Theories for Service Oriented Computing

Emilio Tuosto

Dipartimento di Informatica, Università di Pisa

Largo B. Pontecorvo 3, 56127 Pisa, Italy. +39-50-2212799. +39-50-2212726.

etuosto@di.unipi.it - http://www.di.unipi.it/˜etuosto

Napoli, 29 June 2005

Plan of the talk

- What global computing and service oriented computing are?
  - Abstractions for SOC
    - Constraint semirings
    - $\mathcal{K}oS$
    - APIs for SOC
  - Hypergraphs Model
    - Programming model
    - Specification hypergraphs and related logic
  - Automata Model
- Semantic based verification
  - Model checking
  - Equivalence checking

Service Oriented Computing

- applications are made by gluing *services*
  - “autonomous”
  - independent (local choices, independently built)
  - mobile/stationary
  - “interconnected”
- interactions governed by programmable coordination policies
- services are searched and binded ... offline

- Service Oriented Computing
  - applications are made by gluing *services*
    - "autonomous"
    - independent (local choices, independently built)
    - mobile/stationary
    - "interconnected"
  - interactions governed by programmable coordination policies
  - services are searched and binded ... offline

- Can search/bind be dynamic and at run-time?

- Service Oriented Computing
  - applications are made by gluing *services*
    - “autonomous”
    - independent (local choices, independently built)
    - mobile/stationary
    - “interconnected”
  - interactions governed by programmable coordination policies
  - services are searched and binded ... offline

- Can search/bind be dynamic and at run-time?

- What should be considered relevant for searching?

SOC architectures

SOC architectures are

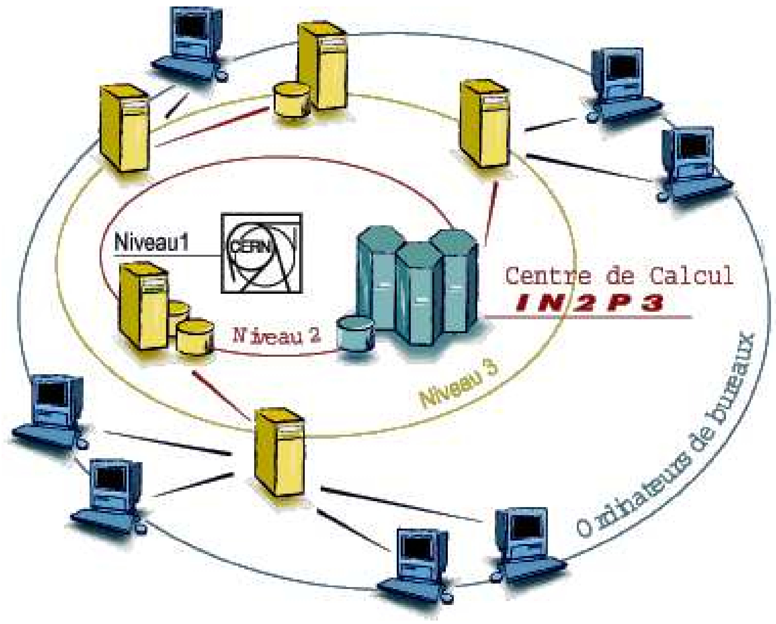
- distributed
- interconnected
- based on different communication infrastructures:
  - IP, wireless, satellites...
  - multi-layered: overlay networks (EC-GC2)

Remark 1 *Designers and end-users may ignore the stratification and complexity of the systems. They usually have a high-level view of the computations!*

...hence, global computing is not just **go**(P), $\bar{s}\langle x\rangle$ or $s(y)$ & search/bind cycle of SOAs must be defined abstractly. For instance,

- Application-level QoS
- Long Running Transactions
- ...

Abstractions for application-level QoS



Application-level QoS

- Lifting QoS issues to application level...
- ...for programming global computers
- with programmable application level QoS
- QoS for designing and implementing SOC applications (SOA)
- no longer considered only at the low-level layers

First steps (extending Klaim) in [DFM+03] and recently in [DFM+05]

Search and bind wrt application level QoS

- application-related, e.g.
  - price
  - transactions
  - payment mode
  - data available in a given format
- low-level related (e.g., throughput, response time) not directly referred but abstracted for expressing their "perception" at high-level.

  Example 1 *When downloading remote information QoS is specified as a constraint on the format of the file (e.g, DVI format instead of Postscript) instead of the underlying network speed (either for application-dependent constraints e.g., editing motivations or download time constraints).*

QoS as Constraint Semirings

C-Semirings [BMR95, BMR97] for abstracting application level QoS [DFM+03]

- $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$, where
  - S is a set (containing $\mathbf{0}$ and $\mathbf{1}$),
  - $+, \star : S \times S \to S$

$+$	$\star$
$x + y = y + x$	$x \star y = y \star x$
$(x + y) + z = x + (y + z)$	$(x \star y) \star z = x \star (y \star z)$
$x + \mathbf{0} = x$	$x \star \mathbf{0} = \mathbf{0}$
$x + \mathbf{1} = \mathbf{1}$	$x \star \mathbf{1} = x$
$x + x = x$	$(x + y) \star z = (x \star z) + (y \star z)$

QoS as Constraint Semirings

C-Semirings [BMR95, BMR97] for abstracting application level QoS [DFM+03]

- $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$, where
  - S is a set (containing $\mathbf{0}$ and $\mathbf{1}$),
  - $+, \star : S \times S \to S$

$+$	$\star$
$x + y = y + x$	$x \star y = y \star x$
$(x + y) + z = x + (y + z)$	$(x \star y) \star z = x \star (y \star z)$
$x + \mathbf{0} = x$	$x \star \mathbf{0} = \mathbf{0}$
$x + \mathbf{1} = \mathbf{1}$	$x \star \mathbf{1} = x$
$x + x = x$	$(x + y) \star z = (x \star z) + (y \star z)$

QoS as Constraint Semirings

C-Semirings [BMR95, BMR97] for abstracting application level QoS [DFM+03]

- $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$, where
  - S is a set (containing $\mathbf{0}$ and $\mathbf{1}$),
  - $+, \star : S \times S \rightarrow S$

$+$	$\star$
$x + y = y + x$	$x \star y = y \star x$
$(x + y) + z = x + (y + z)$	$(x \star y) \star z = x \star (y \star z)$
$x + \mathbf{0} = x$	$x \star \mathbf{0} = \mathbf{0}$
$x + \mathbf{1} = \mathbf{1}$	$x \star \mathbf{1} = x$
$x + x = x$	$(x + y) \star z = (x \star z) + (y \star z)$

- Implicit partial order: $a \leq b \iff a + b = b$ "b is *better* than a"

C-Semirings [BMR95, BMR97] for abstracting application level QoS [DFM+03]

- $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$, where
  - S is a set (containing $\mathbf{0}$ and $\mathbf{1}$),
  - $+, \star : S \times S \to S$

$+$	$\star$
$x + y = y + x$	$x \star y = y \star x$
$(x + y) + z = x + (y + z)$	$(x \star y) \star z = x \star (y \star z)$
$x + \mathbf{0} = x$	$x \star \mathbf{0} = \mathbf{0}$
$x + \mathbf{1} = \mathbf{1}$	$x \star \mathbf{1} = x$
$x + x = x$	$(x + y) \star z = (x \star z) + (y \star z)$

- Implicit partial order: $a \leq b \iff a + b = b$ "b is *better* than a"

Proposition 1 *Cartesian product, exponential and power constructions of c-semirings are c-semiring.*

Examples of c-semirings

Example 2 *The resource c-semiring* $\mathfrak{R} = \langle \omega_\infty, max, min, 0, \infty \rangle$ *is defined on* ω_∞, *the set of natural numbers with infinity.*

C-semirings for specifi c synchronisation mechanisms:

Example 3 *The Hoare synchronisation c-semiring is* $\mathfrak{H} = \langle \mathcal{H}, +_H, \star_H, \mathbf{0}_H, \mathbf{1}_H \rangle$ *where* $\mathcal{H} = Act \cup \{\mathbf{1}_H, \mathbf{0}_H, \perp\}$ *and*

$$\forall a \in Act.\ a \star a = a \tag{1}$$

$$\forall a, b \in Act \cup \{\perp\} : b \neq a \implies a \star b = \perp \tag{2}$$

$$\textit{plus commutative rules and the ones for } \mathbf{0} \textit{ and } \mathbf{1} \tag{3}$$

Operation $+_H$ is obtained by extending the c-semiring axioms for the additive operation with

$$a +_H a = a, \forall a \in \mathcal{H}$$
$$a +_H b = \perp, \forall a, b \in Act \cup \{\perp\}. b \neq a$$

We can defi ne a general synchronisation policy as a c-semiring that combines (using the cartesian product) a classical synchronisation algebra with the QoS requirements of interest

e.g., $\mathfrak{H}\mathfrak{R} = \mathfrak{H} \times \mathfrak{R}$ is the c-semiring of broadcast with priorities

Another bunch of c-semiring examples

C-semirings structures can be defined for many frameworks:

- $\langle \{true, false\}, \vee, \wedge, false, true \rangle$ (boolean): Availability
- $\langle \mathrm{Real}+, min, +, +\infty, 0 \rangle$ (optimization): Price, propagation delay
- $\langle \mathrm{Real}+, max, min, 0, +\infty \rangle$ (max/min): Bandwidth
- $\langle [0, 1], max, \cdot, 0, 1 \rangle$ (probabilistic): Performance and rates
- $\langle [0, 1], max, min, 0, 1 \rangle$ (fuzzy): Performance and rates
- $\langle 2^N, \cup, \cap, \emptyset, N \rangle$ (set-based, where N is a set): Capabilities and access rights

Process Algebraic Foundations of SOC

π-calculus [MPW92] Djoin [FG96, FGL$^+$96] Dπ [HR98, HR00] Fusion [PV98]	Rich theory	basic wrt SOC (only link mobility)
Ambient [CG00] Seal [VC98] Boxed [BCC01] Safe [LS00]	Hierarchical	not very natural
Klaim [BBD$^+$03] Hierarchical [BLP02] OKlaim [BBV03] MetaKlaim [FMPar]	Very natural	Lack of observational semantics
...	...	...

- Twofold nature: calculus and language (logical vs. phisical sites: but no “routing” features)

Klaimology

- Twofold nature: calculus and language (logical vs. phisical sites: but no “routing” features)
- Multiple tuple spaces

[DFP98]

Klaimology

- Twofold nature: calculus and language (logical vs. phisical sites: but no “routing” features)
- Multiple tuple spaces
- Localities: first class citizens

[DFP98]

Klaimology

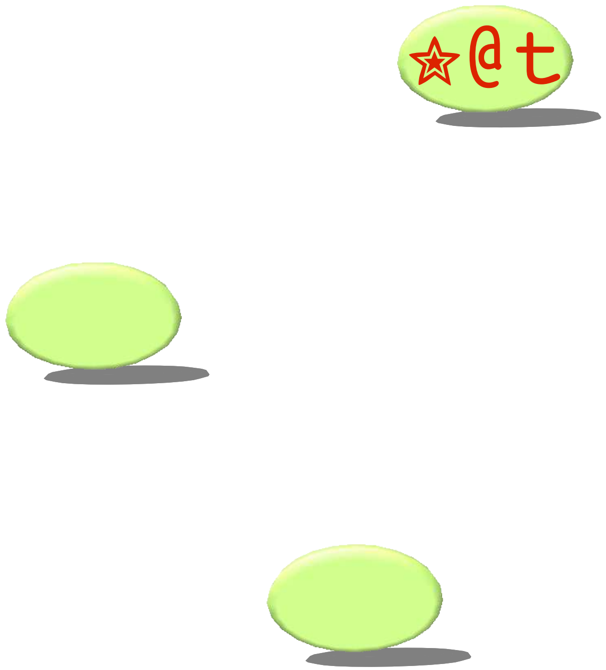
- Twofold nature: calculus and language (logical vs. phisical sites: but no “routing” features)
- Multiple tuple spaces
- Localities: first class citizens
- Process migration

[DFP98]

Klaimology

- Twofold nature: calculus and language (logical vs. phisical sites: but no “routing” features)
- Multiple tuple spaces
- Localities: first class citizens
- Process migration

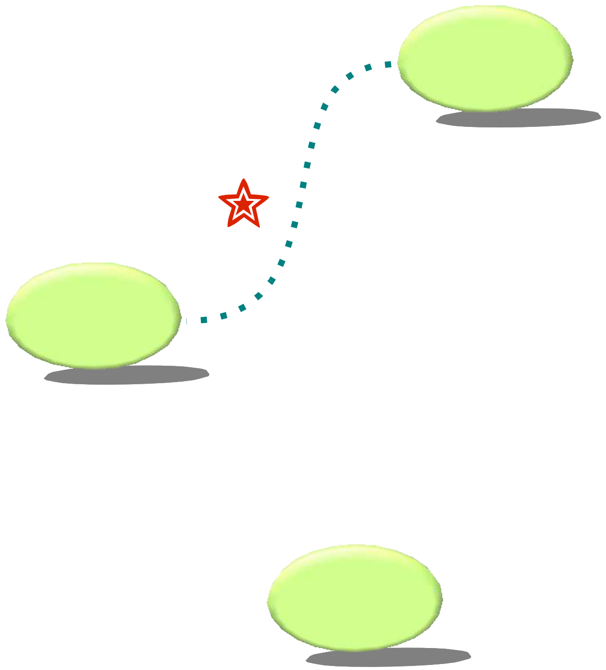
[DFP98]



Klaimology

- Twofold nature: calculus and language (logical vs. phisical sites: but no “routing” features)
- Multiple tuple spaces
- Localities: first class citizens
- Process migration

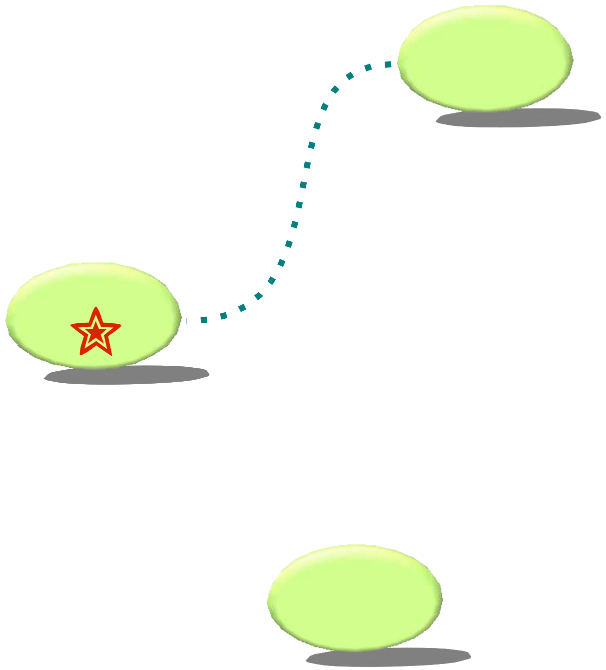
[DFP98]



Klaimology

- Twofold nature: calculus and language (logical vs. phisical sites: but no “routing” features)
- Multiple tuple spaces
- Localities: first class citizens
- Process migration

[DFP98]



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

$\mathcal{K}oS$ characteristics

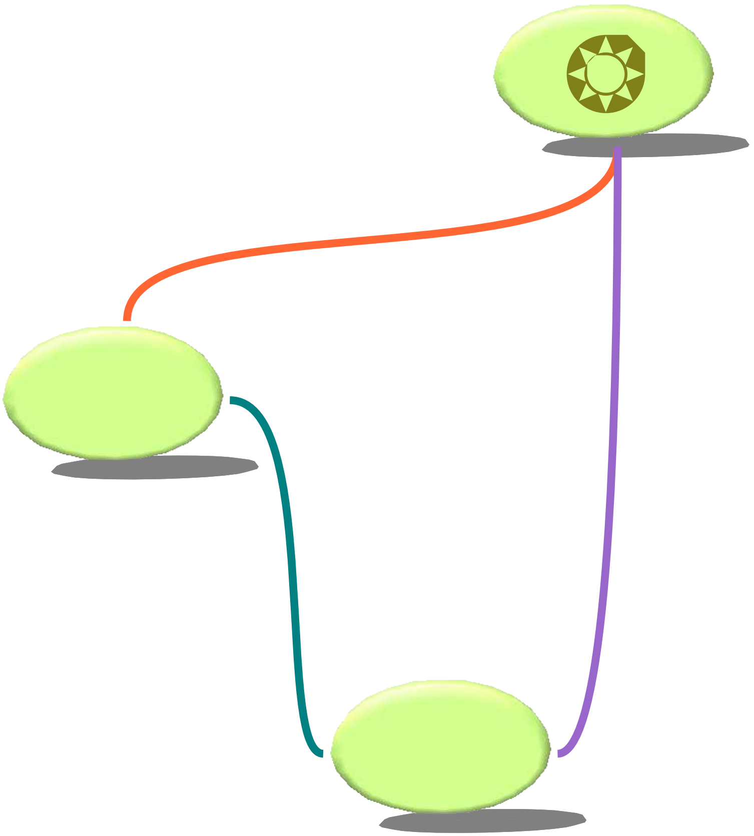
$\mathcal{K}oS$ aims at being a *minimal* calculus for SOC and

- builds on Klaim (e.g., processes are localised)...
- ...and π-calculus
- naturally supports P2P model
- QoS as first class citizen
- QoS-driven semantics

- only local communications (unlike Klaim)
- link construction primitives
- only one remote action
- which relies on link topology
- semantic transitions report the "cost" of the execution

