Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 284 (2012) 55–74

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

# Formal Cellular Machinery

Troels C. Damgaard

*Edlund A/S, Denmark*

Espen Højsgaard

*IT University Copenhagen, PLS group, Denmark*

Jean Krivine[1]

*Univ Paris Diderot, Sorbonne Paris Cité,*
*Laboratoire PPS, UMR 7126, F-75205 Paris, France*

**Abstract**

Various calculi have been proposed to model different levels of abstraction of cell signaling and molecular interactions. In this paper we propose a framework inspired by some of these calculi that structures interactions and agents from the most basic elements of the cell (protein interaction sites) to higher order ones (compartments and molecular species).

*Keywords:* systems biology, compartment, rule-based modeling, kappa, bigraphs, projectivity, membrane, term rewriting

## 1 Introduction

It has been about 10 years now since part of the theoretical computer science community got interested in applying formal methods to systems biology. Since then it seems that the quest for a calculus having proteins, compartments or channels as first class citizens has not reached an end. Among the large variety of languages that have been proposed to tackle various aspects of systems biology (see Refs. [22,5,21,10,12,3,1,18,20,17,19,15,2,4,14] for a non exhaustive list), several ideas seem of particular importance to us: (i) the cellular medium can be described as a graph where nodes represent molecules and edges represent physical contacts between these molecules [10,12,1,14], (ii) languages with a natural notion of location

[1] corresponding author: jean.krivine@pps.jussieu.fr


doi:10.1016/j.entcs.2012.05.015

of reaction can be used to represent cellular compartments [21,19,20,15,2], (iii) interactions between compartments and proteins or vesicle transformations can be described using local patches of membranes, without committing to any particular global curvature [11,4], and (iv) although laws governing interactions of molecular components are numerous, they can be engendered by a small set of generators [3].

The present work proposes to integrate points (i) to (iv) in a single formalism. More specifically we define a language for proteins and cells in an incremental way, making explicit the trade-off between expressiveness and complexity. We decompose the construction of the language in four steps:

– $\mathcal{C}_0$**:** an "untyped" calculus aimed at modeling protein-protein interactions. The dynamics of these interactions is presented as a small set of *generator* rules, which modelers can refine and compose but not change.

– $\mathcal{C}_1$**:** an intermediate version of the term language that allows modelers to type reactions introduced at the previous stage.

– $\mathcal{C}_2$**:** the main expressiveness increment of our language. It introduces compartments and the notion of *projectivity* of membrane reactions, *i.e.* the possibility to mention patches of membrane, without having to deal with their global curvature. We propose a matching algorithm, that is proven both sound and complete. At this stage, generators allow modelers to create and destroy compartments in a projective fashion.

– $\mathcal{C}_3$**:** the final step of the construction deals with the diffusion problem. In particular we incorporate means to talk about connected components of reactants, which is a key feature for a new set of generators modeling diffusion of molecular species and intra-molecular complex formation. To the best of our knowledge $\mathcal{C}_3$ is the first calculus of its kind that allows one to model molecular agents both at a micro level (where interactions are purely local) and a macro level (where interactions involve connected components of agents).

The language we build is inspired by and closely related to the $\kappa$-calculus of Danos and Laneve [9,10] and Milner's bigraphical reactive systems [16], however these connections will be left informal throughout the paper. The reader might refer to Appendix A and to Ref. [6] for some preliminary work on the subject.

## 2 $\mathcal{C}_0$: forming molecules

Proteins are long polymers built over an alphabet of 20 amino acids. Each protein's interaction capabilities are mediated by its 3D folding in space which in turn depends on its amino acid composition. Protein interactions are either *structural* when they form non-covalent bonds to other molecular agents (DNA, RNA, other proteins) or *enzymatic* when they can catalyze the chemical modification of the substrate to which they are bound. In the first case one usually talks about complex formation, in the latter one talks about post-transcriptional modification. It has been observed that the amino acid sequence of most proteins appearing in living organisms can be regrouped into *domains* which are strings of amino acids that have a specific

fold in space that is rather context free. Biologists tend to associate "functions" to domains, for instance zinc finger domains are often linked to the specific DNA binding capability of their host protein.

The first step of our construction, termed $\mathcal{C}_0$, is aimed at representing *domains* as a collection of interaction sites, *proteins* as a collection of domains and *interactions* as protein assembly and complex formation.

