Non-Functional Aspects of Wide Area Network Programming

Emilio Tuosto

Ph.D. Thesis

Dipartimento di Informatica

Università di Pisa

Plan of the talk

Plan of the talk

- WAN programming: A short overview

Plan of the talk

- WAN programming: A short overview
- Declarative programming model: Hypergraphs
  - Hypergraphs and Ambient calculus

Plan of the talk

- WAN programming: A short overview
- Declarative programming model: Hypergraphs
  - Hypergraphs and Ambient calculus
- Programming QoS: Qlaim
  - QoS & Hypergraphs: reasoning on optimal routing

Plan of the talk

- WAN programming: A short overview
- Declarative programming model: Hypergraphs
  - Hypergraphs and Ambient calculus
- Programming QoS: Qlaim
  - QoS & Hypergraphs: reasoning on optimal routing
- Hypergraphs and UML specifications

Plan of the talk

- WAN programming: A short overview
- Declarative programming model: Hypergraphs
  - Hypergraphs and Ambient calculus
- Programming QoS: Qlaim
  - QoS & Hypergraphs: reasoning on optimal routing
- Hypergraphs and UML specifications
- Cryptographic protocols: cIP and $\mathcal{PL}$

Plan of the talk

- WAN programming: A short overview
- Declarative programming model: Hypergraphs
  - Hypergraphs and Ambient calculus
- Programming QoS: Qlaim
  - QoS & Hypergraphs: reasoning on optimal routing
- Hypergraphs and UML specifications
- Cryptographic protocols: cIP and $\mathcal{PL}$
- The Mihda environment

Wide Area Network Programming Issues

- Absence of centralised control
- Administrative domains
- Interoperability
- “Mobility” (of resources and computation)
- *Network Awareness*
- Service Level Agreement
- Security
- ...

Web Services: A programming metaphor

- Applications access *services* that must be
  - Published
  - Searched
  - Binded
- Services are
  - “Autonomous”
  - Independent (local choices, independently built)
  - Mobile/stationary
  - “Interconnected”
- Security issues: hostile environment

WAN Foundations

π-calculus [MPW92] (very basic wrt WAN)

- Klaim [DFP98, DFPV00, BLP02]
- Ambient [CG00]
- Dπ [HR98, HR00]
- Djoin [FG96, FGL^{+}96]
- Seal [VC98]
- ...

A Model for Declarative WAN Programming

Hypergraphs Programming model

Hypergraphs Programming model

- Client-Server metaphor is not enough: P2P

Hypergraphs Programming model

- Client-Server metaphor is not enough: P2P
- Mobility and dynamic linking of components

Hypergraphs Programming model

- Client-Server metaphor is not enough: P2P
- Mobility and dynamic linking of components
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)

Hypergraphs Programming model

- Client-Server metaphor is not enough: P2P
- Mobility and dynamic linking of components
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness
  - Applications are location dependent
  - Locations have different features
  - and allow multiple (security) policies

Hypergraphs Programming model

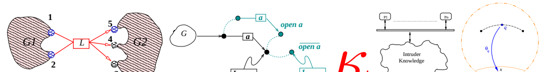
- Client-Server metaphor is not enough: P2P
- Mobility and dynamic linking of components
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness
  - Applications are location dependent
  - Locations have different features
  - and allow multiple (security) policies
- Independently programmed in a distributed environment

Hypergraphs Programming model

- Client-Server metaphor is not enough: P2P
- Mobility and dynamic linking of components
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness
  - Applications are location dependent
  - Locations have different features
  - and allow multiple (security) policies
- Independently programmed in a distributed environment
- Reasoning on space and time

Hypergraphs Programming model[2]

- Graphs for distributed systems [CM83]
- Edge replacement for graph rewritings [DM87]
- Edge replacement/distributed constraint solving problem [MR96]
- Graphs grammars for software architecture styles [HIM00]
- Synchronised Hyperedge Replacement (SHR) with mobility for name passing calculi [HM01]



Hypergraphs Programming model[3]

We aim at tackling new *non-functional* computational phenomena of systems using SHR.
The metaphor is

- “WAN systems *as* Hypergraphs”
- “WAN computations *as* SHR”

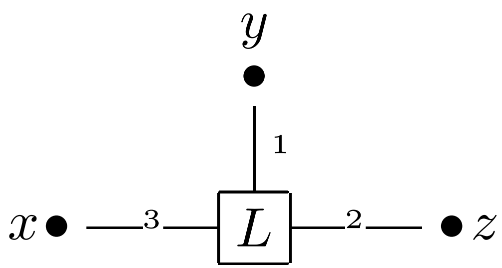
In other words:

- Components are represented by hyperedges
- Systems are *bunches* of (connected) hyperedges
- Computing means to rewrite hyperedge...
- ...according to a synchronisation policy

Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

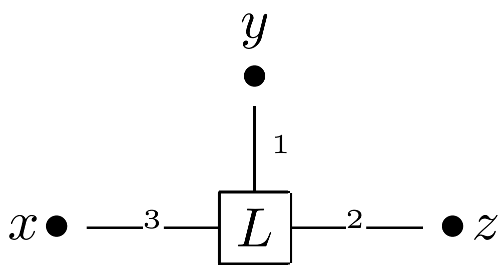
$$L : 3, \quad L(y, z, x),$$



Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

$L : 3, \quad L(y, z, x),$

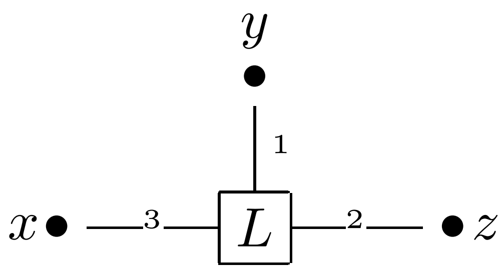


$$
\begin{array}{rcl}
G & ::= & nil \mid \nu\, y.G \\
 & \mid & L(\vec{x}) \mid G|G
\end{array}
$$

Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

$$L : 3, \quad L(y, z, x),$$



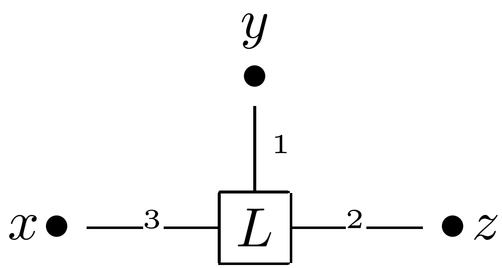
$$G \quad ::= \quad nil \mid \nu\, y.G \mid L(\vec{x}) \mid G|G$$

Syntactic Judgement $\qquad \Gamma \vdash G, \qquad fn(G) \subseteq \Gamma$

Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

$$L : 3, \quad L(y, z, x),$$



$$G \;::=\; nil \;\mid\; \nu\, y.G \;\mid\; L(\vec{x}) \;\mid\; G|G$$

Syntactic Judgement $\qquad \Gamma \vdash G, \qquad fn(G) \subseteq \Gamma$

An example:

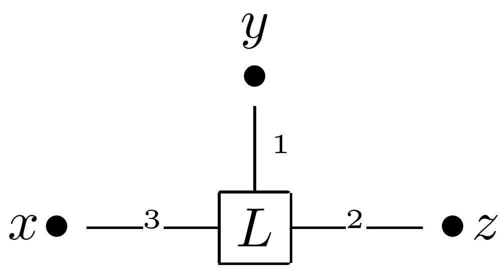
$$L : 3, \quad M : 2$$

$$x, y \vdash \nu\, z.(L(y, z, x) | M(y, z))$$

Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

$L : 3, \quad L(y, z, x),$



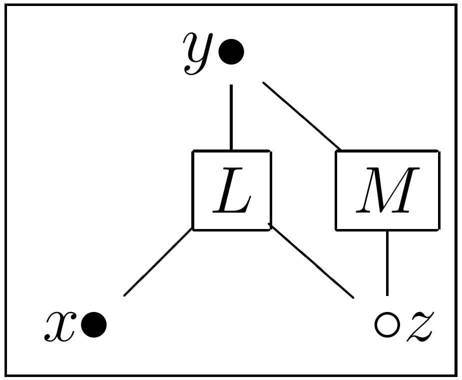
$$G \; ::= \; nil \mid \nu\, y.G \mid L(\vec{x}) \mid G|G$$

Syntactic Judgement $\quad \Gamma \vdash G, \quad fn(G) \subseteq \Gamma$

An example:

$L : 3, \quad M : 2$

$x, y \vdash \nu\, z.(L(y, z, x) | M(y, z))$

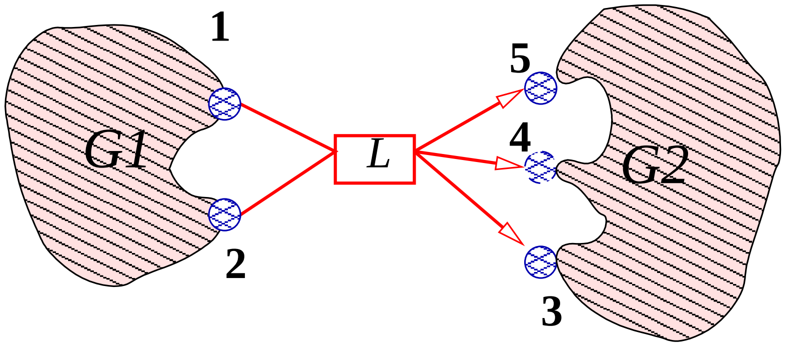


Replacement of Hyperedges

$L \to G$

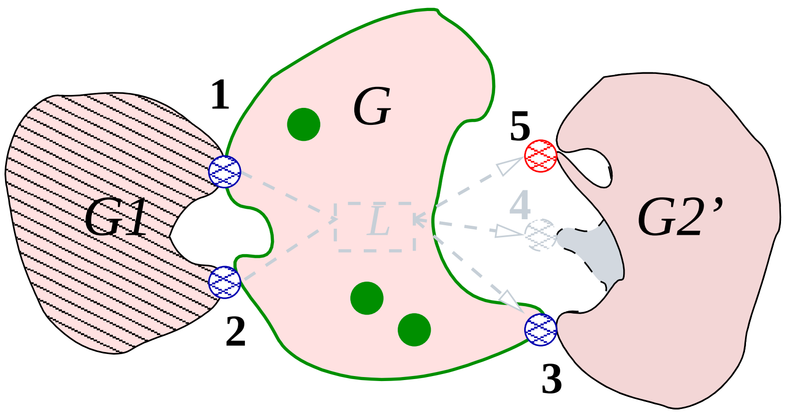
Replacement of Hyperedges

$L \longrightarrow G$



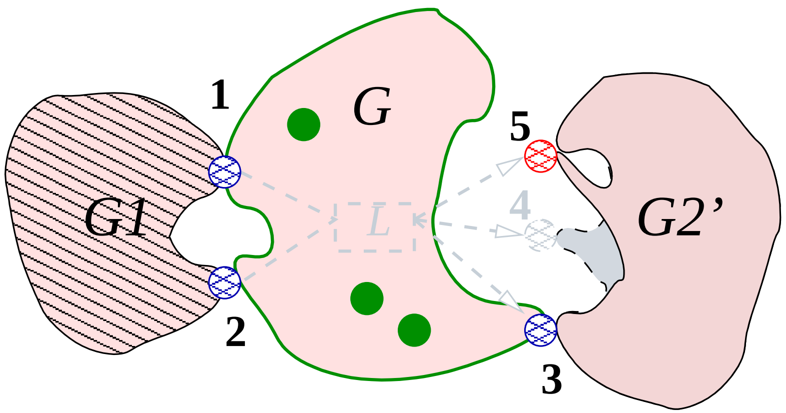
Replacement of Hyperedges

$L \to G$

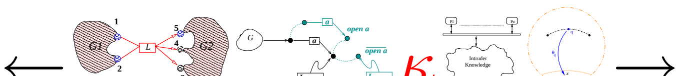


Replacement of Hyperedges

$L \rightarrow G$

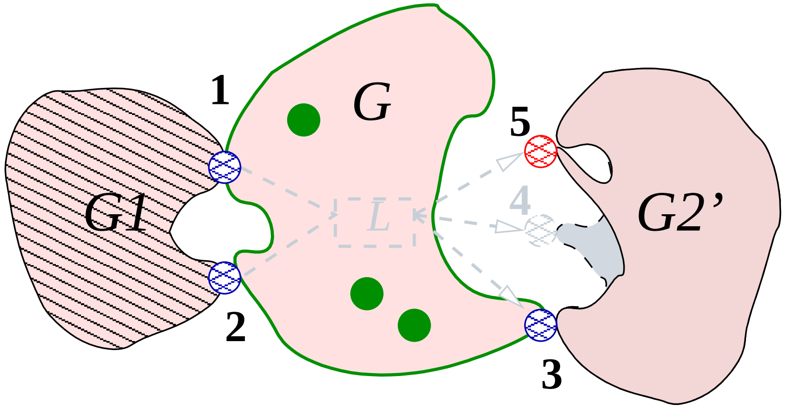


- Edge replacement: local
- Synchronisation as distributed constraint solving
- New node creation
- Node fusion: mobility model



