A Declarative Approach to Wide-Area Network Programming with Service Level Agreement

Emilio Tuosto

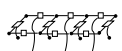
Dipartimento di Informatica, Università di Pisa

via F. Buonarroti 2, 56127 Pisa, Italy. +39-50-2212799. +39-50-2212726.

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

Padova, 6 February 2004





Plan of the talk

- What 'WAN programming' means (for us)
- Declarative programming model: Hypergraphs
  - Hint on its adequacy
    - Ambient calculus
    - Wireless communications
    - GRID
- Programming SLA: KAOS
  - SLA & Hypergraphs: reasoning on optimal routing
- SLAK 

- Final considerations

Global Computing

WAN programming for building *global systems*.
They are hard to be made roboust because:

- Absence of centralised control
- Client-Server not enough: P2P
- Administrative domains (Security)
- Interoperability
  - different platforms
  - different devices
    (e.g. PDA, laptop, mobile phones...)
- “Mobility” (resources and computation)
- ...

Global Computing

WAN programming for building *global systems*.
They are hard to be made roboust because:

- *Network Awareness*
  - Applications are location dependent
  - Locations have different features
  - and allow multiple access policies
- Independently programmed in a distributed environment
- Reasoning on space and time
- ...

Web Services: A programming metaphor

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

Motivations

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

- Lifting SLA issues to application level...
- ...in a WAN scenario, where programming is composition of WS
- SLA as a coordination mechanism
- A Formal Basis for Reasoning on Programmable SLA [DFM+03]
- Resource availability and access as a parameter for the SLA
- Proof techniques and tools

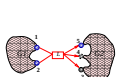
A Model for Declarative WAN Programming

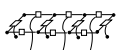
In collaboration with

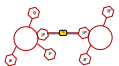
G. Ferrari (Pisa) and U. Montanari (Pisa)

Working Group:

Dan Hirsch (Pisa), Ivan Lanese (Pisa),








Process Algebraic Foundations of WAN

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

Hypergraphs Programming model[1]

- Edge replacement for graph rewritings [Fed71, Pav72]
- Graphs for distributed systems [CM83, DM87]
- Edge replacement/distributed constraint solving problem [MR96]
- Graphs grammars for software architecture styles [HIM00]
- Synchronised Hyperedge Replacement (SHR) with mobility for name passing calculi [HM01]
- Extension to node fusions [FMT01]
- ...

Hypergraphs Programming model^2

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

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

In other words:

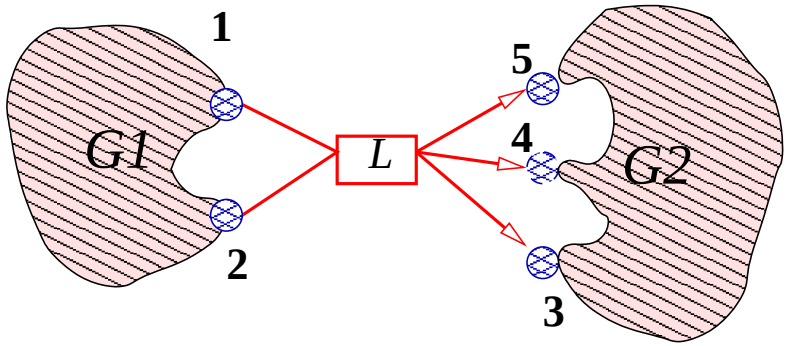
- Components are represented by hyperedges
- Systems are *bunches* of (connected) hyperedges
- Computing means to synchronously rewrite hyperedges...
- ...according to a synchronisation policy

Replacement of Hyperedges

$L \rightarrow G$

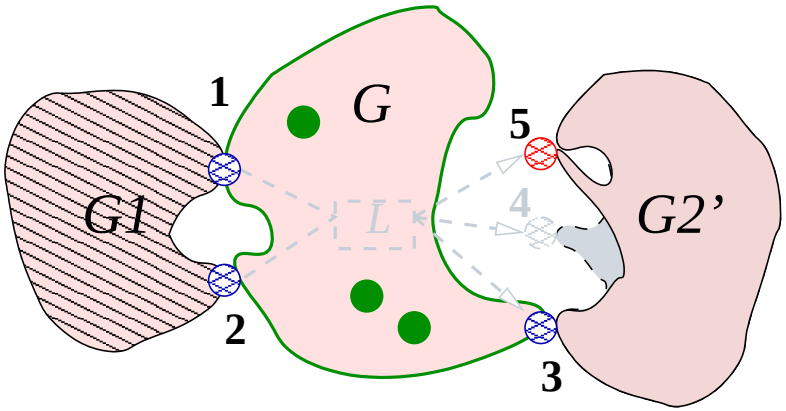
Replacement of Hyperedges

$L \rightarrow G$



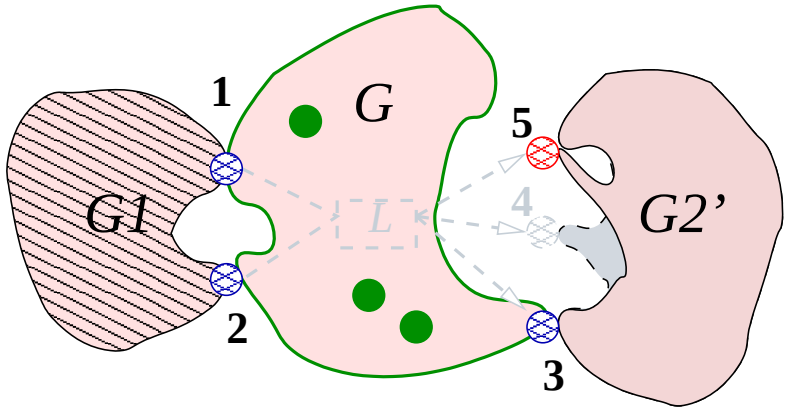
Replacement of Hyperedges

$L \rightarrow G$



Replacement of Hyperedges

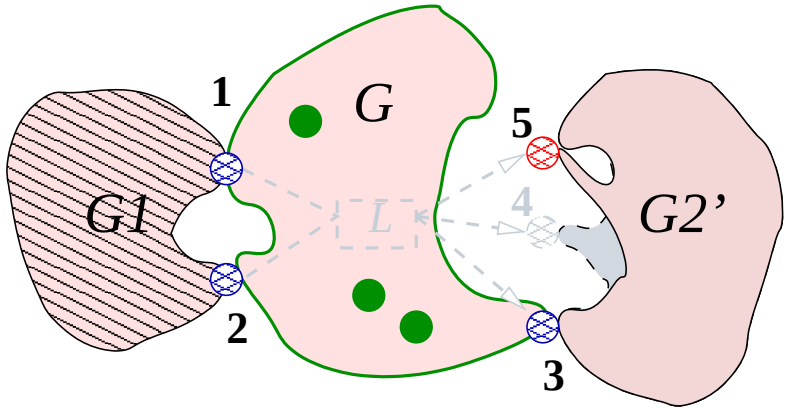
$L \rightarrow G$



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

Replacement of Hyperedges

$L \rightarrow G$



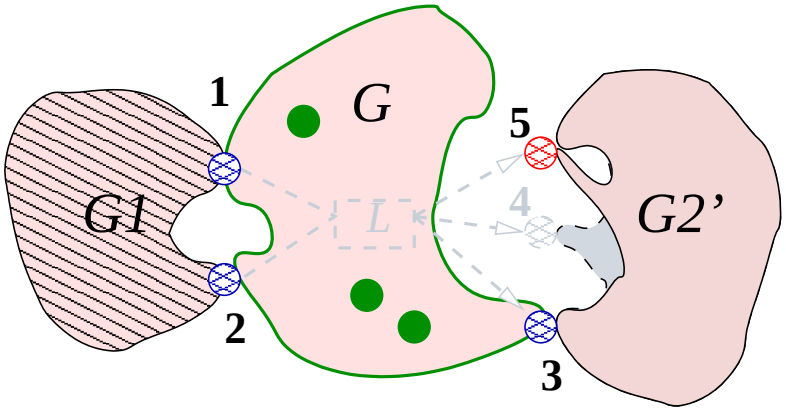
- Edge replacement: local
- Synchronisation as distributed constraint solving
- Multiple synchronisation
- New node creation
- Node fusion: model of mobility and communication

Benefits:

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

Replacement of Hyperedges

$L \rightarrow G$



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

Benefits:

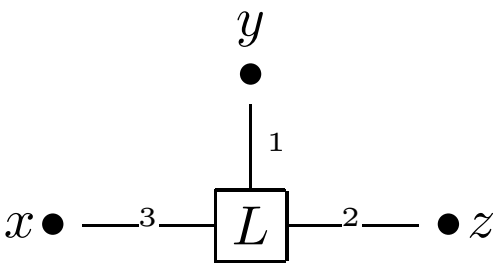
- Uniform framework for π, π-I, fusion
- LTS for Ambient ...
- ... for Klaim ...
- ... and *path reservation* for KAOS
- expressive for distributed coordination
- wireless networks

new

Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

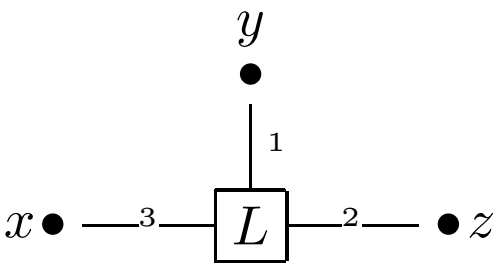
$$L : 3, \quad L(y, z, x),$$



Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

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

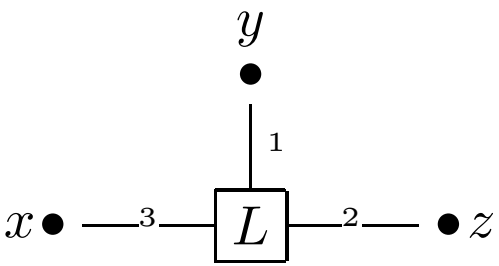


$$\begin{array}{rcl} G & ::= & nil \mid \nu\, y.G \\ & \mid & L(\vec{x}) \mid G|G \end{array}$$

Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

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



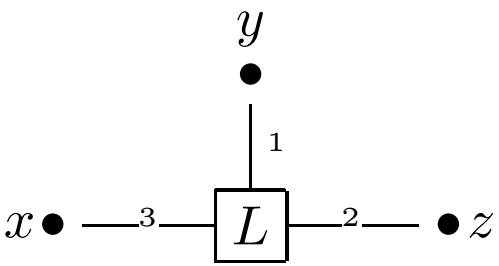
$$\begin{array}{rcl} G & ::= & nil \mid \nu\, y.G \\ & \mid & L(\vec{x}) \mid G|G \end{array}$$