### 2.1 Terms

Consider an infinite set of *site names* $\mathcal{S} = \{x, y, z, \dots\}$ and a disjoint infinite set of *backbone names* $\mathcal{B} = \{a, b, c, \dots\}$. Let $\mathsf{D}$ be a terminal symbol, distinct from all others, that we use to denote domains. Terms $T$ of $\mathcal{C}_0$ are built on the following grammar:

$$D, D' ::= \mathsf{D}^a(x_1, \dots, x_k) \qquad \text{for } a \in \mathcal{B},\ x_i \in \mathcal{S}$$

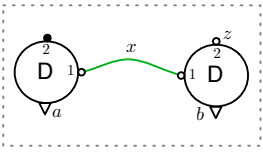
$$T, S \quad ::= D \mid 0 \mid (T, S) \mid T\backslash v \qquad \text{for } v \in \mathcal{S} \cup \mathcal{B}$$

Intuitively a $k$-ary domain $\mathsf{D}^a(x_1, \dots, x_k)$ is the placeholder of $k$ (interaction) sites and one backbone. Each site $i$ is equipped with a name $x_i \in \mathcal{S}$ and each domain with a backbone name $a \in \mathcal{B}$. Backbone name sharing denotes domains that belong to the same protein, site name sharing denotes complex formation. We inductively define free occurrences of names as:

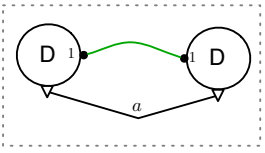
$$\begin{aligned} fn\left(\mathsf{D}^a(x_1, \dots, x_k)\right) &= \{a, x_1, \dots, x_k\} \\ fn(0) &= \emptyset \\ fn(T, S) &= fn(T) \cup fn(S) \\ fn(T\backslash v) &= fn(T) - \{v\} \end{aligned}$$

Symmetrically, one can define the bound occurrences of names, which we shall denote by $bn(T)$. Terms are equipped with a natural notion of structural congruence defined in Fig. 1. The structural congruence relation rules include a natural $\alpha$-equivalence on bound names. In the following we assume that names that are not under the same binder are kept distinct.

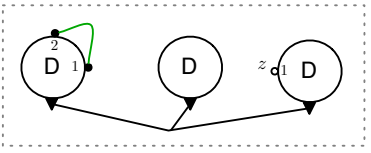
### 2.2 Graphical notation




$T = \left(\mathsf{D}^a(x, y), \mathsf{D}^b(x, z)\right) \backslash y$




$S = (\mathsf{D}^a(x), \mathsf{D}^a(x)) \backslash x$




$U = (\mathsf{D}^a(x, x), \mathsf{D}^a(z), \mathsf{D}^a()) \backslash a \backslash x$

Intuitively, the term to port graph correspondence is the following: domains are nodes, sites and backbones are ports and name sharing denotes *(hyper) edges*. Bound names denote *closed ports* and we use the term *closed edges* to denote a

$$
\begin{array}{lll}
(S,T) & \equiv (T,S) & \\
((T,S),T') & \equiv (T,(S,T')) & \\
(T,0) & \equiv T & \\
T\backslash u & \equiv T & u \notin fn(T) \\
(T\backslash u)\backslash v & \equiv (T\backslash v)\backslash u & \\
T\backslash u & \equiv (T\,\{v/u\})\backslash v & v \notin fn(T) \\
(T\backslash u,S) & \equiv (T,S)\backslash u & u \notin fn(S)
\end{array}
$$

Fig. 1. Structural congruence for $\mathcal{C}_0$.

bound name that is shared. Similarly, free names denote *open ports* and form *open edges* when they are shared. Open ports or edges can be merged or closed in the context (see later). With these conventions, one may view any term (up to structural congruence) as the isomorphism class of a port graph (with hyper-edges), in the style of bigraphs [16], where nodes (domains) are equipped with connection ports (sites and backbones). As an example we give above the port graph representation of terms $T, S$ and $U$. The reader familiar with bigraphs will notice that we drift slightly away from Milner's notation: site ports are represented by small circles that are filled when they are closed. Backbone ports are represented as small triangles that are also filled when they are closed. We use curved lines for site edges and straight lines for backbone edges. We label open edges or open ports with the corresponding free name (closed edges and ports are not labelled). Note that we will omit site numbers whenever they are not necessary.

Connections between sites correspond to physical contacts between protein parts. This connection being exclusive we want to restrict to terms where restrictions bind at most two occurrences of site names. In the following of this paper we will assume that for any term $T$, free site names occur exactly once in $T$ and bound site names have at most two occurrences. Note that we do not impose such restrictions on backbone name sharing.

