Bigraphs: a model for mobile agents

Robin Milner, September 2008

I	How agents are linked and placed independently
II	How to build complex systems from simple ones
III	Dynamical theory, illustrated for CCS
IV	Stochastic dynamics, e.g. for membrane budding
V	Foundation for behavioural equivalence
VI	Ubiquitous systems: a context for bigraphs

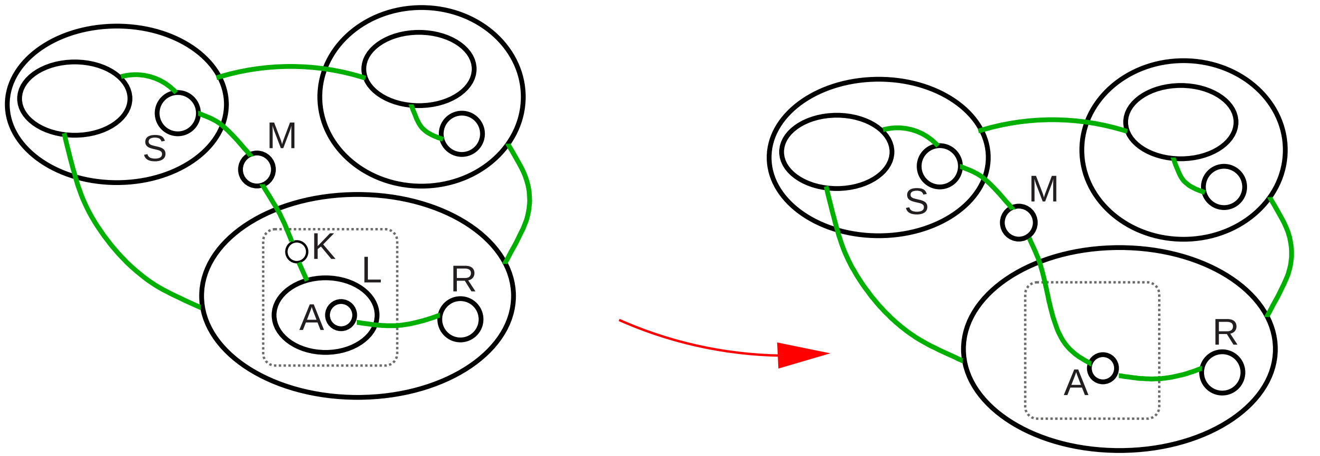
Acknowledging contributions from

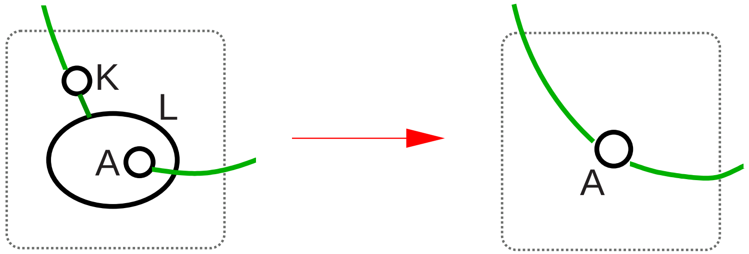
James Leifer and **Ole Høgh Jensen**

Lecture I

How agents are linked and placed independently

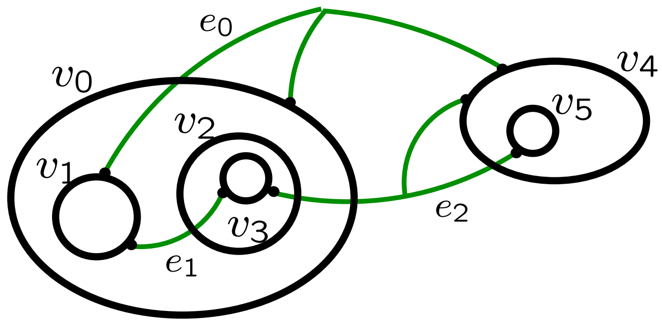
A fanciful system



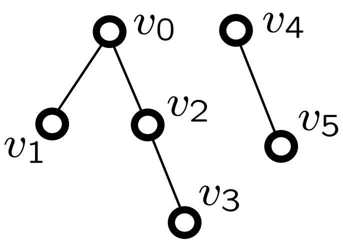
Reaction rule: 

The bi-structure of bigraphs

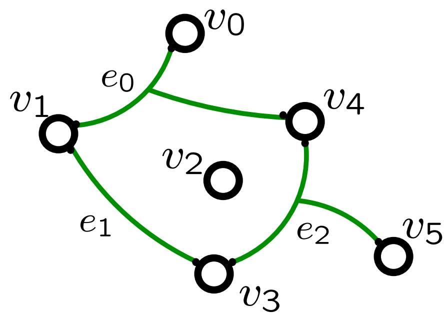
a bare bigraph G



its forest

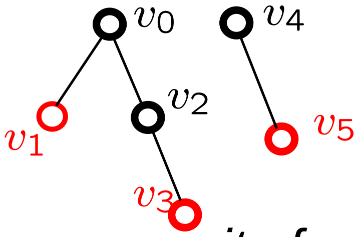


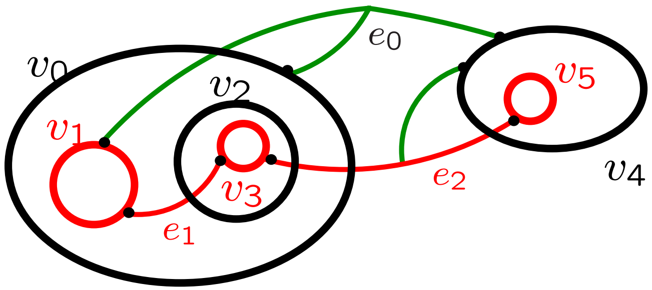
its hypergraph



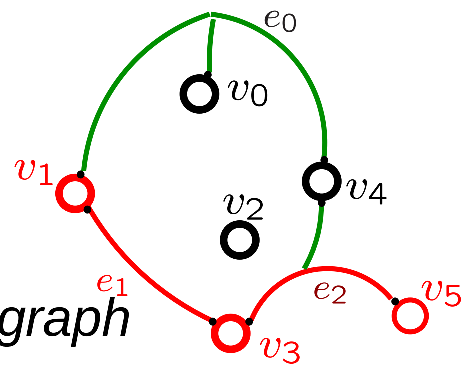
How to build bigraphs? *Give them interfaces . . .*

bare bigraph $\bar{G}$



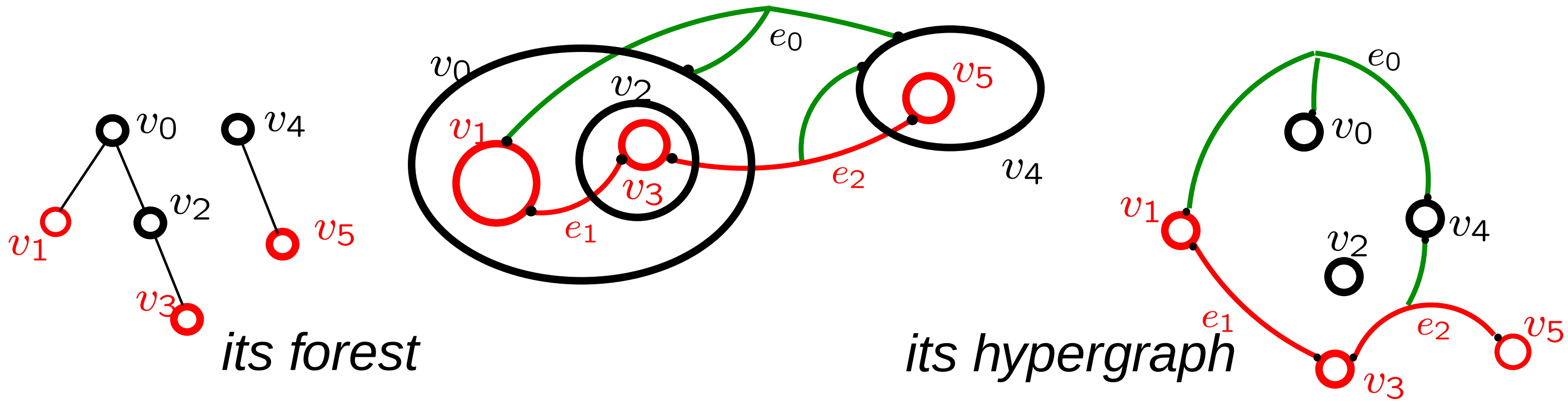


its forest



its hypergraph

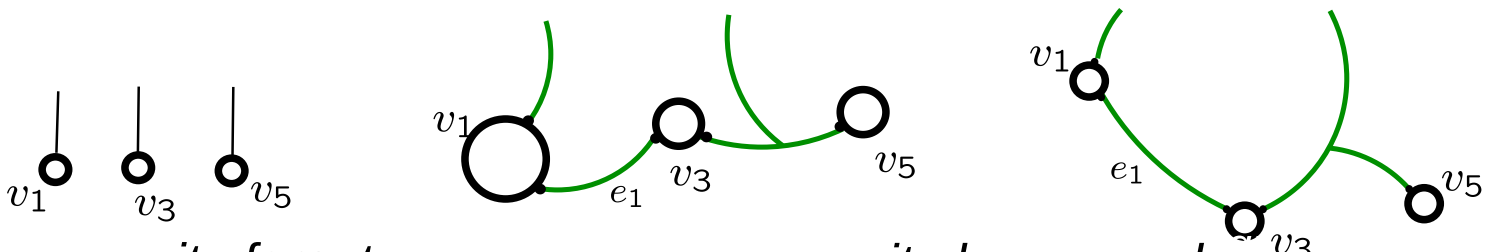
bare bigraph $\bar{G}$



its forest

its hypergraph

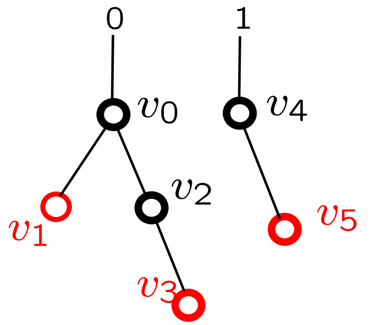
bare bigraph $\bar{F}$

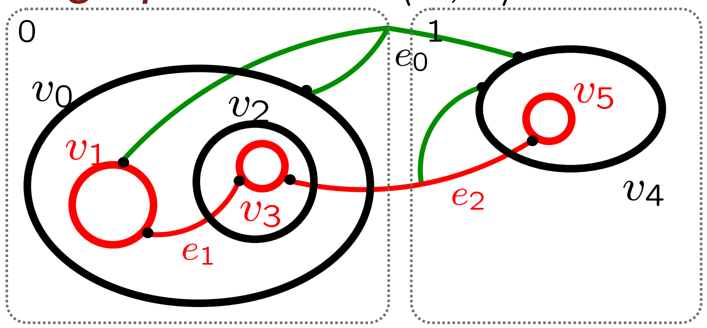


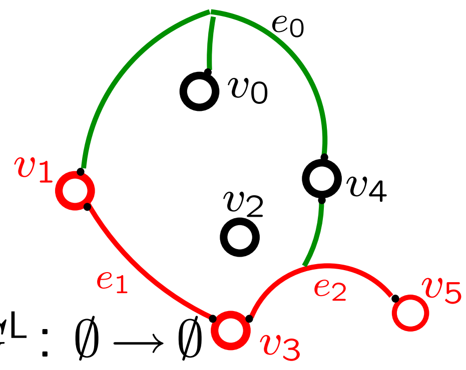
its forest

its hypergraph

ground bigraph $G : \epsilon \to \langle 2, \emptyset \rangle$

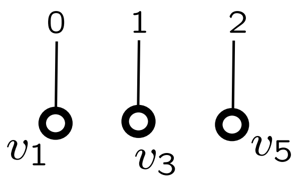


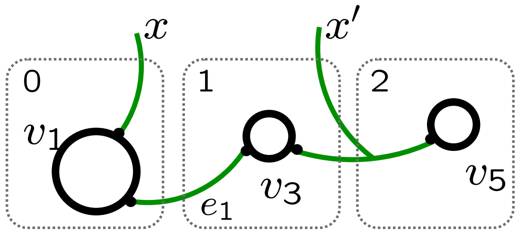


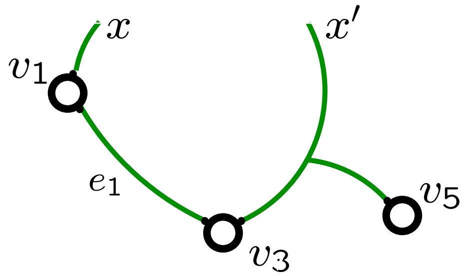


place graph $G^{\mathsf{P}} : 0 \to 2$ *link graph* $G^{\mathsf{L}} : \emptyset \to \emptyset$

ground bigraph $F : \epsilon \to \langle 3, \{x, x'\} \rangle$



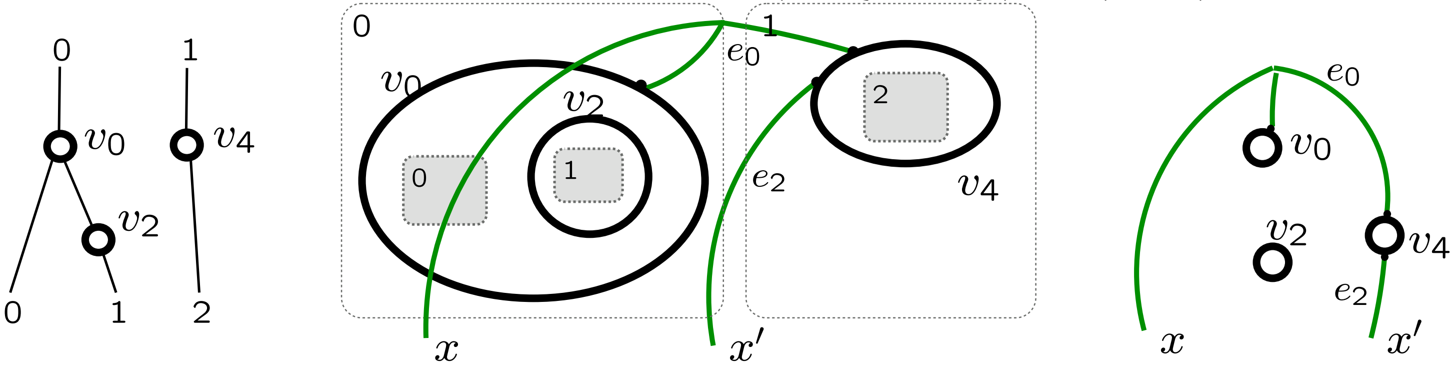




place graph $F^{\mathsf{P}} : 0 \to 3$ *link graph* $F^{\mathsf{L}} : \emptyset \to \{x, x'\}$

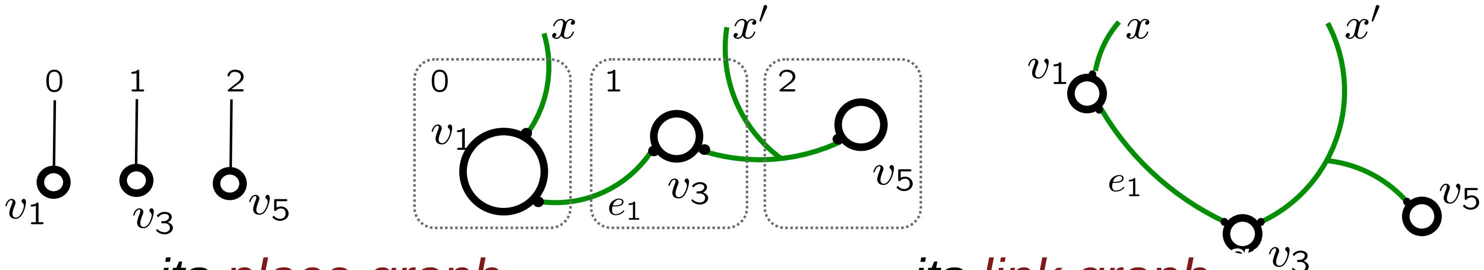
An *interface* takes the form $\langle m, X \rangle$. The *origin* is $\epsilon \stackrel{\text{def}}{=} \langle 0, \emptyset \rangle$.

a *contextual bigraph* $H : \langle 3, \{x, x'\}\rangle \rightarrow \langle 2, \emptyset\rangle$



its place graph *its link graph*

a *ground bigraph* $F : \epsilon \rightarrow \langle 3, \{x, x'\}\rangle$

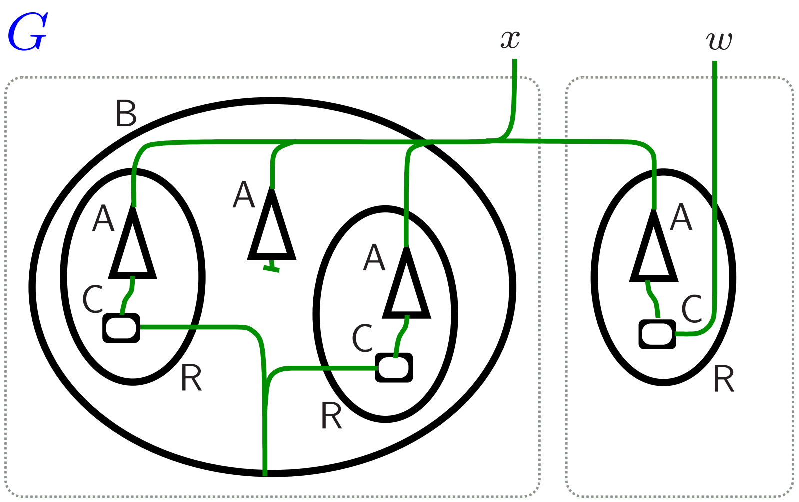


its place graph *its link graph*

An *interface* takes the form $\langle m, X\rangle$. The *origin* is $\epsilon \stackrel{\text{def}}{=} \langle 0, \emptyset\rangle$.