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

Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

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



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

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

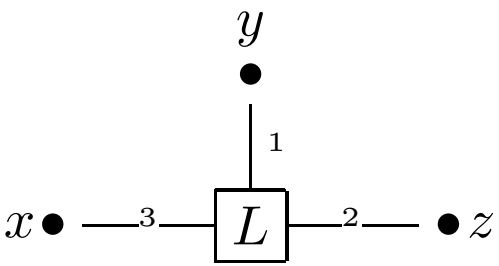
An example:

$$L : 3, \quad M : 2$$
$$x, y \vdash \nu\, z.(L(y, z, x)|M(y, z))$$

Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

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



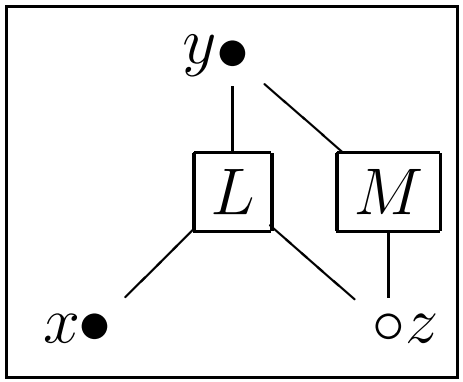
$$\begin{array}{rcl} G & ::= & nil \mid \nu\, y.G \\ & \mid & L(\vec{x}) \mid G|G \end{array}$$

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

An example:

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

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



Hypergraph Semantics: Productions

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

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

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

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

Hypergraph Semantics: Productions

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

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

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

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

Graph Rewritings

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

Hypergraph Semantics: Transitions

$$\Gamma, y \vdash G \xrightarrow[\pi]{\Lambda} \Gamma' \vdash G'$$

$$\Lambda(y) \uparrow \qquad x \simeq_\pi y \Rightarrow y \neq \pi(y)$$

$$\rho = [{}^{\pi(x)}/_{\pi(y)}]$$

$$\Gamma \vdash [{}^{x}/_{y}]G \xrightarrow[(\pi;\rho)_{-y}]{\rho\Lambda} \mathrm{n}(\rho\Lambda) \cup (\pi;\rho)_{-y}(\Gamma) \vdash \rho G'$$

$$\Gamma, y \vdash G \xrightarrow[\pi]{\Lambda \cup \{(x,a,\vec{v}),(y,\overline{a},\vec{w})\}} \Gamma' \vdash G'$$

$$x \simeq_\pi y \Rightarrow y \neq \pi(y) \qquad \rho = mgu\{[{}^{[x/_y]\vec{w}}/_{[x/_y]\vec{v}}], [{}^{\pi(x)}/_{\pi(y)}]\}$$

$$\Gamma'' = \mathrm{n}(\rho\Lambda) \cup (\pi;\rho)_{-y}(\Gamma) \qquad U = \rho(\Gamma') \setminus \Gamma''$$

$$\Gamma \vdash [{}^{x}/_{y}]G \xrightarrow[(\pi;\rho)_{-y}]{(\rho\Lambda \cup (x,\tau,\langle\rangle))} \Gamma'' \vdash \nu\, U.\rho G'$$

Hypergraph Semantics: Transitions

$$\Gamma, y \vdash G \xrightarrow[\pi]{\Lambda} \Gamma' \vdash G'$$

$$\Lambda(y) \uparrow \vee \Lambda(y) = (\tau, \langle\rangle) \qquad x \simeq_\pi y \Rightarrow y \neq \pi(y)$$

$$U = \Gamma' \setminus (\mathrm{n}(\Lambda) \cup \pi_{-y}(\Gamma))$$

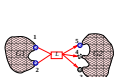
$$\Gamma \vdash \nu\, y.G \xrightarrow[\pi_{-y}]{\Lambda\setminus(y,\tau,\langle\rangle)} \mathrm{n}(\Lambda) \cup \pi_{-y}(\Gamma) \vdash \nu\, U.G'$$

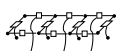
$$\Gamma_1 \vdash G_1 \xrightarrow[\pi]{\Lambda} \Gamma_2 \vdash G_2 \qquad \Gamma_1' \vdash G_1' \xrightarrow[\pi']{\Lambda'} \Gamma_2' \vdash G_2' \qquad \Gamma_1 \cap \Gamma_1' = \emptyset$$

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

Hypergraph Adequacy: Ambient

In collaboration with
G. Ferrari (Pisa) and U. Montanari (Pisa)





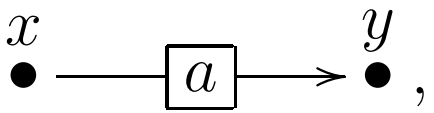

From SHR to Ambient

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

From SHR to Ambient

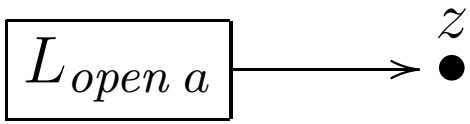
Ambient $a[...]|open\ a \longrightarrow ...$

$a[\cdots]$:



Components

$open\ a$:

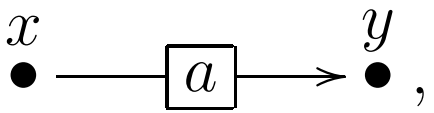


From SHR to Ambient

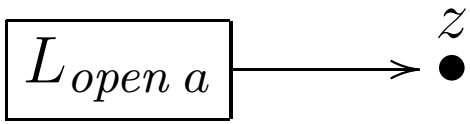
Ambient $a[...]|open\ a \longrightarrow ...$

Components

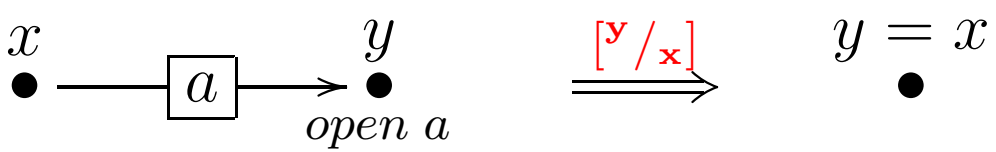
$a[\cdots]$:

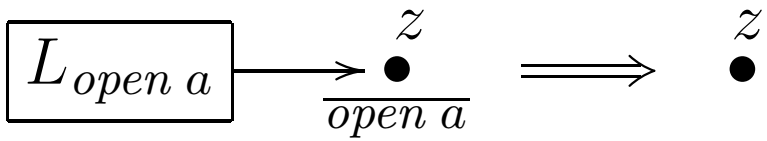


$open\ a$:

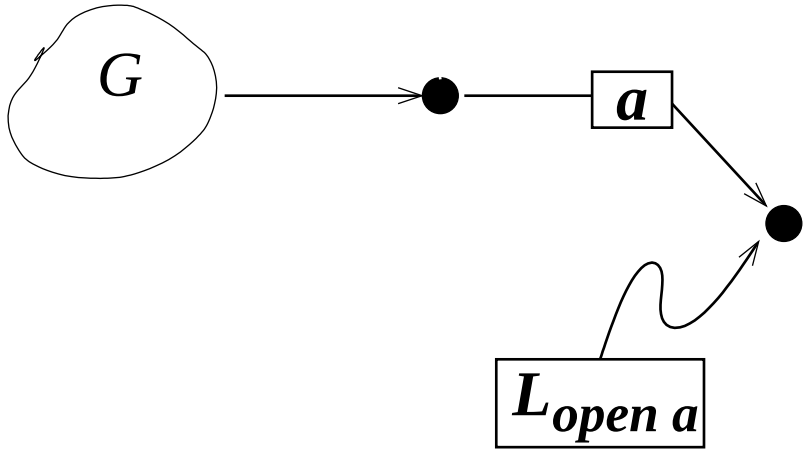


Productions

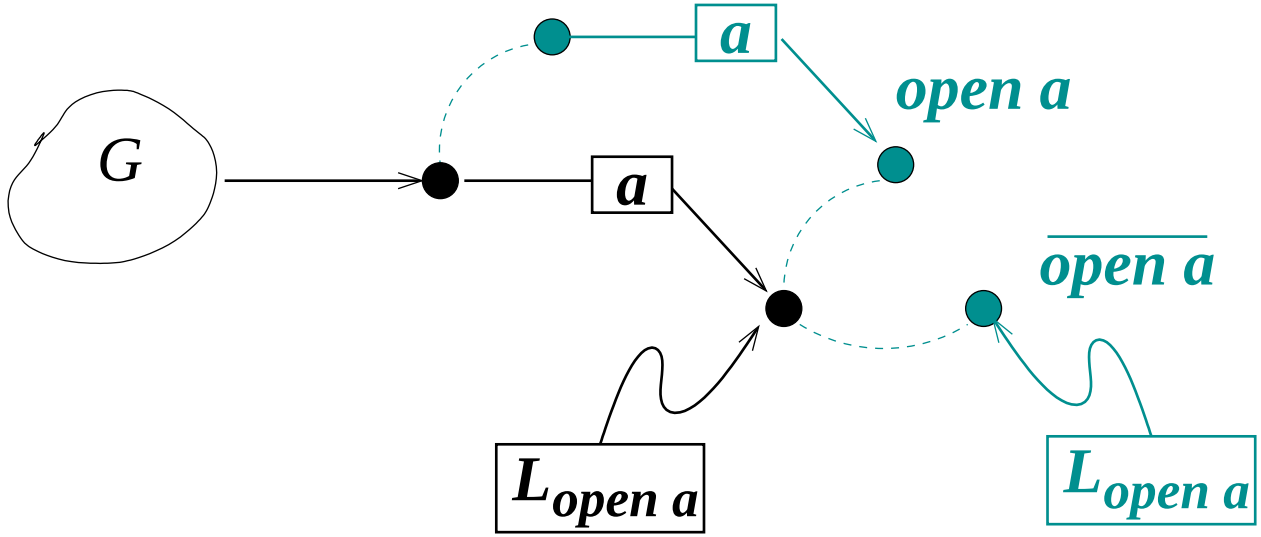




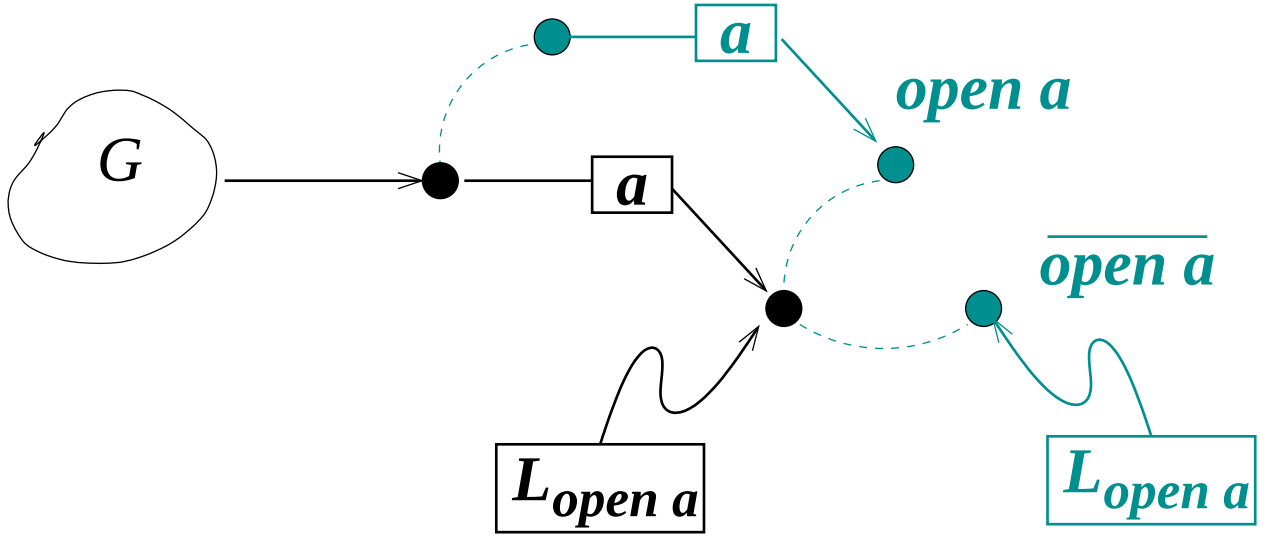
Node Fusion

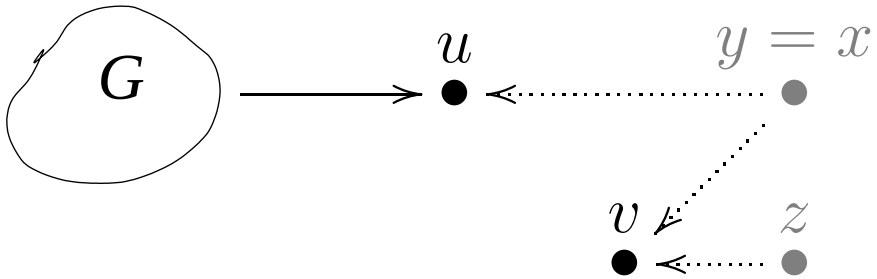


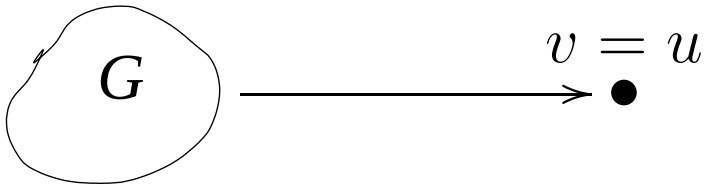
Node Fusion



Node Fusion



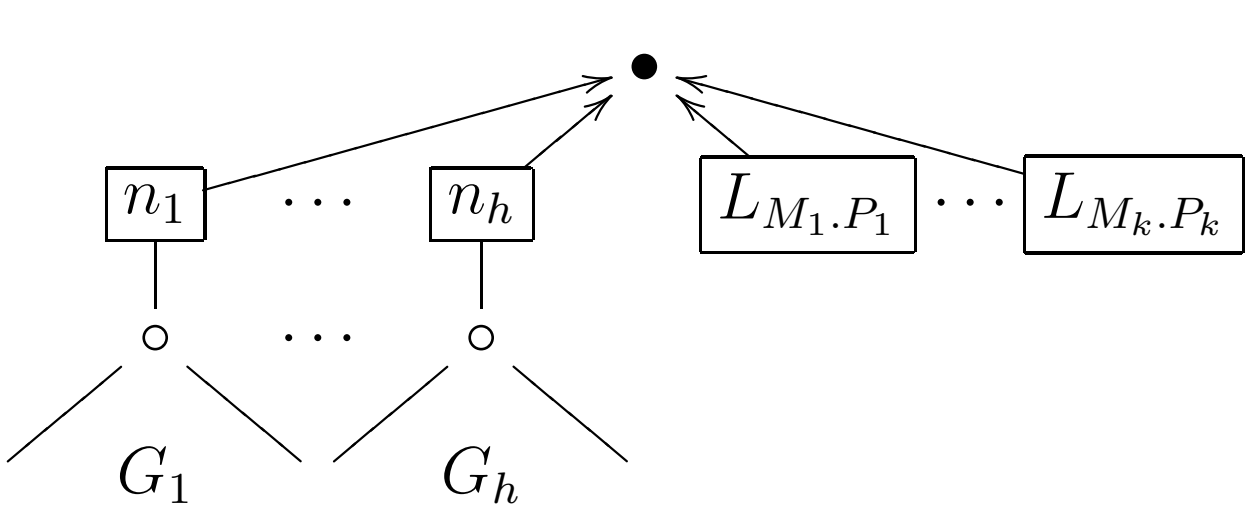




Graphs and Ambient

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

Ambient Graphs

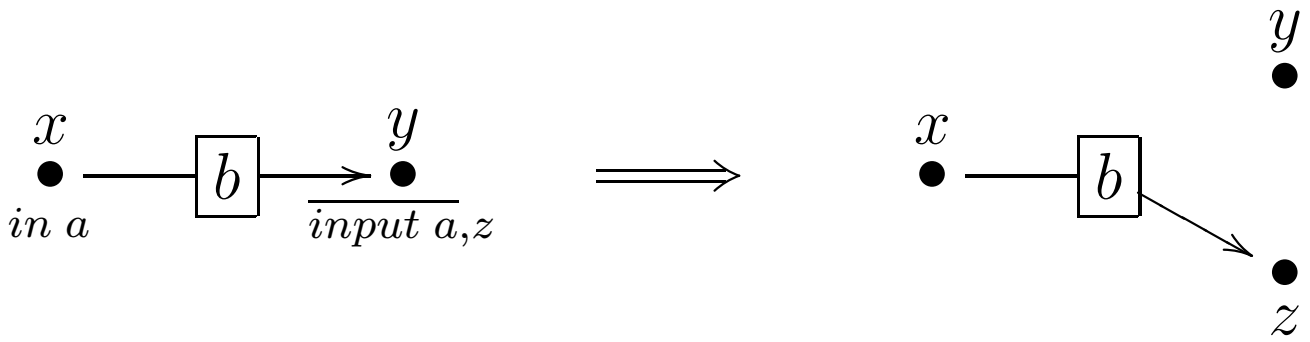


Theorem $[\![\ \_\ ]\!]_\_$ is a bijection on ambient graphs

Coordination Productions for Ambient

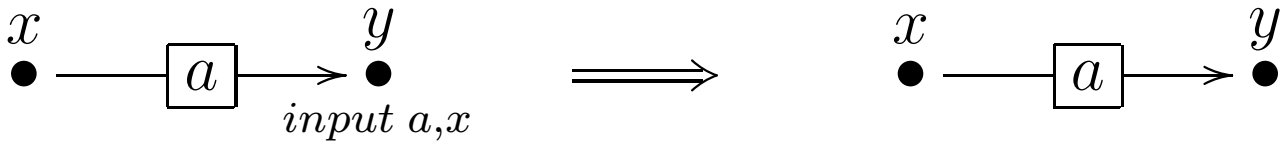
$$x, y \vdash b(x, y) \xrightarrow{\{(x,in\ a,\langle\rangle),(y,\overline{input\ a},\langle z\rangle)\}} x, y, z \vdash b(x, z)$$

$(input1)$



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

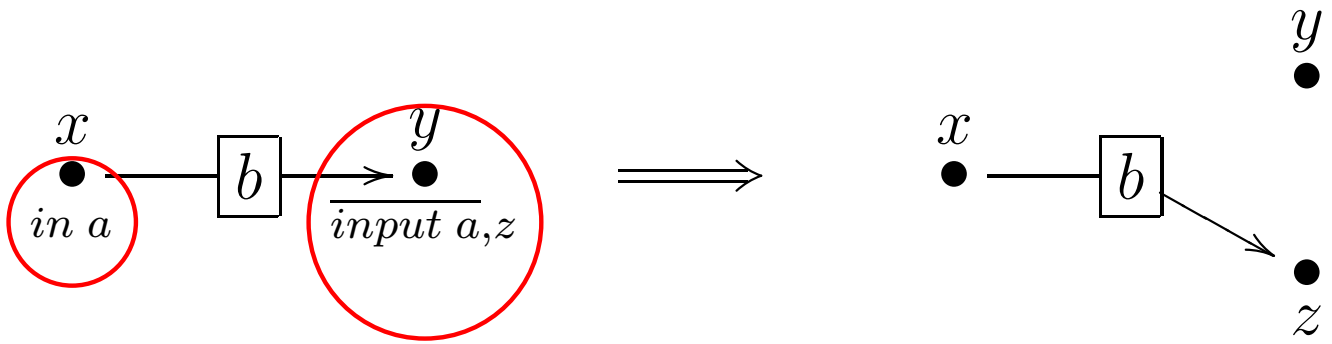
$(input2)$



Coordination Productions for Ambient

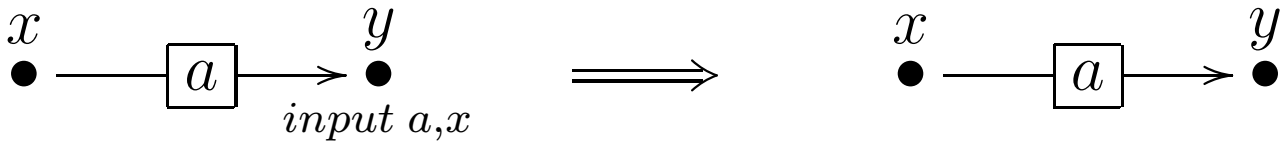
$$x, y \vdash b(x, y) \xrightarrow{\{(x,in\ a,\langle\rangle),(y,\overline{input\ a},\langle z\rangle)\}} x, y, z \vdash b(x, z)$$

$(input1)$



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

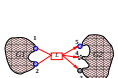
$(input2)$

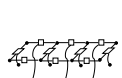


Semantic Correspondence

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

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




Semantic Correspondence

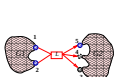
Theorem If $P \rightarrow Q$ then $[\![P]\!]_x \xrightarrow[id]{\Lambda} [\![Q]\!]_x$ and

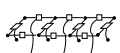
- either $\Lambda = \emptyset$
- or $\Lambda = \{(x, \tau, \langle\rangle)\}$

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

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

Hypergraph Adequacy: Wireless communications






Wireless Phenomena

- Wireless networks devices present peculiarities wrt wired ones
  - Dynamism of network topology
  - Energy constraints
  - Transmitting capacity
- Hence, nodes can asynchronously disappear
- Wireless networks typically are *peer-to-peer*
- the physical environment might cause interferences or interdict communications

For ad-hoc networks that share the same spectrum, new methods of cooperation are required to permit coexistence. Such methods are difficult to research without real-world channel models and simulation methodologies; there is still fundamental work to be done in this area [Mob98]

Traditional Models not Satisfactory

The communication infrastructure does not permit to individuate the position of components by their name

- Ambient: “vicinity” condition does not encompass any distance concept ($|$ is commutative, therefore $a[P|Q|R] = a[P|R|Q]$)
- KAOS can deal with distance between but...