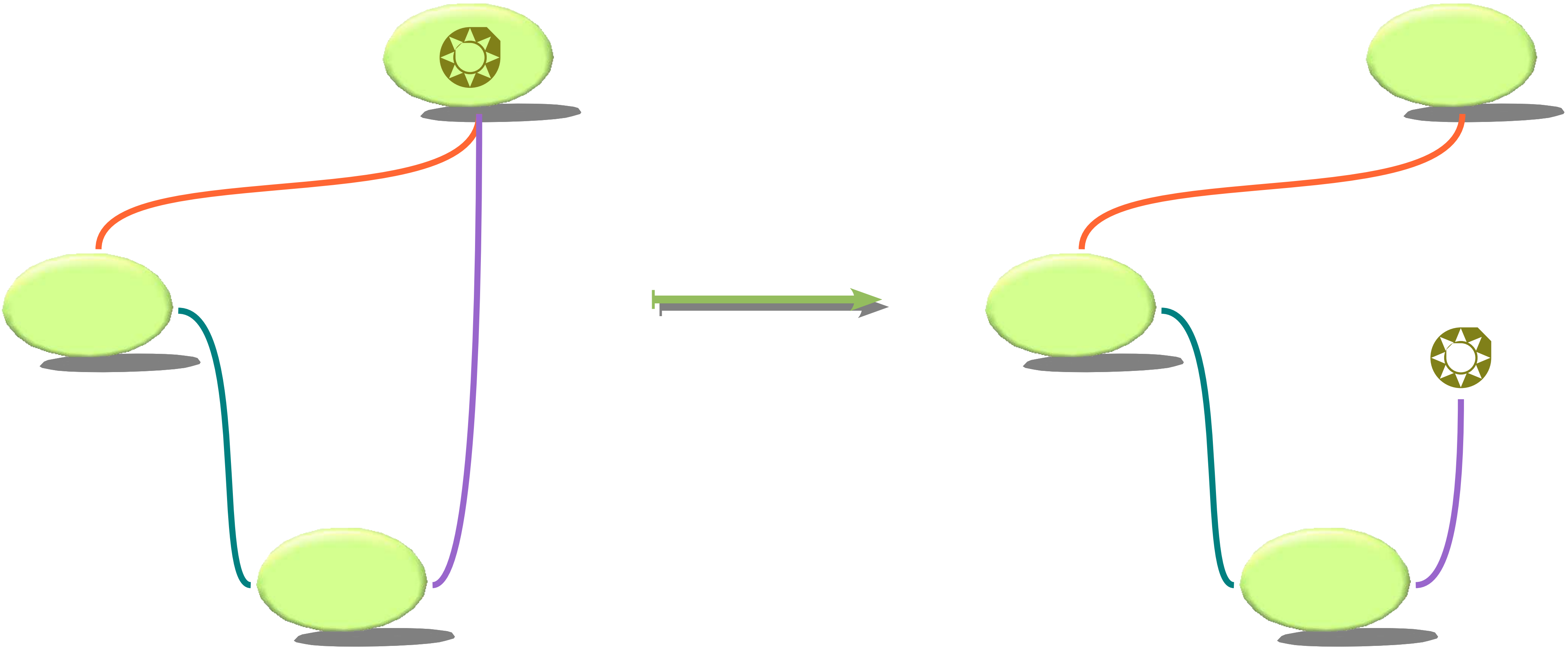
$\mathcal{K}o\mathcal{S}$ ... graphically

$\mathcal{K}oS$ ... graphically



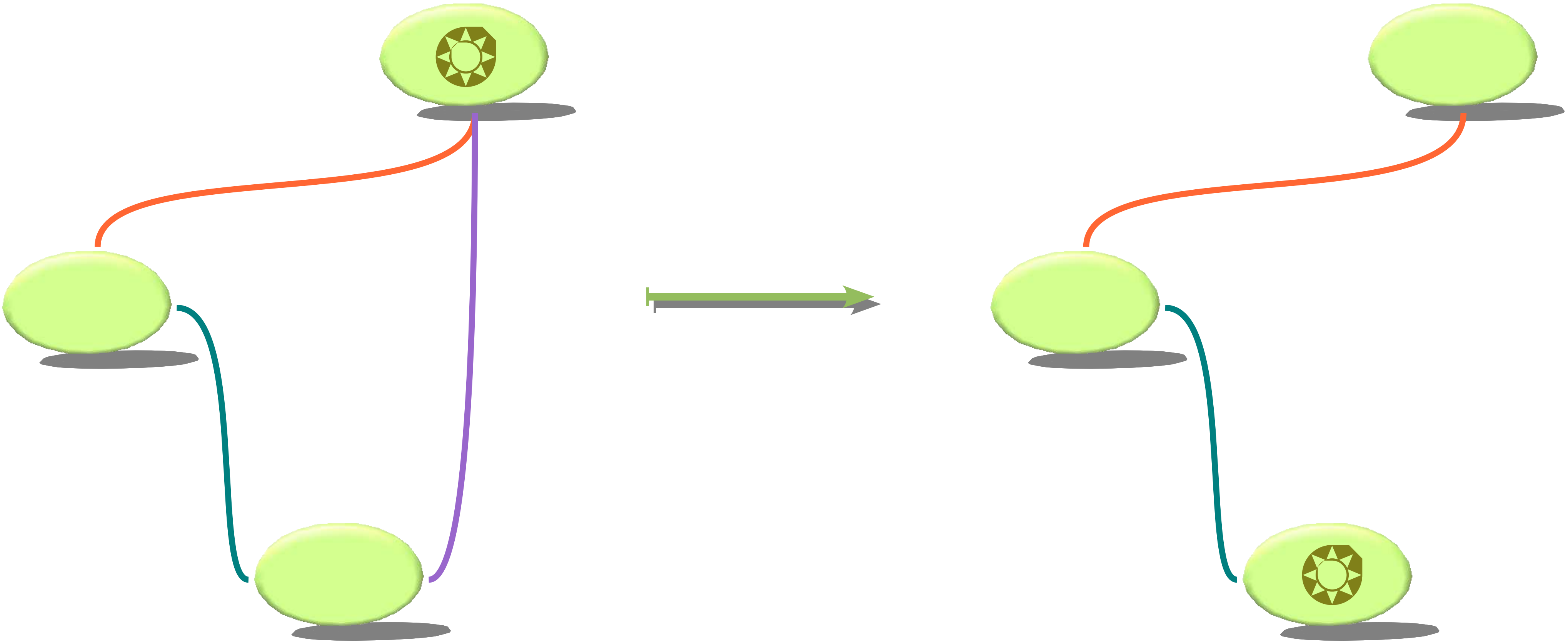
Remote actions in $\mathcal{K}oS$

$\mathcal{K}oS$ ... graphically

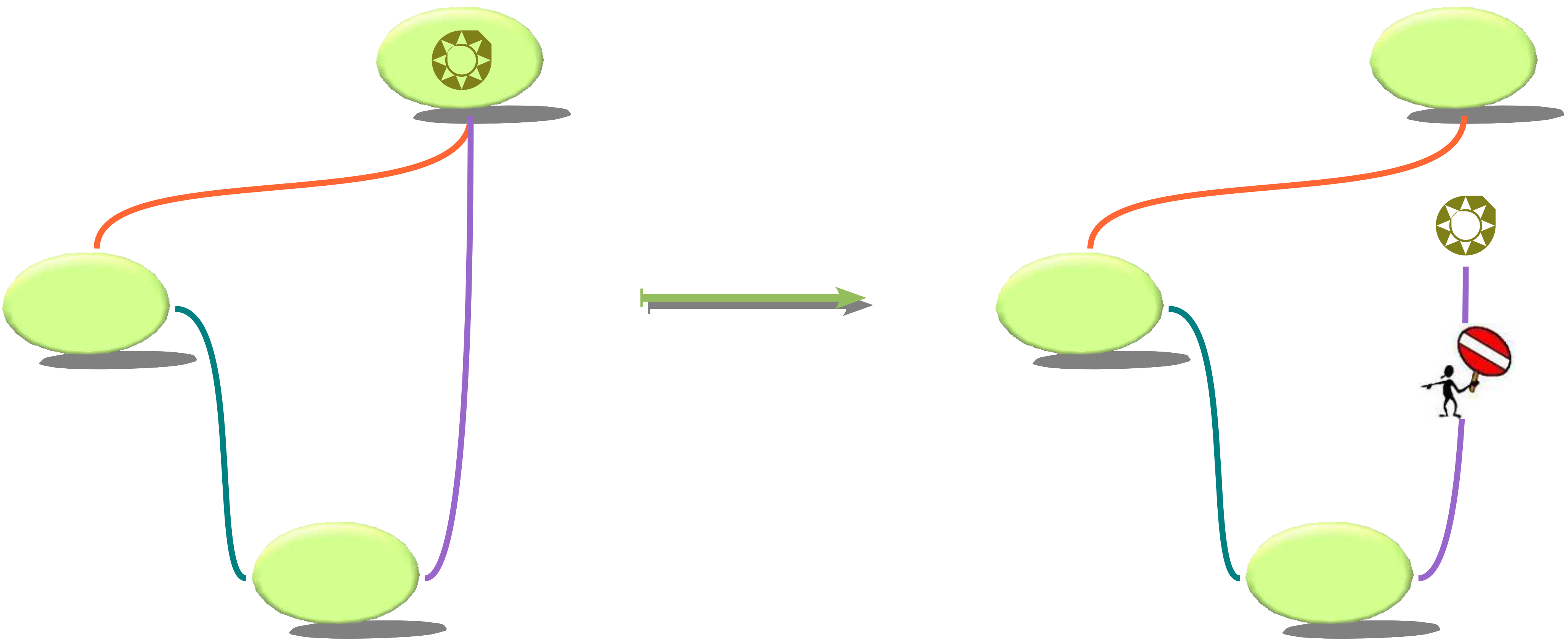


Remote actions in $\mathcal{K}o\mathcal{S}$

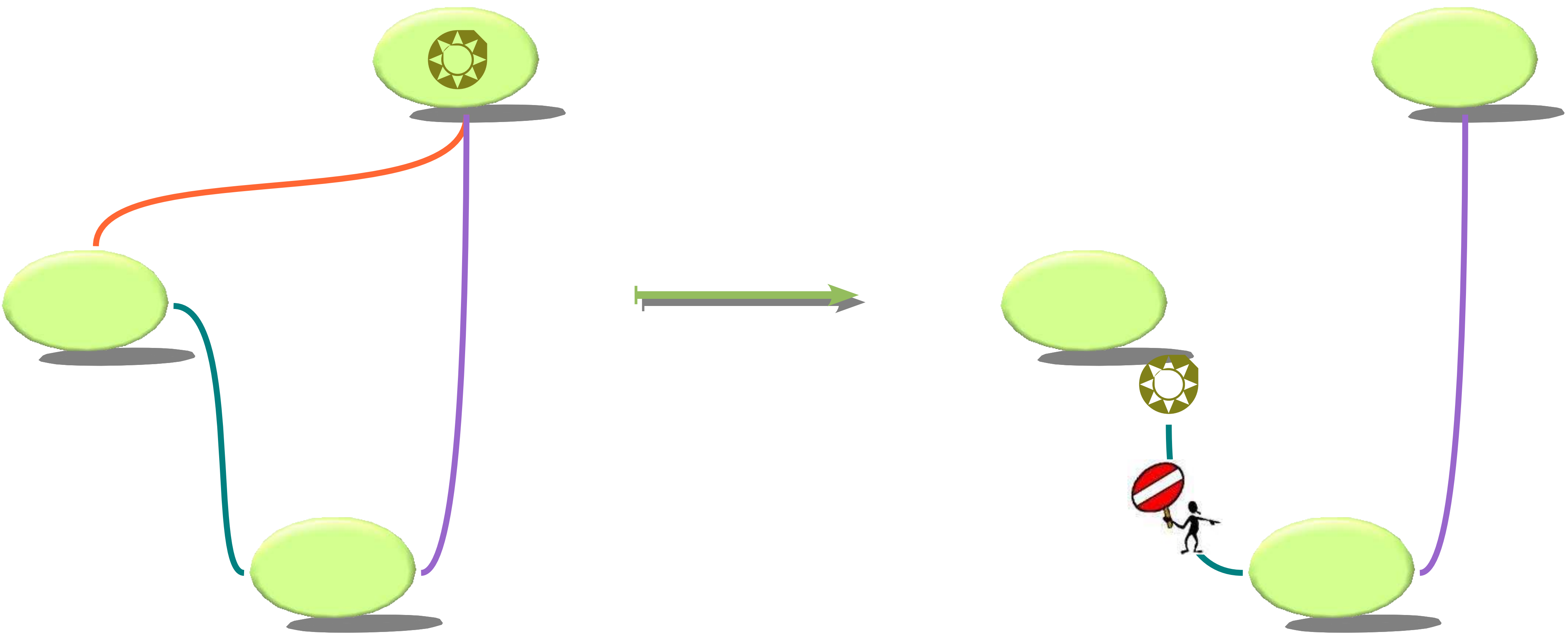
$\mathcal{K}o\mathcal{S}$ ... graphically



$\mathcal{K}o\mathcal{S}$ ... graphically



$\mathcal{K}o\mathcal{S}$ ... graphically



$\mathcal{K}o\mathcal{S}$ syntax

Let $\mathcal{C}$ be the c-semiring of QoS values (ranged over by κ)

$$
\begin{array}{lcl}
N, M & ::= & \mathbf{0} \\
 & | & s :: P \\
 & | & (\nu\, s)N \\
 & | & N \parallel M \\
 & | & s \overset{\kappa}{\frown} t
\end{array}
\qquad
\begin{array}{lcl}
\gamma & ::= & (T) \\
 & | & \langle v_1, \ldots, v_n \rangle \\
 & | & \varepsilon_{\kappa}[P]@t \\
 & | & con_{\kappa}\langle t \rangle \\
 & | & acc_{\kappa}\langle t \rangle \\
 & | & node_{\kappa}\langle t \rangle
\end{array}
$$

$$
\begin{array}{lcl}
P, Q & ::= & \mathbf{nil} \\
 & | & \gamma.P \\
 & | & (\nu\, s)P \\
 & | & P \;|\; Q \\
 & | & !P \\
 & | & \ldots
\end{array}
\qquad
\begin{array}{lcl}
T & ::= & \varepsilon \\
 & | & v \\
 & | & ?x \\
 & | & \neg v \\
 & | & T, T
\end{array}
$$

$\mathcal{K}oS$ Operational Semantics

The semantics of $\mathcal{K}oS$ is defined by the relation

$$N \xrightarrow[\kappa]{\alpha} M$$

which states that N performs α with cost κ and becomes M.

Local transitions (communications, node or link creations) have unitary QoS value, while the only non-trivial QoS values appear on the transitions that spawn processes or show the presence of links.

$\mathcal{K}oS$ Operational Semantics[1]

(PREF) $$s :: \gamma.P \xrightarrow[\mathbf{1}]{\gamma @ s} s :: P,\ \gamma \notin \{node_\kappa\langle t\rangle, con_\kappa\langle s\rangle, acc_\kappa\langle s\rangle\}$$

(CON) $$\frac{N \xrightarrow[\mathbf{1}]{s\ con_\kappa\langle t\rangle} N' \qquad M \xrightarrow[\mathbf{1}]{t\ acc_{\kappa'}\langle s\rangle} M' \qquad \boxed{\mathbf{0} < \kappa \leq \kappa'}}{N \parallel M \xrightarrow[\mathbf{1}]{\tau} N' \parallel M' \parallel s \overset{\kappa}{\frown} t}$$

(COMM) $$\frac{N \xrightarrow[\mathbf{1}]{s\ (T)} N' \qquad M \xrightarrow[\mathbf{1}]{s\ \mathfrak{t}} M' \qquad \bowtie (T, \mathfrak{t}) = \sigma}{N \parallel M \xrightarrow[\mathbf{1}]{\tau} N'\sigma \parallel M'}$$

$\mathcal{K}oS$ Operational Semantics^2

(LINK) $\quad s \overset{\kappa}{\frown} t \xrightarrow[\kappa]{s\ link\ t} \mathbf{0}$

(NODE) $\quad s :: node_\kappa\langle t\rangle.P \xrightarrow[\mathbf{1}]{node\langle t\rangle} s :: P \parallel s \overset{\kappa}{\frown} t \parallel t :: \mathbf{0},\ s \neq t$

(PAR) $$\frac{N \xrightarrow[\kappa]{\alpha} N'}{N \parallel M \xrightarrow[\kappa]{\alpha} N' \parallel M} \text{ if } \begin{cases} \mathrm{bn}(\alpha) \cap \mathrm{fn}(M) = \emptyset \quad \wedge \\ (\mathtt{addr}(N') \setminus \mathtt{addr}(N)) \cap \mathtt{addr}(M) = \emptyset \end{cases}$$

Rule (NODE) allows a process allocated at s to use a name t as the address of a new node and to create a new link from s to t exposing the QoS value κ. The side condition of (PAR) prevents new nodes (and links) to be created by using addresses of existing nodes.

$\mathcal{K}oS$ Operational Semantics[3]

(LEVAL) $\quad s :: \varepsilon_\kappa[Q]@s.P \xrightarrow[\mathbf{1}]{\tau} s :: P \parallel s :: Q$

$$(\text{ROUTE})\quad \frac{N \xrightarrow[\kappa']{r\ \varepsilon^s_\kappa\langle P\rangle@t} N' \qquad M \xrightarrow[\kappa'']{r\ link\ r'} M' \qquad \kappa' \star \kappa'' \leq \kappa}{N \parallel M \xrightarrow[\kappa' \star \kappa'']{r'\ \varepsilon^s_\kappa\langle P\rangle@t} N' \parallel M'}\ \boxed{t \neq r'}$$

$$(\text{LAND})\quad \frac{N \xrightarrow[\kappa']{r\ \varepsilon^s_\kappa\langle P\rangle@t} N' \qquad M \xrightarrow[\kappa'']{r\ link\ t} M' \qquad \kappa' \star \kappa'' \leq \kappa}{N \parallel M \xrightarrow[\kappa' \star \kappa'']{\tau} N' \parallel M' \parallel t :: P}$$

Local spawning is always enabled while $\varepsilon_\kappa[P]@t$ from s is not always possible: the net must contain a path of links from s to t suitable wrt κ.

(ROUTE) states that P can traverse a link go an intermediate node r provided that costs are respected.

(LAND) describes the last hop: in this case, P is spawned at t, provided that the QoS value of the whole path that has been found is lower than κ.

