Formal Molecular Biology

According to V. Danos & C. Laneve

Jérôme Caffaro
jerome.caffaro@epfl.ch

Jean-Philippe Pellet
jean-philippe.pellet@epfl.ch

18th May, 2005

Outline

Introduction

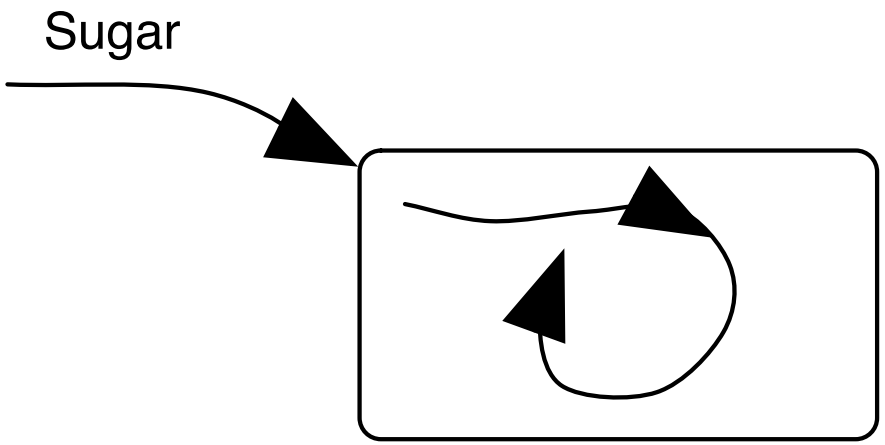
- Goal: apply formal methods to describe and analyze biological networks at the molecular level
  - To do so, define a formal language for proteins interaction: the κ-calculus
  - Then try to define a finer-grained model based on this language: the $m\kappa$-calculus
  - Finally encode $m\kappa$-calculus into π-calculus

For this presentation...

Today we will focus on the first and second languages, the κ-calculus and the $m\kappa$-calculus.

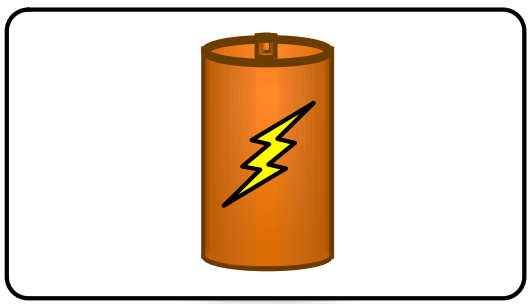
General Considerations & Motivations

- The cell is a billion moving pieces implementing life



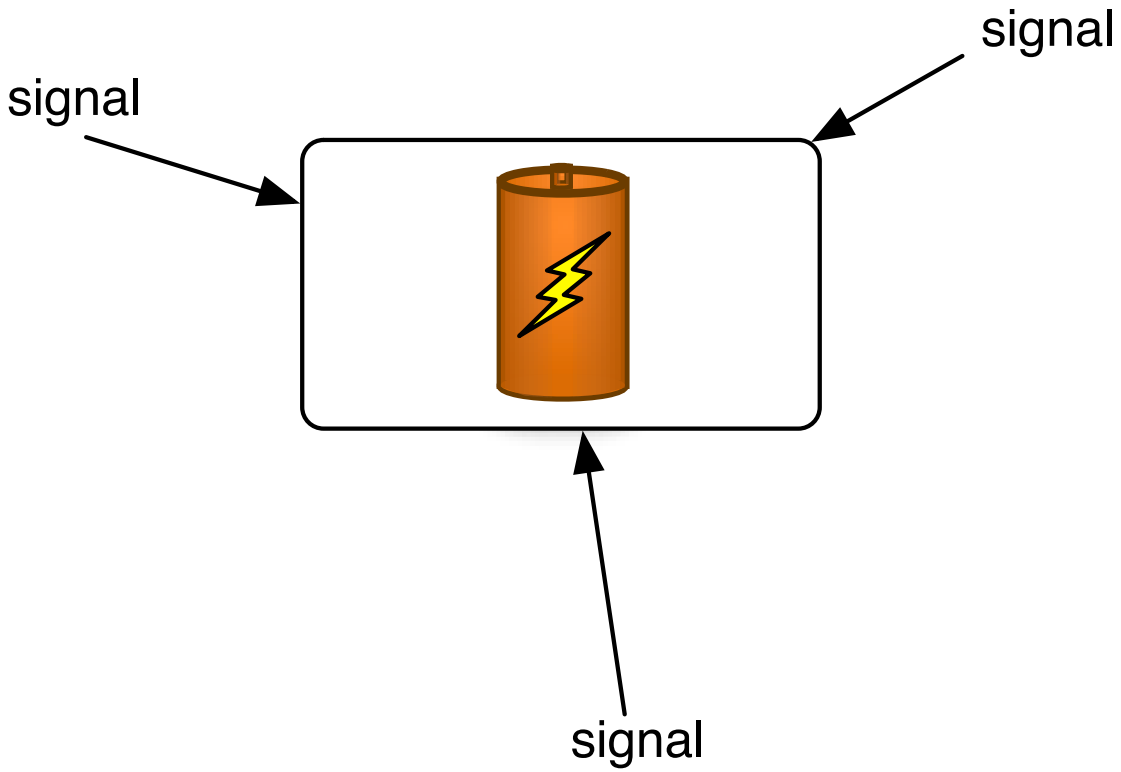
General Considerations & Motivations

- With energy, the cell can detect, collect and compare signals



General Considerations & Motivations

- With energy, the cell can detect, collect and compare signals



- $\Rightarrow$ lots of interaction when considering networks of cells!

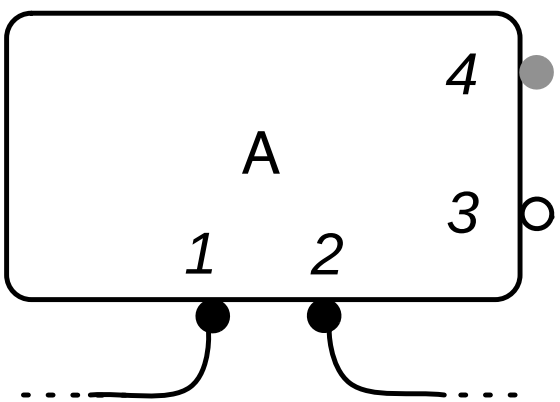
More Motivations!

- Computation in a cell is concurrent and asynchronous
  - $\Rightarrow$ The cell needs to implement synchronisation
- The system semantic depends on stochastic responses but looks deterministic at macroscopic level
- Values are continuous, but discrete states and choices can be considered
- $\Rightarrow$ some work for specialists in concurrency!

A Visual Notation for κ-Calculus

- Let's try to define a visual notation for κ-calculus based on proteins
- We need to express the combinatorics of the interaction between proteins
  - $\Rightarrow$ Abstract the real proteins!

A Visual Notation for κ-Calculus

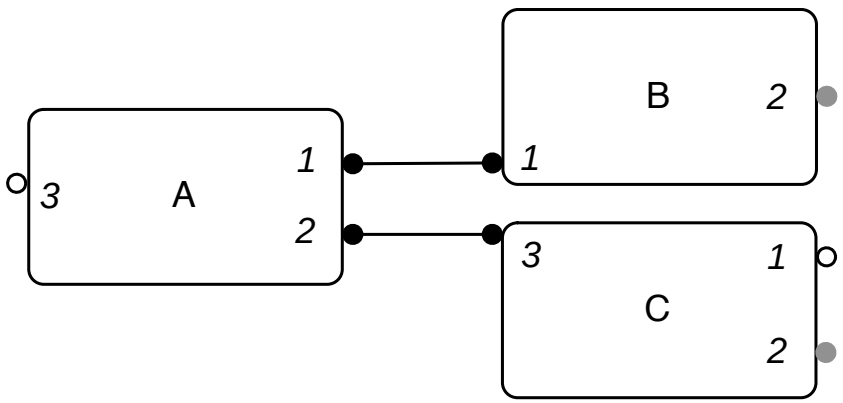


Definition (Sites)

Points of connection to a protein.

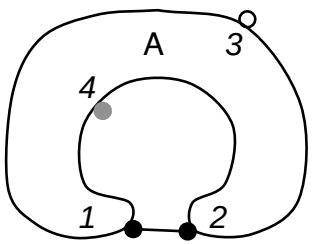
- *bound site*
- *hidden site*
- *visible site*

Proteins Interactions

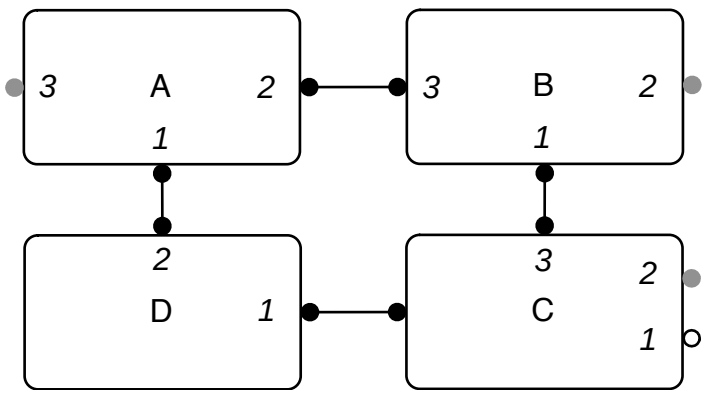


- We can connect proteins to create **complexes**
- Collections of proteins and complexes are called **solutions**
- When the solution has a special shape (= *reactant*), it can evolve by means of **reactions**

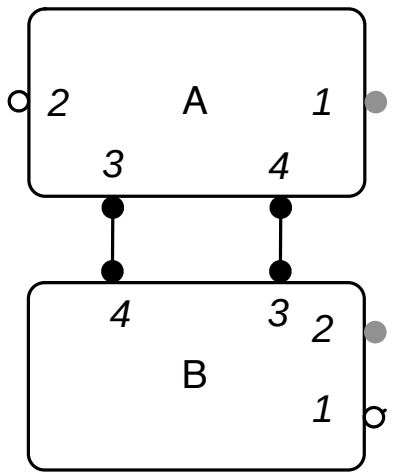
Connection Examples



a self-complexation



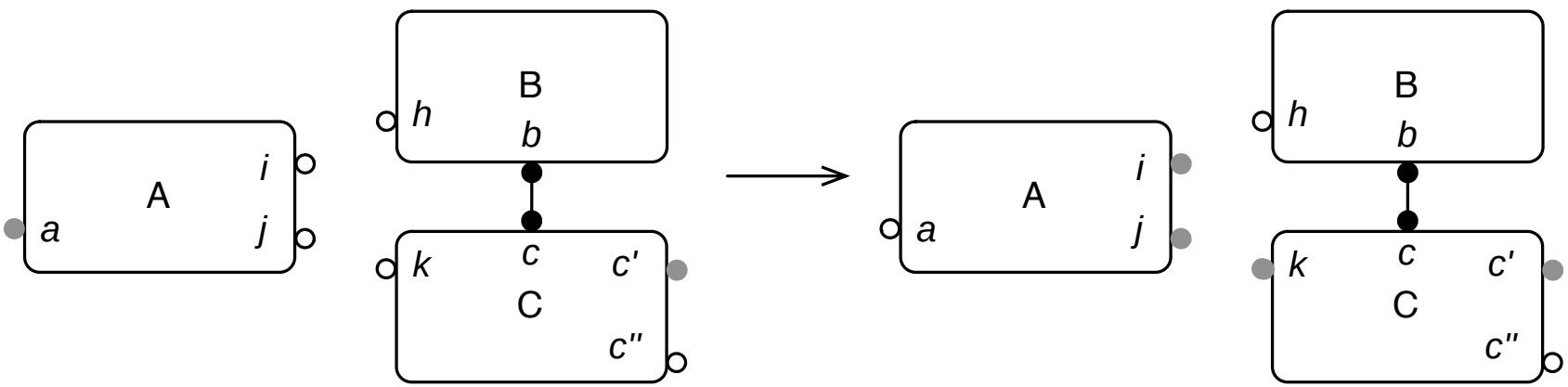
a ring-complex



a double-contact

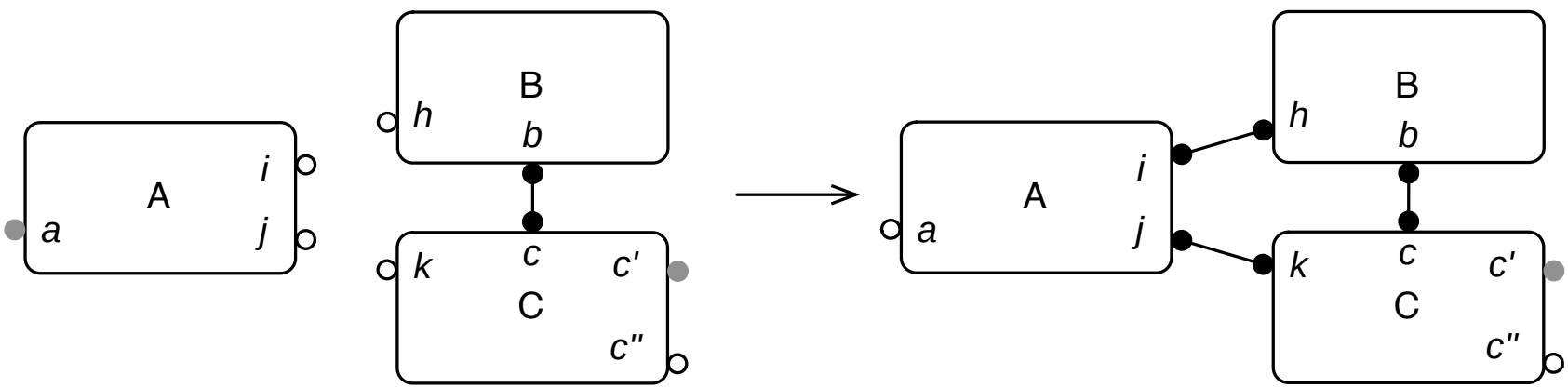
Examples of Reactions

Activation



Examples of Reactions

Complexation

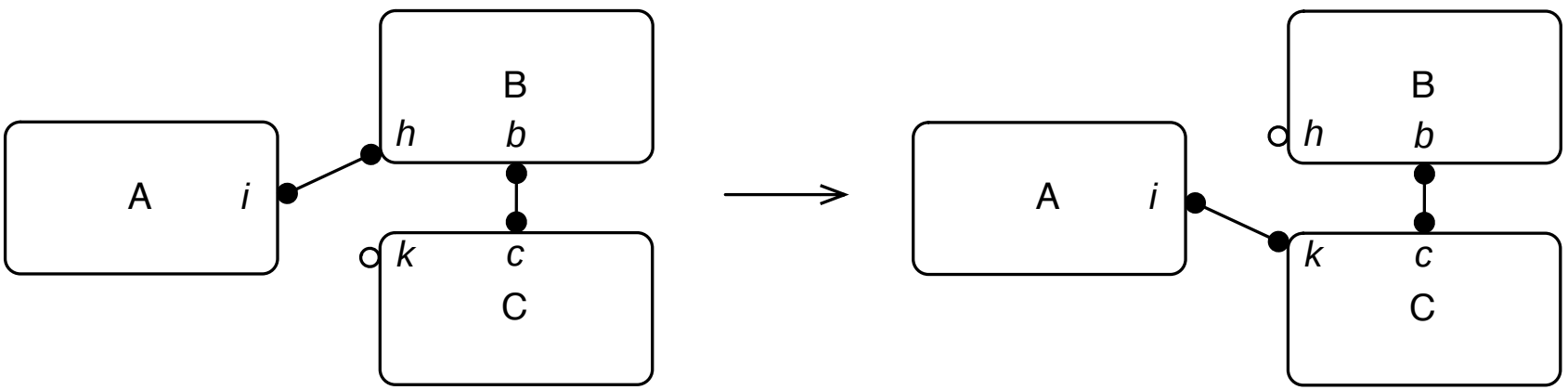


Possible Reactions

- Previous *activation* example shows multiple reaction in one step. Not possible as such in reality
  - We should not be able to activate a site without contact between proteins
  - We cannot consider such reaction as a primitive for κ-calculus
- κ-calculus will roughly only be about **complexations** and **decomplexations**

