Synchronised Hyperedge Replacement as a Model for Service Oriented Computing

FMCO

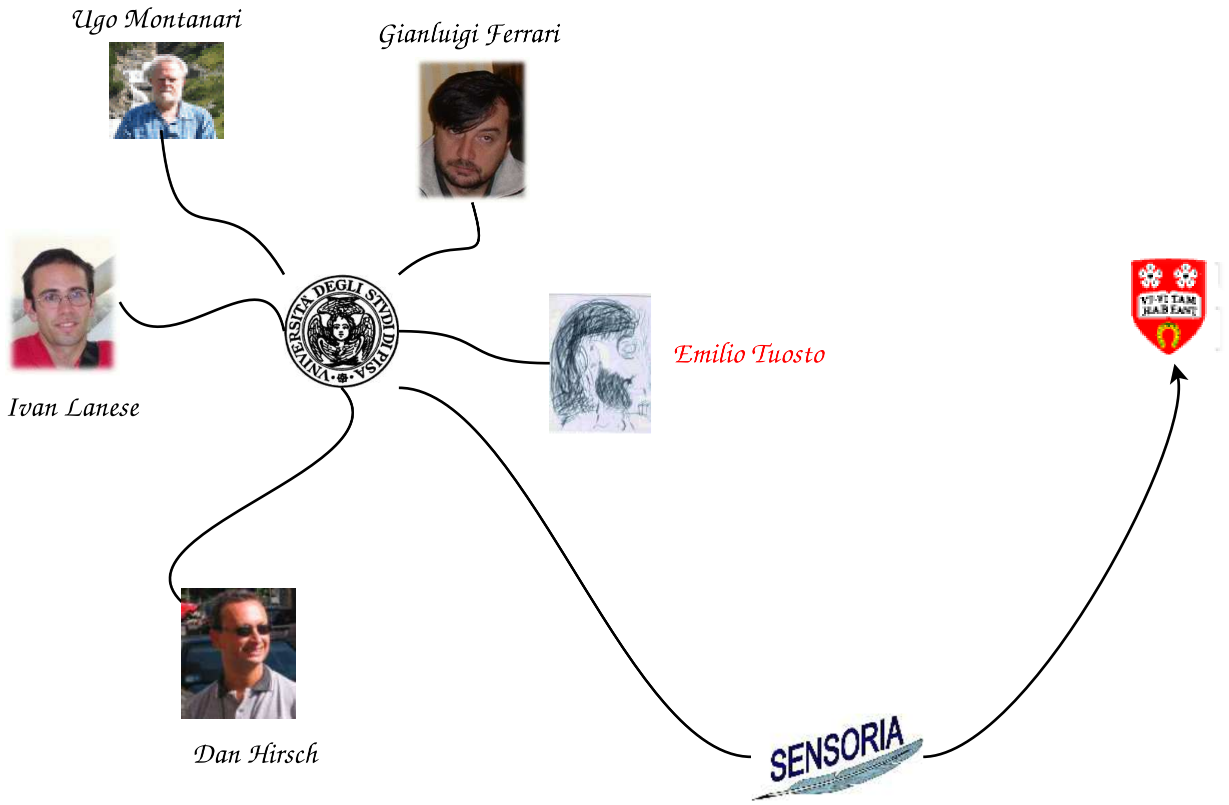
Amsterdam, 1-4 November 2005

Forewords

(with apologies and presentations)

Apologies & presentations

Let me first apologise for not having mentioned co-authors (and myself)...



SHR as a uniform framework for *non-functional* aspects of SOC

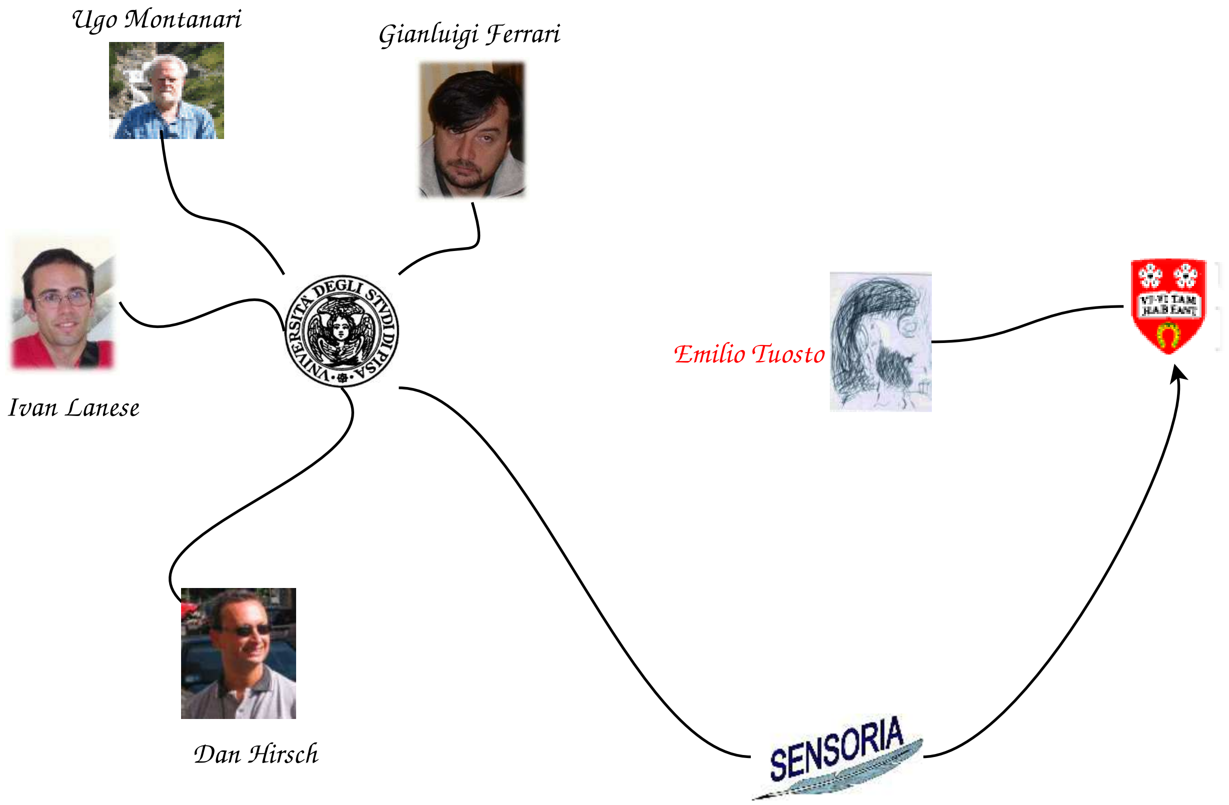
- Context-free flavour
- "SOC systems *as* Hypergraphs"
- "SOC computations *as* SHR"

In other words:

- Components = hyperedges
- Systems = *bunches* of hyperedges
- Computing = rewrite hypergraphs...
- ...with "some" synchronisation policy

Apologies & presentations

Let me first apologise for not having mentioned co-authors (and myself)...



SHR as a uniform framework for *non-functional* aspects of SOC

- Context-free flavour
- "SOC systems *as* Hypergraphs"
- "SOC computations *as* SHR"

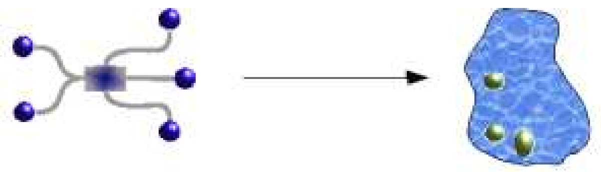
In other words:

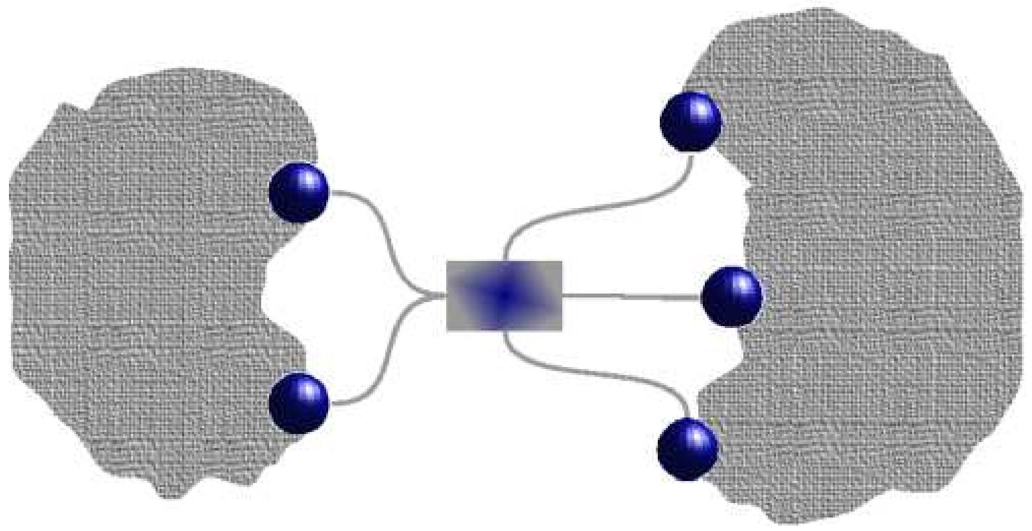
- Components = hyperedges
- Systems = *bunches* of hyperedges
- Computing = rewrite hypergraphs...
- ...with "some" synchronisation policy