A built environment G

A - an agent
B - a building
C - a computer
R - a room



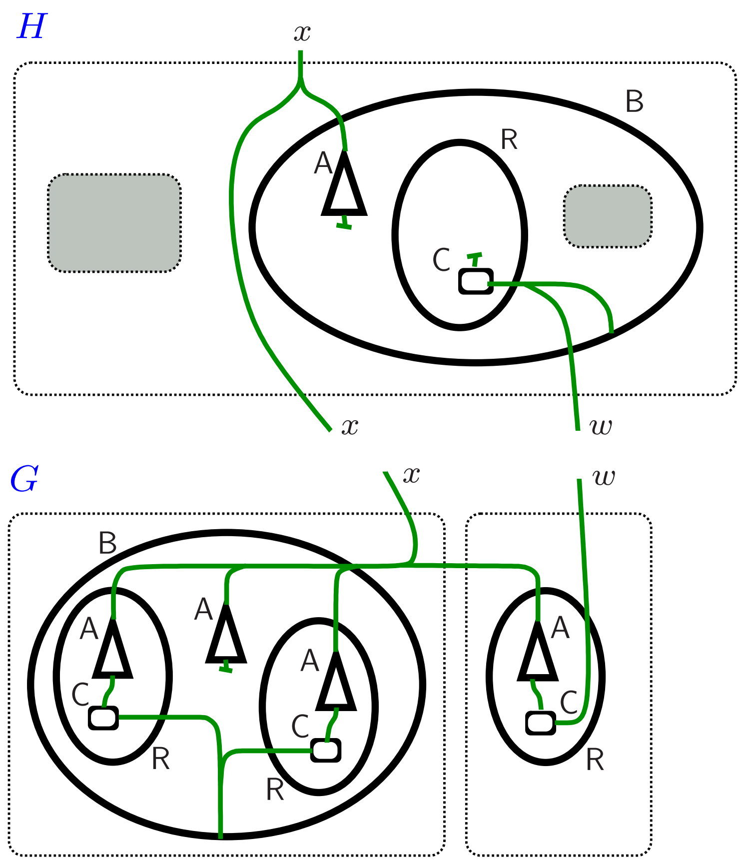
Each node has a *control*, with *arity*, e.g. A has arity 2.

$$G \;=\; /z\,\mathsf{B}_z.(\mathsf{Roomfull}_{xz} \mid /y\mathsf{A}_{xy} \mid \mathsf{Roomfull}_{xz}) \;\parallel\; \mathsf{Roomfull}_{xw}$$

where $\mathsf{Roomfull}_{xz} \stackrel{\text{def}}{=} \mathsf{R}./y\,(\mathsf{A}_{xy} \mid \mathsf{C}_{yz})$.

The *signature* $\mathcal{K} = \{\mathsf{A}:2, \mathsf{B}:1\ldots\}$ gives controls with arities.

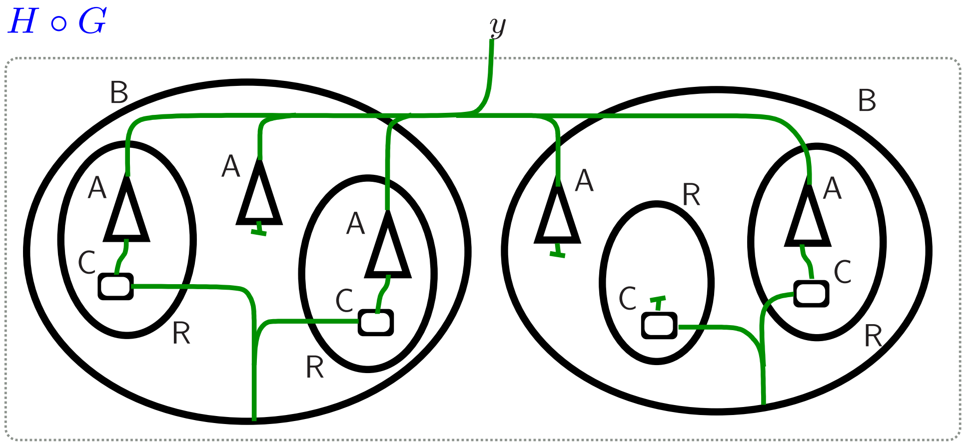
. . . . . . and a host H for G



A – an agent
B – a building
C – a computer
R – a room

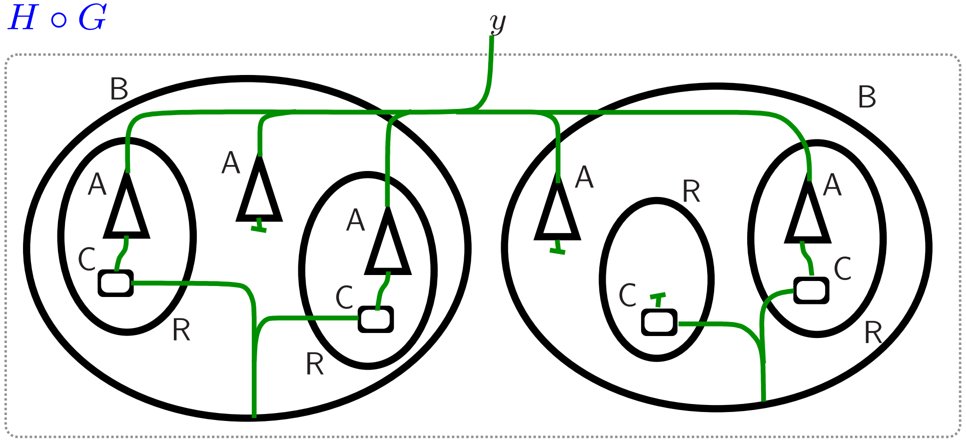
$$H \;=\; \mathsf{id}_1 \mid \mathsf{id}_x \mid /w\, \mathsf{B}_w.(/y\mathsf{A}_{xy} \mid \mathsf{R}./y\mathsf{C}_{yw} \mid \mathsf{id}_w \mid \mathsf{id}_1)\;.$$

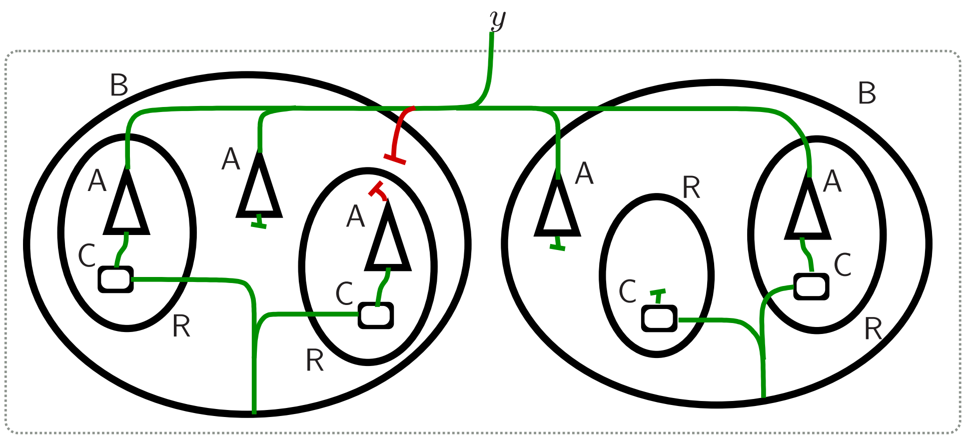
The complete system $H \circ G$



......and after one reaction

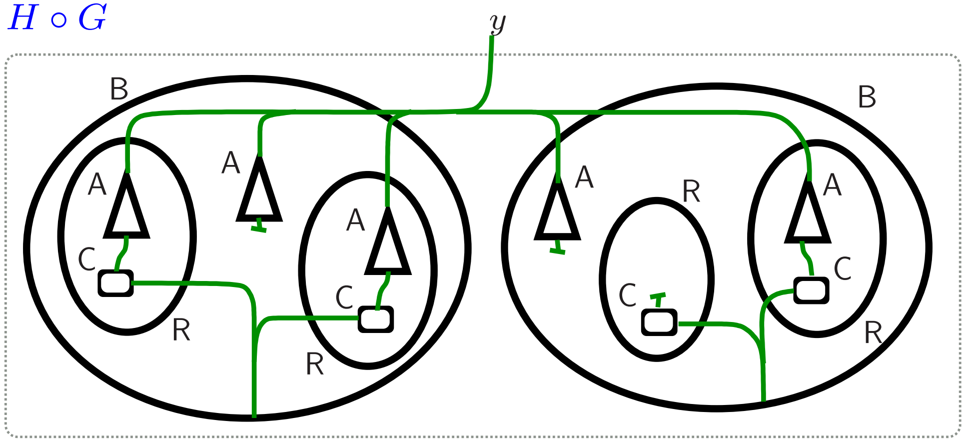
$H \circ G$

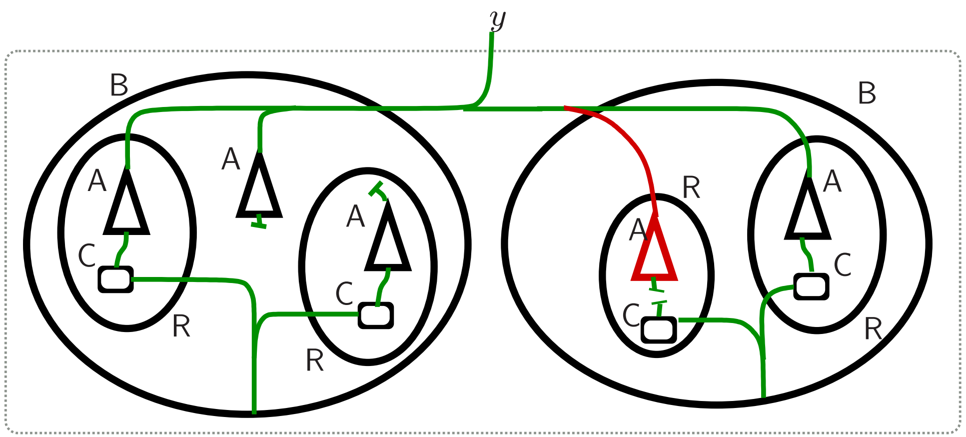




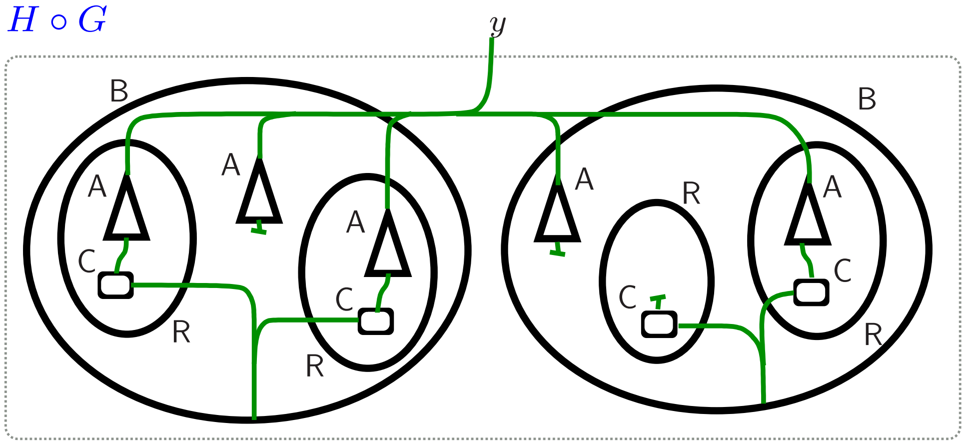
......and after two reactions

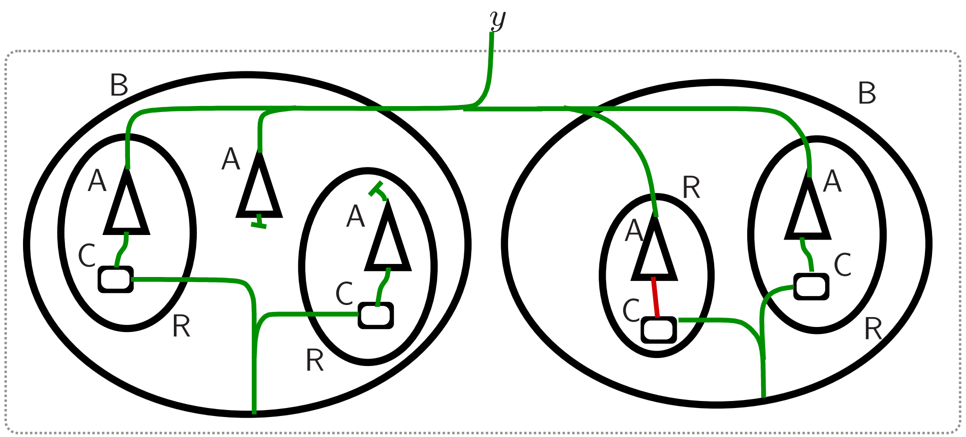
$H \circ G$





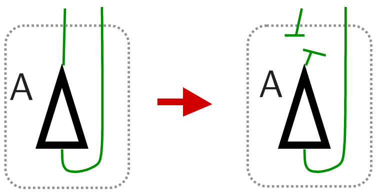
. . . . . . and after three reactions



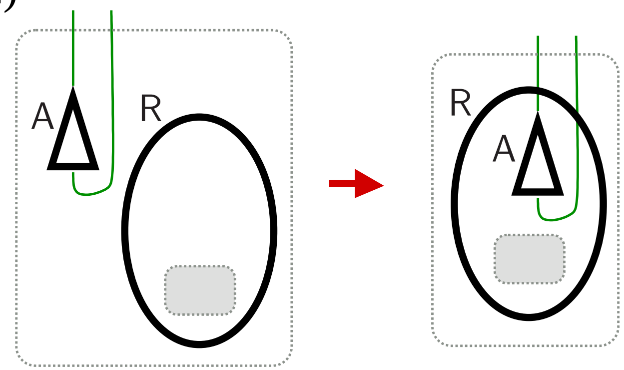


Three possible reaction rules

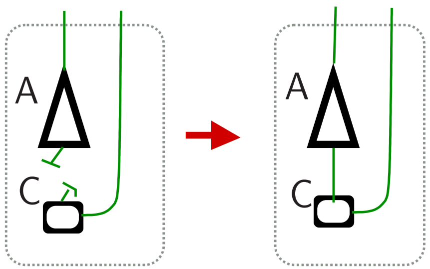
(1)



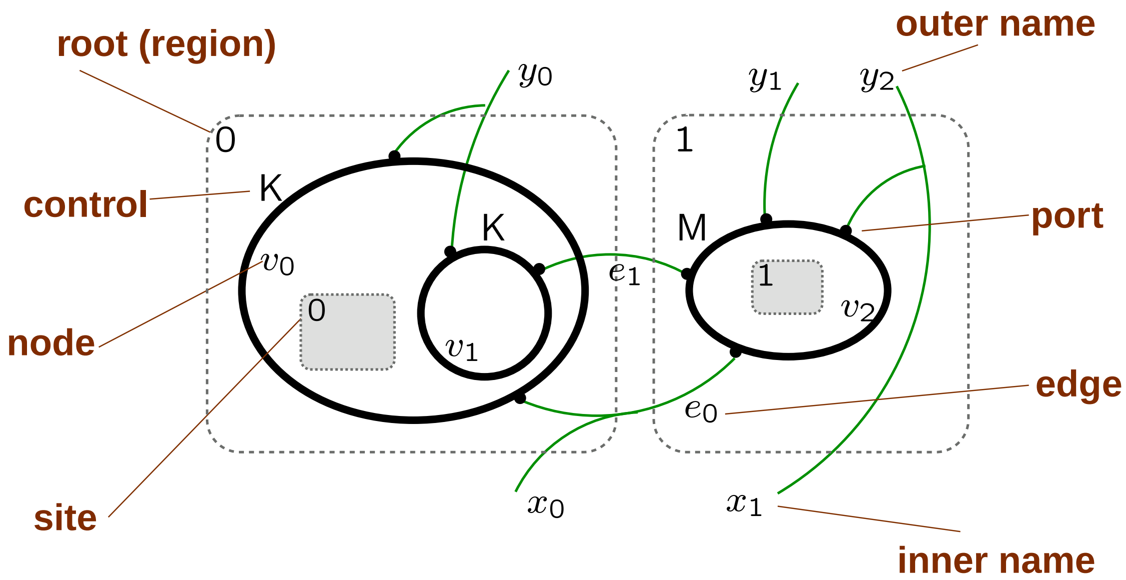
(2)



(3)



The anatomy of bigraphs



place = **root** or **node** or **site**

link = **edge** or **outer name**

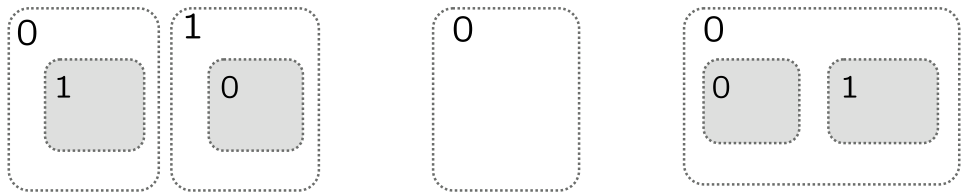
point = **port** or **inner name**

Lecture II

How to build complex systems from simple ones

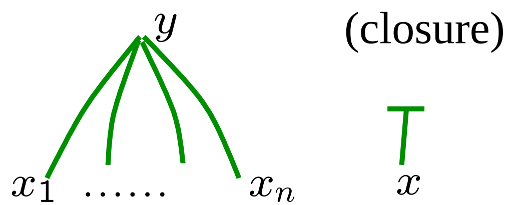
Elementary bigraphs

elementary placings:

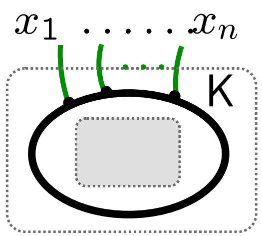


$swap : 2 \to 2$ $\quad 1 : 0 \to 1$ $\quad join : 2 \to 1$

elementary linkings:



discrete ion:



$^{y}/_{X} : X \to y$ $\quad /x : x \to \epsilon$ $\quad K_{\vec{x}} : 1 \to \langle 1, \{\vec{x}\} \rangle$

Compound placings and linkings e.g.

coalesce places: $merge_0 \stackrel{\text{def}}{=} 1\ ,\ merge_{n+1} \stackrel{\text{def}}{=} join \circ (\mathrm{id}_1 \otimes merge_n)$

substitute names: $^{y_1}/_{X_1} \otimes \cdots \otimes {}^{y_n}/_{X_n}$

Combining and composing

A bigraph *interface* takes the form $I = \langle m, X \rangle$.
The *origin* is the trivial interface $\epsilon \stackrel{\text{def}}{=} \langle 0, \emptyset \rangle$.