Replacement of Hyperedges

$L \rightarrow G$



- Edge replacement: local
- Synchronisation as distributed constraint solving
- New node creation
- Node fusion: mobility model

Benefi ts:

- Powerful model of system composition (π, π-I, fusion)
- LTS for Ambient ...
- ...and for Klaim
- and *path reservation* for Qlaim

Hypergraph Semantics: Productions

$$\underbrace{x_1, \ldots, x_n}_{X} \vdash L(x_1, \ldots, x_n) \xrightarrow[\pi]{\Lambda} \Gamma \vdash G,$$

- $\Lambda \subseteq X \times Act \times \mathcal{N}^*$ set of constraints
- $\pi : X \to X$ fusion substitution, i.e.

$$\forall x_i, x_j \in X . \pi(x_i) = x_j \Rightarrow \pi(x_j) = x_j$$

- $\Gamma = \pi(X) \cup (\mathrm{n}(\Lambda) \setminus X)$
- $\mathrm{fn}(G) \subseteq \Gamma$

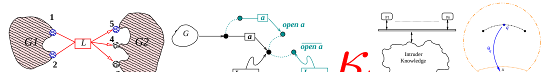
Hypergraph Semantics: Transitions

$$\Gamma_1 \vdash G_1 \xrightarrow[\pi]{\Lambda} \Gamma_2 \vdash G_2$$

Hypergraph Semantics: Transitions

$$\frac{\Gamma, y \vdash G \xrightarrow[\pi]{\Lambda} \Gamma' \vdash G' \qquad \Lambda(y)\uparrow \qquad x \simeq_\pi y \Rightarrow y \neq \pi(y) \qquad \rho = [^{\pi(x)}/_{\pi(y)}]}{\Gamma \vdash [^x/_y]G \xrightarrow[(\pi;\rho)_{-y}]{\rho\Lambda} \mathrm{n}(\rho\Lambda) \cup (\pi;\rho)_{-y}(\Gamma) \vdash \rho G'}$$

$$\frac{\Gamma, y \vdash G \xrightarrow[\pi]{\Lambda \cup \{(x,a,\vec{v}),(y,\overline{a},\vec{w})\}} \Gamma' \vdash G' \qquad x \simeq_\pi y \Rightarrow y \neq \pi(y) \qquad \rho = mgu\{[^{[^x/_y]\vec{w}}/_{[^x/_y]\vec{v}}], [^{\pi(x)}/_{\pi(y)}]\} \qquad \Gamma'' = \mathrm{n}(\rho\Lambda) \cup (\pi;\rho)_{-y}(\Gamma) \qquad U = \rho(\Gamma') \setminus \Gamma''}{\Gamma \vdash [^x/_y]G \xrightarrow[(\pi;\rho)_{-y}]{(\rho\Lambda \cup (x,\tau,\langle\rangle))} \Gamma'' \vdash \nu\, U.\rho G'}$$



Hypergraph Semantics: Transitions

$$\frac{\Gamma, y \vdash G \xrightarrow[\pi]{\Lambda} \Gamma' \vdash G' \qquad \Lambda(y)\uparrow \vee \Lambda(y) = (\tau, \langle\rangle) \qquad x \simeq_\pi y \Rightarrow y \neq \pi(y) \qquad U = \Gamma' \setminus (\mathrm{n}(\Lambda) \cup \pi_{-y}(\Gamma))}{\Gamma \vdash \nu\, y.G \xrightarrow[\pi_{-y}]{\Lambda \setminus (y,\tau,\langle\rangle)} \mathrm{n}(\Lambda) \cup \pi_{-y}(\Gamma) \vdash \nu\, U.G'}$$

$$\frac{\Gamma_1 \vdash G_1 \xrightarrow[\pi]{\Lambda} \Gamma_2 \vdash G_2 \qquad \Gamma'_1 \vdash G'_1 \xrightarrow[\pi']{\Lambda'} \Gamma'_2 \vdash G'_2 \qquad \Gamma_1 \cap \Gamma'_1 = \emptyset}{\Gamma_1 \cup \Gamma'_1 \vdash G_1 \mid G'_1 \xrightarrow[\pi \cup \pi']{\Lambda \cup \Lambda'} \Gamma_2 \cup \Gamma'_2 \vdash G_2 \mid G'_2}$$

Applying the Model

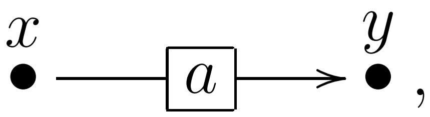
Ambient $a[...] \,|\, open\ a \;\rightarrow\; ...$

Applying the Model

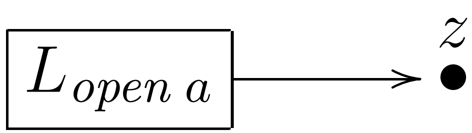
Ambient $a[...] | open\ a \ \rightarrow \ ...$

Components

$a[\cdots]$



$open\ a$

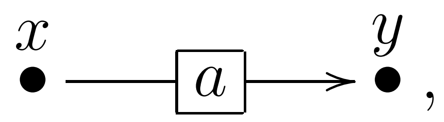


Applying the Model

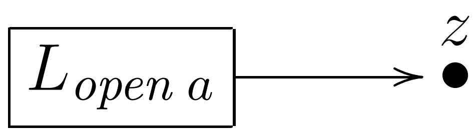
Ambient $a[...] | open\ a \ \rightarrow \ ...$

