Specification and verification in service oriented computing

Emilio Tuosto

Dipartimento di Informatica

Università di Pisa

Lecturership interview, Leicester University

August 2005

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

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...
    - multi-layered: overlay networks
  - Designers, programmers and end-users may ignore the stratification and complexity

- 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...
    - multi-layered: 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 mostly ad-hoc way

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

- SOC architectures are
  - distributed
  - interconn...
  - bas... com...
    - IP...
    - mu... netw...
  - Designer..., programmers and end-users may ignore the stratification and complexity

...ups

...coordination

...nomous"

- independent
- mobile/stationary
- "interconnected" through interfaces

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

- How do we program SOC?
- Can search/bind be dynamically done at run-time?
- What is considered relevant for searching services?
- What about verification?

Specification

- Process calculi
  - Klaim
  - cIP
  - $CCS_{\|}$
  - Symbolic Fusion calculus
- SHR
- QoS as c-semirings
- Spatial logics wrt
  - HD-automata
  - SHR

Verification

- HD-automata
  - operational model
  - minimisation
- cIP & $\mathcal{PL}$
- Spatial model checking

Implementation

- Klaim
- Aspasya
- SHR: Gredy
- Mihda
- JTWS & JSAGA
- Long running transactions

Works with several peoples:

Inside Pisa:

- Pisa Department: Andrea Bracciali, Roberto Bruni, Gianluigi Ferrari, Dan Hirsch, Ivan Lanese, Hernán Melgratti, Ugo Montanari, Daniele Strollo
- Pisa CNR: Stefania Gnesi

Outside Pisa:

- Florence: Rocco De Nicola and Rosario Pugliese
- Uppsala: Kidane Yemane & Björn Victor
- Lisbon: Hugo Torres Viera

Specification	Verification	Implementation

Specification

- Process calculi
  - Klaim... $\mathcal{K}o\mathcal{S}$
  - cIP
  - $\mathrm{CCS}_{\parallel}$
  - Symbolic Fusion calculus
- SHR
- QoS as c-semirings
- Spatial logics wrt
  - HD-automata
  - SHR

Verification

- HD-automata
  - operational model
  - minimisation
- cIP & $\mathcal{PL}$
- Spatial model checking

Implementation

- Klaim
- Aspasya
- SHR: Gredy
- Mihda
- JTWS & JSAGA
- Long running transactions

$\mathcal{K}oS$

Joint works with

R. De Nicola, G. Ferrari, U. Montanari, R. Pugliese

Ὀμνύω Ἀπόλλωνα ἰητρὸν καὶ Ἀσκληπιὸν καὶ Ὑγείαν καὶ Πανάκειαν καὶ θεοὺς πάντας ἵστορας ποιεύμενος ἐπιτελέα ποιήσειν κατὰ δύναμιν καὶ κρίσιν ἐμὴν ὅρκον τόνδε καὶ ξυγγραφὴν τήνδε ....

διαιτήμασι τε χρήσομαι ἐπ'ὠφελείῃ καμνόντων κατὰ δύναμιν καὶ κρίσιν ἐμὴν ἐπὶ δηλήσει δὲ καὶ ἀδικίῃ εἴρξειν.

οὐ δώσω δὲ οὐδὲ φάρμακον οὐδενὶ αἰτηθεὶς θανάσιμον οὐδὲ ὑφηγήσομαι ξυμβουλίην τοιήνδε· ὁμοίως δὲ οὐδὲ γυναικὶ πεσσὸν φθόριον δώσω.

ἁγνῶς δὲ καὶ ὁσίως διατηρήσω βίον ἐμὸν καὶ τέχνην ἐμήν.

οὐ τεμέω δὲ οὐδὲ μὴν λιθιῶντας, ἐκχωρήσω δὲ ἐργάτῃσιν ἀνδράσιν πρήξιος τῆσδε.

ἐς οἰκίας δὲ ὁκόσας ἂν ἐσίω, ἐσελεύσομαι ἐπ' ὠφελείῃ καμνόντων ἐκτὸς ἐὼν πάσης ἀδικίης ἑκουσίης καὶ φθορίης τῆς τε ἄλλης καὶ ἀφροδισίων ἔργων ἐπί τε γυναικείων σωμάτων καὶ ἀνδρείων ἐλευθέρων τε καὶ δούλων.

ἃ δ' ἂν ἐν θεραπείῃ ἢ ἴδω ἢ ἀκούσω ἢ καὶ ἄνευ θεραπηίης κατὰ βίον ἀνθρώπων, ἃ μὴ χρή ποτε ἐκλαλέεσθαι ἔξω, σιγήσομαι ἄρρητα ἡγεύμενος εἶναι τὰ τοιαῦτα.

ὅρκον μὲν οὖν μοι τόνδε ἐπιτελέα ποιέοντι καὶ μὴ ξυγχέοντι εἴη ἐπαύρασθαι καὶ βίου καὶ τέχνης δοξαζομένῳ παρὰ πᾶσιν ἀνθρώποις ἐς τὸν αἰεὶ χρόνον, παραβαίνοντι δὲ καὶ ἐπιορκοῦντι τἀναντία τουτέων.

Oath of Hippocrates

Lifting QoS issues to applications

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

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

Lifting QoS issues to applications

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

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

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

Lifting QoS issues to applications

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

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

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 abstracted as constraint-semiring [BMR95, BMR97]

- for coordinating mobility
- and synchronisations

C-semiring are particularly indicated because

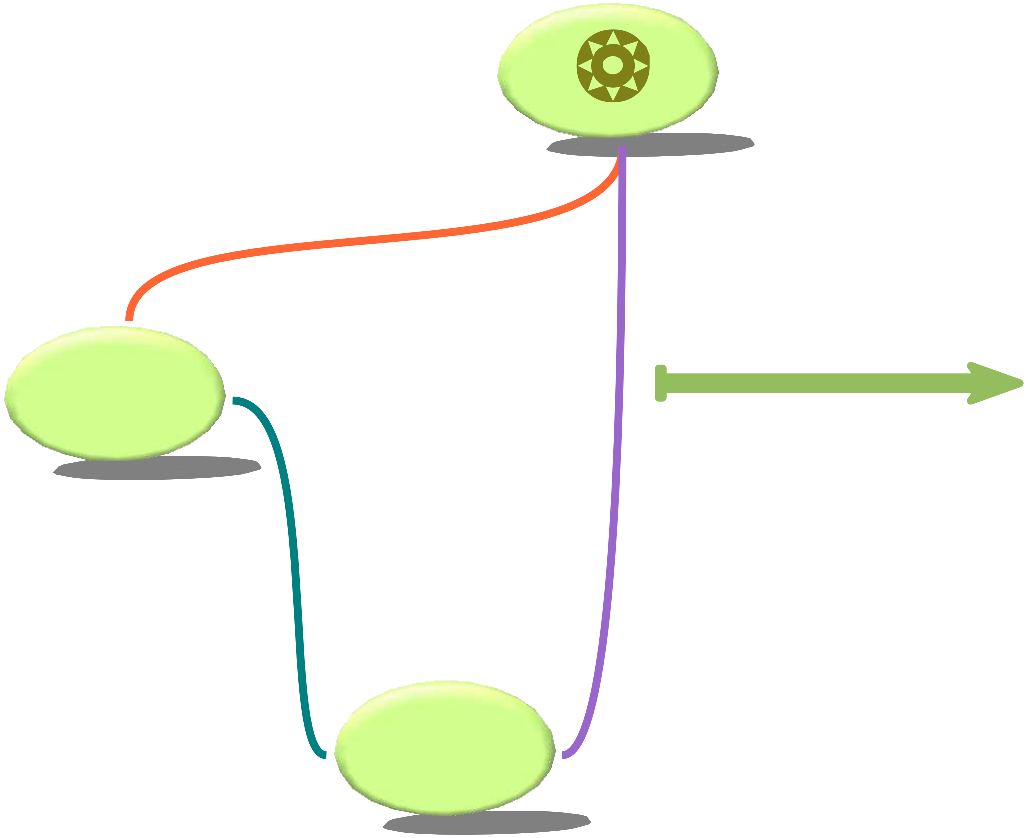
- they have an implicitly defined partial order
- their structure is preserved by many mathematical operations...
- ...in particular cartesian product of c-semirings is a c-semiring
- hence c-semirings are suitable for multi-criteria

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

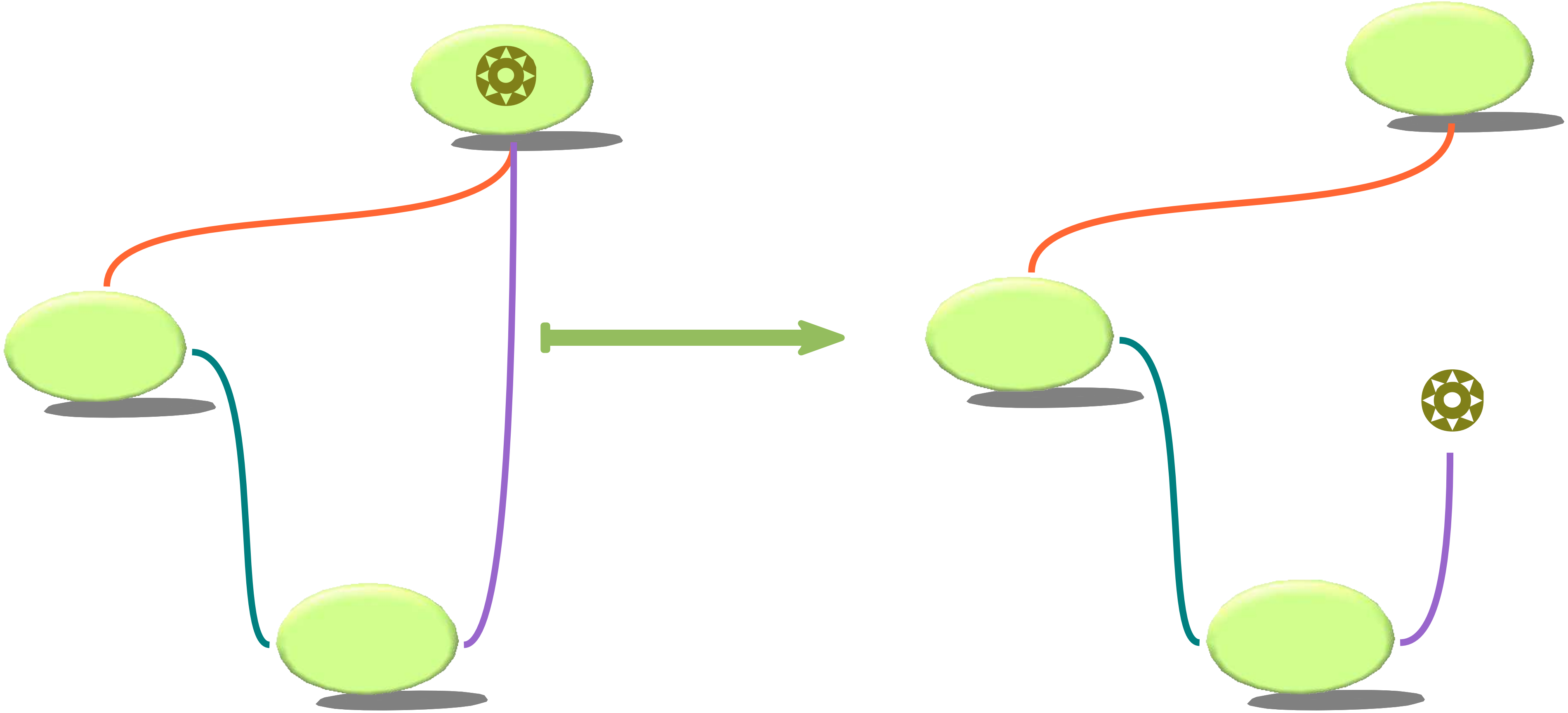
- $\mathcal{K}oS$ builds on Klaim (e.g., processes are localised)
- 'cause it naturally supports a peer-to-peer programming model
- $\mathcal{K}oS$ primitives handle QoS values as first class entities
- $\mathcal{K}oS$ semantics ensures that the QoS values are respected during execution

