# SHReQ: A Framework for Coordinating Application Level QoS


**Dan Hirsch**

**dhirsch@di.unipi.it**

`http://www.di.unipi.it/~dhirsch`

**Emilio Tuosto**

**etuosto@di.unipi.it**

`http://www.di.unipi.it/~etuosto`

Dipartimento di Informatica, Università di Pisa


QAPL: Edinburgh, 2 – 3 April 2005

# Overview of the talk

*“Ayudadme a comprender lo que os digo y os lo explicaré mejor”*

**“Help me in understanding what I’m saying and I will explain it better”**

**(Antonio Machado)**

# Overview

- A few motivations
- Background
  - application level QoS
  - constraint semirings (c-semirings)
  - Synchronised Hyperedge Replacement (SHR)
- Putting things together: SHReQ
  - weighted graphs...
  - ...and their productions
  - synchronising productions for SHReQ
- An example
- Conclusions






# Motivations


**The real technology - behind all of our technologies - is language**

**(N. Fisher)**

# Global Computing

Programming *global systems* is hard because:

- Absence of centralised control (self*)
- Client-Server not enough: P2P
- Administrative domains (Security)
- Interoperability
  - different platforms
  - different devices (e.g. PDA, laptop, mobile phones...)
- “Mobility” (resources & computation)
- Network Awareness
  - Applications are location dependent
  - Locations have different features
  - and allow multiple access policies
- Independently programmed in a distributed environment
- ...






# Global Computing and Services

- 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






# Global Computing and Services

- 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?






# Global Computing and Services

- 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?

- Search and bind wrt application level QoS
  - not low-level performance (e.g., throughput, response time)
  - but application-related, e.g.
    - price services
    - payment mode
    - data available in a given format






# Our approach in brief

WAN programming is not just **go**(P), $\bar{s}\langle x \rangle$ or $s(y)$

- Lifting QoS issues to application level...
- ...for programming global computers
- with programmable application level QoS
- and develop proof techniques and tools

First steps (extending Klaim) in [DFM$^+$03, DFM$^+$05]

We are currently defining SHReQ, an (hyper)graph model which exploits c-semiring for

- expressing application level QoS and
- for coordinating activities...
- ...by synchronisation on c-semiring values






# Background

***"During my nine years at the elementary schools***
***I was not able to teach anything to my professors"***

**(Bertolt Brecht)**

# Constraint Semirings

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

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

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

# Constraint Semirings

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

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

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

# Constraint Semirings

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

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

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

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

# Examples of c-semirings

**Example 1 (Priority)** $P = \langle N, max, min, 0, \infty \rangle$ *where* $N$ *is the set of natural numbers with infinity.*

**Example 2 (Broadcast)** $B = \langle A \cup \overline{A} \cup \{\mathbf{1}, \mathbf{0}, \bot\}, \mathbf{0}, \mathbf{1}, +, \star \rangle$ *where* $A$ *is a set of actions,* $\overline{A} = \{\overline{a} \mid a \in A\}$ *are the* coactions *and*

$$\forall a \in Act . a \star \overline{a} = \overline{a} \wedge a \star a = a$$

$$\forall a, b \in Act \cup \overline{Act} \cup \{\bot\} : b \notin \{a, \overline{a}\} \implies a \star b = \bot$$

*the corresponding commutative rules plus the ones for* $\mathbf{0}$ *and* $\mathbf{1}$
$+$ *also obeys the axioms*

$$\left.\begin{array}{l} a + a = a \\ a + b = \bot \end{array}\right\} \quad \forall a, b \in Act \cup \overline{Act} \cup \{\bot\} . b \neq a$$

**Proposition 1** *Cartesian product of c-semirings is a c-semiring.*

e.g., $BP = B \times P$ is the c-semiring of broadcast with priorities






# Another bunch of c-semiring examples

C-semirings structures can been 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






# Hypergraphs model

- Distributed systems as graphs [CM83, DM87]
  - explicitly describe topology
  - and are suitable for expressing multiparty synchronisation
  - we use Synchronised Hyperedge Replacement (SHR)
- Edge replacement for graph rewritings [Fed71, Pav72]
- Edge replacement/distributed constraint solving problem [MR96]
- Graphs grammars for software architecture styles [HIM00]
- SHR with mobility for nominal calculi [HM01, Hir03]
- Extension to node fusions [FMT01, Tuo03]...
- ...wich accounts for a concurrent semantics of the Fusion calculus [LM03]






# Hypergraphs model$^2$

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

The metaphor is

- “Global computers *as* Hypergraphs”
- “Global computing *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