Components

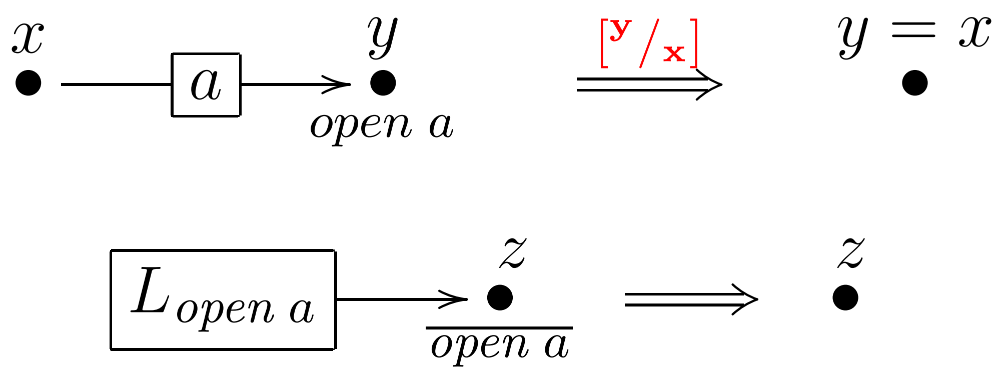
$a[\cdots]$



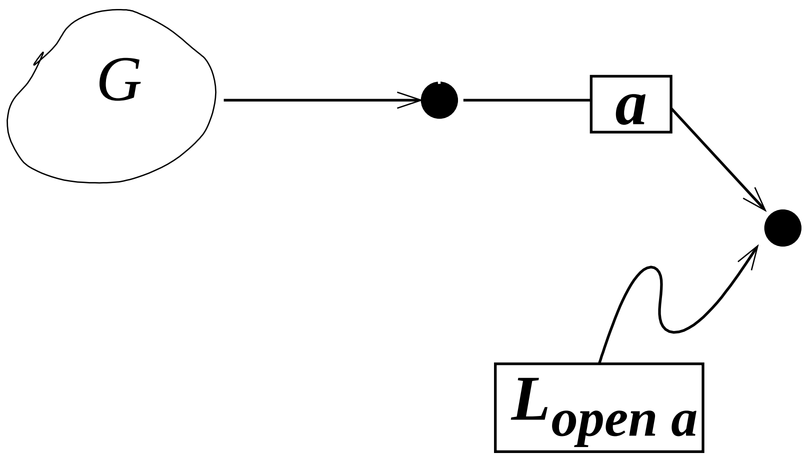
$open\ a$



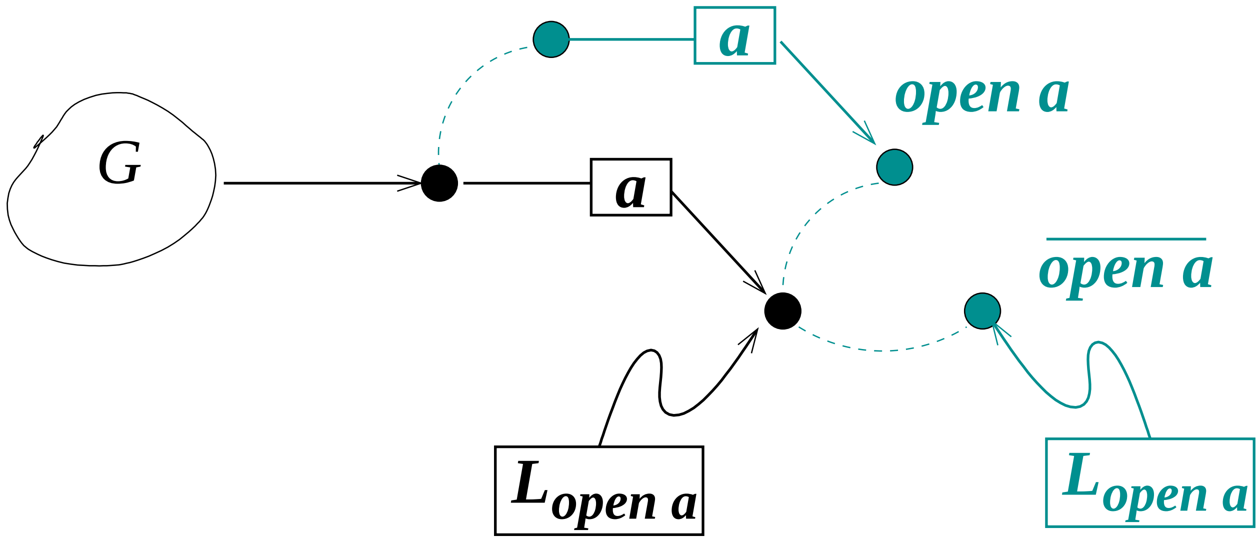
Productions



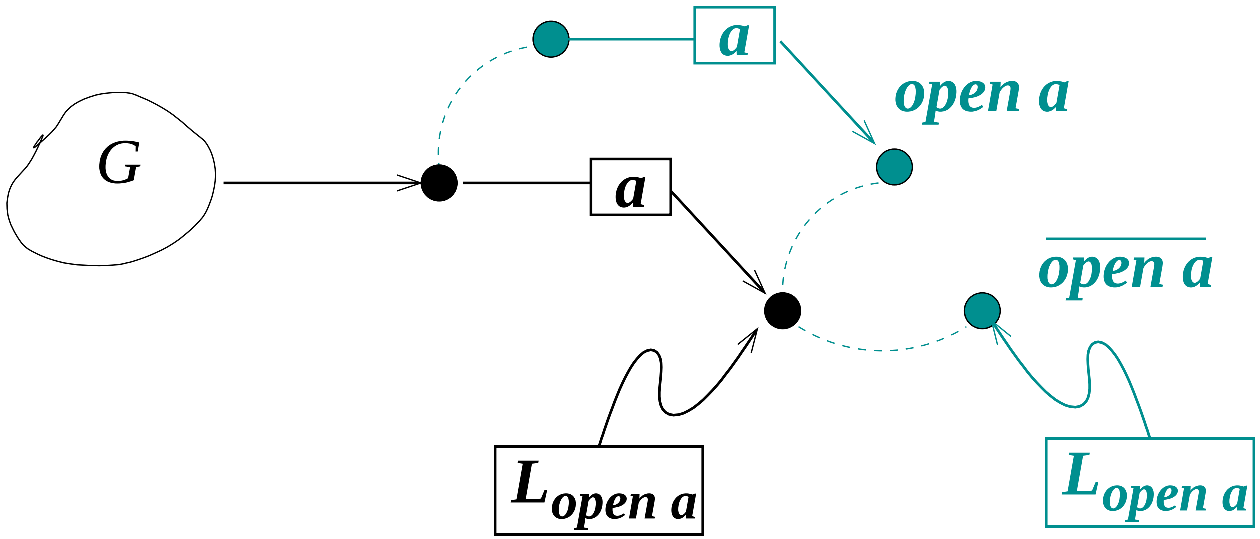
Applying the Model: Node Fusion

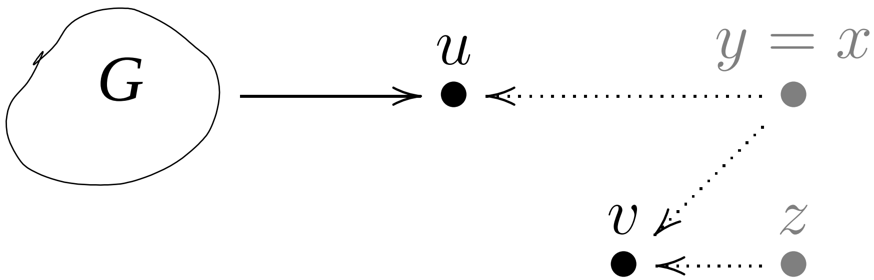


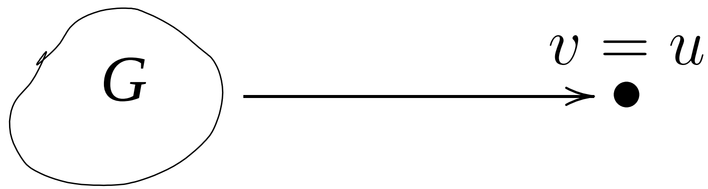
Applying the Model: Node Fusion



Applying the Model: Node Fusion



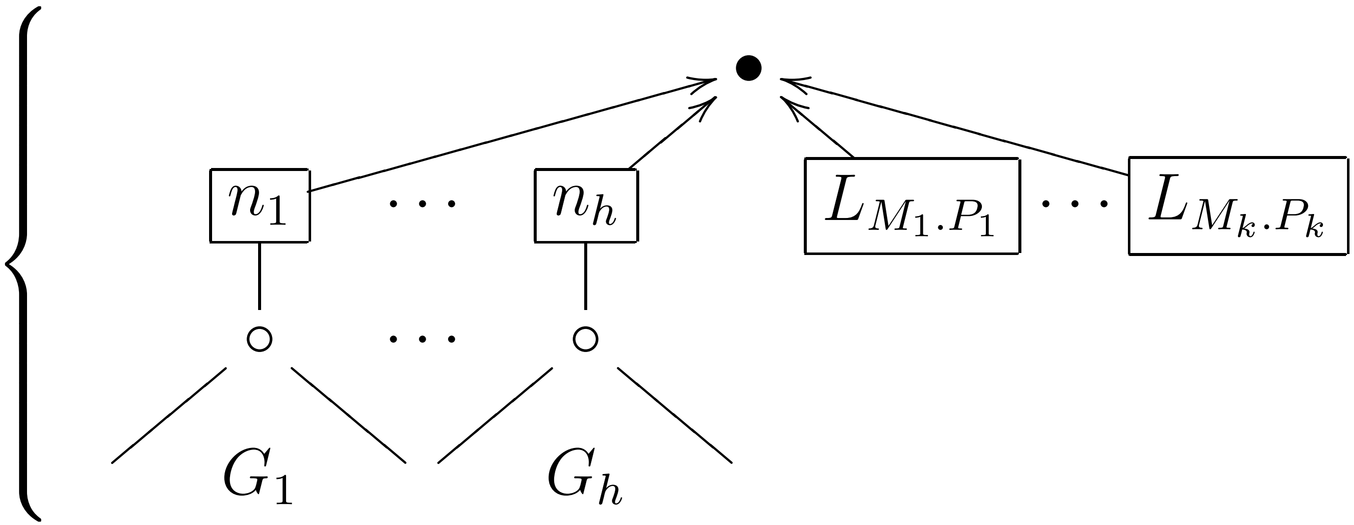




Graphs and Ambient

$$
\begin{aligned}
[\![\, \mathbf{nil} \,]\!]_x &= x \vdash nil \\
[\![\, n[P] \,]\!]_x &= x \vdash \nu\, y.(G \mid n(y,x)), \quad \text{if } \; y \neq x \wedge [\![\, P \,]\!]_y = y \vdash G \\
[\![\, M.P \,]\!]_x &= x \vdash L_{M.P}(x) \\
[\![\, P_1 | P_2 \,]\!]_x &= x \vdash G_1 \mid G_2, \quad \text{if } \; [\![\, P_i \,]\!]_x = x \vdash G_i \; \wedge \; i = 1,2 \\
[\![\, rec\, X.\, P \,]\!]_x &= [\![\, P[^{rec\, X.\, P}/_X] \,]\!]_x
\end{aligned}
$$

Ambient Graphs

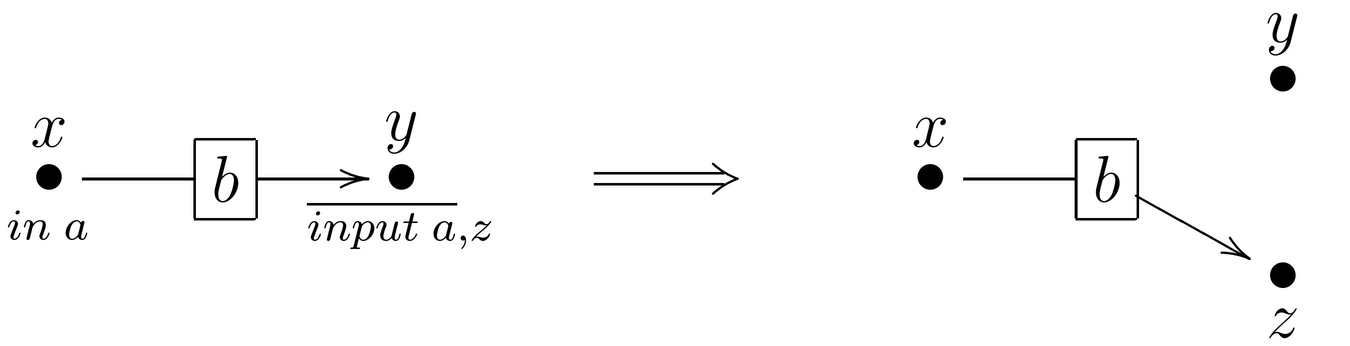


Theorem $[\![\, \_ \,]\!]_\_$ is a bijection on ambient graphs

Coordination Productions for Ambient

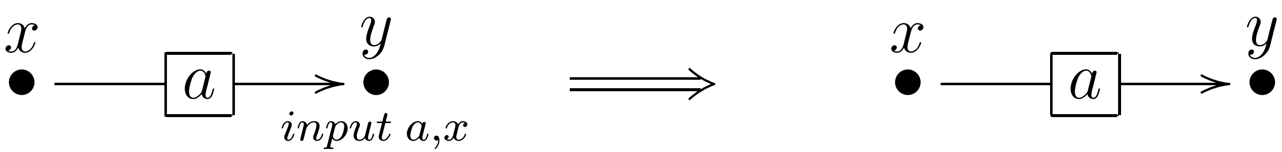
$$x, y \vdash b(x, y) \xrightarrow{\{(x, in\ a, \langle\rangle), (y, \overline{input\ a}, \langle z \rangle)\}} x, y, z \vdash b(x, z)$$

$(input1)$



$$x, y \vdash a(x, y) \xrightarrow{\{(y, input\ a, \langle x \rangle)\}} x, y \vdash a(x, y)$$

$(input2)$



Semantic Correspondence

Theorem If $P \rightarrow Q$ then $[\![P]\!]_x \xrightarrow[id]{\Lambda} [\![Q]\!]_x$ and

- either $\Lambda = \emptyset$
- or $\Lambda = \{(x, \tau, \langle\rangle)\}$

Semantic Correspondence

Theorem If $P \rightarrow Q$ then $[\![\ P\ ]\!]_x \xrightarrow[id]{\Lambda} [\![\ Q\ ]\!]_x$ and

- either $\Lambda = \emptyset$
- or $\Lambda = \{(x, \tau, \langle\rangle)\}$

Theorem If $[\![\ P\ ]\!]_x \xrightarrow[\pi]{\Lambda} \Gamma \vdash G$ is a basic transition, then

- either $[\![\ P\ ]\!]_x = \Gamma \vdash G$
- or $\exists Q \in Proc: \quad P \rightarrow Q \ \wedge \ \Gamma \vdash G = [\![\ Q\ ]\!]_x$

Klaim [DFP98]

Klaim [DFP98]

- Multiple TS

Klaim [DFP98]

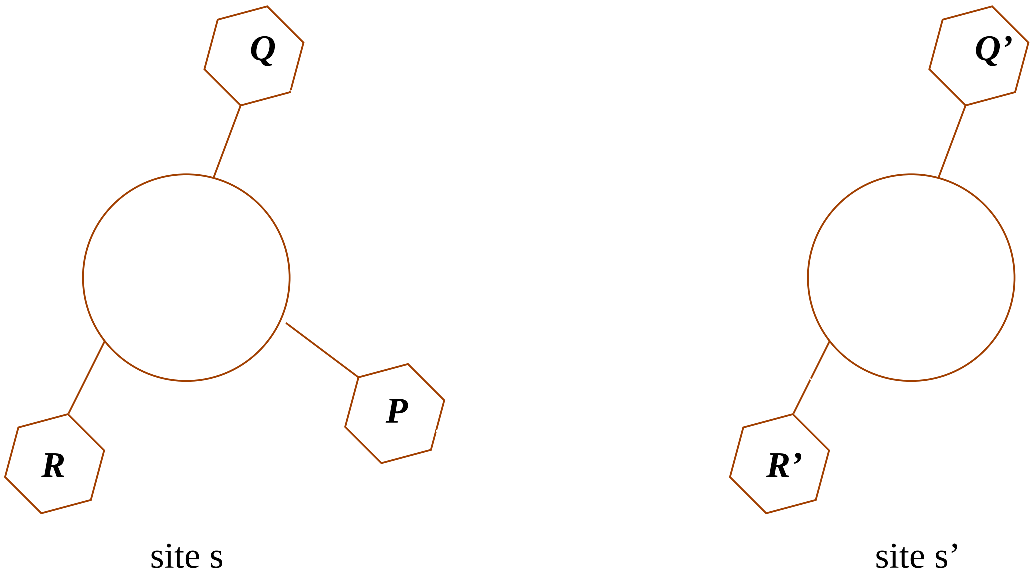
- Multiple TS
- Localities: first class citizens

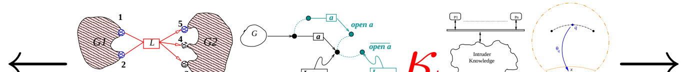
Klaim [DFP98]

- Multiple TS
- Localities: first class citizens
- Process migration

Klaim [DFP98]

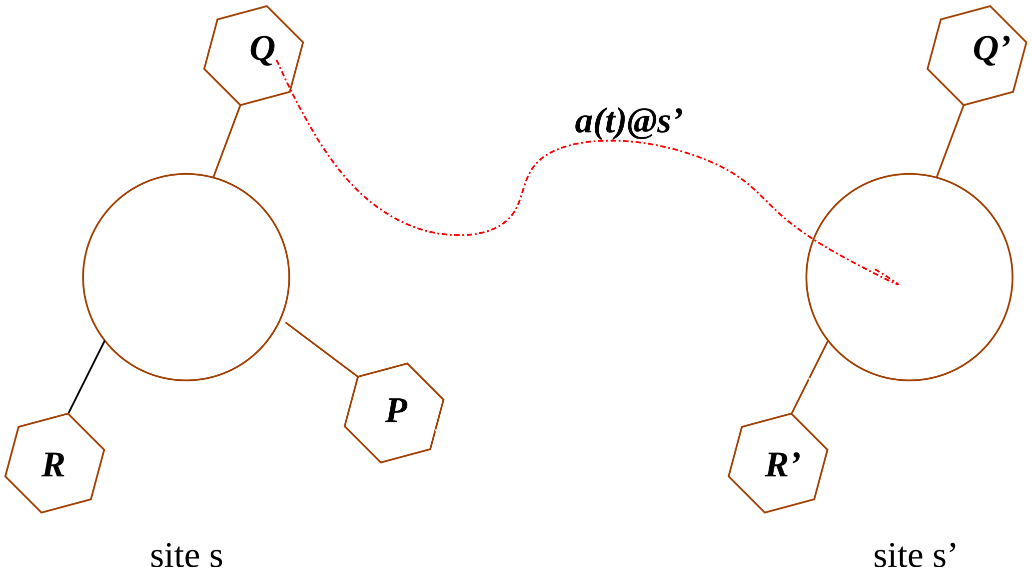
- Multiple TS
- Localities: first class citizens
- Process migration