- 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

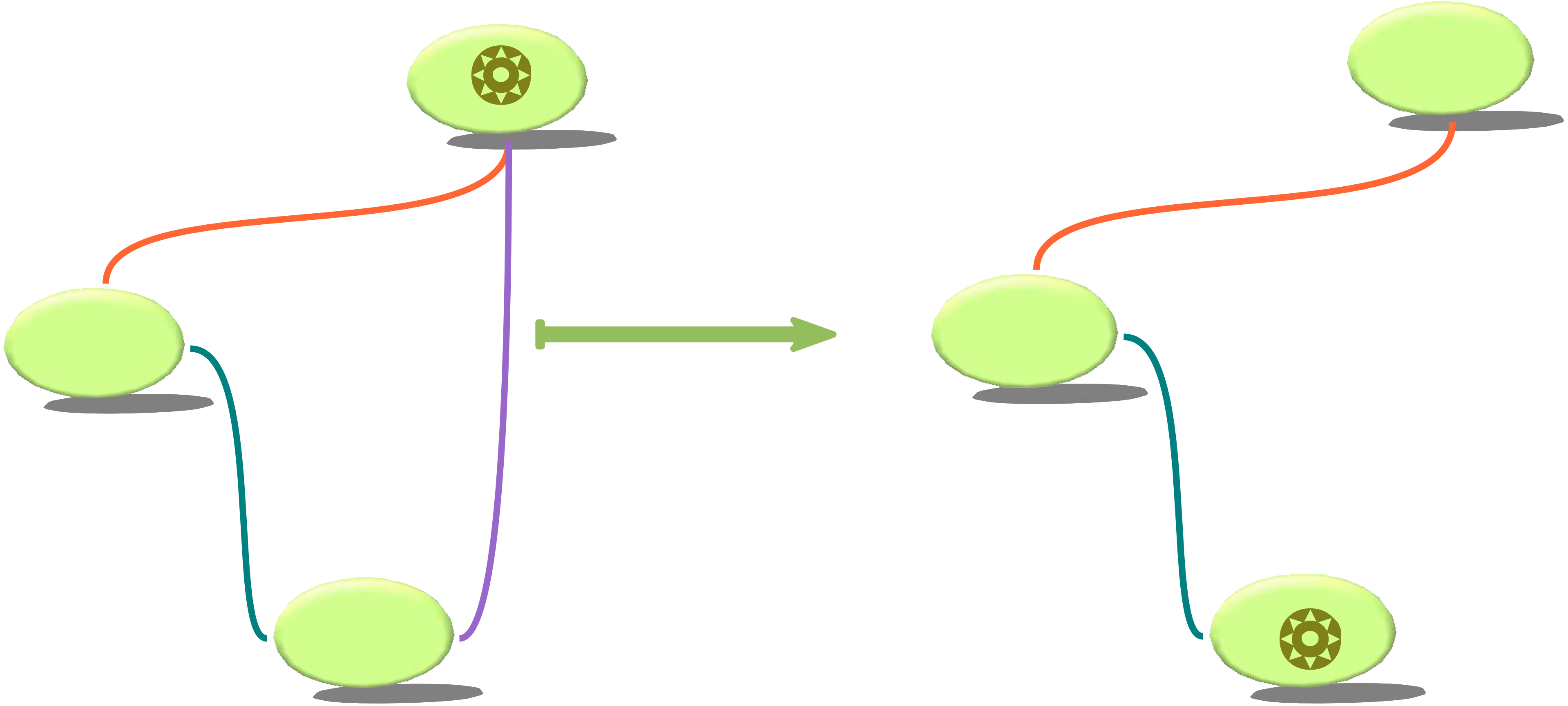
$\mathcal{K}oS$ ... graphically



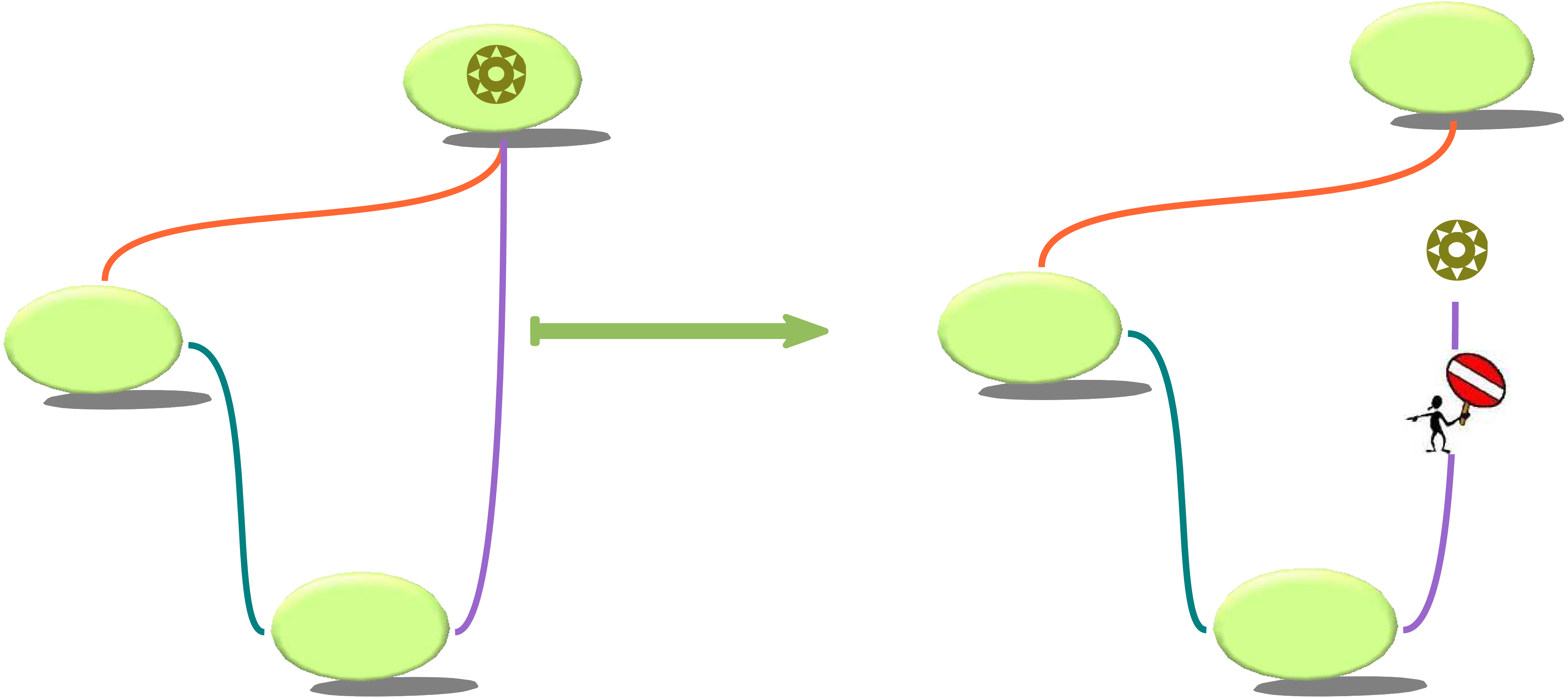
$\mathcal{K}oS$ ... graphically



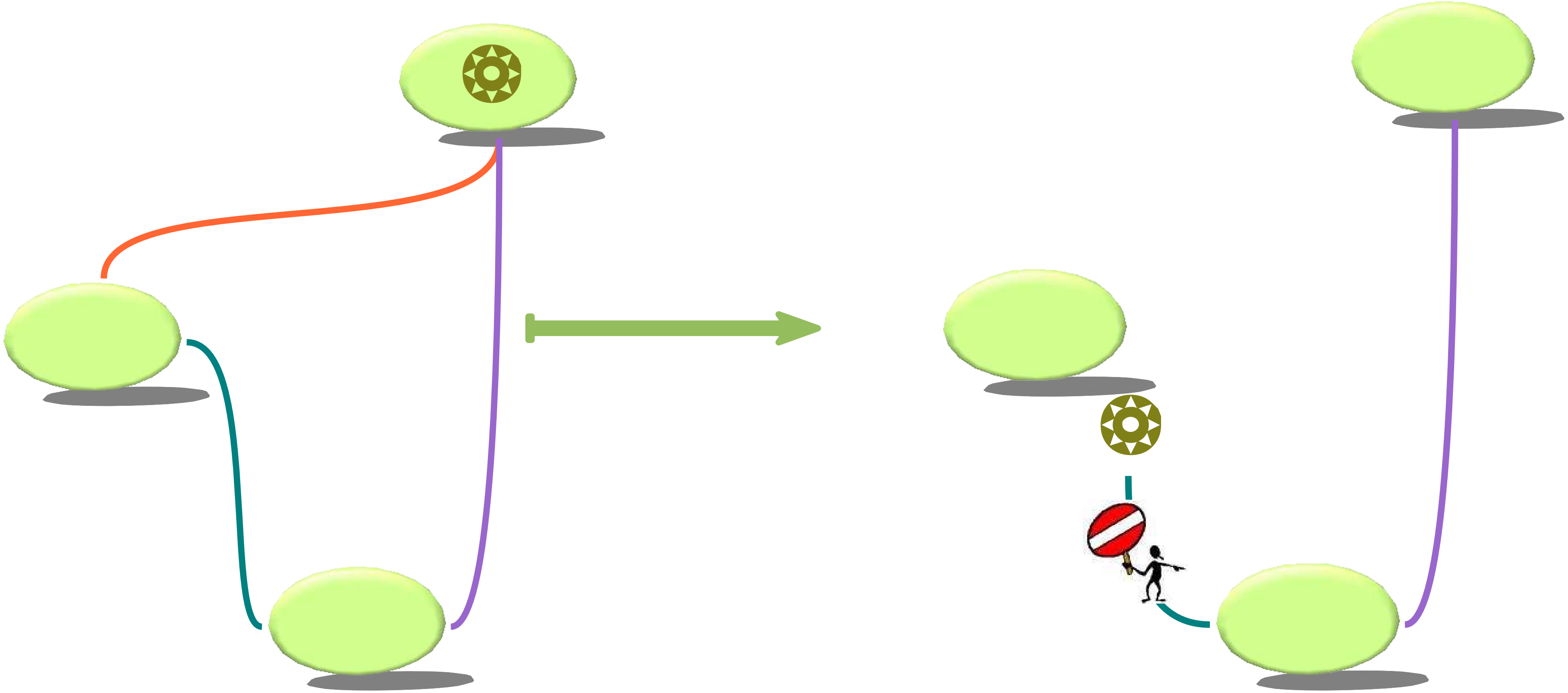
$\mathcal{K}o\mathcal{S}$ … graphically



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



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

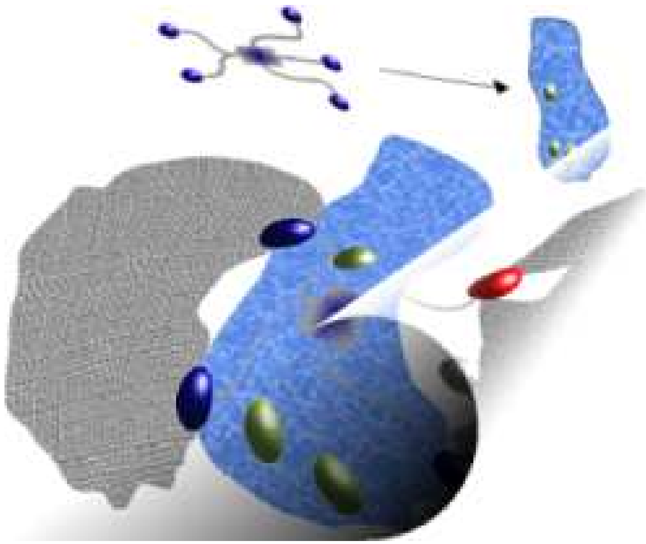


SHR & SOC

Joint works with

G. Ferrari, D. Hirsch, I. Lanese, A.

LLuch-lafuente, U. Montanari



Using SHR, we aim at

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

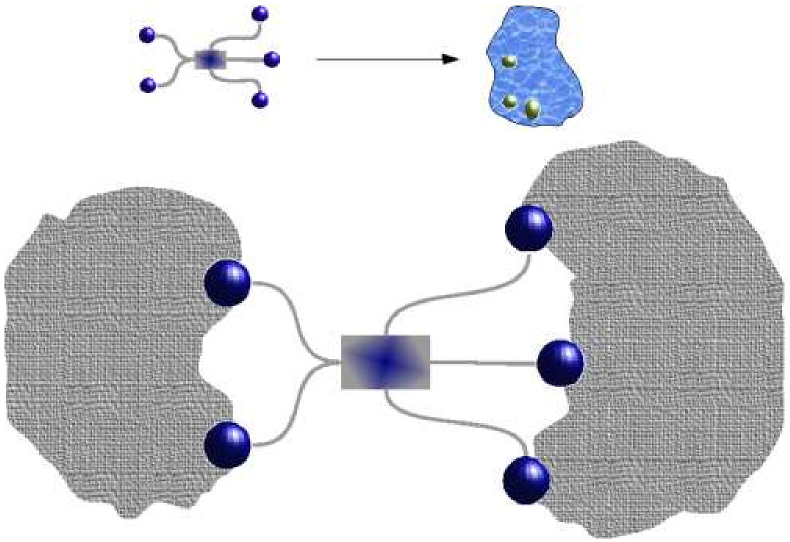
The metaphor is

- "SOC systems *as* Hypergraphs"
- "SOC computations *as* SHR"