Combination
Let $P\colon m \to n$ and $L\colon X \to Y$ be a place graph and a link graph on the same set of nodes.
The bigraph $\langle P, L \rangle\colon \langle m, X \rangle \to \langle n, Y \rangle$ has *constituents* P, L.
For a given bigraph G, denote its constituents by G^{P} and G^{L}.

Composition
For place graphs: place each root i of $P\colon \ell \to m$ in site i of $Q\colon m \to n$, to form $Q \circ P$. Similarly for link graphs.
For $F\colon \langle \ell, X \rangle \to \langle m, Y \rangle$ and $G\colon \langle m, Y \rangle \to \langle n, Z \rangle$, define

$$G \circ F \stackrel{\text{def}}{=} \langle G^{\mathsf{P}} \circ F^{\mathsf{P}}, G^{\mathsf{L}} \circ F^{\mathsf{L}} \rangle \,.$$

Juxtaposing

For two place graphs $P_i \colon m_i \to n_i$ ($i = 0, 1$), augment the sites and roots of P_1 by m_0 and n_0, and juxtapose them to form

$$P_0 \otimes P_1 \colon m_0 + m_1 \to n_0 + n_1 \ .$$

For link graphs $L_i \colon X_i \to Y_i$, provided $X_0 \# X_1$ and $Y_0 \# Y_1$, juxtapose them to form

$$L_0 \otimes L_1 \colon X_0 \uplus X_1 \to Y_0 \uplus Y_1 \ .$$

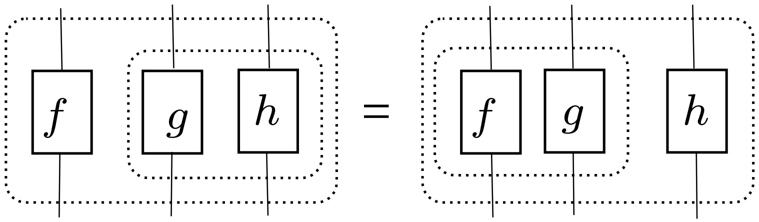
For bigraph faces $I_i = \langle m_i, X_i \rangle$ with $X_0 \# X_1$, the *product* is

$$I_0 \otimes I_1 \stackrel{\text{def}}{=} \langle m_0 + m_1, X_0 \uplus X_1 \rangle \ .$$

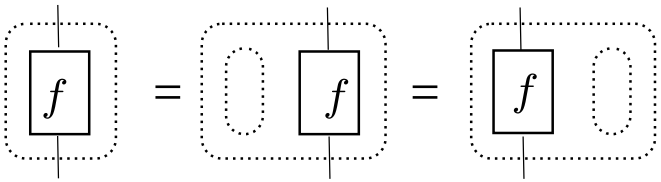
The *product* of two bigraphs G_i, with face products defined, is

$$G_0 \otimes G_1 \stackrel{\text{def}}{=} \langle G_0^{\mathsf{P}} \otimes G_1^{\mathsf{P}}, G_0^{\mathsf{L}} \otimes G_1^{\mathsf{L}} \rangle \ .$$

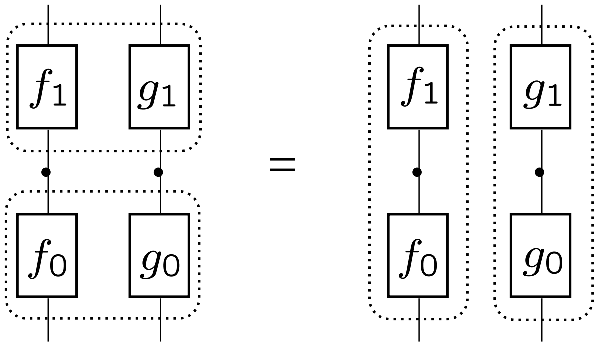
Equations for a monoidal category



$$f \otimes (g \otimes h) = (f \otimes g) \otimes h$$



$$f = \mathsf{id}_\epsilon \otimes f = f \otimes \mathsf{id}_\epsilon$$



$$(f_1 \otimes g_1) \circ (f_0 \otimes g_0)$$
$$= (f_1 \circ f_0) \otimes (g_1 \circ g_0)$$

These are informal pictures – *not* bigraphs!

Partial monoidal categories

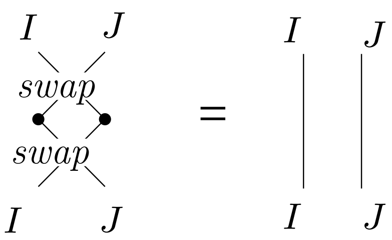
A category is *partial monoidal (pm)* if it has a *product* $\otimes$ on both objects and arrows, such that

ON OBJECTS:	ON ARROWS:
$I \otimes (J \otimes K) = (I \otimes J) \otimes K$	$f \otimes (g \otimes h) = (f \otimes g) \otimes h$
$I = \epsilon \otimes I = I \otimes \epsilon$	$f = \mathsf{id}_\epsilon \otimes f = f \otimes \mathsf{id}_\epsilon$
	$(f_1 \otimes g_1) \circ (f_0 \otimes g_0) = (f_1 \circ f_0) \otimes (g_1 \circ g_0)$

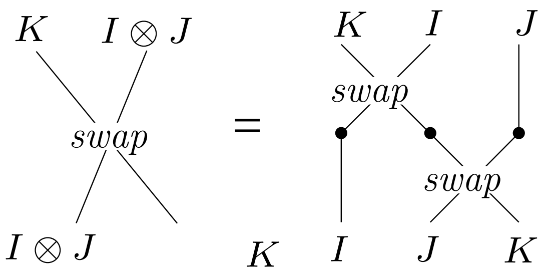
(The product is partial on objects. If $I \otimes J$ exists then so does $J \otimes I$. For $f_i \colon I_i \to J_i$ the product $f_0 \otimes f_1 \colon I_0 \otimes I_1 \to J_0 \otimes J_1$ exists iff $I_0 \otimes I_1$ and $J_0 \otimes J_1$ exist.)

Proposition For any signature $\mathcal{K}$, the three categories $\mathrm{P}\textsc{g}(\mathcal{K})$, $\mathrm{L}\textsc{g}(\mathcal{K})$ and $\mathrm{B}\textsc{g}(\mathcal{K})$ are all partial monoidal.

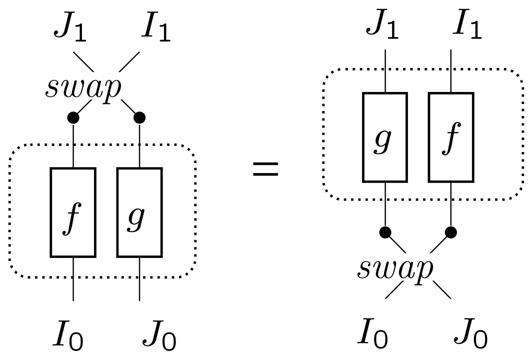
Symmetry equations: ‘swapping’



$$\gamma_{I,J} \circ \gamma_{J,I} = \mathsf{id}_{I \otimes J}$$



$$\gamma_{I \otimes J,K} = (\gamma_{I,K} \otimes \mathsf{id}_J) \circ (\mathsf{id}_I \otimes \gamma_{J,K})$$



$$\gamma_{I_1,J_1} \circ (f \otimes g) = (g \otimes f) \circ \gamma_{I_0,J_0}$$

These are informal pictures – *not* bigraphs!

Symmetric categories

Swapping

For place graphs, we define $\gamma_{m,n} : m+n \to n+m$ that swaps m with n regions, using $swap : 2 \to 2$. Then we can show that $f \otimes g$ and $g \otimes f$ are the same 'up to swapping'.

In general: a pm category is *symmetric (spm)* if there are *swapping* arrows $\gamma_{I,J} : I \otimes J \to J \otimes I$ satisfying four equations:

$$\begin{array}{l|l}
\gamma_{I,\epsilon} = \mathsf{id}_I & \gamma_{I_1,J_1} \circ (f \otimes g) = (g \otimes f) \circ \gamma_{I_0,J_0}{}^{\dagger} \\
\gamma_{J,I} \circ \gamma_{I,J} = \mathsf{id}_{I \otimes J} & \gamma_{I \otimes J,K} = (\gamma_{I,K} \otimes \mathsf{id}_J) \circ (\mathsf{id}_I \otimes \gamma_{J,K})
\end{array}$$

†where $f : I_0 \to I_1$ and $g : J_0 \to J_1$.

Proposition In bigraphs, for any signature $\mathcal{K}$ the three pm categories $\textsc{Pg}(\mathcal{K})$, $\textsc{Lg}(\mathcal{K})$ and $\textsc{Bg}(\mathcal{K})$ are all symmetric.

Placings and linkings

A *placing* ϕ is a node-free bigraph with no links.
A *linking* λ is a node-free bigraph with no places.

So a node-free bigraph takes the form $\phi \otimes \lambda$.

A bijective placing π is called a *permutation* (of places).
A bijective substitution α is called a *renaming*.

Operations *primitive* in process calculi can be *derived* for bigraphs, using $\circ$ and $\otimes$ together with placings and linkings.

ABBREVIATIONS: For $G : I \to \langle m, X \rangle$ write

$$\begin{array}{lll} \phi G & \text{for } (\phi \otimes \mathsf{id}_X) \circ G & (\phi \text{ a placing on } m) \\ \lambda G & \text{for } (\mathsf{id}_m \otimes \lambda) \circ G & (\lambda \text{ a linking on } X) \ . \end{array}$$

Derived operations: product and nesting

We want to juxtapose or to nest two bigraphs that share names.

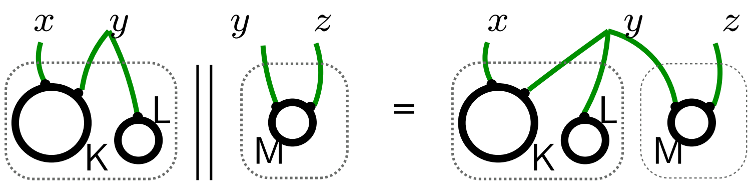
Parallel product $F_0 \parallel F_1 \stackrel{\text{def}}{=} \alpha^{-1}(\alpha F_0 \otimes F_1)$ (α a renaming)

Merge product $F_0 \mid F_1 \stackrel{\text{def}}{=} \mathit{merge}\,(F_0 \parallel F_1)$

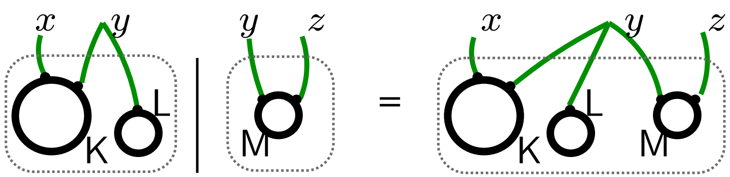
Nesting $G.F \stackrel{\text{def}}{=} (G \parallel \mathsf{id}_X) \circ F$,

where $F : I \to \langle m, X \rangle$ and $G : m \to \langle n, Y \rangle$.

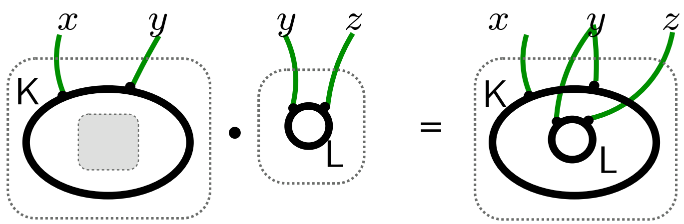
parallel product



merge product



nesting



Interfaces for derived products and nesting

Define derived products on interfaces $I_i = \langle m_i, X_i \rangle$ as follows:

$$\begin{array}{rl} \textit{Parallel product} & I_0 \parallel I_1 \stackrel{\text{def}}{=} \langle m_0 + m_1, X_0 \cup X_1 \rangle \\ \textit{Merge product} & I_0 \mid I_1 \stackrel{\text{def}}{=} \langle 1, X_0 \cup X_1 \rangle \; . \end{array}$$

Then for $F_i \colon I_i \to J_i$ we find

$$\begin{array}{rcl} F_0 \parallel F_1 & : & I_0 \otimes I_1 \to J_0 \parallel J_1 \\ F_0 \mid F_1 & : & I_0 \otimes I_1 \to J_0 \mid J_1 \; . \end{array}$$

Also for $F \colon I \to \langle m, X \rangle$ and $G \colon m \to \langle n, Y \rangle$ we find

$$G.F \colon I \to \langle n, X \cup Y \rangle \; .$$

Thus all these operators allow their operands to share names.

Place sorting, bigraphical category

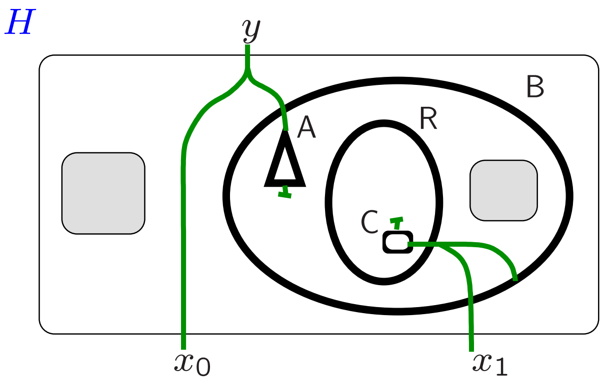
A *place sorting* $\Sigma = \{\Theta, \mathcal{K}, \Phi\}$ has

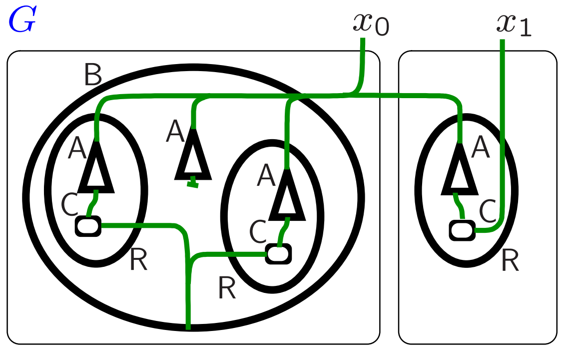
$$\begin{array}{rl} \textit{Sorts} & \Theta \\ \textit{Signature} & \mathcal{K} = \{\mathsf{K}_1{:}(k_1, \theta_1), \mathsf{K}_2{:}(k_2, \theta_2), \ldots\} \\ \textit{Formation rule} & \Phi\ . \end{array}$$

Each *control* K_i is a kind of node, with k_i ports, and sort $\theta_i \in \Theta$.
Sorts are thus ascribed to nodes, and also to interface places.
Using these, the formation rule Φ limits the admissible bigraphs.
(Link sorting is defined similarly.)

This yields the *bigraphical category* $\mathrm{B}\textsc{g}(\Sigma)$.

Place sorting for the built environment





A – an agent
B – a building
C – a computer
R – a room

Sorts Θ: $\{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{r}, \widehat{\mathsf{ac}}, \widehat{\mathsf{ar}}\}$
Signature $\mathcal{K}$: $\{\mathsf{A} : \mathsf{a},\ \mathsf{B} : \mathsf{b},\ \mathsf{C} : \mathsf{c},\ \mathsf{R} : \mathsf{r}\}$
Formation Φ: allowed place-graphs —

PARENT	CHILD SORTS
a-node, c-node	none
b-node	a, r, $\widehat{\mathsf{ar}}$
r-node	a, c, $\widehat{\mathsf{ac}}$
θ-root $(\theta \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{r}\})$	θ
$\widehat{\mathsf{ar}}$-root	a, r, $\widehat{\mathsf{ar}}$
$\widehat{\mathsf{ac}}$-root	a, c, $\widehat{\mathsf{ac}}$

Finite CCS

SYNTAX:

$$
\begin{array}{lcl}
\mu & ::= & \overline{x} \;\big|\; x \\
P & ::= & A \;\big|\; \nu x P \;\big|\; P \,|\, P \quad \textit{processes} \\
A & ::= & \mathbf{0} \;\big|\; \mu.P \;\big|\; A{+}A \quad \textit{alternations}
\end{array}
$$

A handshake on x can occur iff one process can do $\overline{x}$ and another do x.