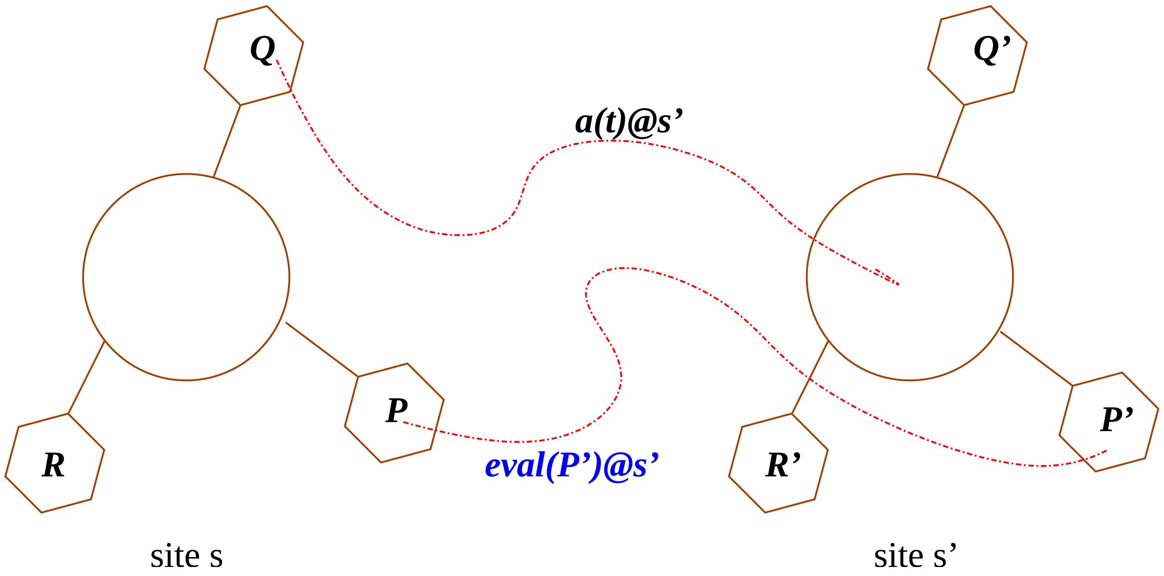
Klaim [DFP98]

- Multiple TS
- Localities: first class citizens
- Process migration



Klaim [DFP98]

- Multiple TS
- Localities: first class citizens
- Process migration



$$
\begin{array}{lcl}
P & ::= & \mathbf{nil} \\
 & | & \alpha.P \\
 & | & P_1 \mid P_2 \\
\alpha & ::= & a@s \\
a & ::= & \ldots \text{ // Klaim actions} \\
 & | & \mathbf{eval}(P)
\end{array}
$$

Qlaim: Gateways

In [BLP02]

Qlaim: Gateways

In [BLP02]

- Coordinators (super processes)

Qlaim: Gateways

In [BLP02]

- Coordinators (super processes)
- Dynamic creation of sites

Qlaim: Gateways

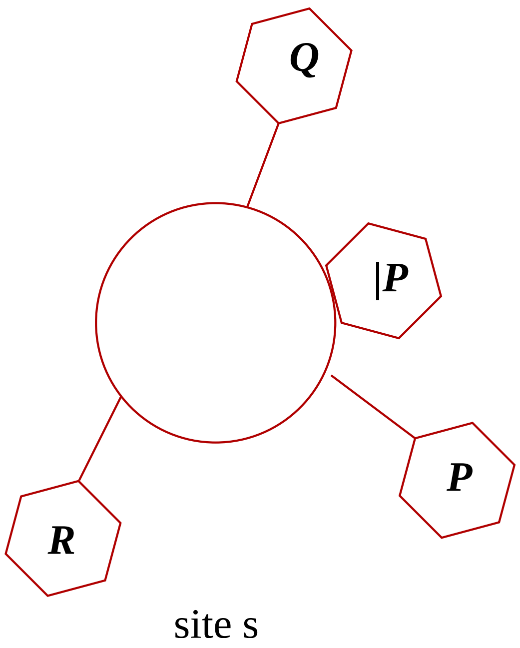
In [BLP02]

- Coordinators (super processes)
- Dynamic creation of sites
- Gateway connection management

Qlaim: Gateways

In [BLP02]

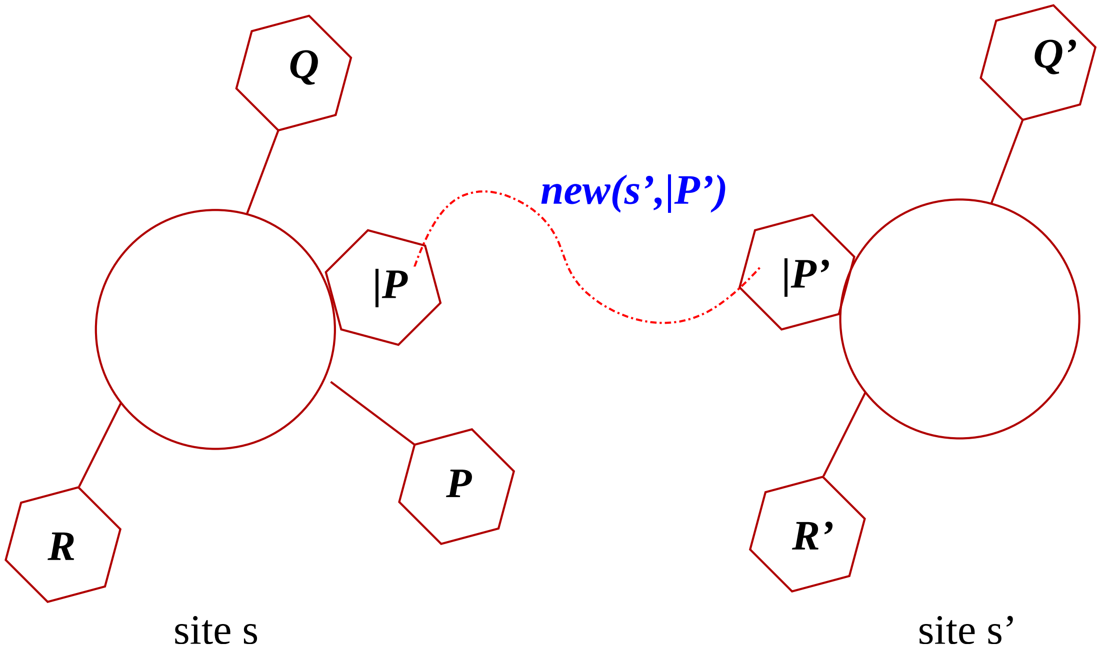
- Coordinators (super processes)
- Dynamic creation of sites
- Gateway connection management



Qlaim: Gateways

In [BLP02]

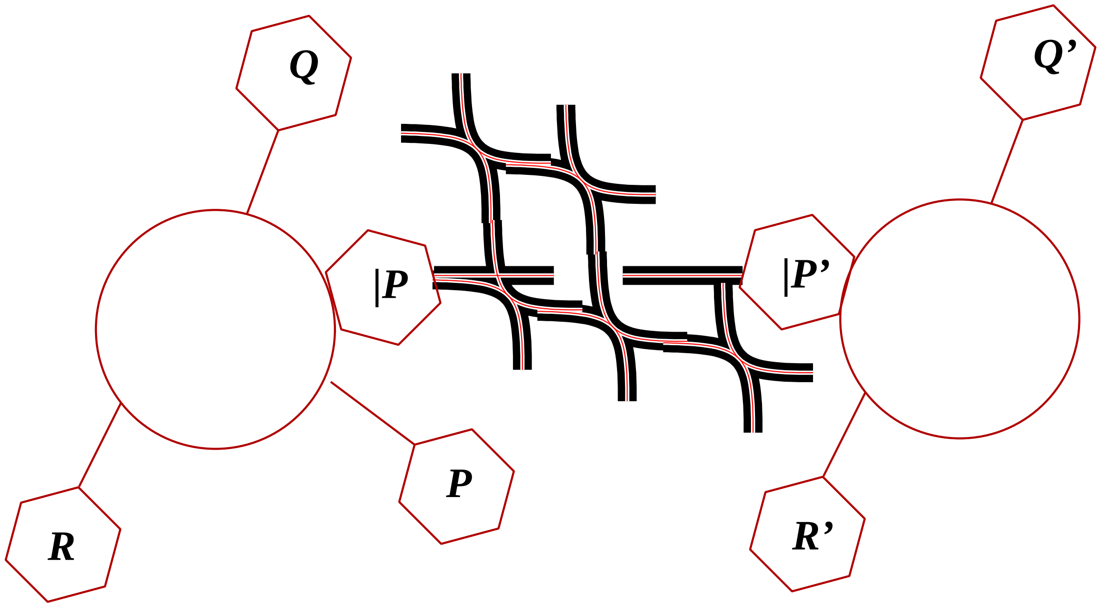
- Coordinators (super processes)
- Dynamic creation of sites
- Gateway connection management



Qlaim: Gateways

In [BLP02]

- Coordinators (super processes)
- Dynamic creation of sites
- Gateway connection management



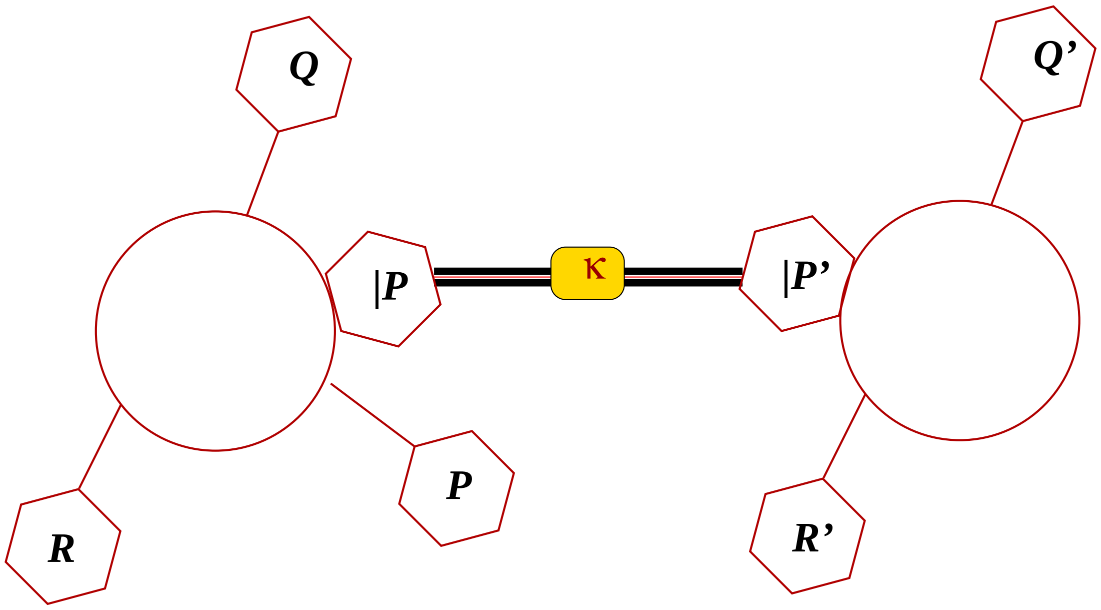
site s

site s’

Qlaim: Gateways

In [BLP02]

- Coordinators (super processes)
- Dynamic creation of sites
- Gateway connection management



$$
\begin{array}{lcl}
\mathbb{P} & ::= & \gamma.\mathbb{P} \mid \mathbb{P}_1 \mid \mathbb{P}_2 \\
\gamma & ::= & \alpha \\
& \mid & \mathbf{new}(s, \mathbb{P}) \\
& \mid & \mathbf{login}(s, \kappa) \\
& \mid & \mathbf{accept}(s, \kappa) \\
& \mid & \mathbf{logout}(s, \kappa) \\
& \mid & \mathbf{disconnect}(s, \kappa)
\end{array}
$$

Connection costs

Cost κ abstracts characteristics of connections (bandwidth, latency, distance, access rights ...)

Connection costs

Cost κ abstracts characteristics of connections (bandwidth, latency, distance, access rights ...)