Public links

Links in $\mathcal{K}oS$ are public:

$$N \quad \triangleq \quad s :: \varepsilon_3[P]@t \quad \| \quad s \overset{1}{\frown} r \quad \| \quad r :: con_2\langle t\rangle.\varepsilon_2[Q]@t \quad \| \quad t :: acc_2\langle r\rangle,$$

- s and r are trying to spawn a process on t (but no path to t exists).
- r is aware that a link must be fi rst created (and t agrees on that).

Initially, only (CON) con be applied:

$$N' \quad \triangleq \quad s :: \varepsilon_3[P]@t \quad \| \quad s \overset{1}{\frown} r \quad \| \quad r :: \varepsilon_2[Q]@t \quad \| \quad r \overset{2}{\frown} t \quad \| \quad t :: \mathbf{nil}.$$

$r \overset{2}{\frown} t$ provides now a path (costing 3) from s to t, hence using (PREF), (LINK), (ROUTE) and (LAND) we derive

$$N' \xrightarrow[3]{\tau} s :: \mathbf{nil} \quad \| \quad r :: \varepsilon_2[Q]@t \quad \| \quad t :: P.$$

Noteworthy, the migration of P prevents Q to be spawned because the link created by r has been used by P.

Private links

Private links can be traversed only by those processes having the appropriate “rights”. Access rights are (particular) names.

$$N \triangleq s :: \varepsilon_{\{r,s\}}[P]@t \parallel s \overset{\{r\}}{\frown} s'$$

$$M \triangleq s :: \varepsilon_{\{r,s\}}[P]@t \parallel s \overset{\{r,u\}}{\frown} s'$$

P can traverse the link in N but not the one in M

Access rights c-semiring: $\mathcal{R} = \langle \wp_{\mathrm{fin}}(\mathcal{S}) \cup \{\mathcal{S}\}, glb, \cup, \mathcal{S}, \emptyset \rangle$

$$X \leq Y \iff Y \subseteq X$$

A private link between the nodes s and t can be specified as

$$(\nu\, p)(s :: P \parallel s \overset{\{p\}}{\frown} t \parallel t :: Q)$$

Permanent and stable links

$\mathcal{K}oS$ links are vanishing but permanent links can be easily encoded:

$$s :: !con_{\kappa}\langle t\rangle \parallel t :: !acc_{\kappa'}\langle s\rangle$$

A slight variation are stable links, which are links existing until a given condition is satisfied.

$$Stable_s\ G\ t \ \triangleq\ !con_{\kappa}\langle t\rangle \ \mid\ \varepsilon[While\ G\ do\ acc_{\kappa}\langle s\rangle\ od\ \mathbf{nil}]@t$$

SOC & Transactions

- Long-Running Transactions (LRTs) equip SOAs with the possibility of atomic execution
- LRTs use compensations for undoing the effects of partial executions when the overall orchestration cannot be completed
- Primitives for LRTs have been informally specified in most of the languages for orchestrating WS (WSCL [WSC], BPML [BPM], WSFL [Ley01], XLANG [XLA], BPEL4WS ["BP])

We propose Java Transactional Web Services (JTWS), a set of JAVA APIs supporting coordination of SOA and their transactional composition.
JTWS exploit a signal passing mechanism and consists of

- JSCL: provides the signal handling primitives
- JTL: provides primitives for transactional flows

The JSCL layer allows to specify generic components that

- creates a signal link between two components
- sends a signal to the components
- handles signals received by other components

JTL is defined on top of JSCL and allows to specify transactional flows by composing according

- transactional sequences
- transactional parallel

We have fruitfully exploited JTWS for implementing the Sagas calculus [BMM05].

Modelling SOC with Syncrhonized Hyperedge Replacement

Hypergraphs Programming model[1]

- Edge replacement for graph rewritings [Fed71, Pav72]
- Graphs for distributed systems [CM83, 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]
- Extension to node fusions [FMT01]
- ...

Hypergraphs Programming model^2

Using SHR, we aim at

- defining a uniform framework
- tackling new *non-functional* computational phenomena of SOC

The metaphor is

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

In other words:

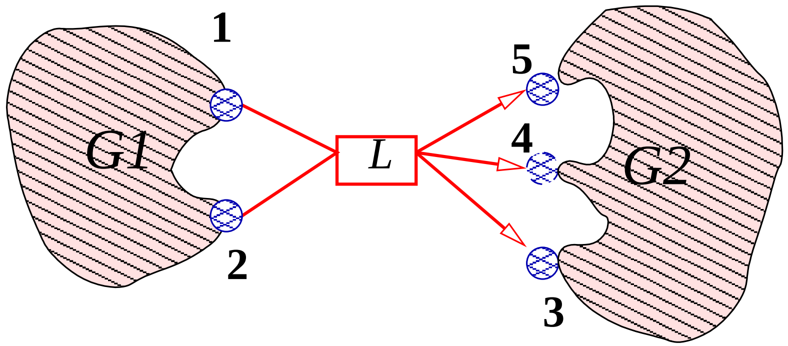
- Components are represented by hyperedges
- Systems are *bunches* of (connected) hyperedges
- Computing means to synchronously rewrite hyperedges...
- ...according to a synchronisation policy

Replacement of Hyperedges

$L \rightarrow G$

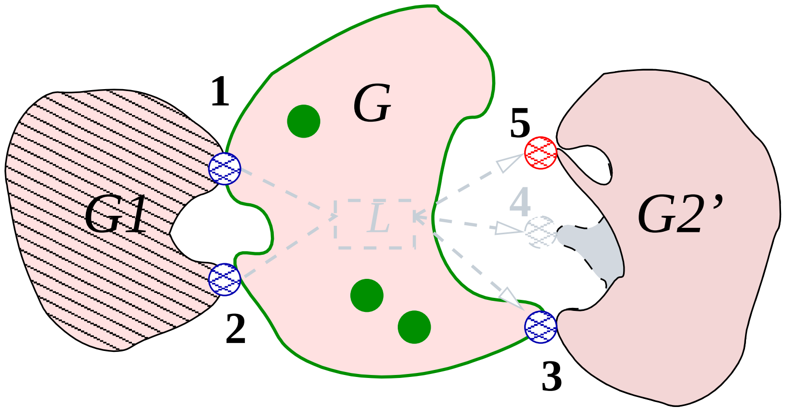
Replacement of Hyperedges

$L \to G$



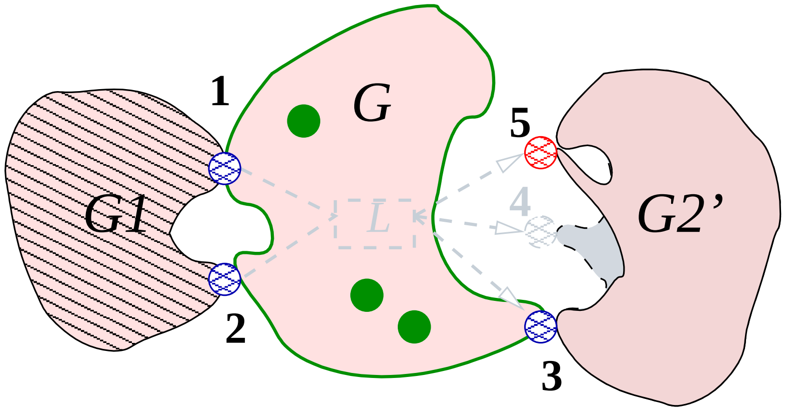
Replacement of Hyperedges

$L \to G$



Replacement of Hyperedges

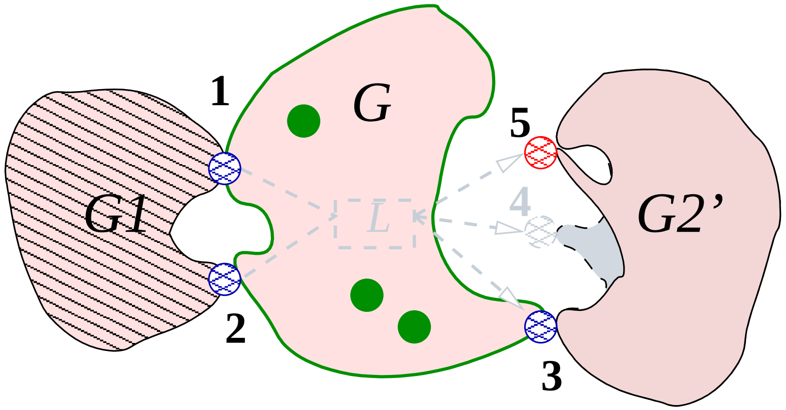
$L \longrightarrow G$



- Edge replacement: local
- Synchronisation as distributed constraint solving
- Multiple synchronisation
- New node creation
- Node fusion: model of mobility and communication

Replacement of Hyperedges

$L \rightarrow G$



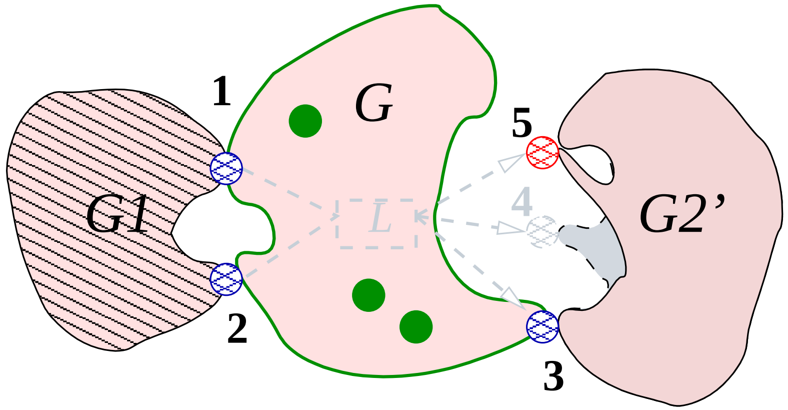
- Edge replacement: local
- Synchronisation as distributed constraint solving
- Multiple synchronisation
- New node creation
- Node fusion: model of mobility and communication

Benefi ts:

- Uniform framework for π, π-I, fusion
- LTS for Ambient ...
- ... for Klaim ...

Replacement of Hyperedges

$L \longrightarrow G$



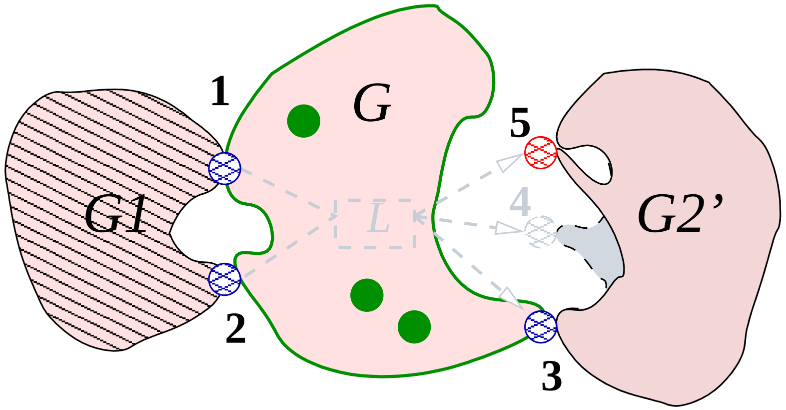
- Edge replacement: local
- Synchronisation as distributed constraint solving
- Multiple synchronisation
- New node creation
- Node fusion: model of mobility and communication

Benefi ts:

- Uniform framework for π, π-I, fusion
- LTS for Ambient ...
- ... for Klaim ...
- ... and *path reservation* for KAOS
- expressive for distributed coordination
- wireless networks

Replacement of Hyperedges

$L \rightarrow G$



- Edge replacement: local
- Synchronisation as distributed constraint solving
- Multiple synchronisation
- New node creation
- Node fusion: model of mobility and communication

Benefi ts:

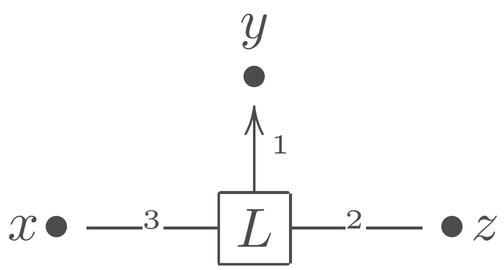
- Uniform framework for π, π-I, fusion
- LTS for Ambient ...
- ... for Klaim ...
- ... and *path reservation* for KAOS
- expressive for distributed coordination
- wireless networks

SHR can combine QoS & sophisticated synchronisations [HT05, LT05, HLT]

Hyperedges and Hypergraphs Syntax

A hyperedge connects more than two nodes (generalisation of edge)

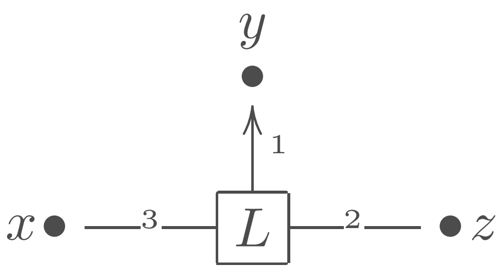
$L : 3,\ L(y, z, x),$



Hyperedges and Hypergraphs Syntax

A hyperedge connects more than two nodes (generalisation of edge)

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

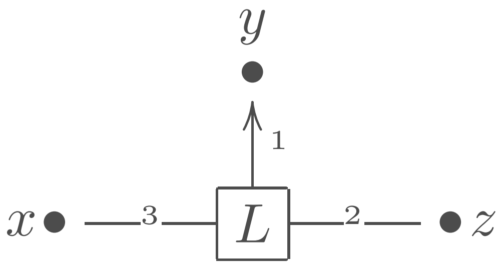


$$G ::= nil \ \big|\ L(\tilde{x}) \ \big|\ G|G \ \big|\ \nu\, y.G$$

Hyperedges and Hypergraphs Syntax

A hyperedge connects more than two nodes (generalisation of edge)

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



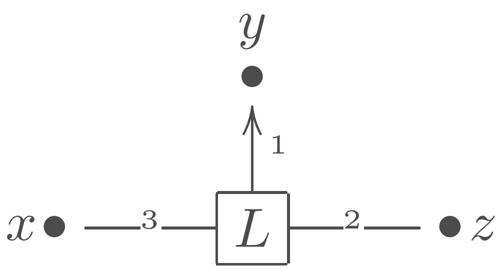
$$G ::= nil \ \big|\ L(\tilde{x}) \ \big|\ G|G \ \big|\ \nu\, y.G$$

Syntactic Judgement $x_1 : s_1, \ldots, x_n : s_n \vdash G$, $fn(G) \subseteq \{x_1, \ldots, x_n\}$

Hyperedges and Hypergraphs Syntax

A hyperedge connects more than two nodes (generalisation of edge)

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



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

Syntactic Judgement $x_1 : s_1, \ldots, x_n : s_n \vdash G,\qquad fn(G) \subseteq \{x_1, \ldots, x_n\}$

An example:

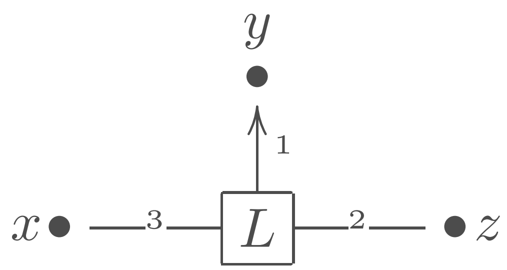
$$L : 3, \quad M : 2$$

$$x : \mathbf{1}, y : \mathbf{0} \vdash \nu\, z.(L(y, z, x)|M(y, z))$$

Hyperedges and Hypergraphs Syntax

A hyperedge connects more than two nodes (generalisation of edge)

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



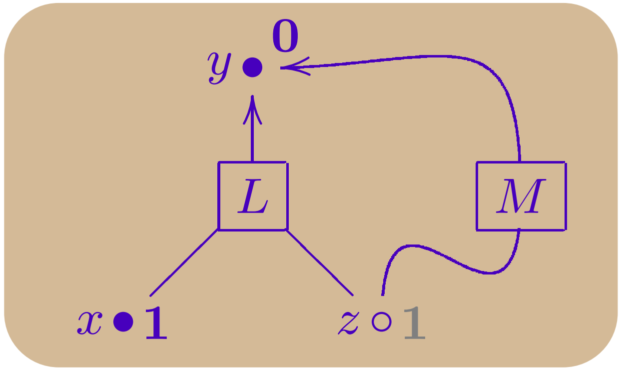
$$G ::= nil \mid L(\tilde{x}) \mid G|G \mid \nu\, y.G$$

Syntactic Judgement $\quad x_1 : s_1, \ldots, x_n : s_n \vdash G, \qquad fn(G) \subseteq \{x_1, \ldots, x_n\}$

An example:

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

$$x : \mathbf{1}, y : \mathbf{0} \vdash \nu\, z.(L(y, z, x) | M(y, z))$$



Productions

Productions of SHReQ are based on requirements: $\mathcal{R} = S \times \mathcal{N}^*$ (where $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ is a fixed c-semiring)

production	$\chi \triangleright L(\tilde{x}) \xrightarrow{\Lambda} G$

Productions of SHReQ are based on requirements: $\mathcal{R} = S \times \mathcal{N}^*$ (where $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ is a fixed c-semiring)

- production $\qquad \chi \triangleright L(\boxed{\tilde{x}}) \xrightarrow{\Lambda} G$

- $\tilde{x}$ is a tuple of pairwise distinguished nodes and $L : |\tilde{x}|$

Productions

Productions of SHReQ are based on requirements: $\mathcal{R} = S \times \mathcal{N}^*$ (where $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ is a fixed c-semiring)

- production $\boxed{\chi} \triangleright L(\tilde{x}) \xrightarrow{\Lambda} G$

- $\tilde{x}$ is a tuple of pairwise distinguished nodes and $L : |\tilde{x}|$
- $\chi : \{|\tilde{x}|\} \rightarrow S$ is the applicability function

Productions

Productions of SHReQ are based on requirements: $\mathcal{R} = S \times \mathcal{N}^*$ (where $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ is a fixed c-semiring)