# SHR...naively

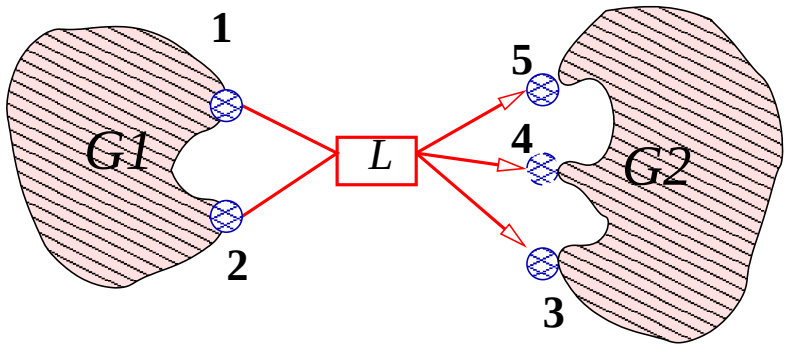
$L \rightarrow G$





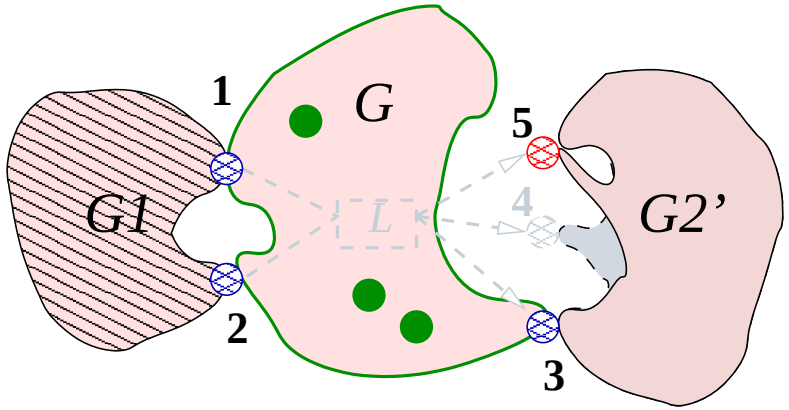

# SHR...naively

$L \rightarrow G$



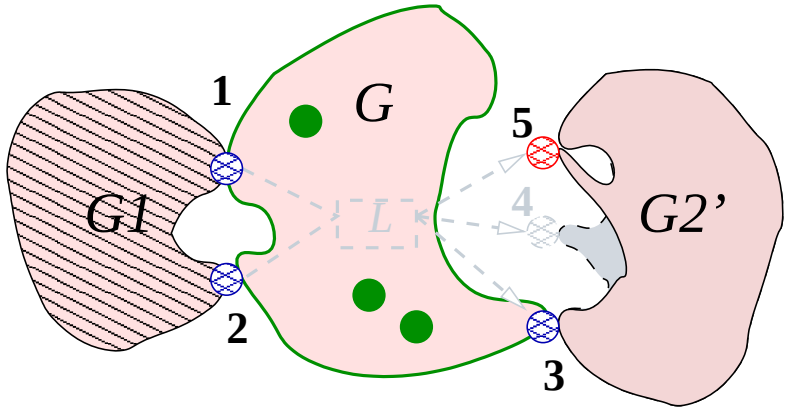
# SHR...naively

$L \rightarrow G$



# SHR...naively

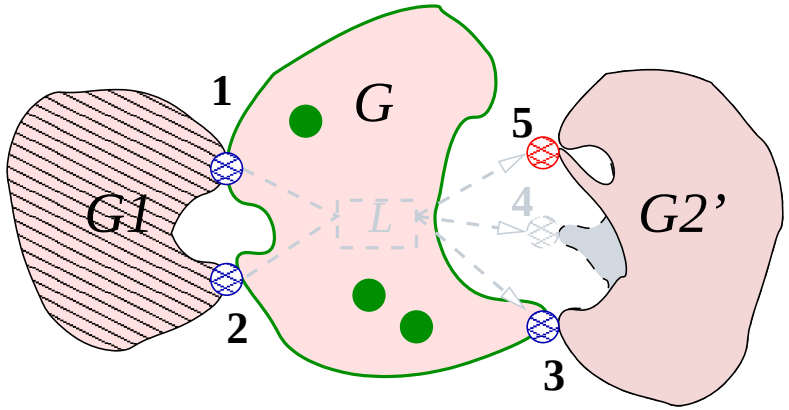
$L \rightarrow G$




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

# SHR...naively

$L \rightarrow G$



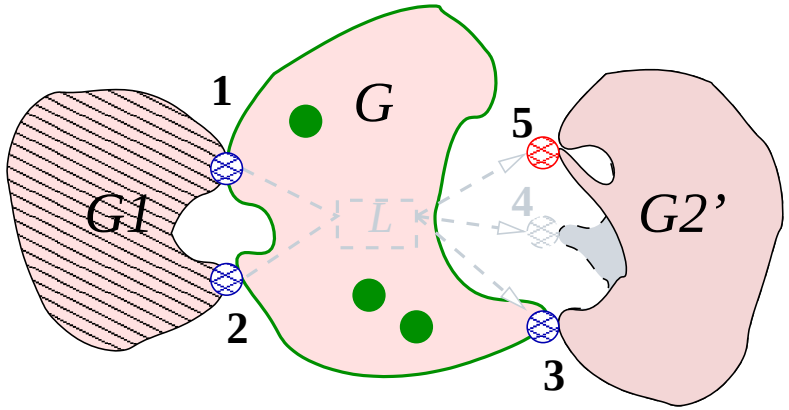

- Edge replacement: local
- Synchronisation as distributed constraint solving
- Multiple synchronisation
- New node creation
- Node fusion: model of mobility and communication
- Truly concurrent semantics

Benefi ts:

- Uniform framework for $\pi$, $\pi$-I, Fusion
- LTS for Ambient ...
- ... for Klaim ...

# SHR...naively

$L \rightarrow G$




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

Benefi ts:

- Uniform framework for $\pi$, $\pi$-I, Fusion
- LTS for Ambient ...
- ... for Klaim ...
- ... and *path reservation* for a Klaim extension
- expressive for distributed coordination
- wireless networks [Tuo05]

# SHReQ

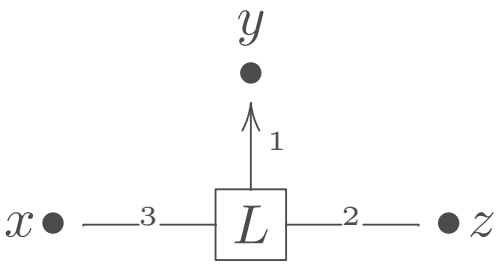

*"[...] Shrek cuts a deal with Farquaad and sets out to rescue the beautiful Princess Fiona to be Farquaad's bride."*

