Synchronized Hyperedge Replacement: Synchronization Styles for Global Computing

Ugo Montanari
Dipartimento di Informatica
Università di Pisa

joint work with

Gian Luigi Ferrari (University of Pisa),
Dan Hirsch (Imperial College),
Ivan Lanese (University of Bologna) and
Emilio Tuosto (University of Leicester)

Outline

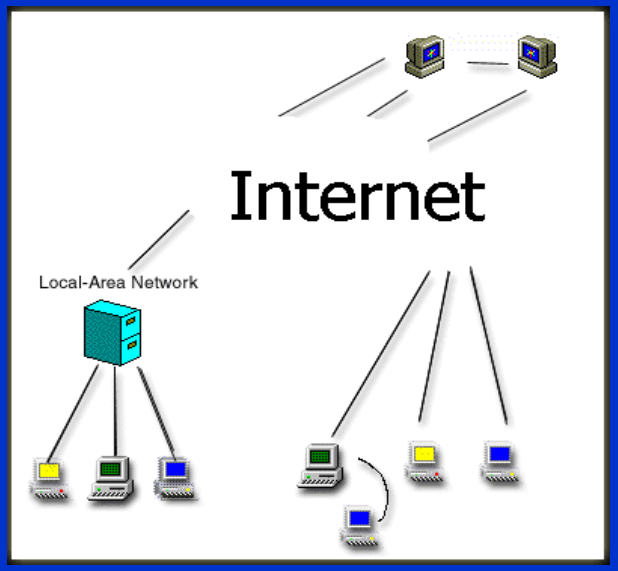
- Global computing
- Synchronized Hyperedge Replacement (SHR)
- Extensions: SAM-parametric SHR, constraint SHR
- Translation into logic programming
- Compositional abstract semantics
- The Fusion calculus
- The Ambient calculus
- UML system development
- Conclusions

Outline

- Global computing
- Synchronized Hyperedge Replacement (SHR)
- Extensions: SAM-parametric SHR, constraint SHR
- Translation into logic programming
- Compositional abstract semantics
- The Fusion calculus
- The Ambient calculus
- UML system development
- Conclusions

What is Global Computing?

- Essentially networks deployed on huge areas
- Global computing systems quite common nowadays
  - Internet, wireless communication networks, overlay networks …



Global Computing (I)

LAN

- client - server
- direct access
- sea of objects
- transparent
- friendly
- one administration
- protected

GC

- best effort communication
- unpredictable bandwidth
- different access policies
- broken by barriers & firewalls
- time outs
- independent administrations
- open to attacks

Global Computing (II)

- decentralized/distributed systems
- heterogeneous systems
- open systems

become dominant

it is not possible
to virtualize resources

Formal Methods for GC

- Building models of the system
- Old aims
  - Analyze the properties of the system before building it
  - Concentrate on a particular aspect
  - Abstract from details
- But new approaches/tools must be used
  - Mobility and non-functional requirements must be modeled explicitly
  - Need for compositionality
  - Need for more abstraction

A Strategy in Two Steps

- Graphical presentation of the network
  - Local graph transformations
  - Globlal constraint solving
  - Types for architectural styles
  - Subject reduction for reconfigurations
- More declarative programming
  - Declarative vs procedural programming
  - Exception handling insufficient for CSCW, etc.
  - SOS specifications for process calculi
  - Logical proof finding
  - Distributed constraint programming

Outline

- Global computing
- Synchronized Hyperedge Replacement (SHR)
- Extensions: SAM-parametric SHR, constraint SHR
- Translation into logic programming
- Compositional abstract semantics
- The Fusion calculus
- The Ambient calculus
- UML system development
- Conclusions

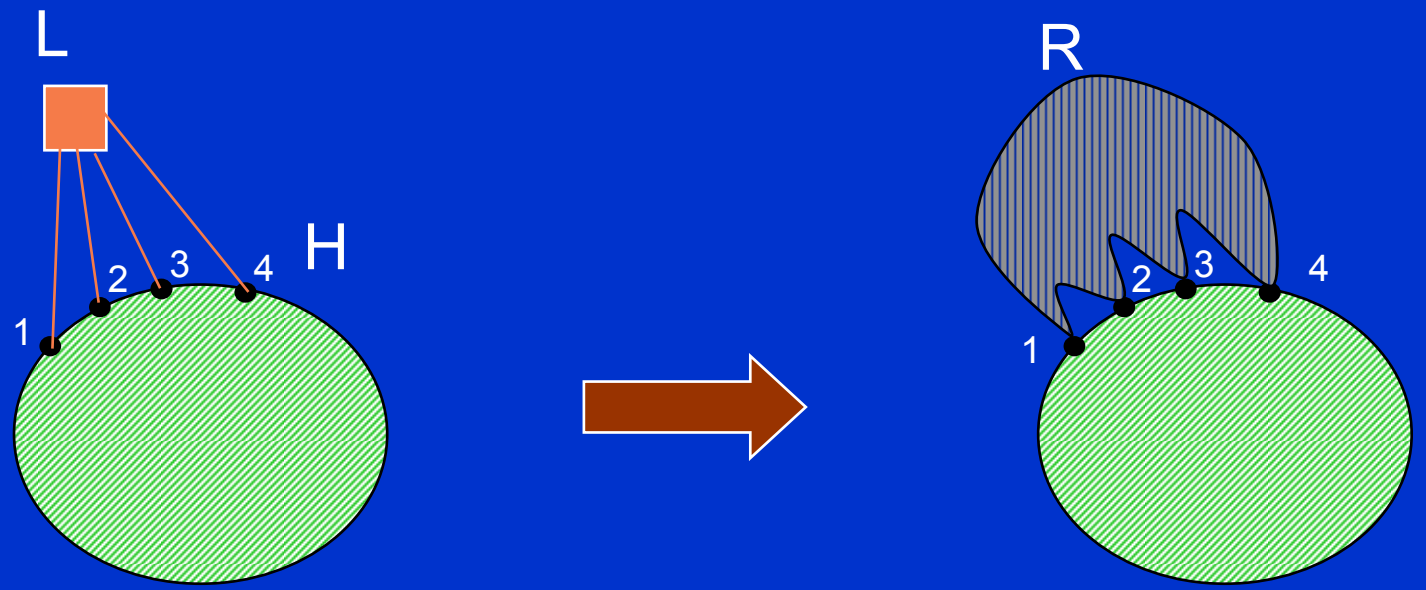
Graphical Approach to Distributed Systems

Motivations:

- Intuitive representation of distribution
- Natural concurrent semantics
- No need of structural axioms (as for process algebras)
- Existing modeling languages, e.g. UML
- Well-developed foundations

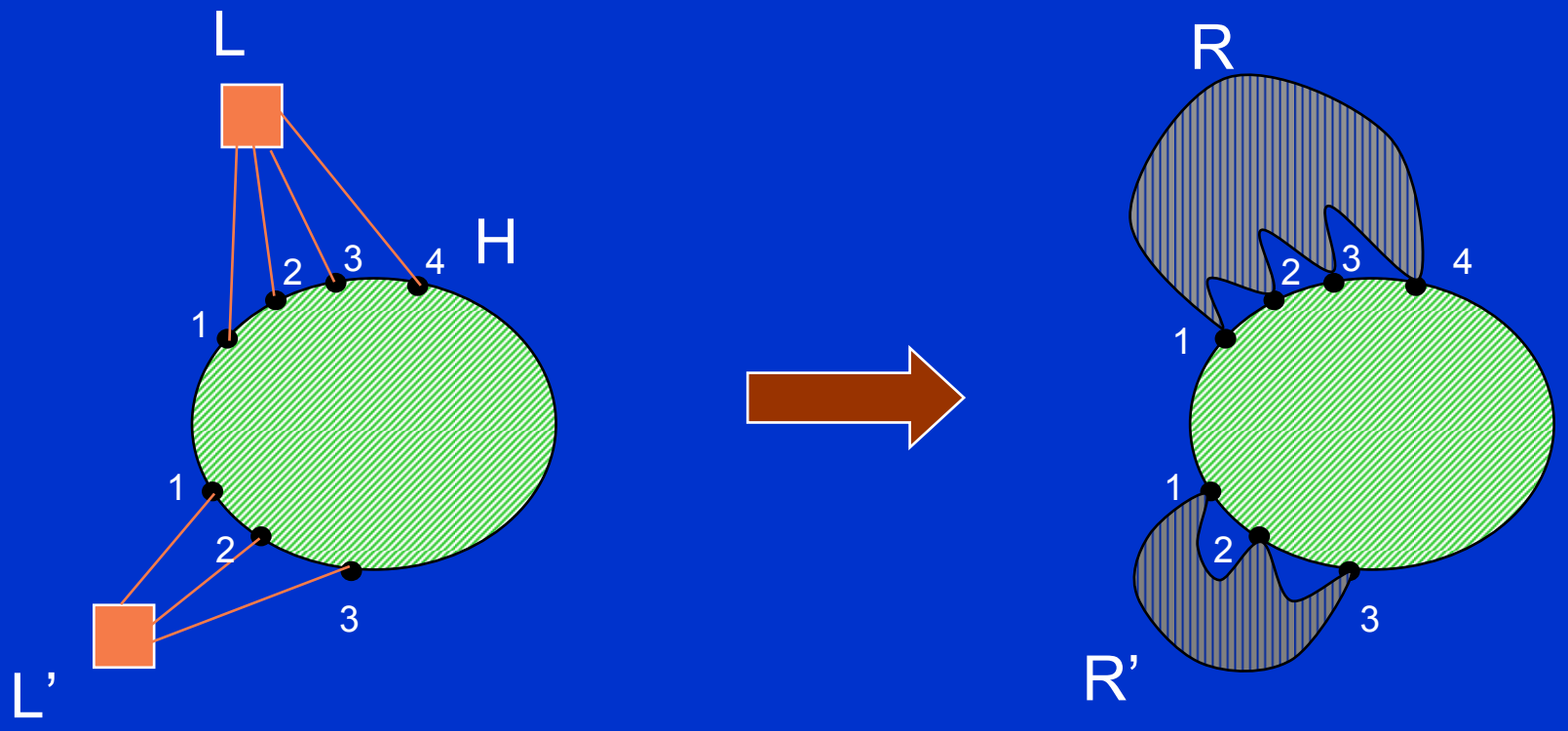
Edge Replacement Systems

- **Productions**: A context free production rewrites a single edge labeled by L into an arbitrary graph R. (Notation: L $\rightarrow$ R)



Edge Replacement Systems

- **Productions**: A context free production rewrites a single edge labeled by L into an arbitrary graph R. (Notation: L $\rightarrow$ R)



Rewritings of different edges can be executed concurrently

Synchronized Edge Replacement

- **Synchronized rewriting**: Actions are associated to nodes in productions. Each rewrite of an edge must match actions with (a number of) its adjacent edges and they have to move simultaneously