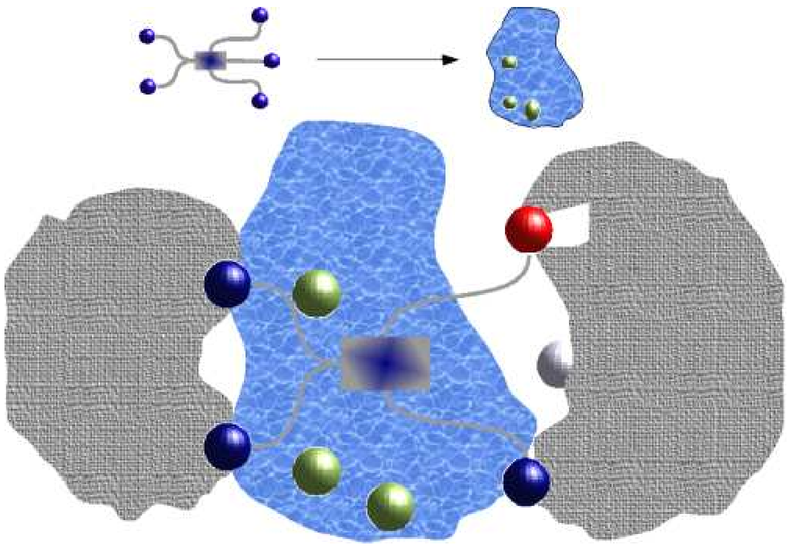
In other words:

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

Synchronised Replacement of Hyperedges

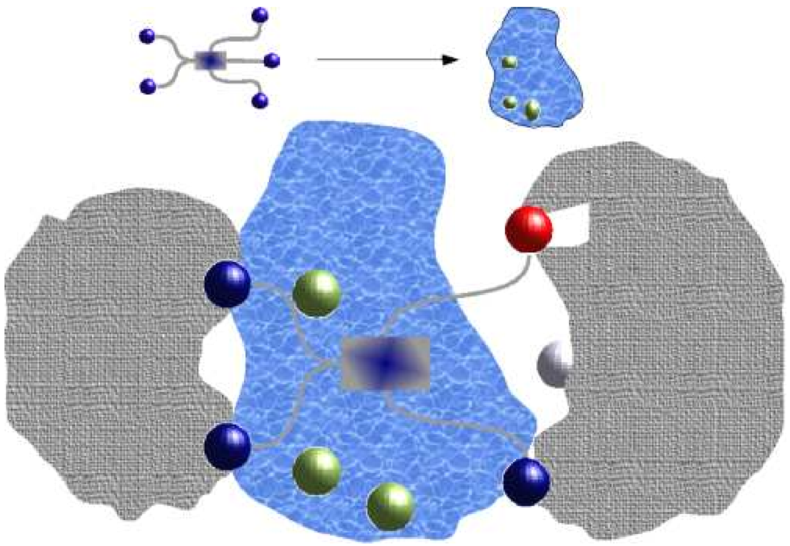


Synchronised Replacement of Hyperedges



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

Synchronised Replacement of Hyperedges

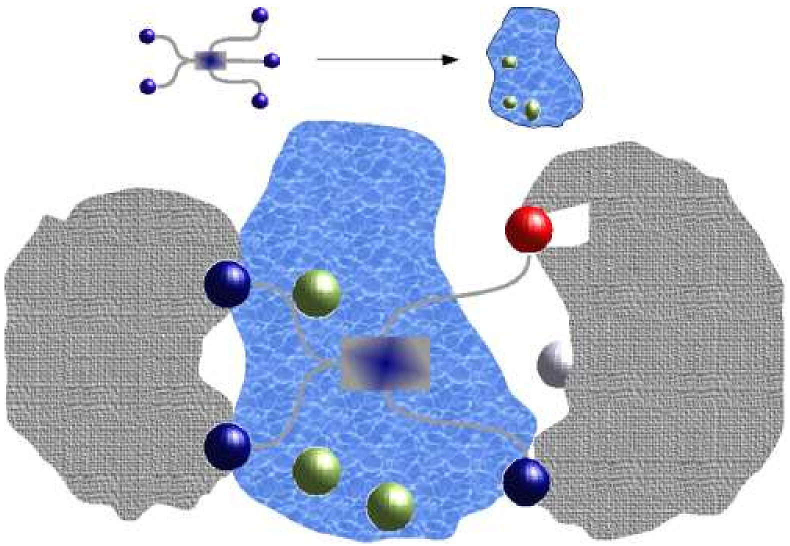


- Edge replacement: local
- Synchronisation as distributed constraint solving
- Multi-party synchronisation
- New node creation
- Node fusion: model of mobility and communication

Benefits:

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

Synchronised Replacement of Hyperedges



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

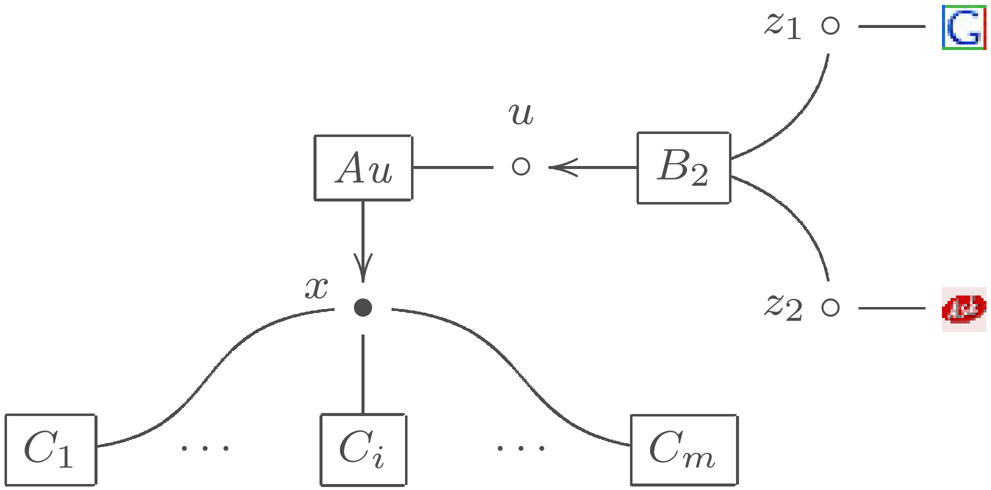
Benefits:

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

SHR can combine
QoS & sophisticated synchronisations
(details in [HT05, LT05, HLT])

SHR & synchronisation algebras via an example

- Clients $C_1, \ldots, C_m$ invoke a service from a remote servers $S_1, \ldots, S_n$ provided that they are authorised
- A trusted authority Au checks for the authorisation

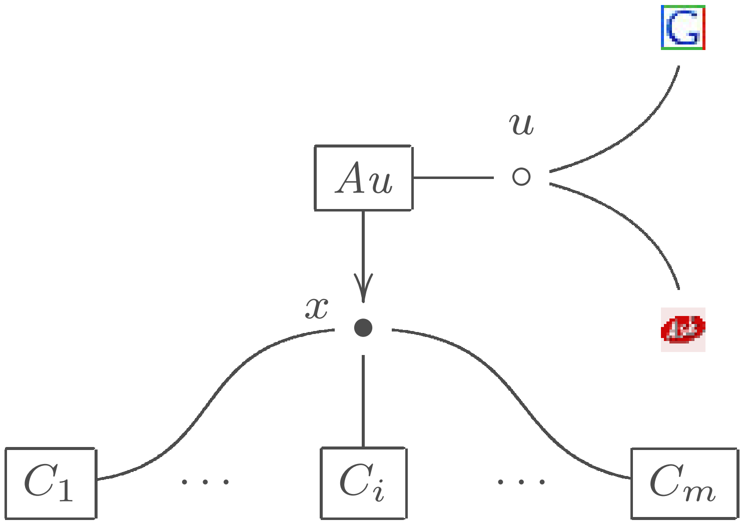


Clients are connected to Au on a "public" node x while servers are connected on a "private" (i.e., restricted) ones.
B_2 will simply acquire the requests from clients and forward them to each server.

Notice that

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

- Clients $C_1, \ldots, C_m$ invoke a service from a remote servers $S_1, \ldots, S_n$ 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) ones.

In SHR we can simply specify

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

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

where Bdc is the broadcast SAM.

A spatio-temporal logic for SHR interpreted over c-semirings has been defined in [HLT].

$$
\begin{array}{rcl}
\phi & ::= & nil \mid \Gamma(\xi) \mid \mathfrak{L}(\tilde{\xi}) \mid \phi|\phi \mid \phi\|\phi \\
& | & f(\phi, \ldots, \phi) \\
& | & \sum_u \phi \mid \prod_u \phi \\
& | & [\sum]\,\phi \mid [\prod]\,\phi \\
& | & u = v \\
& | & \mathfrak{r}(\tilde{\xi}) \mid (\mu\mathfrak{r}(\tilde{u}).\phi)\tilde{\xi} \mid (\nu\mathfrak{r}(\tilde{u}).\phi)\tilde{\xi}
\end{array}
$$

where $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ is a c-semiring, $\mathfrak{L} \subseteq \mathcal{L}$ 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 spatio-temporal logic for SHR interpreted over c-semirings has been defined in [HLT].

$$
\begin{array}{rcl}
\phi & ::= & nil \mid \Gamma(\xi) \mid \mathfrak{L}(\tilde{\xi}) \mid \phi|\phi \mid \phi\|\phi \\
& \mid & f(\phi, \ldots, \phi) \\
& \mid & \sum_u \phi \mid \prod_u \phi \\
& \mid & [\sum] \phi \mid [\prod] \phi \\
& \mid & u = v \\
& \mid & \mathfrak{r}(\tilde{\xi}) \mid (\mu \mathfrak{r}(\tilde{u}).\phi)\tilde{\xi} \mid (\nu \mathfrak{r}(\tilde{u}).\phi)\tilde{\xi}
\end{array}
$$

where $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ is a c-semiring, $\mathfrak{L} \subseteq \mathcal{L}$ 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 spatio-temporal logic for SHR interpreted over c-semirings has been defined in [HLT].

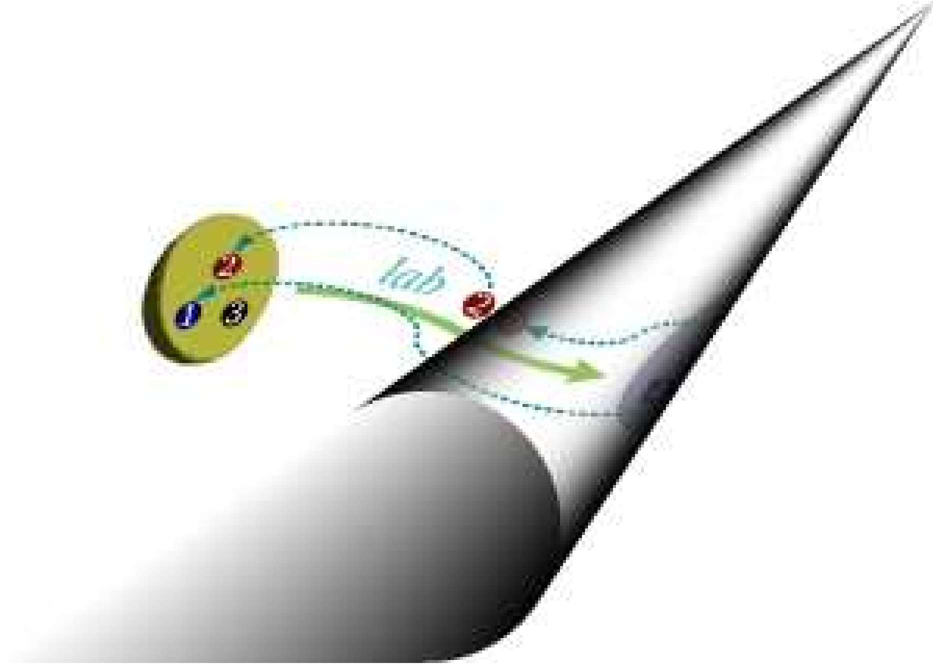
$$
\begin{array}{rcl}
\phi & ::= & nil \mid \Gamma(\xi) \mid \mathfrak{L}(\tilde{\xi}) \mid \phi|\phi \mid \phi\|\phi \\
& \mid & f(\phi,\ldots,\phi) \\
& \mid & \sum_u \phi \mid \prod_u \phi \\
& \mid & [\sum]\,\phi \mid [\prod]\,\phi \\
& \mid & u = v \\
& \mid & \mathfrak{r}(\tilde{\xi}) \mid (\mu\mathfrak{r}(\tilde{u}).\phi)\tilde{\xi} \mid (\nu\mathfrak{r}(\tilde{u}).\phi)\tilde{\xi}
\end{array}
$$

where $\langle S, +, \star, \mathbf{0}, \mathbf{1}\rangle$ is a c-semiring, $\mathfrak{L} \subseteq \mathcal{L}$ 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

Let $\mathcal{G}$ be the set of weighted graphs: 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 $\mathcal{G} \to S$.