STRUCTURAL CONGRUENCE:

(1) $P \equiv_\alpha Q$ implies $P \equiv Q$, and $A \equiv_\alpha B$ implies $A \equiv B$;

(2) '$|$' and '$+$' are associative and commutative under $\equiv$, and $A + \mathbf{0} \equiv A$;

(3) $\nu x \nu y P \equiv \nu y \nu x P$;

(4) $\nu x P \equiv P$ and $\nu x\,(P \,|\, Q) \equiv P \,|\, \nu x Q$ for any x not free in P;

(5) $\nu x\,(A{+}\mu.P) \equiv A + \mu.\nu x P$ for any x not free in A or μ.

CCS in bigraphs

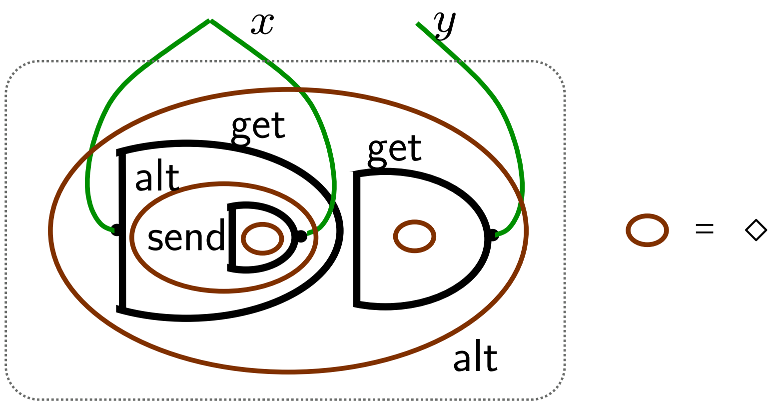
Sorts: $\Theta_{\mathsf{CCS}} \;=\; \{\mathsf{pr},\mathsf{ch}\}$

Signature: $\mathcal{K}_{\mathsf{ccs}} \;=\; \{\mathsf{alt} : (\mathsf{pr},0),\; \mathsf{get} : (\mathsf{ch},1),\; \mathsf{send} : (\mathsf{ch},1)\}$

Formation rule Φ_{CCS}: nest the nodes with sorts alternating.

The CCS choice $(x.(\overline{x}.\mathsf{nil}) + y.\mathsf{nil})$ becomes the bigraph

$$\mathsf{alt.(get}_x\mathsf{.alt.send}_x. \diamond \mid \mathsf{get}_y.\diamond),\ \text{where } \diamond \stackrel{\text{def}}{=} \mathsf{alt.1}\ .$$



Translating CCS into $\mathsf{BG}_{\mathsf{ccs}}$

The **translation map** $\mathcal{P}_X[\cdot]$ from processes to $\epsilon \to \langle 1 : \mathsf{pr}, X \rangle$ is defined for all processes P with free names in X. Similarly $\mathcal{A}_X[\cdot]$ for alternations.

$$
\begin{array}{rcl|rcl}
 & & & \mathcal{A}_X[\mathbf{0}] & = & X \mid 1 \\
\mathcal{P}_X[\nu x P] & = & /y\, \mathcal{P}_{y \uplus X}[\{^y/_x\}P] & \mathcal{A}_X[\overline{x}.P] & = & \mathsf{send}_x.\mathcal{P}_X[P] \\
\mathcal{P}_X[P \mid Q] & = & \mathcal{P}_X[P] \mid \mathcal{P}_X[Q] & \mathcal{A}_X[x.P] & = & \mathsf{get}_x.\mathcal{P}_X[P] \\
\mathcal{P}_X[A] & = & \mathsf{alt}.\,\mathcal{A}_X[A]\ . & \mathcal{A}_X[A+B] & = & \mathcal{A}_X[A] \mid \mathcal{A}_X[B]\ .
\end{array}
$$

Theorem

(1) The translations are surjective.

(2) $P \equiv Q$ iff $\mathcal{P}_X[P] = \mathcal{P}_X[Q]$, and
$A \equiv B$ iff $\mathcal{A}_X[A] = \mathcal{A}_X[B]$.

Lecture III

Dynamical theory, illustrated for CCS

Reaction in CCS

SYNTAX:

$$\begin{array}{rcll} \mu & ::= & \overline{x} \;\big|\; x & \\ P & ::= & A \;\big|\; \nu x P \;\big|\; P \,|\, P & \textit{processes} \\ A & ::= & \mathbf{0} \;\big|\; \mu.P \;\big|\; A + A & \textit{alternations} \end{array}$$

REACTION: $(\overline{x}.P + A) \mid (x.Q + B) \longrightarrow P \mid Q$ with three rules:

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \frac{P \longrightarrow P'}{\nu x P \longrightarrow \nu x P'}$$

$$\frac{P \longrightarrow P'}{Q \longrightarrow Q'} \quad \text{if } P \equiv Q \text{ and } P' \equiv Q'$$

So, reaction can occur anywhere except within a choice.

How do we match these reactions in bigraphs?

Bigraphical reactive system

Where may reactions occur in a bigraph?

To determine this, a basic signature is enriched to a *dynamic* signature, declaring each control to be either *active* or *passive*. A context D is *active* if no site lies within a passive node.

A BRS $\mathrm{B{\scriptstyle G}}(\Sigma,\mathcal{R})$ has *ground reaction rules*, each of the form

$$\mathsf{R} \;=\; (r,\ r')$$

with *redex* $r : \epsilon{\rightarrow}I$ and *reactum* $r' : \epsilon{\rightarrow}I$.

The following reactions are generated by R, where D is active:

$$D \circ r \longrightarrow_{\mathsf{R}} D \circ r' \,.$$

Parametric reaction rules

The rules in a BRS $\mathrm{B}\textsc{g}(\Sigma,\mathcal{R})$ are usually *parametric*, each generating a family of ground rules.

A parametric rule takes the form

$$\mathsf{R} = (R,\ R',\ \eta)$$

with *redex* $R\colon m \to I$, *reactum* $R'\colon m' \to I$, and *instance map* $\eta\colon m' \to m$.

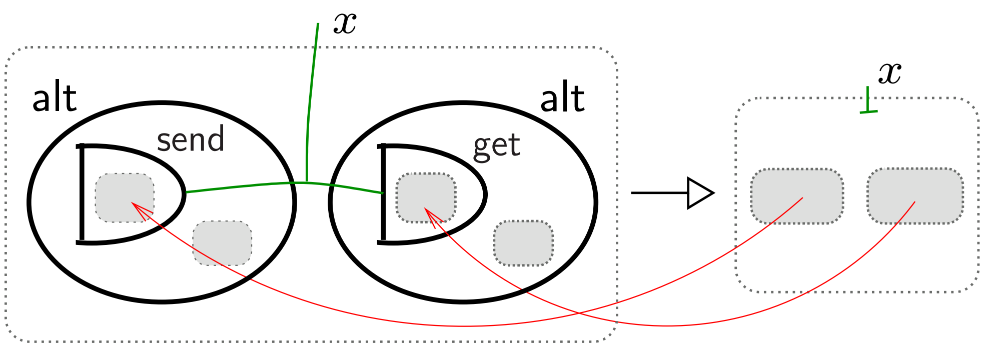
A *parameter* is a discrete* bigraph $d = d_0 \otimes \cdots \otimes d_{m-1}$ with names not in I. Its *instance* is $d' \stackrel{\text{def}}{=} d_{\eta(0)} \parallel \cdots \parallel d_{\eta(m'-1)}$. This generates the ground reaction rule

$$(r, r') = (R.d, R'.d') \ .$$

**discrete*: all links open and distinct.

Reaction in CCS bigraphs

$\mathrm{BG}_{\mathrm{CCS}}$, the BRS for CCS, has a single parametric reaction rule:



$\mathsf{alt}.(\mathsf{send}_x \mid \mathsf{id}) \mid \mathsf{alt}.(\mathsf{get}_x \mid \mathsf{id})$ $\qquad$ $x \mid \mathsf{id} \mid \mathsf{id}$

The back-pointing arrows show the instance map.

Theorem

Translation of CCS into bigraphs strongly preserves reaction:
$P \longrightarrow P'$ iff $\mathcal{P}_X[P] \longrightarrow \mathcal{P}_X[P']$.

Behavioural equivalence

The big challenge of process calculi: Find a nice behavioural equivalence $\asymp$, or preorder $\succeq$ such that

$P \asymp Q$ means *in all contexts* P and Q behave the same;
$P \succeq Q$ means *in all contexts* P can do everything that Q can do.

Thus, these relations must be *congruential*; $P \asymp Q$ must imply $\mathcal{C}[P] \asymp \mathcal{C}[Q]$ for any context $\mathcal{C}$.

We can't define $\asymp$ to mean 'having same reactions'. For:
– nil and $\overline{x}.\mathsf{nil}$ have exactly the same reactions in CCS;
– but in the context $\cdot\cdot \mid x.\mathsf{nil}$, we find $\overline{x}.\mathsf{nil}$ can react but nil cannot.

For this reason, *labelled transitions* **were defined for CCS.**

Labelled transitions in CCS

SYNTAX:

$$
\begin{array}{lcl}
\mu & ::= & \overline{x} \;\big|\; x \;\big|\; \tau \\
P & ::= & A \;\big|\; \nu x P \;\big|\; P \,|\, P \quad \textit{processes} \\
A & ::= & \mathbf{0} \;\big|\; \mu.P \;\big|\; A{+}A \quad \textit{alternations}
\end{array}
$$

TRANSITION: $\mu.P + A \xrightarrow{\mu} P$, with four rules:

$$
\frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}
\qquad\qquad
\frac{P \xrightarrow{\mu} P'}{\nu x P \xrightarrow{\mu} \nu x P'} \quad \text{if } \mu \notin \{x, \overline{x}\}
$$

$$
\frac{P \xrightarrow{\overline{x}} P' \quad Q \xrightarrow{x} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}
\qquad\qquad
\frac{P \xrightarrow{\mu} P'}{Q \xrightarrow{\mu} Q'} \quad \text{if } P \equiv Q \text{ and } P' \equiv Q'
$$

Now, how to define behavioural equivalence using transitions?

Bisimulation for CCS

A simulation is a binary relation $\mathcal{S}$ between processes such that:
if $P\mathcal{S}Q$ and $P \xrightarrow{\ell} P'$ then there exists
Q' such that $Q \xrightarrow{\ell} Q'$ and $P'\mathcal{S}Q'$.

A bisimulation is a symmetric simulation. Then **bisimilarity**, denoted by $\sim$, is the largest bisimulation.

Theorem: Bisimulation is a congruence, i.e. $P \sim Q$ implies $\mathcal{C}[P] \sim \mathcal{C}[Q]$ for every context $\mathcal{C}$.

Similar congruence theorems have been proved for other process calculi and other equivalences and pre-orders.
Can we prove such a result uniformly in bigraphs?

Aims for a general behavioural theory for BRSs

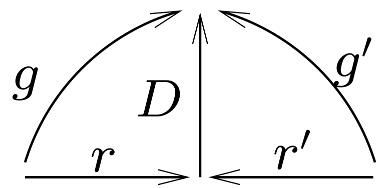
- To *derive* labelled transition systems from reaction rules.

- The label L in a transition $g \xrightarrow{L} g'$ should represent how an environment should contribute to the transition.

- This must illuminate behavioural equivalences and preorders based upon transitions.

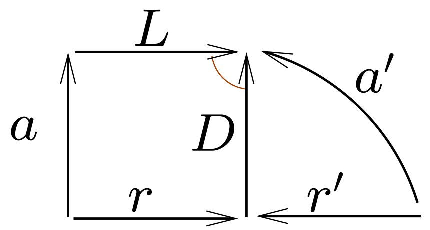
Contexts as transition labels

- A transition $a \xrightarrow{L} a'$ should imply the reaction $L \circ a \longrightarrow a'$.
- So reactions and transitions have similar diagrams, for some active D and ground reaction rule (r, r'):



reaction $g \longrightarrow g'$

COMPARE!

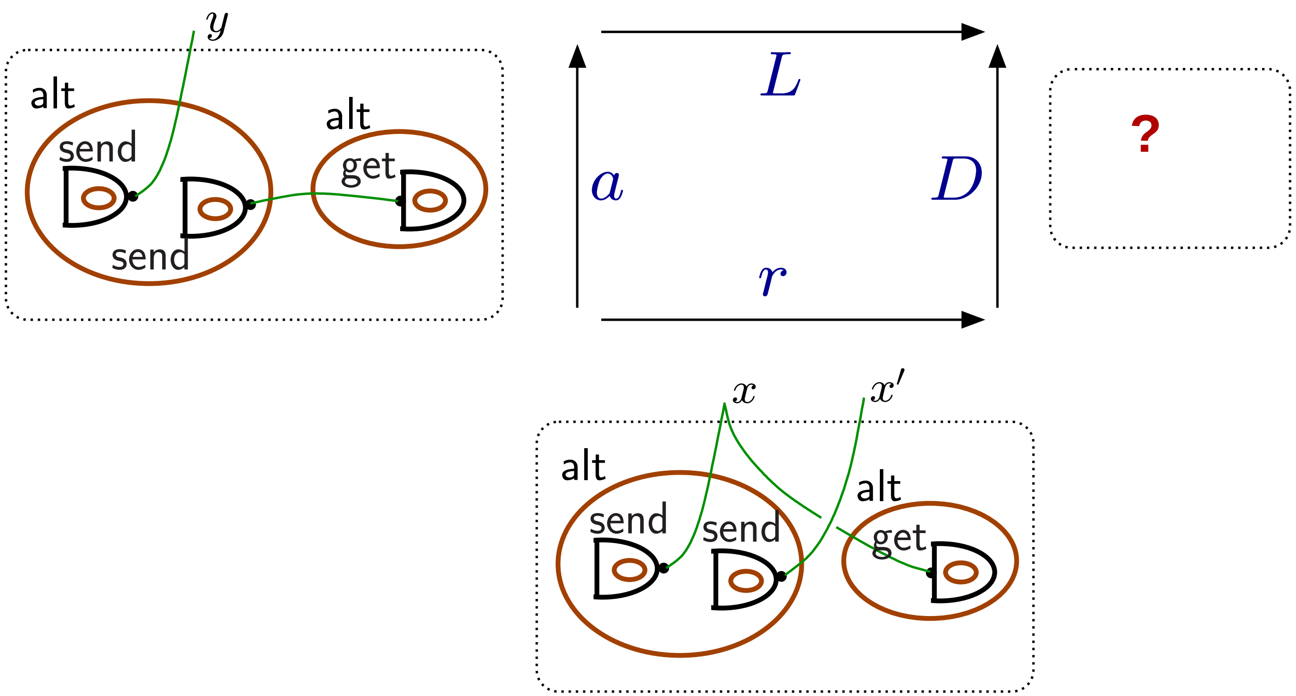


transition $a \xrightarrow{L} a'$

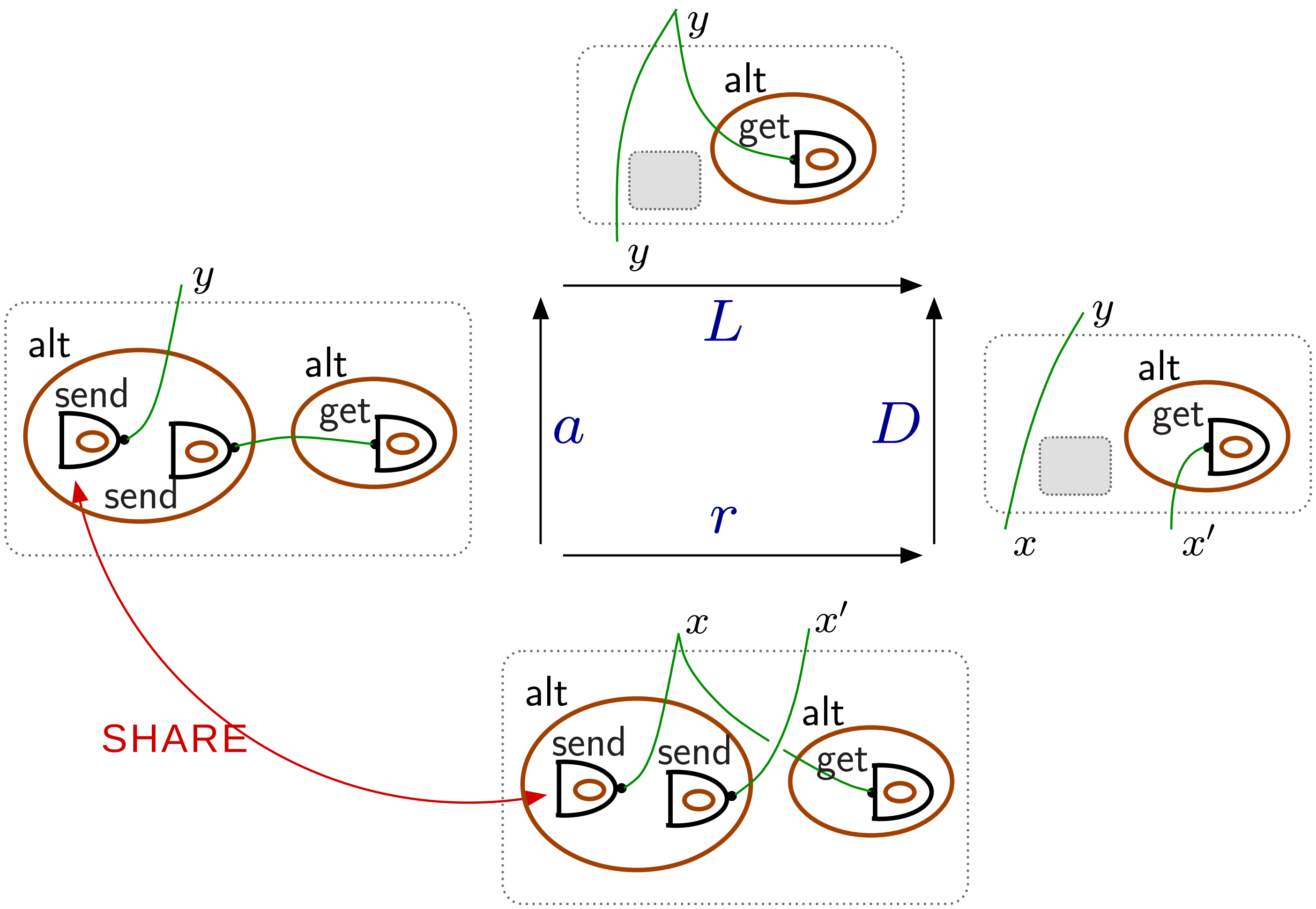
- But we must impose some constraint on the *bound* (L, D) for (a, r), to prevent it becoming enormous!
- L must contain only that part of r that is not in a. Such a bound, and the resulting transition, will be called *minimal*.