### *2.3 Pattern matching and dynamics*

A *match* for $T$ in $S$ is defined as a context $\mathbb{C}[\bullet]$ with exactly one hole such that $\mathbb{C}[T] \equiv S$. Such contexts are defined inductively as:

$$\mathbb{C}[\bullet] ::= \bullet \mid \mathbb{C}[\bullet]\backslash u \mid \mathbb{C}[\bullet], T \quad u \in \mathcal{B} \cup \mathcal{S}$$

*A rule* is a pair of terms $\langle T, S\rangle$ such that $fn(S) \subseteq fn(T)$. Given a set $\mathcal{R}$ of such pairs, one may rewrite terms by letting these rules be applied in a context free

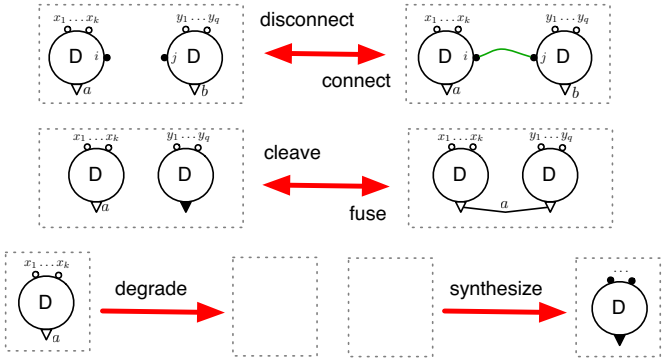



Fig. 2. The set $\mathcal{G}_0$ of generators for $\mathcal{C}_0$.

manner, *i.e.* :

$$\frac{r = \langle T, S\rangle \in \mathcal{R} \quad T' \equiv \mathbb{C}[T\sigma] \quad S' \equiv \mathbb{C}[S\sigma]}{T' \to_r S'}$$

for some name substitution $\sigma$.

### *2.4 Generators*

It is clear that not all rules make sense from a biological point of view: the fact that backbone names denote the core of a protein and that site names denote connection between protein domains is purely conventional and this convention could be easily broken. A way to proceed is to define some sorting discipline that allows one to screen off undesired terms from admissible ones [2], invalid rule applications being discarded "on the fly". Instead of doing this, we adopt a strategy of pre-conceiving what "laws" a modeler is able to invoke when defining her own rule set. This is achieved by defining a set $\mathcal{G}_0$ of basic rule *generators* that a modeler can only refine to her needs, cf. Fig. 2. These generators allow one to perform standard atomic actions of graph rewriting. It is noteworthy that these generators, including degrade, are side effect free. We shall carry this set of generators throughout the rest of this paper, incorporating new generators as the language grows.

Say a rule $r = \langle T, S\rangle$ is *generated* if and only if it can be obtained by:

- **refinement**: there exists $\langle T', S'\rangle \in \mathcal{G}_0$ such that $T \equiv \mathbb{C}[T'\sigma]$ and $S \equiv \mathbb{C}[S'\sigma]$ for some context $\mathbb{C}[\bullet]$ and substitution $\sigma$.
- **composition**: one can generate two rules $\langle T, T'\rangle$ and $\langle T', S\rangle$.

### *2.5 Discussion*

We have introduced so far a simple calculus that rewrites proteins structured as connected domains. Proteins can be connected to each other (as in complex formation), new domains can be fused to proteins (as in protein synthesis) or severed (as trans-membrane proteins can be cleaved to emit signals into the inter cellular medium). This calculus is fairly abstract in the sense that two proteins may only differ in the number of domains they have and in the number of sites these domains

possess. It is clear that we lack means of *naming* molecular components such as domain names (SH2, Tyrosine, PWWP etc.) or protein names (SOS, EGF, IGF, p53, etc.). Before performing a bigger increment in expressiveness, when we introduce compartments in Section 4, we would like to briefly introduce a way to deal with names as a particular type of context in which unamed proteins can be embedded. The intent is to provide a way to define molecular reactions as refinements of the generators we have just presented, in keeping with the biological intuition that information about molecular objects is always partial and that more context could reveal more about the nature of a molecule. In particular, we have the ontology problem in mind that several names can denote the same protein or gene.