In this speech

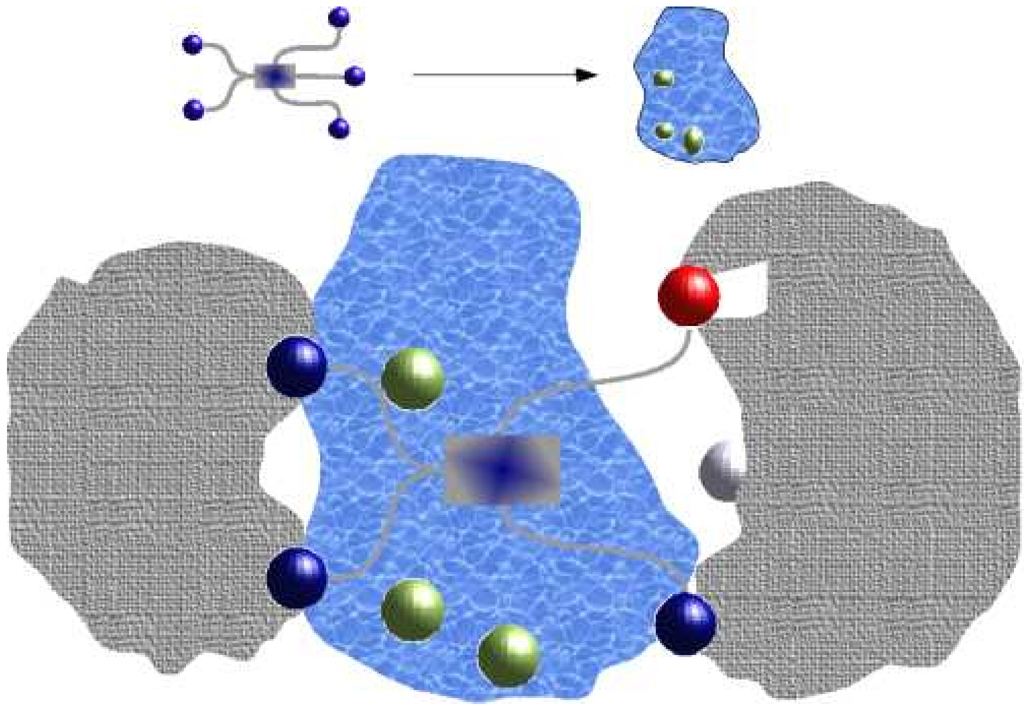
- several results on SHR are collected
- and a brand new, "modular" presesentation is given

Anatomy of a title...Synchronised Hyperedge Replacement



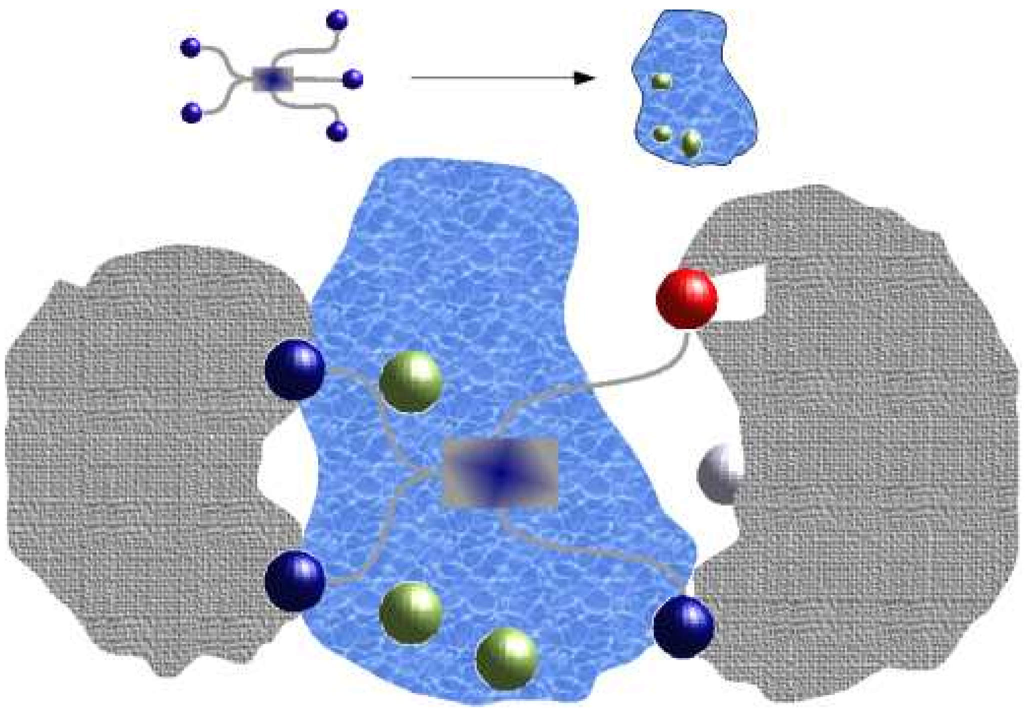


Anatomy of a title...Synchronised Hyperedge Replacement



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

Anatomy of a title...Synchronised Hyperedge Replacement



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

Benefits:

- Uniform framework
  - for π, π-I, fusion
  - LTS for Ambient ...
  - ... for Klaim ...
- Expressive for
  - distributed coordination
  - application level QoS
  - sophisticated synchronisations

Anatomy of a title...Service Oriented Computing

- Distributed computing is moving toward SOC
- Integration of software and (heterogeneous) networks of (heterogeneous) systems (e.g., Internet & mobile phones, wireless & wired networks)

Anatomy of a title...Service Oriented Computing

- Distributed computing is moving toward SOC
- Integration of software and (heterogeneous) networks of (heterogeneous) systems (e.g., Internet & mobile phones, wireless & wired networks)

- SOC architectures are
  - distributed
  - interconnected
  - based on different communication infrastructures:
    - IP, wireless, satellites...
    - overlay networks
  - Designers, programmers and end-users may ignore the stratification and complexity

Anatomy of a title...Service Oriented Computing

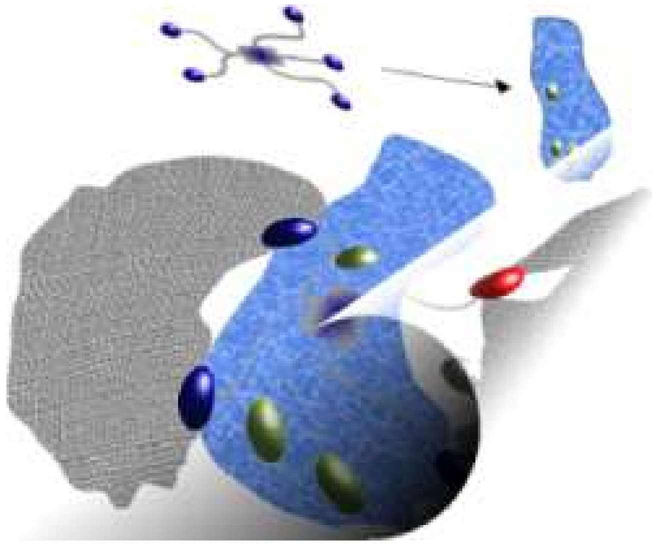
- Distributed computing is moving toward SOC
- Integration of software and (heterogeneous) networks of (heterogeneous) systems (e.g., Internet & mobile phones, wireless & wired networks)

- SOC architectures are
  - distributed
  - interconnected
  - based on different communication infrastructures:
    - IP, wireless, satellites...
    - overlay networks
  - Designers, programmers and end-users may ignore the stratification and complexity

- SOC applications (SOAs) are soups of services
  - programmable coordination
  - “autonomous”
  - independent
  - mobile/stationary
  - “interconnected” through interfaces

  and published, searched and binded ... offline and in a mostly ad-hoc way

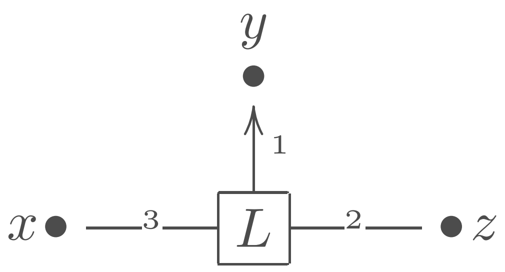
SHR family step by step



Hypergraphs Syntax

Fixed a set of nodes $\mathcal{N}$, hyperedges connect any number of nodes (generalisation of edge)

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



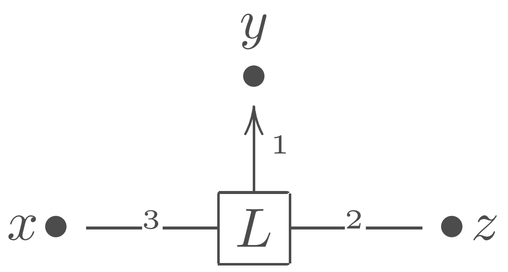
$$\begin{array}{lcl} G & ::= & nil \\ & | & L(\tilde{x}) \\ & | & G|G \\ & | & \nu\, y; A.G \end{array}$$

Syntactic Judgement $\quad x_1 : A_1, \ldots, x_n : A_n \vdash G, \qquad fn(G) \subseteq \{x_1, \ldots, x_n\}$

Hypergraphs Syntax

Fixed a set of nodes $\mathcal{N}$, hyperedges connect any number of nodes (generalisation of edge)

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



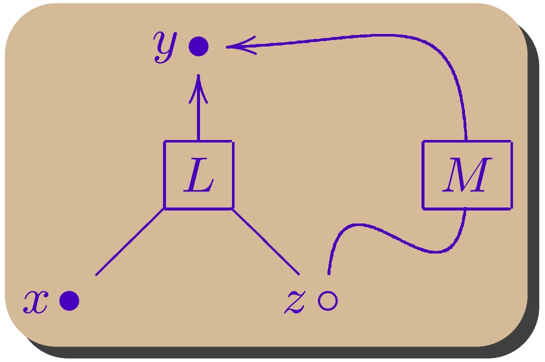
$$\begin{array}{rcl} G & ::= & nil \\ & \mid & L(\tilde{x}) \\ & \mid & G|G \\ & \mid & \nu\, y; A.G \end{array}$$