What’s a minimal bound?

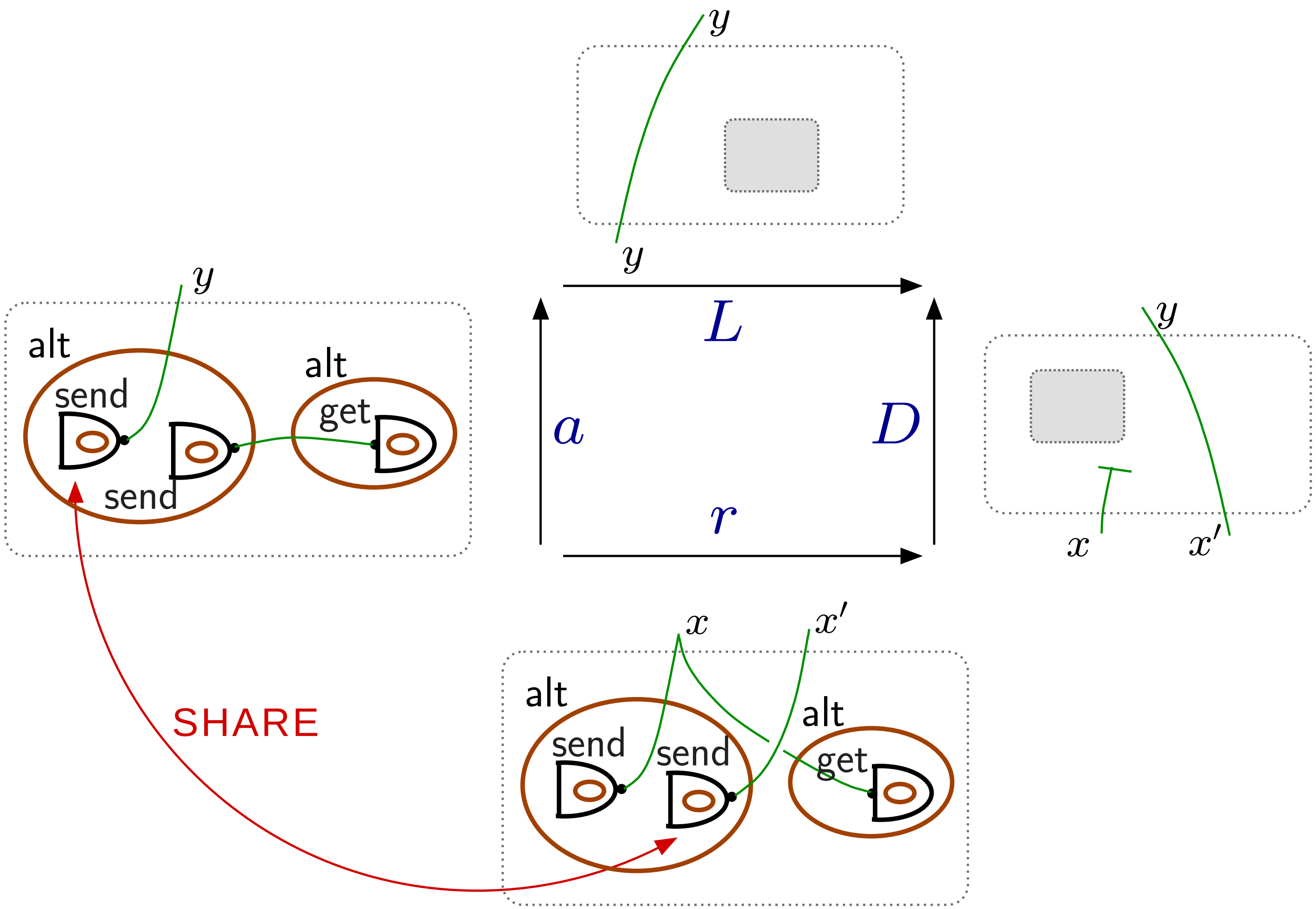
Make L minimal
so that $L \circ a = D \circ r$



What's a minimal bound?-(1)



What's a minimal bound?-(2)



Behavioural congruence

Theorem In any safe[†] BRS, the bisimulation equivalence $\sim$ for the system of minimal transitions is a *congruence*; that is, if $a \sim b$ then $C \circ a \sim C \circ b$, for any context C.

- The same holds for other equivalences and pre-orders.

- It holds also in a much wider class of reactive systems, provided that *minimality* can be defined.

- For CCS, how does derived bisimilarity relate to that for original CCS, where the labels are *raw*, not *contextual*?

[†] The property 'safe' refers to sorting. Σ_{CCS} is safe.

Raw transitions for CCS

The table characterizes the three components in any raw CCS transition $s \xrightarrow{\mu} s'$:

	s	μ	s'	condition
1	$\nu Z((\overline{x}.p + \cdot\cdot) \mid q)$	$\overline{x}$	$\nu Z(p \mid q)$	$x \notin Z$
2	$\nu Z((x.p + \cdot\cdot) \mid q)$	x	$\nu Z(p \mid q))$	$x \notin Z$
3	$\nu Z((\overline{x}.p_0 + \cdot\cdot) \mid (x.p_1 + \cdot\cdot) \mid q)$	τ	$\nu Z(p_0 \mid p_1 \mid q)$	

Denote by $\sim_{\mathsf{CCS}}$ the bisimilarity for these raw transitions. It is preserved by all CCS contexts.

Derived transitions for CCS

The table characterizes the transitions $g \xrightarrow{L} g'$ derived for CCS in bigraphs (with a little fine-tuning).

	g	L	g'	condition
1	$/Z\big(\mathsf{alt}.(\mathsf{send}_x.a\ \cdot\cdot) \mid b\big)$	$\mathsf{id} \mid \mathsf{alt}.(\mathsf{get}_x.\diamond)$	$/Z(a \mid b)$	$x \notin Z$
2	$/Z\big(\mathsf{alt}.(\mathsf{get}_x.a\ \cdot\cdot) \mid b\big)$	$\mathsf{id} \mid \mathsf{alt}.(\mathsf{send}_x.\diamond)$	$/Z(a \mid b)$	$x \notin Z$
3	$/Z\big(\mathsf{alt}.(\mathsf{send}_x.a_0\ \cdot\cdot)$ $\mid \mathsf{alt}.(\mathsf{get}_x.a_1\ \cdot\cdot) \mid b\big)$	id	$/Z(a_0 \mid a_1 \mid b)$	
4	$/Z\big(\mathsf{alt}.(\mathsf{send}_x.a_0\ \cdot\cdot)$ $\mid \mathsf{alt}.(\mathsf{get}_y.a_1\ \cdot\cdot) \mid b\big)$	y/x	$/Z\, y/x$ $(a_0 \mid a_1 \mid b)$	$x \neq y$; $x, y \notin Z$

Corollary Bisimilarity for these transitions is a congruence; i.e. if $g \sim g'$ then $C \circ g \sim C \circ g'$, for any bigraph context C.

Theorem **Omitting case 4**, this bisimilarity agrees exactly with the original CCS bisimilarity $\sim_{\mathsf{CCS}}$.

Lecture IV

Stochastic dynamics, e.g. for membrane budding

Joint work with Jean Krivine and Angelo Troina

Rated rules and reactions

A *rated* reaction rule has form $\mathsf{R} = (R, R', \eta, \rho)$, with *rate* $\rho > 0$.
How likely is a reaction $g \longrightarrow_{\mathsf{R}} g'$? Define

$$rate_{\mathsf{R}}[g, g'] \stackrel{\text{def}}{=} \rho \cdot \mu_{\mathsf{R}}[g, g']$$

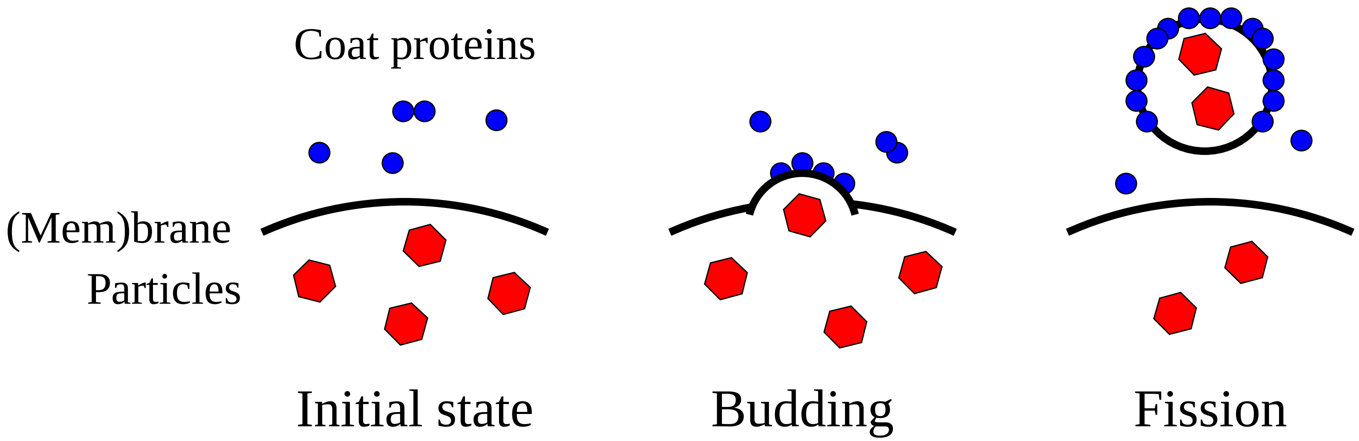
where $\mu_{\mathsf{R}}[g, g']$ is the number of *distinct* ground rules (r, r') of R with $(C \circ r, C \circ r') = (g, g')$ for some active C.

REMARK: When the redex R is *solid*, r determines C and r'.

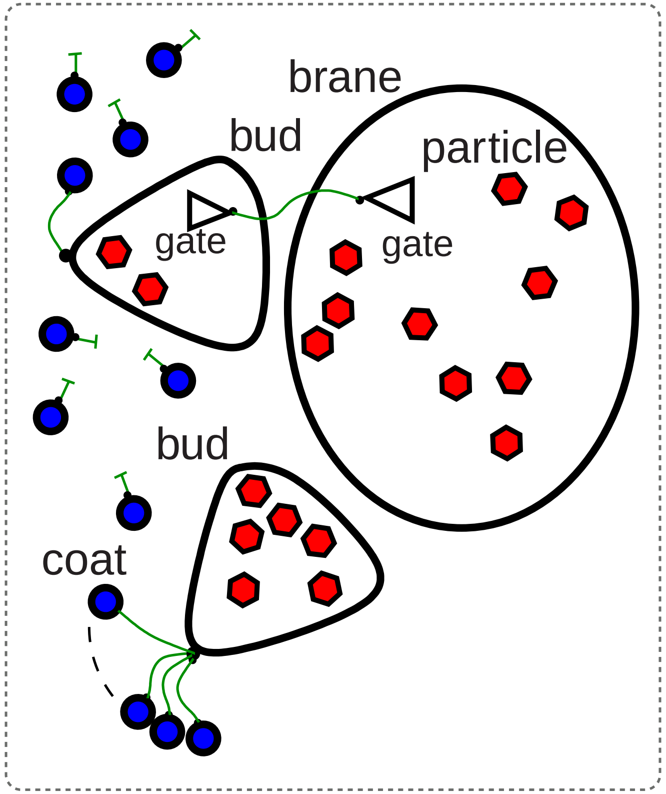
If $\mathcal{R}$ is a set of rules, define $rate_{\mathcal{R}}[g, g'] \stackrel{\text{def}}{=} \sum_{\mathsf{R} \in \mathcal{R}} rate_{\mathsf{R}}[g, g']$.

Proposition $\quad g \longrightarrow_{\mathcal{R}} g'$ iff $rate_{\mathcal{R}}[g, g'] > 0$.

Membrane budding



A membrane-bud system



Sorting $\Sigma = (\Theta, \mathcal{K}, \Phi)$:

Sorts $\Theta = \{\mathsf{b}, \mathsf{c}, \mathsf{p}, \mathsf{g}, \widehat{\mathsf{bc}}, \widehat{\mathsf{pg}}\}$.

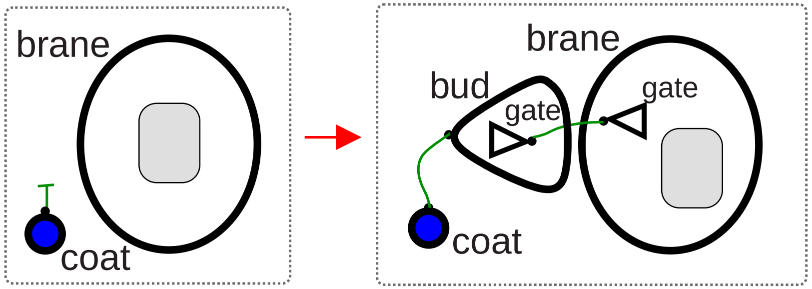
Signature $\mathcal{K}$ =

$\{\mathsf{brane}\colon (\mathsf{b}, 0), \mathsf{bud}\colon (\mathsf{b}, 1), \mathsf{coat}\colon (\mathsf{c}, 1),$
$\mathsf{particle}\colon (\mathsf{p}, 0), \mathsf{gate}\colon (\mathsf{g}, 1)\}$

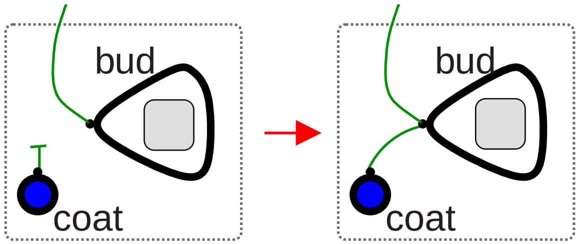
Formation rule Φ:

PARENT	CHILD SORTS
b-node	$\mathsf{p}, \mathsf{g}, \widehat{\mathsf{pg}}$
c-node, p-node, g-node	none
θ-root $(\theta \in \{\mathsf{b}, \mathsf{c}, \mathsf{p}, \mathsf{g}\})$	θ
$\widehat{\mathsf{bc}}$-root	$\mathsf{b}, \mathsf{c}, \widehat{\mathsf{bc}}$
$\widehat{\mathsf{pg}}$-root	$\mathsf{p}, \mathsf{g}, \widehat{\mathsf{pg}}$

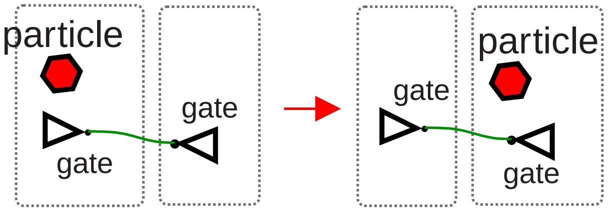
Reaction rules for budding



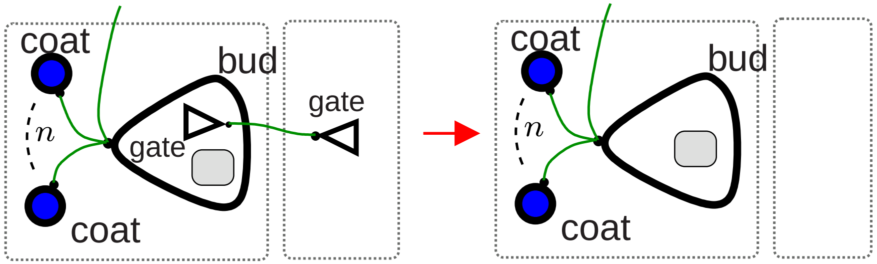
bud formation



coating

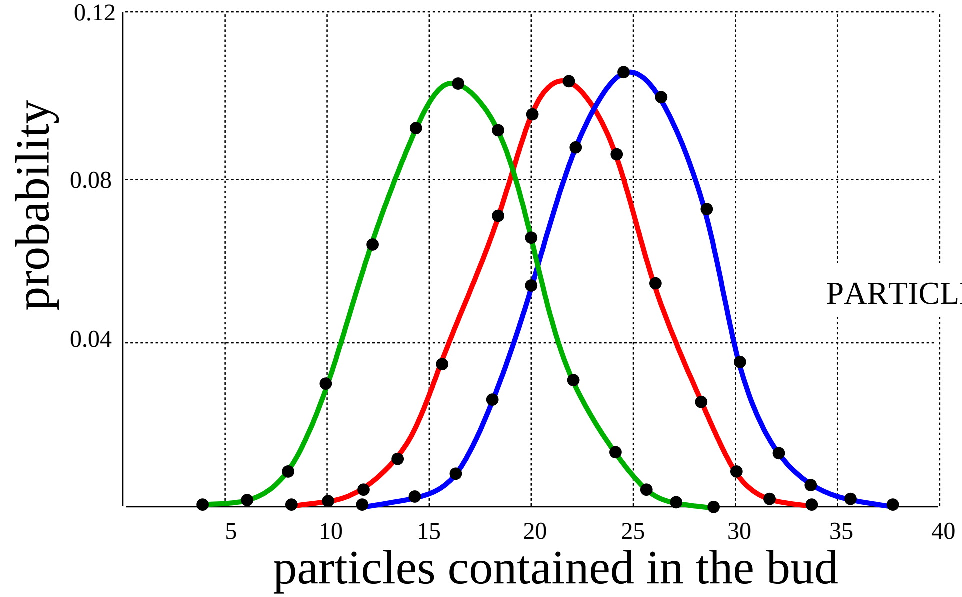


particle migration



bud fission

A simulation of budding, using PRISM



RATES:

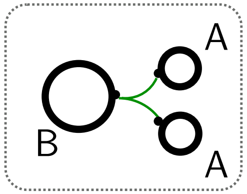
PARTICLE MIGRATION	1	1	2
COATING	2	1	1

As the rate of particle migration increases, relative to the coating rate, the expected number of particles in a bud increases.