*(from http://www.shrek.com)*

# Hyperedges and Hypergraphs Syntax

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

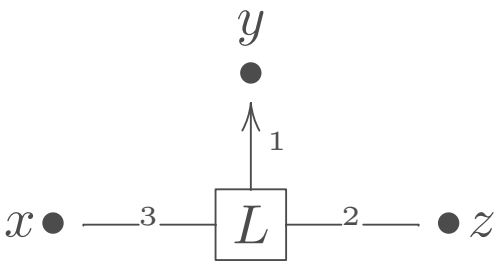
$L : 3,\ L(y, z, x),$



# Hyperedges and Hypergraphs Syntax

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

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

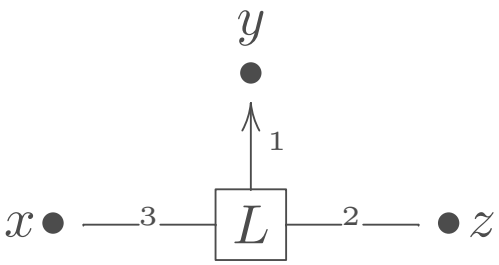



$$G ::= nil \mid L(\tilde{x}) \mid G|G \mid \nu\ y.G$$

# Hyperedges and Hypergraphs Syntax

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

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



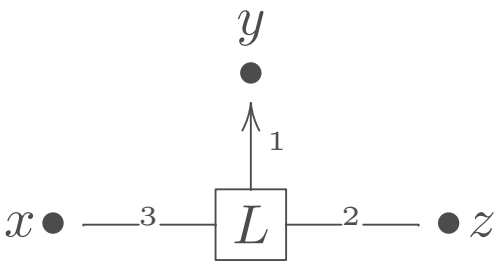

$$G ::= nil \mid L(\tilde{x}) \mid G|G \mid \nu\, y.G$$

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

# 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:

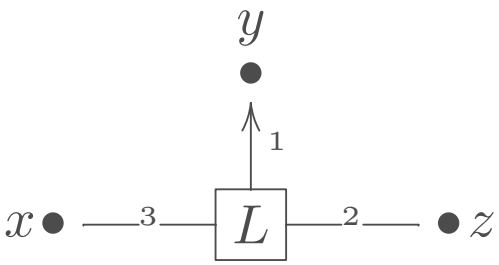
$$L : 3, \quad M : 2$$

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

# Hyperedges and Hypergraphs Syntax

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

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



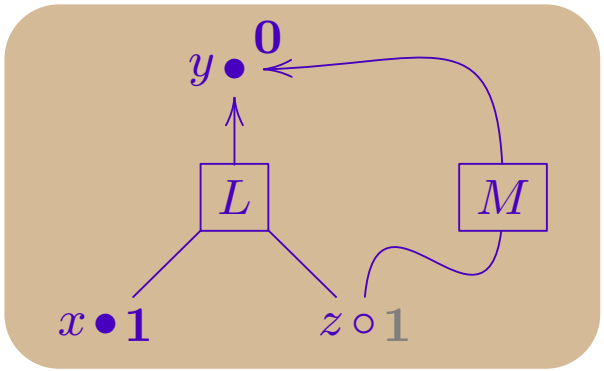

$$G ::= nil \mid L(\tilde{x}) \mid G|G \mid \nu\ y.G$$

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

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

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

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

- $\tilde{x}$ is a tuple of pairwise distinguished nodes and $L : |\tilde{x}|$
- $\chi : \{|\tilde{x}|\} \to S$ 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 $\chi \triangleright L(\tilde{x}) \xrightarrow{\Lambda} G$

- $\tilde{x}$ 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 $\Lambda$: those nodes appearing in a requirement in the range of $\Lambda$.
  The set of new nodes of $\Lambda$ 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 $\langle S, +, \star, 0, 1 \rangle$ is a fixed c-semiring)

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

- $\tilde{x}$ 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 $\Lambda$: those nodes appearing in a requirement in the range of $\Lambda$
  The set of new nodes of $\Lambda$ is $\mathrm{new}(\Lambda) = \mathrm{n}(\Lambda) \setminus \mathrm{dom}(\Lambda)$
- $G$ is a graph s.t. $\mathrm{fn}(G) \subseteq \{\!|\tilde{x}|\!\} \cup \mathrm{n}(\Lambda)$