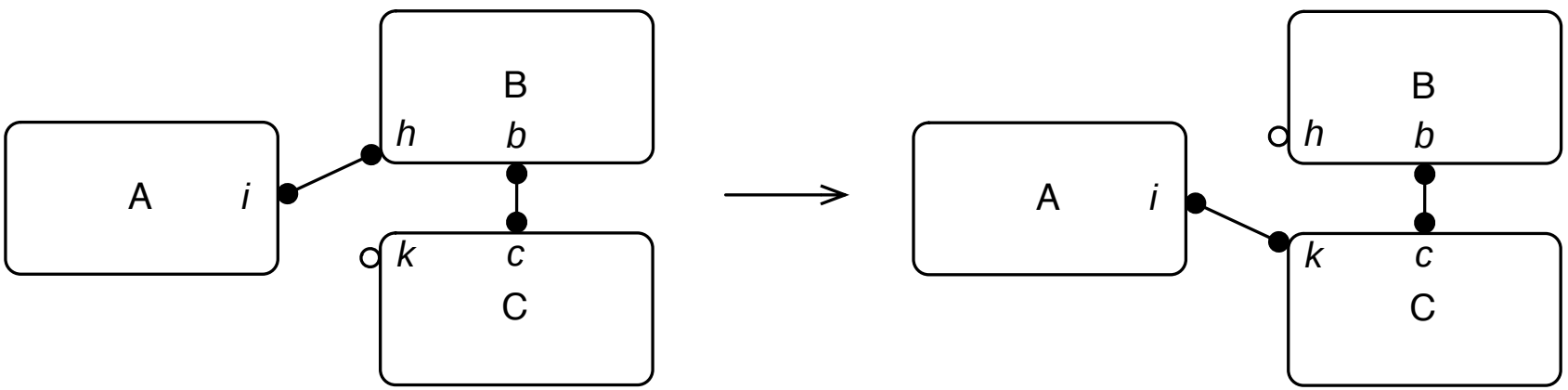
Other Forbidden Atomic Reaction

Edge-flipping



Other Forbidden Atomic Reaction

- Previous *edge-flipping* breaks monotonicity
  - $\Rightarrow$ We should not create and edge and remove another at the same time



Outline

κ-calculus

- Now that we have had a visual approach to the calculus, let's see an algebraic notation
- Try to stay in the classical style of the π-calculus
- We will only need parallel composition & name creation

The Syntax of κ-Calculus

The syntax relies on

- a countable set of *protein names* $\mathcal{P}$, ranged over by A, B, C, ...
- a countable set of *edge names* $\mathcal{E}$, ranged over by x, y, z, ...
- a *signature map*, written $\mathfrak{s}$, from $\mathcal{P}$ to natural number $\mathbb{N}$.
  - $\Rightarrow \mathfrak{s}(A)$ is the number of sites of A and the pair (A, i) is a *site* of A

Interface

Definition (Interface)

Partial map from $\mathbb{N}$ to $\mathcal{E} \cup \{\text{h,v}\}$ ranged over by ρ, σ, ...
A site (A, i) is said to be:

- *visible* if $\rho(i) = v$
- *hidden* if $\rho(i) = h$
- *bound* if $\rho(i) \in \mathcal{E}$

Interface are used to depict partial states of A's sites.

interface $\approx$ state, but with that notation, we emphasize the notion of interaction capabilities of the protein.

Example of Interface

if A is such that $\mathfrak{s}(A) = 3$, then $\rho(1) = v$, $\rho(2) = h$, $\rho(3) = x$ is a well defined interface map for A that declares site 1 to be visible, site 2 to be hidden and site 3 to be bound to some name x.
We write:

$$\rho = 1 + \overline{2} + 3^x$$

Syntax of a Solution $\mathcal{S}$

$\mathcal{S} :=$		**solution**
	0	empty solution
	$A(\rho)$	protein
	$\mathcal{S}, \mathcal{S}$	group
	$(\nu x)(\mathcal{S})$	new

Abbreviation: $(\nu x_1, \ldots, x_n)(\mathcal{S})$ or $(\nu \tilde{x})(\mathrm{S})$ instead of $(\nu x_1) \ldots (\nu x_n)(\mathcal{S})$

Syntax

- The "new" operator is a binder: in $(\nu x)(\mathcal{S})$, $\mathcal{S}$ is the *scope* of the binder (νx)
- We inductively define the set $\mathsf{fn}(\mathcal{S})$ of *free names* in a solution $\mathcal{S}$:

$$\begin{aligned} \mathsf{fn}(0) &= \emptyset \\ \mathsf{fn}(A(\rho)) &= \mathsf{fn}(\rho) \\ \mathsf{fn}(\mathcal{S}, \mathcal{S}') &= \mathsf{fn}(\mathcal{S}) \cup \mathsf{fn}(\mathcal{S}') \\ \mathsf{fn}((\nu x)(\mathcal{S})) &= \mathsf{fn}(\mathcal{S}) \setminus \{x\} \end{aligned}$$

- An occurrence of x in $\mathcal{S}$ is *bound* if it occurs in a sub-solution which is in the scope of the binder x.
- A solution $\mathcal{S}$ is *closed* if all occurrences of names in $\mathcal{S}$ are bound ($\approx$ if $\mathsf{fn}(S) = \emptyset$).

Example

$$\mathcal{S} = C(1^x + 2), (\nu x)(A(1^x + 2 + \overline{3}), B(1 + 2^x))$$

both occurrences of x in A and B are bound, while the occurrence in C is outside the scope of (νx), and hence is not bound in $\mathcal{S}$. $\mathrm{fn}(\mathcal{S}) = \{x\}$, and $\mathcal{S}$ is not closed.

Structural Congruence

- We now have a precise but too much rigid notation:
  - $\Rightarrow$ it separates solutions that we do not want to distinguish for semantic reasons
- Introduce an equivalence relation between solutions, the *structural congruence*

Definition of Structural Congruence

Definition (Structural Congruence)

Structural congruence, written $\equiv$, is the least equivalence closed under syntactic conditions, containing α-equivalence (injective renaming of bound variables), taking "," to be associative (as the choice of symbols suggests) and commutative, with 0 as neutral element, and satisfying the scope laws:

$$\begin{array}{rcll} (\nu x)(\nu y)(\mathcal{S}) & \equiv & (\nu y)(\nu x)(\mathcal{S}), & \\ (\nu x)(\mathcal{S}) & \equiv & \mathcal{S} & \text{when } x \notin \mathsf{fn}(\mathcal{S}), \\ (\nu x)(\mathcal{S}),\, \mathcal{S}' & \equiv & (\nu x)(\mathcal{S},\, \mathcal{S}') & \text{when } x \notin \mathsf{fn}(\mathcal{S}'). \end{array}$$

For example, we have that

$$\begin{array}{rl} \mathcal{S} & = C(1^x + 2),\, (\nu x)(A(1^x + 2 + \overline{3}),\, B(1 + 2^x)) \\ & \equiv (\nu y) C(1^x + 2),\, (A(1^y + 2 + \overline{3}),\, B(1 + 2^y)) \quad = \mathcal{T} \end{array}$$

Using structural congruence, we can define *connectedness*:

- $\mathcal{A}(\rho)$ is connected;
- if $\mathcal{S}$ is connected so is $(x)(\mathcal{S})$
- if $\mathcal{S}$ and $\mathcal{S}'$ are connected and $\mathsf{fn}(\mathcal{S}) \cap \mathsf{fn}(\mathcal{S}') \neq \emptyset$ then $\mathcal{S}, \mathcal{S}'$ is connected;
- if $\mathcal{S}$ is connected and $\mathcal{S} \equiv \mathcal{T}$ then $\mathcal{T}$ is connected.

Graph-likeness

- The language defined up to now allows to define objects that we would not be able to draw as graph
  - For instance, in $(\nu x)(A(1^x))$, x would bind only one site of the protein...
- $\Rightarrow$ We need to put some more restriction on the language

Graph-likeness

Definition (Graph-likeness)

A solution is said to be *graph-like* iff:

- free names occur at most twice in $\mathcal{S}$;
- binders in $\mathcal{S}$ bind either zero or two occurrences.

if in addition free names occurs exactly twice in $\mathcal{S}$, we say that $\mathcal{S}$ is *strongly graph-like*.

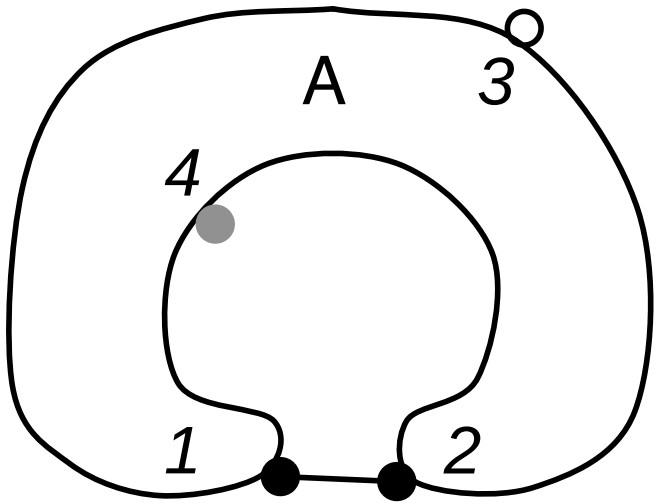
From Graph-like Solutions to Graph With Sites

Definition ($[\![.]\!]_g$)

Let $[\![.]\!]_g$ be the following function from graph-like solutions to graphs with sites:

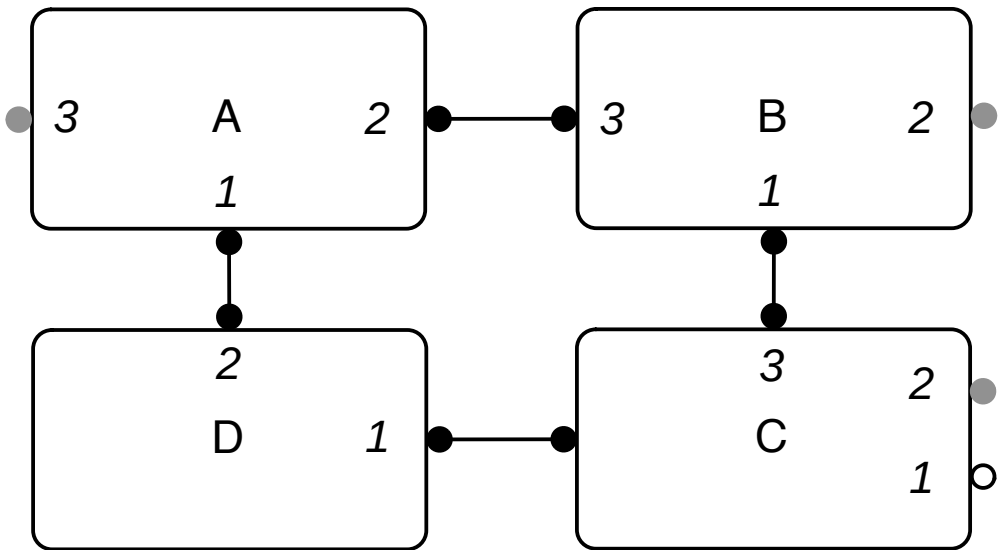
- $[\![A(\rho)]\!]_g$ is the graph with a single node labeled A, sites in $\{1, \ldots, \mathfrak{s}(A)\}$, bound sites k being labeled by $\rho(\mathrm{k})$, and free sites being in the state prescribed by ρ;
- $[\![\mathcal{S}, \mathcal{S}']\!]_g$ is the union graph of $[\![\mathcal{S}]\!]_g$ and $[\![\mathcal{S}']\!]_g$, with sites labeled with the same name being connected by an edge, and their common name erased;
- $[\![(\nu x)(\mathcal{S})]\!]_g$ is $[\![\mathcal{S}]\!]_g$.