> How many edges synchronize depends on the synchronization policy

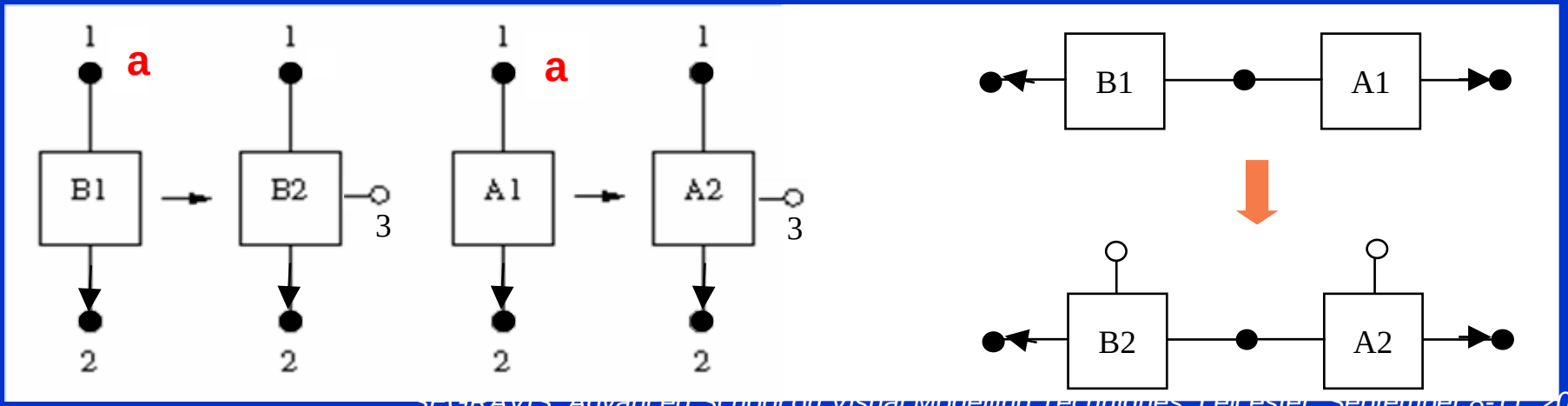
- Synchronized rewriting propagates synchronization all over the graph

Synchronized Edge Replacement

- **Hoare Synchronization**: All adjacent edges must produce the same action on the shared node

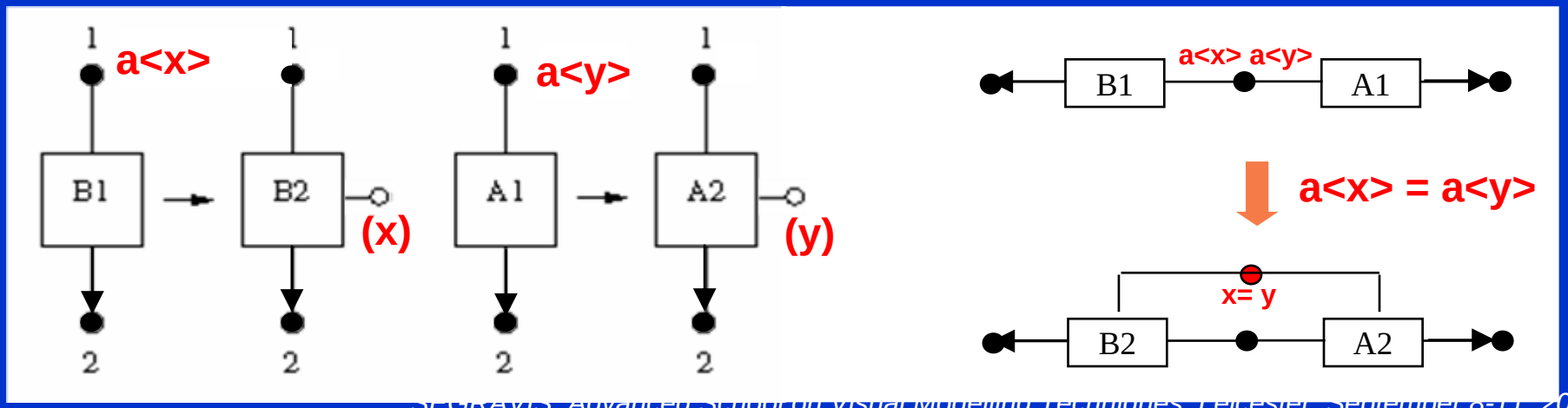
- **Milner Synchronization**: Only two of the adjacent edges synchronize by matching their complementary actions

Hoare synchronization



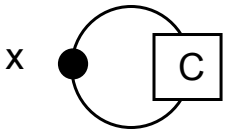
Adding Mobility

- Synchronized rewriting with name mobility
  - Allow declaration of new nodes in productions
  - Add to an action in a node a tuple of names that it wants to communicate
  - The synchronization step has to match actions and tuples
  - The declared names that were matched are used to merge the corresponding nodes

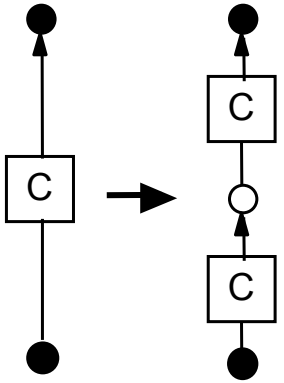


Example

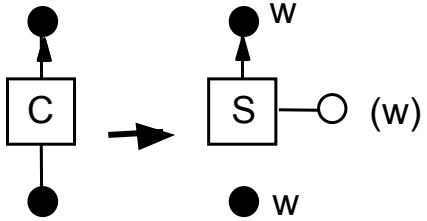
Initial Graph

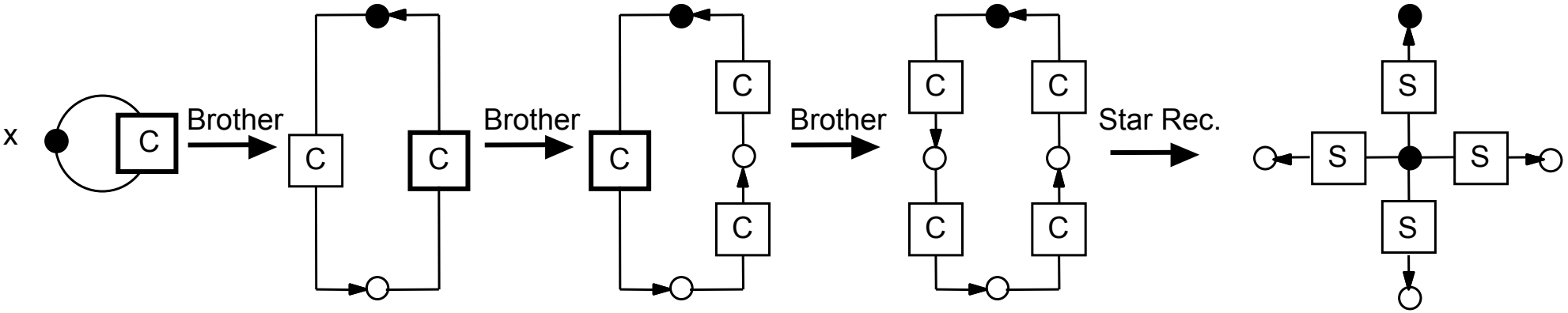


Brother:



Star Reconfiguration:



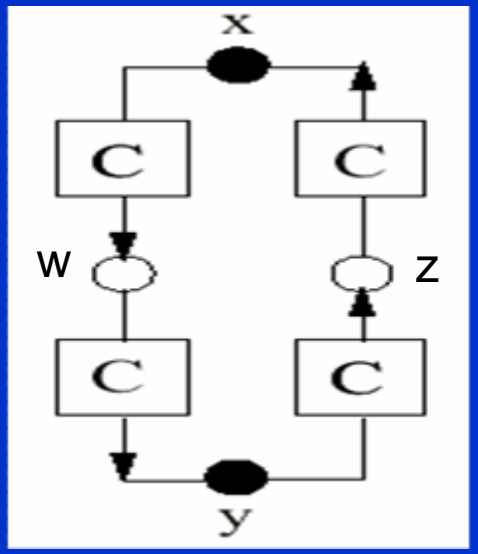


(1) **(2)** **(3)** **(4)** **(5)**

b)

A Notation For Graphs

Ring Example



$$x,y \vdash \nu\, z, w.\ C(x,w) \mid C(w,y) \mid C(y,z) \mid C(z,x)$$

Transitions as Judgements

Formalization of synchronized rewriting as judgements

- **Transitions**

$$\Gamma \vdash G_1 \xrightarrow{\bigwedge} \Gamma, \Delta \vdash G_2$$

$$\bigwedge : \Gamma \xrightarrow{o} (A \times N^*) \qquad (x, a, y) \in \bigwedge \text{ if } \bigwedge(x) = (a, y)$$

Δ is the set of new names that are used in synchronization

$$\Delta = \{z \mid \exists\, x.\ \bigwedge(x) = (a, y),\ z \notin \Gamma,\ z \in \text{set}(y)\}$$

Transitions as Judgements

Formalization of synchronized rewriting as judgements

- **Productions**

$$x_1,\ldots,x_n \vdash L(x_1,\ldots,x_n) \xrightarrow{\wedge} x_1,\ldots,x_n\ ,\ \Delta \vdash G$$

Free names can: i) be added to productions; and
ii) Identity productions are always available

- **Transitions**

are generated from the productions by applying the transition rules of the chosen synchronization mechanism

- **Derivations**

$$\Gamma_0 \vdash G_0 \xrightarrow{\wedge_1} \Gamma_1 \vdash G_1 \xrightarrow{\wedge_2} \ldots \xrightarrow{\wedge_n} \Gamma_n \vdash G_n$$

Adding Fusion

Synchronized rewriting

with Milner synchronization

with mobility and fusion

$$\Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \Gamma, \phi \vdash G_2$$

$\Lambda : \Gamma \xrightarrow{n} (A \times N^*)$ $\quad (x,a,\mathbf{y}) \in \Lambda$ if $\Lambda(x) = (a, \mathbf{y})$

$\pi: \Gamma \longrightarrow \Gamma$ and collapsing

$n(\Lambda) = \{ z \mid \exists z.\ \Lambda(x)=(a,\mathbf{y}),\ z \in Set(\mathbf{y}) \}$