- production $\chi \triangleright L(\tilde{x}) \xrightarrow{\boxed{\Lambda}} G$

- $\tilde{x}$ is a tuple of pairwise distinguished nodes and $L : |\tilde{x}|$
- $\chi : \{|\tilde{x}|\} \to S$ is the applicability function
- $\Lambda : \{|\tilde{x}|\} \to \mathcal{R}$ is the communication function.
  $\mathrm{n}(\Lambda)$ communicated nodes of Λ: those nodes appearing in a requirement in the range of Λ.
  The set of new nodes of Λ is $\mathrm{new}(\Lambda) = \mathrm{n}(\Lambda) \setminus \mathrm{dom}(\Lambda)$

Productions

Productions of SHReQ are based on requirements: $\mathcal{R} = S \times \mathcal{N}^*$ (where $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ is a fixed c-semiring)

- production $\qquad \chi \triangleright L(\tilde{x}) \xrightarrow{\Lambda} \boxed{G}$

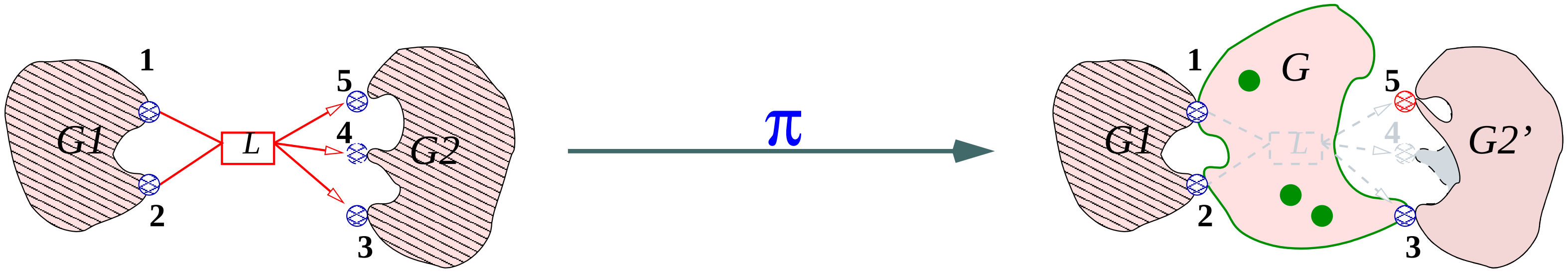
- $\tilde{x}$ is a tuple of pairwise distinguished nodes and $L : |\tilde{x}|$
- $\chi : \{|\tilde{x}|\} \to S$ is the applicability function
- $\Lambda : \{|\tilde{x}|\} \to \mathcal{R}$ is the communication function.
  $\mathrm{n}(\Lambda)$ communicated nodes of Λ: those nodes appearing in a requirement in the range of Λ
  The set of new nodes of Λ is $\mathrm{new}(\Lambda) = \mathrm{n}(\Lambda) \setminus \mathrm{dom}(\Lambda)$
- G is a graph s.t. $\mathrm{fn}(G) \subseteq \{|\tilde{x}|\} \cup \mathrm{n}(\Lambda)$

Interpreting SHReQ productions

Consider

$$\pi : \chi \triangleright L(\tilde{x}) \xrightarrow{\Lambda} G$$

and a graph H having an arc labelled by L, e.g.:

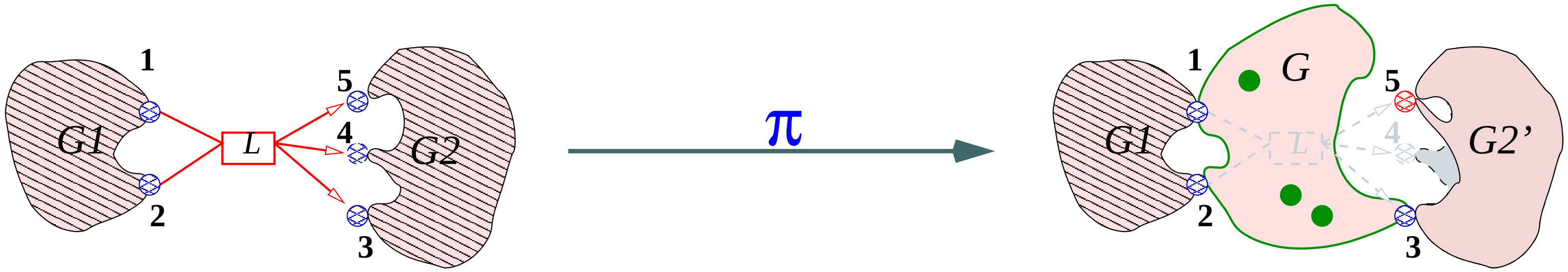


Interpreting SHReQ productions

Consider

$$\pi : \chi \triangleright L(\tilde{x}) \xrightarrow{\Lambda} G$$

and a graph H having an arc labelled by L, e.g.:



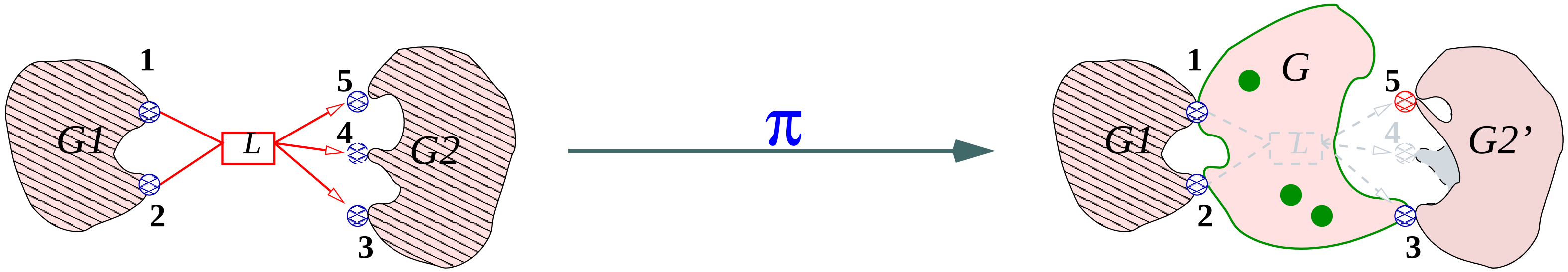
- Replacing L with G in H according to π requires that H satisfies the conditions expressed by χ on the attachment nodes of L.

Interpreting SHReQ productions

Consider

$$\pi : \chi \triangleright L(\tilde{x}) \xrightarrow{\Lambda} G$$

and a graph H having an arc labelled by L, e.g.:



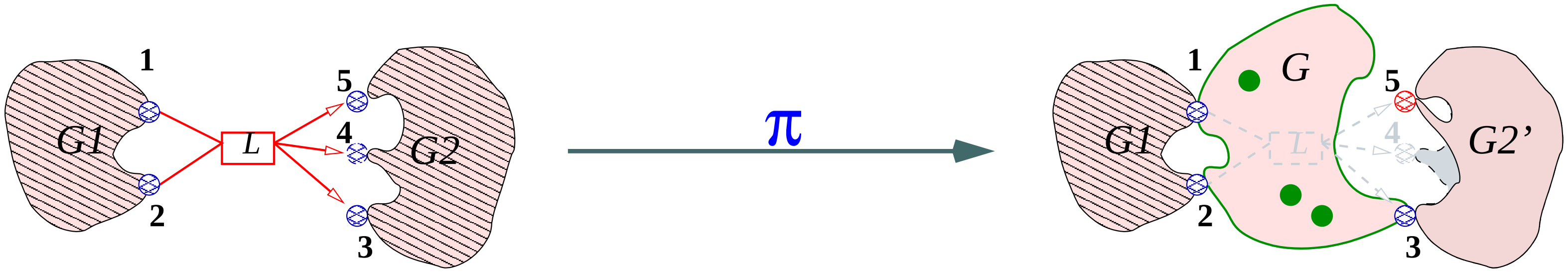
- Replacing L with G in H according to π requires that H satisfies the conditions expressed by χ on the attachment nodes of L.

Interpreting SHReQ productions

Consider

$$\pi : \chi \triangleright L(\tilde{x}) \xrightarrow{\Lambda} G$$

and a graph H having an arc labelled by L, e.g.:



- Replacing L with G in H according to π requires that H satisfies the conditions expressed by χ on the attachment nodes of L.

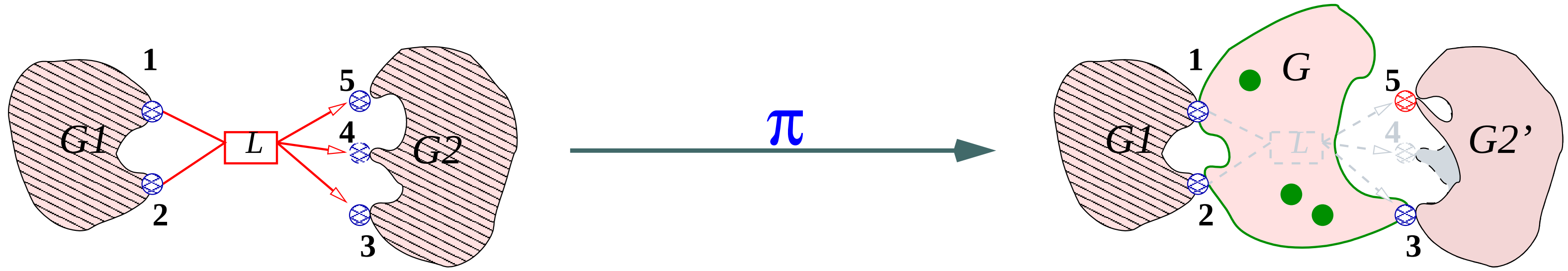
- Once χ is satisfied in H, $L(\tilde{x})$ contributes to the rewriting by offering Λ in the synchronisation with all the edges connected to nodes in $\tilde{x}$.

Interpreting SHReQ productions

Consider

$$\pi : \chi \triangleright L(\tilde{x}) \xrightarrow{\Lambda} G$$

and a graph H having an arc labelled by L, e.g.:



- Replacing L with G in H according to π requires that H satisfies the conditions expressed by χ on the attachment nodes of L.

- Once χ is satisfied in H, $L(\tilde{x})$ contributes to the rewriting by offering Λ in the synchronisation with all the edges connected to nodes in $\tilde{x}$.

Synchronised Rewriting for SHReQ

Events for

Synchronisation $Sync$ and Fin s.t.

- $Sync \subseteq Fin \subseteq S$
- $\mathbf{1} \in Sync$

No synchronisation $NoSync \subseteq S \setminus Fin$ s.t.

- $S \star NoSync \subseteq NoSync$
- $\mathbf{0} \in NoSync$

SHReQ semantics exploits a mgu accounting for node fusions.

Let Ω be a fi nite multiset over $\mathcal{N} \times \mathcal{R}$: $\mathbf{mgu}(\Omega)$ for denoting an idempotent substitutionis defi ned iff

$$(x, s, \tilde{u}), (x, s', \tilde{v}) \in \Omega@x$$

implies

$$|\tilde{u}| = |\tilde{v}|$$

$$\forall i \in \{1, \ldots, |\tilde{u}|\} : \tilde{u}_i \in \text{new}(\Omega) \vee \tilde{v}_i \in \text{new}(\Omega)$$

$$|\Omega@x| > 1 \implies \prod_{(x,s,\tilde{y}) \in \Omega@x} s \notin NoSync$$

Synchronised Rewriting for SHReQ

Events for

Synchronisation Sync and Fin s.t.

- $\mathit{Sync} \subseteq \mathit{Fin} \subseteq S$
- $\mathbf{1} \in \mathit{Sync}$

No synchronisation $\mathit{NoSync} \subseteq S \setminus \mathit{Fin}$ s.t.

- $S \star \mathit{NoSync} \subseteq \mathit{NoSync}$
- $\mathbf{0} \in \mathit{NoSync}$

SHReQ semantics exploits a mgu accounting for node fusions.

Let Ω be a fi nite multiset over $\mathcal{N} \times \mathcal{R}$: $\mathbf{mgu}(\Omega)$ for denoting an idempotent substitutionis defi ned iff

$$(x, s, \tilde{u}), (x, s', \tilde{v}) \in \Omega@x$$

implies

$$|\tilde{u}| = |\tilde{v}|$$

$$\forall i \in \{1, \ldots, |\tilde{u}|\} : \tilde{u}_i \in \mathrm{new}(\Omega) \vee \tilde{v}_i \in \mathrm{new}(\Omega)$$

$$|\Omega@x| > 1 \implies \prod_{(x,s,\tilde{y}) \in \Omega@x} s \notin \mathit{NoSync}$$

Synchronised Rewriting for SHReQ

Events for

Synchronisation Sync and Fin s.t.

- $\mathit{Sync} \subseteq \mathit{Fin} \subseteq S$
- $\mathbf{1} \in \mathit{Sync}$

No synchronisation $\mathit{NoSync} \subseteq S \setminus \mathit{Fin}$ s.t.

- $S \star \mathit{NoSync} \subseteq \mathit{NoSync}$
- $\mathbf{0} \in \mathit{NoSync}$

SHReQ semantics exploits a mgu accounting for node fusions.

Let Ω be a fi nite multiset over $\mathcal{N} \times \mathcal{R}$: $\mathbf{mgu}(\Omega)$ for denoting an idempotent substitutionis defi ned iff

$$(x, s, \tilde{u}), (x, s', \tilde{v}) \in \Omega@x$$

implies

$$|\tilde{u}| = |\tilde{v}|$$

$$\forall i \in \{1, \ldots, |\tilde{u}|\} : \tilde{u}_i \in \mathrm{new}(\Omega) \vee \tilde{v}_i \in \mathrm{new}(\Omega)$$

$$|\Omega@x| > 1 \implies \prod_{(x,s,\tilde{y}) \in \Omega@x} s \notin \mathit{NoSync}$$

and obtained by computing the mgu of the equations

$$\{\tilde{u}_i = \tilde{v}_i \mid \exists s, t \in S : (x, s, \tilde{u}), (x, t, \tilde{v}) \in \Omega \wedge 1 \leq i \leq |\tilde{u}|\}$$

Quasi-productions

The set $\mathcal{QP}$ of quasi-productions on $\mathcal{P}$ is the smallest set s.t. $\mathcal{P} \subseteq \mathcal{QP}$ and

$$\chi \triangleright L(\tilde{x}) \xrightarrow{\Omega} G \in \mathcal{QP} \qquad \wedge \qquad y \in \mathcal{N} \setminus \mathrm{new}(\Omega)$$

$$\Downarrow$$

$$\chi' \triangleright L(\tilde{x}\{^y/_x\}) \xrightarrow{\Omega\{^y/_x\}} G\{^y/_x\} \in \mathcal{QP}$$

where

$$\chi' : \{\!|\tilde{x}|\!\} \setminus \{x\} \cup \{y\} \to S \qquad \chi'(z) = \begin{cases} \chi(z), & z \in \{\!|\tilde{x}|\!\} \setminus \{x, y\} \\ \chi(x) + \chi(y), & z = y \wedge y \in \tilde{x} \\ \chi(x), & z = y \wedge y \notin \{\!|\tilde{x}|\!\} \end{cases}$$

SHReQ rewriting system: $(\mathcal{QP}, \Gamma \vdash G)$

(REN)

$$\frac{\chi \triangleright L(\tilde{x}) \xrightarrow{\Omega} G \in \mathcal{QP} \qquad \rho = \mathbf{mgu}(\Omega) \qquad \bigwedge_{x \in \mathrm{dom}(\chi)} \chi(x) \leq \Gamma(x)}{\Gamma \vdash L(\tilde{x}) \xrightarrow{\Omega} \Gamma_{\Omega} \vdash (\nu\ Z)(G\rho)}$$

(COM)

$$\frac{\Gamma_1 \vdash G_1 \xrightarrow{\Lambda_1} \Gamma_1' \vdash G_1' \qquad \Gamma_2 \vdash G_2 \xrightarrow{\Lambda_2} \Gamma_2' \vdash G_2' \qquad \rho = \mathbf{mgu}(\Lambda_1 \uplus \Lambda_2) \quad \bigwedge_{x \in \mathrm{dom}(\Gamma_1) \cap \mathrm{dom}(\Gamma_2)} \Gamma_1(x) = \Gamma_2(x)}{\Gamma_1 \cup \Gamma_2 \vdash G_1 \mid G_2 \xrightarrow{\Lambda_1 \uplus \Lambda_2} (\Gamma_1 \cup \Gamma_2)_{(\Lambda_1 \uplus \Lambda_2)} \vdash (\nu\ Z)(G_1' \mid G_2')\rho}$$

where $Z = \mathrm{new}(\Omega) \setminus \mathrm{new}(\underline{\Omega})$

Graph transitions

(REN)

$$\frac{\chi \triangleright L(\tilde{x}) \xrightarrow{\Omega} G \in \mathcal{QP} \qquad \rho = \mathbf{mgu}(\Omega) \qquad \bigwedge_{x \in \mathrm{dom}(\chi)} \chi(x) \leq \Gamma(x)}{\Gamma \vdash L(\tilde{x}) \xrightarrow{\Omega} \Gamma_\Omega \vdash (\nu\ Z)(G\rho)}$$

(COM)

$$\frac{\Gamma_1 \vdash G_1 \xrightarrow{\Lambda_1} \Gamma_1' \vdash G_1' \qquad \Gamma_2 \vdash G_2 \xrightarrow{\Lambda_2} \Gamma_2' \vdash G_2' \qquad \rho = \mathbf{mgu}(\Lambda_1 \uplus \Lambda_2) \qquad \bigwedge_{x \in \mathrm{dom}(\Gamma_1) \cap \mathrm{dom}(\Gamma_2)} \Gamma_1(x) = \Gamma_2(x)}{\Gamma_1 \cup \Gamma_2 \vdash G_1 \mid G_2 \xrightarrow{\Lambda_1 \uplus \Lambda_2} (\Gamma_1 \cup \Gamma_2)_{(\Lambda_1 \uplus \Lambda_2)} \vdash (\nu\ Z)(G_1' \mid G_2')\rho}$$

where $Z = \mathrm{new}(\Omega) \setminus \mathrm{new}(\underline{\Omega})$

Induced communication functions

Let $\rho = \mathbf{mgu}(\Omega)$. The communication function induced by Ω is the function $\underline{\Omega} : \mathrm{dom}(\Omega) \to \mathcal{R}$ defi ned as

$$\underline{\Omega}(x) = \begin{cases} (t, \tilde{y}\rho), & t = \prod_{(x,s,\tilde{y}) \in \Omega @ x} s \quad \notin \ Sync \\ (t, \langle\rangle), & t = \prod_{(x,s,\tilde{y}) \in \Omega @ x} s \quad \in \ Sync \end{cases}$$

Basically, $\underline{\Omega}(x)$ yields the synchronisation of requirements in $\Omega @ x$ according to the c-semiring product.