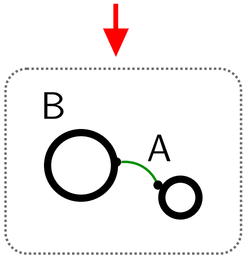
This number has a normal distribution of constant width.

A reaction: how many ways can it happen? . . .

g

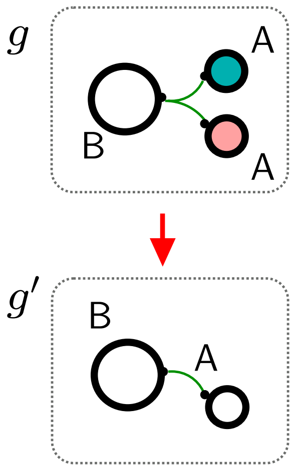


g'



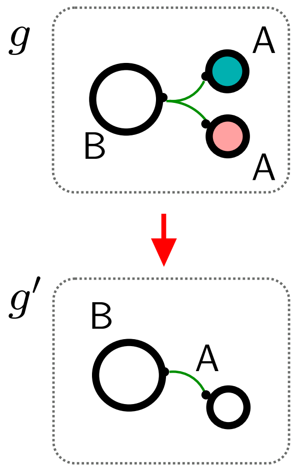
A reaction: how many ways can it happen? . . .

First identify the A-nodes

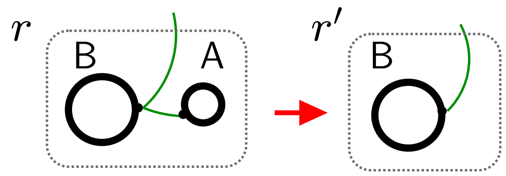


A reaction: how many ways can it happen? . . .

How many ways with rule R?

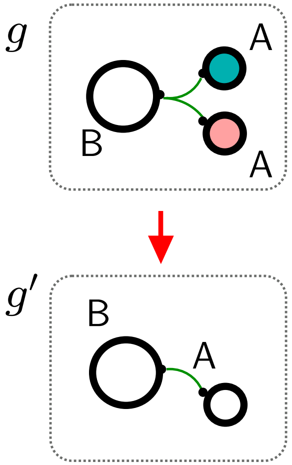


rule R

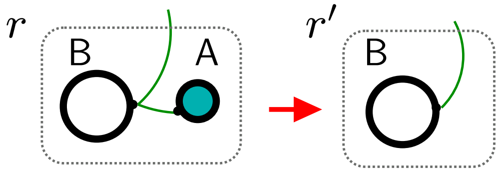


A reaction: how many ways can it happen? . . .

How many ways with rule R? . . . ONE WAY . . .

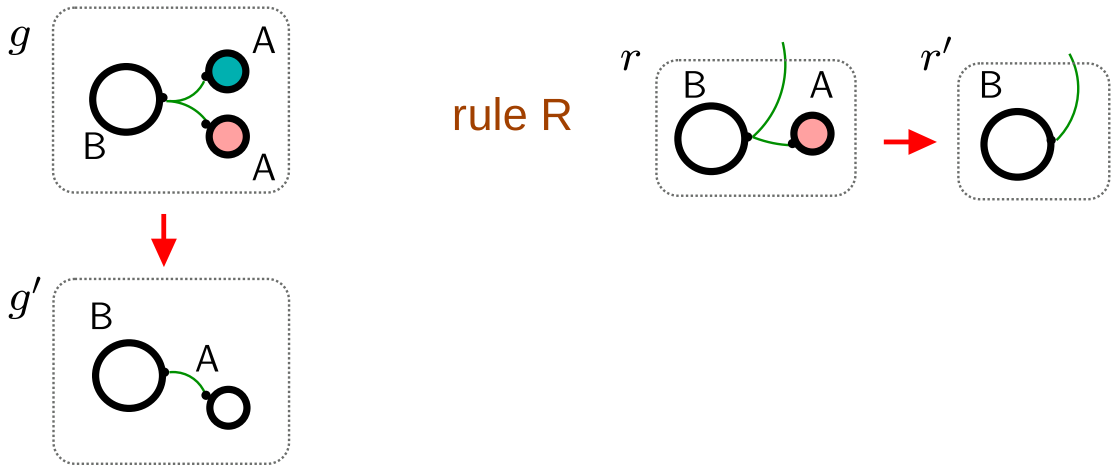


rule R



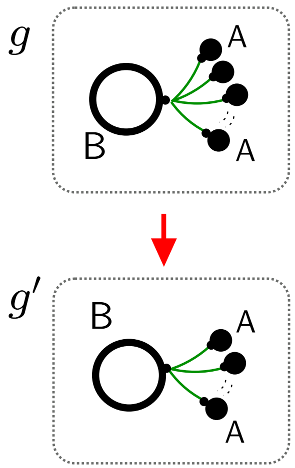
A reaction: how many ways can it happen? . . .

How many ways with rule R? . . . and ANOTHER WAY!

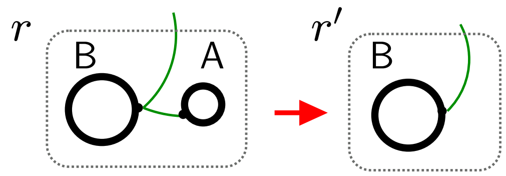


A reaction: how many ways can it happen? ...

How many ways with rule R? ... OR n WAYS...



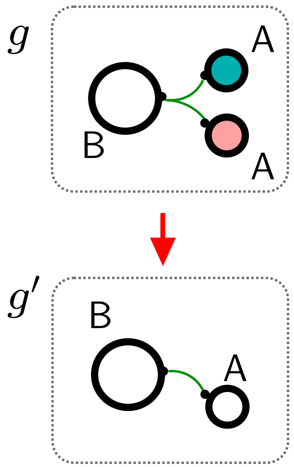
rule R



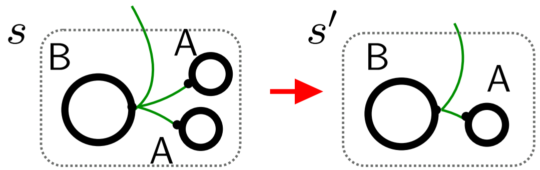
if g has n A-nodes!

A reaction: how many ways can it happen? . . .

How many ways with rule S?

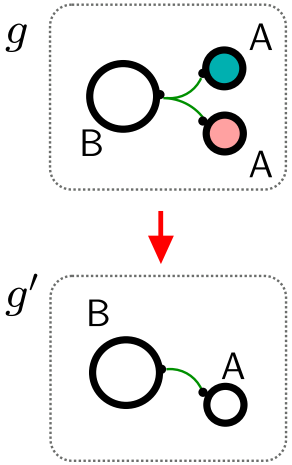


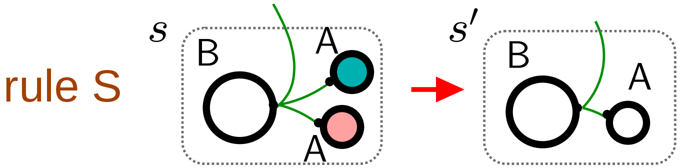
rule S



A reaction: how many ways can it happen? . . .

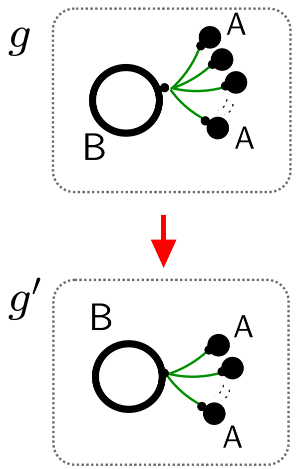
How many ways with rule S? . . . ONLY ONE WAY!





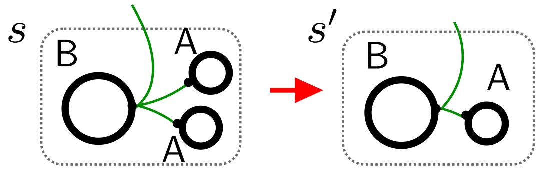
A reaction: how many ways can it happen? . . .

How many ways with rule S? . . . OR $n(n-1)/2$ WAYS. . .



if g has n A-nodes!

rule S



Computing transition rates

Let R be an rated reaction rule with rate $\rho > 0$. How likely is a transition $a \xrightarrow{L}_{\mathsf{R}} a'$? Define

$$rate_{\mathsf{R}}[a, L, a'] \stackrel{\text{def}}{=} \rho \cdot \mu_{\mathsf{R}}[a, L, a']$$

where $\mu_{\mathsf{R}}[g, g']$ is the number of *distinct* ground rules (r, r') of R such that there is a minimal bound (L, D) for (a, r), with D active, such that $D \circ r' = g'$.

If $\mathcal{R}$ is a rule-set, $\mu_{\mathcal{R}}[a, L, a']$ is defined by summation over $\mathcal{R}$.

Thus transition rates need not be defined independently from reaction rates.

Weighting communications in a process calculus

We assign rates only to reaction rules, and *derive* rates for reactions and transitions. In a process calculus, each participant in a communication can modify its rate as follows:

In CCS, a participant is $(\mu_1.P_1 + \cdots + \mu_n.P_n)$. To multiply the rate of a single summand $\mu.P$ by k, replace it by $k{\times}\mu.P$.

This language-extension is easy to derive: in CCS, in place of $k{\times}\mu.P$ we can write simply

$$\overbrace{\mu.P + \cdots + \mu.P}^{k} .$$

The rate increases, just because we have replaced a single summand by a population of k identical summands.

Of course, it wouldn't be *implemented* that way!

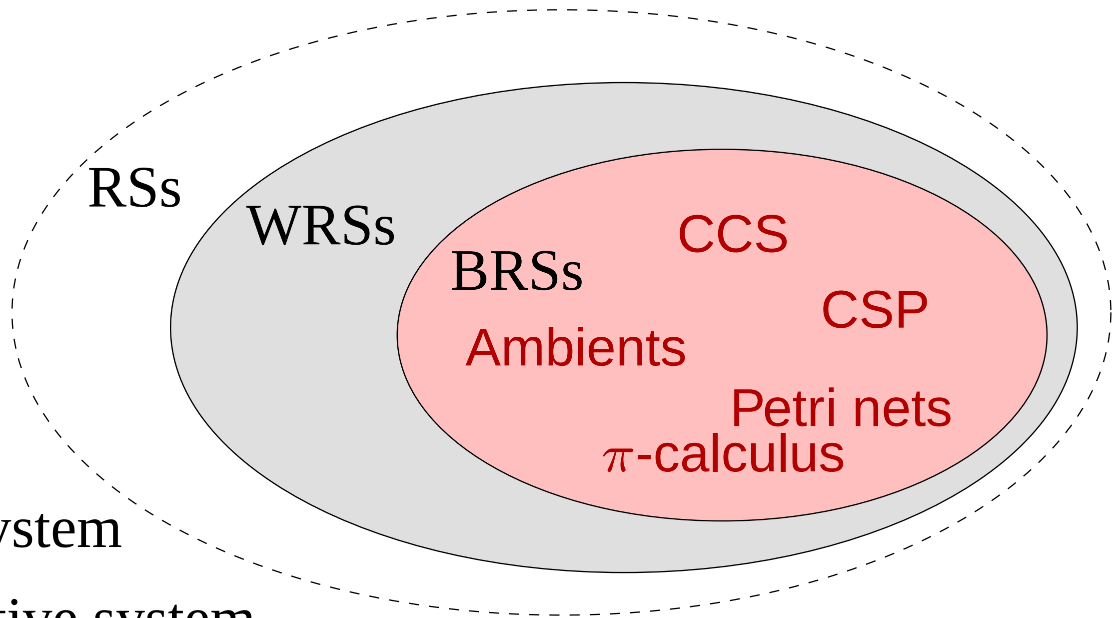
Lecture V

Foundation for behavioural equivalence

Classes of reactive system

We seek behavioural congruence for reactive systems that can control *where* reactions can happen.

Wide reactive systems have just enough notion of *place*.



RS −− reactive system

WRS −−wide reactive system

BRS −−bigraphical reactive system

What is a discrete process?

• This is fundamental question for informatics: compare the question “**What is a computable function?**”.

• But we have to consider much more than the classical notion of computation: non-determinism, non-sequentiality, locality, ...

• Bold attempted answer: **An equivalence class of systems indistinguishable by observation**.

• **Bisimilarity** serves this purpose uniformly for BRSs . . .
. . . and for an even broader class of reactive systems.

• It depends on the humble idea of **tagging**!

Why is the tagging of components useful??

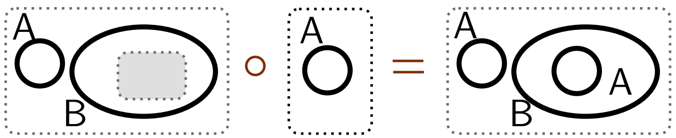
Tagging of subexpressions is used for many purposes in the λ-calculus. The same holds in processes generally:

- Tagging has helped us to count the *number of distinct ways* a reaction $g \longrightarrow g'$ can occur.
- Tagging can keep track of agents through their reactions; this leads to understanding *causality* in bigraphs.
- With tagging, we'll define *minimal labelled transitions*, and thus recover the behaviour of process calculi within bigraphs.

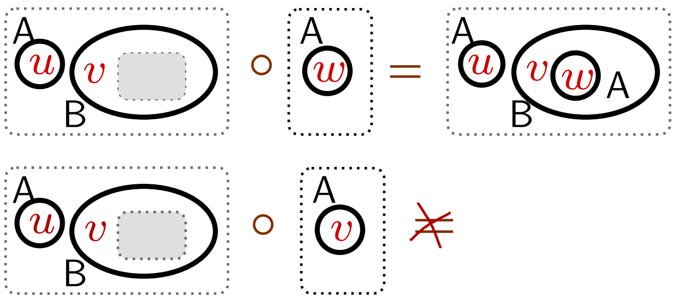
For this purpose, we'll call bigraph *concrete* if it is tagged.

Support, and concrete bigraphs

In $\mathrm{B}\textsc{g}(\Sigma)$:



In $\grave{\mathrm{B}}\textsc{g}(\Sigma)$:



The *concrete* bigraphs $\grave{\mathrm{B}}\textsc{g}(\Sigma)$ are like the *abstract* ones $\mathrm{B}\textsc{g}(\Sigma)$, except:

- every node and edge has a distinct tag;
- in composition and product the tags must be disjoint.

The set of tags for G is its *support*, written $|G|$.

S-categories, in general

Assume an infinite set $\mathcal{S}$ of *support tags*. Then an **s-category** is just like an spm category, except

- Each arrow f has a finite *support set* $|f| \subset \mathcal{S}$.
- For $g \circ f$ or $f \otimes g$ to be defined, we require $|f| \cap |g| = \emptyset$.
- If defined, $|g \circ f| = |f| \uplus |g|$ and $|f \otimes g| = |f| \uplus |g|$.
- The identities id_I and symmetries $\gamma_{I,J}$ have empty support.
- All the spm equations hold when both sides are defined.

An spm category is just an s-category with *empty support sets*!

General reactive system

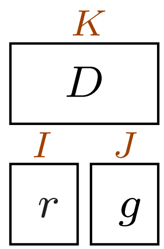
A *general reactive system* $(\grave{\mathbf{C}}, \grave{\mathcal{R}})$ consists of an s-category $\grave{\mathbf{C}}$ and a set $\grave{\mathcal{R}}$ of *ground reaction rules*, each of the form

$$\mathsf{R} = (r,\ r')$$

with *redex* $r : \epsilon \to I$ and *reactum* $r' : \epsilon \to I$. This generates reactions $D \circ r \longrightarrow D \circ r'$ whenever D has inner face I.