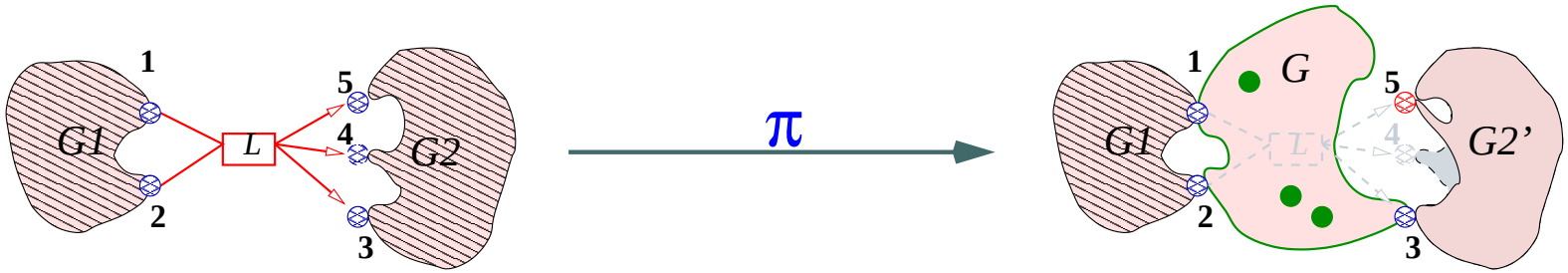



# Interpreting SHReQ productions

Consider

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

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

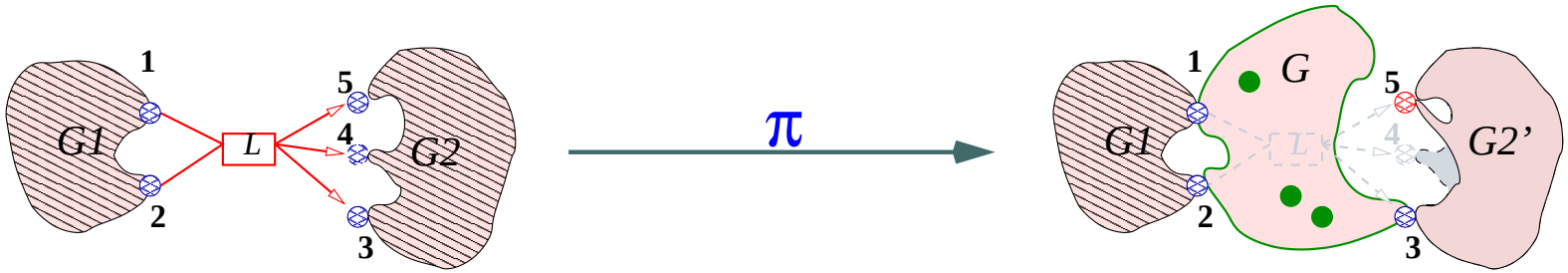


# Interpreting SHReQ productions

Consider

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

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



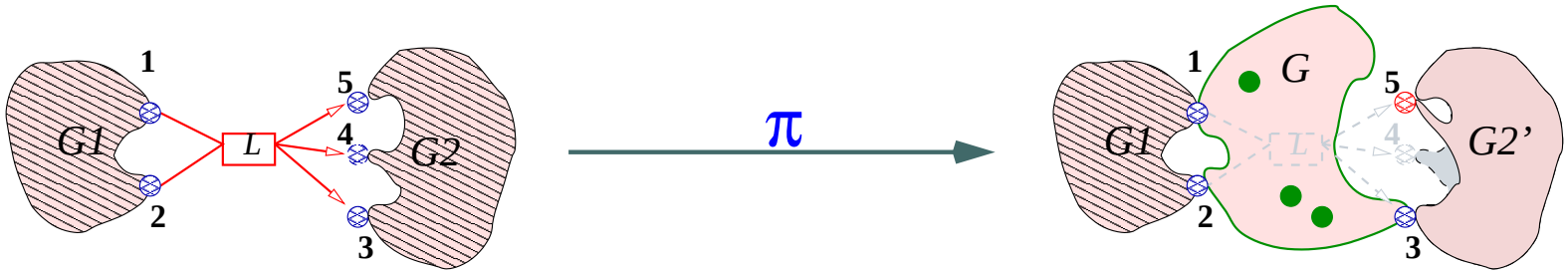

- Replacing $L$ with $G$ in $H$ according to $\pi$ requires that $H$ satisfies the conditions expressed by $\chi$ on the attachment nodes of $L$.

# 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.:



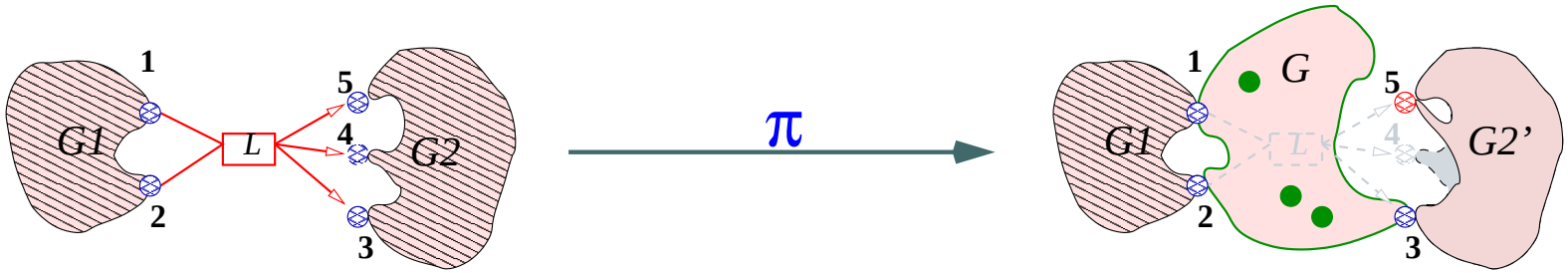

- Replacing $L$ with $G$ in $H$ according to $\pi$ requires that $H$ satisfies the conditions expressed by $\chi$ on the attachment nodes of $L$.

# 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 $\pi$ requires that $H$ satisfies the conditions expressed by $\chi$ on the attachment nodes of $L$.

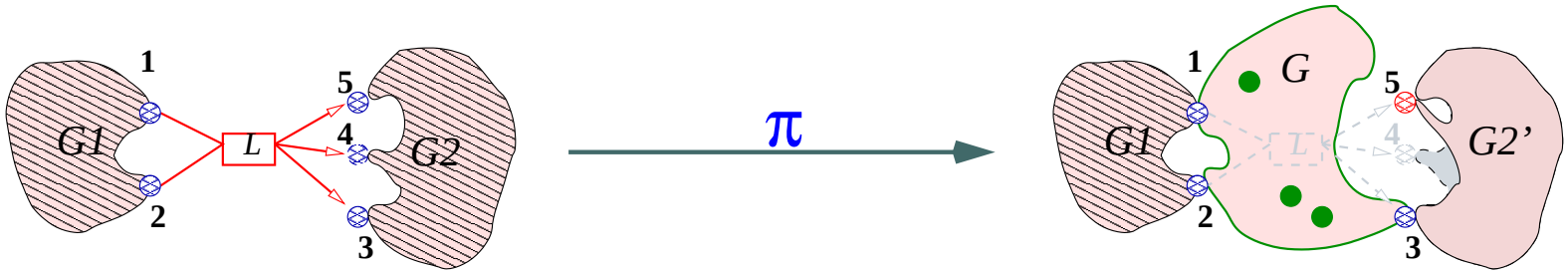
- Once $\chi$ is satisfied in $H$, $L(\tilde{x})$ contributes to the rewriting by offering $\Lambda$ in the synchronisation with all the edges connected to nodes in $\tilde{x}$.