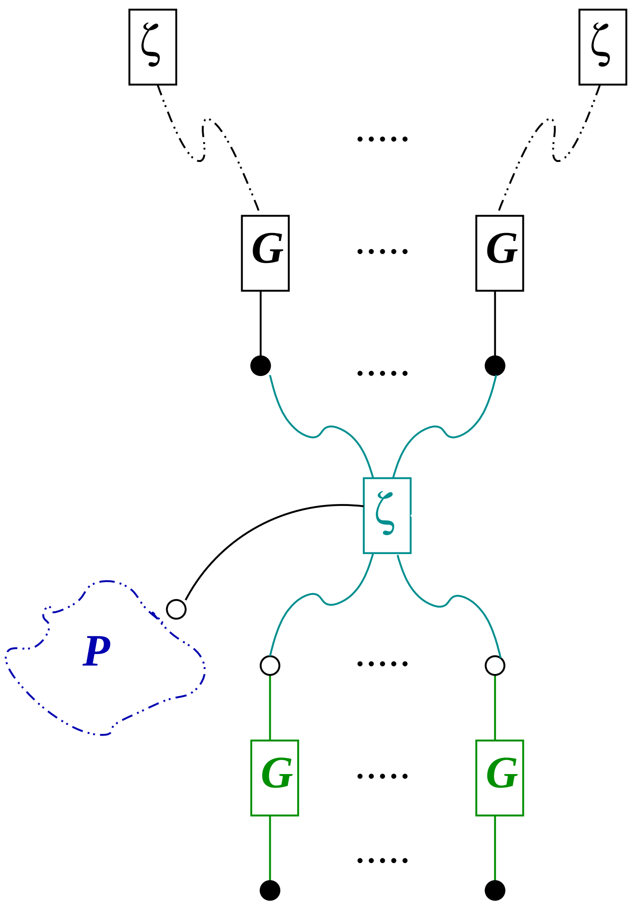
Algebra on costs: c-semiring. For instance

$$\langle c_1, \pi_1 \rangle \oplus \langle c_2, \pi_2 \rangle = \langle c_1 + c_2, \pi_1 \cup \pi_2 \rangle$$

$$\langle c_1, \pi_1 \rangle \otimes \langle c_2, \pi_2 \rangle = \begin{cases} \langle c_1 + c_2, \pi_1 \cap \pi_2 \rangle & \text{if } c_2 < c_1 \text{ and } \pi_2 \subset \pi_1 \\ \bot & \text{otherwise} \end{cases}$$

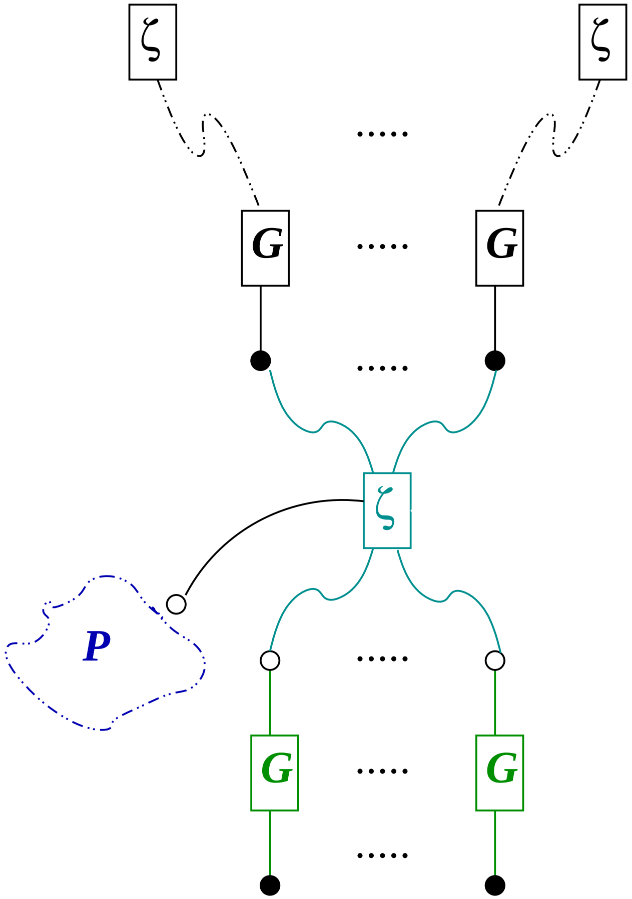
Qlaim & Hypergraphs

$$[\![\, s ::^{L,}\, P \,]\!] \;=\; \Gamma \vdash (\nu\, \vec{x}, p)([\![\, P \,]\!]_p \mid \mathfrak{S}^{s}_{m,n}(\vec{u}, \vec{x}, p) \mid \prod_{j=1}^{n} G^{\kappa_j}_{t_j}(x_j, v_j))$$



Qlaim & Hypergraphs

$$\llbracket\, s ::^{L,}\ P \,\rrbracket \;=\; \Gamma \vdash (\nu\ \vec{x}, p)(\llbracket\, P \,\rrbracket_p \mid \mathfrak{S}^{s}_{m,n}(\vec{u}, \vec{x}, p) \mid \prod_{j=1}^{n} G^{\kappa_j}_{t_j}(x_j, v_j))$$



$$\begin{array}{lcl}
\llbracket\, \textbf{nil} \,\rrbracket_p & = & nil \\
\llbracket\, \textbf{out}t \,\rrbracket_p & = & L_{\textbf{out}t}(p) \\
\llbracket\, \gamma.P \,\rrbracket_p & = & L_{\gamma.P}(p) \\
\llbracket\, \textbf{eval}(P)@s \,\rrbracket_p & = & (\nu\, u)(\textbf{eval}^{T(P)}_{s}(u, p) \mid S_P(u)) \\
\llbracket\, P_1 \mid P_2 \,\rrbracket_p & = & \llbracket\, P_1 \,\rrbracket_p \mid \llbracket\, P_2 \,\rrbracket_p \\
\llbracket\, rec\, X.\, P \,\rrbracket_p & = & \llbracket\, P[^{rec\, X.\, P}/_{X}] \,\rrbracket_p .
\end{array}$$

Qlaim’s Graph semantics: pros & cons

Qlaim's Graph semantics: pros & cons

– Many productions (recently reduced :-)

Qlaim’s Graph semantics: pros & cons

– Many productions (recently reduced :-)

+ Determines the “optimal” path (also Qlaim)

Qlaim’s Graph semantics: pros & cons

– Many productions (recently reduced :-)

+ Determines the “optimal” path (also Qlaim)

+ Path reservation

Qlaim's Graph semantics: pros & cons

– Many productions (recently reduced :-)

+ Determines the "optimal" path (also Qlaim)

+ Path reservation

+ Path routing

Qlaim's Graph semantics: pros & cons

– Many productions (recently reduced :-)

+ Determines the "optimal" path (also Qlaim)

+ Path reservation

+ Path routing

Theorem Qlaim remote actions are routed on paths with minimal cost (wrt the c-semiring operations)

Hypergraph & Software Design

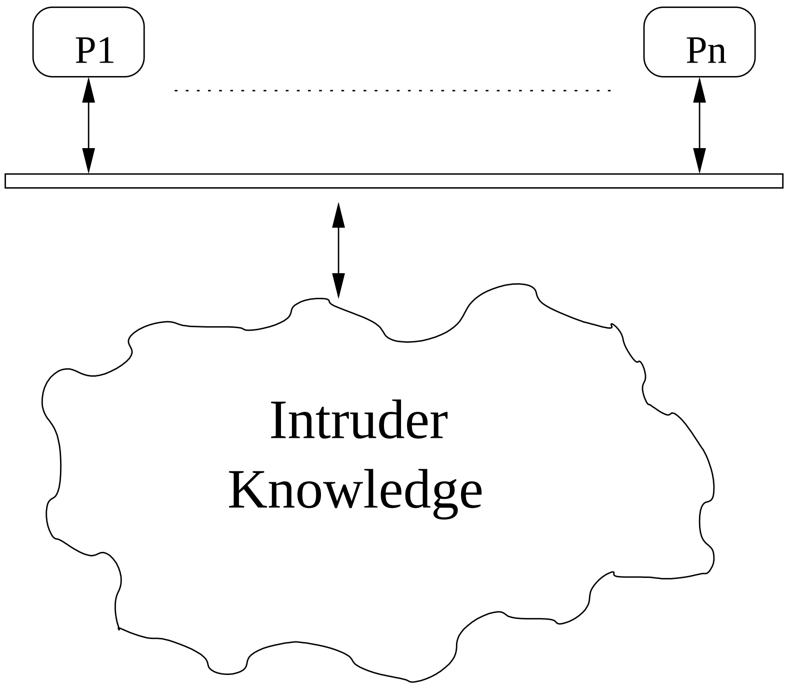
In [KGKK02] graph transformation is used for modelling dynamic behaviour of UML specifications.

+ Formal semantics of computations
– No local rewritings
– Distribution is not considered

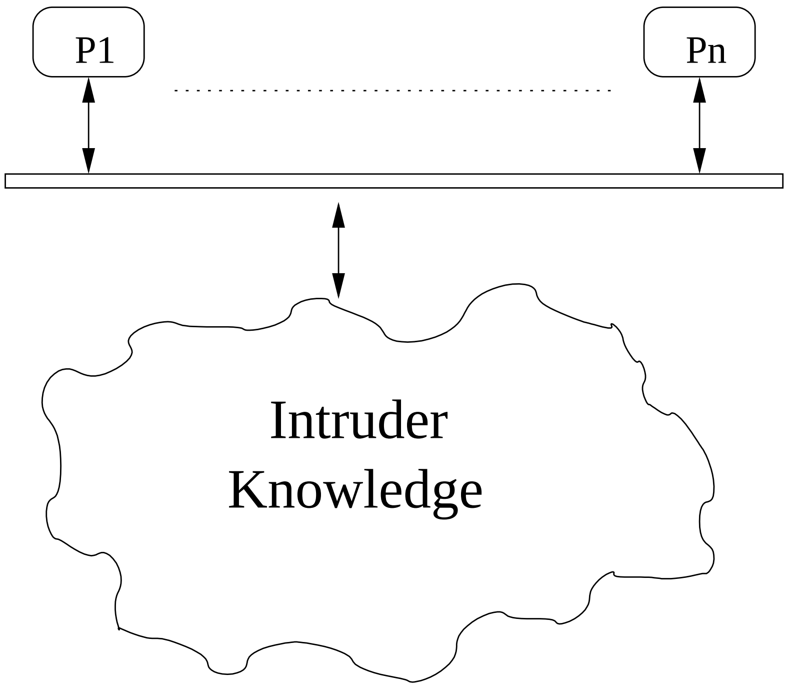
SHR has been applied as a further refinement step in the software design process.

Security

The Dolev-Yao Model

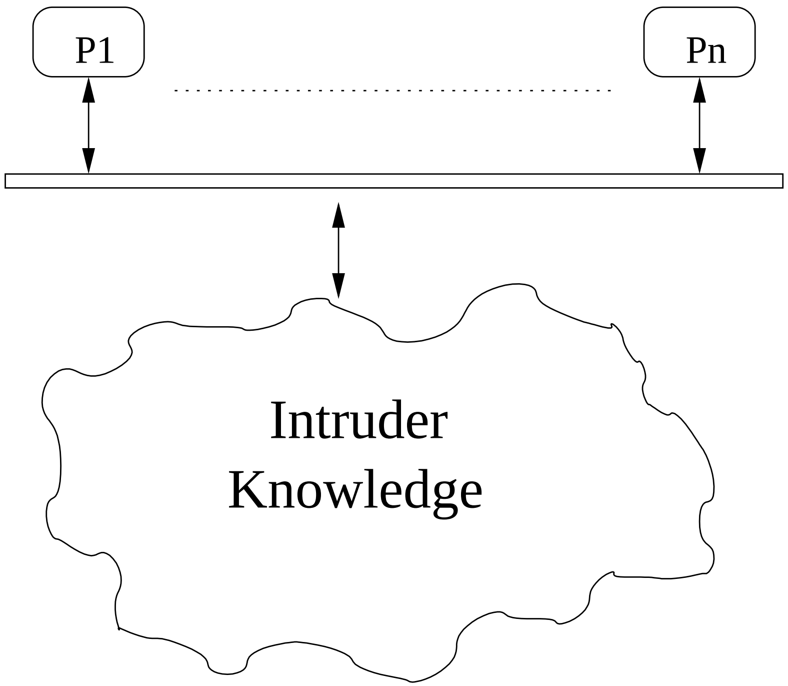


The Dolev-Yao Model



- Receive and store any transmitted message
- Hinder a message
- Decompose messages into parts
- Forge messages using known data
- Perfect Encryption Hypothesis

The Dolev-Yao Model



- Receive and store any transmitted message
- Hinder a message
- Decompose messages into parts
- Forge messages using known data
- Perfect Encryption Hypothesis

Names $n, m, ..., A, B, S, ...$

Keys $k, k', ..., A^+, A^-, ...$

Messages $M \quad ::= \quad N \mid K \mid M, M \mid \{M\}_M$

Intruder capabilities: $\bowtie$

$$\frac{m \in \kappa}{\kappa \bowtie m}\,(\in) \qquad \frac{\kappa \bowtie m \quad \kappa \bowtie n}{\kappa \bowtie m, n}\,(,) \qquad \frac{\kappa \bowtie m \quad \kappa \bowtie \lambda}{\kappa \bowtie \{m\}_\lambda}\,(\{\})$$

$$\frac{\kappa \bowtie m, n}{\kappa \bowtie m}\,(+_1) \qquad \frac{\kappa \bowtie m, n}{\kappa \bowtie n}\,(+_2) \qquad \frac{\kappa \bowtie \{m\}_\lambda \quad \kappa \bowtie \lambda^-}{\kappa \bowtie m}\,(\}\{)$$