Examples (1)



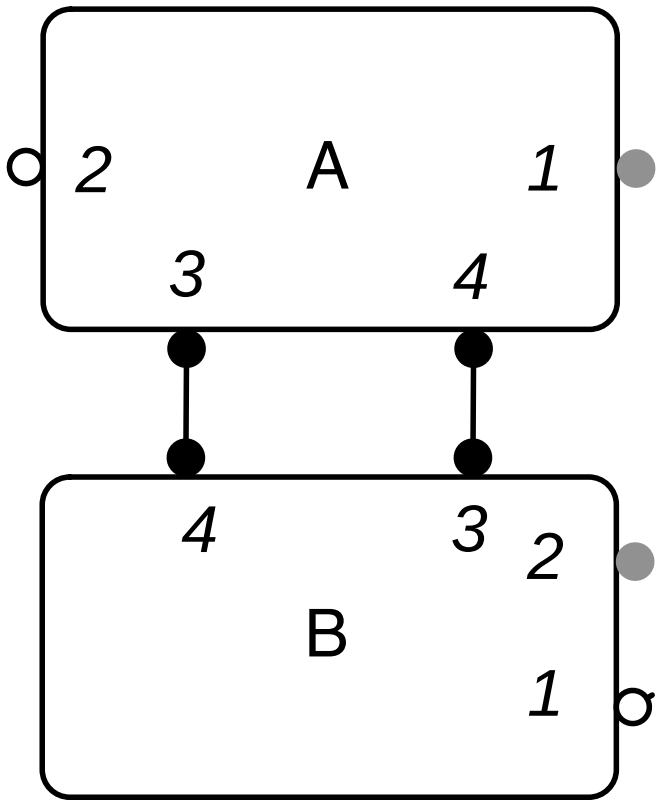
$$(\nu x)(A(1^x + 2^x + 3 + \overline{4}))$$

Examples (2)



$$(\nu wxyz)(A(1^x+2^x+3), B(1^z+\overline{2}+3^y), C(1+\overline{2}+3^z+4^w), D(1^w+2^x))$$

Examples (3)



$$(\nu xy)(A(\overline{1} + 2 + 3^x + 4^y), B(1 + \overline{2} + 3^y + 4^x))$$

Outline

The Growth Relation $\leq$ (I): Motivation

Why define growth relation?

- Restrict possible reactions
- Later, define *monotonicity* for reactions using growth relation

The growth relation $\leq$

- Defined (now) on partial interfaces
- Interpretation:
  $\rho \leq \rho' \quad \widehat{=} \quad \rho'$ has more connections than ρ
- Parametrized by set of names $\tilde{x}$
- $\tilde{x}$ represents the new edges of the interface

The Growth Relation $\leq$ (II): Inductive Definition

$$(\textsc{create}): \quad \frac{x \in \tilde{x}}{\tilde{x} \vdash \imath \leq \imath^{x}}$$

$$(\textsc{hv-switch}): \quad \frac{}{\tilde{x} \vdash \overline{\imath} \leq \imath}$$

$$(\textsc{vh-switch}): \quad \frac{}{\tilde{x} \vdash \imath \leq \overline{\imath}}$$

$$(\textsc{reflex}): \quad \frac{\tilde{x} \cap \mathsf{fn}(\rho) = \emptyset}{\tilde{x} \vdash \rho \leq \rho}$$

$$(\textsc{sum}): \quad \frac{\tilde{x} \vdash \rho \leq \rho' \qquad \tilde{x} \vdash \sigma \leq \sigma'}{\tilde{x} \vdash \rho + \sigma \leq \rho' + \sigma'}$$

The Growth Relation $\leq$ (III): Comments

Suppose $\tilde{x} \vdash \rho \leq \rho'$.

- Only visible sites in ρ can be bound in ρ'
- Unbound sites in ρ can be toggled from visible to hidden and conversely in ρ'
- $\mathrm{dom}(\rho) = \mathrm{dom}(\rho')$, i.e., both interface describe same sites
- Sites bound in ρ can't be unbound in ρ'
- Created edges in ρ' have to belong to $\tilde{x}$ and their names must be fresh (not used in ρ)
- $\leq$ is *not* transitive

The Growth Relation $\leq$ (IV): Extension

- $\leq$ defined only on (partial) interfaces
- Extend definition to groups of proteins

Definition (Pre-Protein)

A *pre-protein* $A(\rho)$ is a protein defined by a partial interface ρ, i.e. not all sites of A are described in ρ.

$\Rightarrow$ Write proteins more concisely

Definition (Pre-Solution)

A *pre-solution* is a group of pre-proteins.

$\Rightarrow$ Describe only sites that are involved in a reaction

The Growth Relation for Pre-Solutions (I)

We extend the growth relation to pre-solutions:

$$(\textsc{nil}): \quad \frac{}{\tilde{x} \vdash 0 \leq 0} \qquad \text{(0 is the empty solution)}$$

$$(\textsc{group}): \quad \frac{\tilde{x} \vdash \mathcal{S} \leq \mathcal{S}' \qquad \tilde{x} \vdash \rho \leq \rho' \qquad \mathsf{dom}(\rho') \subseteq \mathfrak{s}(A)}{\tilde{x} \vdash \mathcal{S}, A(\rho) \;\leq\; \mathcal{S}', A(\rho')}$$

$$(\textsc{synth}): \quad \frac{\tilde{x} \vdash \mathcal{S} \leq \mathcal{S}' \qquad \mathsf{fn}(\rho) \subseteq \tilde{x} \qquad \mathsf{dom}(\rho) = \mathfrak{s}(A)}{\tilde{x} \vdash \mathcal{S} \;\leq\; \mathcal{S}', A(\rho)}$$

$\mathcal{S}, A(\rho)$ is the (pre-)solution $\mathcal{S}'$ obtained by the addition of $A(\rho)$ to $\mathcal{S}$.

The Growth Relation for Pre-Solutions (II): Comments

Suppose $\tilde{x} \vdash \mathcal{S} \leq \mathcal{S}'$.

- Interpretation: new edges have been created in $\mathcal{S}'$
- The (SYNTH) rule also allows creation of new proteins (with full interfaces)
- **Lemma:** $\mathrm{fn}(\mathcal{S}) = \mathrm{fn}(\mathcal{S}') \setminus \tilde{x}$ and $\mathrm{fn}(\mathcal{S}') \subseteq \mathrm{fn}(\mathcal{S}) \cup \tilde{x}$
- **Proof:** Induction on definition of $\leq$ for interfaces. Induction on definition of $\leq$ for pre-solutions.

Outline

Biological Reactions (I): Definition

Let $\mathcal{S}, \mathcal{S}'$ be two pre-solutions.

$\mathrm{r}_1 : \mathcal{S} \to (\nu\tilde{x})\mathcal{S}'$ is a *monotonic reaction* iff:

- $\tilde{x} \vdash \mathcal{S} \leq \mathcal{S}'$
- $\mathcal{S}$ and $(\nu\tilde{x})\mathcal{S}'$ are graph-like
- $\mathcal{S}'$ is connected
- **Lemma:** $\mathsf{fn}(\mathcal{S}) = \mathsf{fn}((\nu\tilde{s})\mathcal{S}') \stackrel{\mathsf{def}}{=} \mathsf{fn}(\mathrm{r}_1)$

$\mathrm{r}_2 : (\nu\tilde{x})\mathcal{S} \to \mathcal{S}'$ is an *antimonotonic reaction* iff:

- its dual $\mathcal{S}' \to (\nu\tilde{x})\mathcal{S}$ is monotonic
- **Lemma:** $\mathsf{fn}((\nu\tilde{x})\mathcal{S}) = \mathsf{fn}(\mathcal{S}') \stackrel{\mathsf{def}}{=} \mathsf{fn}(\mathrm{r}_2)$

A reaction which is either monotonic or antimonotonic is called a *biological reaction*.

Biological Reactions (II): Comments

The left handside solution of a biological reaction is called the *reactant* and the right handside the *product*.

- A monotonic reaction only creates new bounds and/or proteins in the solution
- Its product must be connected, i.e., bound
- Similarly, an antimonotonic reaction only deletes bounds and/or proteins
- Its reactant must be connected
- Bound names of a biological reaction are the created/deleted edges
- Free names correspond to the untouched bounds

Biological Reactions (III): Interpretation & Justification

Monotonicity and antimonotonicity (incl. connectedness requirement) impose serious restrictions on possible reactions.

- Trying to define a reaction as atomically as possible
- Must not "hide" certain aspects of a reaction in the syntax, make as many biological/chemical "transitions" as possible explicitly visible directly in κ
- Example: edge-flipping reaction. Lacks monotonicity; we are not told everything
- More complex reactions described through *transition systems*
- Is it atomic enough? Why not model only binary interactions?
- $\Rightarrow$ $m\kappa$-calculus does this

Renamings

Definition (Renaming)

A *renaming* r is a partial finite injection on $\mathcal{E} \cup \{h, v\}$, which is the identity on $\{h, v\}$ and maps $\mathcal{E}$ onto $\mathcal{E}$.

- Allows to rename protein bounds without touching the hidden or visible sites

Matching Biological Reactions (I): Definition$_1$

Definition (Matching solutions (monotonic))

Let $\mathcal{R} \rightarrow (\nu\tilde{x})\mathcal{P}$ be a monotonic reaction, and $\mathcal{S}, \mathcal{T}$ be two solutions.

$$\mathcal{S}, \mathcal{T} \text{ match } \mathcal{R} \rightarrow (\nu\tilde{x})\mathcal{P} \quad \Leftrightarrow \quad \mathcal{S}, \mathcal{T} \models \mathcal{R} \rightarrow (\nu\tilde{x})\mathcal{P}$$

$\Leftrightarrow$ $\mathcal{S}$ contains the same number of proteins as $\mathcal{R}$,
$\mathcal{T}$ contains the same number of proteins as $\mathcal{P}$,
$\exists$ a renaming r and, $\forall$ proteins $\exists$ partial interfaces ξ_i, such that interfaces in $\mathcal{S}$ and $\mathcal{T}$ are equal to those in $\mathcal{P}$ and $\mathcal{R}$ renamed with r and extended with ξ_i

Matching Biol. Reactions (II): Def.$_2$ & Interpretation

Definition (Matching solutions (antimonotonic))

Let $(\nu\tilde{x})\mathcal{R} \to \mathcal{P}$ be a monotonic reaction, and $\mathcal{S}, \mathcal{T}$ be two solutions.

$$\mathcal{S}, \mathcal{T} \text{ match } (\nu\tilde{x})\mathcal{R} \to \mathcal{P} \quad \Leftrightarrow \quad \mathcal{S}, \mathcal{T} \models (\nu\tilde{x})\mathcal{R} \to \mathcal{P}$$
$$\Leftrightarrow \mathcal{T}, \mathcal{S} \models \mathcal{P} \to (\nu\tilde{x})\mathcal{R}$$

$\mathcal{S}, \mathcal{T} \models \mathcal{R} \to (\nu\tilde{x})\mathcal{P}$ means:

- $\mathcal{S}$ and $\mathcal{T}$ are two solutions which can be partially described using the pre-solutions $\mathcal{R}$ and $\mathcal{P}$ (incl. possible renamings)
- The solution $\mathcal{S}$ can be transformed to a solution $\mathcal{T}$ using the biological reaction specified by $\mathcal{R} \to (\nu\tilde{x})\mathcal{P}$

The Transition Relation $\rightarrow_{\mathrm{R}}$ (I)

The transition relation $\rightarrow_{\mathrm{R}}$

- is defined on solutions
- is parametrized by a set of known biological reactions R
- allows to derive all possible output solutions given an input solution and a set of biological reactions

Definition (R-system)

Given a set of biological reactions, the associated R-*system* is the pair $(\mathrm{S}, \rightarrow_{\mathrm{R}})$, where S is the set of all solutions, and $\rightarrow_{\mathrm{R}}$ the transition relation as defined by the following rules...

The Transition Relation $\rightarrow_{\mathrm{R}}$ (II)

Given a set of biological reactions R:

$$(\textsc{mon}): \quad \frac{\mathcal{S}, \mathcal{T} \models \mathcal{R} \rightarrow (\nu \tilde{x})\mathcal{P} \quad \in \mathrm{R}}{\mathcal{S} \rightarrow_{\mathrm{R}} \mathcal{T}}$$

$$(\textsc{antimon}): \quad \frac{\mathcal{S}, \mathcal{T} \models (\nu \tilde{x})\mathcal{R} \rightarrow \mathcal{P} \quad \in \mathrm{R}}{\mathcal{S} \rightarrow_{\mathrm{R}} \mathcal{T}}$$

$$(\textsc{new}): \quad \frac{\mathcal{S} \rightarrow_{\mathrm{R}} \mathcal{T}}{(\nu x)\mathcal{S} \rightarrow_{\mathrm{R}} (\nu x)\mathcal{T}}$$

$$(\textsc{group}): \quad \frac{\mathcal{S} \rightarrow_{\mathrm{R}} \mathcal{T}}{\mathcal{S}, \mathcal{U} \rightarrow_{\mathrm{R}} \mathcal{T}, \mathcal{U}}$$

$$(\textsc{struct}): \quad \frac{\mathcal{S} \rightarrow_{\mathrm{R}} \mathcal{T} \quad \mathcal{S} \equiv \mathcal{S}' \quad \mathcal{T} \equiv \mathcal{T}'}{\mathcal{S}' \rightarrow_{\mathrm{R}} \mathcal{T}'}$$

The Transition Relation $\rightarrow_{\mathrm{R}}$ (III): Properties

Given a set of biological reactions R, suppose $\mathcal{S} \rightarrow_{\mathrm{R}} \mathcal{T}$. Then:

1. Occurrences of free names are in bijection between $\mathcal{S}$ and $\mathcal{T}$

   (Interpretation: free names are preserved by a biological reaction, i.e., all created/deleted edges correspond to bound names and other edges are untouched)

2. $\mathcal{S}$ is graph-like $\Leftrightarrow$ $\mathcal{T}$ is graph-like

   (Interpretation: biological reactions preserve the graph-likeness property of solutions)

Proof Idea. Induction on the definition of $\rightarrow_{\mathrm{R}}$. Easy to show that (NEW), (GROUP) and (STRUCT) preserve the properties. Harder for (MON) and (ANTIMON). Use definition of renaming r and of matching $\models$.