## 3 $\mathcal{C}_1$: naming molecules

### *3.1 Terms*

Consider a new set of names $\mathcal{M}$ that is pairwise disjoint from $\mathcal{B}$ and $\mathcal{S}$. Terms of $\mathcal{C}_1$ are essentially those of $\mathcal{C}_0$ where domains have an extra *meta name* $m, m' \in \mathcal{M}$ that will point to new type of terms called *info* terms (denoted by $I, J, \dots$). Let $\mathcal{I}$ be a set of terminal symbols (distinct from all previous ones) called *informations* (think of protein or domain names). The grammar of $\mathcal{C}_1$ is:

$$
\begin{array}{lll}
D, D' ::= \mathsf{D}^a_m(x_1, \dots, x_k) & a \in \mathcal{B},\ m \in \mathcal{M},\ x_i \in \mathcal{S} & \text{(domains)} \\
I, J \quad ::= \mathsf{Info}_m & \mathsf{Info} \in \mathcal{I},\ m \in \mathcal{M} & \text{(info)} \\
T, S \quad ::= 0 \mid D \mid I \mid (T, S) \mid T \backslash v & \text{for } v \in \mathcal{S} \cup \mathcal{B} \cup \mathcal{M} & \text{(named terms)}
\end{array}
$$

Structural congruence coincides with the one defined earlier.

### *3.2 Graphical notation*

This simple extension has a natural impact on the graphical notation, as shown in Fig. 3 with an example of amino acid synthesis. *Info* nodes are represented by their type (Nucl., G, Ribosome, Prot. compl., Amino acid, Glycine) without drawing borders around them. Meta names that are shared by nodes induce thin straight hyper edges. Open meta ports are not drawn, and closed meta edges are represented with filled arrowheads (as in the Amino acid and Glycine nodes on the right hand side).

There are only two specific generators for $\mathcal{C}_1$, for all $\mathsf{Info} \in \mathcal{I}$:

$$
\begin{array}{ll}
\text{(Concretize)} & \mathsf{D}^a_m(x_1, \dots, x_k) \to \mathsf{D}^a_m(x_1, \dots, x_k), \mathsf{Info}_m \\
\text{(Abstract)} & \mathsf{D}^a_m(x_1, \dots, x_k), \mathsf{Info}_m \to \mathsf{D}^a_m(x_1, \dots, x_k)
\end{array}
$$

and again, rules can be generated by refinement and composition of generators.

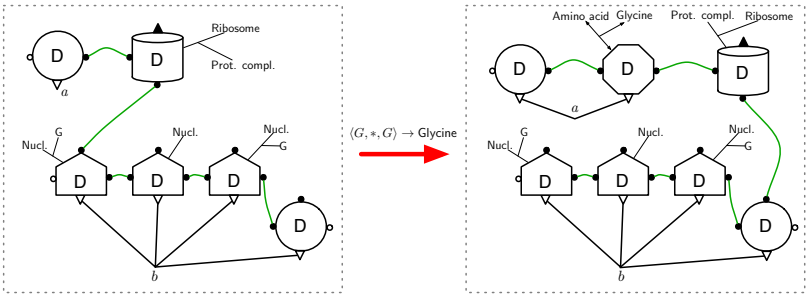



Fig. 3. Graphical illustration of the role of *info* nodes and *meta* names, with the rule for the RNA translation of a Glycine amino acid. Node shape is purely illustrative. A ribosome is bound to a guanine being part of an RNA strand (backbone $b$) and has started to assemble a new protein (backbone $a$). The next nucleotide on the right is of unspecified type followed by a $G$ nucleotide, this triplet $\langle G, *, G\rangle$ codes for the Glycine that is produced on the right.

### *3.3 Discussion*

With little symbol pushing burden we obtain a fairly expressive language which, at this stage, is already a reasonable candidate for representing most types of synthetic biology systems. It is noteworthy that the nature of an interaction can be expressed here as a form of type instantiation. One may think of $\mathcal{C}_0$ generators as polymorphic reaction types: $(\alpha, \beta)$ connect or $\alpha$ synthesize. They can be instantiated as $(A,B)$ connect or *Amino acid* synthesize.

This second step brings us closer to the $\kappa$-calculus of Danos and Laneve [10]. In fact, our calculus now encompasses $\kappa$ in a straightforward way (see Appendix A).