But we have lost what it means for a context D to be ***active****!*

An s-category lets us assemble things vertically ($\circ$) and horizontally ($\otimes$). Suppose $r : \epsilon \to I$, with $g : \epsilon \to J$ and $D : I \otimes J \to K$; does D permit a reaction $r \longrightarrow r'$?



$D \circ (r \otimes g)$

Wide reactive system (WRS)

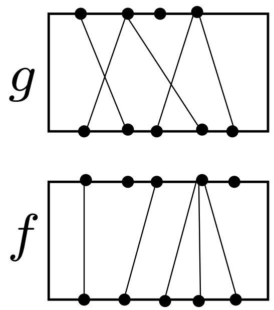
A *wide* reactive system has a *Width* functor and an *Activity* map.

Width of an interface: $Width(I)$ is a finite ordinal $m = \{0, \ldots, m{-}1\}$.
Width of an arrow: $Width(f : I \to J)$ maps $Width(I)$ to $Width(J)$.
Conditions: $Width(g \circ f) = Width(g) \circ Width(f)$, ...

Activity of an arrow:

$\mathsf{Act}(f : I \to J)$ is a subset of $Width(I)$;
if $i \in \mathsf{Act}(f)$ we say "f is *active* at i".

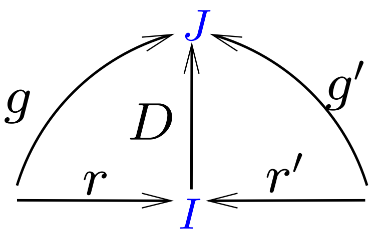
Conditions: $i \in \mathsf{Act}(g \circ f)$ if and only if
$i \in \mathsf{Act}(f)$ and $Width(f)(i) \in \mathsf{Act}(g)$, ...



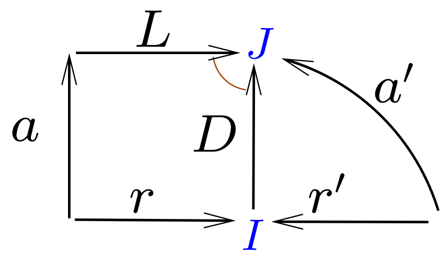
Localised behaviour in a WRS

A *location* of an interface I is a subset of its places $Width(I)$. Denote locations by $\tilde{\imath}$, $\tilde{\jmath}$.

A reaction or transition happens due to a ground rule (r, r') in an active context D.† We index it with $\tilde{\jmath} = Width(D)(Width(I))$, a location of $Width(J)$, as follows:



wide reaction $g \longrightarrow_{\tilde{\jmath}} g'$



wide transition $a \xrightarrow{L}_{\tilde{\jmath}} a'$

†To be precise, the reaction rules and the reaction and transition relations are required to be closed under support equivalence $\simeq$.

How a context helps a reaction

A transition may happen when an agent a and a redex r both occur, perhaps overlapping, within some larger system g. Part of the redex may lie in a. How do we determine the other part?

$$\begin{tikzcd} \cdot \arrow[r, "L"] \arrow[rr, bend left, "E"] & \cdot \arrow[r, "G"] & \cdot \\ \cdot \arrow[u, "a"] \arrow[r, "r"'] & \cdot \arrow[u, "D"] \arrow[ur, bend right, "F"'] & \end{tikzcd}$$

$$g = E \circ a = F \circ r$$

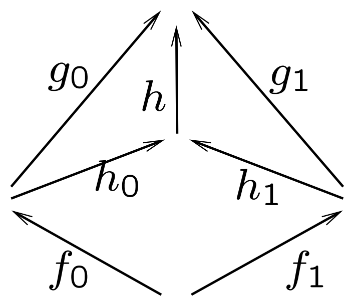
We must find the 'minimal' triple (L, D, G), as shown, making the diagram commute. Then L is the part of r not in a; it will be the label of a transition $a \xrightarrow{L}_{\tilde{\imath}} a'$.

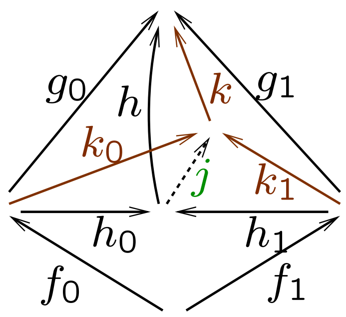
This minimal construction is called a *relative pushout*.
It's a purely static phenomenon. *Does it exist?*

Relative pushouts (RPOs) in an s-category

A cospan $\vec{g} : \vec{I} \to K$ *bounds* a span $\vec{f} : H \to \vec{I}$ if $g_0 \circ f_0 = g_1 \circ f_1$.

A *bound for $\vec{f}$ relative to $\vec{g}$* is a triple $(\vec{h}, h)$ such that $\vec{h}$ bounds $\vec{f}$ and $h \circ h_i = g_i$.





$(\vec{h}, h)$ is a *relative pushout* for $\vec{f}$ relative to $\vec{g}$ if, for any relative bound $(\vec{k}, k)$, there is a unique arrow j for which $j \circ h_i = k_i$ and $k \circ j = h$.

An s-category *has RPOs* if every span $\vec{f}$, given any bound, has an RPO relative to that bound.

Idem pushouts (IPOs), and properties

$\vec{h}$ is an *IPO* for $\vec{f}$ if $(\vec{h}, \mathsf{id})$ is an RPO for $\vec{f}$ relative to $\vec{h}$.
Essential for deriving transitions!

Properties: (assuming RPOs exist)

- Any RPO for $\vec{f}$ relative to $\vec{g}$ is unique up to isomorphism.

- If $(\vec{h}, h)$ is an RPO for $\vec{f}$ to $\vec{g}$, then $\vec{h}$ is an IPO for $\vec{f}$.

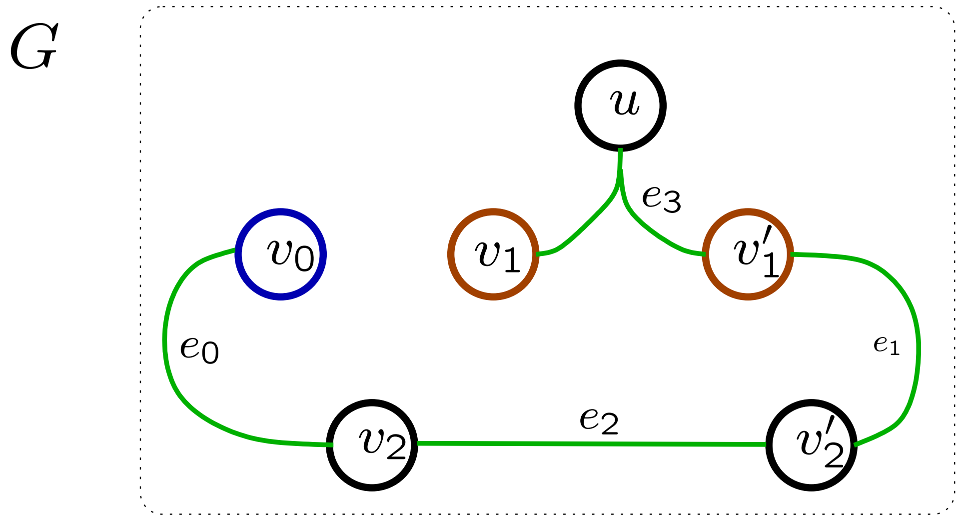
- If $\vec{h}$ is an IPO for $\vec{f}$, then any triple $(\vec{h}, h)$ is an RPO.

- if the two squares are IPOs, so is the rectangle;
- if the rectangle and the left square are IPOs,
  then so is the right square.

$$\begin{tikzcd}
\cdot \arrow[r, "h_0"] & \cdot \arrow[r, "h_1"] & \cdot \\
\cdot \arrow[u, "f_0"] \arrow[r, "g_0"] & \cdot \arrow[u, "f_1"] \arrow[r, "g_1"] & \cdot \arrow[u, "f_2"']
\end{tikzcd}$$

A RPO example in link graphs

Consider a simple link graph G:

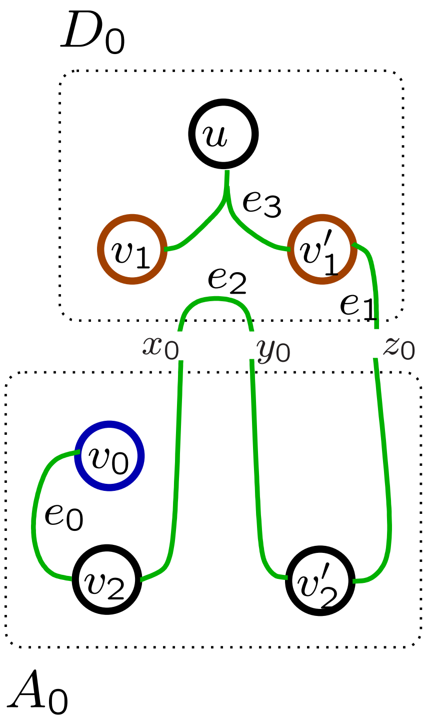


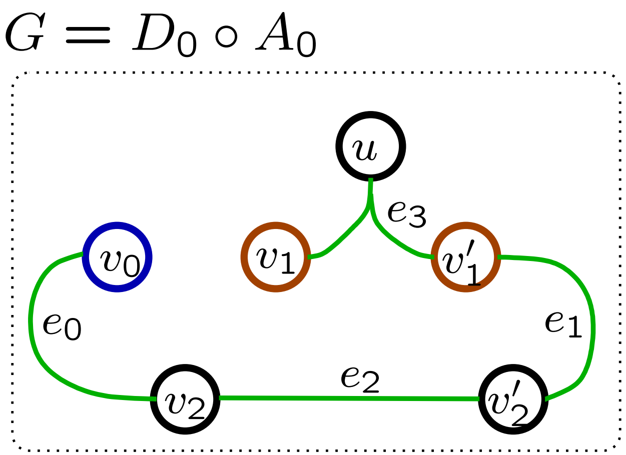
We shall exhibit a span $\vec{A}$, having a bounding cospan $\vec{D}$ such that $G = D_0 \circ A_0 = D_1 \circ A_1$.

Then we find an RPO $(\vec{B}, B)$ for $\vec{A}$ relative to $\vec{D}$.

A RPO example in link graphs (1)

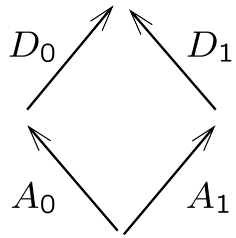
ONE DECOMPOSITION . . .

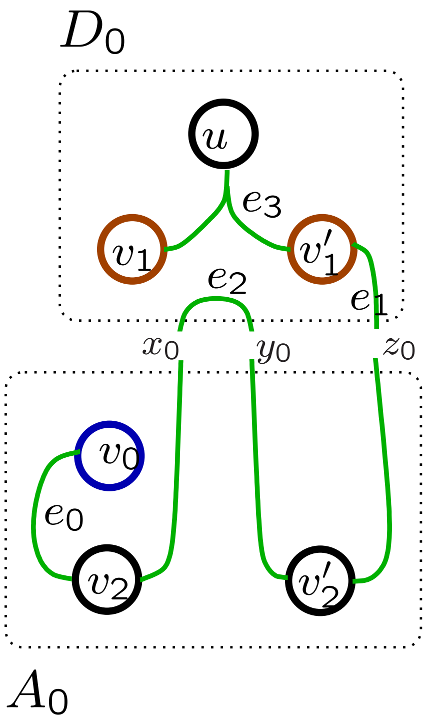


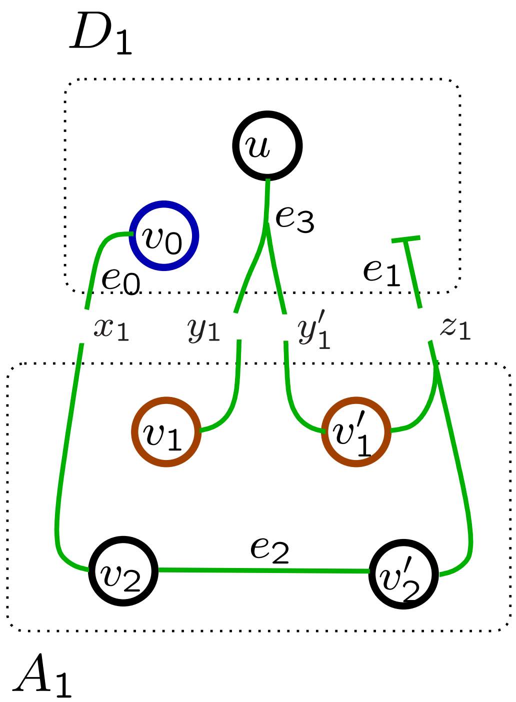


A RPO example in link graphs(2)

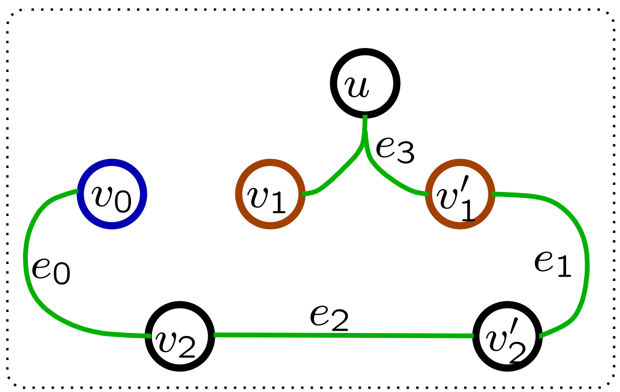
ONE DECOMPOSITION . . . . . . AND ANOTHER







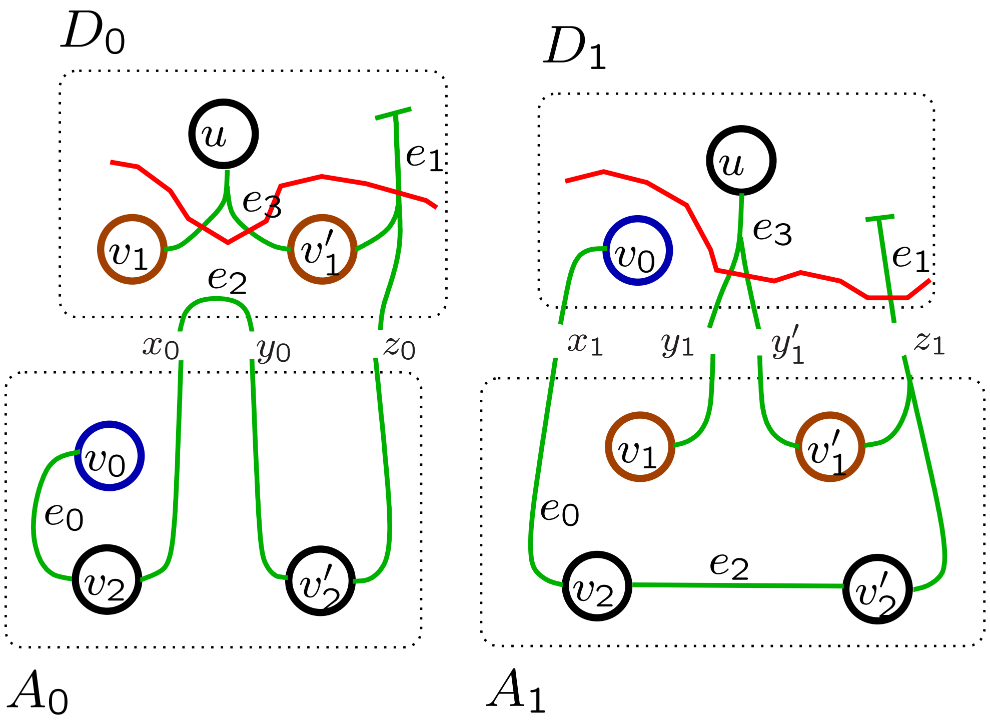
$G = D_0 \circ A_0 = D_1 \circ A_1$



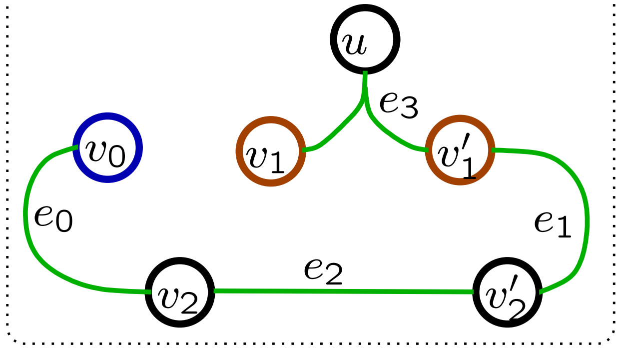
A RPO example in link graphs (3)

DECAPITATE !