Syntactic Judgement $\quad x_1 : A_1, \ldots, x_n : A_n \vdash G, \qquad fn(G) \subseteq \{x_1, \ldots, x_n\}$

An example:

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

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



Productions

Productions are the context free rules upon which hypergraph rewriting is defined

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

$L(\tilde{x})$ rewrites as G synchronising with the edges connected to nodes $\tilde{x}$ according to Λ

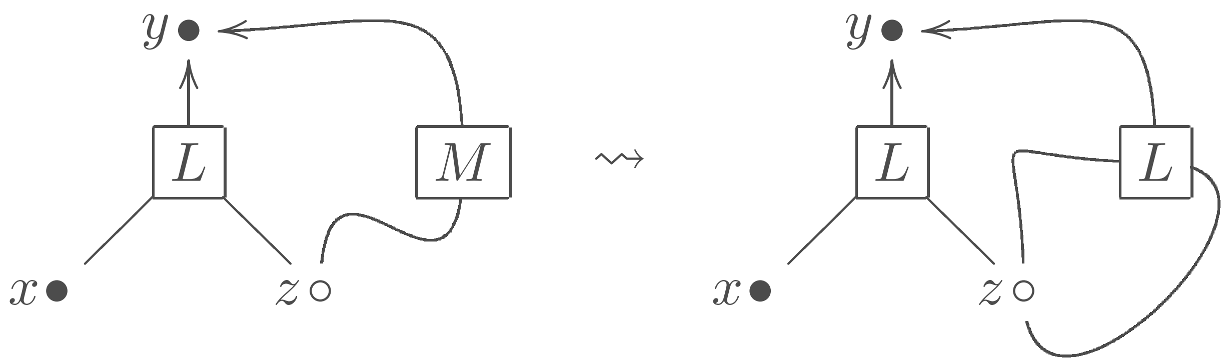
- $\tilde{x}$ is a tuple of pairwise distinguished nodes and $L : |\tilde{x}|$
- χ is a typing for nodes $\tilde{x}$
  - in the simplest case there is simply a type “node”
  - types “drive” synchronisations
- $\Lambda : \{\!|\tilde{x}|\!\} \to \mathcal{E}$ is a communication function associating events to nodes
  - $\mathcal{E} = \mathcal{A} \times \mathcal{N}^*$ is the set of events, where $\mathcal{A}$ is an alphabet of actions
  - $\mathrm{n}(\Lambda) = \{z | \exists x \in \mathrm{dom}(\Lambda).z \in \Lambda(x)\}$
- G is a graph s.t. $\mathrm{fn}(G) \subseteq \{\!|\tilde{x}|\!\} \cup \mathrm{n}(\Lambda)$,

A toy example

$$x_1, x_2, x_3 \triangleright L(x_1, x_2, x_3) \xrightarrow{\{(x_1, \mathbf{a}, \langle\rangle\}} L(x_1, x_2, x_3)$$

$$x_1, x_2 \triangleright M(x_1, x_2) \xrightarrow{\{(x_1, \bar{\mathbf{a}}, \langle\rangle\}} L(x_1, x_2, x_2)$$

$$x, y \vdash \nu\, z.(L(y, z, x) | M(y, z)) \xrightarrow{\{(y, \tau, \langle\rangle)\}} x, y \vdash \nu\, z.(L(y, z, x) | L(x, z, z))$$



The simplest SHR rewriting system

In order to introduce the basic concepts of SHR, let us first consider the case that nodes are not communicated [HIM00].
A SHR rewriting system consists of a triple

$$(\mathcal{A}\mathfrak{lg}, \mathcal{P}, \chi \vdash G)$$

where

- $\mathcal{A}\mathfrak{lg}$ is an algebra specifying the synchronisation policy (namely the types of nodes)
- $\mathcal{P}$ is a set of productions and
- $\chi \vdash G$ is the initial labelled graph

The set of transitions of $(\mathcal{A}\mathfrak{lg}, \mathcal{P}, \chi \vdash G)$ is the smallest set obtained by applying the inference rules on the next slide starting from the productions in $\mathcal{P}$

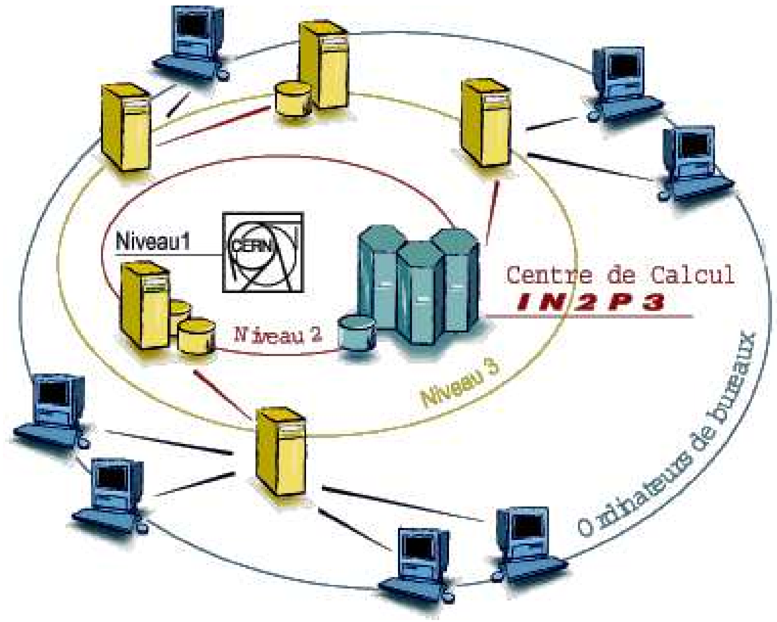
Semantics for the simplest SHR rewriting system

$$\frac{\chi, x : A \vdash G \xrightarrow{\Lambda} \chi \vdash G' \qquad \boxed{\mathrm{act}_\Lambda(x) \in \mathit{Sync}_A}}{\chi \vdash \nu\, x : A.G \xrightarrow{\Lambda \setminus \{(x,a,\langle\rangle) \mid (a,\langle\rangle) \in \mathcal{E}\}} \chi \vdash \nu\, x : A.G'} \; \texttt{(res)}$$

$$\frac{\chi_1 \vdash G_1 \xrightarrow{\Lambda_1} \chi_1' \vdash G_1' \qquad \chi_2 \vdash G_2 \xrightarrow{\Lambda_2} \chi_2' \vdash G_2' \qquad \mathrm{dom}(\chi_1) \cap \mathrm{dom}(\chi_2) = \emptyset}{\chi_1, \chi_2 \vdash G_1 | G_2 \xrightarrow{\Lambda_1, \Lambda_2} \chi_1', \chi_2' \vdash G_1' | G_2'} \; \texttt{(par)}$$

$$\frac{\chi, x : A, y : A \vdash G \xrightarrow{\Lambda, x \mapsto (a_1, \langle\rangle), y \mapsto (a_2, \langle\rangle)} \chi' \vdash G' \qquad \boxed{(a_1, a_2, c) \in \Sigma_A}}{\chi, x : A \vdash G\{^x/_y\} \xrightarrow{\Lambda, x \mapsto (c, \langle\rangle)} \chi' \setminus \{y : A\} \vdash G'\{^x/_y\}} \; \texttt{(merge)}$$

SHR family: adding mobility



Adding a bit of complexity

Let we exchange nodes in synchronisations. Why is this needed? Consider the Ambient calculus e.g.,

Adding a bit of complexity

Let we exchange nodes in synchronisations. Why is this needed? Consider the Ambient calculus e.g.,

$$\mathit{open}\ a \mid a[\ldots] \ \rightarrow\ \ldots$$

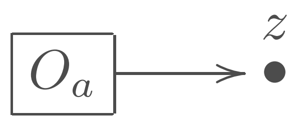
Adding a bit of complexity

Let we exchange nodes in synchronisations. Why is this needed? Consider the Ambient calculus e.g.,

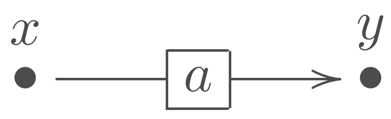
$$open\ a \mid a[...] \ \rightarrow \ \ldots$$

Components

$open\ a$:



$a[\cdots]$:



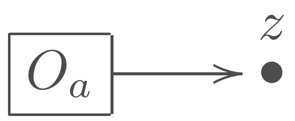
Adding a bit of complexity

Let we exchange nodes in synchronisations. Why is this needed? Consider the Ambient calculus e.g.,

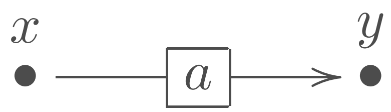
$$open\ a \mid a[...] \ \rightarrow \ ...$$

Components

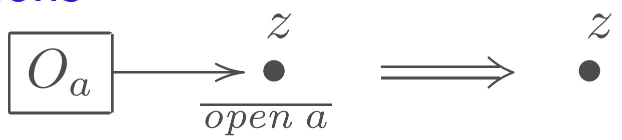
$open\ a$:

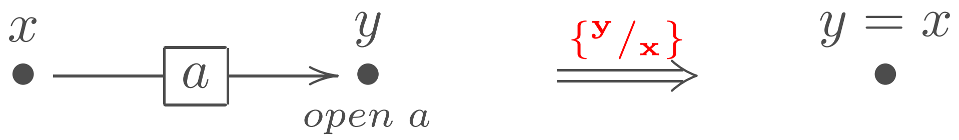


$a[\cdots]$:



Productions





Adding a bit of complexity

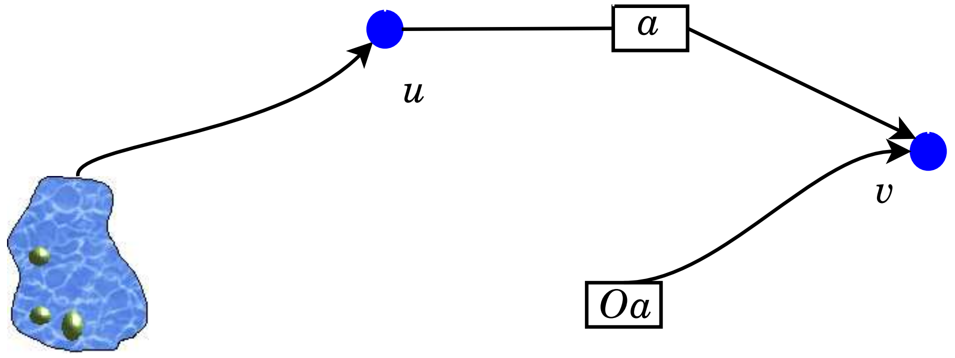
Let we exchange nodes in synchronisations. Why is this needed? Consider the Ambient calculus e.g.,

$$open\ a \mid a[...] \ \rightarrow\ ...$$

Components

$$open\ a : \quad \begin{tikzcd} \boxed{O_a} \arrow[r] & \overset{z}{\bullet} \end{tikzcd}$$

$$a[\cdots] : \quad \begin{tikzcd} \overset{x}{\bullet} \arrow[r, dash] & \boxed{a} \arrow[r] & \overset{y}{\bullet} \end{tikzcd}$$



Productions

$$\begin{tikzcd} \boxed{O_a} \arrow[r] & \underset{\overline{open\ a}}{\overset{z}{\bullet}} \arrow[r, Rightarrow] & \overset{z}{\bullet} \end{tikzcd}$$

$$\begin{tikzcd} \overset{x}{\bullet} \arrow[r, dash] & \boxed{a} \arrow[r] & \underset{open\ a}{\overset{y}{\bullet}} \arrow[r, Rightarrow, "\{{}^{y}/_{x}\}"] & \overset{y = x}{\bullet} \end{tikzcd}$$

Adding a bit of complexity

Let we exchange nodes in synchronisations. Why is this needed? Consider the Ambient calculus e.g.,

$$open\ a \mid a[...] \;\rightarrow\; ...$$

Components

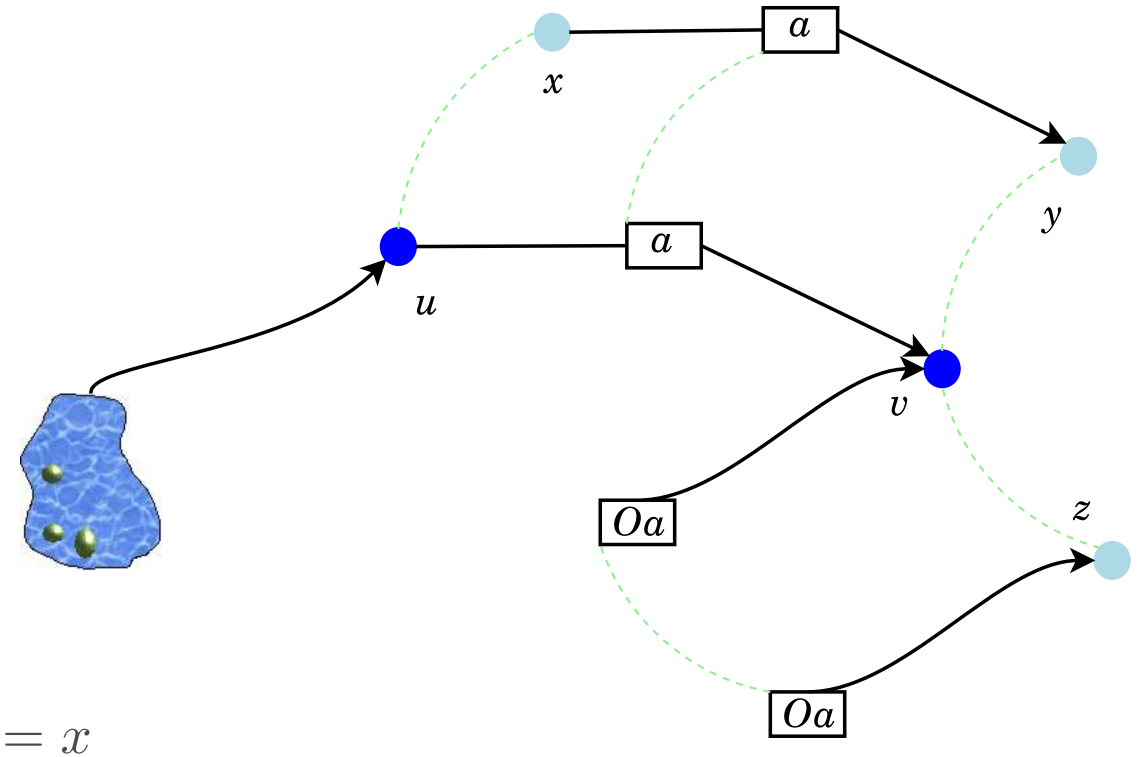
$open\ a$:
$$\begin{tikzcd} \boxed{O_a} \arrow[r] & \overset{z}{\bullet} \end{tikzcd}$$

$a[\cdots]$:
$$\begin{tikzcd} \overset{x}{\bullet} \arrow[r, dash] & \boxed{a} \arrow[r] & \overset{y}{\bullet} \end{tikzcd}$$

Productions

$$\begin{tikzcd} \boxed{O_a} \arrow[r] & \underset{\overline{open\ a}}{\overset{z}{\bullet}} \arrow[r, Rightarrow] & \overset{z}{\bullet} \end{tikzcd}$$

$$\begin{tikzcd} \overset{x}{\bullet} \arrow[r, dash] & \boxed{a} \arrow[r] & \underset{open\ a}{\overset{y}{\bullet}} \arrow[r, Rightarrow, "\{^{y}/_{x}\}"] & \overset{y=x}{\bullet} \end{tikzcd}$$



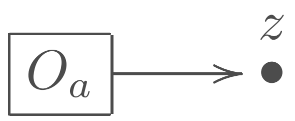
Adding a bit of complexity

Let we exchange nodes in synchronisations. Why is this needed? Consider the Ambient calculus e.g.,

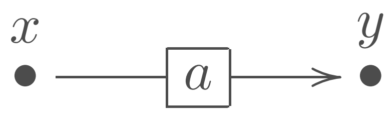
$$open\ a \mid a[...] \ \rightarrow \ ...$$

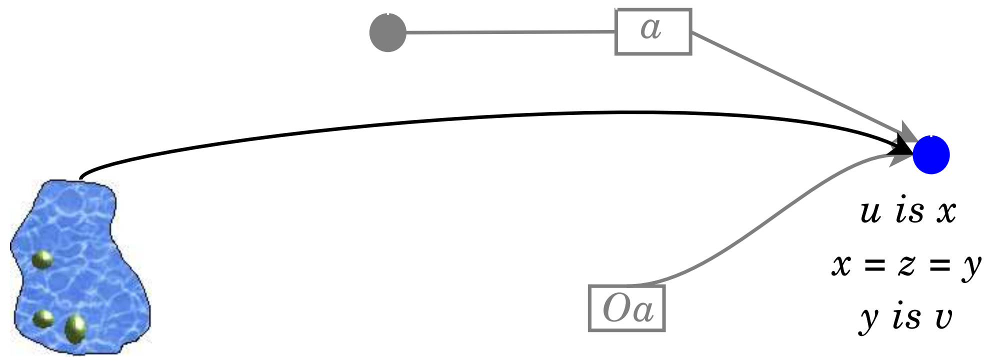
Components

$open\ a$:

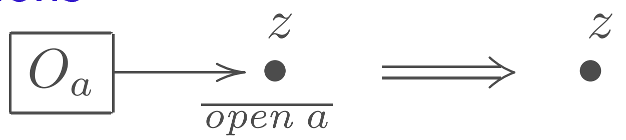


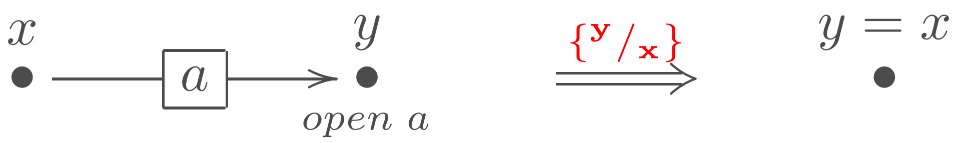
$a[\cdots]$:





Productions





For details about Ambient and SHR see [FMT01]

Extending SHR with node passing

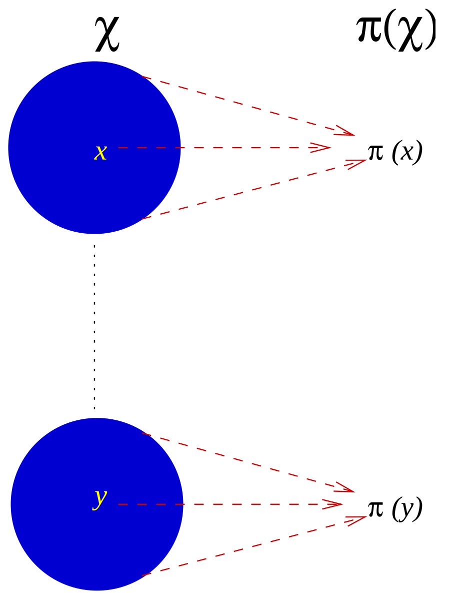
Technicalities to face with:

- nodes vs restriction
- node "fusions" (e.g., see Ambient)
- effects of node fusions on node types

Reconsidering graph transitions...

$$\chi \vdash G \xrightarrow{\Lambda,\pi} \chi' \vdash G'$$

- $\pi : \mathrm{dom}(\chi) \to \mathrm{dom}(\chi)$ is an idempotent substitution
- if $\Lambda(x) = (a, \tilde{y})$ then $|\tilde{y}| = \mathrm{ar}(a)$
- $\mathrm{dom}(\chi') = \pi(\mathrm{dom}(\chi)) \cup (\mathrm{n}(\Lambda) \setminus \mathrm{dom}(\chi))$



Synchonisation algebras

Synchronisation algebras with mobility [LM04] extend Winskel's synchronisation algebras to encompass mobility in the style of name-passing calculi.