The weighting function induced by Γ and Ω is

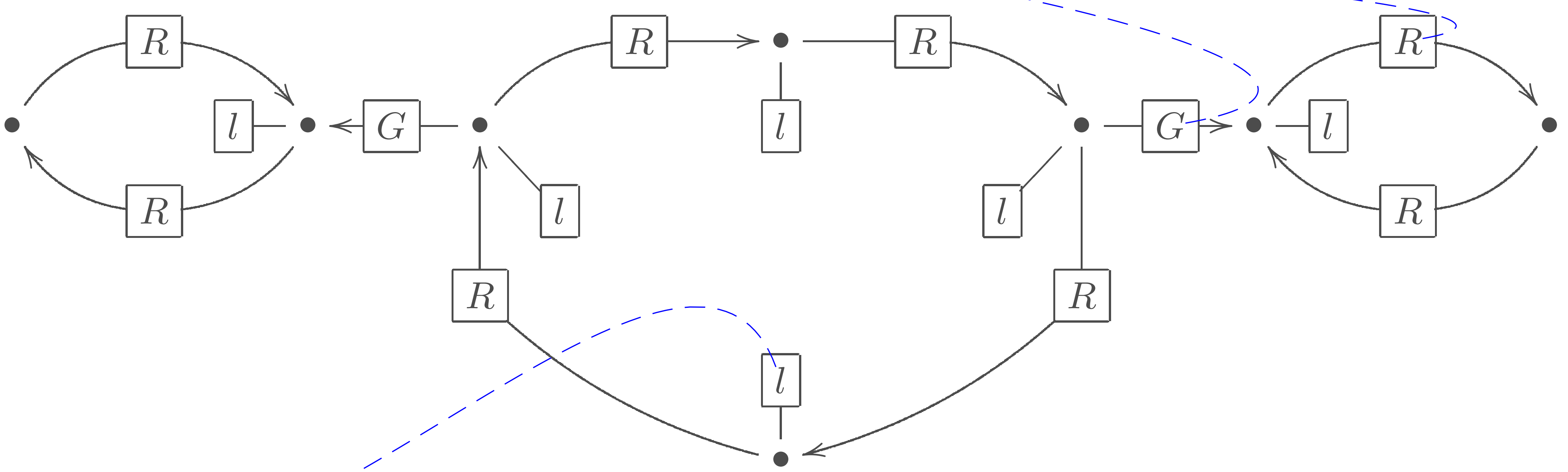
$$\Gamma_\Omega : \mathrm{dom}(\Gamma) \to S,$$

$$\Gamma_\Omega(x) = \begin{cases} \mathbf{1}, & x \in \mathrm{new}(\underline{\Omega}) \\ \Gamma(x), & |\Omega @ x| = 1 \\ \boxed{\Gamma_\Omega(x) = \underline{\Omega}(x)\downarrow_1}, & \text{otherwise} \end{cases}$$

The weighting function computes the new weights of graphs after the synchronisations induced by Ω.

QoS & Synchronisations in SHReQ

A network of rings consists of “rings” of different sizes connected by gates.



$\imath$-edges avoid new gates to be attached on the node they insist on. (e.g., above, only the 2-rings can create (gates to) new rings).

The nodes with no $\imath$-edges, can be used to generate new rings and will be weighted by the amount of available resource.

C-semirings for the ring case study

The c-semiring for networks of rings is $\mathfrak{H}\mathfrak{R}$ given by the cartesian product of the Hoare synchronisations c-semiring $\mathfrak{H} = \langle \mathcal{H}, +_H, \star_H, \mathbf{0}_H, \mathbf{1}_H \rangle$, where

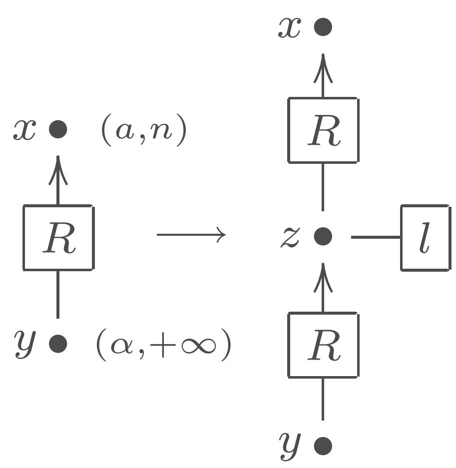
$$\mathcal{H} = \{a, b, c, \mathbf{1}_H, \mathbf{0}_H, \perp\}$$

and the $\mathfrak{R} = \langle \omega_\infty, max, min, 0, +\infty \rangle$.
The idea is that

- $\mathfrak{H}$ coordinates the network rewritings
- $\mathfrak{R}$ handles resource availability
- the initial graph is a ring
- non-limited nodes have weights $(\mathbf{1}_H, u)$ where value u is the maximal amount of available resource
- the limited nodes created during ring evolution are weighted with $(b, +\infty)$ which is constantly maintained.

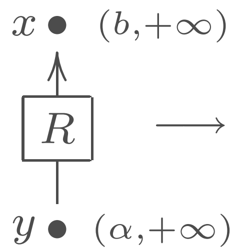
Productions for the ring case study

Create Brother $(n < u)$



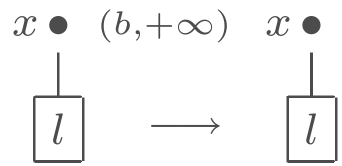
$$x:(\mathbf{0}_H,u),y:\mathbf{0} \triangleright R(x,y) \xrightarrow{\substack{x \,\mapsto\, (a,\,n)\\ y \,\mapsto\, (\alpha,\,+\infty)}} R(x,z) \mid R(z,y) \mid l(z)$$

Accept Syncrhonisation R



$$x:(\mathbf{0}_H,+\infty),y:\mathbf{0} \triangleright R(x,y) \xrightarrow{\substack{x \,\mapsto\, (b,\,+\infty)\\ y \,\mapsto\, (\alpha,\,+\infty)}} R(x,y)$$

Accept Syncrhonisation l

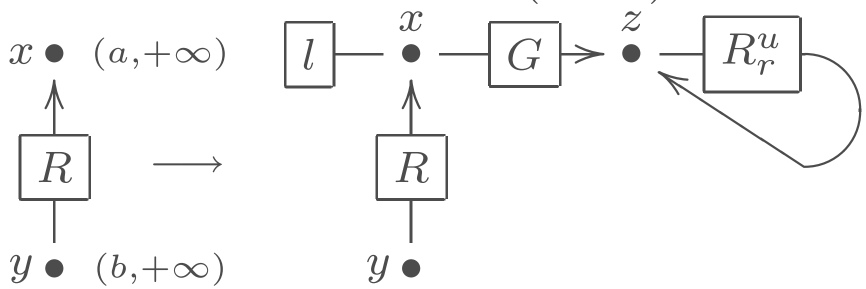


$$x:\mathbf{0} \triangleright l(x) \xrightarrow{x\mapsto(b,+\infty)} l(x)$$

where $\alpha \in \{a, b\}$

Productions for the ring case study[2]

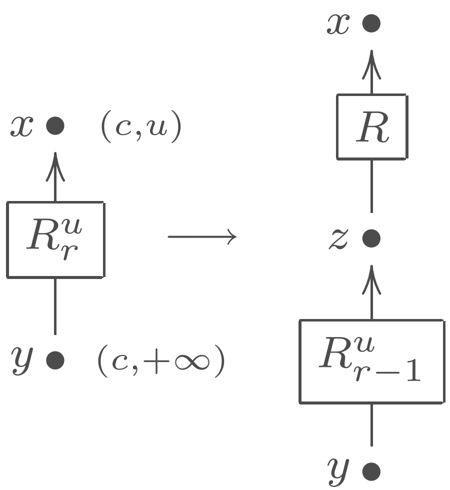
Create Gate $(r>0)$



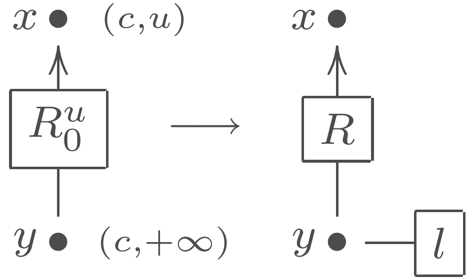
$$x:(\mathbf{0}_H,u),y:\mathbf{0} \triangleright R(x,y) \xrightarrow{\substack{x\,\mapsto\,(a,\,+\infty)\\ y\,\mapsto\,(b,\,+\infty)}} R(x,y) \mid l(x) \mid G(z,x) \mid R^u_r(z,z)$$

Init Ring

$(r>0)$



$$x:\mathbf{0},y:\mathbf{0} \triangleright R^u_r(x,y) \xrightarrow{\substack{x\,\mapsto\,(c,\,u)\\ y\,\mapsto\,(c,\,+\infty)}} R(x,z) \mid R^u_{r-1}(z,y)$$

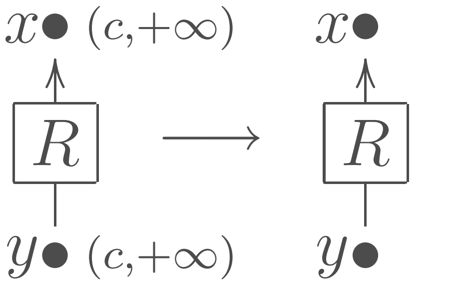


$$x:\mathbf{0},y:\mathbf{0} \triangleright R^u_0(x,y) \xrightarrow{\substack{x\,\mapsto\,(c,\,u)\\ y\,\mapsto\,(c,\,+\infty)}} R(x,y) \mid l(y)$$

where $\beta \in \{b, c\}$

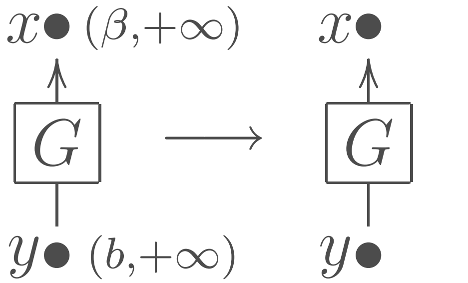
Productions for the ring case study[3]

Accept Synchronisation Init



$$x : \mathbf{0}, y : \mathbf{0} \triangleright R(x,y) \xrightarrow[y \,\mapsto\, (c,\,+\infty)]{x \,\mapsto\, (c,\,+\infty)} R(x,y)$$

Accept Synchronisation Gate

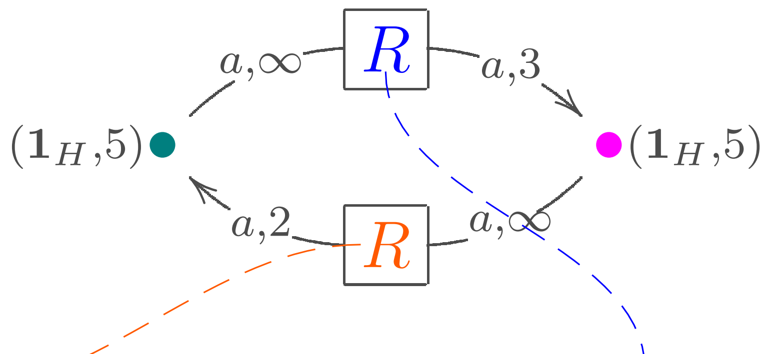


$$x : \mathbf{0}, y : \mathbf{0} \triangleright G(x,y) \xrightarrow[y \,\mapsto\, (b,\,+\infty)]{x \,\mapsto\, (\beta,\,+\infty)} G(x,y)$$

where $\beta \in \{b, c\}$

SHReQ for the ring case study

The derivation starts from a 2-ring components with resource value 5.

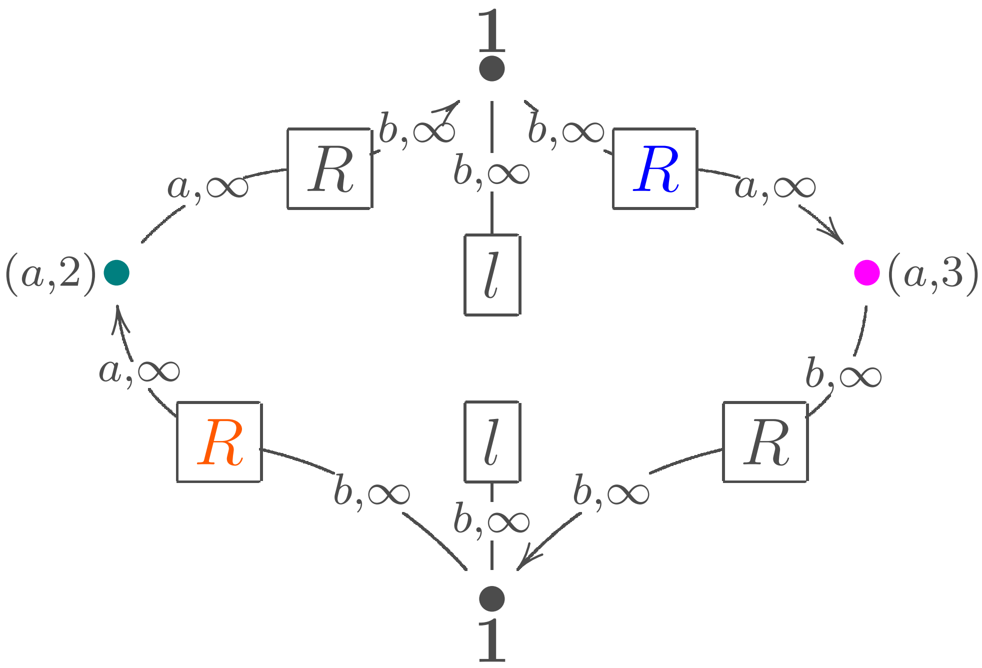


$$\textbf{CreatBrother}(u=5,n=2)\times\textbf{CreatBrother}(u=4,n=3)$$

R chooses production **Create Brother** $u = 5$ (satisfying condition $5 \leq 5$) and $n = 2$ while R chooses $u = 4$ (satisfying condition $4 \leq 5$) and $n = 3$. The resulting synchronisation produces the new weights for the nodes as,

$$(a,2) = (a,2) \star (a,+\infty) = (a \star_H a, \min(2,+\infty))$$
$$(a,3) = (a,3) \star (a,+\infty) = (a \star_H a, \min(3,+\infty)).$$

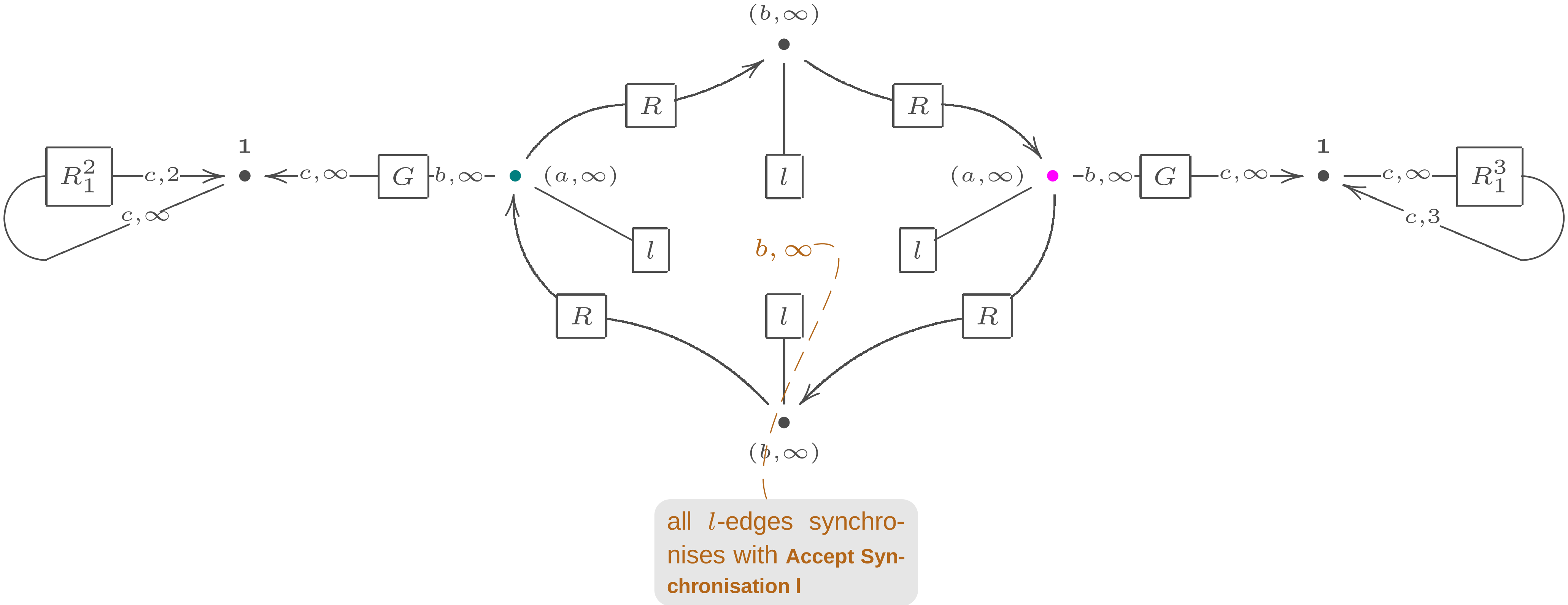
SHReQ for the ring case study



$$\mathbf{CreatGate}(r = 1, u = 2) \times \mathbf{CreatGate}(r = 1, u = 3) \times \mathbf{Accept}^*$$

