$

- $b \in \mathbf{B}$ bundle, $e \in E_b$,

$$\lceil e \rceil_b = \cap \{b' \in \mathbf{B} \mid e \in b' \text{ and } b' \subseteq b\} \quad \text{(primes)}$$



Prop:

- $\lceil e \rceil_b$ is a bundle
- if $b \in \mathbf{B}$ then $b = \cup \{p \mid p \subseteq b, p \text{ prime}\}$

Strand Spaces and Event Structures (II)

Def: $\Pr(\mathbf{B}) = (\mathbf{P}, \#, \subseteq)$

- **P** the primes of **B**
- p#p' if $\neg\exists$ prime p'' s.t. $p \subseteq p''$ and $p' \subseteq p''$ (p,p' not compatible)

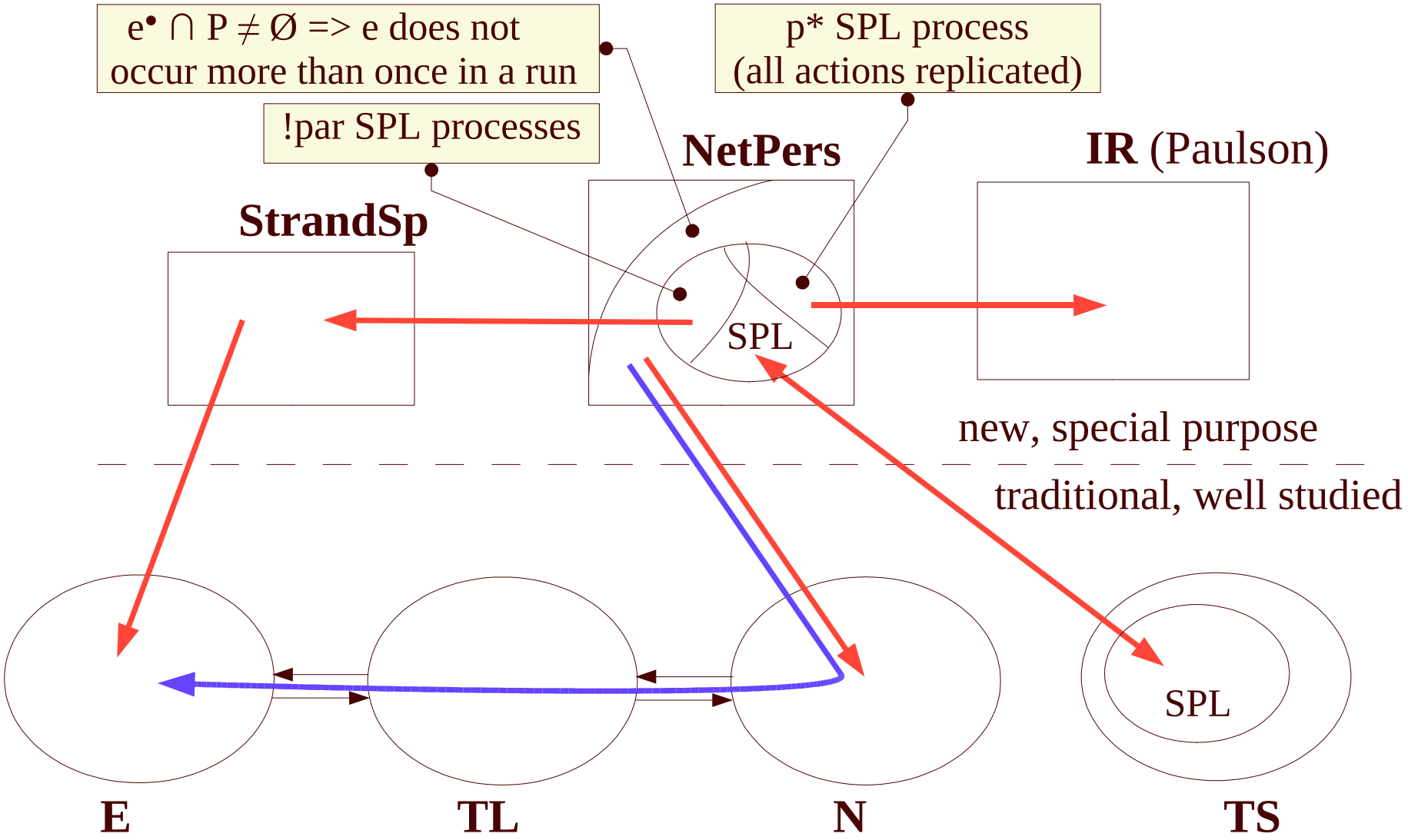
Th: $\Pr(\mathbf{B})$ is a prime event structure &

$$\Phi : (\mathbf{B}, \subseteq) \cong (F^{fin}\ \Pr(\mathbf{B}), \subseteq)$$

where

- $\Phi(b) = \{p \mid p \subseteq b,\ p \text{ prime}\}$ iso of partial orders with
- inverse $\Theta : F^{fin}\ \Pr(\mathbf{B}) \rightarrow \mathbf{B}$ where $\Theta(x) = \cup x$.

Summary (II)


e• ∩ P ≠ Ø => e does not occur more than once in a run
p* SPL process (all actions replicated)
!par SPL processes
NetPers
IR (Paulson)
StrandSp
SPL
new, special purpose
traditional, well studied
SPL
E
TL
N
TS

References

- F. Crazzolara. *Language, Semantics, and Methods for Security Protocols.* Ph.D. Thesis, BRICS, May 2003.
- F. Crazzolara, G. Winskel. *Composing Strand Spaces*. FSTTCS'02.
- F. Crazzolara, G. Winskel. *Events in Security Protocols*. ACM CCS'01.
- F. Crazzolara, G. Winskel. *Petri nets in Cryptographic Protocols*. FMPPTA'01.
- F. Crazzolara, G. Milicia. Implementation of SPL @ www.chispaces.com.