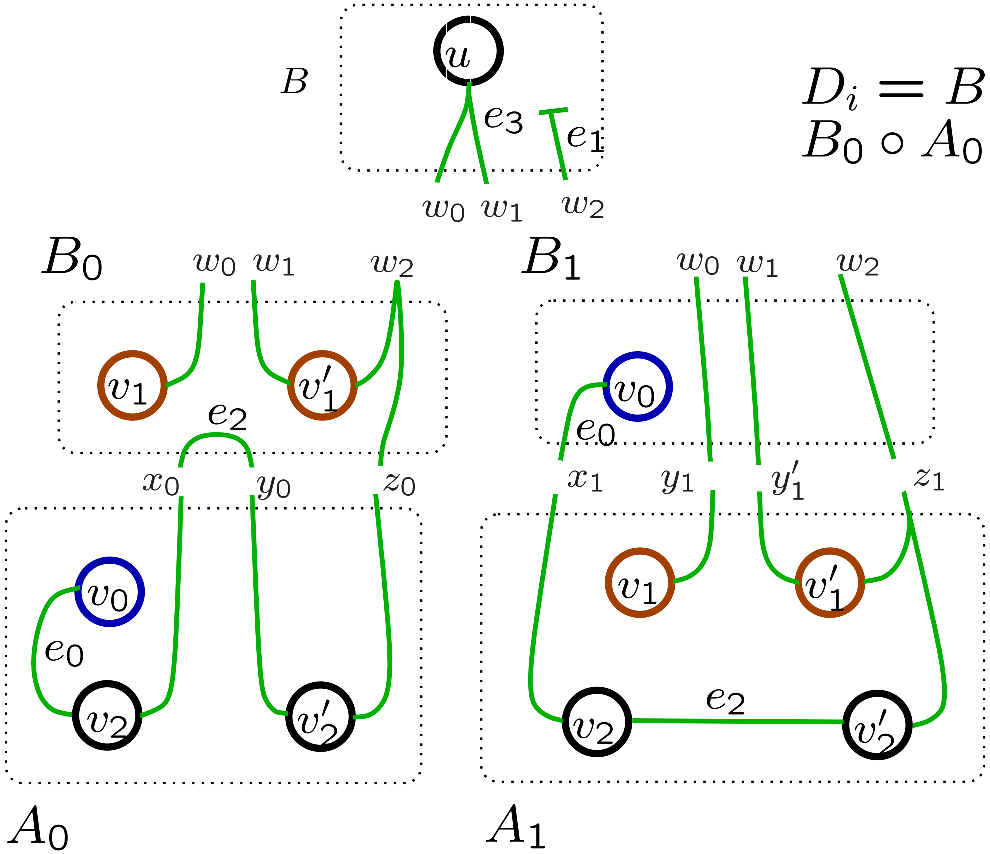
$$\begin{tikzcd} & {} & \\ {} \arrow[ur, "D_0"] & & {} \arrow[ul, "D_1"'] \\ & {} \arrow[ul, "A_0"] \arrow[ur, "A_1"'] & \end{tikzcd}$$



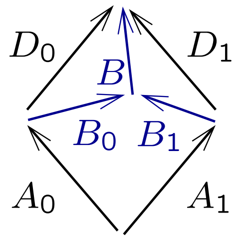
$$G = D_0 \circ A_0 = D_1 \circ A_1$$



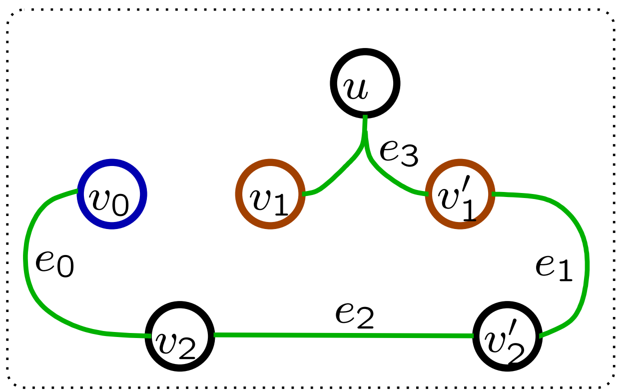
A RPO example in link graphs: completed



$$D_i = B \circ B_i$$
$$B_0 \circ A_0 = B_1 \circ A_1$$



$$G = D_0 \circ A_0 = D_1 \circ A_1$$



Minimal transitions and bisimilarity

In a WRS with RPOs, a *minimal* transition $a \xrightarrow{L}_{\tilde{\jmath}} a'$ is one which

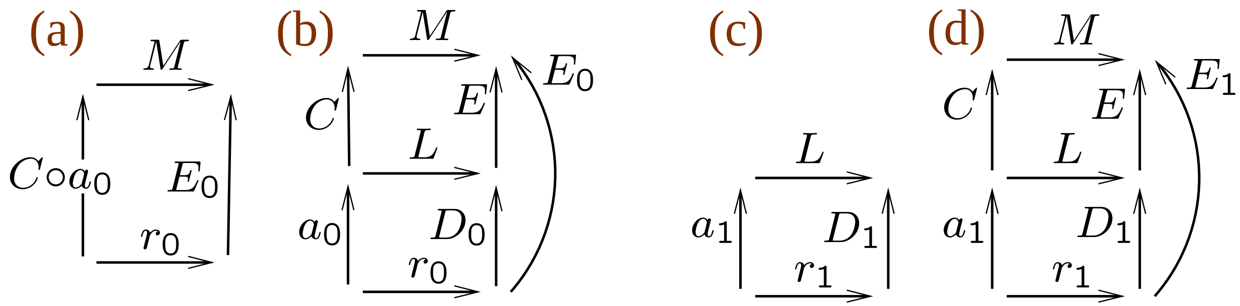
- The agents a, a' are ground arrows;
- The underlying commuting diagram is an IPO.

A *bisimulation* is a symmetric relation $\mathcal{S}$ such that, whenever $a\mathcal{S}b$ and $a \xrightarrow{L}_{\tilde{\jmath}} a'$ with $L \circ b$ defined, there exists a transition $b \xrightarrow{L}_{\tilde{\jmath}} b'$ such that $a'\mathcal{S}b'$.

Agents a and b are *bisimilar*, written $a \sim b$, if there exists a bisimulation $\mathcal{S}$ with $a\mathcal{S}b$.

Congruence for minimal transitions

Theorem: In a WRS with RPOs and minimal transitions, bisimilarity is a congruence: if $a_0 {\sim} a_1$ then $C{\circ}a_0 {\sim} C{\circ}a_1$.



Proof (outline): Establish the bisimulation

$$\mathcal{S} \stackrel{\text{def}}{=} \{(C{\circ}a_0, C{\circ}a_1) \mid a_0{\sim}a_1,\ C \text{ a context } \}\ .$$

1. Let (a) underlie a transition $C{\circ}a_0 \xrightarrow{M}_{\tilde{\jmath}} b'_0$. Take an RPO.
2. The lower square of (b) underlies a transition $a_0 \xrightarrow{L}_{\tilde{\imath}} a'_0$.
3. Then by $\sim$, (c) underlies $a_1 \xrightarrow{L}_{\tilde{\imath}} a'_1$, with $a'_0 \sim a'_1$.
4. By cut-and-paste, (d) underlies a transition $C{\circ}a_1 \xrightarrow{M}_{\tilde{\jmath}} b'_1$. $\square$

How to apply the congruence theorem

The theorem does no apply to an abstract BRS $\mathbf{C}{=}\mathrm{B}\textsc{g}(\Sigma,\mathcal{R})$, because abstract BRSs lack RPOs!

But it applies *indirectly*. In outline:

1. From $\mathbf{C}$, create a concrete BRS $\grave{\mathbf{C}}{=}\,\grave{\mathrm{B}}\textsc{g}(\Sigma,\grave{\mathcal{R}})$, taking $\grave{\mathcal{R}}$ to be all preimages of $\mathcal{R}$ under the support quotient functor.

2. Taking minimal transitions, define bisimilarity in $\grave{\mathbf{C}}$ and prove it a congruence by the theorem.

3. Then, applying the forgetful functor, this yields a transition system and a congruent bisimilarity in $\mathbf{C}$.

This is what we exhibited for CCS, and the same can be done for Petri nets, mobile ambients and CSP.

Lecture VI

Ubiquitous systems: a context for bigraphs

Bigraphs in context: a broad view

Informatic Models form a tower.

Ubiquitous Computing demands a tower whose higher models embody sophisticated concepts: *trust, reflectivity, . . .* .

Bigraphs provide a rigorous basis for the tower, on which to *program, simulate and analyse* behaviour expressed at higher levels.

An informatic model

Entities in a model explain, or are realised by,
entities in the physical world—as in natural science.

ENTITIES

PROGRAMS

realised by

COMPUTERS

An informatic model with behaviour

Entities *and behaviour* in a model explain, or are realised by, entities in the physical world—as in natural science.

ENTITIES	BEHAVIOUR
PROGRAMS	**action on memory, i/o**
realised by	
COMPUTERS	**keyboard & screen events**

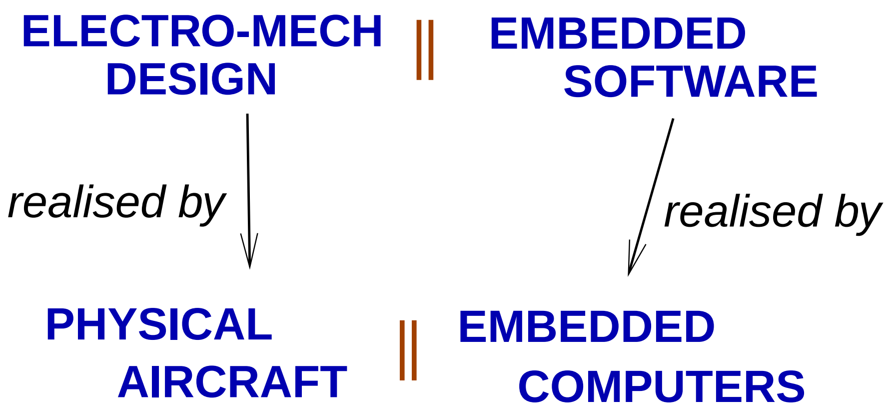
Layered informatic models with behaviour

Entities *and behaviour* in a model explain, or are realised by, entities in the physical world *or in a lower model*.

ENTITIES	BEHAVIOUR
LOGICAL FORMULAE	**valuation as sets & predicates**
specify ↓	↓
PROGRAMS	**action on memory, i/o**
↓	↓
ASSEMBLY CODE	**action on memory, i/o**
↓	↓
HARDWARE DESIGN	**voltage, bitmaps, switching**
realised by ↓	↓
COMPUTERS	**keyboard & screen events**

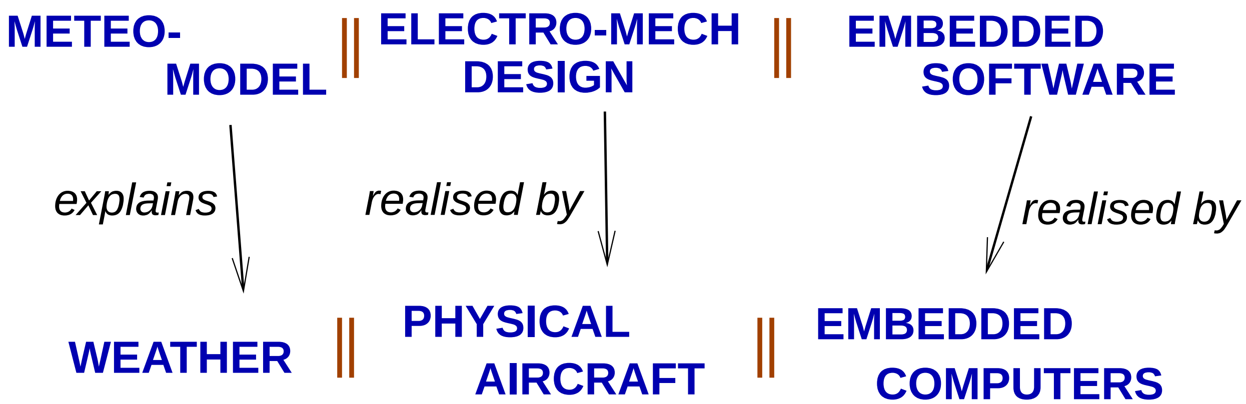
Combining models

Real systems combine interacting parts; we must also combine partial models. Thus, combine models of the electro-mechanical and informatic parts of an aircraft:



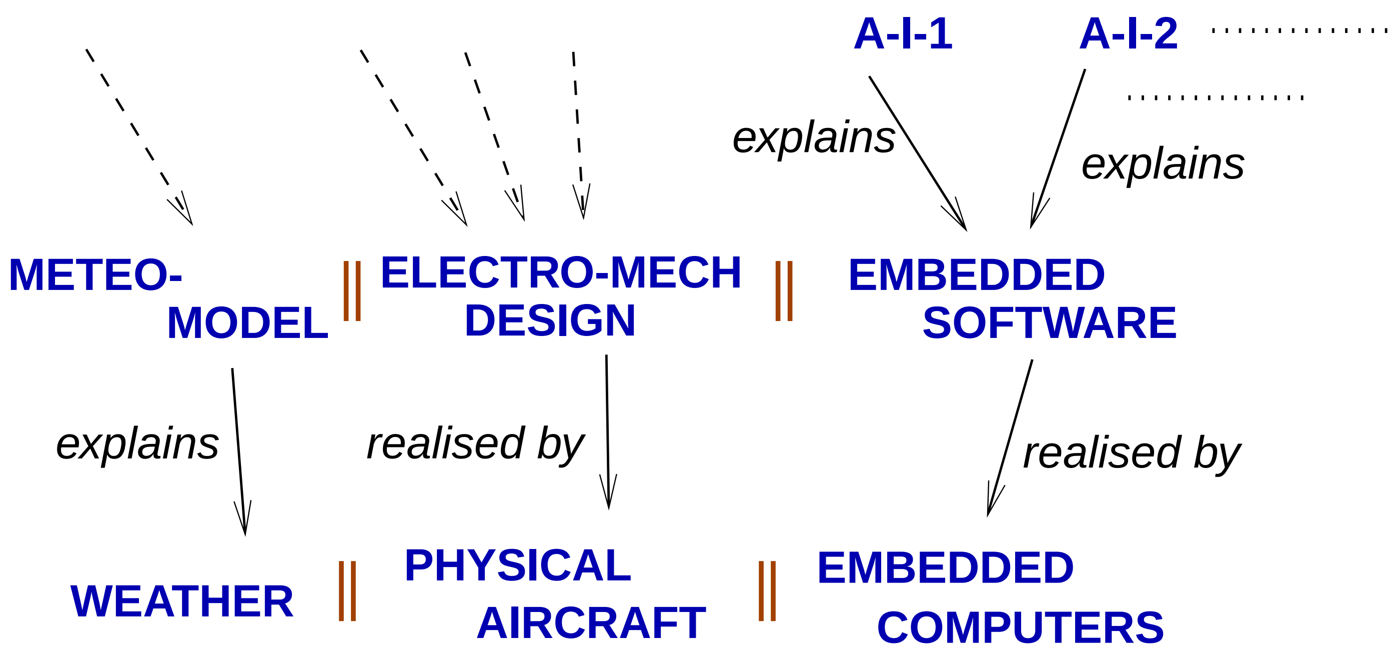
Combining models

Real systems combine interacting parts; we must also combine partial models. Also, combine models of artifactual and natural systems:



Combining models

For a program, we may combine different explanatory models. INRIA did this for the **Airbus** using **abstract interpretation**, following successful analysis of the failure of the Ariane-5 rocket:



Models and their tower

A **model** consists of some *entities*, and their *behaviour*.

EXAMPLE: flowcharts, and how to execute them.

A **tower** of models is built by *explanation* and *combination* :

Model A **explains** model B if
A *abstracts from* or *specifies* B, or if
B *implements* or *realises* A.

EXAMPLE: a specification logic specifies programs.

Model C **combines** models A and B if
its entities and behaviours combine those of A and B.

EXAMPLE: combine distributed programs with a network model.

How do we validate an explanation?

Natural science:

Explanation of reality by a model can only be supported by *observation*. Complete validation impossible (Karl Popper).

Informatics at lowest level:

Similar (e,g. realisation of circuit diagrams by a computer).

Informatics at higher levels:

Higher levels abound in the model tower. Can aspire to *complete validation* between precise models.

PROPOSITION: Informatics is a science just to the extent that it aspires to complete validation.

Scientific status of the Tower of Models

- Useful models, and validations, may well be **informal**
- **Different models** suit different people, including **non-experts**
- **Many instances** of models and validations exist
- Can we derive **languages from models**, not vice-versa?

Two visions of Ubiquitous Computing

Populations of computing entities will be a significant part of our environment, performing tasks that support us, and we shall be largely unaware of them. (after Mark Weiser, 1994)

> In the next five to ten years the computer will be erased from our consciousness. We will simply not talk about it any longer, we will not read about it, *apart from experts of course*.
>
> *(my emphasis)* Joseph Weizenbaum (2001)

. . . . . . **and my vision:**

Ubiquitous computing will **empower us**, if we **understand** it!

Qualities of a ubiquitous computing system (UCS)

What is new about a UCS?

- It will continually make **decisions** hitherto made by us
- It will be **vast**, maybe 100 times today's systems
- It must continually **adapt**, on-line, to new requirements
- Individual UCSs will **interact** with one another

Can traditional software engineering cope?

Concepts for Ubicomp

Each ubicomp **domain**, hence each **model**, will involve several concepts. Here are a few:

provenance obligations
self-management
locality intentions specification data-protection
beliefs continuous space simulation
authorisation
encapsulation mobility
continuous time role
compilation policy failure
delegation reflectivity
verification
stochastics negotiation
connectivity
trust security authenticity

Managing the conceptual overload

- Using **bigraphs**, Define the *Ubiquitous Abstract Machine* (**UAM**) in terms of locality, connectivity, mobility, stochastics.
- Build a *model tower* above **UAM**, layering the concepts.

What's the point of a Grand Challenge in informatics?

To make applications that startle the world?

(e.g. beating a grandmaster at chess)

OR

To organise the principles for an engineering science?

The first alone may (or may not) spin off science

The two together will embed computing
in our scientific culture

....oooo0000OOOO0000oooo....