... neither Ambient nor KAOS can easily model interference on wireless communications caused by *third party* movements.

This is diffi cult to capture in traditional frameworks because even if a link encompass the distance between nodes, it is under the control of the connected nodes and a third entity cannot “break” or modify it.

Tarzan

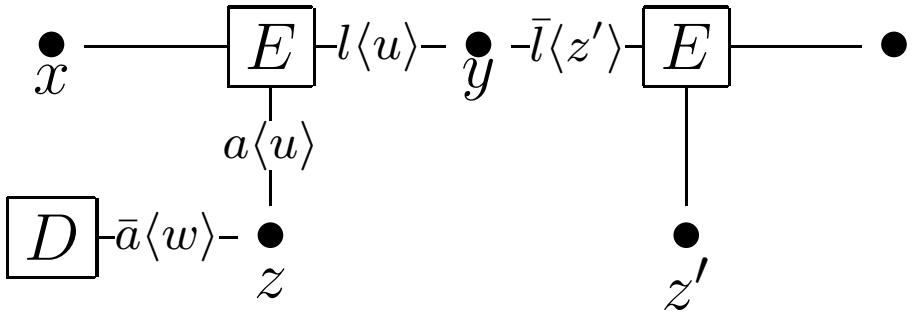
Tarzan is a SHR-based framework that captures

- Radio/Infrared Signal propagation
- Devices nomadism
- Physical environment characteristics

Tarzan

Tarzan is a SHR-based framework that captures

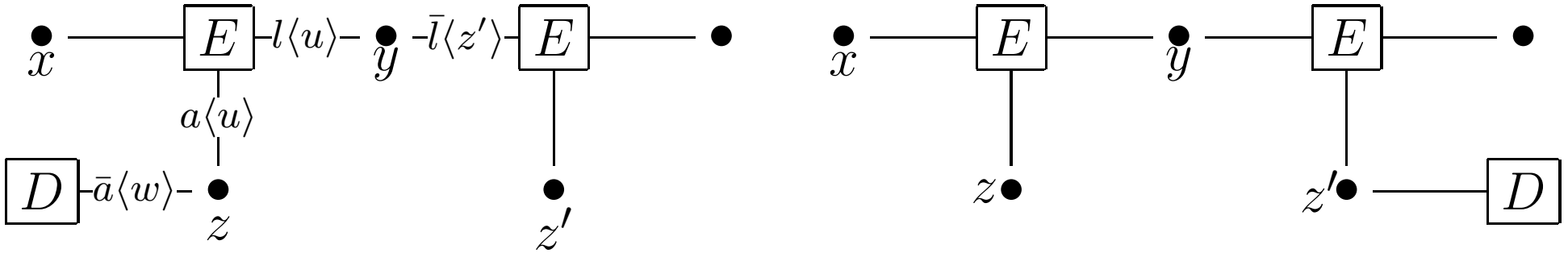
- Radio/Infrared Signal propagation
- Devices nomadism
- Physical environment characteristics



Tarzan

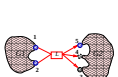
Tarzan is a SHR-based framework that captures

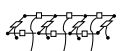
- Radio/Infrared Signal propagation
- Devices nomadism
- Physical environment characteristics

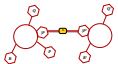


Hypergraph Adequacy: GRID

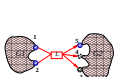
In collaboration with
Marco Aldinucci (Pisa)

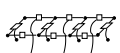







Semantics for GRID





Semantics for GRID

- Grid-aware applications are usually made of cooperating components with a graph topology

Semantics for GRID

- Grid-aware applications are usually made of cooperating components with a graph topology
- Successively behaviors to adhere performance and fault-tolerance constraints are defi ned

Semantics for GRID

- Grid-aware applications are usually made of cooperating components with a graph topology
- Successively behaviors to adhere performance and fault-tolerance constraints are defi ned
- These behaviors regard parallelism degree adaptivity matching both performance and fault-tolerance requirements

Semantics for GRID

- Grid-aware applications are usually made of cooperating components with a graph topology
- Successively behaviors to adhere performance and fault-tolerance constraints are defi ned
- These behaviors regard parallelism degree adaptivity matching both performance and fault-tolerance requirements
- Point out patterns of behaviors for abstracting suitable primitives

SHR vs GRID Programming

SHR vs GRID Programming

- Our goal is to formally defi ne a basic language containing primitives suitable for GRID.

SHR vs GRID Programming

- Our goal is to formally defi ne a basic language containing primitives suitable for GRID.
- Grid computing tries to enable the development of large applications

SHR vs GRID Programming

- Our goal is to formally defi ne a basic language containing primitives suitable for GRID.
- Grid computing tries to enable the development of large applications
- Grid-aware applications make use of computational power of distributed resources

SHR vs GRID Programming

- Our goal is to formally defi ne a basic language containing primitives suitable for GRID.
- Grid computing tries to enable the development of large applications
- Grid-aware applications make use of computational power of distributed resources
- Developing algorithms able to exploit GRID is diffi cult

SHR vs GRID Programming

- Our goal is to formally defi ne a basic language containing primitives suitable for GRID.
- Grid computing tries to enable the development of large applications
- Grid-aware applications make use of computational power of distributed resources
- Developing algorithms able to exploit GRID is diffi cult
- Programmers must design highly concurrent WAN algorithms with few homogeneity hypothesis

SHR vs GRID Programming

- Our goal is to formally defi ne a basic language containing primitives suitable for GRID.
- Grid computing tries to enable the development of large applications
- Grid-aware applications make use of computational power of distributed resources
- Developing algorithms able to exploit GRID is diffi cult
- Programmers must design highly concurrent WAN algorithms with few homogeneity hypothesis
- Hence programmers have to face up classical problems of parallel computing as well as Grid-specifi c ones

SHR as a Semantics for GRID

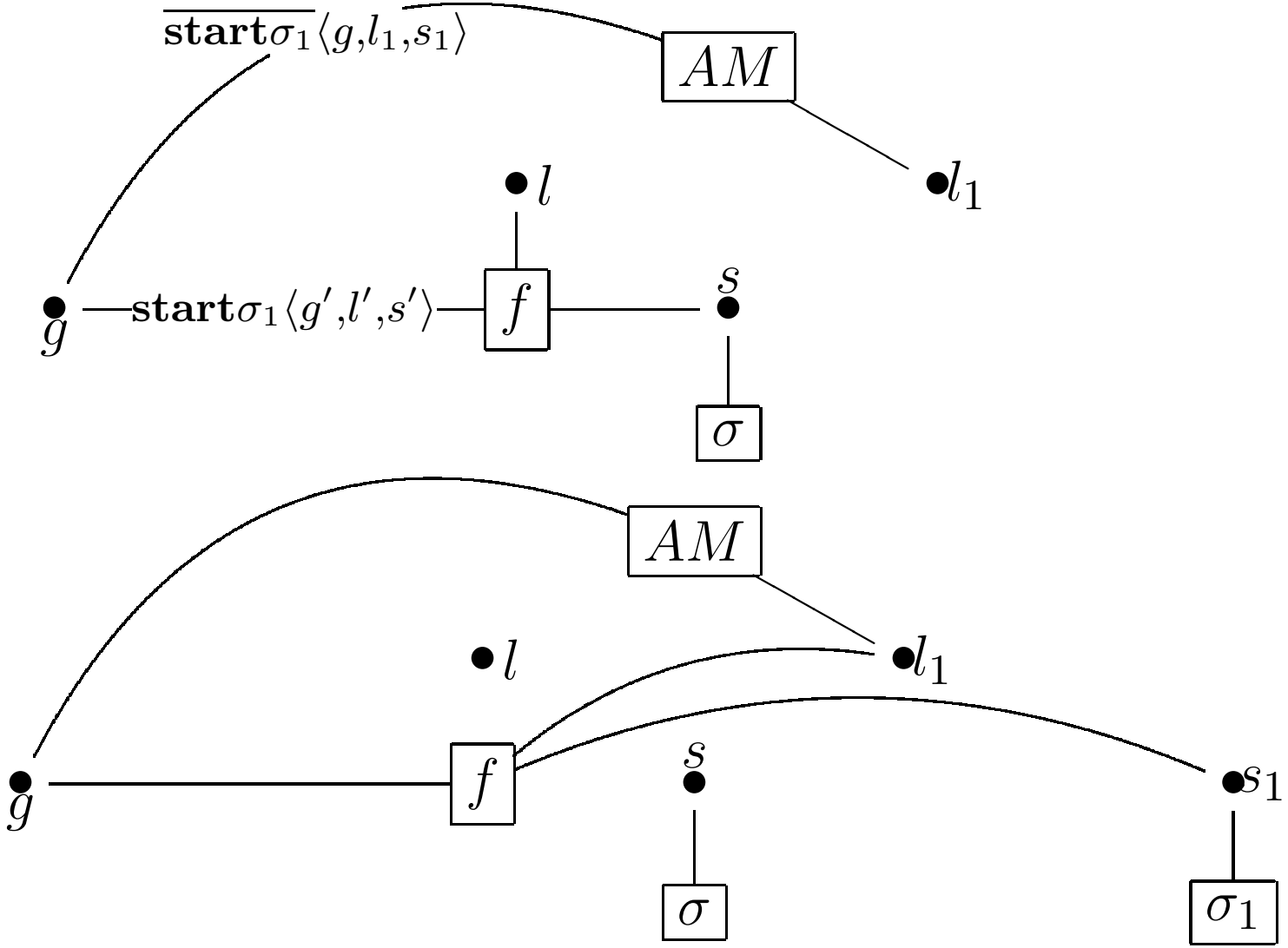
- We provide a high-level programming model for GRID programming
- We describe a SHR semantics of the framework

For instance

- Migration
- Replication
- Kill

- Components as hyperedges
- Coordination interface separated by its computational activity
- SHR rewriting mechanism for coordinating components

SHR as a Semantics for GRID



KAOS: Expressing and reasoning on Connection Properties

In collaboration with

R. De Nicola (Firenze) , G. Ferrari, U. Montanari, R. Pugliese (Firenze)

Klaim

Klaim

- Multiple tuple spaces

[DFP98]

Klaim

- Multiple tuple spaces
- Localities: fi rst class citizens

[DFP98]