Outline

κ-Calculus: Summary (I)

- κ *syntax* is derived from graphical notation

$\Rightarrow$ Always possible to visualize a formula graphically

- *Interfaces* model a protein's state
  - Free sites can be visible or hidden
  - Bound sites are associated with a name

- *Properties:* solutions can be *(strongly) graph-like* and/or *connected*

κ-Calculus: Summary (II)

- *Growth relation* $\leq$ defined on (partial) interfaces, pre-proteins and pre-solutions

$\Rightarrow$ Used to impose conditions on how atomic reactions should look like

- *Biological reactions:*
  - Monotonic: $\mathcal{R} \rightarrow (\nu \tilde{x})\mathcal{P}$, edges are created
  - Antimonotonic: $(\nu \tilde{x})\mathcal{R} \rightarrow \mathcal{P}$, edges are deleted

$\Rightarrow$ Define the two possible atomic reactions for pre-solutions in κ

- *Matching solutions* and *transition relation* $\rightarrow_{\mathrm{R}}$ on solutions

$\Rightarrow$ Relies on the concept of biological reaction defined on pre-solutions to define possible transitions between solution or solution groups

Outline

The $m\kappa$-Calculus

- Finer-grained language, less idealized molecular biology
- "Bridge" between κ-calculus and π-calculus
- Always only binary interactions
- Implement κ in $m\kappa$
- Later, implement $m\kappa$ in π
- Describe:
  - Syntactic changes in $m\kappa$
  - New rules for transition relation $\longrightarrow$
  - From κ to $m\kappa$, the monotonic protocol
- Prove:
  - Simulation of κ by $m\kappa$ using the monotonic protocol

Informal Comparison of $m\kappa$ with κ

κ-**calculus**	$m\kappa$-**calculus**
Proteins with sites	Agents with extended sites
Sites on proteins	Extended sites: ability to store an number
Interfaces: $\mathbb{N} \to \mathcal{E} \cup \{h, v\}$	Extended interfaces: $\mathbb{N} \to (\mathcal{E} \cup \mathcal{G} \cup \{h, v\}) \times \mathbb{N}$
"Reactions" possibly several proteins	"Interactions" at most two agents at a time

- Sites are given an additional state called the *log*
- Interface are updated to include the sites' log
- Sites can now also be bound by *group names* belonging to $\mathcal{G}$

Implementing κ in $m\kappa$ (I)

- A κ reaction can be implemented in $m\kappa$
- $m\kappa$ allows only binary interactions
- Arity of κ not limited by transition relation

$\Rightarrow$ Decompose κ reaction into several $m\kappa$ interactions

- Keep properties of κ reactions
- Define a protocol for conversion of reactions
  - Protocol for monotonic reactions
  - Protocol for antimonotonic reactions
- Examine the $\longrightarrow$ rules for the monotonic protocol then illustrate with a non-trivial example

Implementing κ in $m\kappa$ (II)

Reaction is decomposed in a two-phase interaction series:

1. *Recruitment.* A signal is sent from an initiator agent (a chosen protein) down to recruit and reserve the other agents needed for the reaction (which will enter a special state in $m\kappa$); a success signal is then sent back
2. *Completion.* Now the reaction cannot fail; this information is propagated down again to let the agents project back to κ-identical proteins

$\Rightarrow$ Use *micro-scenario* to propagate signal along agents

We need extended possibilities to:

- Mark agents as "reserved" for the current reaction
- Know for each agent in which phase we currently are

$\Rightarrow$ Use an *extended interface* and *group names* to describe agents

Extended Interfaces (I): Notation

Definition (Extended interface)

An *extended interface* (θ, ρ, σ, etc.) is a map from $\mathbb{N}$ to $(\mathcal{E} \cup \mathcal{G} \cup \{h, v\}) \times \mathbb{N}$

Definition ($(m\kappa)$ Agent)

An *agent* is a pair, e.g. written $A(\theta)$,
with $A \in \mathcal{P}$ and θ: an extended interface.

Suppose a protein A with three sites, labeled 1 through 3.

- Extended interface: $\theta = \{1 \mapsto (x, 1), 2 \mapsto (r, 0), 3 \mapsto (h, 0)\}$
- "+" Notation: $A(1^{x,1} + 2^{r,0} + \bar{3}^{0})$
- Non-null notation: $A(1^{x,1} + 2^{r} + \bar{3})$

$\Rightarrow$ κ's notation is now a special case of $m\kappa$'s

Extended Interfaces (II): Projection

The log part of the extended interface is left out by the *projection map* $[\cdot]^-$ defined as follows:

- Sites bound with an edge name project to bound sites
  $[\imath^{x,n}]^- = \imath^x$
- Sites bound with a group name project to visible sites
  $[\imath^{r,n}]^- = [\imath^{v,n}]^- = [\imath^n]^- = \imath^v = \imath$
- Hidden sites project to hidden sites
  $[\imath^{h,n}]^- = [\overline{\imath}^n]^- = \imath^h = \overline{\imath}$

Projection is extended to interfaces, agents and solutions

Interactions

Recall that only two agents may interact at a time in $m\kappa$.

Definition ((Anti)monotonic interaction)

With $\mathcal{R}, \mathcal{P}$ two pre-solutions, $\mathcal{R} \to \mathcal{P}$ is a *monotonic* (resp. an *antimonotonic*) *interaction* iff:

1. $\mathcal{R}$ and $\mathcal{P}$ consist of at most two agents
2. $\mathrm{fn}(\mathcal{R}) \supseteq \mathrm{fn}(\mathcal{P})$ (i.e., no new unbound name in $\mathcal{P}$)
3. $\mathrm{bn}(\mathcal{R}) \cap \mathcal{G} = \emptyset$ ($\mathcal{G}$ = set of group names)
4. its projection $[\mathcal{R}]^- \to [\mathcal{P}]^-$ is monotonic (resp. antimonotonic) in κ

Micro-scenario (I)

Definition (Micro-scenario)

A *micro-scenario* for a monotonic reaction $\mathrm{r} : \mathcal{R} \rightarrow (\nu \tilde{x})\mathcal{P}$ is a tuple $(\mathcal{F}_\mathrm{r}, \mathcal{T}_\mathrm{r}, \mathsf{init})$, where:

- $\mathcal{F}_\mathrm{r}$: flow graph. A directed acyclic version of $[\![\mathcal{P}]\!]_g$ (the graph of the products)

  Used to recreate all bounds from the original reaction
- $\mathcal{T}_\mathrm{r}$: tree spanning the flow graph $\mathcal{F}_\mathrm{r}$ (a version of $\mathcal{F}_\mathrm{r}$ where each node has only one parent)

  Used in the recruitment phase to contact all agents once and only once
- init is the common root of both $\mathcal{F}_\mathrm{r}$ and $\mathcal{T}_\mathrm{r}$

  Used to initiate the phases

Multiple micro-scenarios always exists for each reaction in κ

Micro-scenario (II): Properties

- Define $\mathcal{F}_{\mathrm{r}}^{*}$ as the reverse flow graph, which corresponds to the reverse orientation of $\mathcal{F}_{\mathrm{r}}$

$\Rightarrow$ $\mathrm{dom}(\mathcal{F}_{\mathrm{r}}) \cup \mathrm{dom}(\mathcal{F}_{\mathrm{r}}^{*}) =$ all connected nodes from $\mathcal{P}$

- Flow graph $\mathcal{F}_{\mathrm{r}}$ can be decomposed uniquely into $\mathcal{T}_{\mathrm{r}} \cup \mathcal{T}_{\mathrm{r}}^{c}$

$\Rightarrow$ $\mathcal{T}_{\mathrm{r}}^{c}$ is empty iff $\mathcal{F}_{\mathrm{r}}$ is a tree

$\mathcal{F}_{\mathrm{r}}$ is a tree iff no proteins in the products $\mathcal{P}$ are bound cyclically

Notation:

$$(a, i) \notin \mathrm{dom}(\mathcal{F}_{\mathrm{r}}) \quad \Leftrightarrow \quad \mathcal{F}_{\mathrm{r}}(a, i) \stackrel{\mathrm{def}}{=} \perp$$

(also valid for $\mathcal{F}_{\mathrm{r}}^{*}$ and $\mathcal{T}_{\mathrm{r}}$)

Signal Ordering Relation $\succ$

Motivation: define an order over sites in order to have a well-defined propagation path for signals used in the monotonic protocol. Use it for proofs.

Definition (Signal ordering)

The relation over sites $\succ$ is defined as the least transitive relation such that:

$$\mathcal{F}_{\mathrm{r}}(a,i) = (b,j) \quad \Rightarrow \quad (a,i) \succ (b,j)$$

$$\underbrace{\mathcal{F}^*_{\mathrm{r}}(a,i) \neq \bot}_{(a,i) \text{ is an input}} \wedge \underbrace{\mathcal{F}_{\mathrm{r}}(a,j) \neq \bot}_{(a,j) \text{ is an output}} \quad \Rightarrow \quad (a,i) \succ (a,j)$$

$\succ$ is a strict partial order on sites

New "Group" Site; IN and OUT Interfaces

Extend agents' interfaces with new site $*$: $[\![A(\sigma)]\!]_m = A(* + \sigma)$

- "Mark" agents recruited for a new reaction attempt
- Notation: $A(*^{r,a} + \sigma) \stackrel{\text{def}}{=} A^{r,a}(\sigma)$
- r: group name; a: agent role in attempted reaction

Notation: IN and OUT interfaces. With $\tilde{x} = (x_1, x_2, \cdots, x_k)$:

$$\text{IN}_a^{\tilde{x},n} \stackrel{\text{def}}{=} \bigcup_{\{i \mid \mathcal{F}_r^*(a,i) \neq \perp\}} i^{x_i,n}$$

$$\text{OUT}_a^{\tilde{x},n} \stackrel{\text{def}}{=} \bigcup_{\{i \mid \mathcal{F}_r(a,i) \neq \perp\}} i^{x_i,n}$$

Outline

The Monotonic Protocol, Rules_1 (I)

Initiation and first contacts:

$$(\text{INIT}): \quad \frac{a = \text{init}(\mathcal{F}_\text{r})}{A(\sigma) \to (\nu r)(A^{r,a}(\sigma'))}$$

$$(\text{FC}_1): \quad \frac{\mathcal{T}_\text{r}(a,i) = (b,j) \qquad x \in \text{fn}(\text{r})}{A^{r,a}(\text{IN}_a^{\tilde{y},1} + i^x), B(j^x + \sigma) \to A^{r,a}(\text{IN}_a^{\tilde{y},1} + i^{x,1}), B^{r,b}(j^{x,1} + \sigma')}$$

$$(\text{FC}_2): \quad \frac{\mathcal{T}_\text{r}(a,i) = (b,j) \qquad x \notin \text{fn}(\text{r}) \qquad b \in \mathcal{R}}{A^{r,a}(\text{IN}_a^{\tilde{y},1} + i), B(j + \sigma) \to (\nu x)\big(A^{r,a}(\text{IN}_a^{\tilde{y},1} + i^{x,1}), B^{r,b}(j^{x,1} + \sigma')\big)}$$

$$(\text{FC}_3): \quad \frac{\mathcal{T}_\text{r}(a,i) = (b,j) \qquad x \notin \text{fn}(\text{r}) \qquad b \notin \mathcal{R}}{A^{r,a}(\text{IN}_a^{\tilde{y},1} + i) \to (\nu x)\big(A^{r,a}(\text{IN}_a^{\tilde{y},1} + i^{x,1}), B^{r,b}(j^{x,1} + \sigma)\big)}$$

The Monotonic Protocol, Rules_1 (II): Interpretation

(Initiation and first contacts)

- Always begin with (INIT), mark first agent (all other rules need a marked agent)
- With $(\text{FC}_{1,2,3})$, contact all agents once (using the tree $\mathcal{T}_\text{r}$) and mark them
- Change free sites when needed from h to v or from v to h (when going from σ to σ')
- (FC_1): contact agent B using an already existing edge in $\mathcal{R}$
- (FC_2): contact agent B, creating a new edge from A to B
- (FC_3): agent B does not exist yet, create it and mark it
- Always set the log of visited sites to 1

The Monotonic Protocol, Rules_2 (I)

Later contacts and responses:

$$(\mathrm{LC}_1): \quad \frac{\mathcal{T}_{\mathrm{r}}^{c}(a,i) = (b,j) \qquad x \in \mathrm{fn}(\mathrm{r})}{A^{r,a}(\mathrm{IN}_a^{\tilde{y},1} + i^{x}), B^{r,b}(j^{x}) \to A^{r,a}(\mathrm{IN}_a^{\tilde{y},1} + i^{x,1}), B^{r,b}(j^{x,1})}$$

$$(\mathrm{LC}_2): \quad \frac{\mathcal{T}_{\mathrm{r}}^{c}(a,i) = (b,j) \qquad x \notin \mathrm{fn}(\mathrm{r})}{A^{r,a}(\mathrm{IN}_a^{\tilde{y},1} + i^{x}), B^{r,b}(j) \to (\nu x)\big(A^{r,a}(\mathrm{IN}_a^{\tilde{y},1} + i^{x,1}), B^{r,b}(j^{x,1})\big)}$$

$$(\mathrm{R}): \quad \frac{\mathcal{F}_{\mathrm{r}}(a,i) = (b,j)}{A^{r,a}(i^{x,1}), B^{r,b}(j^{x,1} + \mathrm{OUT}_b^{\tilde{y},2}) \to A^{r,a}(i^{x,2}), B^{r,b}(j^{x,2} + \mathrm{OUT}_b^{\tilde{y},2})}$$

The Monotonic Protocol, Rules_2 (II): Interpretation

(Later contacts)

- All agents are now marked, we need to log 1 on sites that were not visited using $\mathcal{T}_{\mathrm{r}}$
- With $(\mathrm{LC}_{1,2})$, use the complementary tree $\mathcal{T}_{\mathrm{r}}^{c}$ to traverse the remaining sites
- (LC_1): use an already existing edge in $\mathcal{R}$
- (LC_2): create a new edge from A to B

(Responses)

- With (R), propagate the success signal (by setting the logs to 2) from the bottom of $\mathcal{F}_{\mathrm{r}}$ up to init
- Agents are only allowed to propagate the signal when they have received it from all children

The Monotonic Protocol, Rules_3 (I)