Synchonisation algebras

Synchronisation algebras with mobility [LM04] extend Winskel's synchronisation algebras to encompass mobility in the style of name-passing calculi.
Nodes are typed with SAMs over the actions set $\mathcal{A} \neq \emptyset$

$$A = \langle \mathtt{N}, \mathcal{A}, \mathrm{ar}, \epsilon, \mathrm{Sync}, \Sigma \rangle$$

Synchonisation algebras

Synchronisation algebras with mobility [LM04] extend Winskel's synchronisation algebras to encompass mobility in the style of name-passing calculi.
Nodes are typed with SAMs over the actions set $\mathcal{A} \neq \emptyset$

$$A = \langle \mathtt{N}, \mathcal{A}, \mathrm{ar}, \epsilon, \mathrm{Sync}, \Sigma \rangle$$

where

- ϵ is a distinguished action s.t. $\mathrm{ar}(\epsilon) = 0$
- $\mathrm{Sync} \subseteq \mathcal{A}$ are "final" actions ($\epsilon \in \mathrm{Sync}$)
- Σ for action composition triples like $(a, b, (c, \mathrm{Mb}))$ s.t.
  - $a, b, c \in \mathcal{A}$
  - $c = \epsilon \Leftrightarrow a = b = \epsilon$
  - $\mathrm{Mb} : Int_{\mathrm{ar}(a)} \uplus Int_{\mathrm{ar}(b)} \to \{1, 2, \ldots\}$ states how nodes of a and b are fused and how they correspond to nodes of c

Synchonisation algebras

Synchronisation algebras with mobility [LM04] extend Winskel's synchronisation algebras to encompass mobility in the style of name-passing calculi.
Nodes are typed with SAMs over the actions set $\mathcal{A} \neq \emptyset$

$$A = \langle \mathtt{N}, \mathcal{A}, \mathrm{ar}, \epsilon, \mathrm{Sync}, \Sigma \rangle$$

The "Milner" SAM on actions L is $\langle \mathrm{Mil}, \mathcal{A}, \mathrm{ar}, \epsilon, \mathrm{Sync}, \Sigma \rangle$, where

$$\mathcal{A} = \{a, \overline{a} \mid a \in L\} \cup \{\tau, \epsilon\}$$

where

- $\mathrm{ar}(\overline{a_i}) = \mathrm{ar}(a_i)$ and $\mathrm{ar}(\tau) = 0$
- $\mathrm{Sync} = \{\tau, \epsilon\}$.
- $MP_{n,m} : Int_n \uplus Int_m \to Int_{\max(n,m)}$ is undefined on $i > \min(x, y)$ and $\forall i \leq \min(n, m).MP([1, i]) = MP([2, i]) = i$
- $\forall a \in L.\ (a, \epsilon, (a, MP_{\mathrm{ar}(a),0})) \in \Sigma \wedge (a, \overline{a}, (\tau, MP_{\mathrm{ar}(a),\mathrm{ar}(\overline{a})})) \in \Sigma$

SHR transitions and SAMs [LT05]

Let $(\mathcal{A}\mathfrak{lg}, \diamond)$ be the commutative monoid of SAMs of interest and consider transitions (once more)...

$$\chi \vdash G \xrightarrow{\Lambda,\pi} \chi' \vdash G'$$

- for all $x \in \mathrm{n}(\Lambda)\rho \setminus \mathrm{dom}(\chi')$ (i.e., $x \in \mathrm{bn}(G)$), $\chi'(x)$ is determined according to the freely assigned SAM in the productions
- for each $x \in \mathrm{dom}(\chi')$

  $$\chi'(x) \overset{\text{def}}{=} A_1 \diamond \ldots \diamond A_n$$

  where $\{A_1, \ldots, A_n\}$ are the SAMs assigned by χ to the nodes in $\pi^{-1}(x)$
- χ' is well-defined since SAMs form a commutative monoid

When assigning all nodes the types "Milner" (resp. "Hoare"), we obtain the (very) special case of CCS-like (resp. CSP-like) synchronisations! [HM01, Hir03]

Technicalities...

Names mainly affect rules (`res`) and (`merge`):

$$\frac{\chi, x : A \vdash G \xrightarrow{\Lambda} \chi' \vdash G' \qquad \mathrm{act}_\Lambda(x) \in Sync_A \qquad \boxed{x\pi = y\pi \wedge x \neq y \implies x\pi \neq x}}{\chi \vdash \nu\, x : A.G \xrightarrow{\Lambda \setminus \{(x,a,\langle\tilde{y}\rangle) \mid (a,\langle\tilde{y}\rangle)\in\mathcal{E}\},\ \pi|_{\mathrm{dom}(\chi)}} \chi'' \vdash (\nu\ \chi' \setminus \chi'').G'}\ (\mathtt{res})$$

x cannot be the representative element when its class is not trivial (otherwise you might have undesired scope extrusions!)

Technicalities...

Names mainly affect rules `(res)` and `(merge)`:

$$\frac{\chi, x : A \vdash G \xrightarrow{\Lambda} \chi' \vdash G' \qquad \mathrm{act}_\Lambda(x) \in Sync_A \qquad \boxed{x\pi = y\pi \wedge x \neq y \implies x\pi \neq x}}{\chi \vdash \nu\, x : A.G \xrightarrow{\Lambda \setminus \{(x,a,\langle\tilde{y}\rangle) \mid (a,\langle\tilde{y}\rangle)\in\mathcal{E}\},\ \pi|_{\mathrm{dom}(\chi)}} \chi'' \vdash (\nu\ \chi' \setminus \chi'').G'} \ \texttt{(res)}$$

This is a sort of “close” mechanism (similar to π-calculus). Nodes extruded on x must be bound after the transition ($x \in \mathrm{dom}(\chi' \setminus \chi'')$, unless elsewhere extruded).

Technicalities...

Names mainly affect rules (`res`) and (`merge`):

$$\frac{\chi, x : A \vdash G \xrightarrow{\Lambda} \chi' \vdash G' \qquad \mathrm{act}_\Lambda(x) \in Sync_A \qquad \boxed{x\pi = y\pi \wedge x \neq y \implies x\pi \neq x}}{\chi \vdash \nu\, x : A.G \xrightarrow{\Lambda \setminus \{(x,a,\langle\tilde{y}\rangle) \mid (a,\langle\tilde{y}\rangle) \in \mathcal{E}\},\ \pi|_{\mathrm{dom}(\chi)}} \chi'' \vdash (\nu\ \chi' \setminus \chi'').G'} \ (\texttt{res})$$

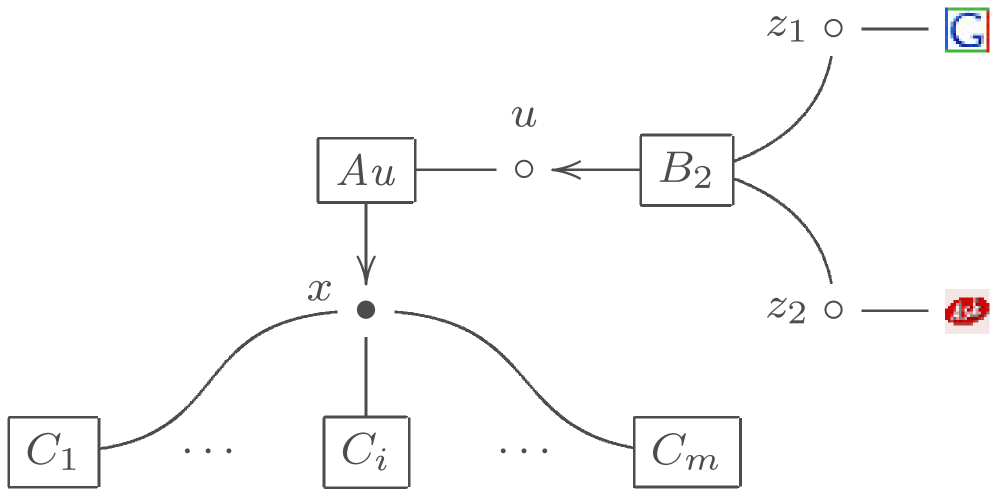
$$\frac{\chi, x : A, y : A \vdash G \xrightarrow{\Lambda, x \mapsto (a_1,\langle\tilde{v}_1\rangle), y \mapsto (a_2,\langle\tilde{v}_2\rangle),\ \pi} \chi' \vdash G' \qquad (a_1, a_2, (c, \mathrm{Mb})) \in \Sigma_A}{\chi, x : A \vdash G\{^x/_y\} \xrightarrow{\Lambda\rho, x \mapsto (c,\langle\tilde{w}\rangle),\ \rho|_{\mathrm{dom}(\chi)\cup\{x\}}} \chi'' \vdash (\nu\, \chi'\rho \setminus \chi'').G'\rho} \ (\texttt{merge})$$

where:

- $\rho = \mathbf{mgu}\ (\{\tilde{v}_1[j_1] = \tilde{v}_2[j_2] | \mathrm{Mb}([1,j_1]) = \mathrm{Mb}([2,j_2])\} \cup \{u = v \mid u\pi = v\pi\} \cup y \mapsto x)$
- $\tilde{w}[i] = (\tilde{v}_j[k])\{^x/_y\}\rho$ if $\mathrm{Mb}([j,k]) = i$, $i \in Int_{\mathrm{ar}(c)}$;
- the type of z in $\mathrm{fn}(G'\rho)$ is $A_1 \diamond \ldots \diamond A_n$ where $A_1, \ldots, A_n$ are the types in χ of $\{x_1, \ldots, x_n\}$, the equivalence class of z