Klaim

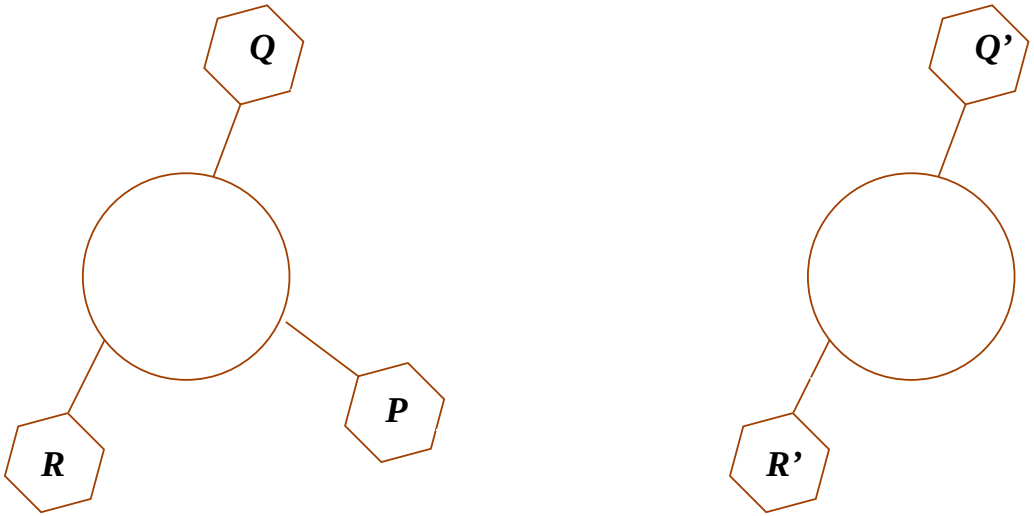
- Multiple tuple spaces
- Localities: fi rst class citizens
- Process migration

[DFP98]

Klaim

[DFP98]

- Multiple tuple spaces
- Localities: fi rst class citizens
- Process migration

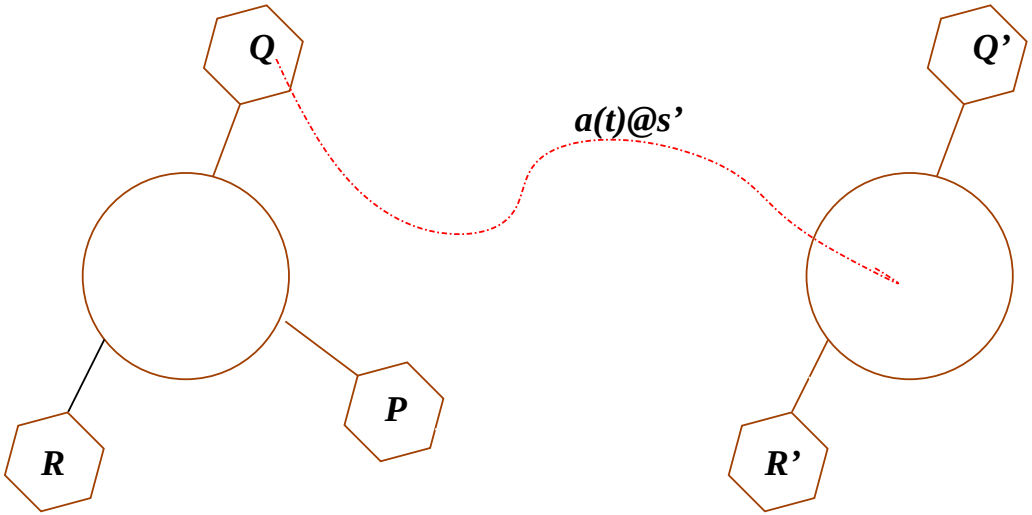


site s site s'

Klaim

- Multiple tuple spaces
- Localities: fi rst class citizens
- Process migration

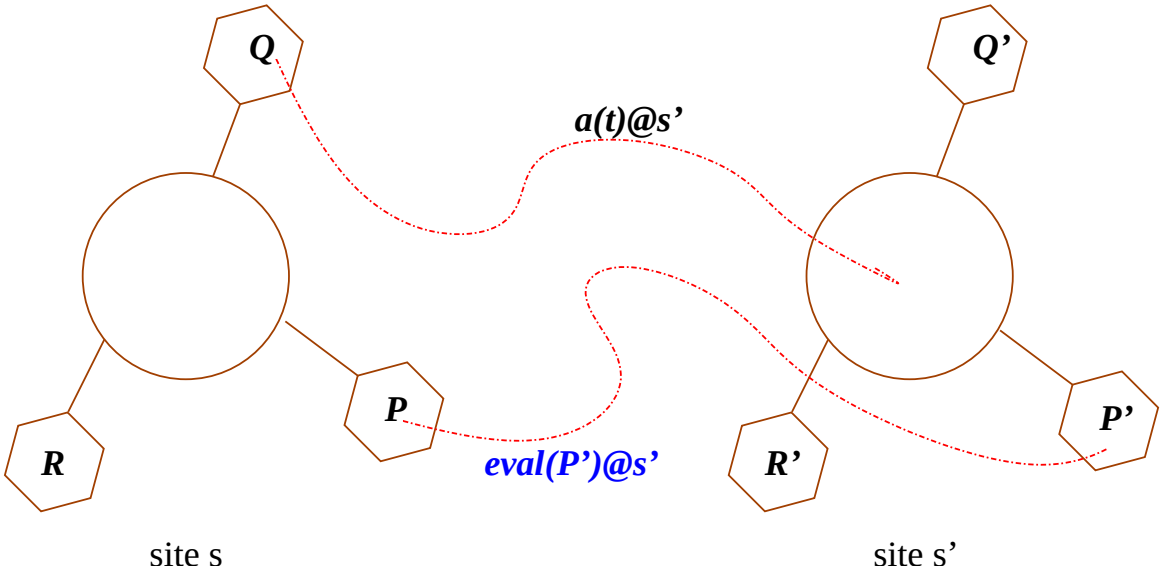
[DFP98]



Klaim

- Multiple tuple spaces
- Localities: fi rst class citizens
- Process migration

[DFP98]



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

KAOS: Gateways

KAOS: Gateways

- Coordinators (super processes)

KAOS: Gateways

- Coordinators (super processes)
- Dynamic creation of sites

KAOS: Gateways

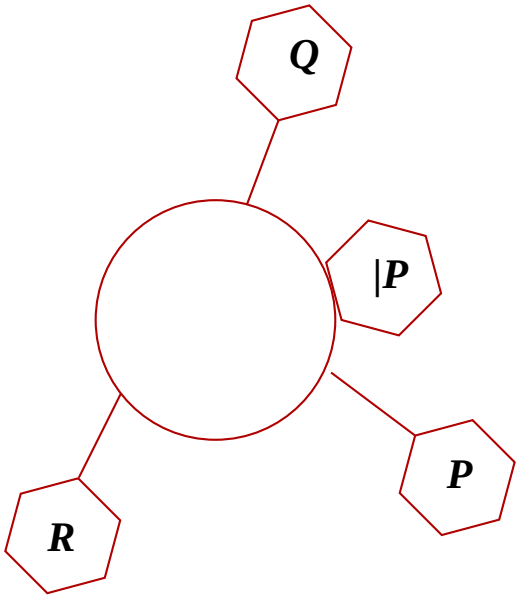
- Coordinators (super processes)
- Dynamic creation of sites
- Gateway connection management

[BLP02]

KAOS: Gateways

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

[BLP02]

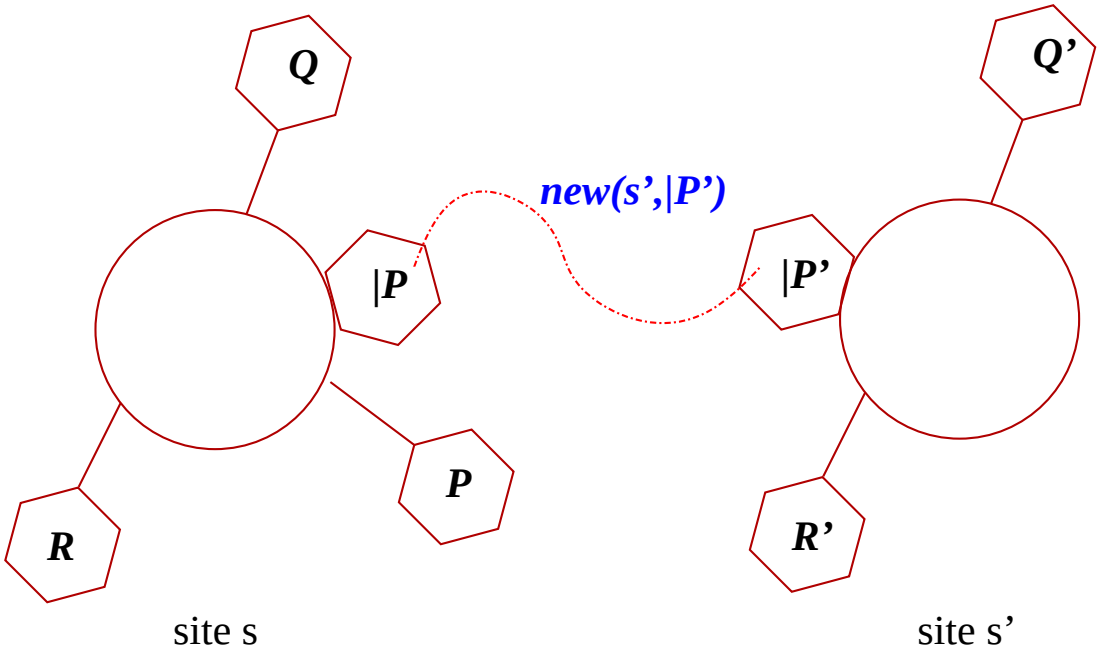


site s

KAOS: Gateways

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

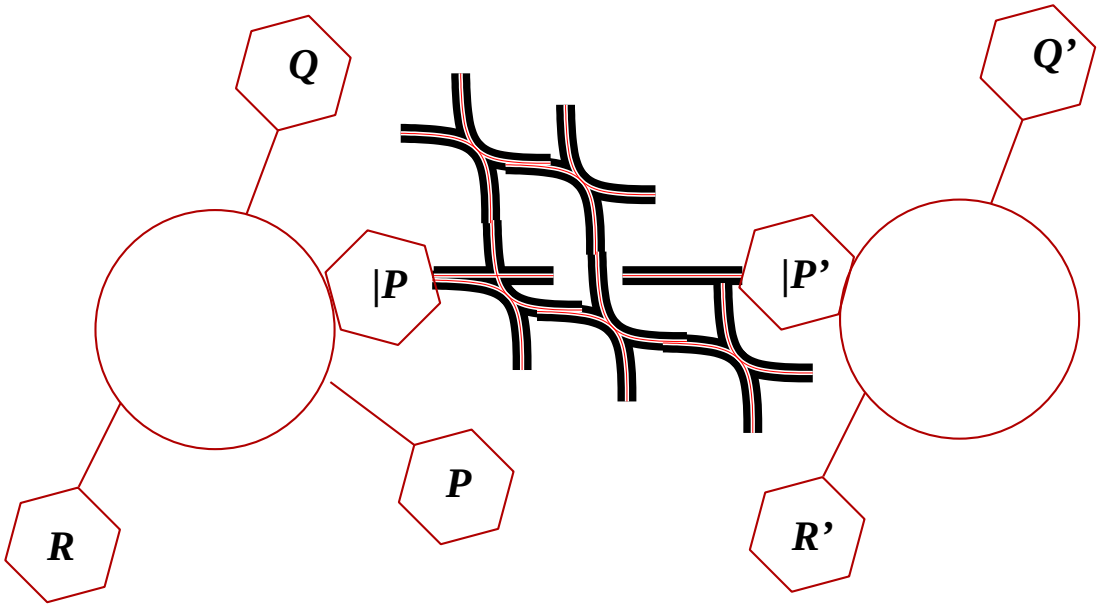
[BLP02]