$$
\begin{array}{rcl}
[\![\, \phi_1|\phi_2 \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \sum_{(G_1,G_2)\in\Theta(G)}\{[\![\, \phi_1 \,]\!]_{\sigma;\rho}(\Gamma \vdash G_1) \star [\![\, \phi_2 \,]\!]_{\sigma;\rho}(\Gamma \vdash G_2)\} \\
[\![\, \phi_1\|\phi_2 \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \prod_{(G_1,G_2)\in\Theta(G)}\{[\![\, \phi_1 \,]\!]_{\sigma;\rho}(\Gamma \vdash G_1) + [\![\, \phi_2 \,]\!]_{\sigma;\rho}(\Gamma \vdash G_2)\} \\
[\![\, \sum_u \phi \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \sum_{x\in \mathrm{n}(G)} [\![\, \phi \,]\!]_{\sigma[^x/_u];\rho}(\Gamma \vdash G) \\
[\![\, \prod_u \phi \,]\!]_{\sigma;\rho}(\Gamma \vdash G) & = & \prod_{x\in \mathrm{n}(G)} [\![\, \phi \,]\!]_{\sigma[^x/_u];\rho}(\Gamma \vdash G) \\
[\![\, [\sum]\,\phi \,]\!](\Gamma \vdash G) & = & \sum_{\Gamma\vdash G \xrightarrow{\Lambda} \Gamma'\vdash G'} [\![\, \phi \,]\!](\Gamma' \vdash G') \\
[\![\, [\prod]\,\phi \,]\!](\Gamma \vdash G) & = & \prod_{\Gamma\vdash G \xrightarrow{\Lambda} \Gamma'\vdash G'} [\![\, \phi \,]\!](\Gamma' \vdash G')
\end{array}
$$

HD-automata



Joint works with

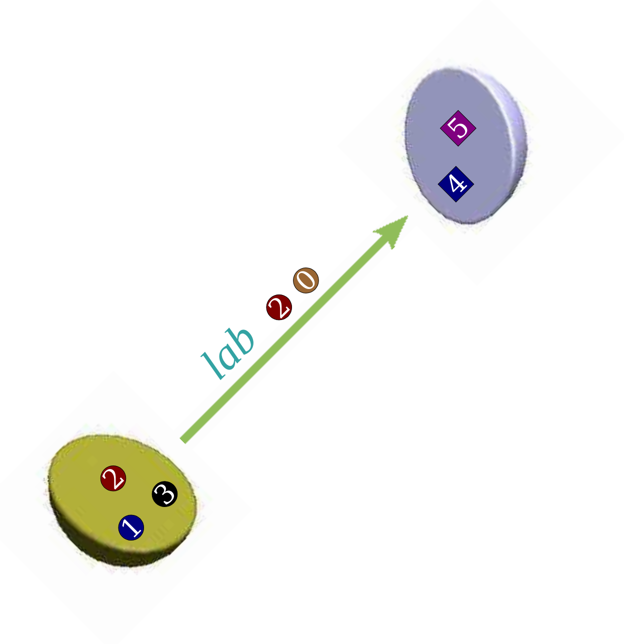
G. Ferrari, U. Montanari, K. Yemane and B. Victor

- HD-automata as an operational model of history-dependent calculi [Pis99, MP98]
- allow a finite representation of classes of infinite LTS

- HD-automata as an operational model of history-dependent calculi [Pis99, MP98]
- allow a finite representation of classes of infinite LTS

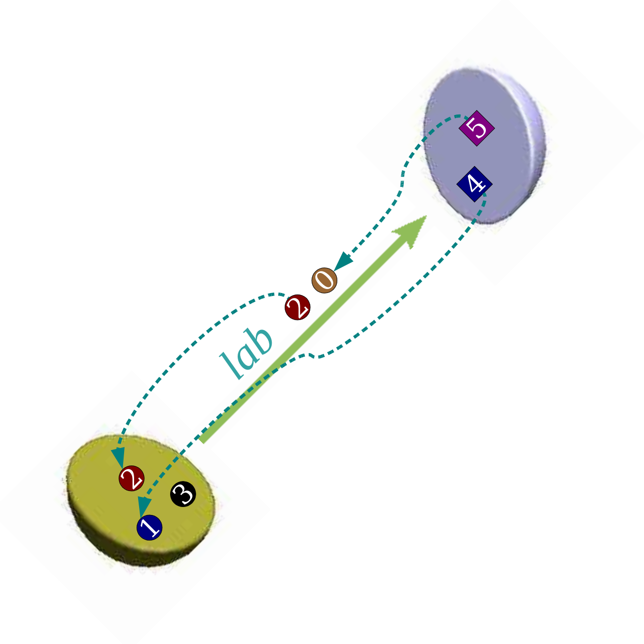
A HD-automaton associates a "history" to names of the states appearing in the computation: it is possible to reconstruct the associations that lead to the state containing the name. If a state is reached in two different computations, different histories could be assigned to its names.

- HD-automata as an operational model of history-dependent calculi [Pis99, MP98]
- allow a finite representation of classes of infinite LTS



- states and transitions have local names:
  - names explicit in the operational model
  - so that HD-automata model name creation/deallocation or extrusion
- State s has three names: 1, 2 and 3
- State d has two names: 4 and 5
- The transition is labelled by lab and exposes names 2 (of s) and a fresh name 0

- HD-automata as an operational model of history-dependent calculi [Pis99, MP98]
- allow a finite representation of classes of infinite LTS

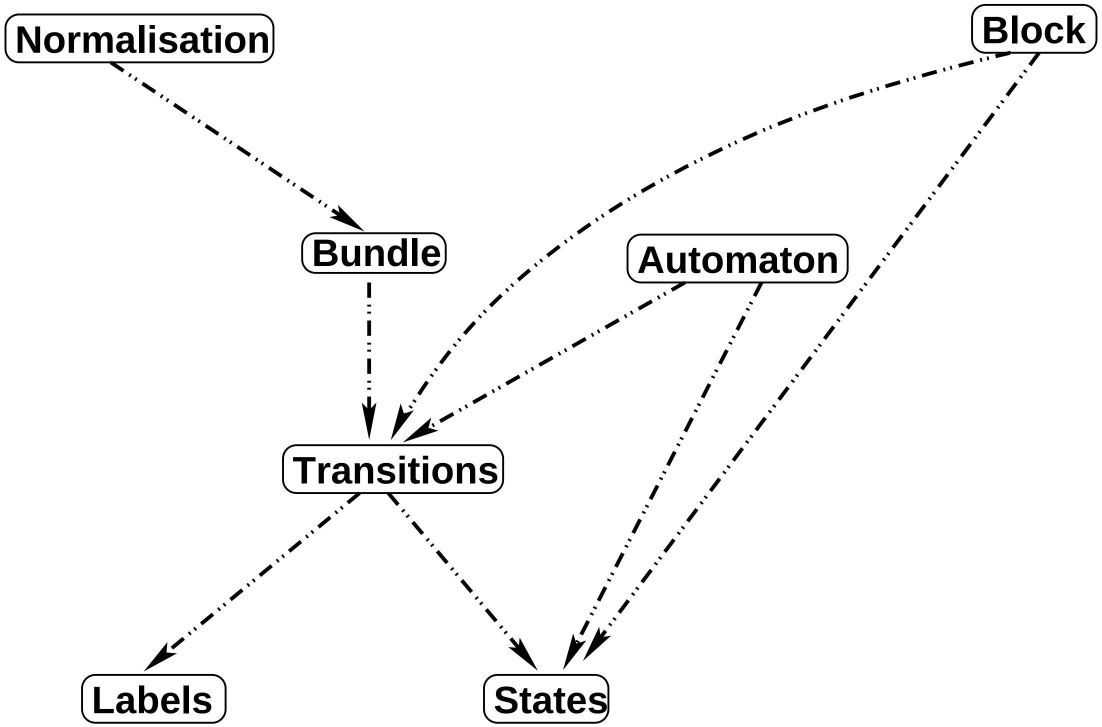


- states and transitions have local names:
  - names explicit in the operational model
  - so that HD-automata model name creation/deallocation or extrusion
- State s has three names: 1, 2 and 3
- State d has two names: 4 and 5
- The transition is labelled by lab and exposes names 2 (of s) and a fresh name 0
- $\sigma : 4 \mapsto 1$ and $\sigma : 5 \mapsto 0$, the new name
- 3 is "discharged"

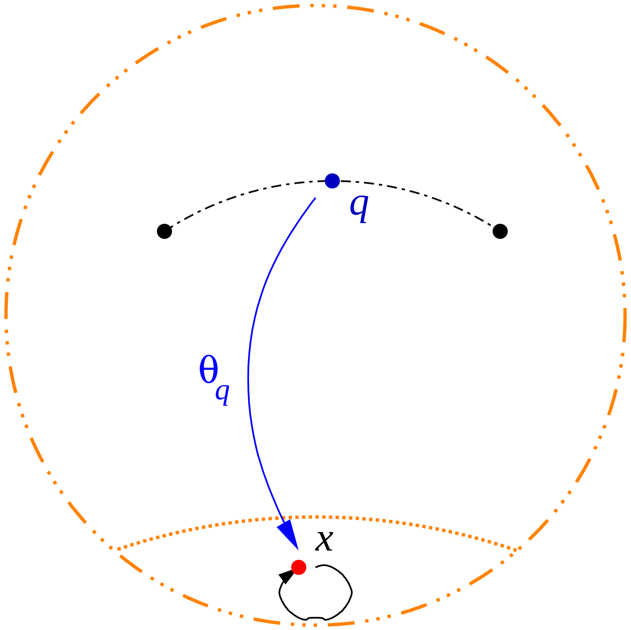
Minimising History Dependent Automata:

- Co-algebraic specification
- Partition Refinement Algorithm based on co-algebraic specification [FMP02]
- Mihda: Ocaml implementation (refining $\lambda^{\rightarrow,\Pi,\Sigma}$ spec. [FMT05a])

	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

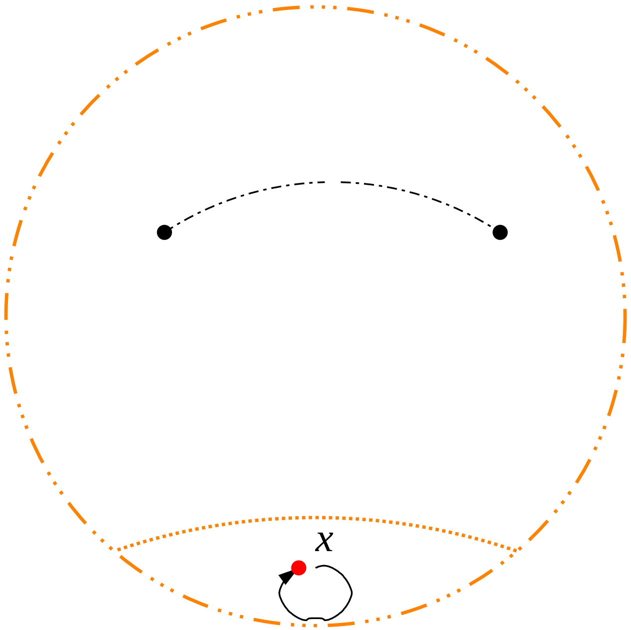


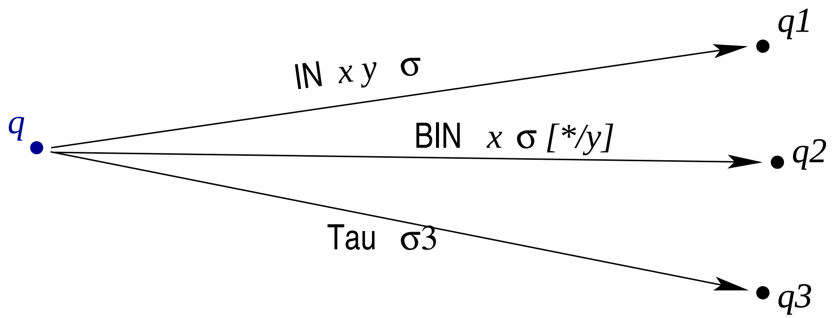
- Adherent to specs
- Highly modular
- Easily extendible