$\Delta = n(\Lambda) - \Gamma$

$\phi = \pi(\Gamma) \approx \Delta$

Milner SHR with Mobility and Fusion, 1

$$\textit{(par-M)}\quad \frac{\Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \Phi \vdash G_2 \qquad \Gamma' \vdash G_1' \xrightarrow{\Lambda',\pi'} \Phi' \vdash G_2' \qquad (\Gamma \cup \Phi) \cap (\Gamma' \cup \Phi') = \emptyset}{\Gamma, \Gamma' \vdash G_1 | G_1' \xrightarrow{\Lambda \cup \Lambda', \pi \cup \pi'} \Phi, \Phi' \vdash G_2 | G_2'}$$

$$\textit{(merge-M)}\quad \frac{\Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \Phi \vdash G_2}{\Gamma\sigma \vdash G_1\sigma \xrightarrow{\Lambda',\pi'} \Phi' \vdash \nu U\ G_2\sigma\rho}$$

where $\sigma : \Gamma \to \Gamma$ *is an idempotent renaming and:*

1. *for all* $x, y \in \Gamma$ *such that* $x \neq y$, *if* $x\sigma = y\sigma \wedge \Lambda(x) \neq \varepsilon \wedge \Lambda(y) \neq \varepsilon$ *then*
   $(\forall z \in \Gamma \setminus \{x,y\}.z\sigma = x\sigma \Rightarrow \Lambda(z) = \varepsilon) \wedge \Lambda(x) = a \wedge \Lambda(y) = \overline{a} \wedge a \neq \tau$
2. $S_1 = \{\mathrm{n}_\Lambda(x) = \mathrm{n}_\Lambda(y) \mid x\sigma = y\sigma\}$
3. $S_2 = \{x = y \mid x\pi = y\pi\})$
4. $\rho = \mathrm{mgu}((S_1 \cup S_2)\sigma)$ *and* ρ *maps names to representatives in* $\Gamma\sigma$ *whenever possible*
5. $$\Lambda'(z) = \begin{cases} (\tau, \langle\rangle) & \text{if } x\sigma = y\sigma = z \wedge x \neq y \wedge \mathrm{act}_\Lambda(x) \neq \varepsilon \wedge \mathrm{act}_\Lambda(y) \neq \varepsilon \\ (\Lambda(x))\sigma\rho & \text{if } x\sigma = z \wedge \mathrm{act}_\Lambda(x) \neq \varepsilon \\ (\varepsilon, \langle\rangle) & \text{otherwise} \end{cases}$$
6. $\pi' = \rho\lfloor_{\Gamma\sigma}$
7. $U = (\Phi\sigma\rho) \setminus \Phi'$

Milner SHR with Mobility, 2

$$\textit{(res-M)} \quad \frac{\Gamma, x \vdash G_1 \xrightarrow{\Lambda,\pi} \Phi \vdash G_2}{\Gamma \vdash \nu x\, G_1 \xrightarrow{\Lambda\lfloor_\Gamma,\pi\lfloor_\Gamma} \Phi' \vdash \nu Z\, G_2}$$

where:

6. $(\exists y \in \Gamma . x\pi = y\pi) \Rightarrow x\pi \neq x$
7. $\mathrm{act}_\Lambda(x) = \varepsilon \vee \mathrm{act}_\Lambda(x) = \tau$
8. $Z = \{x\}$ *if* $x \notin \mathrm{n}(\Lambda\lfloor_\Gamma), Z = \emptyset$ *otherwise*

$$\textit{(new-M)} \quad \frac{\Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \Phi \vdash G_2 \qquad x \notin \Gamma \cup \Phi}{\Gamma, x \vdash G_1 \xrightarrow{\Lambda \cup \{(x,\varepsilon,\langle\rangle)\},\pi} \Phi, x \vdash G_2}$$

Outline

- Global computing
- Synchronized Hyperedge Replacement (SHR)
- Extensions: SAM-parametric SHR, constraint SHR
- Translation into logic programming
- Compositional abstract semantics
- The Fusion calculus
- The Ambient calculus
- UML system development
- Conclusions

Milner vs Hoare



- Surprisingly the most difficult step
- Expressiveness as sets of reconfigurations that can be specified
- Simulating Hoare using Milner
  - Must implement n-ary synchronization using binary synchronization
- Simulating Milner using Hoare
  - Milner synchronization is asymmetric
  - Milner restriction affects the behaviour, Hoare restriction just the observation

Some results

- Not equivalent in general
- In closed 2-shared graphs Milner is more powerful than Hoare
  - Hoare implemented by dropping the distinction between actions and coactions
- A translation of graphs can be used to bridge the gap in many cases
  - Amoeboids to simulate synchronization
- Hoare amoeboids are broadcasters
- Milner amoeboids are routers
  - Mutual exclusion can not be enforced

Parametric SHR

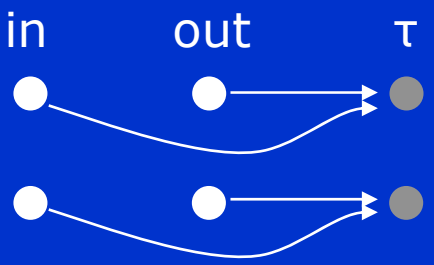
- A member of SHR family
- Synchronization and mobility patterns not fixed but user-definable
  - Specified with Synchronization Algebras with Mobility (SAMs)
- Allows to use each time the most suitable synchronization primitives
- Heterogeneous SHR: different SAMs in the same system
- Constraint SHR: not only fusions but also constraint composition

Synchronization Algebras with Mobility

- Specify how actions synchronize
  - Two at the time, associativity and commutativity required
- From Winskel's synchronization algebras
  - Partial operator ● for action synchronization
  - Action ε for "not taking part to the synchronization"
- Added
  - Arities of actions
  - Function from parameters of the synchronizing actions to parameters of the result
  - Set of final actions

Milner SAM

- Normal actions, coactions, τ, ε
- in ● out = τ

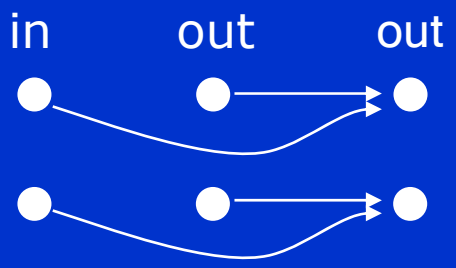


- a ● ε = a

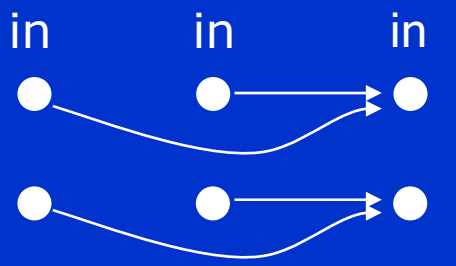
- Final actions: τ, ε

Broadcast SAM

- Normal actions, coactions, ε
- in • out = out



- in • in = in



- ε • ε = ε
- Final actions: out, ε

And many more

- SAMs can be defined for many synchronization policies
  - Mutual exclusion
  - Priority synchronization
  - …
- SAMs can be combined
  - e.g. Milner synchronization and broadcasting

Outline

- Global computing
- Synchronized Hyperedge Replacement (SHR)
- Extensions: SAM-parametric SHR, constraint SHR
- Translation into logic programming
- Compositional abstract semantics
- The Fusion calculus
- The Ambient calculus
- UML system development
- Conclusions

Logic Programming

- Traditionally language for AI and problem solving
- In the GC scenario seen as goal rewriting framework
- Unification as synchronization primitive
- Focus on partial computations

Hoare SHR vs Logic Programming

- Hoare synchronization strictly related to unification
- Strong relation between Hoare SHR and Synchronized Logic Programming
  - A subset of logic programming
  - Transactional application of many clauses
  - Exploits function symbols for synchronization

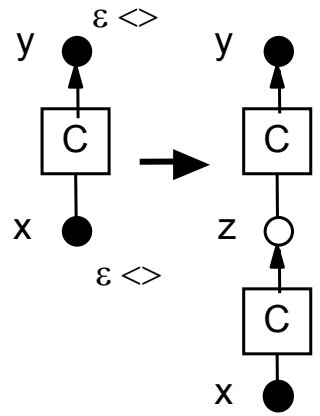
Summary of the Comparison

Hoare SHR	SLP
Graph	Goal
Hyperedge	Atom
Node	Variable
Parallel comp.	AND comp.
Action	Function sym.
Production	Clause
Transition	Transaction

Main Results

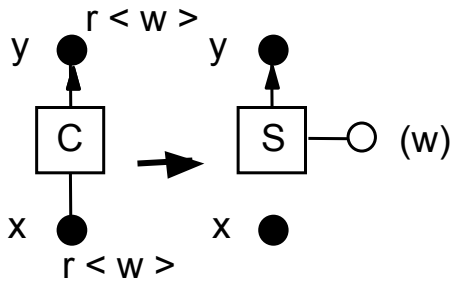
- Simple (homomorphic) mapping from Hoare SHR to SLP
- Complete correspondance
- Suggests how to introduce restriction in logic programming

The Ring-Star Example, I



$$C(x,y) \leftarrow C(x,z), C(z,y)$$

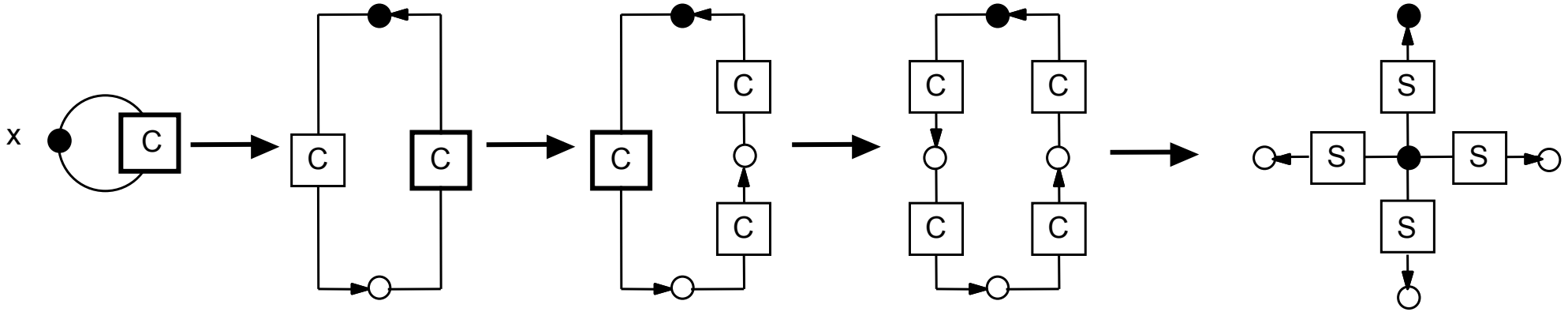
x, y⊢ C(x,y) $\xrightarrow{\{(x,\varepsilon,<>),\ (y,\varepsilon,<>)\}}$ x, y⊢ νz.C(x,z) | C(z,y)



$$C(r(x,w), r(y,w)) \leftarrow S(y,w)$$

x, y⊢ C(x,y) $\xrightarrow{\{(x,r,<w>),\ (y,r,<w>)\}}$ x, y⊢ S(w,y)