An example

- Clients $C_1, \ldots, C_m$ invoke a service from remote servers and provided that they are authorised
- A trusted authority Au checks for the authorisation



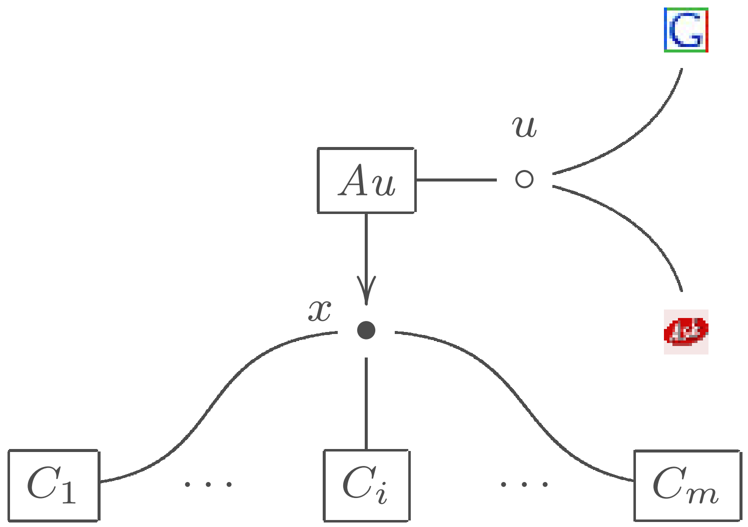
Clients are connected to Au on a "public" node x while servers are connected on a "private" (i.e., restricted) one.
B_2 will simply acquire the requests from clients and forward them to each server.

Notice that

- synchronisations C_i — Au are "Milner" (e.g., PPP)
- B_2 is required when broadcast is not primitive
- then broadcast must be "encoded"

An example

- Clients $C_1, \ldots, C_m$ invoke a service from remote servers and provided that they are authorised
- A trusted authority Au checks for the authorisation



Clients are connected to Au on a “public” node x while servers are connected on a “private” (i.e., restricted) one.

In SHR we can simply specify

$$x : Mil \vdash C_i(x) \xrightarrow{(x,\overline{\mathrm{auth}}_i,\langle y \rangle)} x : Mil, y : Bdc \vdash C_i'(y)$$

$$x : Mil, u : Bdc \vdash Au(x,u) \xrightarrow{(x,\mathrm{auth}_i,\langle u \rangle)} x : Mil, u : Bdc \vdash Au(x,u)$$

SHR family: dealing with application level QoS

Application-level QoS

Lifting QoS issues to applications

- with programmable application level QoS
- QoS in designing and implementing SOAs

Application-level QoS

Lifting QoS issues to applications

- with programmable application level QoS
- QoS in designing and implementing SOAs

Search and bind wrt application level QoS

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

Application-level QoS

Lifting QoS issues to applications

- with programmable application level QoS
- QoS in designing and implementing SOAs

Search and bind wrt application level QoS

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

Constraint-semiring [BMR95, BMR97] are particularly suitable because

- they have an implicit partial order
- preserved by many constructions...
- ...e.g., cartesian product
- hence multi-criteria

SHR uses c-semiring as a synchronisation mechanism! [HT05]

An algebraic structure $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ is a c-semiring iff $\mathbf{0} \neq \mathbf{1} \in S$, and

$$\begin{aligned}
x + y &= y + x \\
(x + y) + z &= x + (y + z) \\
(x + y) \star z &= (x \star z) + (y \star z) \\
x + \mathbf{1} &= \mathbf{1}
\end{aligned}$$

$$\begin{aligned}
x \star y &= y \star x \\
(x \star y) \star z &= x \star (y \star z) \\
x \star \mathbf{1} &= x \\
x \star \mathbf{0} &= \mathbf{0}
\end{aligned}$$

An algebraic structure $\langle S, +, \star, \mathbf{0}, \mathbf{1}\rangle$ is a c-semiring iff $\mathbf{0} \neq \mathbf{1} \in S$, and

$$\begin{aligned} x + y &= y + x \\ (x+y)+z &= x+(y+z) \\ (x+y)\star z &= (x\star z)+(y\star z) \\ x + \mathbf{1} &= \mathbf{1} \end{aligned}$$

$$\begin{aligned} x \star y &= y \star x \\ (x\star y)\star z &= x\star(y\star z) \\ x \star \mathbf{1} &= x \\ x \star \mathbf{0} &= \mathbf{0} \end{aligned}$$

- Implicit partial order:
  $a \leq b \iff a + b = b$
  "b is better than a"
- The cartesian product of c-semirings is a c-semiring

SHReQ Productions

An algebraic structure $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ is a c-semiring iff $\mathbf{0} \neq \mathbf{1} \in S$, and

$$
\begin{aligned}
x + y &= y + x \\
(x + y) + z &= x + (y + z) \\
(x + y) \star z &= (x \star z) + (y \star z) \\
x + \mathbf{1} &= \mathbf{1}
\end{aligned}
\qquad
\begin{aligned}
x \star y &= y \star x \\
(x \star y) \star z &= x \star (y \star z) \\
x \star \mathbf{1} &= x \\
x \star \mathbf{0} &= \mathbf{0}
\end{aligned}
$$

- Implicit partial order: $a \leq b \iff a + b = b$ "b is better than a"
- The cartesian product of c-semirings is a c-semiring

Transitions rewrite "weighted" graphs

$$\chi \vdash G \xrightarrow{\Lambda} \chi' \vdash G'$$

as before, but now node types are c-semiring values and requirements $\mathcal{R} = S \times \mathcal{N}^*$ represent 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 Synchronised Rewriting

When fusing two nodes of a production $\chi \triangleright L(\tilde{x}) \xrightarrow{\Lambda} G$...

$$\chi' \triangleright L(\tilde{x}\{^y/_x\}) \xrightarrow{\Lambda\{^y/_x\}} G\{^y/_x\}$$

updated requirements are $\chi'(z) = \begin{cases} \chi(z), & z \in \{\!|\tilde{x}|\!\} \setminus \{x, y\} \\ \chi(x) + \chi(y), & z = y \wedge y \in \tilde{x} \\ \chi(x), & z = y \wedge y \notin \{\!|\tilde{x}|\!\} \end{cases}$

SHReQ Synchronised Rewriting

When fusing two nodes of a production $\chi \triangleright L(\tilde{x}) \xrightarrow{\Lambda} G$

$\chi' \triangleright L$

yields an idempotent substitution defined iff

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

updated requirements are $\chi'(z) = \begin{cases} \chi(z), & z \in \{|\tilde{x}|\} \setminus \{x, y\} \\ \chi(x) + \chi(y), & z = y \wedge y \in \tilde{x} \\ \chi(z), & z = y \wedge y \notin \{|\tilde{x}|\} \end{cases}$

$$\frac{\chi \triangleright L(\tilde{x}) \xrightarrow{\Lambda} G \in \mathcal{P} \qquad \rho = \textbf{mgu}\ \Lambda \qquad \bigwedge_{x \in \mathrm{dom}(\chi)} \chi(x) \leq \chi'(x)}{\chi' \vdash L(\tilde{x}) \xrightarrow{\Lambda} \chi_{'}\Lambda \vdash \nu\ Z.(\ G\rho\ )}$$

SHReQ Synchronised Rewriting

When fusing two nodes of a production $\chi \rhd L(\tilde{x}) \xrightarrow{\Lambda} G$...

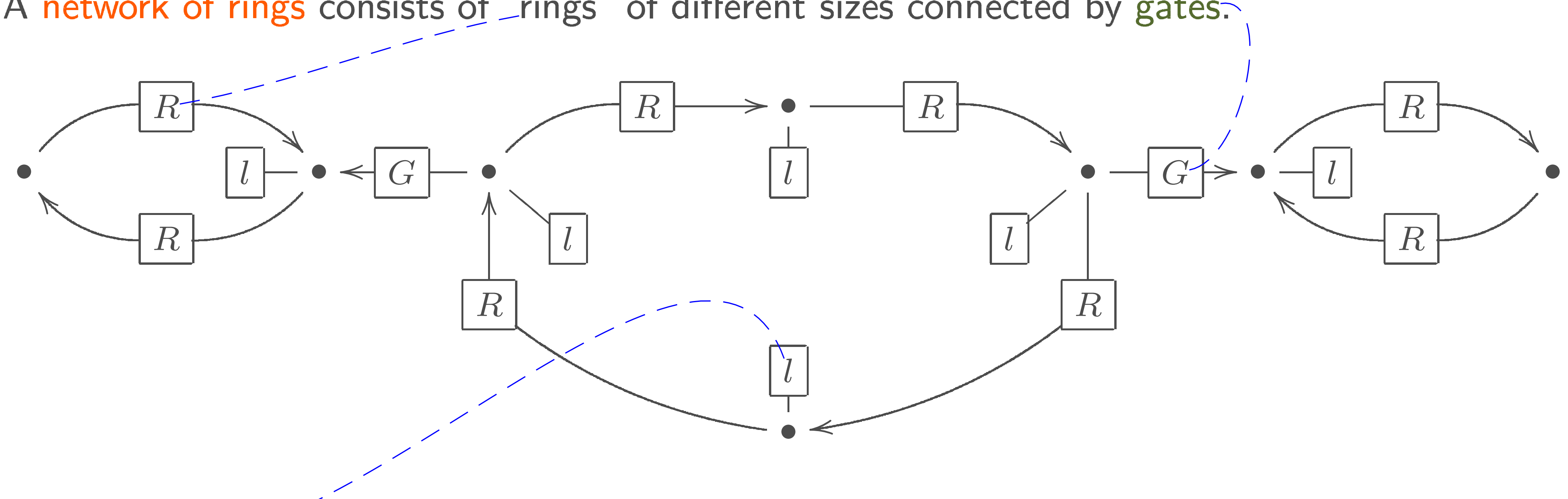
$$\chi' \rhd L(\tilde{x}\{{}^{y}/_{x}\}) \xrightarrow{\Lambda\{{}^{y}/_{x}\}} G\{{}^{y}/_{x}\}$$