Completions:

$$(\text{SHIFT}): \quad \frac{a = \text{init}(\mathcal{F}_\text{r})}{A^{r,a}(\text{OUT}_a^{\tilde{y},2}) \to A^{r,a}(\text{OUT}_a^{\tilde{y},3})}$$

$$(\text{I-PPG}): \quad \frac{a = \text{init}(\mathcal{F}_\text{r}) \qquad \mathcal{F}_\text{r}(a,i) = (b,j)}{A^{r,a}(i^{x,3}), B^{r,b}(j^{x,2}) \to A^{r,a}(i^{x,4}), B^{r,b}(j^{x,3})}$$

$$(\text{PPG}): \quad \frac{a \neq \text{init}(\mathcal{F}_\text{r}) \qquad \mathcal{F}_\text{r}(a,i) = (b,j)}{A^{r,a}(\text{IN}_a^{\tilde{y},3} + i^{x,2}), B^{r,b}(j^{x,2}) \to A^{r,a}(\text{IN}_a^{\tilde{y},3} + i^{x,3}), B^{r,b}(j^{x,3})}$$

$$(\text{I-EXIT}): \quad \frac{a = \text{init}(\mathcal{F}_\text{r})}{A^{r,a}(\text{OUT}_a^{\tilde{x},4}) \to A(o_a^{\tilde{x}})}$$

$$(\text{EXIT}): \quad \frac{a \neq \text{init}(\mathcal{F}_\text{r})}{A^{r,a}(\text{IN}_a^{\tilde{y},3} + \text{OUT}_a^{\tilde{z},3}) \to A(\imath_a^{\tilde{y}} + o_a^{\tilde{z}})}$$

The Monotonic Protocol, Rules_3 (II): Interpretation

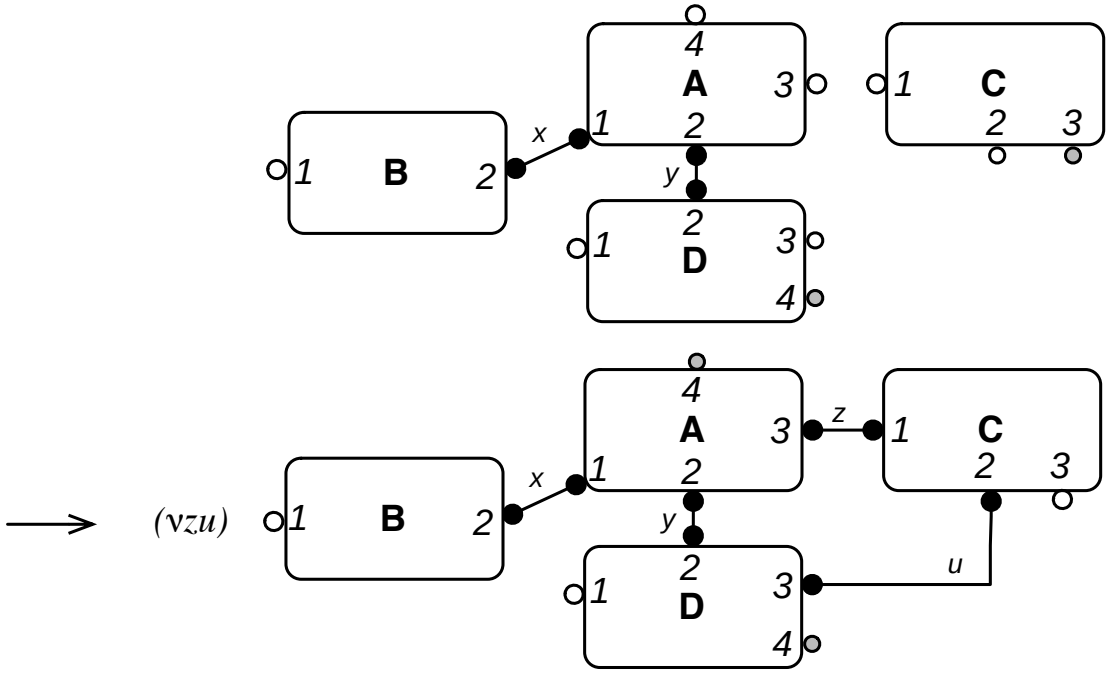
(Completions)

- When the success signal reaches init, all its output have log 2, agents are marked and reaction can't fail
- Now: propagate the completion signal down (log $= 3$) and project the agents to κ proteins
- (SHIFT) initiates the completion phase on init
- (I-PPG) and (PPG) propagate the signal (resp. for init and for other agents)
- Agents may only propagate the signal when they have received it from all parents
- (I-EXIT) and (EXIT) project the agents back to κ proteins
- Agents may only project when they have propagated the signal to all children

Outline

The Monotonic Protocol, Example (I): r_{ex}

Suppose the following monotonic κ reaction r_{ex}:

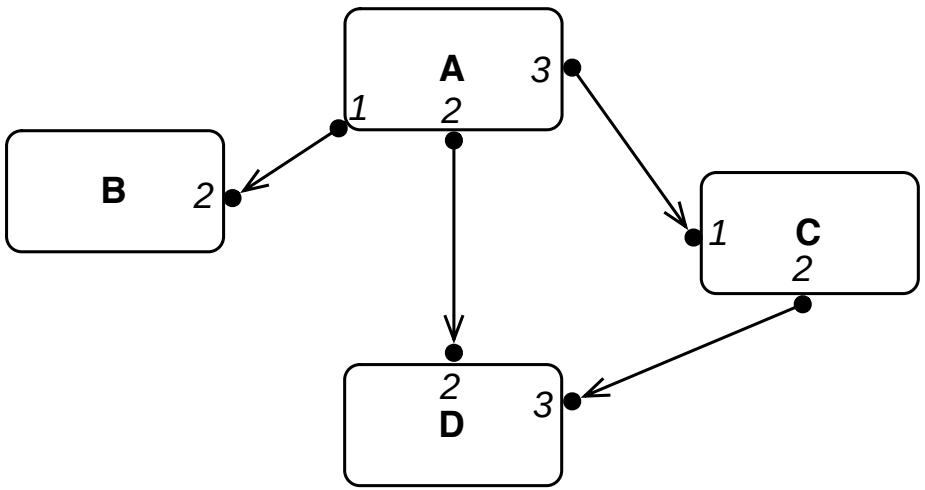


$$A(1^x + 2^y + 3 + 4), B(1 + 2^x), C(1 + 2 + \bar{3}), D(1 + 2^y + 3 + \bar{4}) \quad \longrightarrow$$
$$(\nu zu)\big(A(1^x + 2^y + 3^z + \bar{4}), B(1 + 2^x), C(1^z + 2^u + 3), D(1 + 2^y + 3^u + \bar{4})\big)$$

Example (II): Micro-scenario for r_{ex}, Defining $\mathcal{F}_{\mathrm{r}_{ex}}$

Possible micro-scenario for r_{ex}: $(\mathcal{F}_{\mathrm{r}_{ex}}, \mathcal{T}_{\mathrm{r}_{ex}}, \mathsf{init})$

- $\mathcal{F}_{\mathrm{r}_{ex}}$: acyclic orientation of the graph of the products of r_{ex}

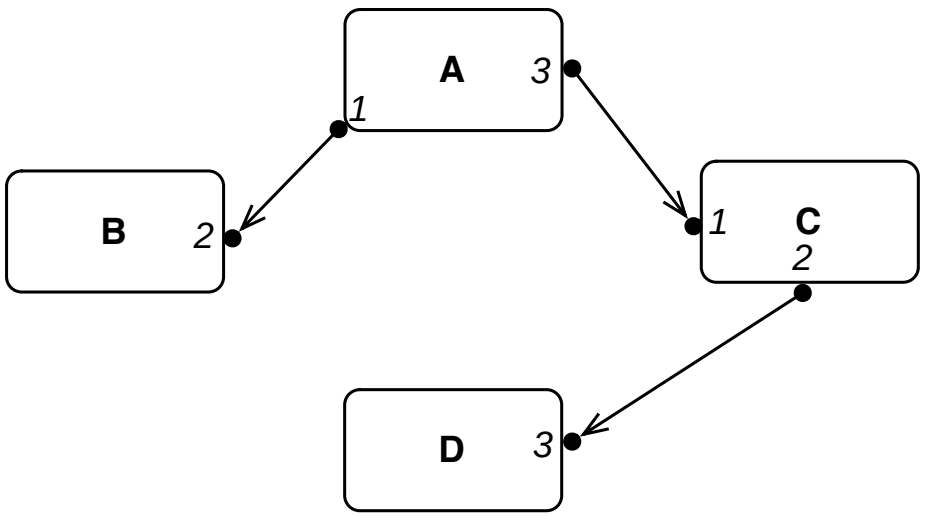


$$\mathcal{F}_{\mathrm{r}_{ex}} = \{(A,1) \mapsto (B,2), (A,2) \mapsto (D,2), (A,3) \mapsto (C,1), (C,2) \mapsto (D,3)\}$$

$$\mathcal{F}^*_{\mathrm{r}_{ex}} = \{(B,2) \mapsto (A,1), (D,2) \mapsto (A,2), (C,1) \mapsto (A,3), (D,3) \mapsto (C,2)\}$$

Example (III): Micro-scenario for r_{ex}, Def. $\mathcal{T}_{\mathrm{r}_{ex}}$ and init

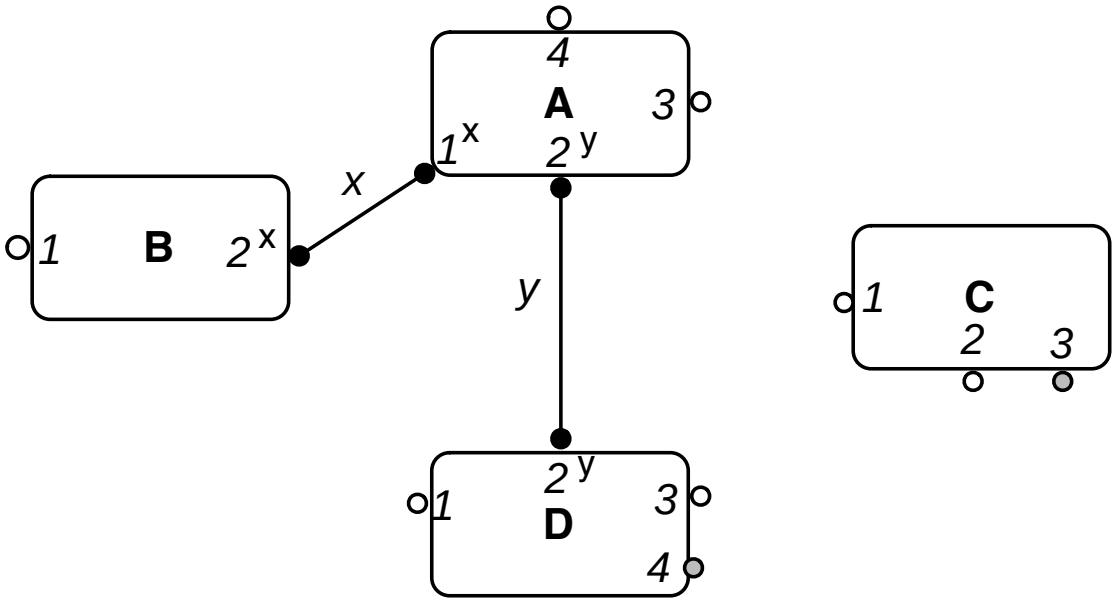
- $\mathcal{T}_{\mathrm{r}_{ex}}$: tree spanning $\mathcal{F}_{\mathrm{r}_{ex}}$



$$\mathcal{T}_{\mathrm{r}_{ex}} = \{(A,1) \mapsto (B,2), (A,3) \mapsto (C,1), (C,2) \mapsto (D,3)\}$$
$$\mathcal{T}^{c}_{\mathrm{r}_{ex}} = \mathcal{F}_{\mathrm{r}_{ex}} \setminus \mathcal{T}_{\mathrm{r}_{ex}} = \{(A,2) \mapsto (D,2)\}$$

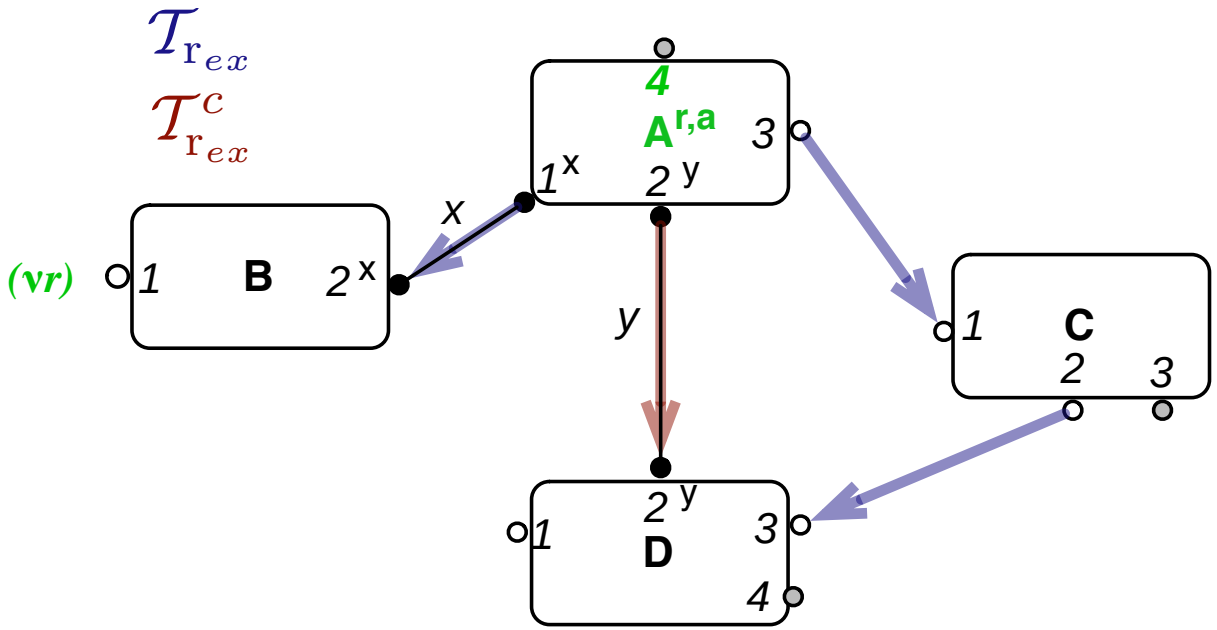
- init $=$ common root of $\mathcal{F}_{\mathrm{r}_{ex}}$ and $\mathcal{T}_{\mathrm{r}_{ex}} \stackrel{\text{def}}{=} A$