Generalising [CJM98] to asymmetric key cryptography

Theorem $\bowtie$ is decidable

A Calculus of Principals

Some design choices:

- Cryptography & communication (pattern-matching)
- Key-sharing via “name fusion”
- Rôle based calculus
- Multi-session facilities

Syntax of cIP

Extension of IP [BBT01]

$$
\begin{array}{lcl}
E, F & ::= & \mathbf{nil} \mid \alpha.E \mid E + E \mid E|E \\
\alpha, \beta & ::= & in(d) \mid out(d) \\
d & ::= & N \mid K \mid d, d \mid \{d\}_d \mid x \mid ?x
\end{array}
$$

$$
\begin{array}{l}
1. A \rightarrow B : \{na, A\}_{B+} \\
2. B \rightarrow A : \{na, nb\}_{A+} \\
3. A \rightarrow B : \{nb\}_{B+}
\end{array}
$$

$$
A \triangleq (y)[\quad \begin{array}{l} out(\{na, A\}_{y^+}). \\ in(\{na, ?u\}_{A^-}). \\ out(\{u\}_{y^+})] \end{array}
$$

$$
B \triangleq ()[\quad \begin{array}{l} in(\{?x, ?z\}_{B^-}). \\ out(\{x, nb\}_{z^+}). \\ in(\{nb\}_{B^-})] \end{array}
$$

cIP Semantics

$$\frac{}{\alpha.E \xrightarrow{\alpha} E} \qquad \frac{E \xrightarrow{\alpha} E'}{E + F \xrightarrow{\alpha} E'}$$

$$\frac{E \xrightarrow{\alpha} E'}{E \parallel F \xrightarrow{\alpha} E' \parallel F}\ \mathrm{bn}(\alpha) \cap \mathrm{fn}(F) = \emptyset$$

$$\frac{E_i \xrightarrow{in(d)} E'_i \qquad \partial(\kappa) \triangleright m : \ \exists \sigma \text{ ground s.t. } d\sigma \sim m}{\langle (\tilde{X}_i)[E_i] \cup \mathcal{C}, \chi, \kappa \rangle \ \mapsto \ \langle (\tilde{X}_i)[E'_i\sigma] \cup \mathcal{C}, \chi\sigma, \kappa \rangle}$$

$$\frac{E_i \xrightarrow{out(m)} E'_i}{\langle (\tilde{X}_i)[E_i] \cup \mathcal{C}, \chi, \kappa \rangle \ \mapsto \ \langle (\tilde{X}_i)[E'_i] \cup \mathcal{C}, \chi, \kappa \cup m \rangle}$$

$$\frac{\mathcal{C}' = join(A_i, \gamma, \mathcal{C}) \qquad A \triangleq (\tilde{X})[E] \qquad i \text{ new}}{\langle \mathcal{C}, \chi, \kappa \rangle \ \mapsto \ \langle \mathcal{C}', \chi\gamma, \kappa \cup \{A_i\} \rangle}$$

$\mathcal{PL}$: Formalising Security Properties

$$
\begin{array}{lcl}
\phi, \psi & ::= & \delta \in \mathfrak{K} \\
& \mid & \forall \alpha : A.\phi \\
& \mid & x@\alpha = \delta \\
& \mid & \alpha = \beta \mid \\
& \mid & \neg\phi \mid \phi \wedge \psi \\
\\
\delta & ::= & d \mid \alpha \mid x@\alpha
\end{array}
$$

$$\kappa \models_{\chi} \phi$$

“If B completes a protocol session and thinks that he has been talking to A, then A had started a protocol session thinking that she has been talking to B”

$$\forall \beta : B.\exists \alpha : A.(z@\beta = \alpha \rightarrow y@\alpha = \beta)$$

Mihda: Co-Algebraic Minimisation of Automata

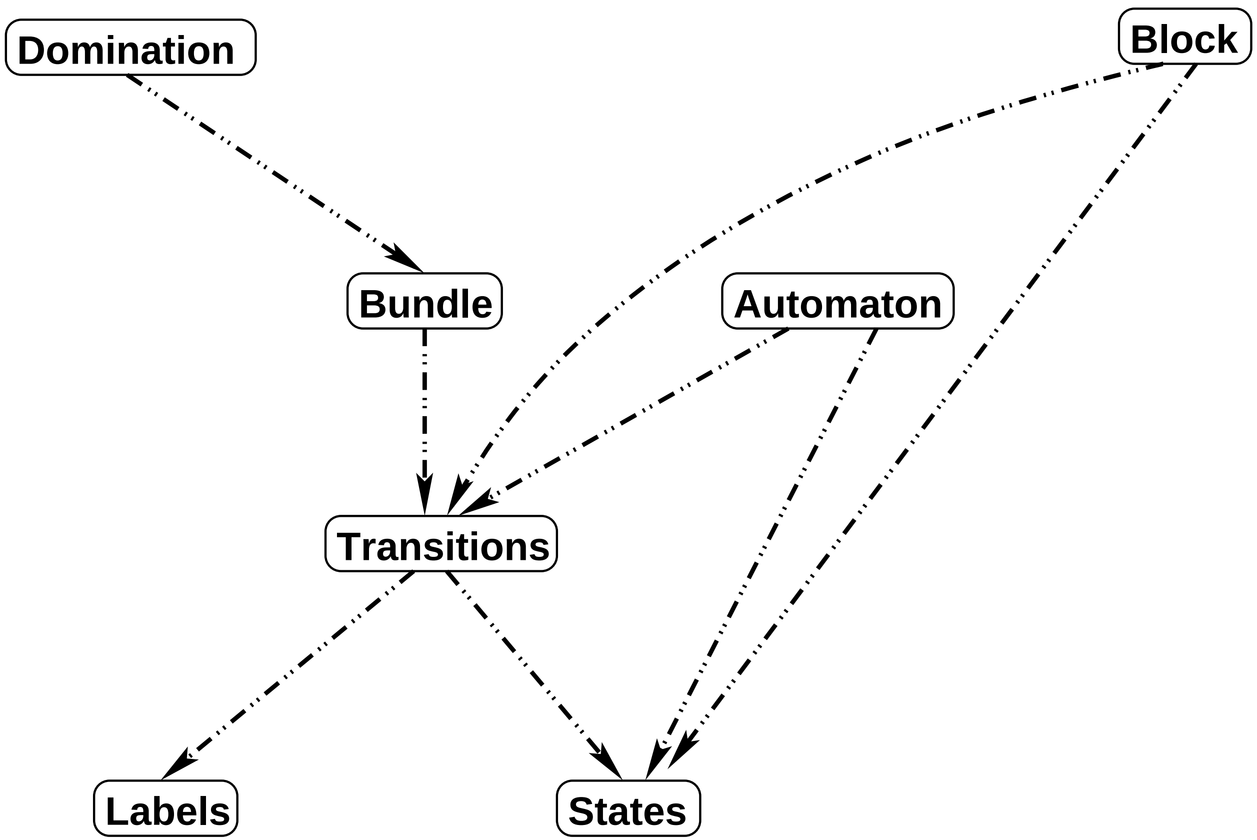
Mihda

Minimizing History Dependent Automata:

- HD-automata for history dependent calculi
- Co-algebraic specification
- Partition Refinement Algorithm based on co-algebraic specification [FMP02]
- Mihda: Ocaml implementation

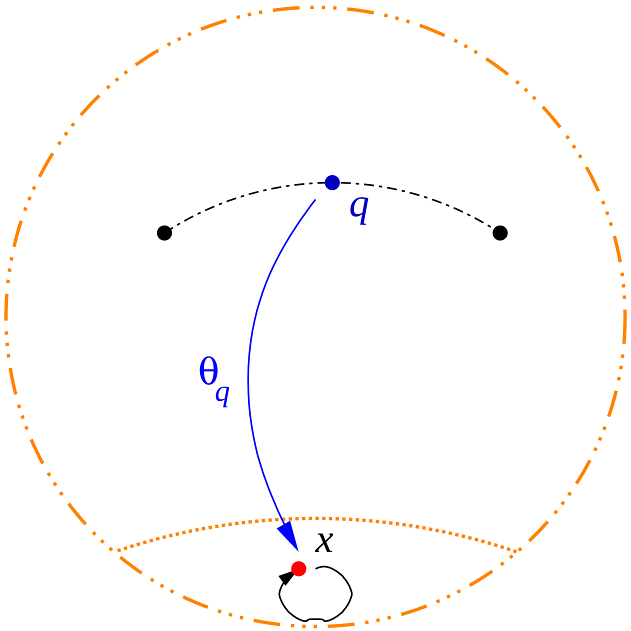
	Comp. Time	States	Trans.	Min. Time	States	Trans.
GSM small	0m 0.931s	211	398	0m 4.193s	105	197
GSM full	0m 8.186s	964	1778	0m 54.690s	137	253

Mihda **Architecture**

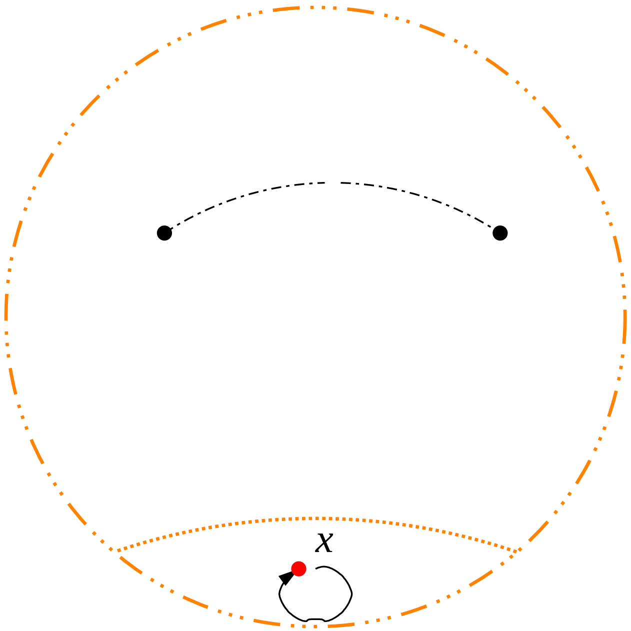


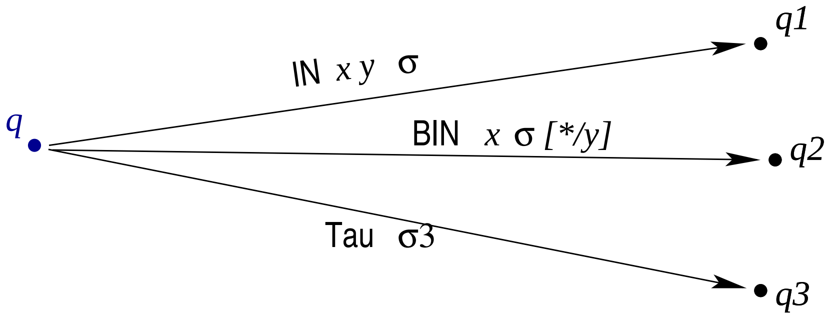
- Adherent to specs
- Highly modular
- Easily extendible