updated requirements are $\chi'(z) = \begin{cases} \chi(z), & z \in \{\!|\tilde{x}|\!\} \setminus \{x, y\} \\ \chi(x) + \chi(y), & z = y \wedge y \in \tilde{x} \\ \chi(x), & z = y \wedge y \notin \{\!|\tilde{x}|\!\} \end{cases}$

$$\frac{\chi \rhd L(\tilde{x}) \xrightarrow{\Lambda} G \in \mathcal{P} \qquad \rho = \mathbf{mgu}\ \Lambda \qquad \bigwedge_{x \in \mathrm{dom}(\chi)} \chi(x) \leq \chi'(x)}{\chi' \vdash L(\tilde{x}) \xrightarrow{\Lambda} \chi_{'}\Lambda \vdash \nu\ Z.(\ G\rho\ )}$$

$$\frac{\chi_1 \vdash G_1 \xrightarrow{\Lambda_1} \chi_1' \vdash G_1' \qquad \chi_2 \vdash G_2 \xrightarrow{\Lambda_2} \chi_2' \vdash G_2' \qquad \rho = \mathbf{mgu}\ \Lambda_1 \uplus \Lambda_2 \qquad \bigwedge_{x \in \mathrm{dom}(\chi_1) \cap \mathrm{dom}(\chi_2)} \chi_1(x) = \chi_2(x)}{\chi_1 \cup \chi_2 \vdash G_1 \mid G_2 \xrightarrow{\Lambda_1 \uplus \Lambda_2} (\chi_1 \cup \chi_2)_{(\Lambda_1 \uplus \Lambda_2)} \vdash \nu\, Z.(G_1' \mid G_2')\rho}$$

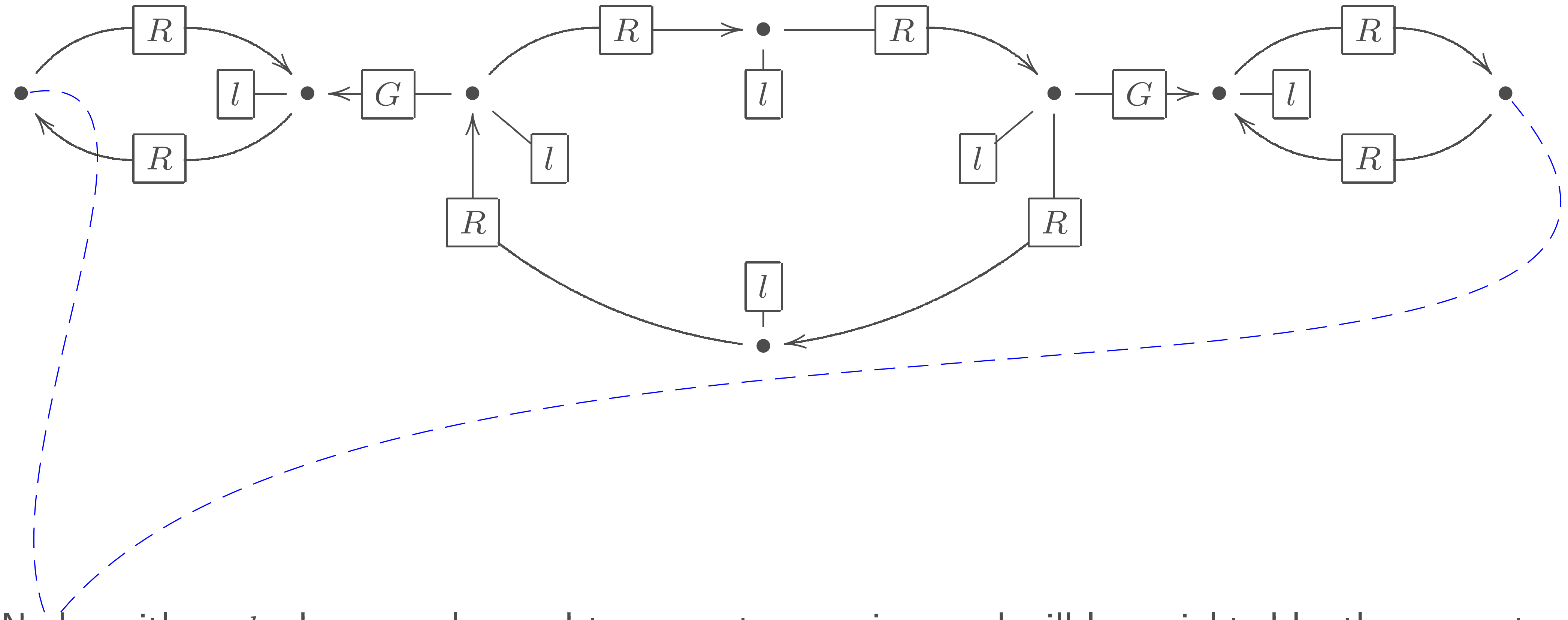
A network of rings consists of "rings" of different sizes connected by gates.



The l-edges avoid new gates to be attached on the node they insist on

QoS & Synchronisations

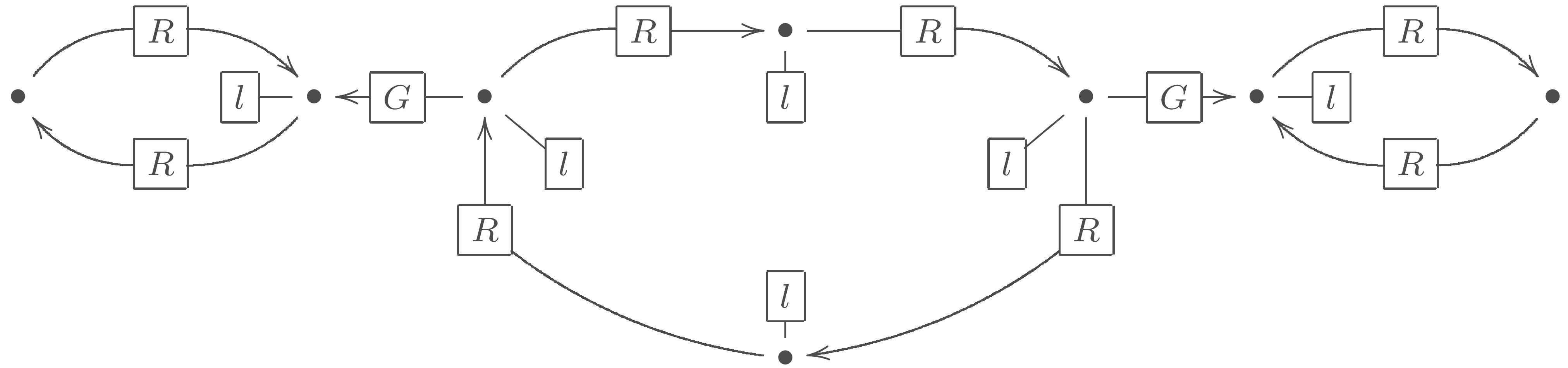
A network of rings consists of "rings" of different sizes connected by gates.



Nodes with no l-edges, can be used to generate new rings and will be weighted by the amount of available resource.

QoS & Synchronisations

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

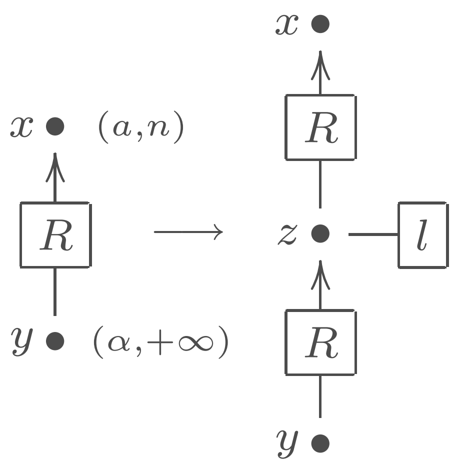


The c-semiring is $\mathfrak{H}\mathfrak{R}$, the cartesian product of the Hoare c-semiring $\mathfrak{H} = \langle \mathcal{H}, +_H, \star_H, \mathbf{0}_H, \mathbf{1}_H \rangle$ (where $\mathcal{H} = \{a, b, c, \mathbf{1}_H, \mathbf{0}_H, \perp\}$) and $\mathfrak{R} = \langle \omega_\infty, max, min, 0, +\infty \rangle$
The idea is that

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

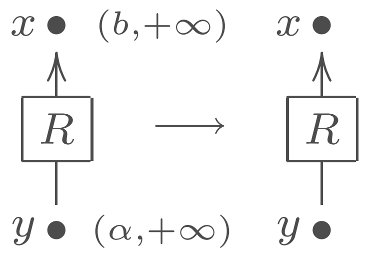
Productions for the ring case study

Create Brother $(n < u)$



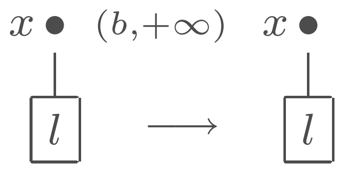
$$x : (\mathbf{0}_H, u), y : \mathbf{0} \triangleright R(x,y) \xrightarrow[]{\substack{x \mapsto (a,\, n) \\ y \mapsto (\alpha,\, +\infty)}} R(x,z) \mid R(z,y) \mid l(z)$$