The Ring-Star Example, II



Variant	*Unifier*
$\leftarrow C(x,x)$	
$C(x_1,x_1') \leftarrow C(x_1,y_1), C(y_1,x_1')$	$\{x_1/x, x_1/x_1'\}$
$\leftarrow C(x_1,y_1), C(y_1,x_1)$	
$C(y_2,x_2) \leftarrow C(y_2,z_2), C(z_2,x_2)$	$\{y_2/y_1, x_2/x_1\}$
$\leftarrow C(x_2,y_2), C(y_2,z_2), C(z_2,x_2)$	
$C(z_3,x_3) \leftarrow C(z_3,v_3), C(v_3,x_3)$	$\{z_3/z_2, x_3/x_2\}$
$\leftarrow C(x_3,y_2), C(y_2,z_3), C(z_3,v_3), C(v_3,x_3)$	
$C(r(x_4,w),r(y_4,w)) \leftarrow S(y_4,w)$	$\{r(x_4,w)/x_3, r(y_4,w)/y_2\}$
$\leftarrow S(y_4,w), C(r(y_4,w),z_3), C(z_3,v_3), C(v_3,r(x_4,w))$	
$C(r(y_5,w),r(z_5,w)) \leftarrow S(z_5,w)$	$\{y_5/y_4, r(z_5,w)/z_3\}$
$\leftarrow S(y_5,w), S(z_5,w), C(r(z_5,w),v_3), C(v_3,r(x_4,w))$	
$C(r(z_6,w),r(v_6,w)) \leftarrow S(v_6,w)$	$\{z_6/z_5, r(v_6,w)/v_3\}$
$\leftarrow S(y_5,w), S(z_6,w), S(v_6,w), C(r(v_6,w),r(x_4,w))$	
$C(r(v_7,w),r(x_7,w)) \leftarrow S(x_7,w)$	$\{v_7/v_6, x_7/x_4\}$
$\leftarrow S(y_5,w), S(z_6,w), S(v_7,w), S(x_7,w)$	

Outline

- Global computing
- Synchronized Hyperedge Replacement (SHR)
- Extensions: SAM-parametric SHR, constraint SHR
- Translation into logic programming
- Compositional abstract semantics
- The Fusion calculus
- The Ambient calculus
- UML system development
- Conclusions

Observational Semantics and Compositionality

- Allows to have an abstract description of system behaviour
- Compositionality useful to
  - Compute abstract behavior of the system from the behavior of the components
  - Compute the behavior of a system when plugged in its execution environment
- Bisimilarity is a standard tool
- Bisimilarity is a congruence is a key property

Abstract Semantics for Parametric SHR

- Bisimulation can be defined in a standard way for SHR
- Under reasonable conditions on the SAM bisimilarity is a congruence for parametric SHR
  - Milner, Hoare and many others satisfy the conditions
- Proof exploits bialgebraic techniques

Congruence Results for Fusion Calculus

- Bisimilarity is not a congruence for Fusion Calculus (not closed under substitutions)
- The comparison with SHR shows why congruence fails and suggests how to solve the problem
- We have proposed a new concurrent semantics which is compositional

The Idea of the Semantics

- Allowing many actions in the same transition but on different channels
  - Process a|b can execute a and b concurrently going to 0 (but can also execute either a or b)
  - Process a|a is bisimilar to a.a
  - Process $\bar{a}|a|b$ can perform τ and b concurrently going to 0
- Allows to observe the degree of parallelism of a process

Congruence Properties

- $a.\overline{b} + \overline{b}.a \approx a \mid \overline{b}$ no more a counterexample since the two terms are not bisimilar
- Observing where a synchronization is performed becomes important
  - Otherwise congruence non preserved by context a|[-]
  - Actions aτ in addition to normal τ

Outline

- Global computing
- Synchronized Hyperedge Replacement (SHR)
- Extensions: SAM-parametric SHR, constraint SHR
- Translation into logic programming
- Compositional abstract semantics
- The Fusion calculus
- The Ambient calculus
- UML system development
- Conclusions

Fusion Calculus

- Calculus for mobility inspired by π-calculus
- Symmetric input/output
- Arbitrary fusions allowed
- Can simulate π-calculus

Milner SHR vs Fusion Calculus

- Many common features
  - Synchronization in Milner style
  - Mobility using fusions
  - LTS semantics
- Straightforward mapping of Fusion into Milner SHR
- SHR adds:
  - Graphical presentation
  - Multiple synchronizations
  - Concurrent semantics

Summary of the comparison

Fusion	Milner SHR
Processes	Graphs
Sequential processes	Hyperedges
Names	Nodes
Parallel comp.	Parallel comp.
Scope	Restriction
Prefixes	Productions
Transitions	Interleaving tr.

Main Results

- Simple (homomorphic) mapping
- Complete correspondance
- Suggests many generalizations of Fusion
  - A concurrent semantics
  - PRISMA Calculus

Outline

- Global computing
- Synchronized Hyperedge Replacement (SHR)
- Extensions: SAM-parametric SHR, constraint SHR
- Translation into logic programming
- Compositional abstract semantics
- The Fusion calculus
- The Ambient calculus
- UML system development
- Conclusions

The Ambient Calculus, I (Cardelli & Gordon)

Syntax

M ::= in n | out n | open n

P,Q ::= 0 | n[P] | M.P | P|Q | rec X.P | X

Structural Equivalence

- _|_ is associative, commutative and 0 is its identity
- rec X.P is α-convertible

The Ambient Calculus, II

Reduction semantics

$m[n[\text{out } m.P \mid Q] \mid R] \longrightarrow n[P \mid Q] \mid m[R]$

$n[\text{in } m.P \mid Q] \mid m[R] \longrightarrow m[n[P \mid Q] \mid R]$

$\text{open } n.P \mid n[Q] \longrightarrow P \mid Q$

$$\frac{P \longrightarrow Q}{P \mid R \longrightarrow Q \mid R} \qquad \frac{P \longrightarrow Q}{n[P] \longrightarrow n[Q]}$$

Modeling the AC in SHR

From ambient terms to ambient graphs

$$[0]_x = x \vdash \text{nil}$$

$$[n[P]]_x = x \vdash \nu y.\ G \mid n(y,x)) \quad \text{if } y \neq x \text{ and } [P]_y = y \vdash G$$

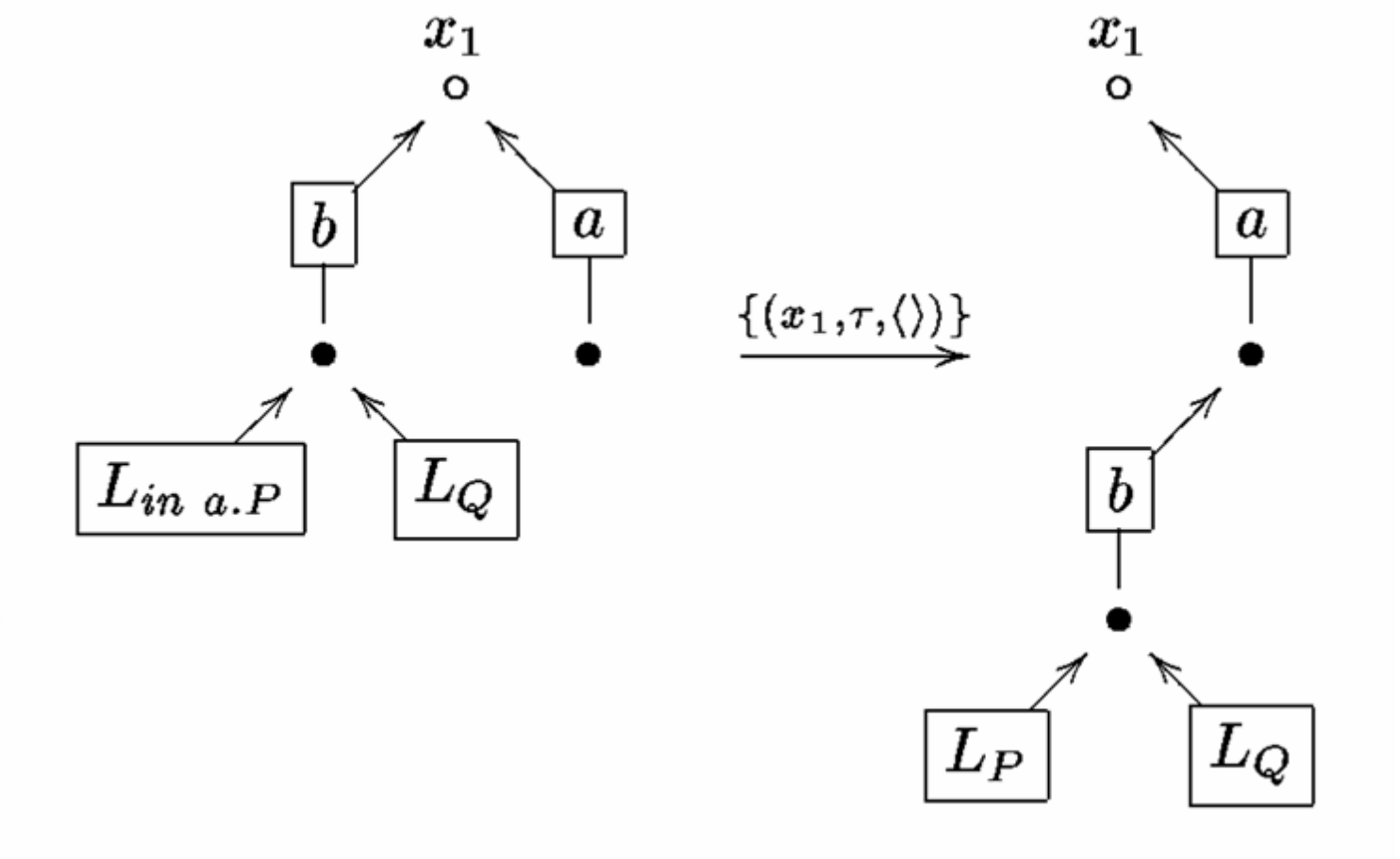
$$[M.P]_x = x \vdash L_{M.P}(x)$$

$$[P_1|P_2]_x = x \vdash G_1 \mid G_2 \quad \text{if } [P_i]_x = y \vdash G_i \quad i = 1,2$$