The main step



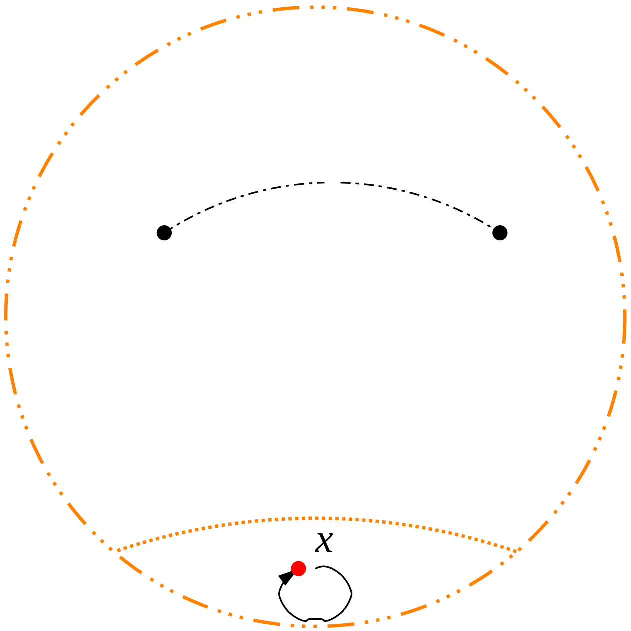
The main step

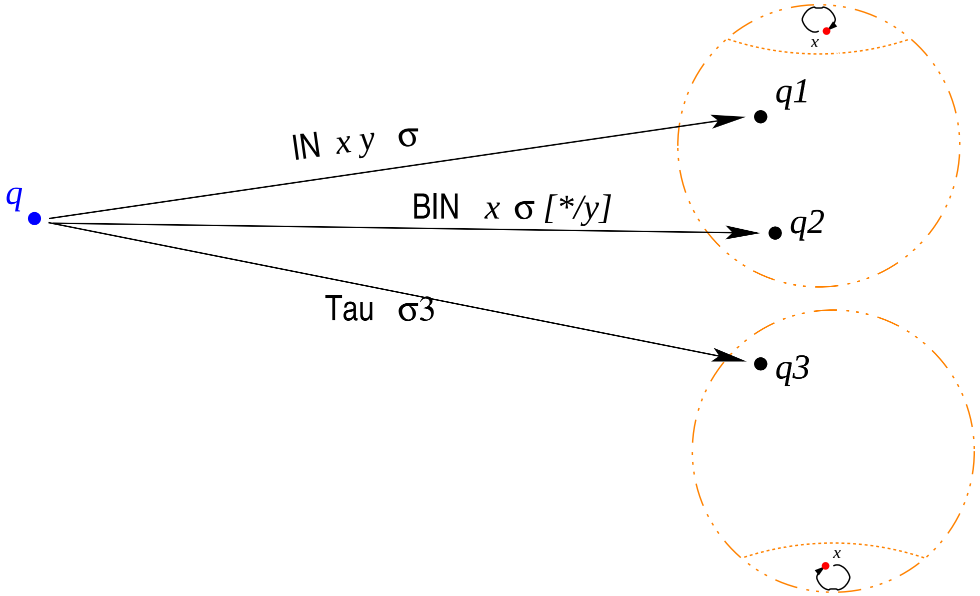





```
let bundle hd q =
  List.sort compare
    (List.filter (fun h → (Arrow.source h) = q) (arrows hd))
```

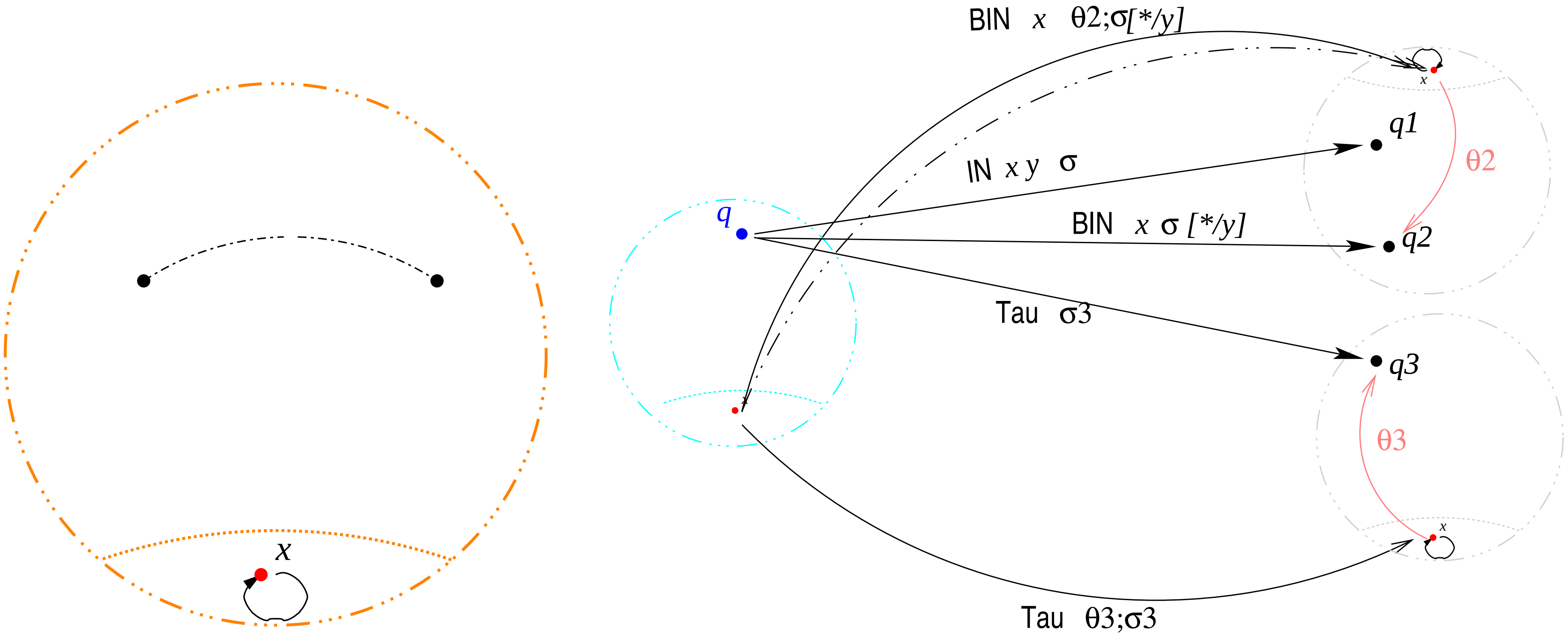
The main step





List.map h_n bundle

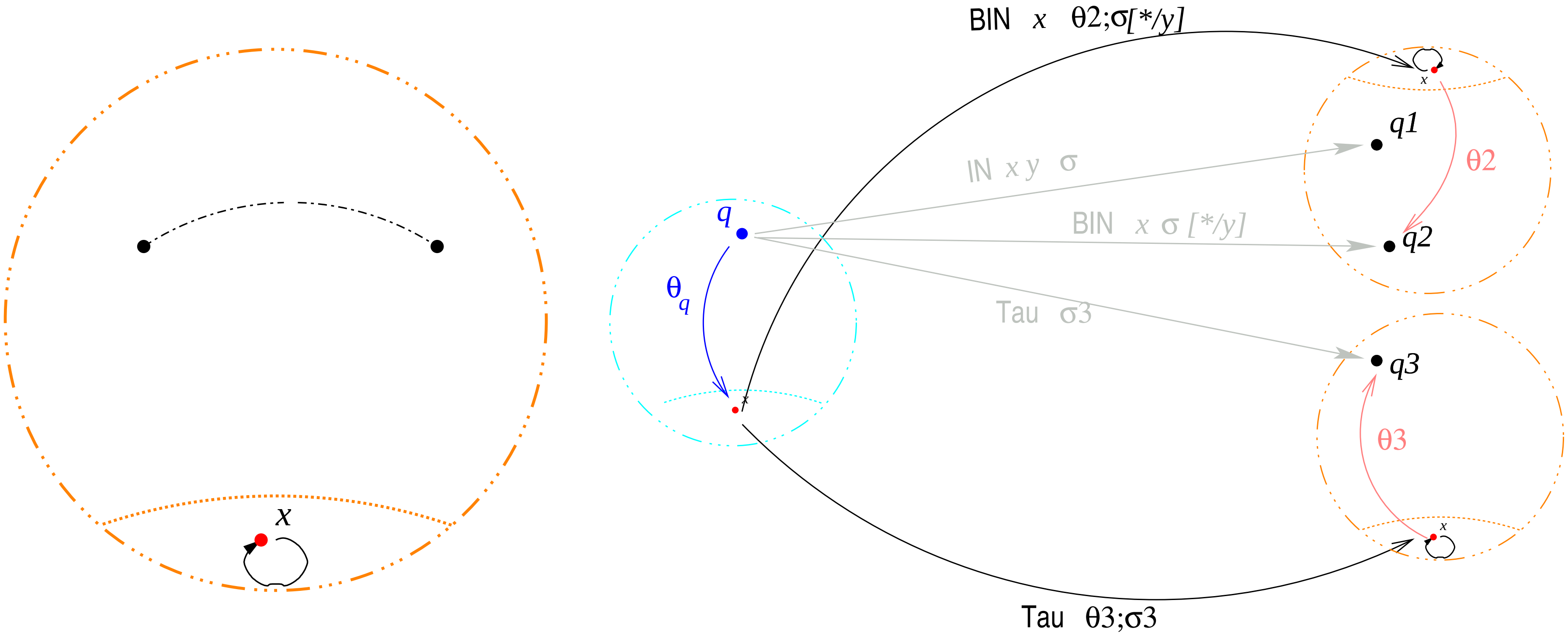
The main step



$$h_{n+1} = norm\langle \text{states}, \{\langle \ell, \pi, h_n(q'), \sigma'; \sigma\rangle \mid q \xrightarrow{\ell\ \pi\ \sigma} q' \ \wedge\ \sigma' \in \Sigma_n(q')\}\rangle$$

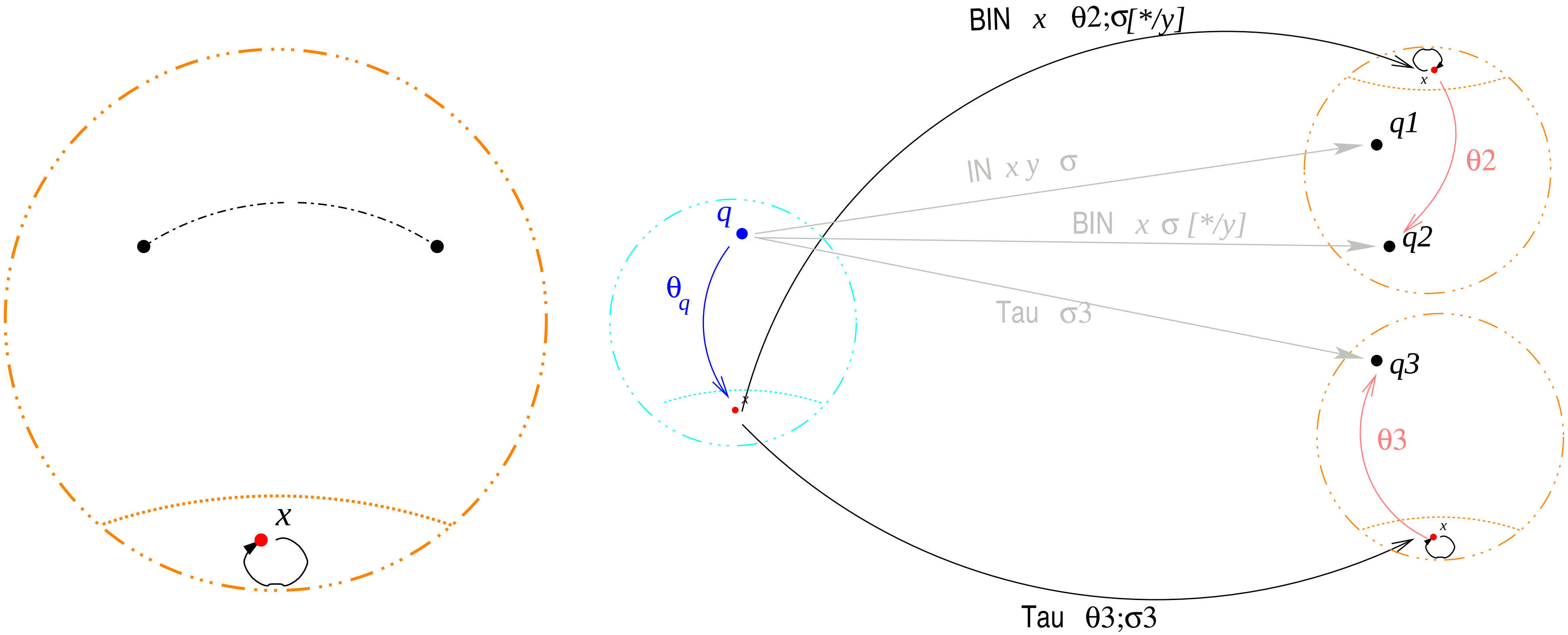
```
let red bl = ......
  let bl_in = List.filter covered_in bl
  in list_diff bl bl_in
```