# 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 $\pi$ requires that $H$ satisfies the conditions expressed by $\chi$ on the attachment nodes of $L$.

- Once $\chi$ is satisfied in $H$, $L(\tilde{x})$ contributes to the rewriting by offering $\Lambda$ 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 $\Omega$ 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

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

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

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

# 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 $\Omega$ 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

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

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

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

# 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 $\Omega$ 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

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

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

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

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






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






# 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 $\Omega$ is the function $\underline{\Omega} : \mathrm{dom}(\Omega) \to \mathcal{R}$ defined as

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

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

The weighting function induced by $\Gamma$ and $\Omega$ is

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

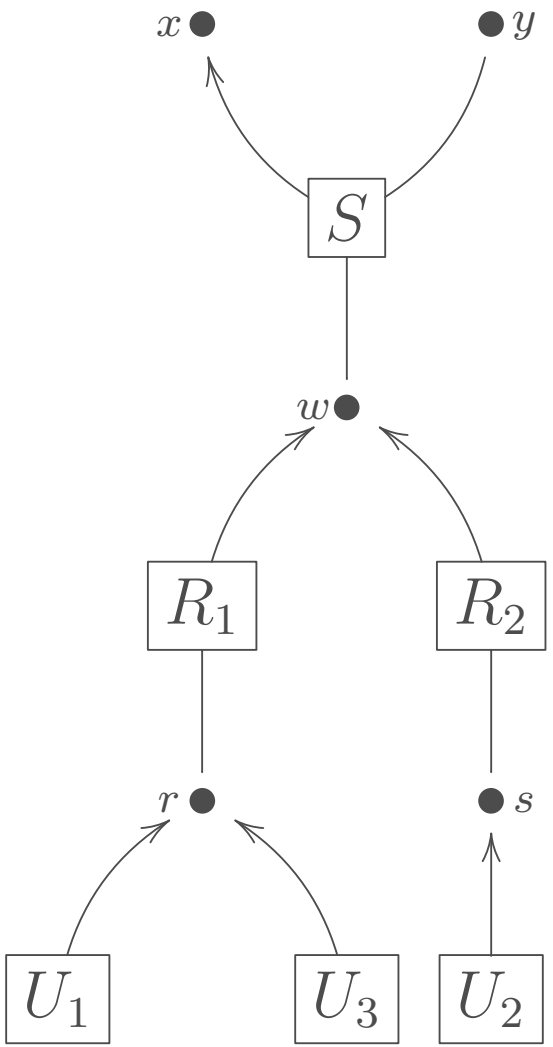
$$\Gamma_\Omega(x) = \begin{cases} \mathbf{1}, & x \in \mathrm{new}(\underline{\Omega}) \\ \Gamma(x), & |\Omega@x| = 1 \\ \Gamma_\Omega(x) = \underline{\Omega}(x)\downarrow_1, & \text{otherwise} \end{cases}$$

The weighting function computes the new weights of graphs after the synchronisations induced by $\Omega$.






# Applying SHReQ's semantics



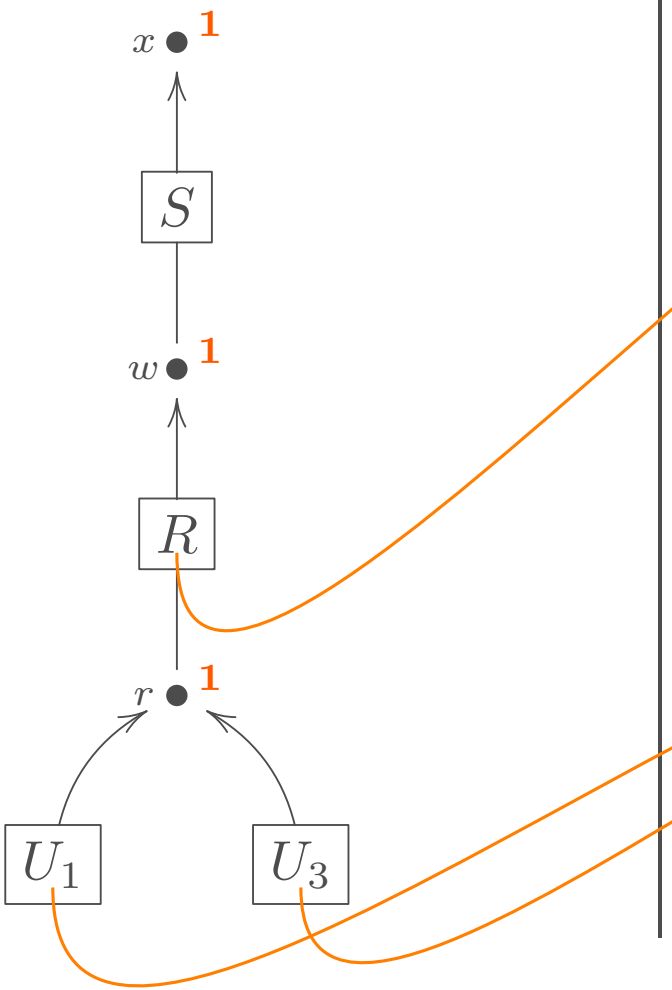

Let's compose broadcast and priority c-semirings:

$$Sync_{BP} = \{\mathbf{1}\} = \{\langle \mathbf{1}, \infty \rangle\}$$
$$Fin_{BP} = \{\mathbf{1}\} \cup \{(\overline{a}, n) | \overline{a} \in W, n > 0\}$$
$$NoSync_{BP} = \{\mathbf{0} = \langle \mathbf{0}, 0 \rangle, \bot\} \cup \{(a, 0) | a \in W\} \cup \{(\mathbf{0}, n) | n \in N\}$$

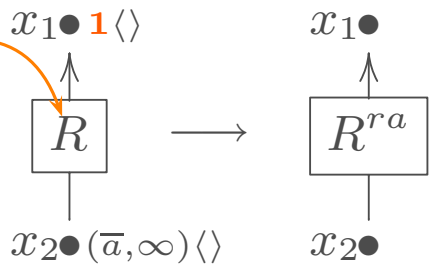
- The only value in $Sync_{BP}$ is $\mathbf{1}$
- $Fin_{BP}$ are all coactions together whith any valid priority $n > 0$
- $NoSync_{BP}$ contains all pairs with at least one "zero" in their components.