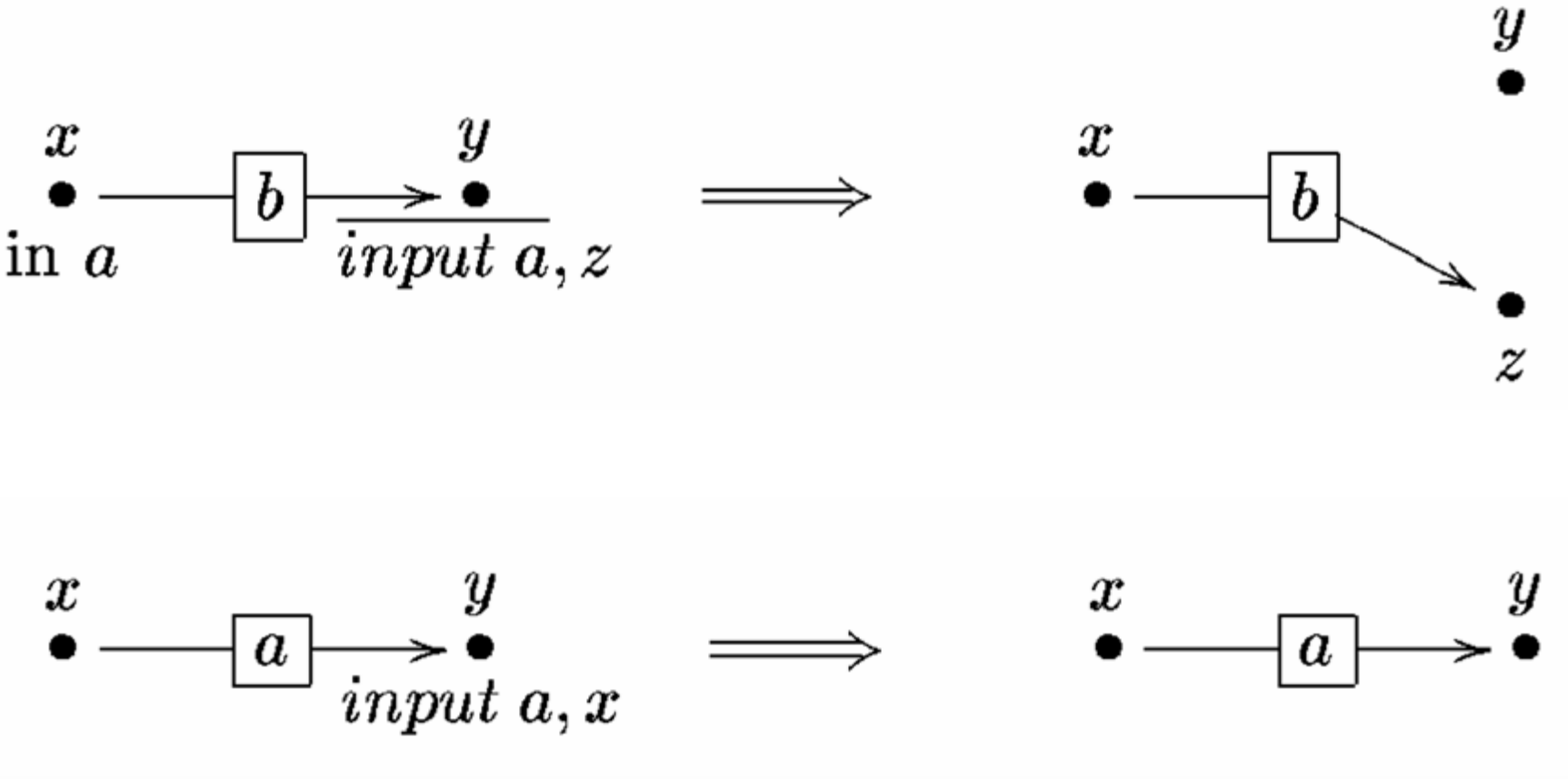
$$[\text{rec } X.P]_x = [P[\text{rec } X.\ P/X]]_x$$

Example, I

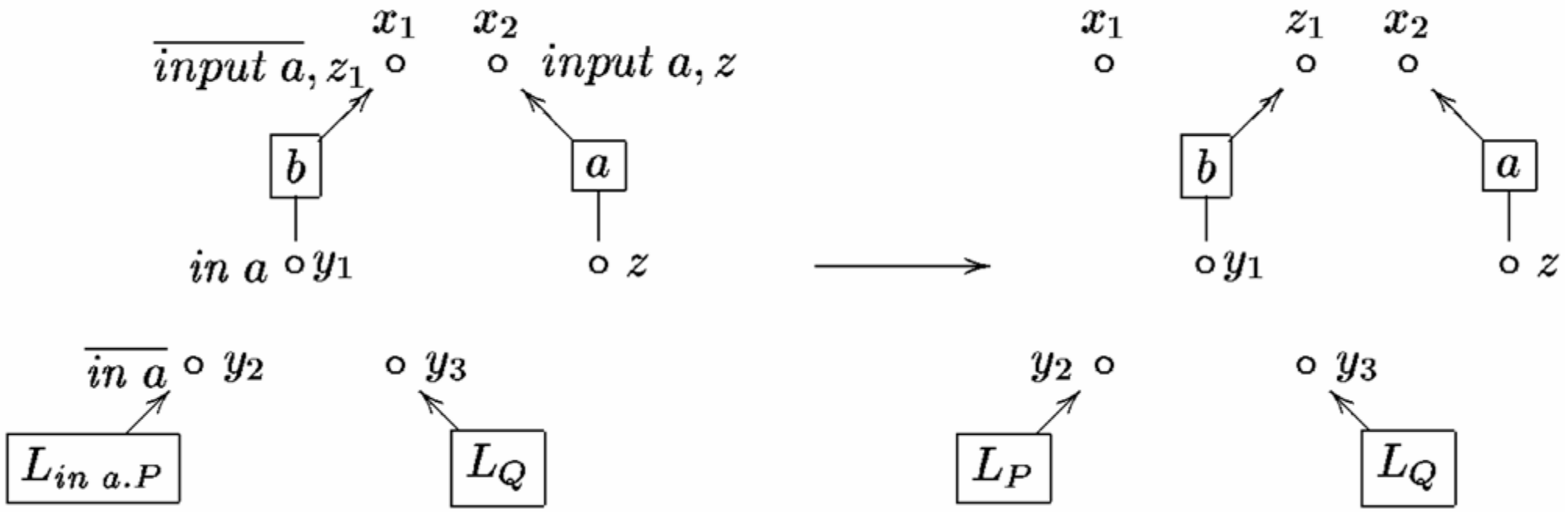
$$b[in\ a.P \mid Q] \mid a[0] \rightarrow a[b[P \mid Q]]$$