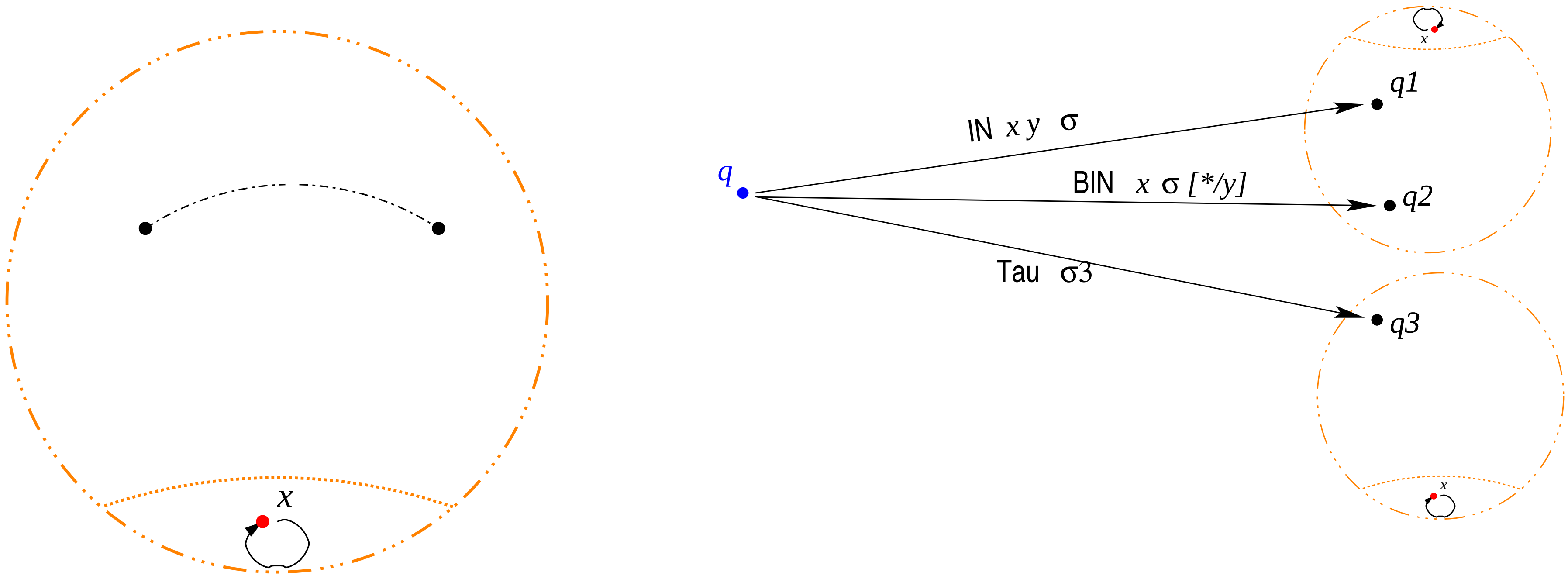
q
θ_q
x

The main step

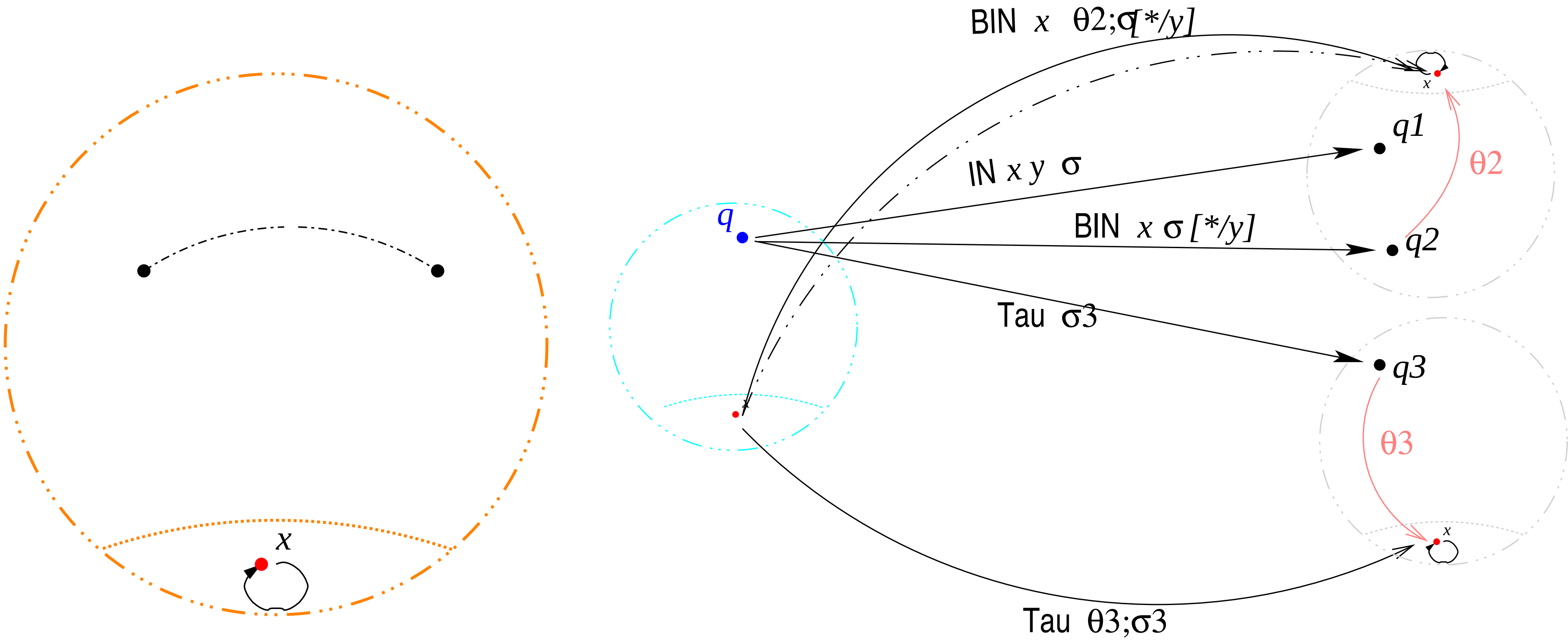




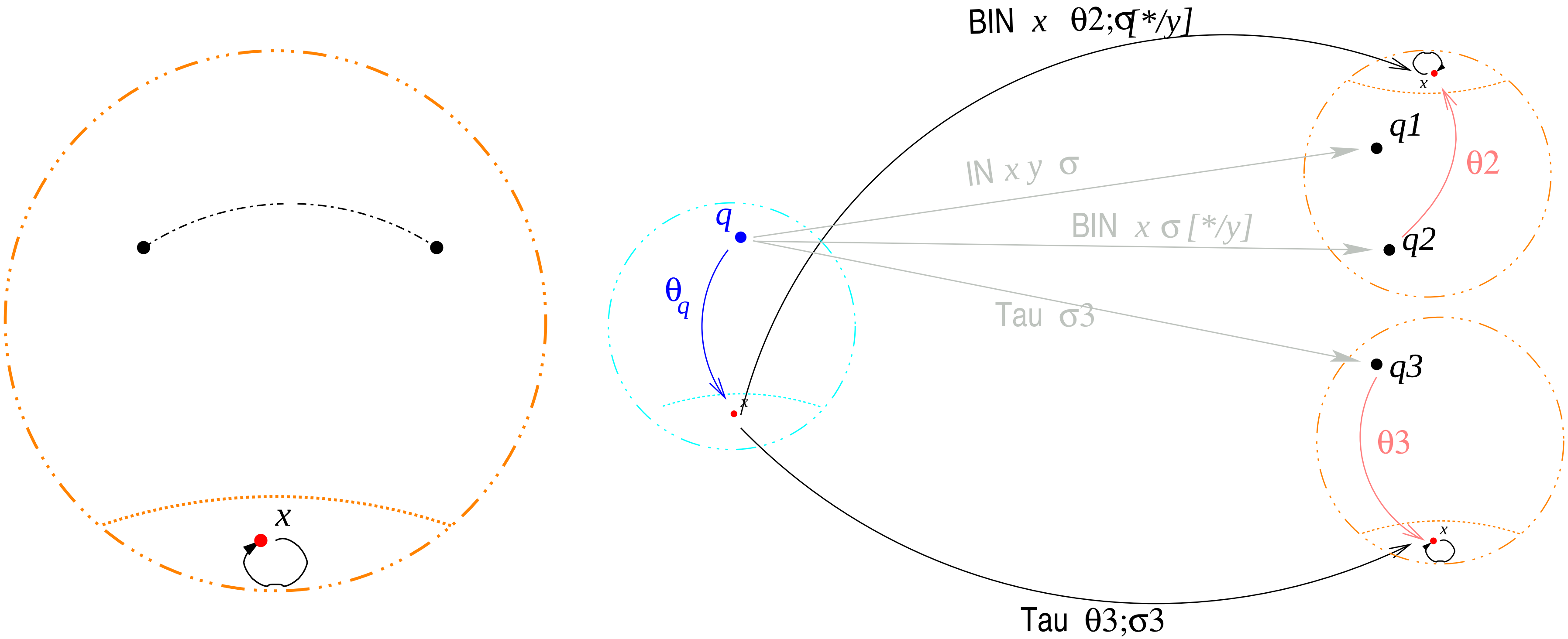
The main step



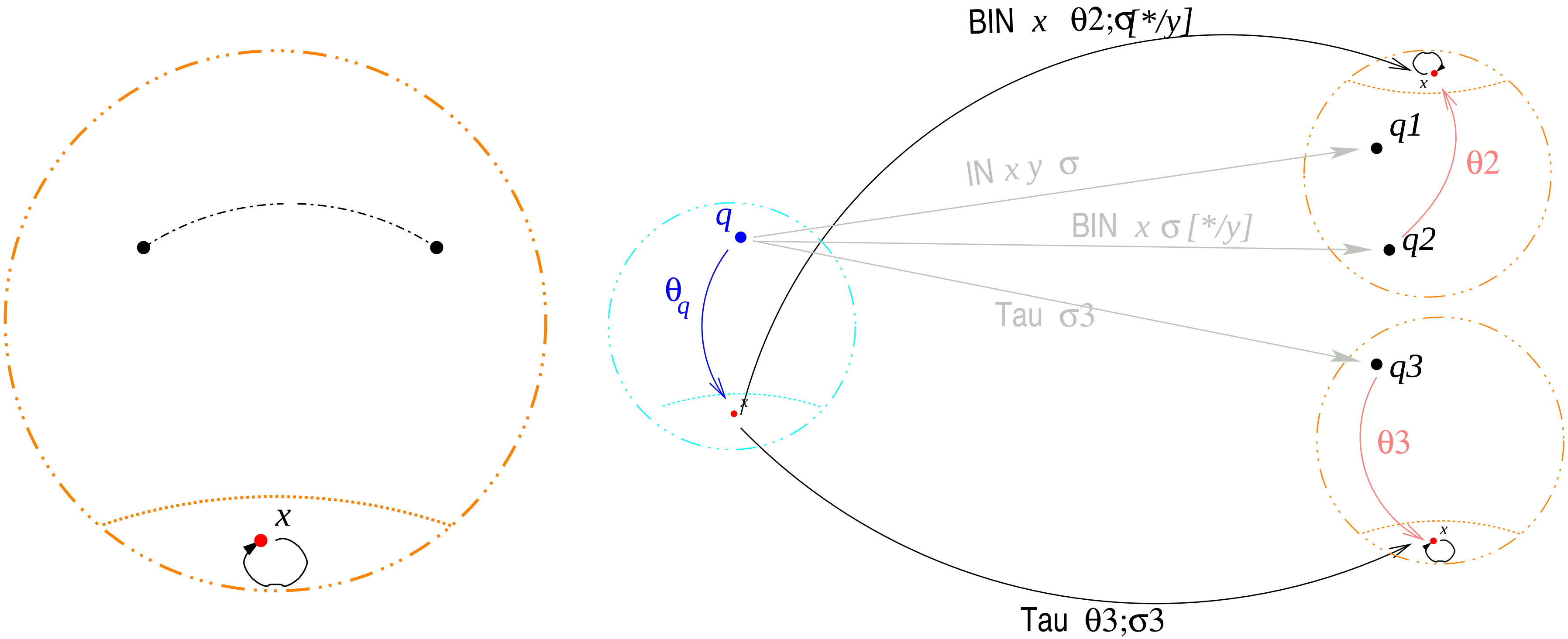
The main step



The main step



The main step



Theorem Any block at a generic iteration i collects those states that cannot be distinguished in i-steps.

Theorem The algorithm converges on finite HD-automata (for π-calculus [FMT05a])

Theorem The iterative partition refinement algorithm is convergent on finite HD-automata whenever the normalisation functor is monotone on nfs (for π-calculus [FMT$^+$05b])

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.

[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 *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.

[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.

[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.

[FMT^{+}05b] 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.

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

[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.

[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.