Example (IV): Transitions$_1$



Start situation: this is a κ solution

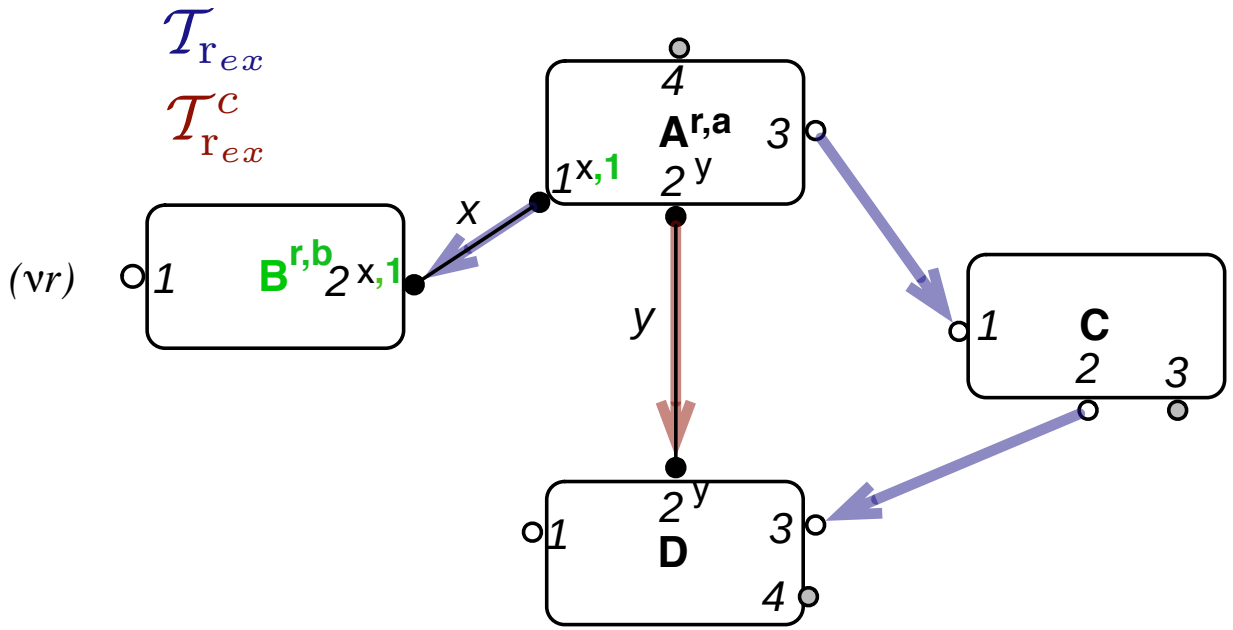
Example (V): Transitions$_2$



(INIT): $A(\underbrace{1^x + 2^y + 3 + 4}_{\sigma}) \rightarrow (\nu r)(A^{r,a}(\underbrace{1^x + 2^y + 3 + \bar{4}}_{\sigma'}))$

$\sigma \neq \sigma'$, i.e., there were changes in free sites:
$(a, 4)$ has switched from v to h

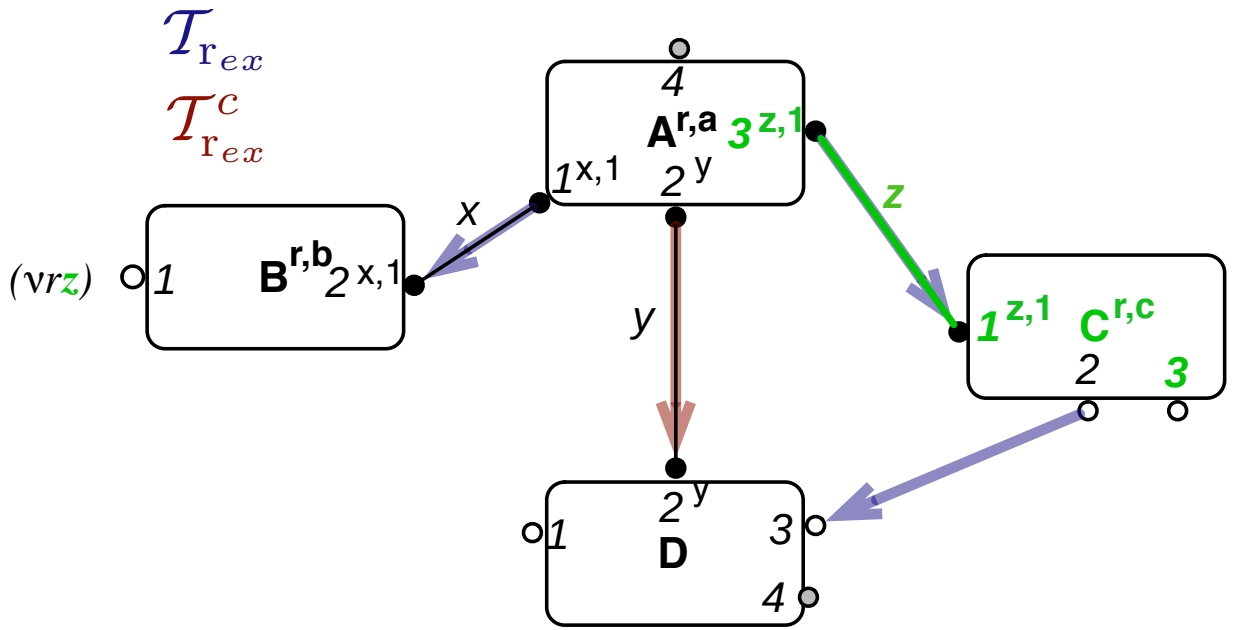
Example (VI): Transitions$_3$



$$(\mathrm{FC}_1):\ A^{r,a}(\overbrace{1^{x}}^{i^x} + 2^{y} + 3 + 4),\ B(\overbrace{1}^{\sigma} + \overbrace{2^{x}}^{j^x})$$
$$\rightarrow (A^{r,a}(\underbrace{1^{x,1}}_{i^{x,1}} + 2^{y} + 3 + \bar{4}),\ B^{r,b}(\underbrace{1}_{\sigma'} + \underbrace{2^{x,1}}_{j^{x,1}}))$$

$\mathrm{IN}_a^{\tilde{y},1} = \emptyset$; $\sigma = \sigma'$, i.e., no change in free sites (h to v or v to h)

Example (VII): Transitions$_4$

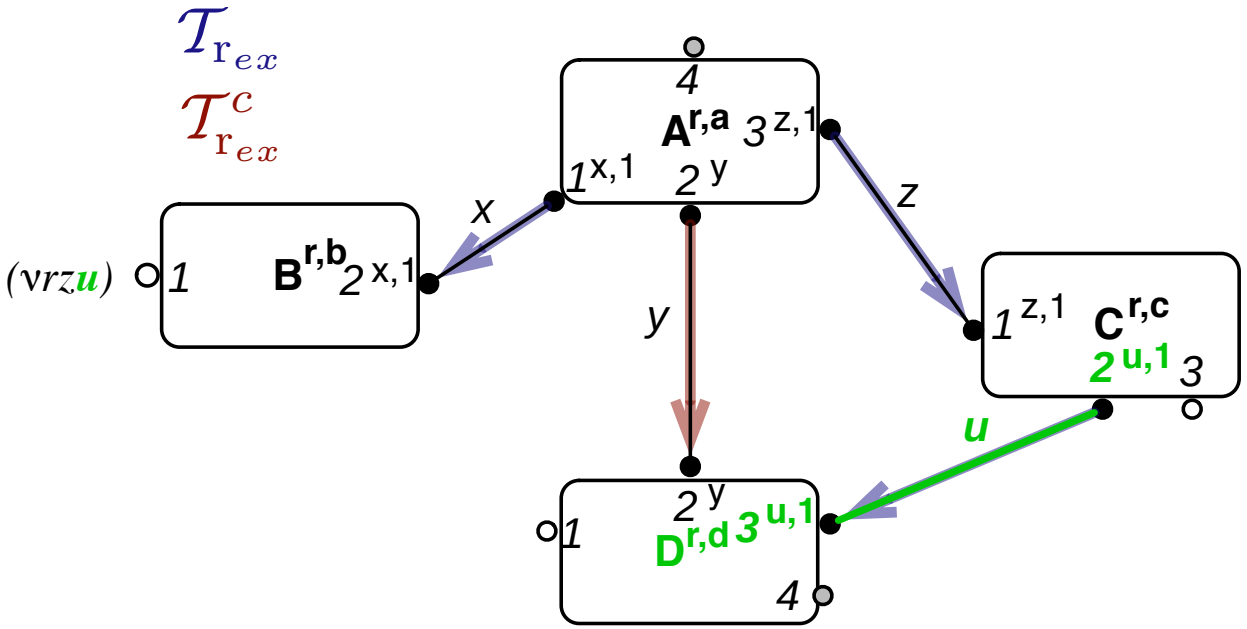


(FC_2): $A^{r,a}(1^{x,1} + 2^{y} + \overbrace{3}^{i} + \bar{4}), C(\overbrace{1}^{j} + \overbrace{2+\bar{3}}^{\sigma})$

$\longrightarrow (\nu z)(A^{r,a}(1^{x,1} + 2^{y} + \underbrace{3^{z,1}}_{i^{z,1}} + \bar{4}), C^{r,c}(\underbrace{1^{z,1}}_{j^{z,1}} + \underbrace{2+3}_{\sigma'}))$

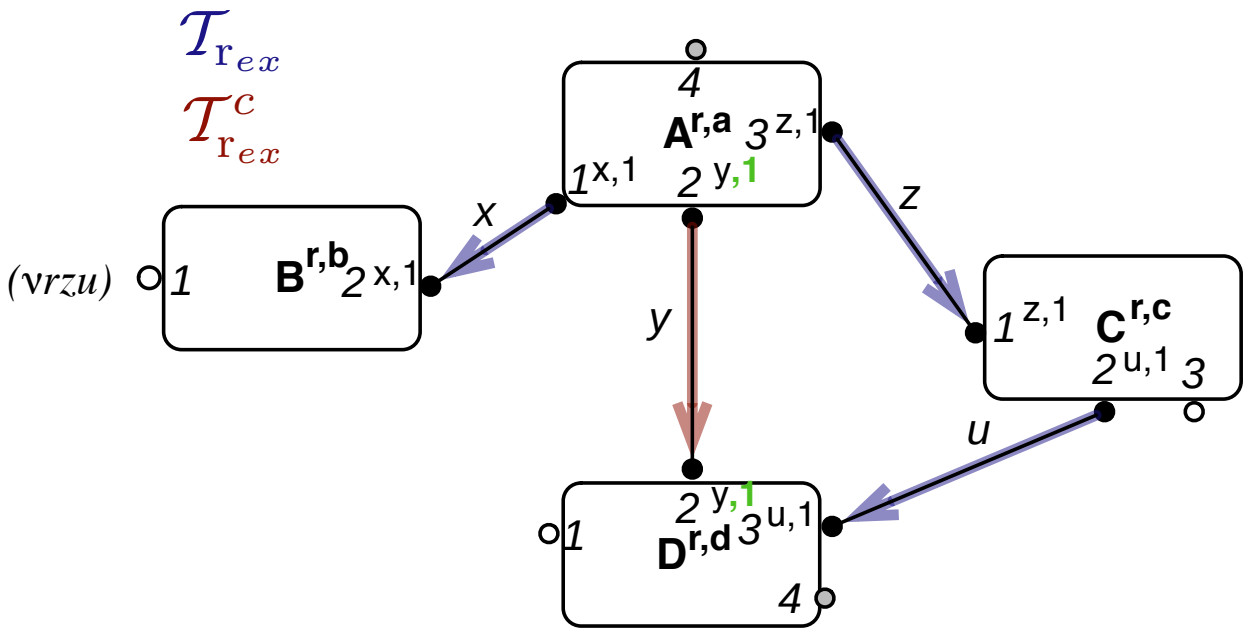
$\sigma \neq \sigma'$: $(c, 4)$ has switched from h to v

Example (VIII): Transitions$_5$



$$(\mathrm{FC}_2):\ C^{r,c}(\overbrace{1^{z,1}}^{\mathrm{IN}_c^{\tilde{y},1}} + \overbrace{2}^{i} + 3),\ D(\overbrace{1 + 2^{y} + \bar{4}}^{\sigma} + 3)$$
$$\rightarrow (\nu u)(C^{r,c}(\underbrace{1^{z,1}}_{\mathrm{IN}_c^{\tilde{y},1}} + \underbrace{2^{u,1}}_{i^{u,1}} + 3),\ D^{r,d}(\underbrace{1 + 2^{y} + \bar{4}}_{\sigma'} + 3^{u,1}))$$
$$\sigma = \sigma'$$