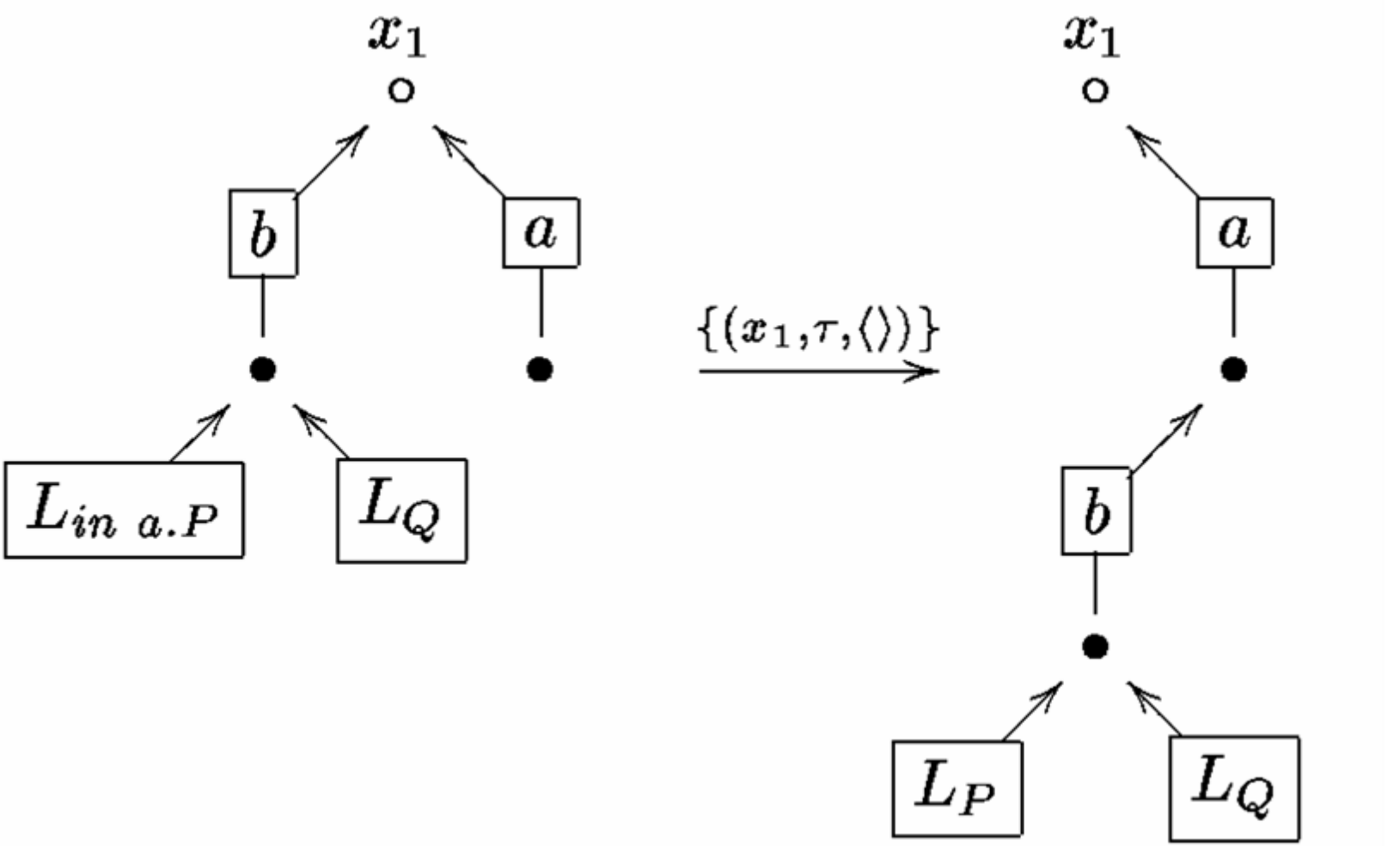
Example, II



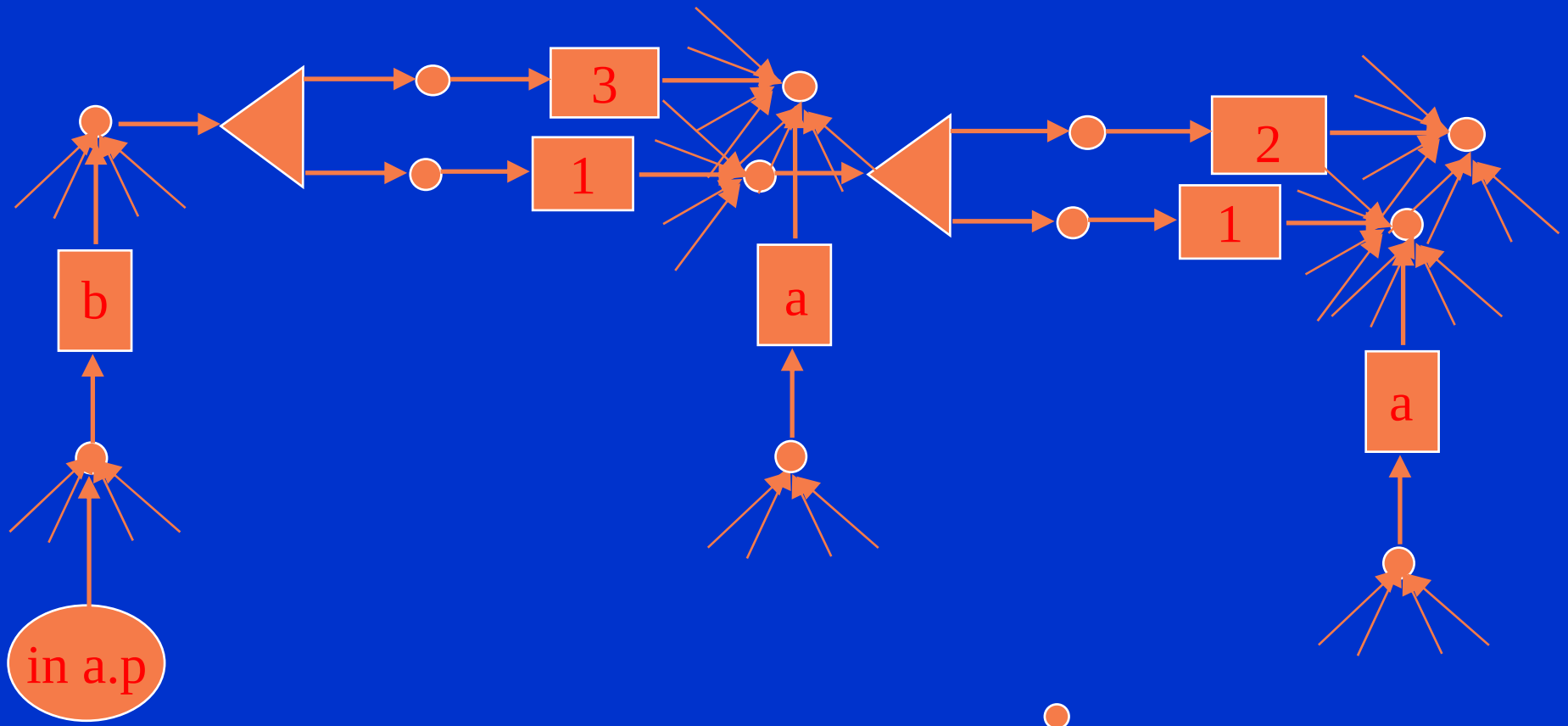
Example, III

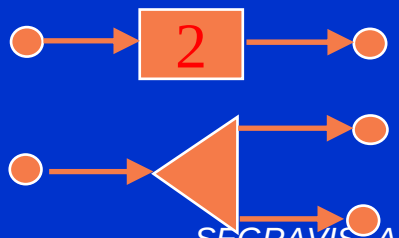

$\overline{input\ a}, z_1$
x_1
x_2
$input\ a, z$
b
a
$in\ a$
y_1
z
$\overline{in\ a}$
y_2
y_3
$L_{in\ a.P}$
L_Q
x_1
z_1
x_2
b
a
y_1
z
y_2
y_3
L_P
L_Q

Example, IV



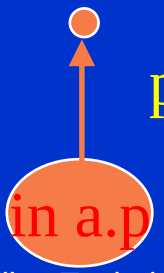
Minimizing Routing Cost, I





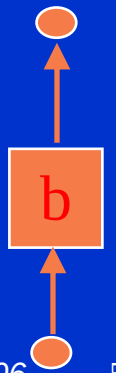
network link

router

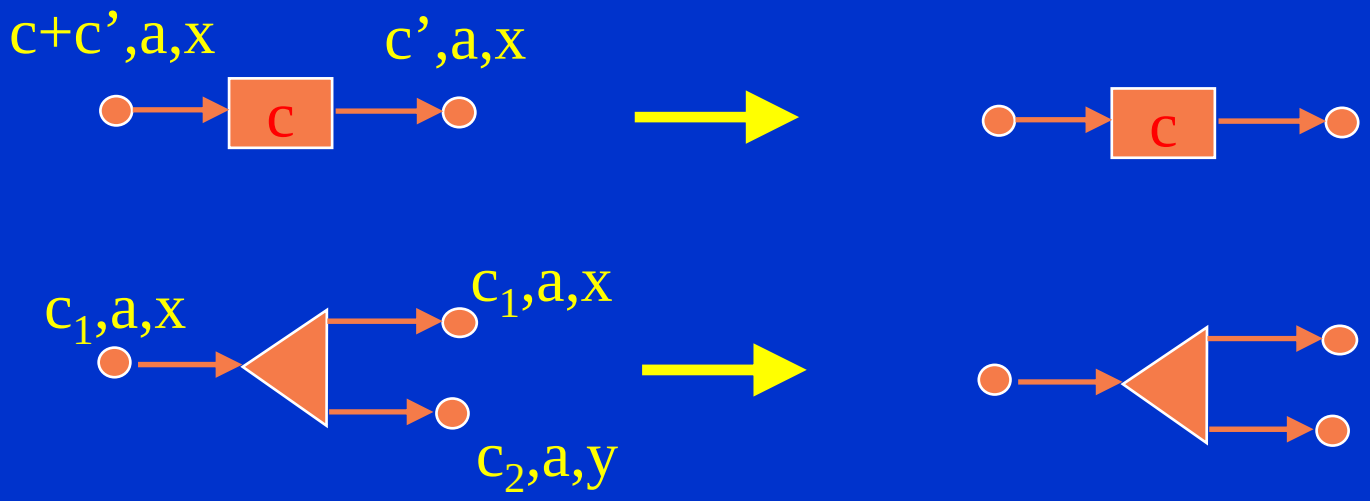


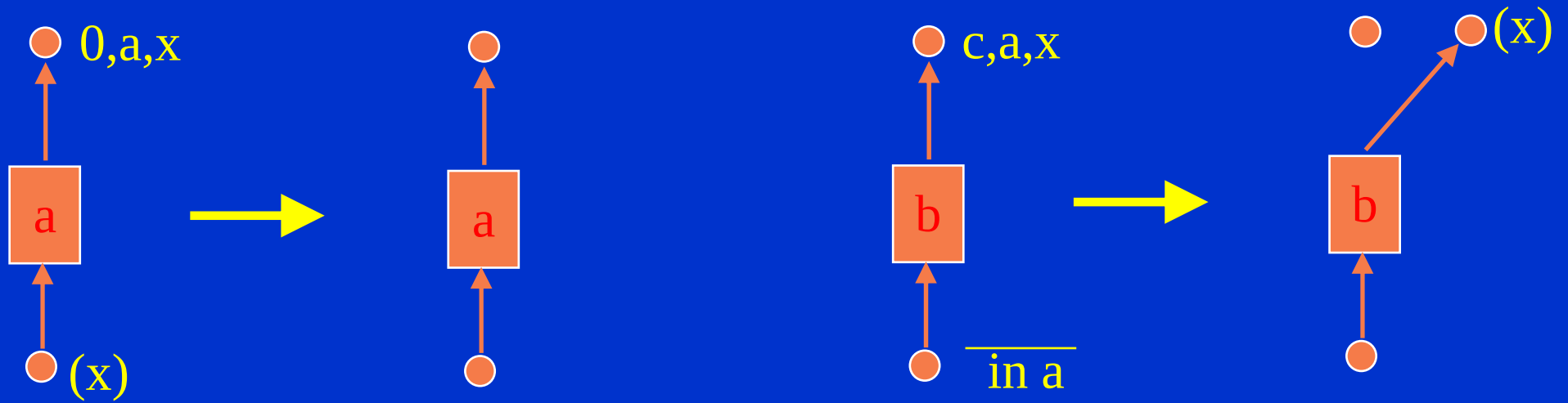
process

ambient



Minimizing Routing Cost, II





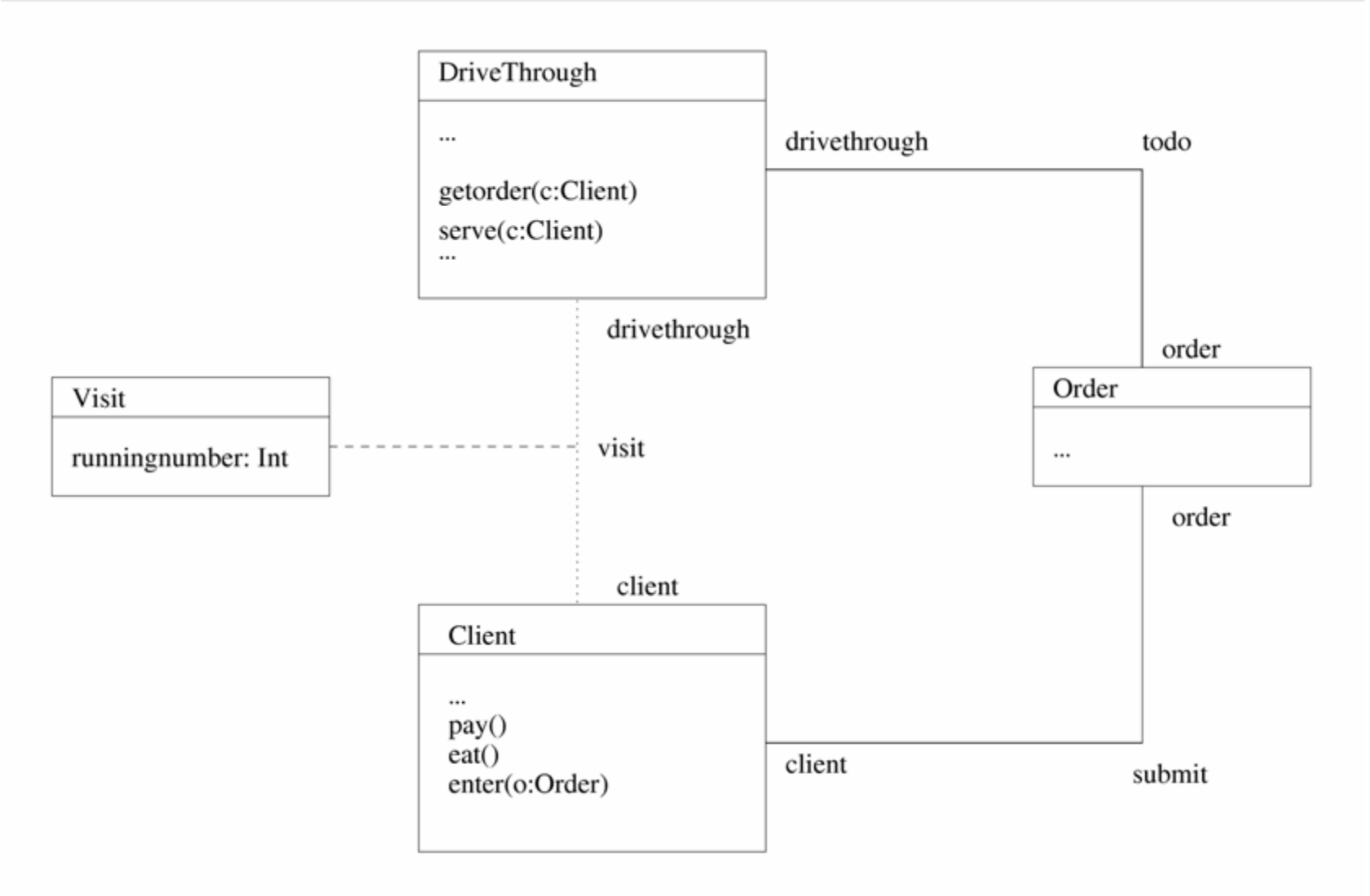
Outline

- Global computing
- Synchronized Hyperedge Replacement (SHR)
- Extensions: SAM-parametric SHR, constraint SHR
- Translation into logic programming
- Compositional abstract semantics
- The Fusion calculus
- The Ambient calculus
- UML system development
- Conclusions