# Selecting productions[2]

The initial graph

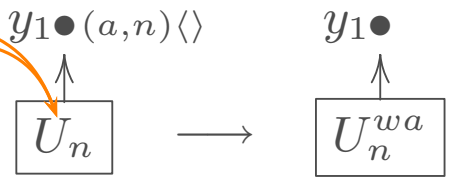



**Checking alarm**



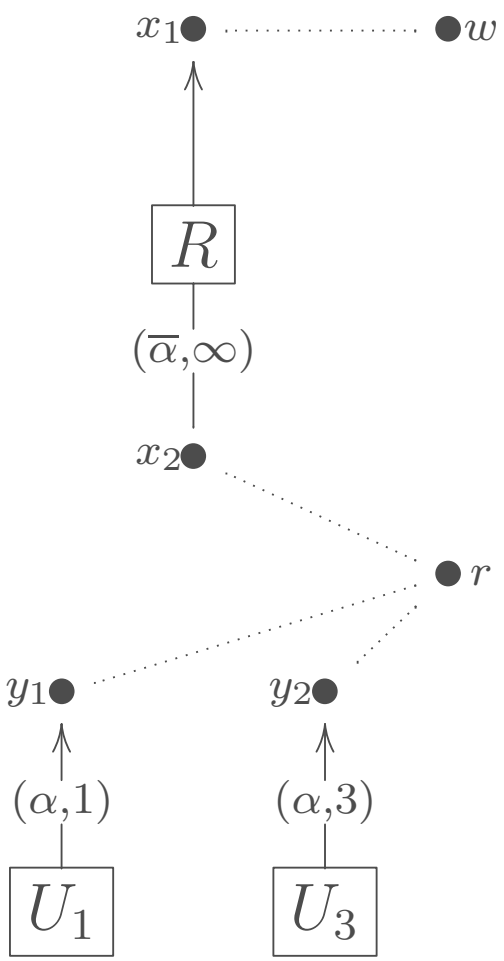

$$x_1, x_2 : \mathbf{0} \triangleright R(x_1, x_2) \xrightarrow{(x_1,\mathbf{1},\langle\rangle)\ (x_2,(\overline{a},\infty),\langle\rangle)} R^{ra}(x_1, x_2)$$

**Sending alarm**

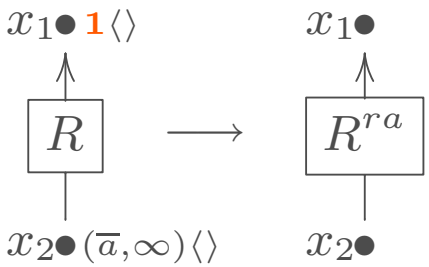



$$y_1 : \mathbf{0} \triangleright U_n(y_1) \xrightarrow{(y_1,(a,n),\langle\rangle)} U_n^{wa}(y_1)$$

# Selecting productions[2]

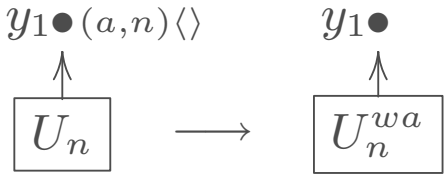



**Checking alarm**




$$x_1, x_2 : \mathbf{0} \triangleright R(x_1, x_2) \xrightarrow{(x_1,\mathbf{1},\langle\rangle)\ (x_2,(\overline{a},\infty),\langle\rangle)} R^{ra}(x_1, x_2)$$

**Sending alarm**




$$y_1 : \mathbf{0} \triangleright U_n(y_1) \xrightarrow{(y_1,(a,n),\langle\rangle)} U_n^{wa}(y_1)$$

$$(\alpha, 1) \star (\alpha, 3) \star (\overline{\alpha}, \infty) = (\overline{\alpha}, 1)$$

# A SHReQ synchronisation

$$\frac{r:\mathbf{0} \triangleright U_1(r) \xrightarrow{(r,(\alpha,1),\langle\rangle)} U_1^{wa}(r)\,\{r/y_1\}}{r:\mathbf{1} \vdash U_1(r) \xrightarrow{(r,(\alpha,1),\langle\rangle)} r:\mathbf{1} \vdash U_1^{wa}(r)}\ (\mathsf{REN}_1)$$

$$\frac{w, r:\mathbf{0} \triangleright R(w,r) \xrightarrow[(r,(\overline{\alpha},\infty),\langle\rangle)]{(w,\mathbf{1},\langle\rangle)} R^{ra}(w,r)\,\{w/x_1, r/x_2\}}{w, r:\mathbf{1} \vdash R(w,r) \xrightarrow[(r,(\overline{\alpha},\infty),\langle\rangle)]{(w,\mathbf{1},\langle\rangle)} w, r, z:\mathbf{1} \vdash R^{ra}(w,r)}\ (\mathsf{REN}_2)$$

$$w, r:\mathbf{1} \vdash U_1(r) \mid R(w,r) \xrightarrow[(r,(\overline{\alpha},1),\langle\rangle)]{(w,\mathbf{1},\langle\rangle)} w, z:\mathbf{1}, r:(\overline{\alpha},1) \vdash U_1^{wa}(r) \mid R^{ra}(w,r)\ (\mathsf{COM}_1)$$