KAOS: Gateways

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

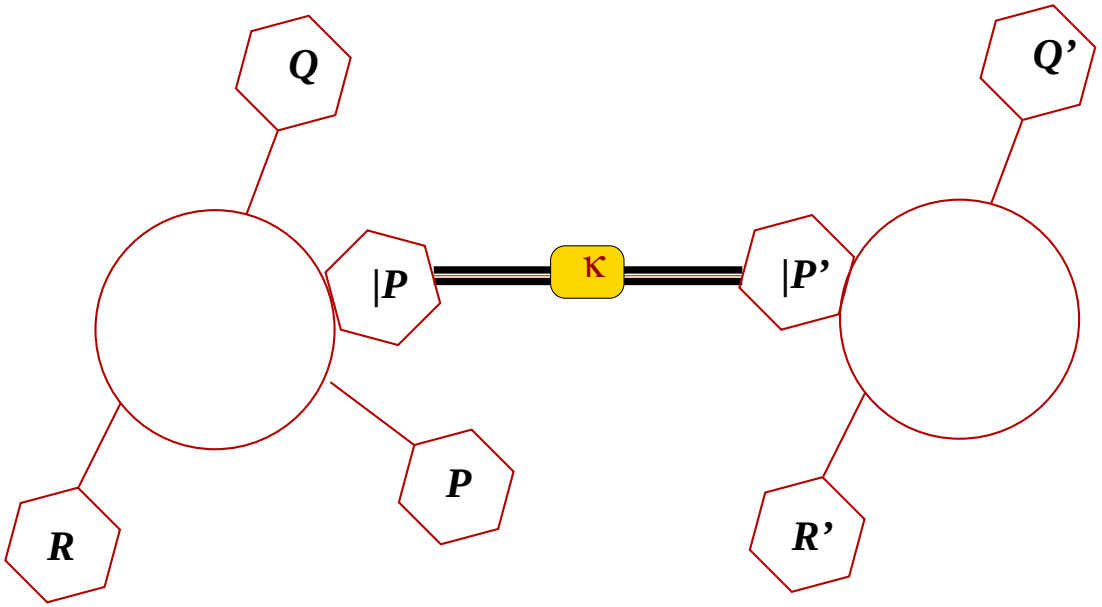
[BLP02]



KAOS: Gateways

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

[BLP02]



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

Connection costs

κ abstracts characteristics of connections (distance, access rights, price ...)

Connection costs

κ abstracts characteristics of connections (distance, access rights, price ...)
Algebra on costs: c-semiring [BMR95, BMR97]
$\langle A, +, \star, \mathbf{0}, \mathbf{1} \rangle$ where

- A is a set
- $\mathbf{0},\ \mathbf{1} \in A$
- $+ : A \times A \to A$
- $\star : A \times A \to A$

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

Connection costs

κ abstracts characteristics of connections (distance, access rights, price ...)
Algebra on costs: c-semiring [BMR95, BMR97]
$\langle A, +, \star, \mathbf{0}, \mathbf{1} \rangle$ where

- A is a set
- $\mathbf{0},\ \mathbf{1} \in A$
- $+ : A \times A \to A$
- $\star : A \times A \to A$

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

$$a \leq b \iff \exists c : a + c = b$$

$a \leq b$ means that a is more constrained than b.

Connection costs: examples

- $(\{T, F\}, \vee, \wedge, F, T)$ truth values
- $\langle N, min, +, +\infty, 0\rangle$, the c-semiring of natural numbers N
- $\langle \wp(\{A\}), \cup, \cap, A, A\}\rangle$, the powerset semiring

Cartesian product of c-semirings is a c-semiring. For instance

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

$$\langle c_1, \pi_1\rangle \otimes \langle c_2, \pi_2\rangle = \langle c_1 + c_2, \pi_1 \cap \pi_2\rangle$$

Syntax of KAOS

N	$::=$		Nets
		$s ::^{L} P$	Single node
	$\mid$	$(\nu\, s)N$	Node restriction
	$\mid$	$N_1 \parallel N_2$	Net composition

γ	$::=$		Actions
		(s)	Input
	$\mid$	$\mathbf{new}(s_\kappa)$	Node creation
	$\mid$	$\mathbf{link}(s_\kappa)$	Login
	$\mid$	$\mathbf{accept}(s_\kappa)$	Accept
	$\mid$	$\delta\, l$	Disconnect

l	$::=$		Links
		$\langle s, \kappa \rangle$	Incoming link
	$\mid$	$\langle \kappa, s \rangle$	Outgoing link

P	$::=$		Processes
		$\mathbf{nil}$	Null process
	$\mid$	$\gamma.P$	Action prefi xing
	$\mid$	$\mathbf{out}(t)$	Output
	$\mid$	$\varepsilon(P)@s$	Remote spawning
	$\mid$	$P_1 \mid P_2$	Parallel
	$\mid$	X	Process vars

Semantics of KAOS

(OUT) $$s ::^{L} \mathbf{out}(t) \xrightarrow{s \triangleright t} s ::^{L} \mathbf{nil}$$

(IN) $$s ::^{L} (x).P \xrightarrow{s \triangleleft t} s ::^{L} P[{}^{t}/_{x}]$$

(LEVAL) $$s ::^{L \cup \langle s,\kappa \rangle} \varepsilon(P)@s \xrightarrow{\tau} s ::^{L \cup \langle s,\kappa \rangle} P, \qquad \text{if } \kappa \models T(P)$$

(EVAL) $$s ::^{L} \varepsilon(P)@t \xrightarrow[s,\mathbf{1}]{s(\emptyset,\, P)@t} s ::^{L} \mathbf{nil}, \qquad \text{if } s \neq t$$

(NEW) $$s ::^{L} (\mathbf{new}(x_\kappa).P) \mid Q \xrightarrow{\tau} (\nu\, x)(s ::^{L \cup \langle \kappa, x \rangle} P \mid Q \parallel x ::^{\langle s,\kappa \rangle} \mathbf{nil}),$$
$$\text{if } x \notin \mathrm{n}(L) \cup \{s\} \cup \mathrm{fn}(Q)$$

(LLOGIN) $$s ::^{L} \mathbf{link}(s_\kappa).P \xrightarrow{\tau} s ::^{L \cup \{\langle s,\kappa \rangle, \langle \kappa, s \rangle\}} P$$

(LOGIN) $$s ::^{L} \mathbf{link}(t_\kappa).P \xrightarrow{s \overset{\kappa}{\frown} t} s ::^{L \cup \langle \kappa, t \rangle} P, \qquad \text{if } s \neq t$$

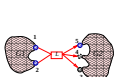
(ACCEPT) $$s ::^{L} \mathbf{accept}(t_{\kappa'}).P \xrightarrow{t \overset{\kappa}{\smile} s} s ::^{L \cup \langle t,\kappa \rangle} P, \qquad \text{if } \kappa \leq \kappa'$$

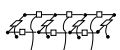
(LDISC) $$s ::^{L} \delta\langle s, \kappa \rangle.P \xrightarrow{\tau} s ::^{L \setminus \langle s,\kappa \rangle \setminus \langle \kappa . s \rangle} P$$

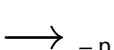
(IDISC) $$s ::^{L} \delta\, \langle t, \kappa \rangle.P \xrightarrow{\delta(s, \langle t,\kappa \rangle)} s ::^{L \setminus \langle t,\kappa \rangle} P, \qquad \text{if } t \neq s$$

(ODISC) $$s ::^{L} \delta\, \langle \kappa, t \rangle.P \xrightarrow{\delta(s, \langle \kappa,t \rangle)} s ::^{L \setminus \langle \kappa,t \rangle} P, \qquad \text{if } t \neq s$$

(NODE) $$s ::^{L \cup \langle r,\kappa \rangle} P \xrightarrow[r,\kappa]{X@s} s ::^{L \cup \langle r,\kappa \rangle} P \mid X, \qquad \text{if } X \text{ fresh}$$