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

Application level QoS abstracted as constraint-semiring [BMR95, BMR97]

- for coordinating mobility
- and synchronisations

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

- $\mathbf{0}, \mathbf{1} \in S$ and $\mathbf{0} \neq \mathbf{1}$
- and

$+ : S \times S \to S$	$\star : S \times S \to S$
$x + y = y + x$	$x \star y = y \star x$
$(x + y) + z = x + (y + z)$	$(x \star y) \star z = x \star (y \star z)$
$x + \mathbf{0} = x$	$x \star \mathbf{1} = x$
$x + \mathbf{1} = \mathbf{1}$	$x \star \mathbf{0} = \mathbf{0}$
$x + x = x$	$(x + y) \star z = (x \star z) + (y \star z)$

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

The cartesian product of c-semirings is a c-semiring

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

$$
\begin{array}{rcl}
\mathbf{N}, \mathbf{M} & ::= & \\
& & \mathbf{0} \\
& \mid & s :: \mathbf{P} \\
& \mid & (\nu\, s)\mathbf{N} \\
& \mid & \mathbf{N} \parallel \mathbf{M} \\
& \mid & s \overset{\kappa}{\frown} t
\end{array}
\qquad
\begin{array}{rcl}
\mathbf{P}, \mathbf{Q} & ::= & \\
& & \mathbf{nil} \\
& \mid & \gamma.\mathbf{P} \\
& \mid & (\nu\, s)\mathbf{P} \\
& \mid & \mathbf{P} \mid \mathbf{Q} \\
& \mid & !\mathbf{P} \\
& \mid & \cdots
\end{array}
$$

$$
\begin{array}{rcl}
\gamma & ::= & \\
& & (T) \\
& \mid & \langle v_1, \ldots, v_n \rangle \\
& \mid & \varepsilon_{\kappa}[\mathbf{P}]@t \\
& \mid & con_{\kappa}\langle t \rangle \\
& \mid & acc_{\kappa}\langle t \rangle \\
& \mid & node_{\kappa}\langle t \rangle
\end{array}
\qquad
\begin{array}{rcl}
T & ::= & \\
& & \varepsilon \\
& \mid & v \\
& \mid & ?x \\
& \mid & \neg v \\
& \mid & T, T
\end{array}
$$

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

$$\mathbf{N}, \mathbf{M} ::= \mathbf{0} \mid s :: \mathbf{P} \mid (\nu\, s)\mathbf{N} \mid \mathbf{N} \parallel \mathbf{M} \mid s \overset{\kappa}{\frown} t$$

$$\mathbf{P}, \mathbf{Q} ::= \mathbf{nil} \mid \gamma.\mathbf{P} \mid (\nu\, s)\mathbf{P} \mid \mathbf{P} \mid \mathbf{Q} \mid\ !\mathbf{P} \mid \cdots$$

$$\gamma ::= (T) \mid \langle v_1, \ldots, v_n \rangle \mid \varepsilon_{\kappa}[\mathbf{P}]@t \mid con_{\kappa}\langle t \rangle \mid acc_{\kappa}\langle t \rangle \mid node_{\kappa}\langle t \rangle$$

$$T ::= \varepsilon \mid v \mid ?x \mid \neg v \mid T, T$$

$\mathbf{N} \xrightarrow[\kappa]{\alpha} \mathbf{M}$ states that $\mathbf{N}$ performs α with a cost κ and becomes $\mathbf{M}$

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

$$\mathbf{N}, \mathbf{M} \quad ::= \quad \mathbf{0} \;\mid\; s :: \mathbf{P} \;\mid\; (\nu\, s)\mathbf{N} \;\mid\; \mathbf{N} \parallel \mathbf{M} \;\mid\; s \overset{\kappa}{\frown} t$$

$$\mathbf{P}, \mathbf{Q} \quad ::= \quad \mathbf{nil} \;\mid\; \gamma.\mathbf{P} \;\mid\; (\nu\, s)\mathbf{P} \;\mid\; \mathbf{P} \mid \mathbf{Q} \;\mid\; !\mathbf{P} \;\mid\; \ldots$$

$$\gamma \quad ::= \quad (T) \;\mid\; \langle v_1, \ldots, v_n \rangle \;\mid\; \varepsilon_{\kappa}[\mathbf{P}]@t \;\mid\; con_{\kappa}\langle t \rangle \;\mid\; acc_{\kappa}\langle t \rangle \;\mid\; node_{\kappa}\langle t \rangle$$

$$T \quad ::= \quad \varepsilon \;\mid\; v \;\mid\; ?x \;\mid\; \neg v \;\mid\; T, T$$

$\mathbf{N} \xrightarrow[\kappa]{\alpha} \mathbf{M}$ states that $\mathbf{N}$ performs α with a cost κ and becomes $\mathbf{M}$

$$(\textsc{Route})\quad \frac{\mathbf{N} \xrightarrow[\kappa']{\mathbf{r}\ \varepsilon^{\mathbf{s}}_{\kappa}\langle\mathbf{P}\rangle@\mathbf{t}} \mathbf{N}' \qquad \mathbf{M} \xrightarrow[\kappa'']{\mathbf{r}\ \mathbf{link}\ \mathbf{r}'} \mathbf{M}' \qquad \kappa' \star \kappa'' \leq \kappa}{\mathbf{N} \parallel \mathbf{M} \xrightarrow[\kappa' \star \kappa'']{\mathbf{r}'\ \varepsilon^{\mathbf{s}}_{\kappa}\langle\mathbf{P}\rangle@\mathbf{t}} \mathbf{N}' \parallel \mathbf{M}'}\ \ t \neq r'$$

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

$$\mathbf{N},\mathbf{M} ::= \mathbf{0} \mid s :: \mathbf{P} \mid (\nu\, s)\mathbf{N} \mid \mathbf{N} \parallel \mathbf{M} \mid s \overset{\kappa}{\frown} t$$

$$\mathbf{P},\mathbf{Q} ::= \mathbf{nil} \mid \gamma.\mathbf{P} \mid (\nu\, s)\mathbf{P} \mid \mathbf{P} \mid \mathbf{Q} \mid !\mathbf{P} \mid \dots$$

$$\gamma ::= (T) \mid \langle v_1, \dots, v_n \rangle \mid \varepsilon_{\kappa}[\mathbf{P}]@t \mid con_\kappa\langle t\rangle \mid acc_\kappa\langle t\rangle \mid node_\kappa\langle t\rangle$$

$$T ::= \varepsilon \mid v \mid ?x \mid \neg v \mid T, T$$

$\mathbf{N} \xrightarrow[\kappa]{\alpha} \mathbf{M}$ states that $\mathbf{N}$ performs α with a cost κ and becomes $\mathbf{M}$

$$(\textsc{land})\quad \frac{\mathbf{N} \xrightarrow[\kappa']{\mathbf{r}\ \varepsilon^{\mathbf{s}}_{\kappa}\langle \mathbf{P}\rangle @\mathbf{t}} \mathbf{N}' \qquad \mathbf{M} \xrightarrow[\kappa'']{\mathbf{r\ link\ t}} \mathbf{M}' \qquad \kappa' \star \kappa'' \le \kappa}{\mathbf{N} \parallel \mathbf{M} \xrightarrow[\kappa' \star \kappa'']{\tau} \mathbf{N}' \parallel \mathbf{M}' \parallel \mathbf{t} :: \mathbf{P}}$$

Consider a scenario where n servers provide services to m clients and focus on balancing the load of the servers.

- clients (c_i) and servers (s_j) are located on different nodes
- c_i issues requests to s_j by spawning a process R

A generic client is described by the following term:

$$c_i :: \langle s_1, \kappa_1 \rangle \quad | \quad \ldots \quad | \quad \langle s_n, \kappa_n \rangle \quad | \; !C_\delta$$

- $\langle s_j, \kappa_j \rangle$ represents the load κ_j of the server s_j perceived b c_i
- C_δ and R specify the behaviour of c_i:

$$\begin{aligned} C_\delta &\triangleq (?u, ?v).\varepsilon_v[R]@u.con_{v \star \delta}\langle u \rangle.\langle u, v \star \delta \rangle \\ R &\triangleq (?x).\langle x + 1 \rangle \ldots \textit{actual request} \ldots (?y).\langle y - 1 \rangle \end{aligned}$$

Remark 1 *Remote spawning consumes the traversed links, hence c_i attempts to re-establish a connection with the server!*

s_j is described as:

$$s_j :: \langle h \rangle \quad | \quad \langle c_1, \kappa'_1 \rangle \quad | \quad \ldots \quad | \quad \langle c_m, \kappa'_m \rangle \quad | \; !(S\ c_1\ s_j) \quad | \quad \ldots \quad | \; !(S\ c_m\ s_j)$$

- $\langle c_i, \kappa'_i \rangle$ records the QoS value κ'_i assigned to the link towards c_i
- $\langle h \rangle$ is the current load of s_j
- $S\ c_i\ s_j$ is a load manager for c_i

$$\begin{array}{lcl} S\ c\ s & \triangleq & (?l).\langle l \rangle.\mathit{If}_s\ l < max \\ & & \quad \mathit{then}\ (c, ?v).acc_{f(v,l)}\langle c \rangle.\langle c, f(v,l) \rangle \end{array}$$

S repeatedly acquires $\langle h \rangle$ and depending on the load decides whether to accept requests for new connections coming from c.

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

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

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

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

$$(\textsc{node}) \qquad \mathbf{s} :: \mathbf{node}_\kappa\langle \mathbf{t}\rangle.\mathbf{P} \xrightarrow[\mathbf{1}]{\mathbf{node}\langle \mathbf{t}\rangle} \mathbf{s} :: \mathbf{P} \parallel \mathbf{s} \overset{\kappa}{\frown} \mathbf{t} \parallel \mathbf{t} :: \mathbf{0}, \ \boxed{s \neq t}$$

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

Rule $(\textsc{node})$ 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 $(\textsc{par})$ prevents new nodes (and links) to be created by using addresses of existing nodes.

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

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

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

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.

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

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

- 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 first created (and t agrees on that).

Initially, only $(\textsc{con})$ con be applied:

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

$r \overset{2}{\frown} t$ provides now a path (costing 3) from s to t, hence using $(\textsc{pref})$, $(\textsc{link})$, $(\textsc{route})$ and $(\textsc{land})$ we derive

$$\mathbf{N'} \xrightarrow[\mathbf{3}]{\tau} \mathbf{s :: nil} \;\parallel\; \mathbf{r} :: \varepsilon_\mathbf{2}[\mathbf{Q}]@\mathbf{t} \;\parallel\; \mathbf{t :: P}.$$

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

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

$$N \stackrel{\triangle}{=} s :: \varepsilon_{\{r,s\}}[P]@t \parallel s \stackrel{\{r\}}{\frown} s' \qquad M \stackrel{\triangle}{=} s :: \varepsilon_{\{r,s\}}[P]@t \parallel s \stackrel{\{r,u\}}{\frown} s'$$

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

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

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

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

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

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

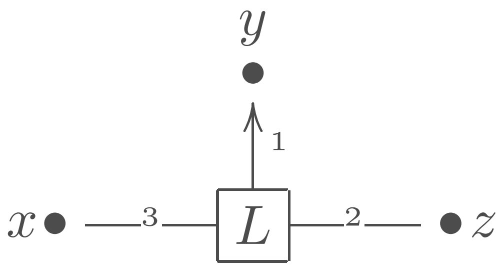
$$s :: !con_{\kappa}\langle t\rangle \parallel t :: !acc_{\kappa'}\langle s\rangle$$

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

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

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

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

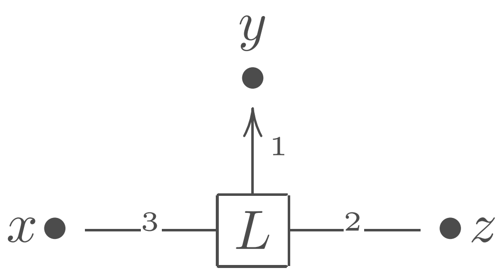


$$G ::= nil \ \big|\ L(\tilde{x}) \ \big|\ G|G \ \big|\ \nu\, y.G$$

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

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

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



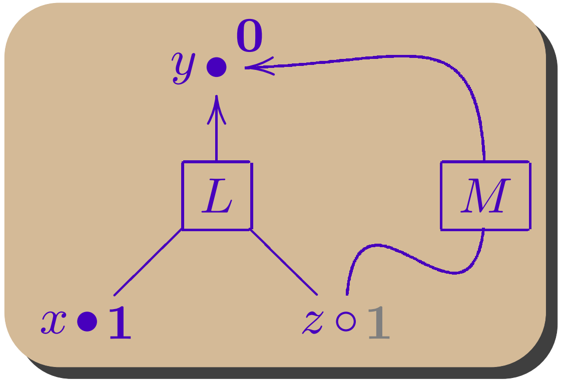
$G ::= nil \mid L(\tilde{x}) \mid G|G \mid \nu\, y.G$

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 based on requirements: $\mathcal{R} = S \times \mathcal{N}^*$ (where $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ is a 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
- G is a graph s.t. $\mathrm{fn}(G) \subseteq \{\!|\tilde{x}|\!\} \cup \mathrm{n}(\Lambda)$

Productions based on requirements: $\mathcal{R} = S \times \mathcal{N}^*$ (where $\langle S, +, \star, \mathbf{0}, \mathbf{1} \rangle$ is a 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
- G is a graph s.t. $\mathrm{fn}(G) \subseteq \{\!|\tilde{x}|\!\} \cup \mathrm{n}(\Lambda)$

Replacing L with G in H requires that H satisfies the conditions expressed by χ on the attachment nodes of L

$\Longrightarrow$

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 $\tilde{x}$

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.

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

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.

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

mgu accounting for node fusions; let Ω be a finite multiset over $\mathcal{N} \times \mathcal{R}$.

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

is an idempotent substitution defined iff

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

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.

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

mgu accounting for node fusions; let Ω be a finite multiset over $\mathcal{N} \times \mathcal{R}$.

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

is an idempotent substitution defined iff

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

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

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

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

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

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

Let $\rho = \mathbf{mgu}(\Omega)$. The communication function induced by Ω 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.