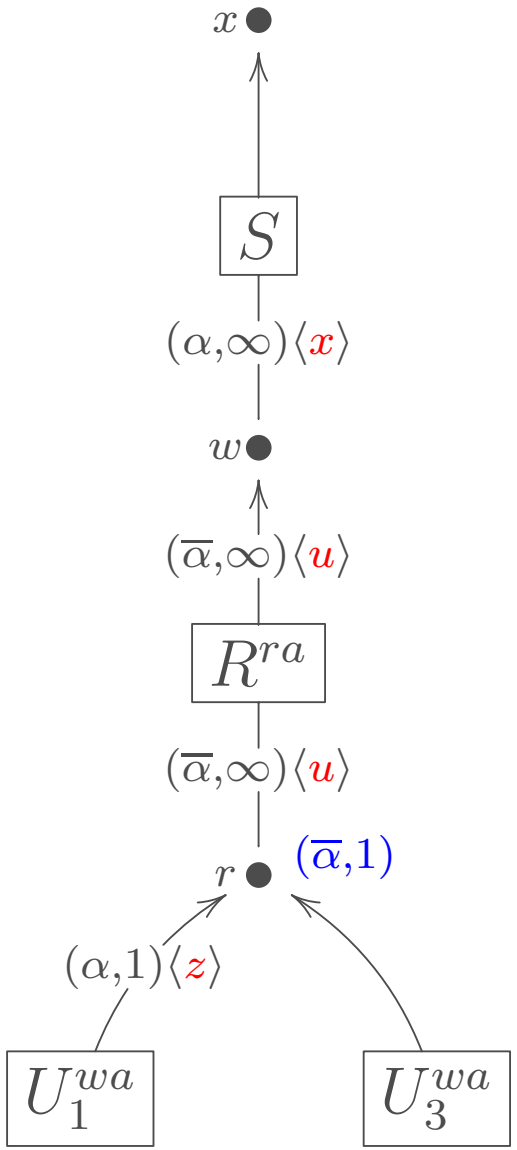
$$\frac{r:\mathbf{0} \triangleright U_3(r) \xrightarrow{(r,(\alpha,3),\langle\rangle)} U_3^{wa}(r)\{r/y_2\}}{r:\mathbf{1} \vdash U_3(r) \xrightarrow{(r,(\alpha,3),\langle\rangle)} r:\mathbf{1} \vdash U_3^{wa}(r)}\ (\mathsf{REN}_3)$$

$$w, r:\mathbf{1} \vdash U_1(r) \mid R(w,r) \mid U_3(r) \xrightarrow[(r,(\overline{\alpha},1),\langle\rangle)]{(w,\mathbf{1},\langle\rangle)} w, z:\mathbf{1}, r:(\overline{\alpha},1) \vdash U_1^{wa}(r) \mid R^{ra}(w,r) \mid U_3^{wa}(r)\ (\mathsf{COM}_2)$$

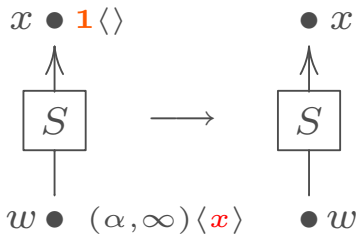





# Exploiting the mgu

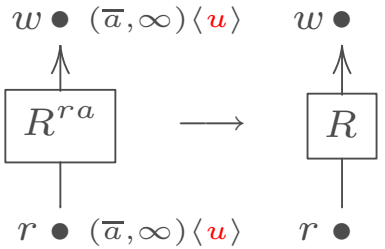



**Sending ambulance assistance**



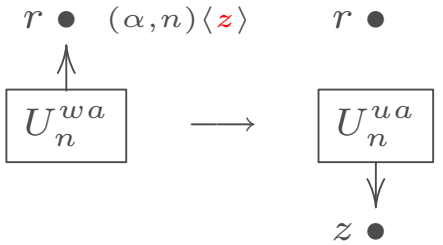

$$x,w:\mathbf{0} \triangleright S(x,w) \xrightarrow{(x,\mathbf{1},\langle\rangle)\ (w,(\alpha,\infty),\langle x\rangle)} S(x,w)$$

**Forwarding alarm**




$$w:\mathbf{0}, r:(\overline{a},0) \triangleright R^{ra}(w,r) \xrightarrow{(w,(\overline{a},\infty),\langle u\rangle)\ (r,(\overline{a},\infty),\langle u\rangle)} R(w,r)$$

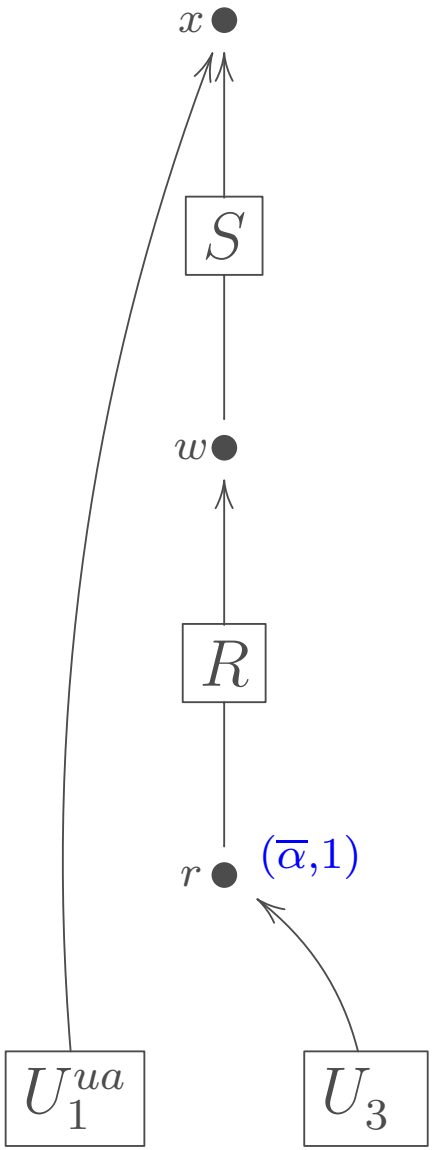
**Receiving ambulance assistance**



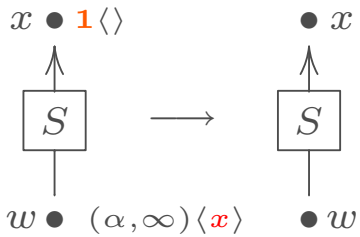

$$r:(\overline{\alpha},n) \triangleright U_n^{wa}(r) \xrightarrow{(r,(\alpha,n),\langle z\rangle)} U_n^{ua}(z)$$