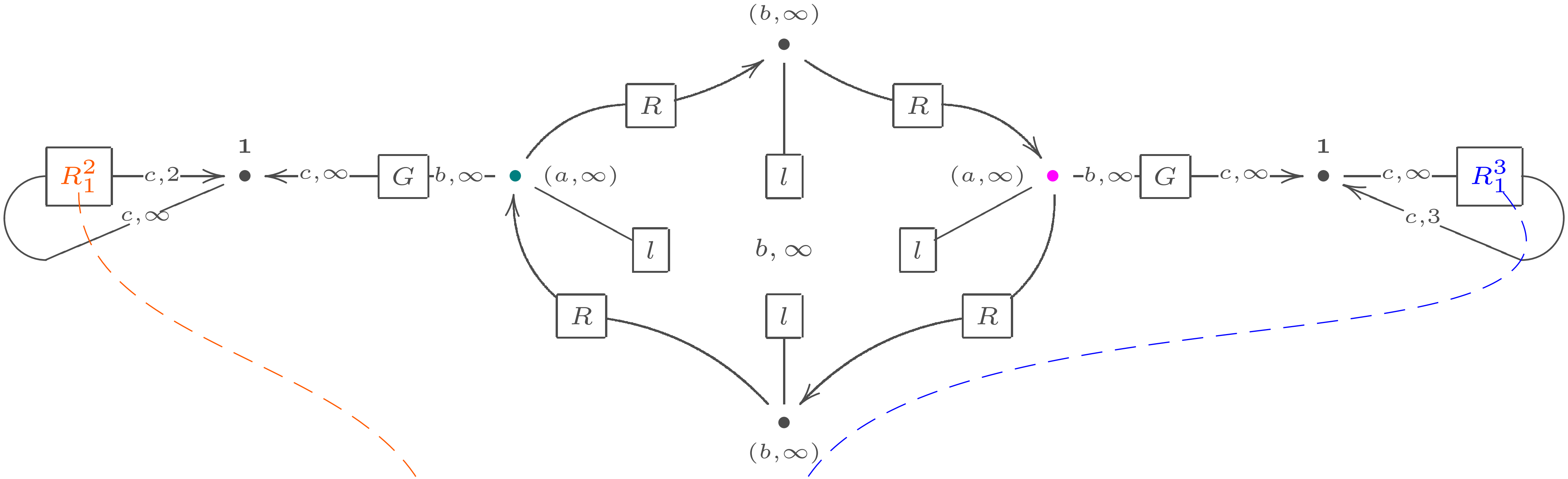
Only R and R can create brothers or gates and they use the remaining resources to create gates to two 2-rings ($r = 1$); the other edges apply the **Accept** productions.

SHReQ for the ring case study



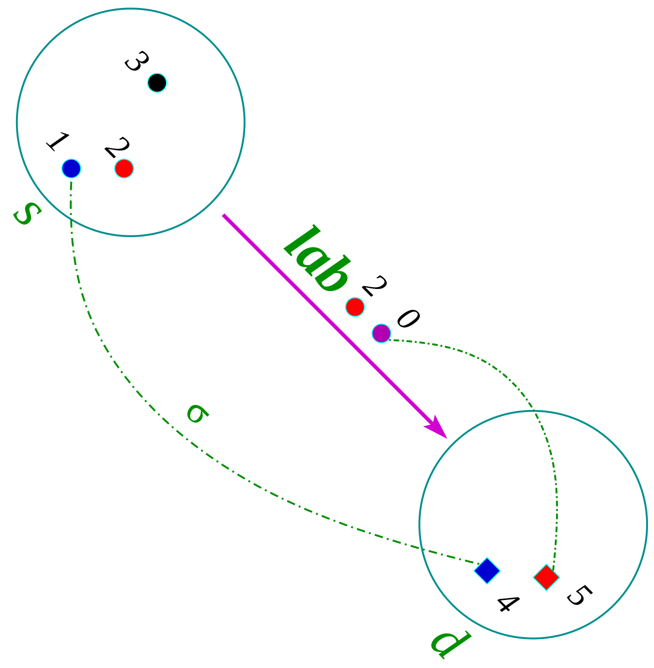
Note that . and . now are internal.

SHReQ for the ring case study



$\textbf{InitRing}(r=1, u=2) \times \textbf{InitRing}(r=1, u=3) \times \textbf{Accept}^*$

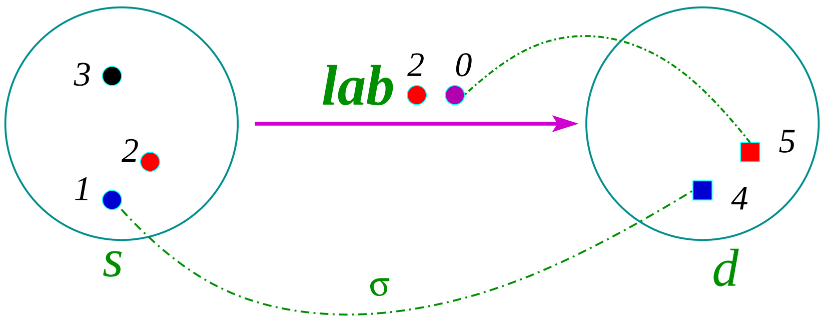
History Dependent Automata



HD-automata

- HD-automata as an operational model of *history-dependent* calculi [Pis99, MP98, MP00, FMP02]
- allow a finite representation of classes of infinite LTS
- states and transitions equipped with names:
  - names no longer dealt as syntactic components: they become explicit in the operational model
  - as a consequence...HD-automata model name creation/deallocation or name extrusion
- names of HD-automata are local...
- ...hence a mechanism for describing how names correspond each other along transitions is required
- so that, a “history” of names in the computation can be determined

HD-automata: an intuition



- The transition is labelled by lab and exposes names 2 (of s) and a fresh name 0
- State s has three names: 1, 2 and 3
- State d has two names: 4 and 5
- 4 correspond to 1 and 5 to the new name 0
- Notice that names 3 in s is “discharged” along such transition

A HD-automaton associates a “history” to names of the states appearing in the computation: it is possible to reconstruct the associations that lead to the state containing the name.

If a state is reached in two different computations, different histories could be assigned to its names.

HD-automata Foundations

- HD-automata with symmetries are based on permutation algebras [MP00]
- In [FMT05a] a type-theoretic definition of HD-automata in terms of a polymorphic lambda calculus ($\lambda^{\rightarrow,\Pi,\Sigma}$) has been given
- Dependent types formally state the relationships between the different components of HD-automata...
- ...and have been exploited for implementing Mihda: a partition refinement algorithm over HD-automata (introduced in [FMP02])
- Mihda minimizes HD-automata representing π-calculus agents (wrt early bisimilation)
- HD-automata and Mihda have been also used for modelling Fusion calculus [FMT$^+$05c]

Named sets

Named sets are used for representing the states of HD-automata. Basically, a named set is a set whose elements are equipped with a fi nite set of names and a symmetry

Definition 1 *A named set is a structure* $\langle Q, |\_|, \mathcal{G}\rangle$ *such that*

- $\eta : Q \to \wp_{\mathit{fin}}(\mathcal{N})$
- $\mathcal{G}$ *is a function on* Q *such that, for any* $q \in Q$*,* $\mathcal{G}(q)$ *is a group of permutations of* $\eta(q)$

Given a named set A, we write Q_A, $|\_|_A$ and $\mathcal{G}_A$ for denoting the components of A

Example 4 *Consider the* π*-calculus agent*

$$A(x, y) \stackrel{\triangle}{=} (\nu\, z)(\bar{x}z.P + \bar{y}z.P).$$

A state $q_A \in NS_A$ *of a named set representing* $A(x, y)$ *has two local names (namely,* $|q_A| = \{x, y\}$*). The symmetry of* q_A *consists of the identity permutations and the permutation that exchanges* x *with* y*.*

Named functions

Transitions among states are represented by means of named functions:

Definition 2 *Let S and D be two named sets. A named function is a pair $\langle h : S \rightarrow D, \Sigma \rangle$ such that h is a function from Q_S to Q_D and for all $q \in S$, $\Sigma(q)$ yields a finite set of functions from $|h(q)|_D$ to names in $|q|_S$ of to the distinguished name $\star$ such that*

1. $\forall \sigma \in \Sigma(q).\mathcal{G}_D(h(q)); \sigma = \Sigma(q)$,
2. $\forall \sigma \in \Sigma(q).\sigma; \mathcal{G}_S(q) \subseteq \Sigma(q)$,
3. *any function of $\Sigma(q)$ is injective.*

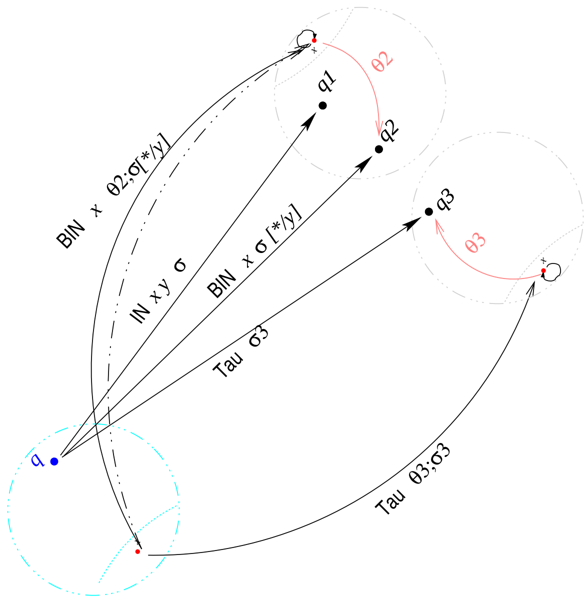
Given a named function $H = \langle h : S \rightarrow D, \Sigma \rangle$ we write

- $\mathrm{dom}(H) = S$,
- $\mathrm{cod}(H) = D$,
- $\mathrm{h}_H = h$,
- $\Sigma_H = \Sigma$,

Definition 3 *Let H, K be two named functions. We say that H and K can be composed iff $\mathrm{cod}(H) = \mathrm{dom}(K)$. In this case, the composition of H and K is the named function $H; K$ such that*

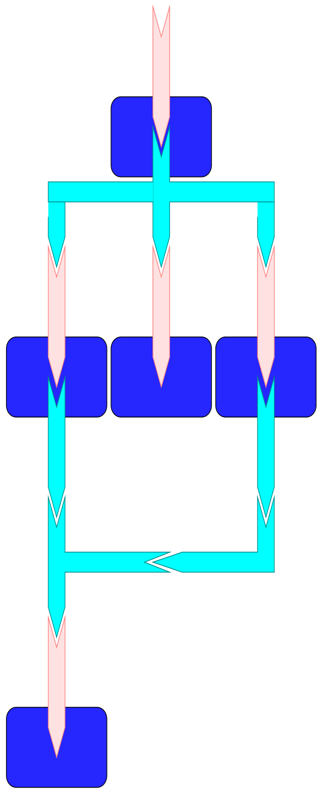
- $\mathrm{dom}(H;K) = \mathrm{dom}(H)$,
- $\mathrm{cod}(H;K) = \mathrm{cod}(K)$, $\mathrm{h}_{H;K} = \mathrm{h}_H; \mathrm{h}_K$ *and*
- $\Sigma_{H;K} = \lambda q \in \mathrm{dom}(H;K).\Sigma_K(\mathrm{h}_H(q)); \Sigma_H(q)$

Verification techniques for SOC



Q: What is interesting in a SOA computation?

A: No precise answer...our interpretation is



Systems for SOC are obtained by gluing services that are distributed and evolve "together" (causality, parallelism,...)

Systems evolve both "in time" and "in space". Time evolution is usually interpreted as the dynamic of a system, while spatial evolution corresponds to structural reconfiguration of systems.

SHReQ has been recently equipped with a spatio-temporal logic interpreted over c-semirings [HLT].

Let $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ be a fixed c-semiring and $\mathcal{G}$ the set of weighted graphs.

$$
\begin{array}{rcll}
\phi & ::= & nil \mid \Gamma(\xi) \mid \mathfrak{L}(\tilde{\xi}) \mid \phi|\phi \mid \phi\|\phi & \text{spatial operators} \\
& \mid & f(\phi, \ldots, \phi) & \text{c-semiring operators} \\
& \mid & \sum_u \phi \mid \prod_u \phi & \text{node quantification} \\
& \mid & u = v & \text{node equality} \\
& \mid & [\sum] \phi \mid [\prod] \phi & \text{temporal operator} \\
& \mid & \mathfrak{r}(\tilde{\xi}) \mid (\mu \mathfrak{r}(\tilde{u}).\phi)\tilde{\xi} \mid (\nu \mathfrak{r}(\tilde{u}).\phi)\tilde{\xi} & \text{fixpoints}
\end{array}
$$

where $\mathfrak{L} \subseteq \mathcal{L}$ is a finite set of labels, ξ is a metavariable for nodes or node variables and f is an operation on the fixed c-semiring values.

A logic for SHReQ[2]

Fixed a SHReQ rewriting system $(\mathcal{QP}, \Gamma \vdash G)$, we interpret formulae as maps $\mathcal{G} \to S$.
Let σ be a map from node variables to nodes and ρ be a map from recursion variables to functions $\mathcal{G} \to S$.

$$
\begin{array}{rcl}
[\![\ nil\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & G \equiv nil \\
[\![\ \Gamma(\xi)\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & (\xi\sigma \in \mathrm{n}(G)) \star \Gamma(\xi\sigma) \\
[\![\ \mathfrak{L}(\tilde{\xi})\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & (G \equiv L(\tilde{\xi}\sigma)) \star (L \in \mathfrak{L}) \\
[\![\ \phi_1|\phi_2\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \sum_{(G_1,G_2)\in\Theta(G)}\{[\![\ \phi_1\ ]\!]_{\sigma;\rho}(\Gamma \vdash G_1) \star [\![\ \phi_2\ ]\!]_{\sigma;\rho}(\Gamma \vdash G_2)\} \\
[\![\ \phi_1\|\phi_2\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \prod_{(G_1,G_2)\in\Theta(G)}\{[\![\ \phi_1\ ]\!]_{\sigma;\rho}(\Gamma \vdash G_1) + [\![\ \phi_2\ ]\!]_{\sigma;\rho}(\Gamma \vdash G_2)\} \\
[\![\ f(\phi_1,\ldots,\phi_n)\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & f([\![\ \phi_1\ ]\!]_{\sigma;\rho}(\Gamma \vdash G),\ldots,[\![\ \phi_n\ ]\!]_{\sigma;\rho}(\Gamma \vdash G)) \\
[\![\ \sum_u \phi\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \sum_{x\in\mathrm{n}(G)} [\![\ \phi\ ]\!]_{\sigma[^x/_u];\rho}(\Gamma \vdash G) \\
[\![\ \prod_u \phi\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \prod_{x\in\mathrm{n}(G)} [\![\ \phi\ ]\!]_{\sigma[^x/_u];\rho}(\Gamma \vdash G) \\
[\![\ \xi = \xi'\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \xi\sigma = \xi'\sigma \\
[\![\ [\sum]\,\phi\ ]\!](\Gamma \vdash G) & = & \sum_{\Gamma\vdash G \xrightarrow{\Lambda} \Gamma'\vdash G'} [\![\ \phi\ ]\!](\Gamma' \vdash G') \\
[\![\ [\prod]\,\phi\ ]\!](\Gamma \vdash G) & = & \prod_{\Gamma\vdash G \xrightarrow{\Lambda} \Gamma'\vdash G'} [\![\ \phi\ ]\!](\Gamma' \vdash G') \\
[\![\ \mathfrak{r}(\tilde{\xi})\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \mathfrak{r}\rho(\tilde{\xi}\sigma) \\
[\![\ (\mu\mathfrak{r}(\tilde{u}).\phi)\tilde{\xi}\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \mathit{lfp}(\lambda\mathfrak{r}'.\lambda\tilde{v}.[\![\ \phi\ ]\!]_{\sigma[^{\tilde{v}}/_{\tilde{u}}],\rho[^{\mathfrak{r}'}/_{\mathfrak{r}}]})(\tilde{\xi}\sigma)(\Gamma \vdash G) \\
[\![\ (\nu\mathfrak{r}(\tilde{u}).\phi)\tilde{\xi}\ ]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \mathit{gfp}(\lambda\mathfrak{r}'.\lambda\tilde{v}.[\![\ \phi\ ]\!]_{\sigma[^{\tilde{v}}/_{\tilde{u}}],\rho[^{\mathfrak{r}'}/_{\mathfrak{r}}]})(\tilde{\xi}\sigma)(\Gamma \vdash G)
\end{array}
$$

A logic for SHReQ[2]

Fixed a SHReQ rewriting system $(\mathcal{QP}, \Gamma \vdash G)$, we interpret formulae as maps $\mathcal{G} \to S$.
Let σ be a map from node variables to nodes and ρ be a map from recursion variables to functions $\mathcal{G} \to S$.