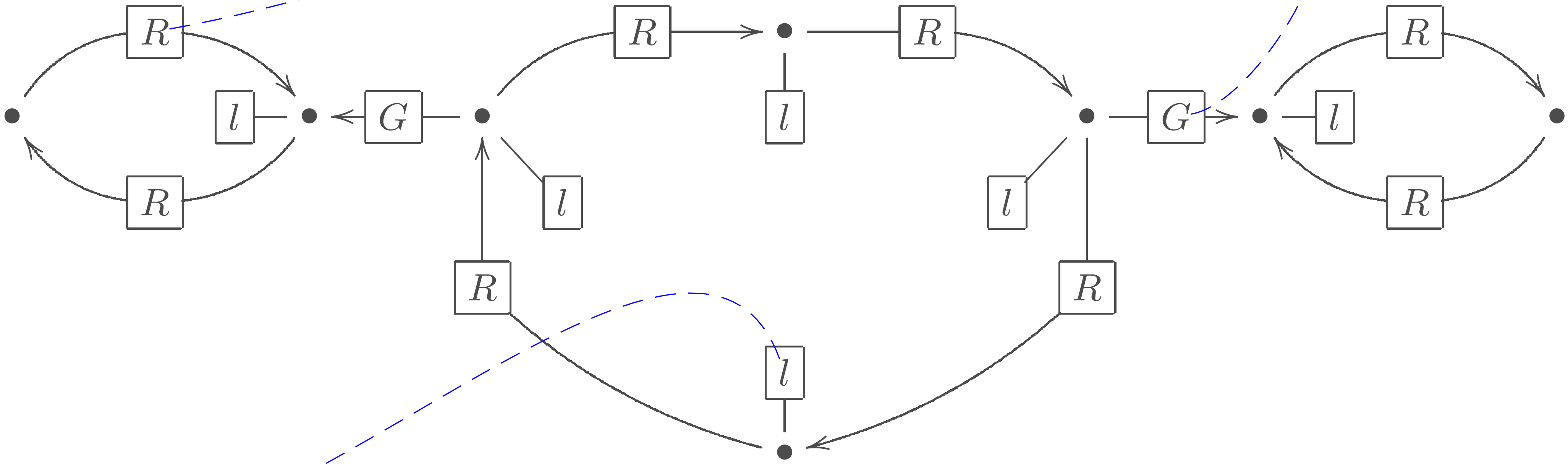
The weighting function induced by Γ and Ω is

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

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

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

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



l-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 l-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

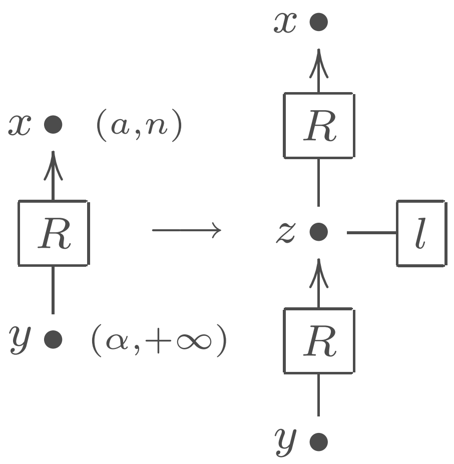
The c-semiring for networks of rings is $\mathfrak{H}\mathfrak{R}$ given by the cartesian product of the Hoare synchronisations c-semiring $\mathfrak{H} = \langle \mathcal{H}, +_H, \star_H, \mathbf{0}_H, \mathbf{1}_H \rangle$, where

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

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

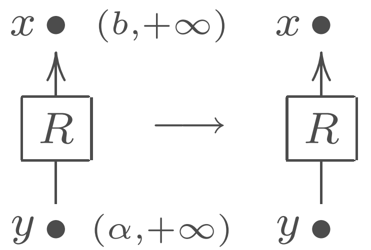
- $\mathfrak{H}$ coordinates the network rewritings
- $\mathfrak{R}$ handles resource availability
- the initial graph is a ring
- non-limited nodes have weights $(\mathbf{1}_H, u)$ where value u is the maximal amount of available resource
- the limited nodes created during ring evolution are weighted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

Create Brother $(n < u)$



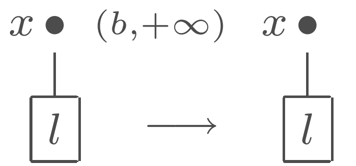
$$x : (\mathbf{0}_H, u), y : \mathbf{0} \triangleright R(x,y) \xrightarrow[y \mapsto (\alpha, +\infty)]{x \mapsto (a, n)} R(x,z) \mid R(z,y) \mid l(z)$$

Accept Syncrhonisation R



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

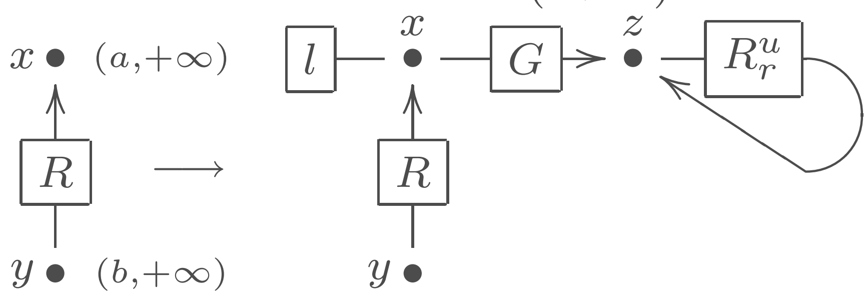
Accept Syncrhonisation l



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

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

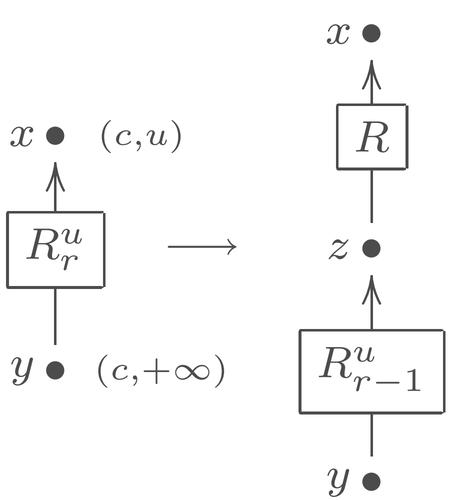
Create Gate $(r > 0)$



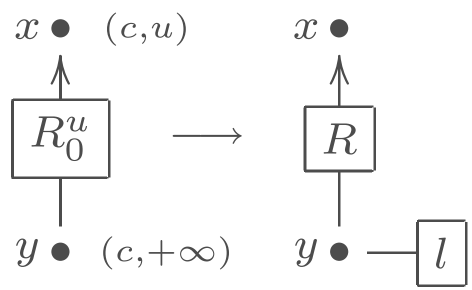
$$x : (\mathbf{0}_H, u), y : \mathbf{0} \triangleright R(x,y) \xrightarrow{\substack{x \mapsto (a,\, +\infty) \\ y \mapsto (b,\, +\infty)}} R(x,y) \mid l(x) \mid G(z,x) \mid R^u_r(z,z)$$

Init Ring

$(r > 0)$



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

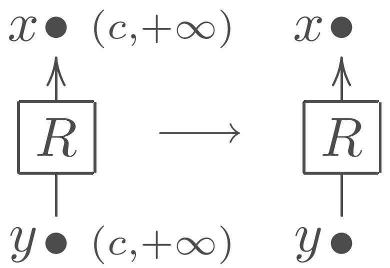


$$x : \mathbf{0}, y : \mathbf{0} \triangleright R^u_0(x,y) \xrightarrow{\substack{x \mapsto (c,\, u) \\ y \mapsto (c,\, +\infty)}} R(x,y) \mid l(y)$$

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

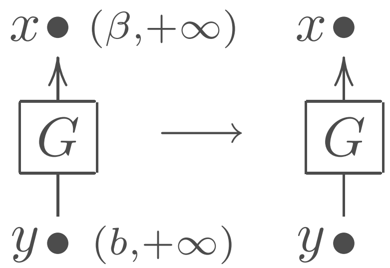
Productions for the ring case study[3]

Accept Synchronisation Init



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

Accept Synchronisation Gate

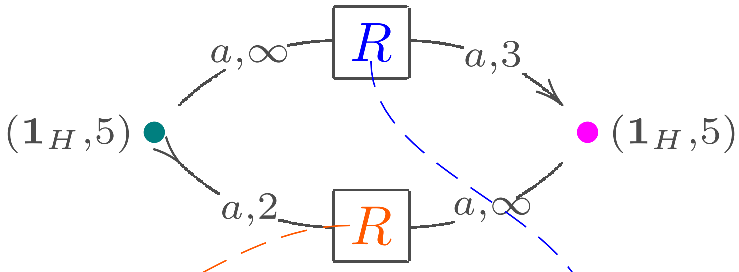


$$x : \mathbf{0}, y : \mathbf{0} \triangleright G(x,y) \xrightarrow{\substack{x \;\mapsto\; (\beta,\,+\infty) \\ y \;\mapsto\; (b,\,+\infty)}} G(x,y)$$

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

The ring case study

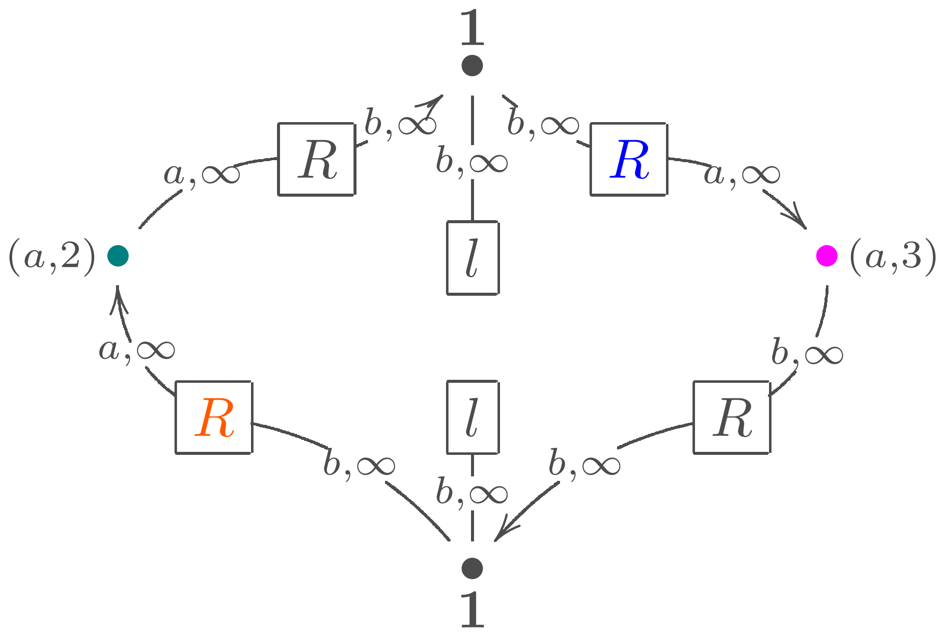
The derivation starts from a 2-ring components with resource value 5.



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

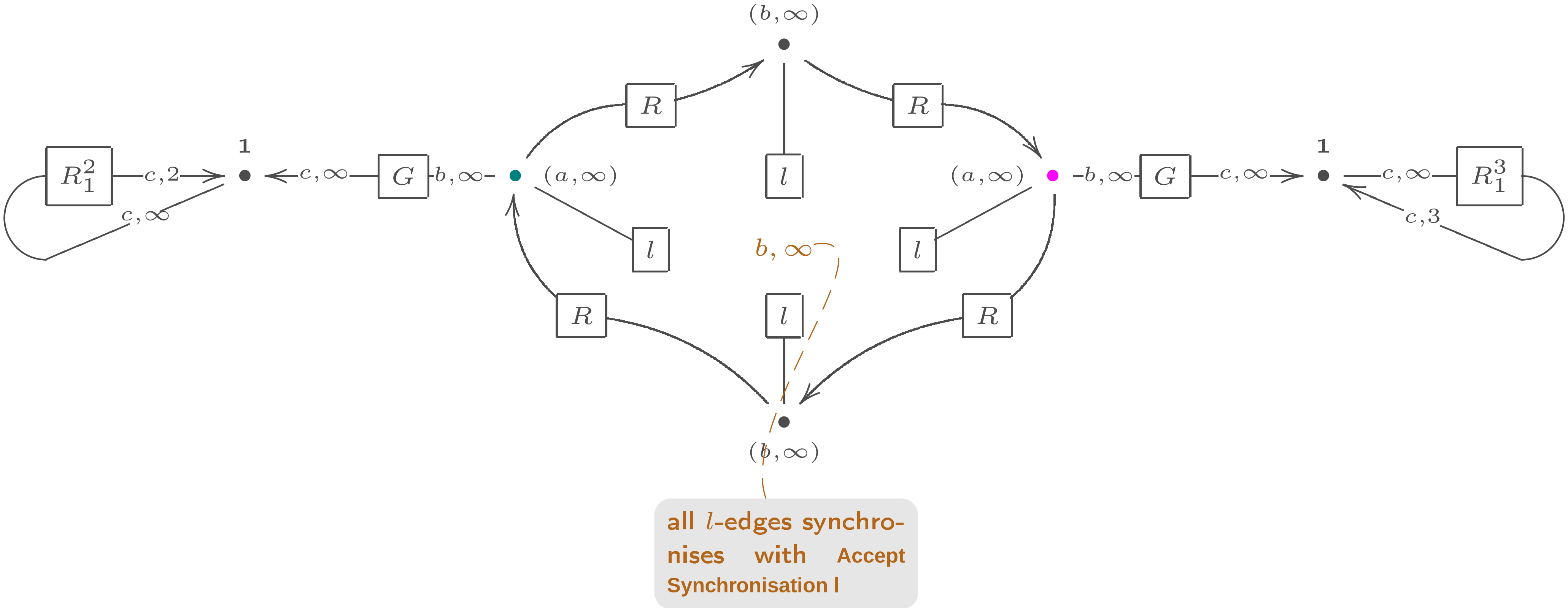
R chooses production **Create Brother** $u = 5$ (satisfying condition $5 \leq 5$) and $n = 2$ while R chooses $u = 4$ (satisfying condition $4 \leq 5$) and $n = 3$. The resulting synchronisation produces the new weights for the nodes as,