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

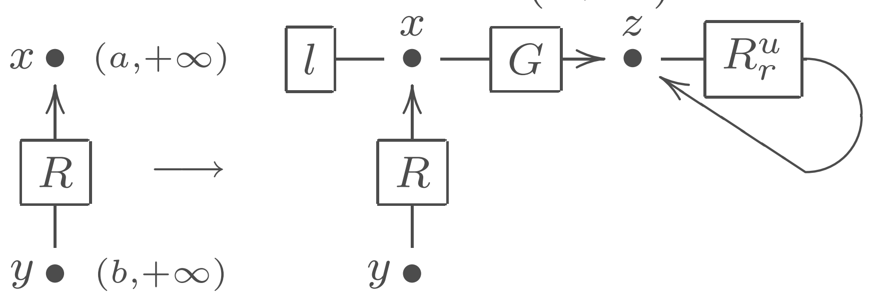


$$x : \mathbf{0} \triangleright l(x) \xrightarrow{x \mapsto (b,+\infty)} l(x)$$

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

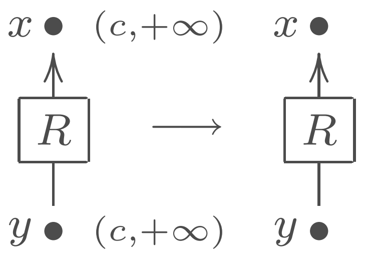
Productions for the ring case study[2]

Create Gate $(r > 0)$



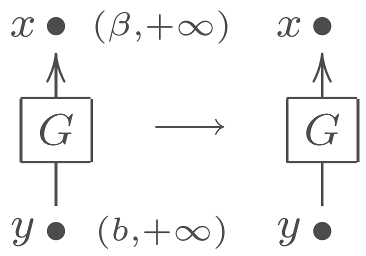
$$x:(\mathbf{0}_H, u), y:\mathbf{0} \triangleright R(x,y) \xrightarrow{\substack{x \mapsto (a,\,+\infty)\\ y \mapsto (b,\,+\infty)}} R(x,y) \mid l(x) \mid G(z,x) \mid R^u_r(z,z)$$

Accept Synchronisation Init



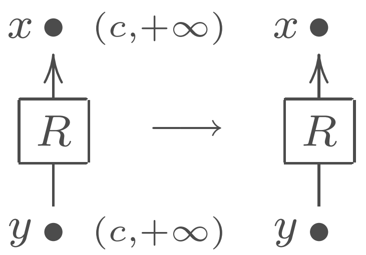
$$x:\mathbf{0}, y:\mathbf{0} \triangleright R(x,y) \xrightarrow{\substack{x \mapsto (c,\,+\infty)\\ y \mapsto (c,\,+\infty)}} R(x,y)$$

Accept Synchronisation Gate



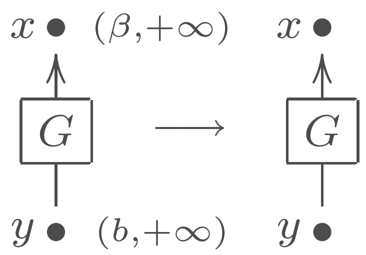
$$x:\mathbf{0}, y:\mathbf{0} \triangleright G(x,y) \xrightarrow{\substack{x \mapsto (\beta,\,+\infty)\\ y \mapsto (b,\,+\infty)}} G(x,y)$$

Accept Synchronisation Init



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

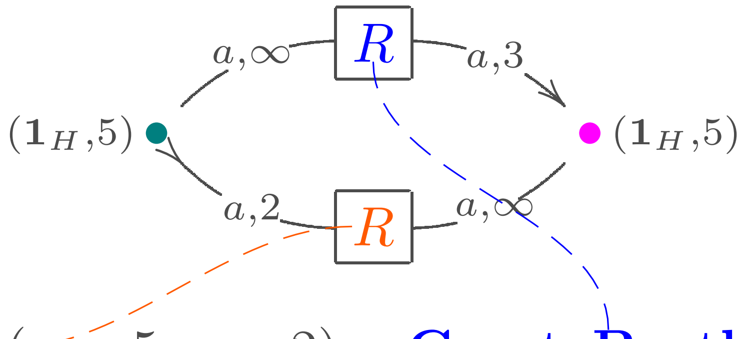
Accept Synchronisation Gate



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

The ring case study

The derivation starts from

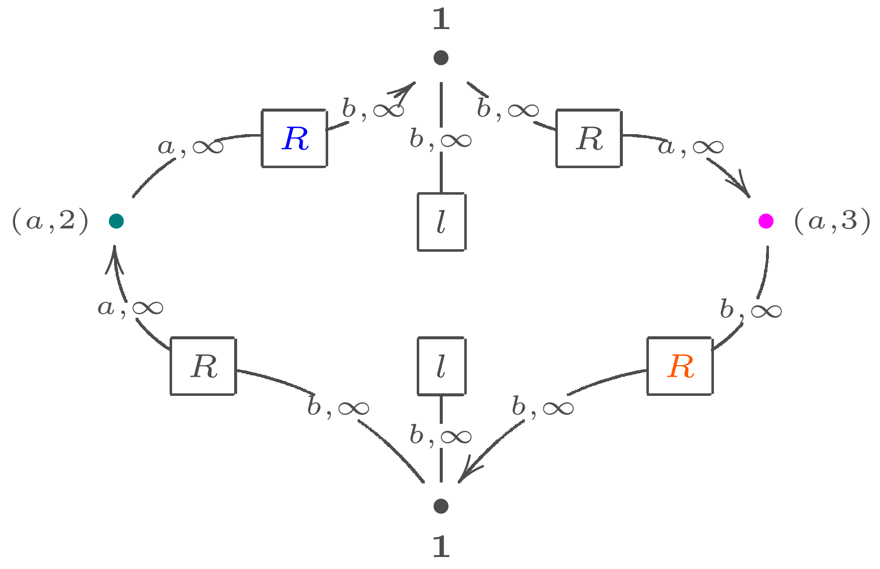


$$\mathbf{CreateBrother}(u = 5, n = 2) \times \mathbf{CreateBrother}(u = 4, n = 3)$$

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,

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

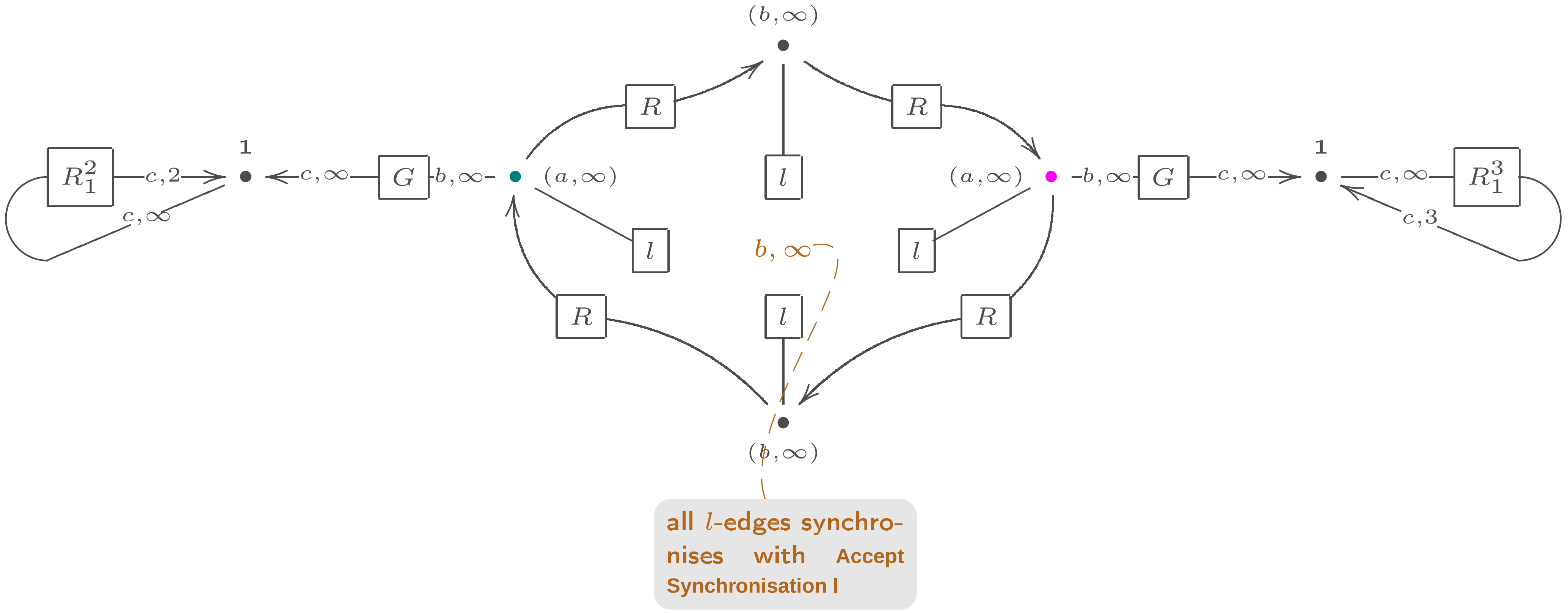
The ring case study



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

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.

The ring case study



Note that ● and ● are now internal.

References

[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.

[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.

[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.

[Hir03] Dan Hirsch. *Graph Transformation Models for Software Architecture Styles*. PhD thesis, Departamento de Computación, Universidad de Buenos Aires, 2003. http://www.di.unipi.it/˜dhirsch.

[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.

[HT05] Dan Hirsch and Emilio Tuosto. SHReQ: A Framework for Coordinating Application Level QoS. In K. Aichernig Bernhard and Beckert Bernhard, editors, *3rd IEEE International Conference on Software Engineering and Formal Methods*, pages 425–434. IEEE Computer Society, 2005.

[LM04] Ivan Lanese and Ugo Montanari. Synchronization algebras with mobility for graph transformations. In *Proc. FGUC'04 – Foundations of Global Ubiquitous Computing*, ENTCS, 2004. To appear.

[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.

[Win85] Glynn Winskel. Synchronization trees. *Theoretical Computer Science*, 34:33–82, May 1985.