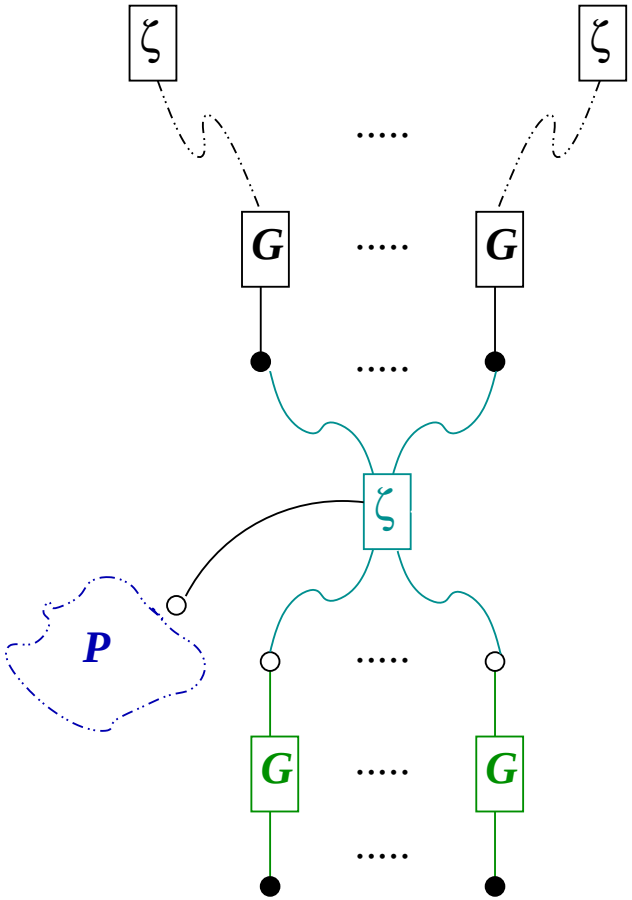




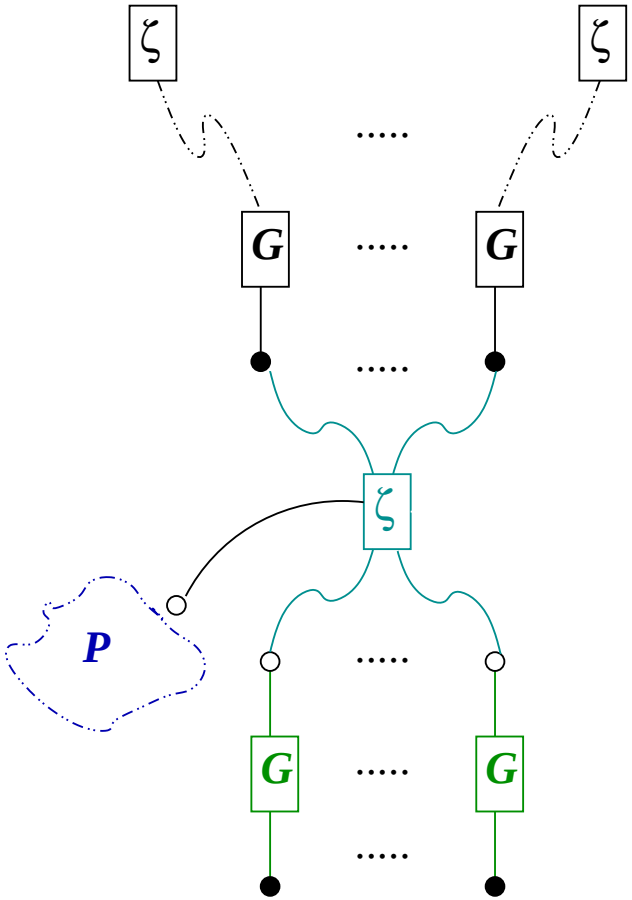
KAOS & Hypergraphs

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



KAOS & Hypergraphs

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



$$
\begin{aligned}
[\![\ \mathbf{nil}\ ]\!]_p &= nil \\
[\![\ \mathbf{out}(t)\ ]\!]_p &= L_{\mathbf{out}(t)}(p) \\
[\![\ \gamma.P\ ]\!]_p &= L_{\gamma.P}(p) \\
[\![\ \varepsilon(P)@s\ ]\!]_p &= (\nu\, u)(\varepsilon^{T(P)}_{s}(u, p) \mid S_P(u)) \\
[\![\ P_1 \mid P_2\ ]\!]_p &= [\![\ P_1\ ]\!]_p \mid [\![\ P_2\ ]\!]_p \\
[\![\ rec\ X.\, P\ ]\!]_p &= [\![\ P[^{rec\ X.\, P}/_{X}]\ ]\!]_p.
\end{aligned}
$$

KAOS's Graph semantics: pros & cons

KAOS's Graph semantics: pros & cons

– Many productions (recently reduced :-)

KAOS's Graph semantics: pros & cons

– Many productions (recently reduced :-)

= Determines the "optimal" path (also KAOS)

Theorem If $\Gamma \vdash G \xrightarrow{\Lambda \cup \{(u,\, v\ \kappa,\, \langle u \rangle)\}} \Gamma' \vdash G'$ then

1. u and v are link-connected by a path of cost κ;
2. for any

there is a $\Gamma \vdash G \xrightarrow{\Lambda \cup \{(u,\, v\ \kappa,\, \langle u \rangle)\}} \Gamma' \vdash G'$ s.t. $\kappa \leq \prod_{i=1}^{h} \kappa_i$.

($\prod$ is the c-semiring multiplication)

KAOS's Graph semantics: pros & cons

– Many productions (recently reduced :-)

= Determines the "optimal" path (also KAOS)

Theorem If $\Gamma \vdash G \xrightarrow{\Lambda \cup \{(u,\, v\ \kappa,\, \langle u \rangle)\}} \Gamma' \vdash G'$ then

1. u and v are link-connected by a path of cost κ;
2. for any

there is a $\Gamma \vdash G \xrightarrow{\Lambda \cup \{(u,\, v\ \kappa,\, \langle u \rangle)\}} \Gamma' \vdash G'$ s.t. $\kappa \leq \prod_{i=1}^{h} \kappa_i$.

($\prod$ is the c-semiring multiplication)

\+ Path reservation

KAOS's Graph semantics: pros & cons

– Many productions (recently reduced :-)

= Determines the "optimal" path (also KAOS)

Theorem If $\Gamma \vdash G \xrightarrow{\Lambda \cup \{(u,\, v\ \kappa,\, \langle u \rangle)\}} \Gamma' \vdash G'$ then

1. u and v are link-connected by a path of cost κ;
2. for any

there is a $\Gamma \vdash G \xrightarrow{\Lambda \cup \{(u,\, v\ \kappa,\, \langle u \rangle)\}} \Gamma' \vdash G'$ s.t. $\kappa \leq \prod_{i=1}^{h} \kappa_i$.

($\prod$ is the c-semiring multiplication)

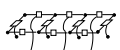
\+ Path reservation

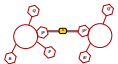
\+ Optimal path routing (e.g., Floyd-Warshall)

SLAK: Service Level Agreement in Klaim

In collaboration with
R. De Nicola








Focussing on SLA

- Modern WAN applications quest for SLA specifi cation and programming
- Consider WS:
  - programmers could drive the search phase of the required services by declaring their SLA constraint, and
  - language support guarantees satisfaction of requirements
  - Synchronizing = signing a “contract”
- KAOS is an attempt in this direction, but it has many concepts (e.g., links, costs, coordinators,...)
- We aim at refi ning KAOS in a calculus that abstracts
  - mechanisms for WAN programming...
  - ...together with SLA constraints
  - in a “flexible” framework

SLAK vs KAOS

Similarities wrt KAOS

- costs are c-semiring [BMR97]
- local (anonymous) communications
- remote spawning of processes

Simplifi cations wrt KAOS

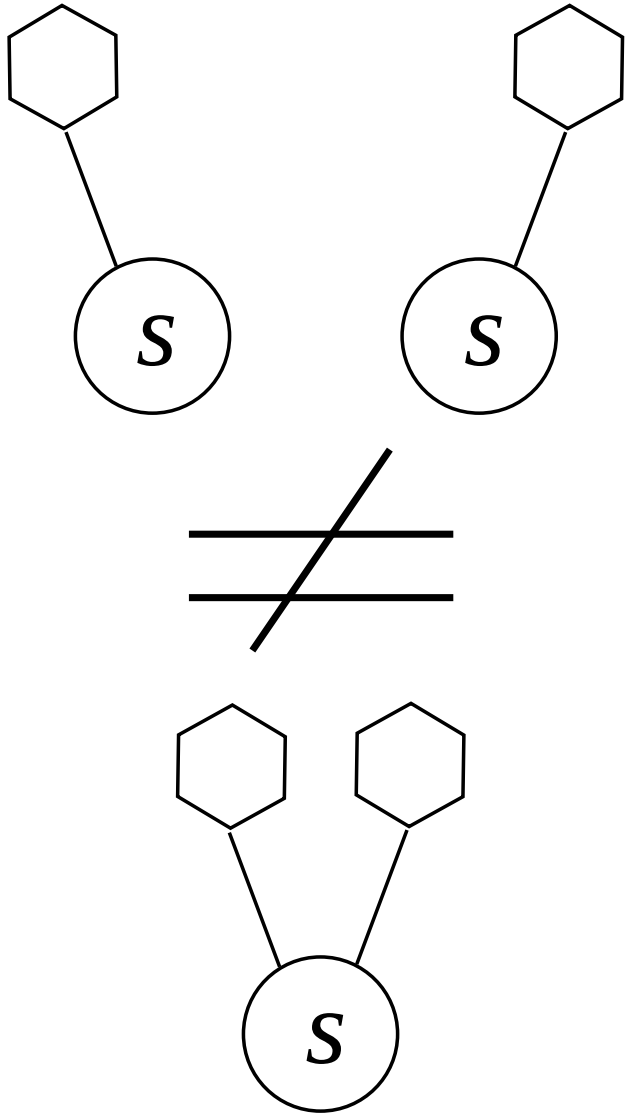
- Bidirectional Links
- Incremental defi nition
- A different (syntactic) concept of “site”
- $\Rightarrow$ *self-links* are dealt with uniformly

SLAK Syntax

N	$::=$		Nets
		$s ::^{L} P$	Located Process
	$\mid$	$N_1 \parallel N_2$	Net composition
γ	$::=$		Prefi xes
		(x)	Input
	$\mid$	$\mathbf{new}(s_\kappa)$	Node creation
	$\mid$	$\varepsilon(P)@s$	Process spawning
P	$::=$		Processes
		$\mathbf{nil}$	Null process
	$\mid$	$\gamma.P$	Action prefi xing
	$\mid$	$\mathbf{out}(t)$	Output

SLAK Syntax

N	$::=$		Nets
		$s ::^{L} P$	Located Process
	$\mid$	$N_1 \parallel N_2$	Net composition
γ	$::=$		Prefi xes
		(x)	Input
	$\mid$	$\mathbf{new}(s_\kappa)$	Node creation
	$\mid$	$\varepsilon(P)@s$	Process spawning
P	$::=$		Processes
		$\mathbf{nil}$	Null process
	$\mid$	$\gamma.P$	Action prefi xing
	$\mid$	$\mathbf{out}(t)$	Output



Local Transitions

(OUT) $$s ::^{L} \mathbf{out}(r) \xrightarrow[s,\mathbf{1}]{\langle r \rangle} s ::^{L} \mathbf{nil}$$

(IN) $$s ::^{L} (x).P \xrightarrow[s,\mathbf{1}]{(r)} s ::^{L} P[{}^{r}/_{x}], \qquad r \in \mathcal{S}$$

(NEW) $$s ::^{L} \mathbf{new}(u_\kappa).P \xrightarrow[s,\mathbf{1}]{\mathbf{new}(r)} s ::^{L \uplus r_\kappa} P \parallel r ::^{\{u_1, s_\kappa\}} \mathbf{nil}, \qquad r \notin \mathrm{dom}(L)$$

(EVAL) $$s ::^{L} \varepsilon(P)@t.Q \xrightarrow[s,\mathbf{1}]{s[P]@t} s ::^{L} Q$$

(SITE) $$s ::^{L \uplus r_\kappa} P \xrightarrow[s,\kappa]{\mathbf{from}(r)} s ::^{L \uplus r_\kappa} P$$