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



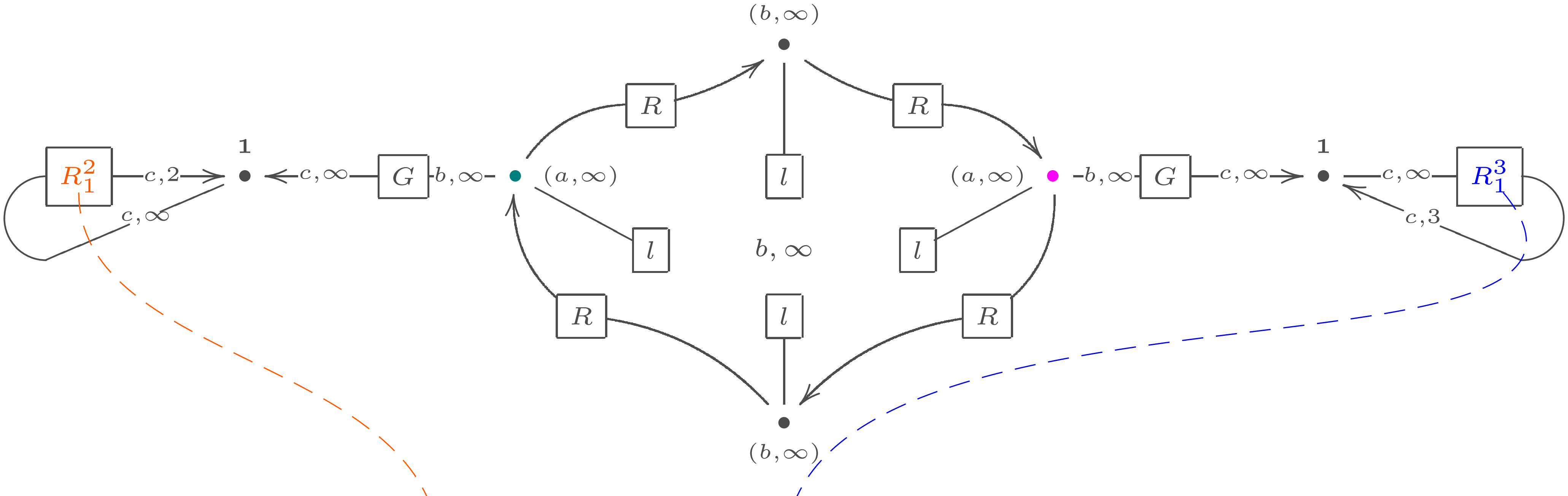
$$\mathbf{CreatGate}(r = 1, u = 2) \times \mathbf{CreatGate}(r = 1, u = 3) \times \mathbf{Accept}^*$$

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.



Note that • and • now are internal.

The ring case study



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

Let $\mathcal{N}$ be a set of names, and $\mathrm{sym}(N) \triangleq \{\rho \in \mathrm{Aut}(\mathcal{N}) \mid \forall x \notin N. \rho(x) = x\}$, if $N \subseteq \mathcal{N}$,

States of HD-automata are defined by named sets

Definition 1 *Permutation algebra* $\langle S, O \rangle$

1. *the carrier* S *is a set and*
2. $O \subseteq \{\widehat{\rho} : S \to S \mid \rho \in \mathrm{Aut}(\mathcal{N})\}$ *are s.t.*

- $\widehat{id} \in O$ *and, for all* $x \in S$
- $x\widehat{id} = x$
- $\forall \rho_1, \rho_2 \in \mathrm{Aut}(\mathcal{N}).\ x\widehat{\rho_1; \rho_2} = (x\widehat{\rho_1})\widehat{\rho_2}$

Definition 2 *Named set (ns)* $\langle Q, \mathrm{g} \rangle$

1. Q *is a permutation algebra;*
2. $\mathrm{g} : Q \to \bigcup_{N \in \wp_{fin}(\mathcal{N})} \{\mathrm{sym}(N)\}$ *s.t.* $\forall \rho \in \mathrm{g}(q). q = q\underline{\widehat{\rho}}$.

$|q| = \mathrm{dom}(\rho) \in \mathrm{g}(q)$ are the names of q.
$\|q\|$ is the cardinality of $|q|$.

Transitions among states are represented by means of named functions

Definition 3 *A named function* $\langle h : S \to D, \Sigma \rangle$ *is s.t.* S *and* D *are nss,* h *is a function from* Q_S *to* Q_D *and* $\forall q \in S. \Sigma(q) \in \wp_{fin}((|q|_S \cup \{\star\})^{|h(q)|_D})$ *s.t.*

1. $\forall \sigma \in \Sigma(q).\ \ \mathrm{g}D(h(q)); \sigma = \Sigma(q)$,
2. $\forall \sigma \in \Sigma(q).\ \ \sigma; \mathrm{g}S(q) \subseteq \Sigma(q)$,
3. *any function of* $\Sigma(q)$ *is injective.*

- The category **NS** has named sets as objects and named functions and
  1. $\perp = \langle \emptyset, \emptyset \rangle$ is initial object, $I = \langle \{*\}, * \mapsto \emptyset \rangle$ is the terminal object and
  2. the covariant powerset functor on $\mathbf{Set}$ is $\wp_{\mathsf{fin}}(D) = \langle \wp_{\mathsf{fin}}(D_Q), \mathrm{g} \rangle$, where, given $Q \subseteq Q_D$, $\mathrm{g}(Q) = \{\rho \mid \rho \text{ is a permutation over } \bigcup_{q \in Q} |q|\} \wedge Q\rho = Q$.

Definition 4 *Given a ns L, a HD-automaton over L is a coalgebra for*

$$T_L(D) = \wp_{\mathit{fin}}(L \otimes D)$$

- where the pairing operation $D \otimes E = \langle Q_D \times Q_E, \mathrm{g} \rangle$ is s.t. $\mathrm{g} : Q_D \times Q_E \to \bigcup_{N,M \in \wp_{\mathsf{fin}}(\mathcal{N})} \{\mathrm{sym}(N) + \mathrm{sym}(M)\}$ where

$$\mathrm{g}(d,e) = \{\rho_1 + \rho_2 \mid \rho_1 \in \mathrm{g}_D(d) \wedge \rho_2 \in \mathrm{g}_E(e)\}$$

(formally, $D \otimes E$ is not a ns but $\mathrm{g}(d,e)$ is a symmetry on $|d| + |e|$

Fixed a HD-automaton K on the functor $T_L(D) = \wp_{\mathsf{fin}}(L \otimes D)$, T : **NS** $\rightarrow$ **NS** is the functor s.t.

$$T(D) = \begin{cases} T_L(D) & D \in \mathsf{obj}(\mathbf{NS}) \\ \langle \mathrm{h}, \Sigma \rangle & D = \langle h_D, \Sigma_D \rangle \in \mathbf{NS}(E, F) \text{ for } E, F \in \mathsf{obj}(\mathbf{NS}) \end{cases}$$

where, $\begin{cases} \mathrm{h}(B) & = & \{\langle l, \mathrm{h}_D(q) \rangle \mid \langle l, q \rangle \in B\} \\ \Sigma(B) & = & \{\langle l, \mathrm{h}_D(q), \sigma; \sigma' \rangle \mid \langle l, q, \sigma' \rangle \in B \wedge \langle l, q', \sigma' \rangle \in \Sigma_D(q)\} \end{cases}$ $B \in \wp_{\mathsf{fin}}(L \otimes E)$, Fixed a HD-automaton K on the functor $T_L(D) = \wp_{\mathsf{fin}}(L \otimes D)$, T : **NS** $\to$ **NS** is the functor s.t.

$$T(D) = \begin{cases} T_L(D) & D \in \mathsf{obj}(\mathbf{NS}) \\ \langle \mathrm{h}, \Sigma \rangle & D = \langle h_D, \Sigma_D \rangle \in \mathbf{NS}(E, F) \text{ for } E, F \in \mathsf{obj}(\mathbf{NS}) \end{cases}$$

where, $\begin{cases} \mathrm{h}(B) & = & \{\langle l, \mathrm{h}_D(q) \rangle \mid \langle l, q \rangle \in B\} \\ \Sigma(B) & = & \{\langle l, \mathrm{h}_D(q), \sigma; \sigma' \rangle \mid \langle l, q, \sigma' \rangle \in B \wedge \langle l, q', \sigma' \rangle \in \Sigma_D(q)\} \end{cases}$ $B \in \wp_{\mathsf{fin}}(L \otimes E)$,

A normalisation functor N is any functor s.t. $N(D)$ is isomorphic to a subset of D.

Fixed a HD-automaton K on the functor $T_L(D) = \wp_{\mathsf{fin}}(L \otimes D)$, T : **NS** $\rightarrow$ **NS** is the functor s.t.

$$T(D) = \begin{cases} T_L(D) & D \in \mathsf{obj}(\mathbf{NS}) \\ \langle \mathrm{h}, \Sigma \rangle & D = \langle h_D, \Sigma_D \rangle \in \mathbf{NS}(E, F) \text{ for } E, F \in \mathsf{obj}(\mathbf{NS}) \end{cases}$$

$$\text{where, } \begin{cases} \mathrm{h}(B) & = & \{\langle l, \mathrm{h}_D(q)\rangle \mid \langle l, q\rangle \in B\} \\ \Sigma(B) & = & \{\langle l, \mathrm{h}_D(q), \sigma; \sigma'\rangle \mid \langle l, q, \sigma'\rangle \in B \wedge \langle l, q', \sigma'\rangle \in \Sigma_D(q)\} \end{cases} \quad B \in \wp_{\mathsf{fin}}(L \otimes E),$$

A normalisation functor N is any functor s.t. $N(D)$ is isomorphic to a subset of D.

The minimisation algorithm on a T_L coalgebra $(D, K : D \rightarrow T_L(D))$ is

$$\begin{array}{lcll} H_{(0)} & \triangleq & \langle q \mapsto \bot, q \mapsto \emptyset \rangle, & \text{where } \mathrm{dom}(H_{(0)}) = D \\ H_{(i+1)} & \triangleq & K; N(T(H_{(i)})), & \end{array}$$

where N is a normalisation functor

Fixed a HD-automaton K on the functor $T_L(D) = \wp_{\mathsf{fin}}(L \otimes D)$, T : **NS** $\to$ **NS** is the functor s.t.

$$T(D) = \begin{cases} T_L(D) & D \in \mathsf{obj}(\mathbf{NS}) \\ \langle \mathrm{h}, \Sigma \rangle & D = \langle h_D, \Sigma_D \rangle \in \mathbf{NS}(E, F) \text{ for } E, F \in \mathsf{obj}(\mathbf{NS}) \end{cases}$$

where, $$\begin{cases} \mathrm{h}(B) & = & \{\langle l, \mathrm{h}_D(q) \rangle \mid \langle l, q \rangle \in B\} \\ \Sigma(B) & = & \{\langle l, \mathrm{h}_D(q), \sigma; \sigma' \rangle \mid \langle l, q, \sigma' \rangle \in B \wedge \langle l, q', \sigma' \rangle \in \Sigma_D(q)\} \end{cases} \quad B \in \wp_{\mathsf{fin}}(L \otimes E),$$

A normalisation functor N is any functor s.t. $N(D)$ is isomorphic to a subset of D.

The minimisation algorithm on a T_L coalgebra $(D, K : D \to T_L(D))$ is

$$\begin{array}{lcll} H_{(0)} & \triangleq & \langle q \mapsto \bot, q \mapsto \emptyset \rangle, & \text{where } \mathrm{dom}(H_{(0)}) = D \\ H_{(i+1)} & \triangleq & K; N(T(H_{(i)})), & \end{array}$$

where N is a normalisation functor

- All the states of K are initially considered equivalent
- At the $(i+1)$-th step, $H_{(i)}$ through T is first normalised and then mapped through K
- at the end, the kernel yields the equivalence classes grouping equivalent states.

Theorem The iterative partition refinement algorithm is convergent on finite HD-automata whenever the normalisation functor is monotone on nfs.

Proof.

By construction, $\wp_{\mathsf{fin}}(\_)$ is monotone, hence T is monotone because it is the composition of two monotone functors. Therefore, $\mathcal{M} : H \mapsto K; T(H)$ is monotone and finite. Finally, all nfs chains having finite domain are finite, hence, the iterative algorithm converges to the maximal fix-point of $\mathcal{M}$.

This proof mimics that in [FMT05a] with the difference that there only the case of the early semantics of π-calculus is dealt with, while here, the result is extended to the general case of finite HD-automata (with the only additional assumption that the normalisation functor is monotone).