# Exploiting the mgu

$\rho = x/u, x/z$

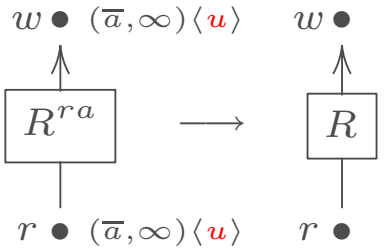



**Sending ambulance assistance**



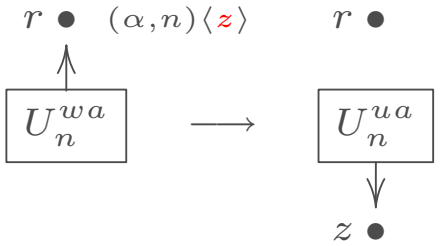

$$x,w:\mathbf{0} \triangleright S(x,w) \xrightarrow{(x,\mathbf{1},\langle\rangle)\ (w,(\alpha,\infty),\langle x\rangle)} S(x,w)$$

**Forwarding alarm**




$$w:\mathbf{0}, r:(\overline{a},0) \triangleright R^{ra}(w,r) \xrightarrow{(w,(\overline{a},\infty),\langle u\rangle)\ (r,(\overline{a},\infty),\langle u\rangle)} R(w,r)$$

**Receiving ambulance assistance**




$$r:(\overline{\alpha},n) \triangleright U_n^{wa}(r) \xrightarrow{(r,(\alpha,n),\langle z\rangle)} U_n^{ua}(z)$$

# Conclusions

*“Run, rabbit run*
*Dig that hole, forget the sun*
*And when at last the work is done*
*Don’t sit down it’s time to dig another one*

*(Breathe, Roger Waters)*

# Final Remarks

- We have presented SHReQ's syntax and semantics
- The main features are
  - general multiparty synchronisations c-semiring based
  - SHReQ uses application level QoS as coordination mechanism of distributed activities
- Surprisingly, they fit the design principles in [Mil96]

We plan to

- Validate our design choice at the light of [Mil96]
- Develop formal methods based on SHReQ for specifying distributed applications that can bargain their non-functional requirements
- Hopefully, develop verification techniques (e.g., model checking) on SHReQ [FL04]
- ...






# 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. *JACM*, 44(2):201–236, March 1997.

[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 *LNCS*, pages 20–38. Springer, 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 Verification – Theory and Practice – Honoring Zohar Manna's 64th Birthday*, volume 2772 of *LNCS*, pages 436–479. Springer, 2003.

[DFM+05] Rocco De Nicola, Gianluigi Ferrari, Ugo Montanari, Rosario Pugliese, and Emilio Tuosto. A Basic Calculus for Modelling Service Level Agreements. In *International Conference on Coordination Models and Languages*, LNCS. Springer, 2005. To appear.

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

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

[FL04] Gianluigi Ferrari and Alberto Luch-Lafuente. A Logic for Graphs with QoS. In *First International Workshop on Views On Designing Complex Architectures*, ENTCS, Bertinoro, Italy, September 2004. Elsevier. To appear.

[FMT01] Gianluigi Ferrari, Ugo Montanari, and Emilio Tuosto. A LTS Semantics of Ambients via Graph Synchronization with Mobility. In *ICTCS*, volume 2202 of *LNCS*, Torino (Italy), October 4-6, 2001. Springer.

[HIM00] Dan Hirsch, Paola Inverardi, and Ugo Montanari. Reconfi guration of Software Architecture Styles

with Name Mobility. In Antonio Porto and Gruia-Catalin Roman, editors, *Coordination 2000*, volume 1906 of *LNCS*, pages 148–163. Springer, 2000.

[Hir03] Dan Hirsch. *Graph Transformation Models for Software Architecture Styles*. PhD thesis, Departamento de Computación, UBA, 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 *CONCUR*, volume 2154 of *LNCS*, pages 121–136, Aalborg, Denmark, 2001. Springer.

[LM03] Ivan Lanese and Ugo Montanari. A graphical fusion calculus. In *Proceedings of CoMeta Final Workshop*, Electronic Notes in Theoretical Computer Science, 2003. To appear.

[Mil96] Robin Milner. Process Constructors and Interpretations. In H.-J. Kugler, editor, *Information Processing*. Elsevier Science Publishers, 1996.

[MR96] Ugo Montanari and Francesca Rossi. Graph rewriting and constraint solving for modelling distributed systems with synchronization. In P. Ciancarini and

C. Hankin, editors, *Proceedings of the First International Conference COORDINATION '96, Cesena, Italy*, volume 1061 of *LNCS*. Springer, April 1996.

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

[Tuo03] Emilio Tuosto. *Non-Functional Aspects of Wide Area Network Programming*. PhD thesis, Dipartimento di Informatica, Università di Pisa, May 2003. TD-8/03.

[Tuo05] Emilio Tuosto. Tarzan: Communicating and Moving in Wireless Jungles. In Aantonio Cerone and Alessandra Di Pierro, editors, *2nd Workshop on Quantitative Aspects of Programming Languages*, volume 112 of *ENTCS*, pages 77–94. Elsevier, January 2005.