$$
\begin{array}{rcl}
[\![\, nil \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & G \equiv nil \\
[\![\, \Gamma(\xi) \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & (\xi\sigma \in \mathrm{n}(G)) \star \Gamma(\xi\sigma) \\
[\![\, \mathfrak{L}(\tilde{\xi}) \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & (G \equiv L(\tilde{\xi}\sigma)) \star (L \in \mathfrak{L}) \\
[\![\, \phi_1 | \phi_2 \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \sum_{(G_1,G_2) \in \Theta(G)} \{ [\![\, \phi_1 \,]\!]_{\sigma;\rho}(\Gamma \vdash G_1) \star [\![\, \phi_2 \,]\!]_{\sigma;\rho}(\Gamma \vdash G_2) \} \\
[\![\, \phi_1 \| \phi_2 \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \prod_{(G_1,G_2) \in \Theta(G)} \{ [\![\, \phi_1 \,]\!]_{\sigma;\rho}(\Gamma \vdash G_1) + [\![\, \phi_2 \,]\!]_{\sigma;\rho}(\Gamma \vdash G_2) \} \\
[\![\, f(\phi_1, \ldots, \phi_n) \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & f([\![\, \phi_1 \,]\!]_{\sigma;\rho}(\Gamma \vdash G), \ldots, [\![\, \phi_n \,]\!]_{\sigma;\rho}(\Gamma \vdash G)) \\
[\![\, \sum_u \phi \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \sum_{x \in \mathrm{n}(G)} [\![\, \phi \,]\!]_{\sigma[^x/_u];\rho}(\Gamma \vdash G) \\
[\![\, \prod_u \phi \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \prod_{x \in \mathrm{n}(G)} [\![\, \phi \,]\!]_{\sigma[^x/_u];\rho}(\Gamma \vdash G) \\
[\![\, \xi = \xi' \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \xi\sigma = \xi'\sigma \\
[\![\, [\sum] \, \phi \,]\!](\Gamma \vdash G) & = & \sum_{\Gamma \vdash G \xrightarrow{\Lambda} \Gamma' \vdash G'} [\![\, \phi \,]\!](\Gamma' \vdash G') \\
[\![\, [\prod] \, \phi \,]\!](\Gamma \vdash G) & = & \prod_{\Gamma \vdash G \xrightarrow{\Lambda} \Gamma' \vdash G'} [\![\, \phi \,]\!](\Gamma' \vdash G') \\
[\![\, \mathfrak{r}(\tilde{\xi}) \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \mathfrak{r}\rho(\tilde{\xi}\sigma) \\
[\![\, (\mu\mathfrak{r}(\tilde{u}).\phi)\tilde{\xi} \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \mathit{lfp}(\lambda\mathfrak{r}'.\lambda\tilde{v}.[\![\, \phi \,]\!]_{\sigma[^{\tilde{v}}/_{\tilde{u}}],\rho[^{\mathfrak{r}'}/_{\mathfrak{r}}]})(\tilde{\xi}\sigma)(\Gamma \vdash G) \\
[\![\, (\nu\mathfrak{r}(\tilde{u}).\phi)\tilde{\xi} \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \mathit{gfp}(\lambda\mathfrak{r}'.\lambda\tilde{v}.[\![\, \phi \,]\!]_{\sigma[^{\tilde{v}}/_{\tilde{u}}],\rho[^{\mathfrak{r}'}/_{\mathfrak{r}}]})(\tilde{\xi}\sigma)(\Gamma \vdash G)
\end{array}
$$

$\mathbf{nil}$ characterises graphs with no edges, $\Gamma(\xi)$ yields the weight of ξ, $L(\tilde{\xi})$ states that an edge L attached to $\tilde{\xi}$ exists s.t. $L \in \mathfrak{L}$. $\phi_1|\phi_2$ sums up all the values of ϕ_i on all decompositions. The temporal operator $[\sum] \, \phi$ sums the values of ϕ after one transition (and similarly for $[\sum] \, \phi$).

Applying SHReQ logic

- Path existence:

$$\textbf{path}(u,v,\mathfrak{L}) \equiv \mu\mathfrak{r}(u,v).(u=v) \; + \sum_{w} \mathfrak{L}(u,w)|\mathfrak{r}(w,v)$$

- Ring membership:

$$\textbf{ring}(u,v) \equiv \textbf{path}(u,v,\{R\}) \mid \textbf{path}(v,u,\{R\})$$

- Highest availability:

$$\sum u.\Gamma(u) \star \neg(\{l\}(u) \mid \mathbf{1})$$

(on a ring evaluates to the maximum over the weights of non limited nodes)

- Inspecting new rings:

$$\begin{aligned} \textbf{resource} \quad &\equiv \quad \sum_{w,v}.(\neg(\{R_0^u\}(w,v) \mid \mathbf{1})) + ((\{R_0^u\}(w,v) \mid \mathbf{1}) \star && (4) \\ & \left[\sum\right] (((\{R\}(w,v) \mid \mathbf{1}) \star \Gamma(w))) \star (\mathbf{0}_H, +\infty)) && (5) \end{aligned}$$

(4) looks for R_0^u-edge, and, after the next rewriting step, (5) selects resource of the newly introduce R-edge.

Verification with HD-automata

- SHReQ is amenable of model checking (forthcoming)
- However, properties of systems can be also stated in terms of equivalences: see [TV04] for spatial properties and [FMT05b] for a more general discussion
- HD-automata can be minimised through a partition refinement algorithm
- the co-algebraic specification of HD-automata and the minimisation algorithm is independent of the chosen language or equivalence
  - π-calculus & early bisimulation
  - Fusion calculus & hyperbisimulation

Minimising HD-automata

The minimisation algorithm builds the minimal realisation $\bar{H}$ of (fi nite) HD-automata by constructing (approximations of) the fi nal coalgebra morphism. The active names of each state q are those in the ranges of $\Sigma_{\bar{H}}(q)$.
Given a T-coalgebra $K : A \to T_1(A)$ (i.e., a HD-automata) on named set A, the minimisation algorithm is specifi ed in a declarative way by the equations

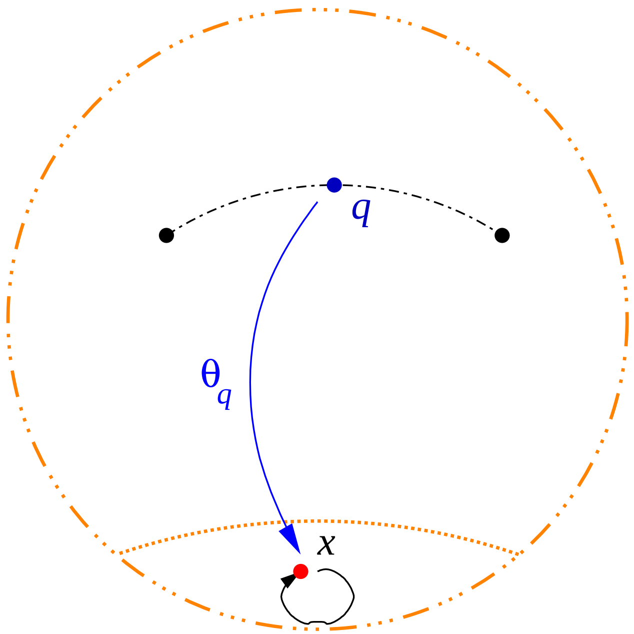
Initial approximation:	$H_0 : \langle q \mapsto \bot,\ \Sigma : q \mapsto \emptyset \rangle$	(6)
Iterative construction:	$H_{i+1} \triangleq K; T_2(H_i).$	(7)

Intuitively, in the starting phase of the algorithm, all the states of automaton K are considered equivalent. At the $(i+1)$-th iteration, the image through T_2 of the i-th iteration is composed with K as prescribed in (7).
At each iteration, two cases can arise:

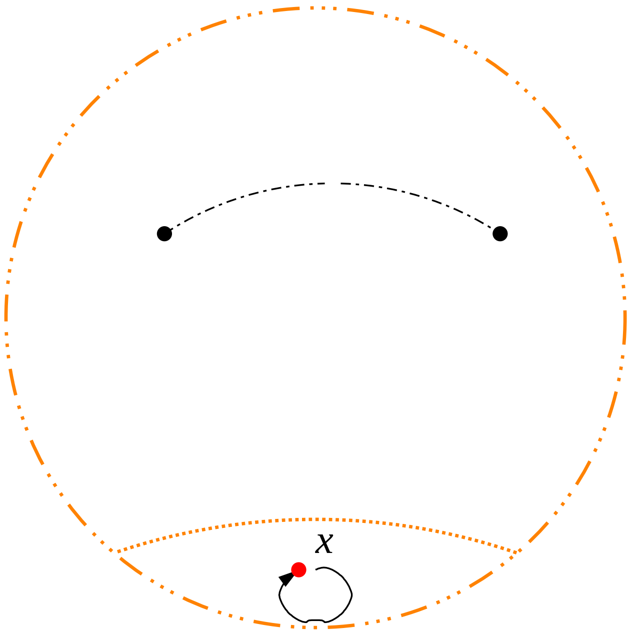
- a class is splitted because the states that it contains are no longer considered equivalent or
- a new active name is discovered.

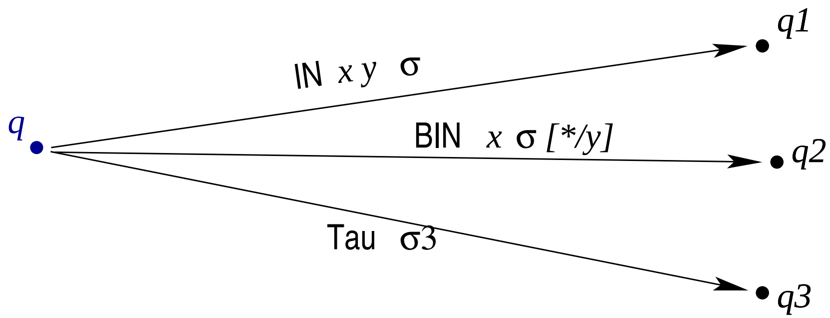
The algorithm terminates when both these two cases do not occur. This is equivalent to saying that there H_{n+1} is *equal* to H_n, for some n.

The main step



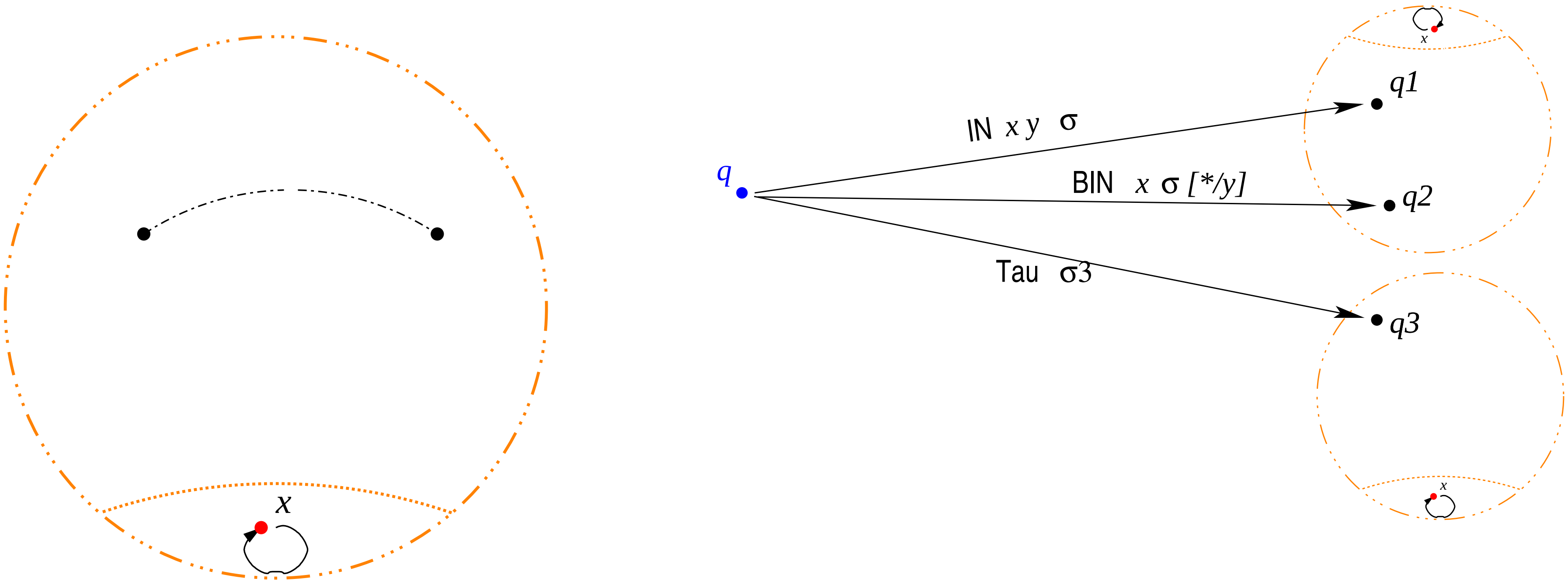
The main step





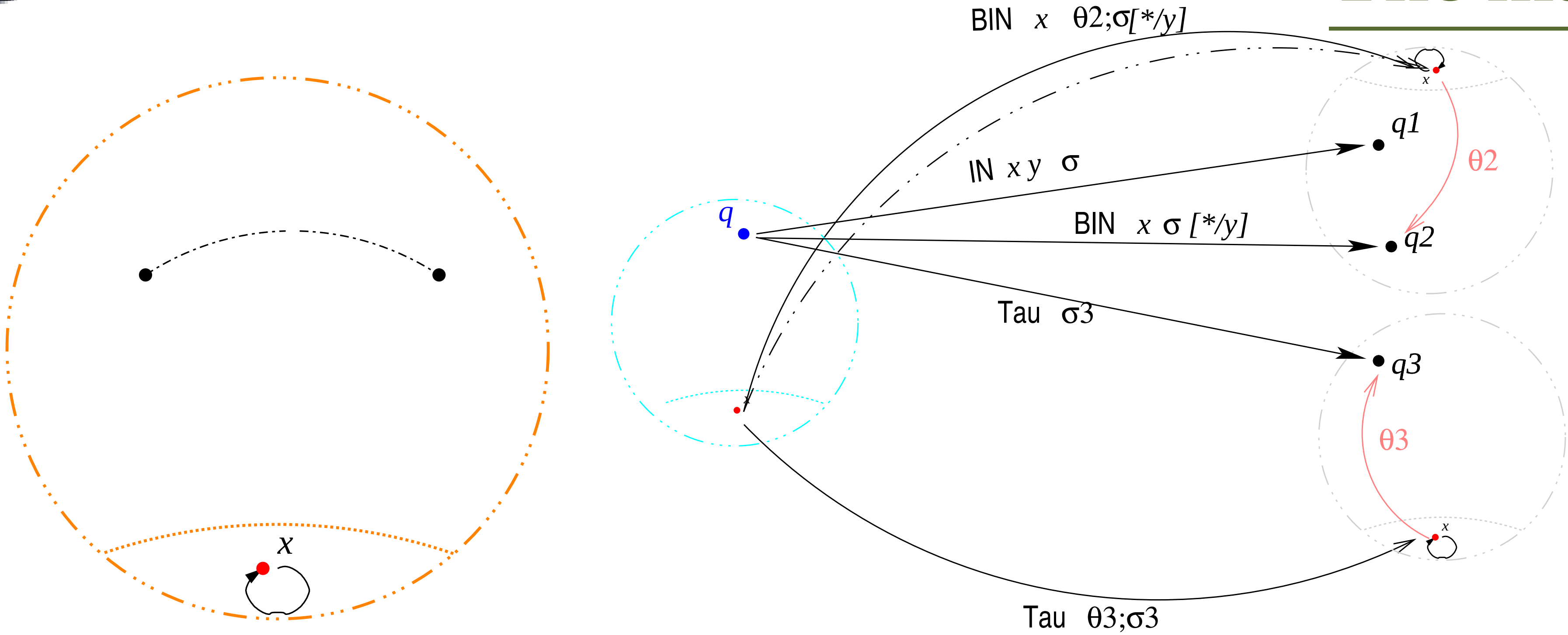

```
let bundle hd q =
  List.sort compare
    (List.filter (fun h → (Arrow.source h) = q) (arrows hd))
```