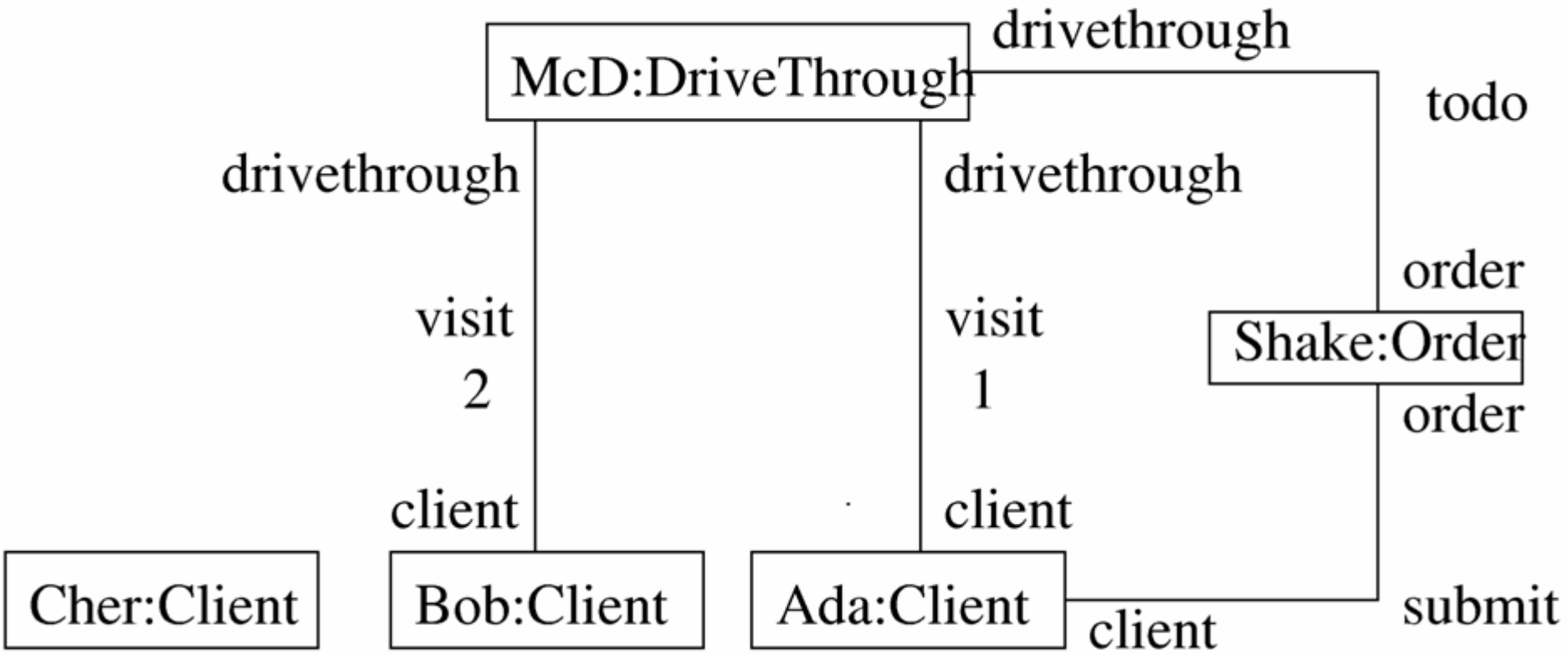
UML System Development

- Graph transformation semantics of UML [Kuske et al., IFM'02]
- The drive-through example
- Synchronized graph rewriting
- Explicit synchronization among components which have to be transformed during reconfigurations