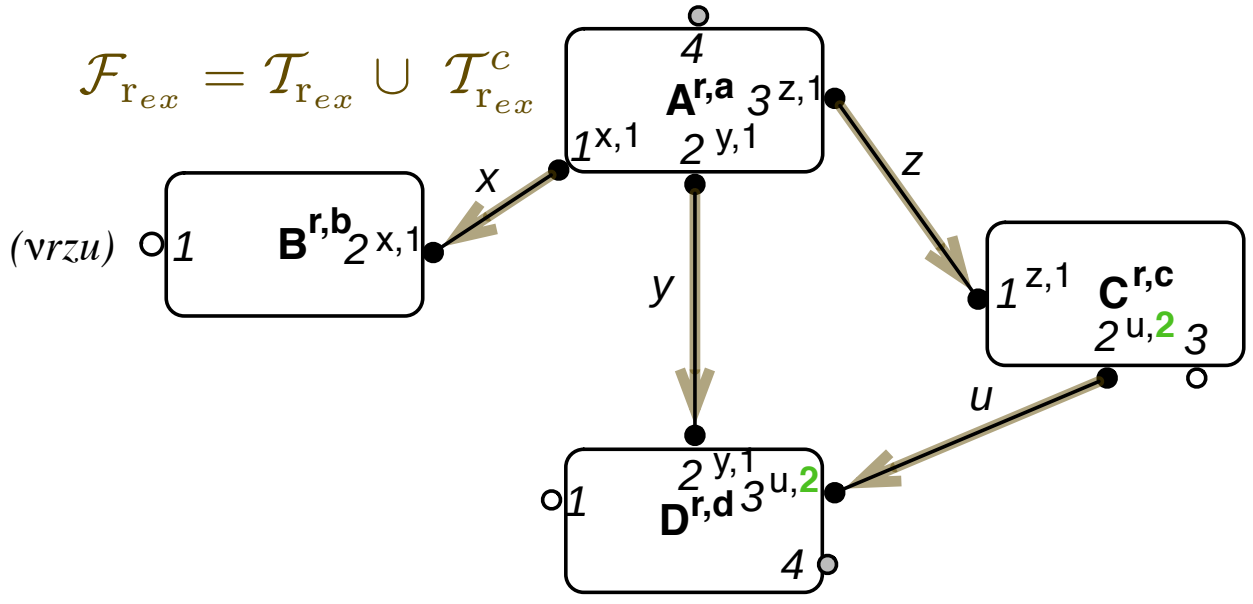
Example (IX): Transitions$_6$



$$(\mathrm{LC}_1):\ A^{r,a}(1^{x,1} + \overbrace{2^{y}}^{i^y} + 3^{z,1} + \bar{4}), D^{r,d}(1 + \overbrace{2^{y}}^{j^y} + 3^{u,1} + \bar{4})$$
$$\longrightarrow A^{r,a}(1^{x,1} + \underbrace{2^{y,1}}_{i^{y,1}} + 3^{z,1} + \bar{4}), D^{r,d}(1 + \underbrace{2^{y,1}}_{j^{y,1}} + 3^{u,1} + \bar{4})$$

$$\mathrm{IN}_a^{\tilde{y},1} = \emptyset$$

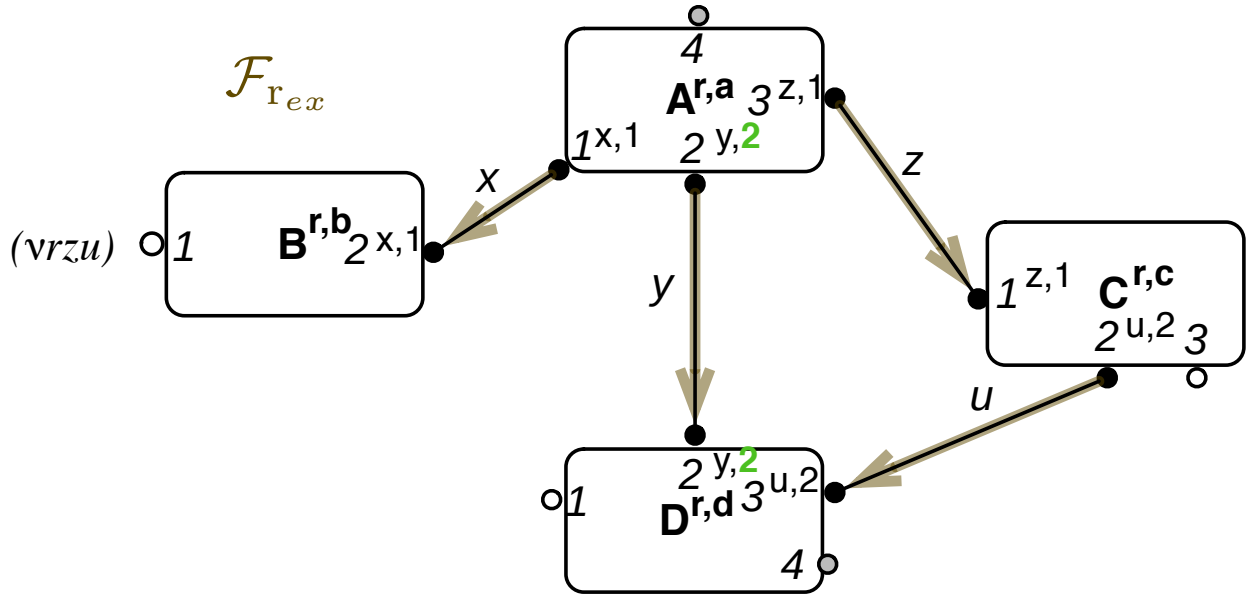
Example (X): Transitions$_7$



$$(\mathrm{R}):\ C^{r,c}(1^{z,1} + \overbrace{2^{u,1}}^{i^{u,1}} + 3),\ D(1 + 2^{y,1} + \overbrace{3^{u,1}}^{j^{u,1}} + \bar{4})$$
$$\rightarrow C^{r,c}(1^{z,1} + \underbrace{2^{u,2}}_{i^{u,2}} + 3),\ D(1 + 2^{y,1} + \underbrace{3^{u,2}}_{j^{u,2}} + \bar{4})$$

$$\mathrm{OUT}_d^{\tilde{y},2} = \emptyset$$

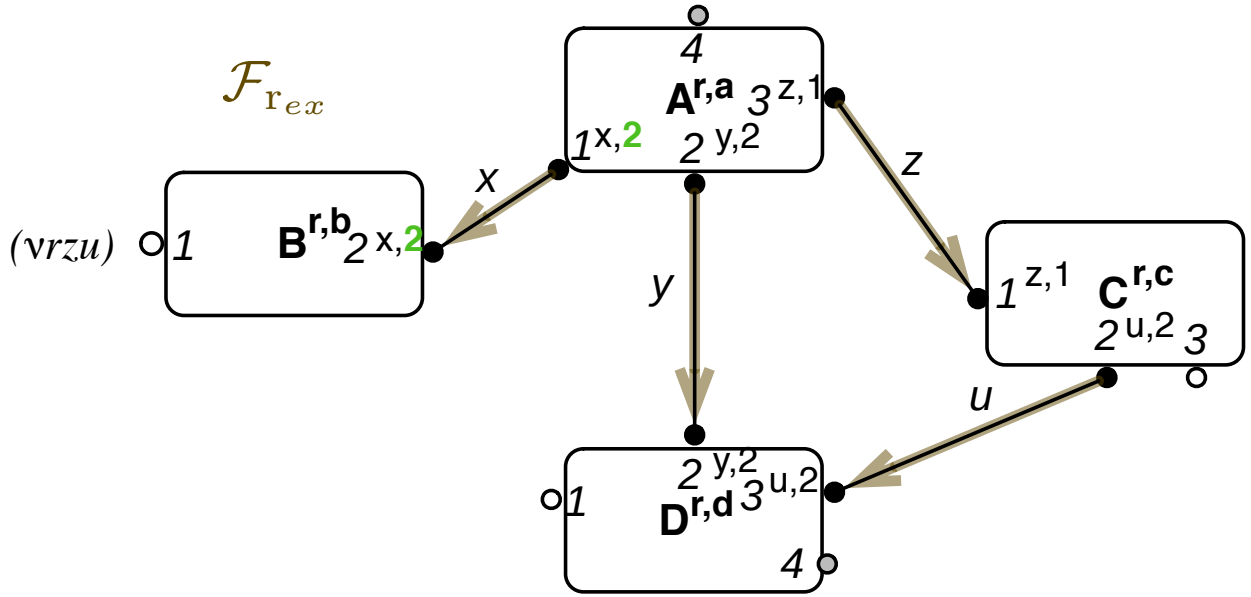
Example (XI): Transitions$_8$



$$(\mathrm{R}):\ A^{r,a}(1^{x,1} + \overbrace{2^{y,1}}^{i^{y,1}} + 3^{z,1} + \bar{4}), D(1 + \overbrace{2^{y,1}}^{j^{y,1}} + 3^{u,2} + \bar{4})$$
$$\rightarrow A^{r,a}(1^{x,1} + \underbrace{2^{y,2}}_{i^{y,2}} + 3^{z,1} + \bar{4}), D(1 + \underbrace{2^{y,2}}_{j^{y,2}} + 3^{u,2} + \bar{4})$$

$$\mathrm{OUT}_d^{\tilde{y},2} = \emptyset$$

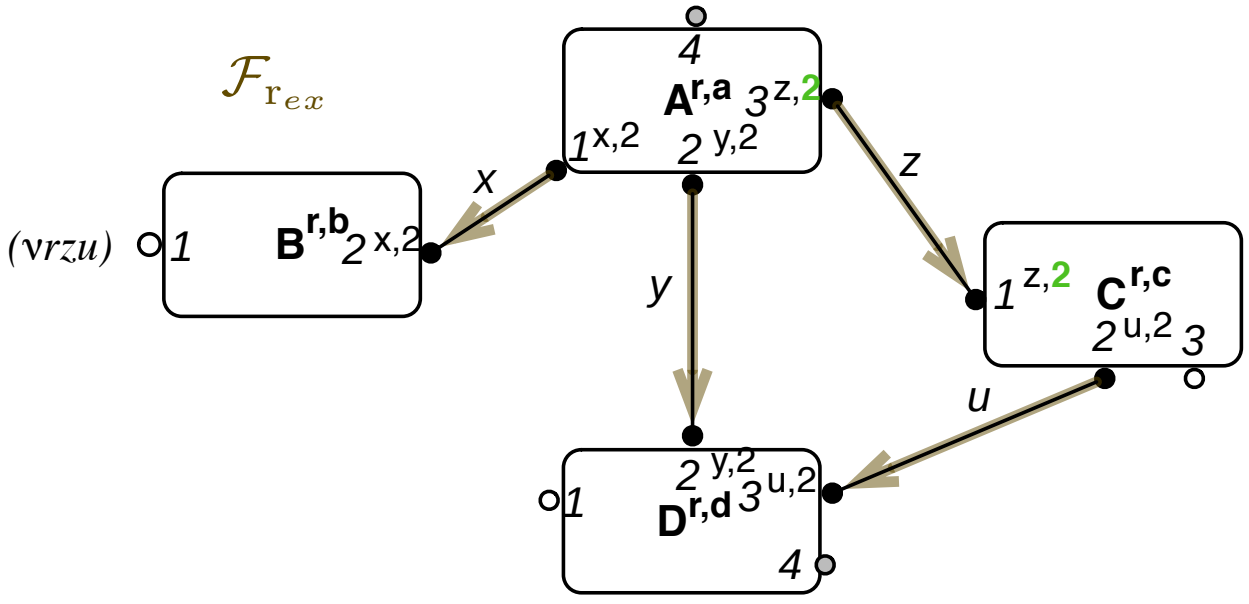
Example (XII): Transitions$_9$



$$\text{(R):}\quad A^{r,a}(\overbrace{1^{x,1}}^{i^{x,1}} + 2^{y,2} + 3^{z,1} + \bar{4}),\ B(1 + \overbrace{2^{x,1}}^{j^{x,1}}) \longrightarrow A^{r,a}(\underbrace{1^{x,2}}_{i^{x,2}} + 2^{y,2} + 3^{z,1} + \bar{4}),\ B(1 + \underbrace{2^{x,2}}_{j^{x,2}})$$

$$\mathrm{OUT}_b^{\tilde{y},2} = \emptyset$$

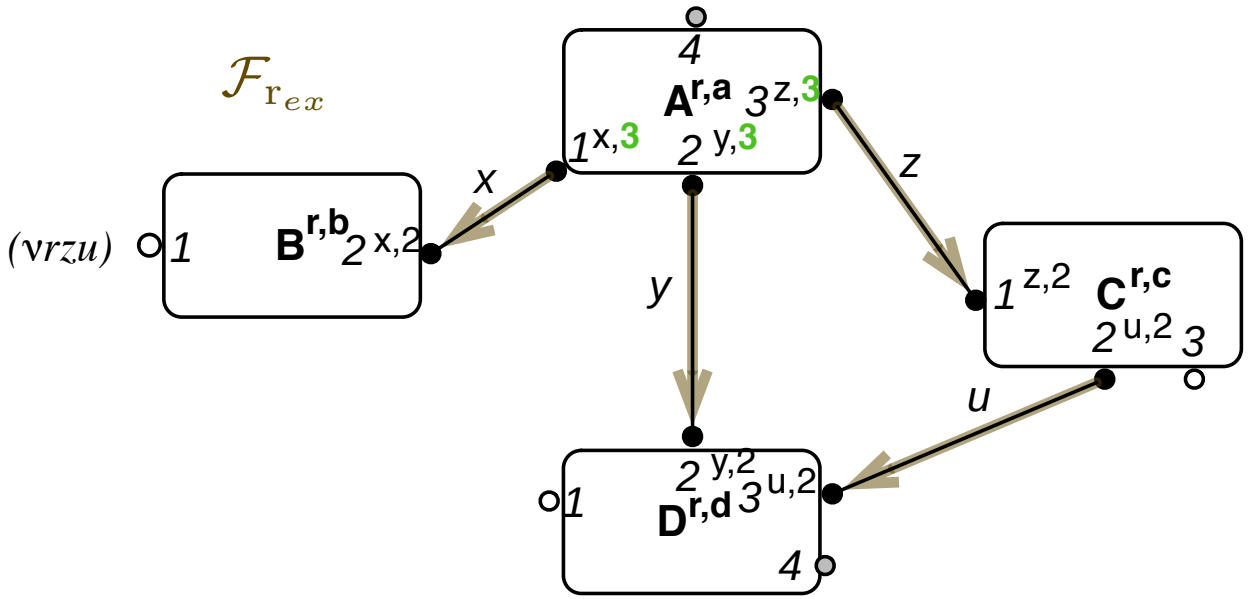
Example (XIII): Transitions$_{10}$



$$\text{(R):}\quad A^{r,a}(1^{x,2} + 2^{y,2} + \overbrace{3^{z,1}}^{i^{z,1}} + \bar{4}),\ C^{r,c}(\overbrace{1^{z,1}}^{j^{z,1}} + \overbrace{2^{u,2}}^{\mathrm{OUT}_c^{\tilde{y},2}} + 3)$$
$$\longrightarrow A^{r,a}(1^{x,2} + 2^{y,2} + \underbrace{3^{z,2}}_{i^{z,2}} + \bar{4}),\ C^{r,c}(\underbrace{1^{z,2}}_{j^{z,2}} + \underbrace{2^{u,2}}_{\mathrm{OUT}_c^{\tilde{y},2}} + 3)$$