The main step



List.map h_n bundle

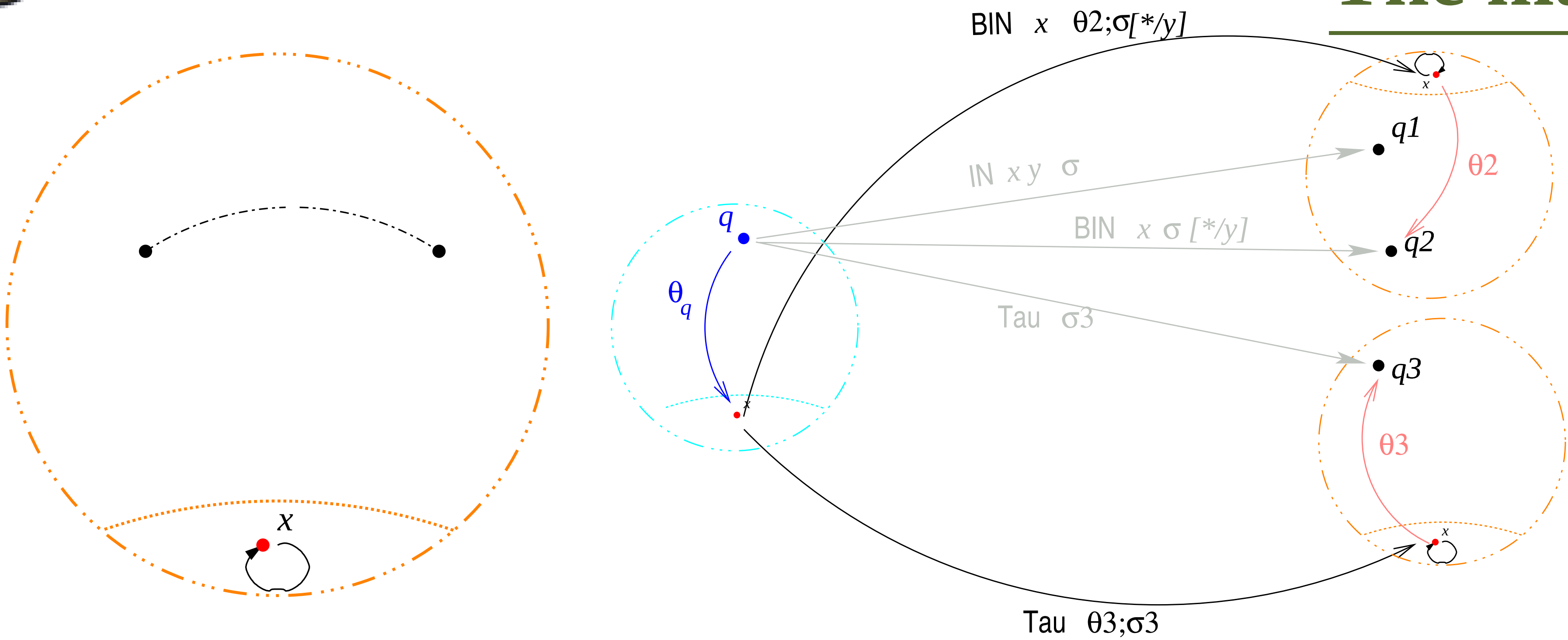
The main step



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

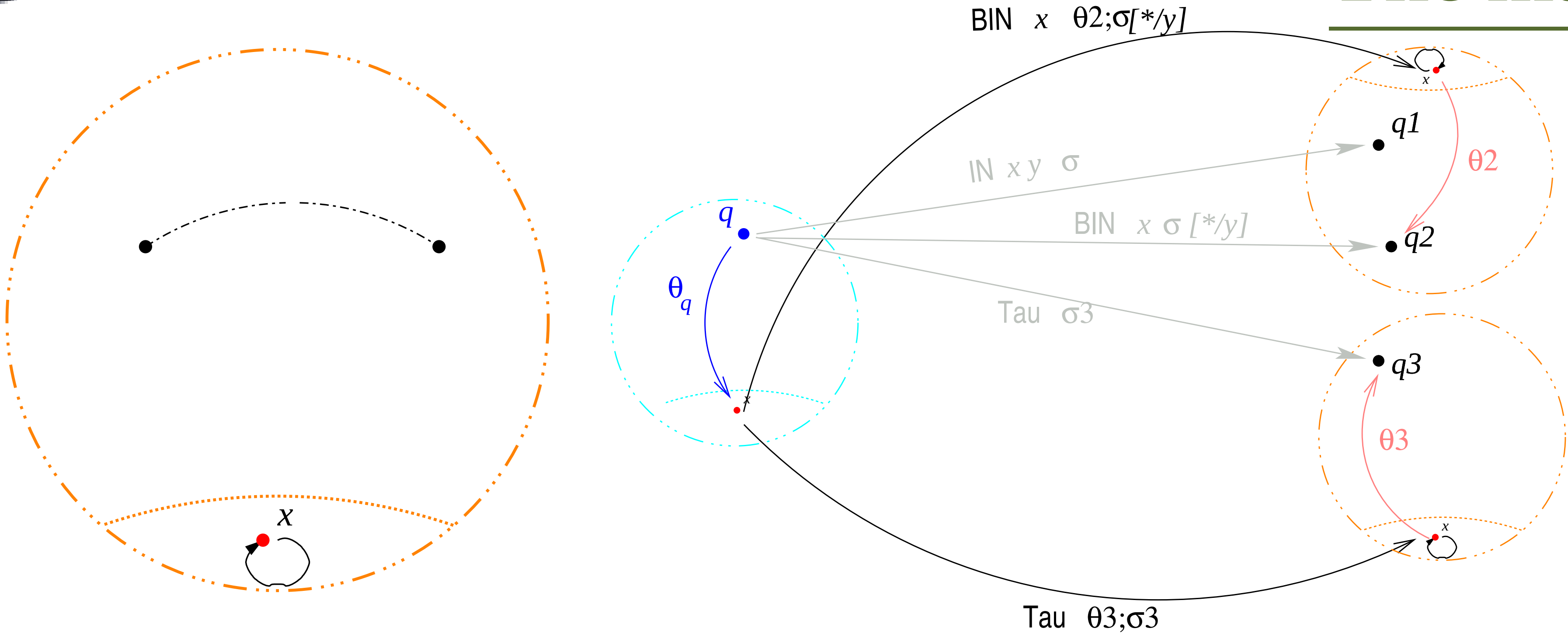
At each iteration, redundant transitions decrease and, when the iterative construction terminates, only the really redundant free inputs are removed

The main step



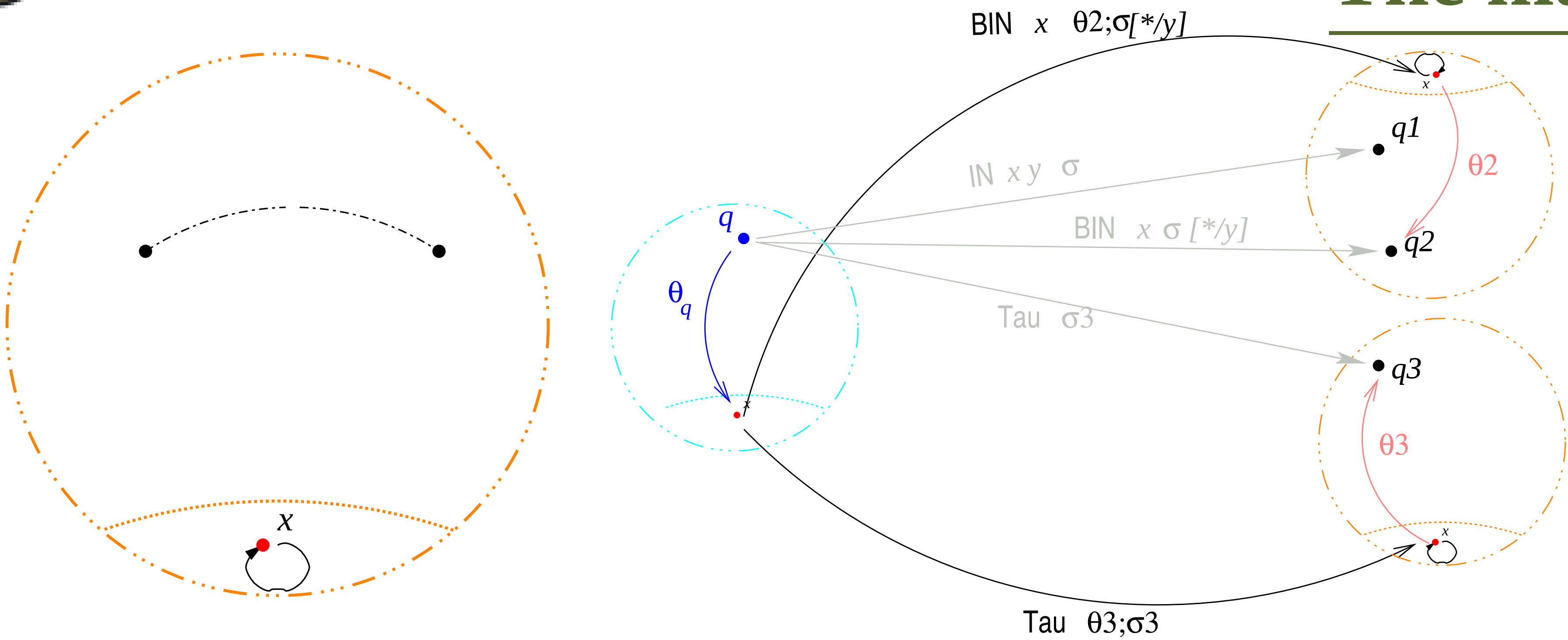

```
let an = active_names_bundle (red bundle) in
let remove_in ar = match ar with
   | Arrow(_,_,In(_,_)) → not (List.mem (obj ar) an)
   | _ → false in
    list_diff bundle (List.filter remove_in bundle)
```

The main step



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

The main step



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

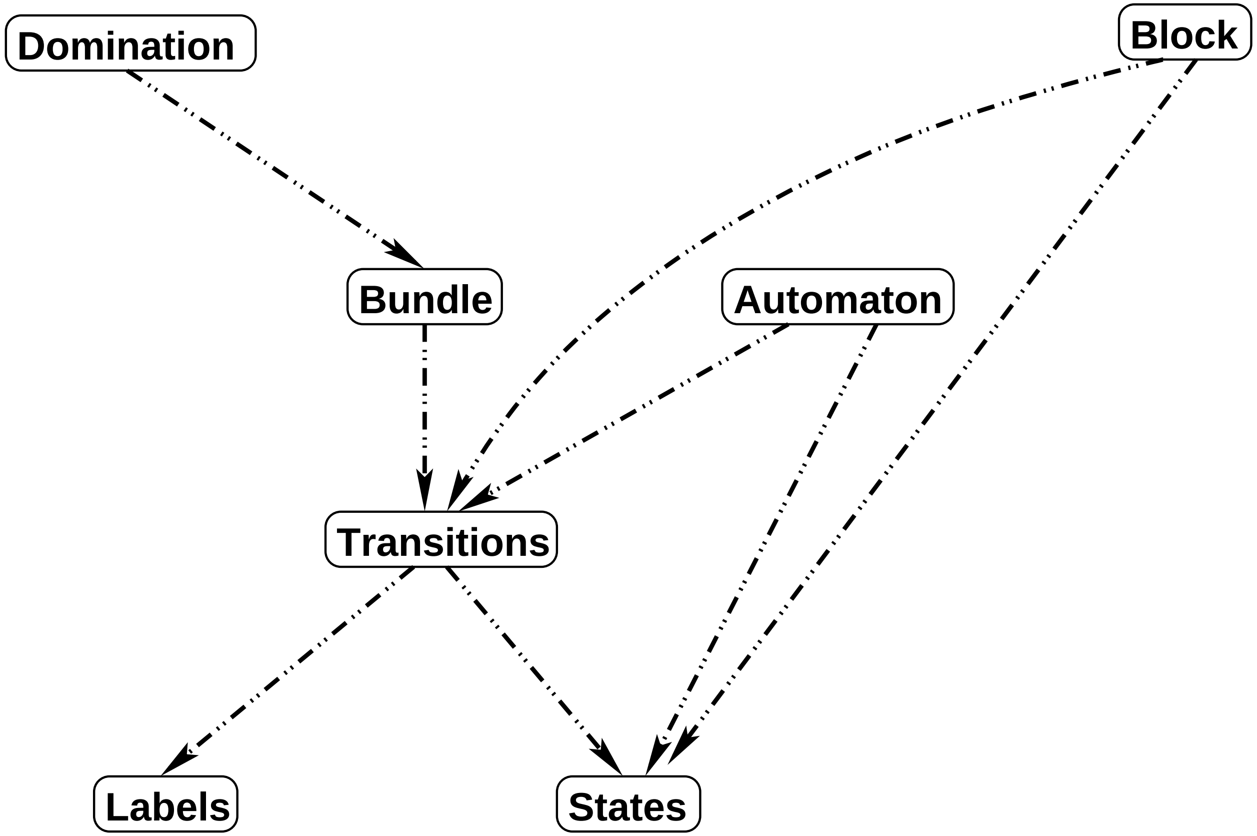
Theorem At the end of each iteration i `blocks` corresponds to h_{H_i}

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 (refining $\lambda^{\rightarrow,\Pi,\Sigma}$ spec.)

	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

References

[BBD+03] Lorenzo Bettini, Viviana Bono, Rocco De Nicola, Gianluigi Ferrari, Daniele Gorla, Michele Loreti, Eugenio Moggi, Rosario Pugliese, Emilio Tuosto, and Betti Venneri. The KLAIM Project: Theory and Practice. In Corrado Priami, editor, *Global Computing: Programming Environments, Languages, Security and Analysis of Systems*, number 2874 in Lecture Notes in Computer Science, pages 88 – 150, Rovereto (Italia), February 9-14, 2003. Springer-Verlag.

[BBV03] Lorenzo Bettini, Viviana Bono, and Betti Venneri. Subtyping Mobile Clasees and Mixins. In *Proc. of Foundation of Object Oriented Languages (FOOL10)*, 2003. Electronic proceedings.

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

[BCC01] Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Boxed Ambients. In *Tools and Algorithms for the Construction and Analysis of Systems*, number 2215 in Lecture Notes in Computer Science, pages 38–63. Springer-Verlag, 2001.

[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.

[BMM05] Roberto Bruni, Hernán Melgratti, and Ugo Montanari. Theoretical Foundations for Compensations in Flow Composition Languages. In *Annual Symposium on Principles of Programming Languages POPL*, 2005. To appear.

[BMR95] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Constraint solving over semiring. In *Proceedings of IJCAI95*, San Matco, 1995. CA: Morgan Kaufman.

[BMR97] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, March 1997.

["BP] BPEL Specifi cation (v.1.1). `http://www.ibm.com/developerworks/library/ws-bpel`.

[BPM] Business Process Modeling Language (BPML). `http://www.bpmi.org/BPML.htm`.

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

[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.

[DFM+03] Rocco De Nicola, Gianluigi Ferrari, Ugo Montanari, Rosario Pugliese, and Emilio Tuosto. A Formal Basis for Reasoning on Programmable QoS. In Nachum Dershowitz, editor, *International Symposium on Verifi cation – Theory and Practice – Honoring Zohar Manna's 64th Birthday*, volume 2772 of *Lecture Notes in Computer Science*, pages 436–479. Springer-Verlag, 2003.

[DFM$^+$05] Rocco De Nicola, Gianluigi Ferrari, Ugo Montanari, Rosario Pugliese, and Emilio Tuosto. A Basic Calculus for Modelling Service Level Agreements. In Jean-Marie Jacquet and Gian Pietro Picco, editors, *International Conference on Coordination Models and Languages*, volume 3454 of *Lecture Notes in Computer Science*, pages 33 – 48. Springer-Verlag, April 2005.

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

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

[Fed71] Jerome Feder. Plex languages. *Information Science*, 3:225–241, 1971.

[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 2303 of *Lecture Notes in Computer Science*, pages 129–143. Springer-Verlag, 2002.

[FMPar] Gianluigi Ferrari, Eugenio Moggi, and Rosario Pugliese. MetaKlaim: A type safe multi-stage language for global computing. *Mathematical Structures in Computer Science*, to appear.

[FMT01] Gianluigi Ferrari, Ugo Montanari, and Emilio Tuosto. A LTS Semantics of Ambients via Graph Synchronization with Mobility. In *Italian Conference on Theoretical Computer Science*, volume 2202 of *Lecture Notes in Computer Science*, Torino (Italy), October 4-6, 2001. Springer-Verlag.

[FMT05a] Gianluigi Ferrari, Ugo Montanari, and Emilio Tuosto. Coalgebraic Minimisation of HD-automata for the π-Calculus in a Polymorphic λ-Calculus. *Theoretical Computer Science*, 331:325–365, 2005.

[FMT05b] Gianluigi Ferrari, Ugo Montanari, and Emilio Tuosto. Model Checking for Nominal Calculi. In Vladimiro Sassone, editor, *Foundations of Software Science and Computation Structures*, volume 3441 of *Lecture Notes in Computer Science*, pages 1–24. Springer-Verlag, 2005. Invited paper to the ETAPS 2005 of Ugo Montanari.

[FMT$^+$05c] Gianluigi Ferrari, Ugo Montanari, Emilio Tuosto, Björn Victor, and Kidane Yemane. Modelling and Minimising the Fusion Calculus Using HD-Automata. In *CALCO2005*, 2005. To appear.

[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 *Lecture Notes in Computer Science*, pages 148–163. Springer-Verlag, 2000.

[HLT] Dan Hirsch, Alberto Lluch-Lafuente, and Emilio Tuosto. A Logic for Application Level QoS. Submitted to QAPL05.

[HM01] Dan Hirsch and Ugo Montanari. Synchronized hyperedge replacement with name mobility: A graphical calculus for name mobility. In *International Conference in Concurrency Theory*, volume 2154 of *Lecture Notes in Computer Science*, 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.

[HT05] Dan Hirsch and Emilio Tuosto. SHReQ: A Framework for Coordinating Application Level QoS. In *3rd IEEE International Conference on Software Engineering and Formal Methods*. ieee, 2005. To appear.

[Ley01] F. Leymann. WSFL Specification (v.1.0). `http://www-306.ibm.com/software/solutions/webservice` May 2001.

[LS00] Francesca Levi and Davide Sangiorgi. Controlling interference in ambients. In *Conference Record of POPL'00: The 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 352–364, Boston, Massachusetts, January 2000.

[LT05] Ivan Lanese and Emilio Tuosto. Synchronized Hyperedge Replacement for Heterogeneous Systems. In Jean-Marie Jacquet and Gian Pietro Picco, editors, *International Conference on Coordination Models and Languages*, volume 3454 of *Lecture Notes in Computer Science*, pages 220 – 235. Springer-Verlag, April 2005.

[MP98] Ugo Montanari and Marco Pistore. History Dependent Automata. Technical report, Computer Science Department, Università di Pisa, 1998. TR-11-98.

[MP00] Ugo Montanari and Marco Pistore. π-Calculus, Structured Coalgebras, and Minimal HD-Automata. In Mogens Nielsen and Branislav Roman, editors, *Mathematical Foundations of Computer Science*, volume 1983 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000. An extended version will be published on Theoretical Computer Science.

[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 *Lecture Notes in Computer Science*. Springer-Verlag, April 1996.

[Pav72] Theodosios Pavlidis. Linear and context-free graph grammars. *Journal of the ACM*, 19(1):11–23, 1972.

[Pis99] Marco Pistore. *History Dependent Automata*. PhD thesis, Computer Science Department, Università di Pisa, 1999.

[PV98] Joachim Parrow and Bjorn Victor. The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes. In *Annual Symposium on Logic in Computer Science*. IEEE Computer Society, 1998.

[TV04] Emilio Tuosto and Hugo T. Vieira. An Observational Model for Spatial Logics. In *First International Workshop on Views On Designing Complex Architectures*, Electronic Notes in Theoretical Computer Science, Bertinoro, Italy, September 2004. Elsevier. To appear.

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

[WSC] Web Services Conversation Language (WSCL) 1.0. `http://www.w3.org/TR/wscl10/`.

[XLA] Web Services for Business Process Design (XLANG). `http://www.gotdotnet.com/team/xml_wsspecs/x`