Global Transitions[1]

$$\text{(PAR)}\quad \frac{N_1 \xrightarrow[s,\kappa]{\alpha} N_1'}{N_1 \parallel N_2 \xrightarrow[s,\kappa]{\alpha} N_1' \parallel N_2} \qquad \mathrm{bn}(\alpha) \cap \mathrm{fn}(N_2) = \emptyset$$

$$\text{(COM)}\quad \frac{N_1 \xrightarrow[s,\mathbf{1}]{\langle r \rangle} N_1' \qquad N_2 \xrightarrow[s,\mathbf{1}]{(r)} N_2'}{N_1 \parallel N_2 \xrightarrow[s,\mathbf{1}]{\tau} N_1' \parallel N_2'}$$

Global Transitions[2]

$$\text{(ROUTE)}\quad \frac{N_1 \xrightarrow[r,\kappa_1]{s[P]@t} N_1' \qquad N_2 \xrightarrow[r',\kappa_2]{\mathbf{from}(r)} N_2' \qquad \kappa_2 \models T(P)}{N_1 \parallel N_2 \xrightarrow[r',\kappa_1 \star \kappa_2]{s[P]@t} N_1' \parallel N_2'} \quad t \neq r'$$

$$\text{(LANDING)}\quad \frac{N_1 \xrightarrow[r,\kappa_1]{s[P]@t} N_1' \qquad N_2 \xrightarrow[t,\kappa_2]{\mathbf{from}(r)} N_2' \qquad \kappa_2 \models T(P)}{N_1 \parallel N_2 \xrightarrow[t,\kappa_1 \star \kappa_2]{s[P]@} N_1' \parallel N_2' \parallel t ::^{\{t_{\kappa_1 \star \kappa_2}\}} P}$$

Correctness & Completeness

Definition Let N be a net and let s and t be two sites in N. We denote with $\mathcal{P}_N(s,t)$ the set of costs of the paths from s to t in N.

Theorem Let N be a net, then

$$N \xrightarrow[t,\kappa]{s[P]@} \Longrightarrow \quad \kappa \in \mathcal{P}_N(s,t)$$

Theorem Given a net N and a process P

$$N \xrightarrow[t,\mathbf{1}]{s[P]@} \Longrightarrow \quad N \xrightarrow[t,\kappa]{s[P]@}$$

for all $\kappa \in \mathcal{P}_N(s,t)$ s.t. $\kappa \models T(P)$

Ranges

A slight variation: $P ::= \ldots \mid \varepsilon_\kappa(P)@s$

(EVAL') $$s ::^L \varepsilon(P)@t \xrightarrow[s,\,\mathbf{1}]{s[P_\kappa]@t} s ::^L \mathbf{nil}$$

(ROUTE') $$\frac{N_1 \xrightarrow[r,\,\kappa_1]{s[P_\kappa]@t} N_1' \qquad N_2 \xrightarrow[r',\,\kappa_2]{\mathbf{from}(r)} N_2' \qquad \kappa_2 \models T(P) \quad \wedge \quad \boxed{\kappa \geq \kappa_1 \star \kappa_2}}{N_1 \parallel N_2 \xrightarrow[r',\,\kappa_1 \star \kappa_2]{s[P_\kappa]@t} N_1' \parallel N_2'} \quad t \neq r'$$

(LANDING') $$\frac{N_1 \xrightarrow[r,\,\kappa_1]{s[P]@t} N_1' \qquad N_2 \xrightarrow[t,\,\kappa_2]{\mathbf{from}(r)} N_2' \qquad \kappa_2 \models T(P) \quad \wedge \quad \boxed{\kappa \geq \kappa_1 \star \kappa_2}}{N_1 \parallel N_2 \xrightarrow[t,\,\kappa_1 \star \kappa_2]{s[P]@} N_1' \parallel N_2' \parallel t ::^{\{t_{\kappa_1 \star \kappa_2}\}} P}$$

Correspondence

Theorem If $N \xrightarrow[t,\kappa']{s[P_\kappa]@} N'$ then $\hat{N} \xrightarrow[t,\kappa']{s[P]@} \hat{N}'$ where $\hat{N}$ and $\hat{N}'$ are obtained by removing costs from ε prefixes in N and N', respectively.

Dynamic Links

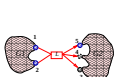
$$
\begin{array}{lll}
\gamma & ::= & \text{Prefixes} \\
 & \ldots & \\
 & \mid \quad \mathbf{link}(s_\kappa) & \text{Link} \\
 & \mid \quad \mathbf{accept}(s_\kappa) & \text{Accept} \\
 & \mid \quad \mathbf{disc}(s) & \text{Disconnect}
\end{array}
$$

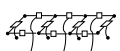
(LINK) $$s ::^{L} \mathbf{link}(r_\kappa).P \xrightarrow[s,\kappa']{\mathbf{link}(r_\kappa)} s ::^{L \uplus r_{\kappa\star\kappa'}} P$$

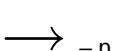
(ACCEPT) $$s ::^{L} \mathbf{accept}(r_\kappa).P \xrightarrow[s,\kappa']{\mathbf{accept}(r_\kappa)} s ::^{L \uplus r_{\kappa\star\kappa'}} P$$

(ACCEPT') $$s ::^{L \uplus r_\kappa} P \xrightarrow[s,\kappa]{\mathbf{accept}(r_\kappa)} s ::^{L \uplus r_\kappa} P$$

(DISC) $$s ::^{L} \mathbf{disc}(r).P \xrightarrow[s,\mathbf{1}]{\tau} s ::^{L \setminus r} P$$





Dynamic Links

$$\text{(ADDLINK)}\quad \frac{N_1 \xrightarrow[s,\,\kappa']{\mathbf{link}(r_\kappa)} N_1' \qquad N_2 \xrightarrow[r,\,\kappa']{\mathbf{accept}(s_\kappa)} N_2'}{N_1 \parallel N_2 \xrightarrow[s,\,\kappa \star \kappa']{\tau} N_1' \parallel N_2'}$$

Summing up...

Conclusions

Declarative Model of WAN programming

- Captures aspects of Web Services metaphor
- Suitable for representing several WAN network issues e.g.,
  - different models (Ambient, Klaim,...)
  - wireless communications
  - routing
- SLA as a Coordination Mechanism
- Toward formal techniques for SLA programming

On-going activity

- Simplifying the hypergraph framework
  Recently a very simple semantics has been found:
  - neither Γ nor π,
  - just one inference rule.

  Decreased expressive power but still an interesting model of WAN

- Modelling GRID via SHR [AT03]
- Modelling wireless phenomena [Tuo04]
- Tools
  - SHE (Synchronized Hyperedge Environment [CTT04])
  - Gredi (GRammar EDItor)

Future work

- Extending synchronizing policies
  - Synchronization algebra = c-semiring?
  - Is it possible to use c-semirings for semantic web?
  - Rule matching problem = semantic serch/bind of services
- Applying SLA to wireless routing problem
- Extending SHR futures to match GRID's data-oriented issues
- Analyzing SLAK expressivity (wrt KAOS)
- Extending SLAK
- Model Checking SLAK by exploiting [Lor02, DL02]
- and Gredi + SHE for verifying wireless networks
- SLAK's observational semantics

References

[AT03] Marco Aldinucci and Emilio Tuosto. Toward a formal semantics for grid computing. Submitted for publication to Special Issue of Semantics and Cost Models for High-Level Parallel Programming, Computer Languages, Systems and Structures - CLSS, December 2003.

[BBD+03] Lorenzo Bettini, Viviana Bono, Rocco De Nicola, Gianluigi Ferrari, Daniele Gorla, Michele Loreti, Eugenio Moggi, Rosario Pugliese, Emilio Tuosto, and Betti Venneri. The klaim project: Theory and practice. In Corrado Priami, editor, *Global Computing: Programming Environments, Languages, Security and Analysis of Systems*, number 2874 in LNCS. Springer-Verlag, 2003.

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

[BC99] Boumediene Bal, Henri E. Belkhouche and Luca Cardelli, editors. *Workshop on Internet Program-*

ming Languages, volume 1686 of *LNCS*. Springer, 1999.

[BCC01] Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Boxed Ambients. In *TACS 2001*, number 2215 in Lecture Notes in Computer Science, pages 38–63. Springer, 2001.

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

[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.

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

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

[CTT04] Pietro Cenciarelli, Ivano Talamo, and Alessandro Tiberi. Ambient graph rewriting. In *5th International Workshop on Rewriting Logic and its Applications*. Elsevier, 2004. To appear.

[DFM+03] Rocco De Nicola, Gianluigi Ferrari, Ugo Montanari, Rosario Pugliese, and Emilio Tuosto. A formal basis for reasoning on programmable qos. In *International Symposium on Verification – Theory and Practice – Honoring Zohar Manna's 64th Birthday*, Lecture Notes in Computer Science. Springer-Verlag, 2003.

[DFP98] Rocco De Nicola, Gianluigi Ferrari, and Rosario Pugliese. KLAIM: A kernel language for agents in-

teraction and mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.

[DL02] Rocco De Nicola and Michele Loreti. A Modal Logic for Mobile Agents. *ACM Transactions on Computational Logic*, 2002. To appear. Available at `http://music.dsi.unifi.it/`.

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

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

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

[FGL$^+$96] Cedric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, and Didier Rémy. A calculus of mobile processes. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96: Concurrency Theory, 7th International Conference*, volume 1119 of

Lecture Notes in Computer Science, pages 406–421, Pisa, Italy, August 1996. Springer-Verlag.

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

[FMT01] Gianluigi Ferrari, Ugo Montanari, and Emilio Tuosto. A LTS semantics of ambients via graph synchronization with mobility. In *7th Italian Conference on Theoretical Computer Science – ICTCS'01*, volume 2202 of *LNCS*. Springer, 2001.

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

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

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

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

[Lor02] Michele Loreti. *Languages and Logics for Network Aware Programming*. PhD thesis, Università di Siena, 2002. Available at `http://music.dsi.unifi.it`.

[LS00] Francesca Levi and Davide Sangiorgi. Controlling interference in ambients. In *Conference Record of POPL'00: The 27th ACM SIGPLAN-SIGACT Sym-*

posium on Principles of Programming Languages, pages 352–364, Boston, Massachusetts, January 2000.

[Mob98] Mobile Man Project. Nfs tetherless t3 and beyond workshop. Interim report, November 1998.

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

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

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

[PV98] Joachim Parrow and Bjorn Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *13th Annual IEEE Symposium on Logic*

and Computer Science. IEEE Computer Society Press, 1998.

[Tuo04] Emilio Tuosto. Tarzan: Communicating and moving in wireless jungles. In *2nd Workshop on Quantitative Aspects of Programming Languages*, Electronic Notes in Computer Science. Elsevier, March 2004.

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