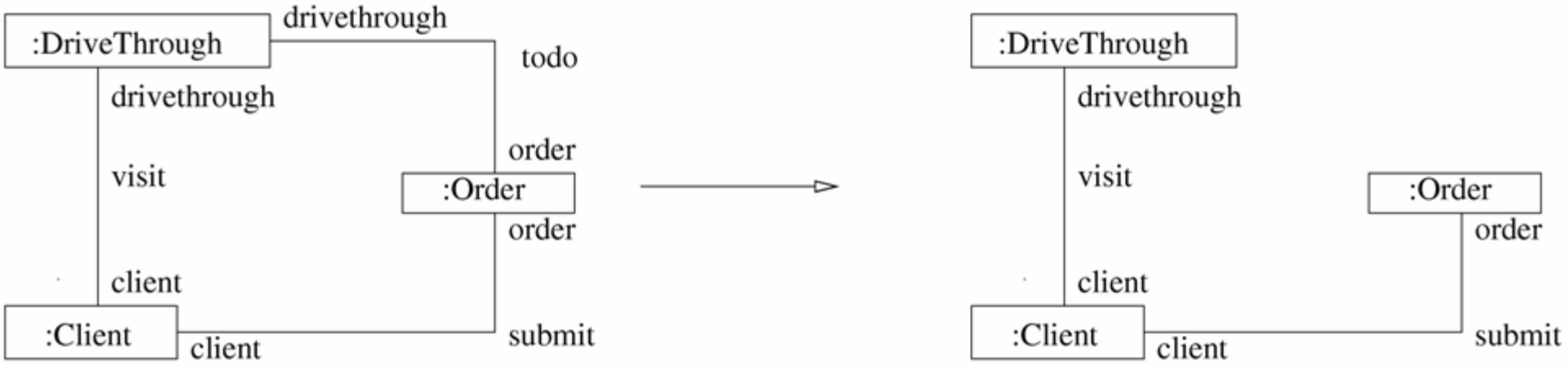
Drive Through: Class Diagram



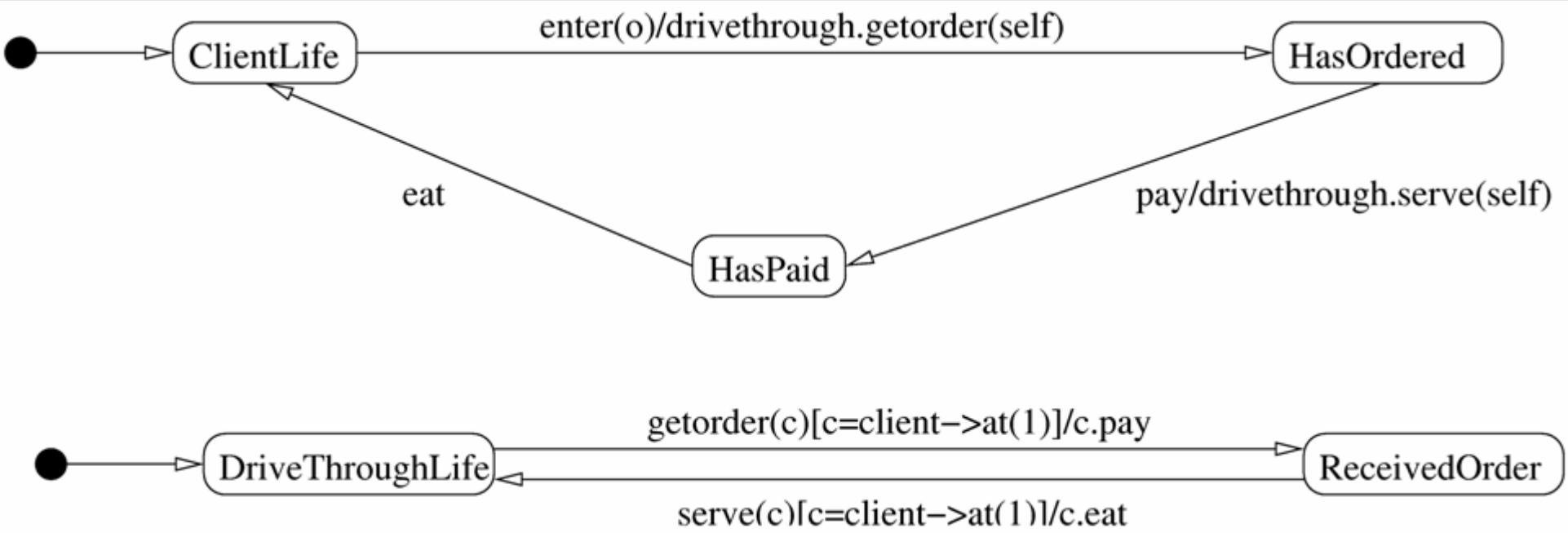
Drive Through: Object Diagram



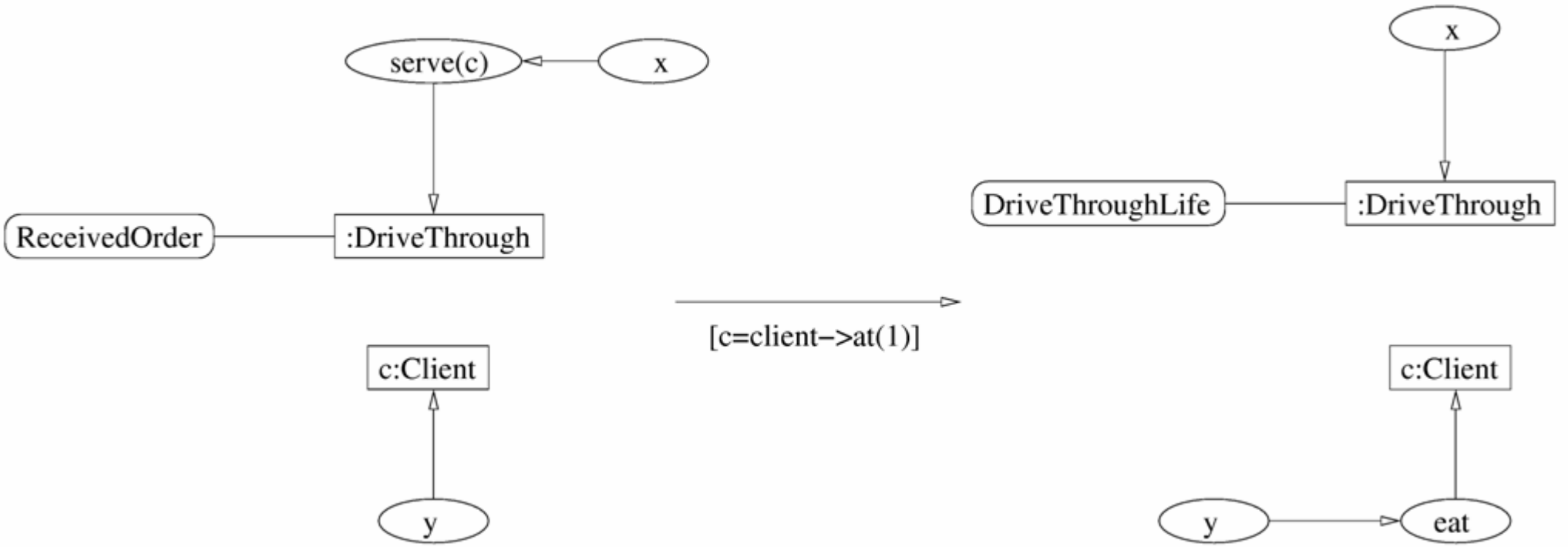
Drive Through: *Serve* Operation



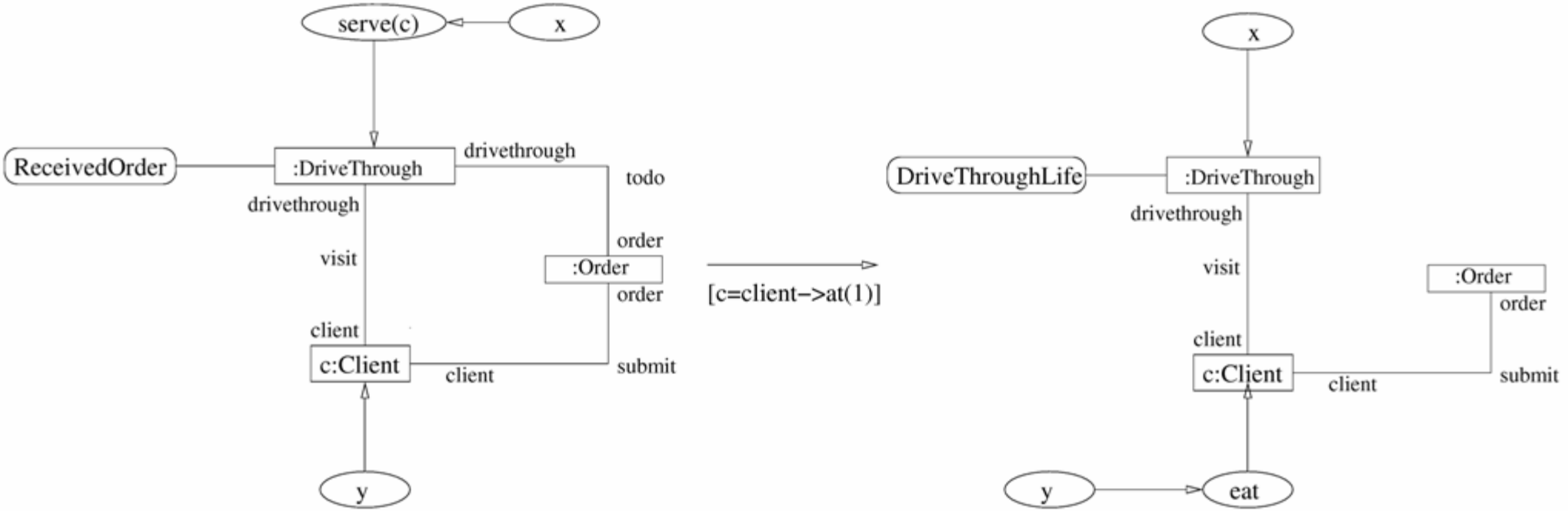
Drive Through: State Diagrams



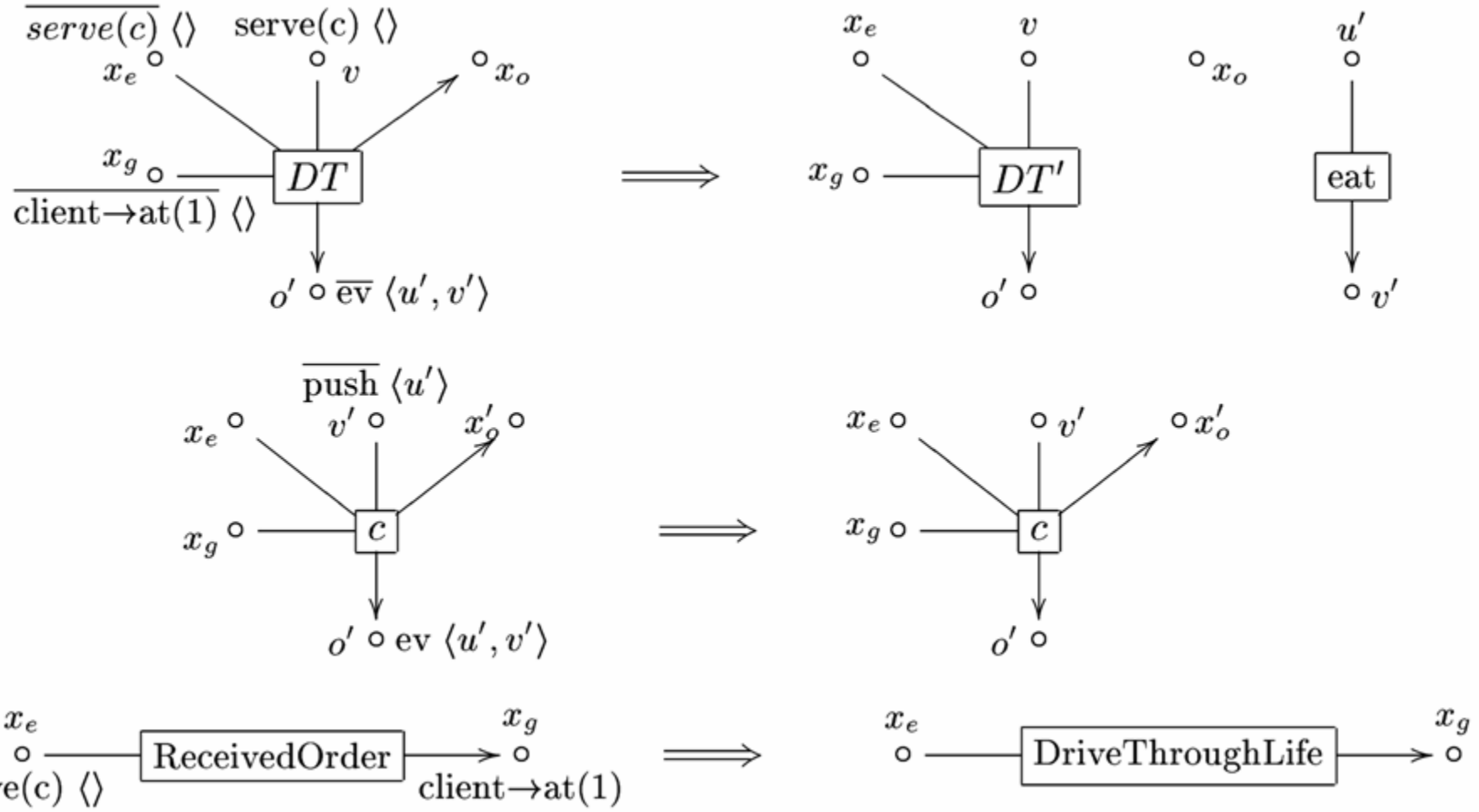
Drive Through: Transition Rule for *Serve*



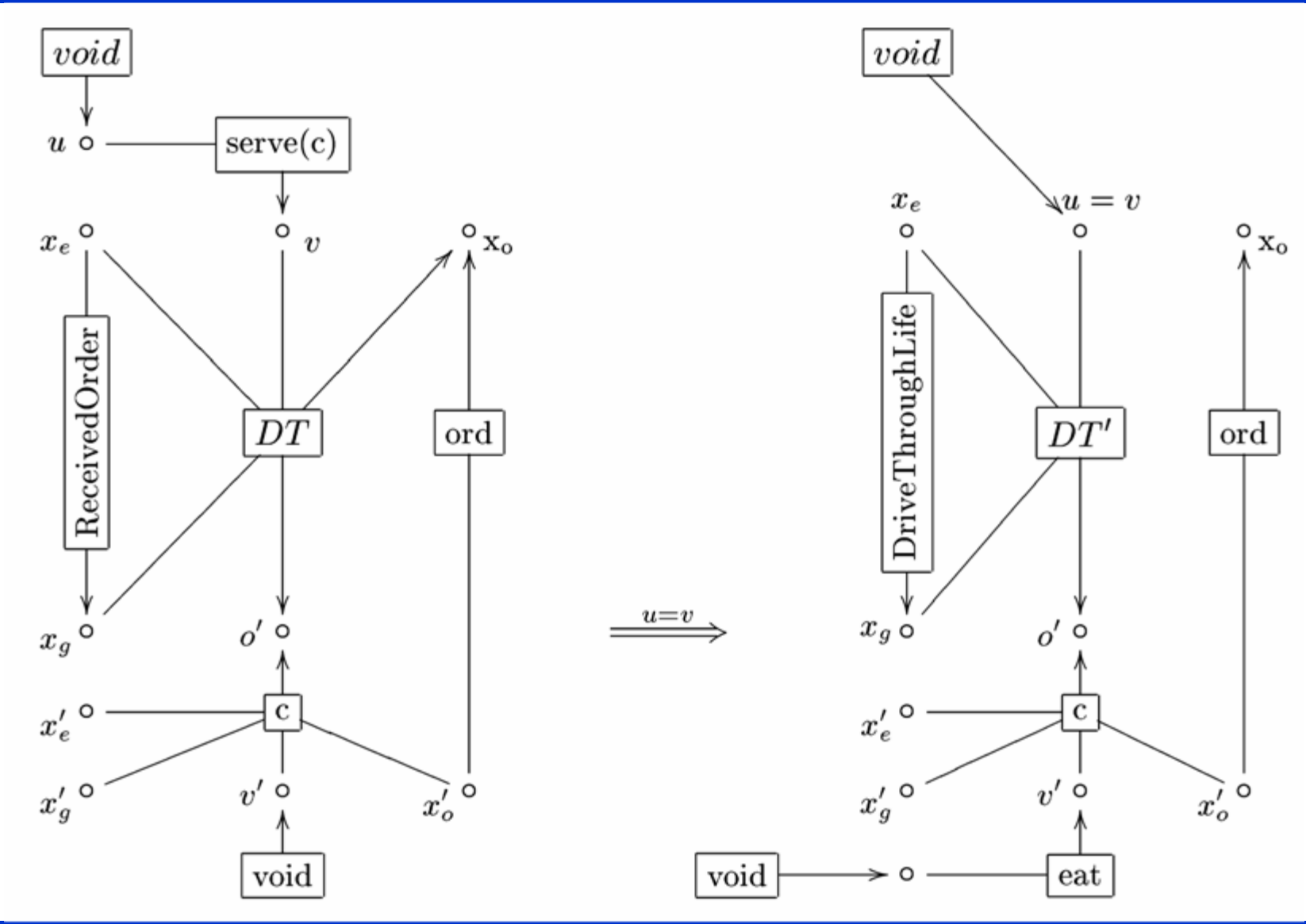
Drive Through: Integrated Rule for *Serve*



Drive Through: Synchronized Productions



Drive Through: A Transition for *Serve*



Outline

- Global computing
- Synchronized Hyperedge Replacement (SHR)
- Extensions: SAM-parametric SHR, constraint SHR
- Translation into logic programming
- Compositional abstract semantics
- The Fusion calculus
- The Ambient calculus
- UML system development
- Conclusions

Conclusions and Future Work

- Synchronized edge replacement for modeling network aware programming
  - graphs rather than terms/trees
  - agents synchronizing at their locations with different synchr. algebras
  - several locations, several agents involved in synchronizations
- Easy modeling/comparison of several formal systems
  - process algebras, Milner-Hoare, fusion calculus
  - logic programming
  - Ambient calculus
- Model-driven development: case studies in Agile, Sensoria
- QoS via constraint semirings