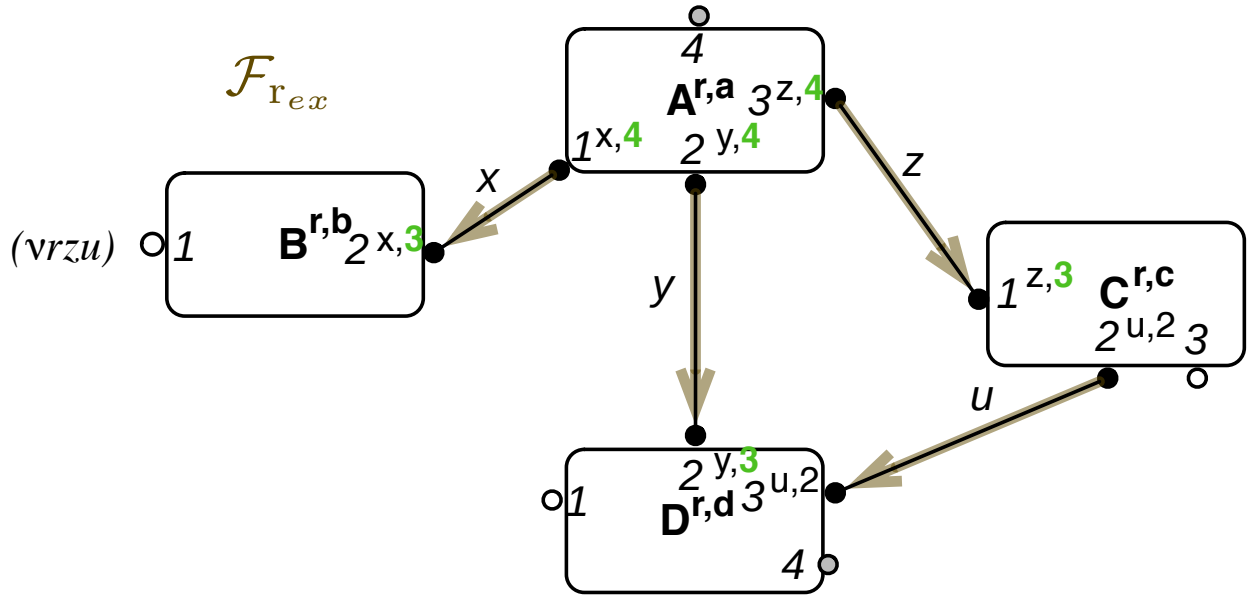
(For the rest of the example, we will use only partial interfaces)

Example (XIV): Transitions$_{11}$



$$(\text{SHIFT}):\ A^{r,a}(\underbrace{1^{x,2}+2^{y,2}+3^{z,2}}_{\text{OUT}_a^{\tilde{y},2}}) \to A^{r,a}(\underbrace{1^{x,3}+2^{y,3}+3^{z,3}}_{\text{OUT}_a^{\tilde{y},3}})$$

Example (XV): Transitions$_{12}$

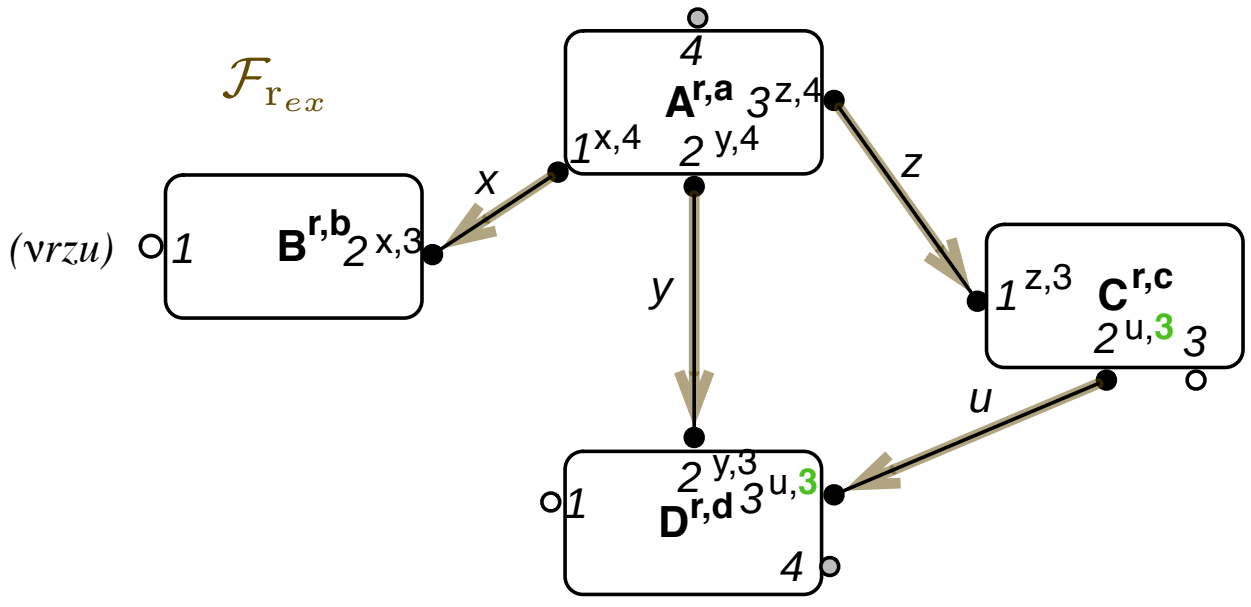


(I-PPG): $A^{r,a}(1^{x,3}), B(2^{x,2}) \longrightarrow A^{r,a}(1^{x,4}), B(2^{x,3})$

(I-PPG): $A^{r,a}(2^{y,3}), D(2^{y,2}) \longrightarrow A^{r,a}(2^{y,4}), D(2^{y,3})$ each with $a = \text{init}$

(I-PPG): $A^{r,a}(3^{z,3}), C(1^{z,2}) \longrightarrow A^{r,a}(3^{z,4}), C(1^{z,3})$

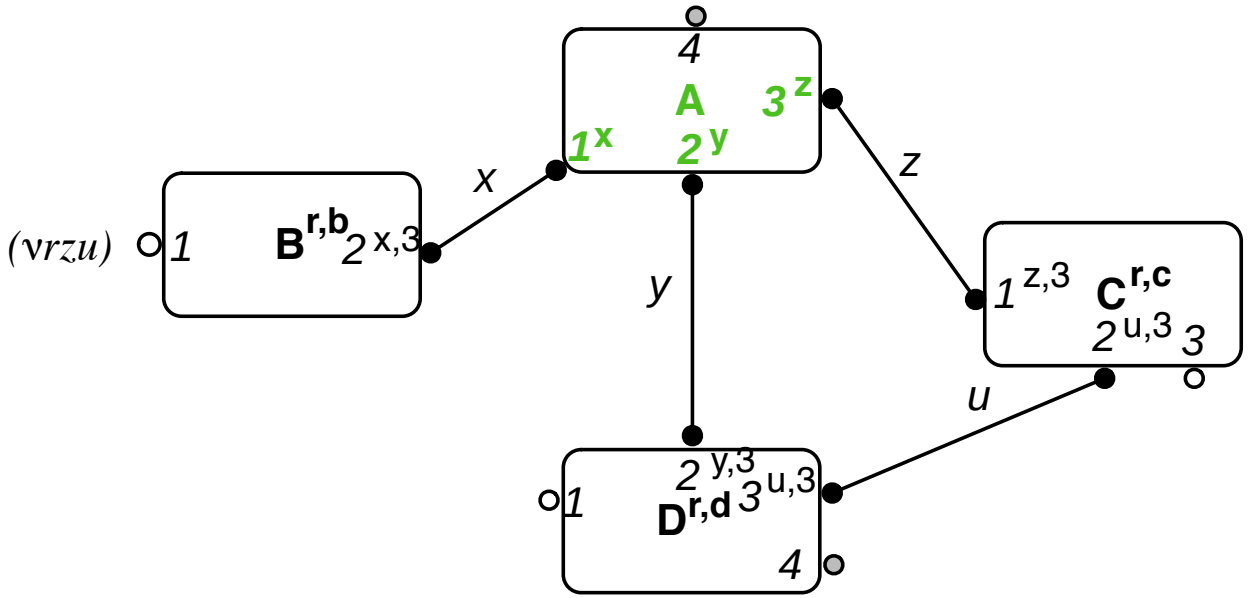
Example (XVI): Transitions$_{13}$



(PPG): $C^{r,c}(\underbrace{1^{z,3}}_{\mathrm{IN}_c^{\tilde{y},3}} + \underbrace{2^{u,2}}_{i^{u,2}}), D^{r,d}(3^{u,2}) \rightarrow C^{r,c}(\underbrace{1^{z,3}}_{\mathrm{IN}_c^{\tilde{y},3}} + \underbrace{2^{u,3}}_{i^{u,3}}), D^{r,d}(3^{u,3})$

$c \neq \mathsf{init}$

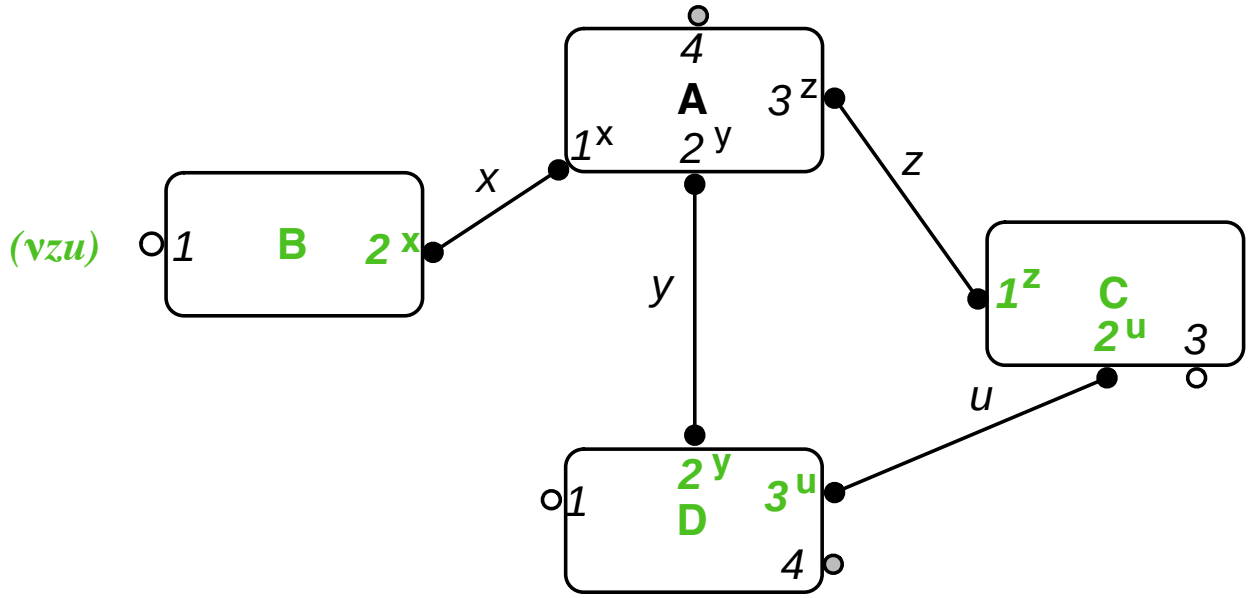
Example (XVII): Transitions$_{14}$



$$\text{(I-EXIT)}:\quad A^{r,a}(\underbrace{1^{x,4} + 2^{y,4} + 3^{z,4}}_{\text{OUT}_a^{\tilde{y},4}}) \to A(1^x + 2^y + 3^z)$$

$$a = \text{init}$$

Example (XVIII): Transitions$_{15}$



(EXIT): $B^{r,b}(2^{x,3}) \longrightarrow B(2^{x})$

(EXIT): $C^{r,c}(1^{z,3} + 2^{u,3}) \longrightarrow C(1^{z} + 2^{u})$

(EXIT): $D^{r,d}(2^{y,3} + 3^{u,3}) \longrightarrow D(2^{y} + 3^{u})$

$b, c, d \neq$ init; restriction on r is dropped with structural congruence

Outline

$m\kappa$-Calculus: Summary

- *Extended sites, extended interfaces* are used in $m\kappa$

$\Rightarrow$ Add additional state information to agents.

$\Rightarrow$ κ solutions are a special case of $m\kappa$

- *Micro-scenario* $(\mathcal{F}_{\mathrm{r}}, \mathcal{T}_{\mathrm{r}}, \mathrm{init})$ are used to implement a κ reaction in $m\kappa$. Two series of interaction:
  1. Recruitment: find & mark needed agents
  2. Completion: with success signal, project back to κ

$\Rightarrow$ The monotonic protocol

From $m\kappa$-Calculus to π-Calculus

- With its binary interaction, $m\kappa$ can be implemented in π
- Basic ideas:
  - Each agent becomes a process
  - Communication is asymmetric in π: decide which processes are senders and which ones are receivers
  - Processes are parametrized by the agents' interfaces
  - Sender sends its interface, receiver checks compatibility:
    - OK $\Rightarrow$ Makes necessary changes and sends back updated interface on success channel
    - not OK $\Rightarrow$ Tells sender to abort interaction on failure channel
  - Conditions are expressed with π's matches: $[u = u']P; Q$

- See original paper for more info:
  Danos & Laneve, *Formal Molecular Biology*
  http://www.cs.unibo.it/~laneve/papers/fmb.pdf

Formal Molecular Biology: Summary

- Biological modeling problem
  - Protein interactions: concurrent, asynchronous
  - Define new process algebra to model protein interactions and biological reactions
- κ-Calculus
  - Idealized protein calculus
  - Easily visualizable
  - Allows two kinds of atomic reactions: monotonic and antimonotonic
- $m\kappa$-Calculus
  - Finer-grained language, extended syntax
  - Allows only binary interactions
  - κ reactions are implementable in $m\kappa$
  - $m\kappa$-calculus can be implemented in π-calculus