The main step



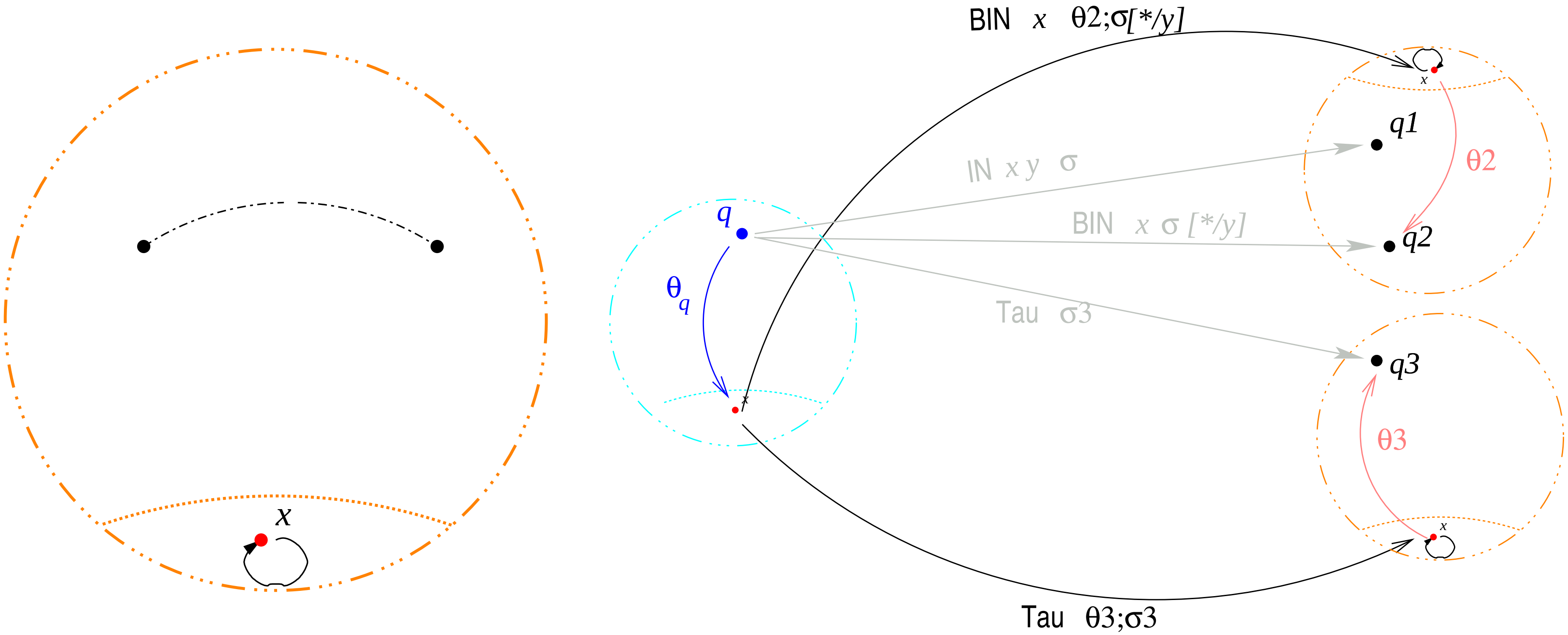

```
let an = active_names_bundle (red bundle) in
let remove_in ar = match ar with
   | Arrow(_,_,In(_,_)) → not (List.mem (obj ar) an)
   | _ → false in
     list_diff bundle (List.filter remove_in bundle)
```

The main step



$$\Sigma_{n+1}(q) = (\mathrm{compute\_group}\ \ (\mathrm{norm}\ \ \textsf{bundle}))\quad ;\quad \theta_q^{-1}$$

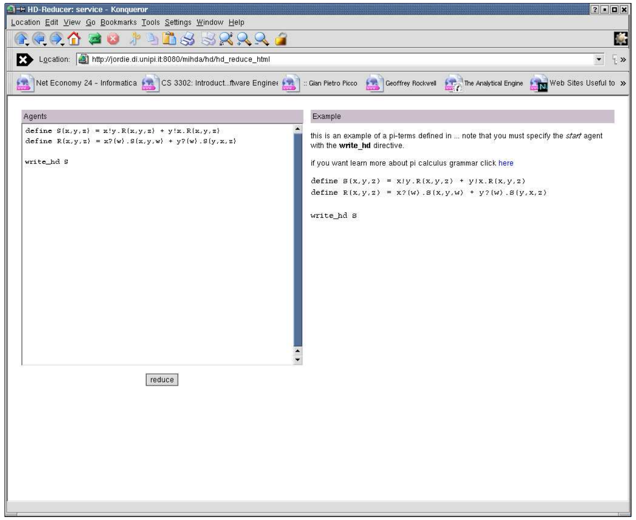
The main step

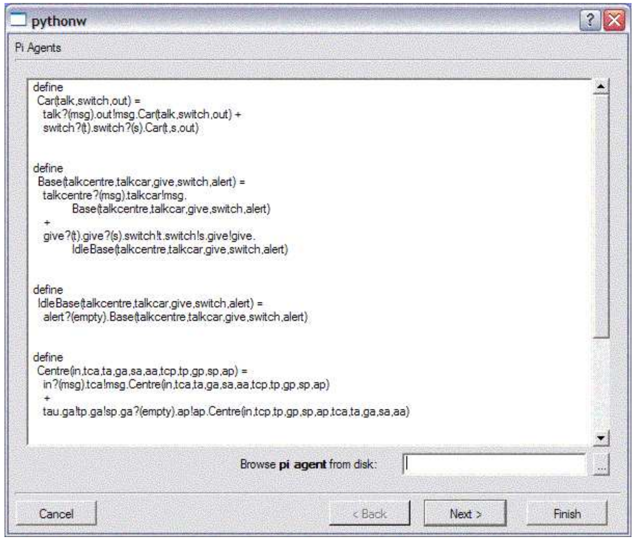


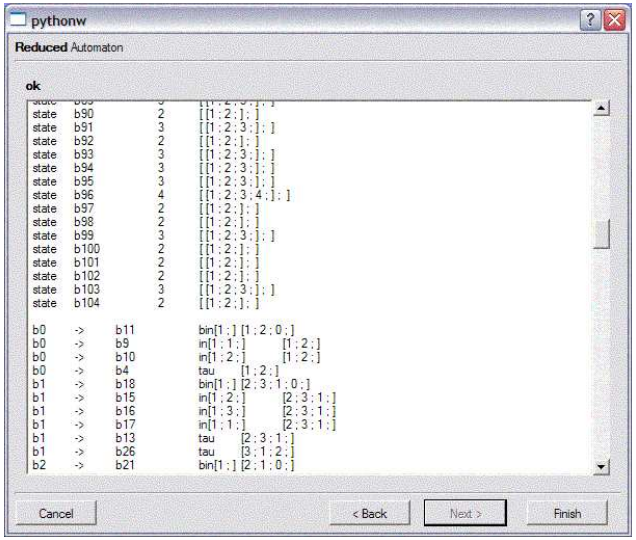
$$\Sigma_{n+1}(q) = (\text{compute\_group}\ \ (\text{norm}\ \ \textsf{bundle}))\ \ ;\ \ \theta_q^{-1}$$

Theorem At the end of each iteration i `blocks` corresponds to h_{H_i}

Mihda **Web Interface**







http://jordie.di.unipi.it:8080/pweb

Summing up...

Initial steps toward:

- Declarative approach to WAN programming
  - Foundational aspects
  - QoS at application level
  - Software Architectures (to be developed)
- Web Services
  - Secure composition of components
  - Coordination mechanism
- Tool development
  - Distributed infrastructure
  - Proof strategies as programmable coordinators

Published papers

- Ferrari, G., Pugliese, R., Tuosto, E. Calculi for Network Aware Programming. Workshop on agents 2000: Dagli oggetti agli agenti
- Ferrari, G., Montanari, U., Tuosto, E. LTS Semantics of Ambients via Graph Synchronization with Mobility. In *7th Italian Conference on Theoretical Computer Science – ICTCS'01*, volume 2202 of LNCS. Springer, 2001
- Bracciali, A., Brogi, A., Ferrari, G., Tuosto, E.. Security Issues in Component Based Design, In *ConCoord Workshop 2001, Lipari - Italy*
- Bracciali, A., Brogi, A., Ferrari, G., Tuosto, E., Security and Dynamic Compositions of Open Systems. In *International conference of Parallel and Distributed Processing Techniques and Applications*, F. Arbarb et al. Editors, PDPTA 2002
- Ferrari, G., Montanari, U., Tuosto, E. Graph-based Models of Internetworking Systems. In *Formal Methods at the Crossroads: from Panaces to Foundational Support*, A. Haeberer editor, LNCS. Springer, 2003

References

[BBT01] Andrea Bracciali, Antonio Brogi, and Franco Turini. Coordinating interaction patterns. In *Proceedings of the* ACM Symposium on Applied Computing*, Las Vegas, USA*. ACM, 2001.

[BC99] Boumediene Bal, Henri E. Belkhouche and Luca Cardelli, editors. *Workshop on Internet Programming Languages*, volume 1686 of *LNCS*. Springer, 1999.

[BLP02] Lorenzo Bettini, Michele Loreti, and Rosario Pugliese. Infrastructure language for open nets. In Proc. of the 2002 ACM Symposium on Applied Computing (SAC'02), Special Track on Coordination Models, Languages and Applications. ACM Press, 2002. Special Track on Coordination Models, Languages and Applications.

[CG00] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *TCS: Theoretical Computer Science*, 240, 2000.

[CJM98] Edmund M. Clarke, Somesh Jha, and Wilfredo R. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *IFIP Working Conference on Programming Concepts and Methods (PROCOMET)*, 1998.

[CM83] Ilaria Castellani and Ugo Montanari. Graph Grammars for Distributed Systems. In Hartmut Ehrig, Manfred Nagl, and Grzegorz Rozenberg, editors, *Proc. 2nd Int. Workshop on Graph-Grammars and Their Application to Computer Science*, volume 153 of *Lecture Notes in Computer Science*, pages 20–38. Springer-Verlag, 1983.

[DFP98] Rocco De Nicola, Gianluigi Ferrari, and Rosario Pugliese. KLAIM: A kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.

[DFPV00] Rocco De Nicola, Gianluigi Ferrari, Rosario Pugliese, and Betti Venneri. Types for access control. *Theoretical Computer Science*, 240(1):215–254, June 2000.

[DM87] Pierpaolo Degano and Ugo Montanari. A model of distributed systems based of graph rewriting. *Journal of the ACM*, 34:411–449, 1987.

[FG96] Cedric Fournet and George Gonthier. The reflexive CHAM and the join-calculus. In *Conference Record of POPL '96: The 23^{rd} ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 372–385, St. Petersburg Beach, Florida, January 1996.

[FGL+96] Cedric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, and Didier Rémy. A calculus of mobile processes. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96: Concurrency Theory, 7th International Conference*, volume 1119 of *Lecture Notes in Computer Science*, pages 406–421, Pisa, Italy, August 1996. Springer-Verlag.

[FMP02] Gianluigi Ferrari, Ugo Montanari, and Marco Pistore. Minimizing transition systems for name passing calculi: A co-algebraic formulation. In Mogens Nielsen and Uffe Engberg, editors, *FOSSACS 2002*, volume LNCS 2303, pages 129–143. Springer Verlag, 2002.

[HIM00] Dan Hirsch, Paola Inverardi, and Ugo Montanari. Reconfiguration of Software Architecture Styles with Name Mobility. In Antonio Porto and Gruia-Catalin Roman, editors, *Coordination 2000*, volume

1906 of *LNCS*, pages 148–163. Springer Verlag, 2000.

[HM01] Dan Hirsch and Ugo Montanari. Synchronized hyperedge replacement with name mobility: A graphical calculus for name mobility. In *12th International Conference in Concurrency Theory (CONCUR 2001)*, volume 2154 of *LNCS*, pages 121–136, Aalborg, Denmark, 2001. Springer Verlag.

[HR98] Mattew Hennessy and James Riely. Resource access control in systems of mobile agents. In Uwe Nestmann and Benjamin C. Pierce, editors, *HLCL '98: High-Level Concurrent Languages (Nice, France, September 12, 1998)*, volume 16.3 of *entcs*, pages 3–17. Elsevier Science Publishers, 1998. Full version as CogSci Report 2/98, University of Sussex, Brighton.

[HR00] Matthew Hennessy and James Riely. Information flow vs. resource access in the asynchronous pi-calculus. In *27th International Colloquium on Automata, Languages and Programming (ICALP '2000)*, July 2000. A longer version appeared as Computer Science Technical Report 2000:03, School of Cognitive and Computing Sciences, University of Sussex.

[KGKK02] Sabine Kuske, Martin Gogolla, Ralf Kollmann, and Hans-Jörg Kreowski. An Integrated Semantics for UML Class, Object, and State Diagrams based on Graph Transformation. In Michael Butler and Kaisa Sere, editors, *3rd Int. Conf. Integrated Formal Methods (IFM'02)*, LNCS. Springer, Berlin, 2002.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40,41–77, September 1992.

[MR96] Ugo Montanari and Francesca Rossi. Graph rewriting and constraint solving for modelling distributed systems with synchronization. In P. Ciancarini and C. Hankin, editors, *Proceedings of the First International Conference COORDINATION '96, Cesena, Italy*, volume 1061 of *LNCS*. Springer Verlag, April 1996.

[VC98] Jan Vitek and Giuseppe Castagna. Towards a calculus of secure mobile computations. In *[BC99]*, Chicago, Illinois, May 1998.