## 4 $\mathcal{C}_2$: placing molecules

As we already stressed in the previous sections, we have for now abstracted away from space and geometry: molecules are assumed to be floating in a uniform medium that lets domains react freely with each other. One could for instance encode a discrete compartment as *info* nodes attached to each domain and make sure they are compatible when two domains encounter. Yet, not only would this induce an explosion in the number of rules to write, but also entail a lot of book keeping rules in order to make sure that protein domains remain co-localized. We propose here to exploit our informal yet underlying relationship with bigraphs in order to add a simple notion of compartmentalization to our language.

### *4.1 Terms*

Let $\mathcal{V}$ be an infinite set of *parameter names* $\{X, Y, Z, \dots\}$ assumed to be pairwise disjoint from $\mathcal{S}$, $\mathcal{B}$ and $\mathcal{M}$. Let $\mathsf{C}$ be a terminal symbol, distinct from previous ones. Terms $P, Q, \dots$ of $\mathcal{C}_2$ are generated by the following extension of the grammar for $\mathcal{C}_1$:

$$T, S ::= \cdots \mid \mathsf{C}_m(T) \mid X \qquad m \in \mathcal{M},\ X \in \mathcal{V} \qquad \text{(local terms)}$$

$$P, Q ::= T \mid (T \parallel P) \mid P \backslash v \qquad v \in \mathcal{M} \cup \mathcal{S} \cup \mathcal{B} \qquad \text{(wide terms)}$$

Terms of the form $\mathsf{C}_m(T)$ denote compartments. They are nodes with a meta name, like domains, but have neither sites nor backbone. In the way defined in the previous section, this meta name allows one to specify a type of compartment: for instance $\mathsf{nucleus}, \mathsf{membrane} \in \mathcal{I}$ (one may also think of $\mathsf{region} \in \mathcal{I}$ to denote compartments with no physical boundaries).

Note also the use parameters as in $\mathsf{C}_m(X)$, where $X$ denotes the unspecified content of compartment $\mathsf{C}_m$. We use $\mathcal{V}(P)$ to denote the set of parameter names in $P$. For simplicity we consider here "linear terms", *i.e.* terms that do not contain multiple copies of the same parameter variables. It entails that a rule may delete parameters but not duplicate them.

Terms of $\mathcal{C}_2$ are either *local*, in which case we use $T, S$ to denote them, or *wide* in which case we use $P, Q$. The term $P = (T \parallel S)$ is a pattern requiring $T$ and $S$ to be separated by *exactly one* compartment boundary in any context; note that this differs from the interpretation of wide composition in bigraphs, where they may be separated by any number of boundaries. Hence we will see that $P$ has a match in both $(\mathsf{C}_m(T), S)$ and $(T, \mathsf{C}_m(S))$. We want to absorb here the projective view of membrane reactions introduced by Danos and Pradalier [11] and also present in a later work by Cardelli [4]. The underlying idea is that membrane curvature is a global property that one may not want to consider when expressing cellular mechanisms. This trait will turn out to be very useful when defining a minimal set of generators for $\mathcal{C}_2$.

**Definition 4.1** [Local contexts] A context $\mathbb{C}[\bullet]$ with exactly one hole is a *local context* if it is of the form:

$$\mathbb{C}[\bullet] ::= \bullet \mid \mathbb{C}[\bullet]\backslash u \mid \mathbb{C}[\bullet], T \quad u \in \mathcal{B} \cup \mathcal{S}$$

Note that the context $\mathsf{C}_m(\bullet)$ is not a local context. It is however a *derivable wide context* as we will see shortly.

Structural congruence for $\mathcal{C}_2$ extends the one of $\mathcal{C}_1$ with the following laws for wide composition of terms:

$$\begin{array}{lll}
\mathsf{C}_m(T) & \equiv \mathsf{C}_m(T') & \text{if } T \equiv T' \\
\mathsf{C}_m(T\backslash u) & \equiv \mathsf{C}_m(T)\backslash u & \text{if } u \neq m \\
(P\backslash u)\backslash v & \equiv (P\backslash v)\backslash u & \\
P\backslash u & \equiv (P\,\{v/u\})\backslash v & v \notin fn(P) \\
T\backslash u \parallel P & \equiv (T \parallel P)\backslash u & u \notin fn(P) \\
T \parallel P\backslash u & \equiv (T \parallel P)\backslash u & u \notin fn(T)
\end{array}$$

It is clear that any wide term is structurally congruent to a term of the form $(T_1 \parallel \cdots \parallel T_n)\backslash V$ (using the shorthand $P\backslash V$ for the restriction of the names of $V$). We sometimes write $P \parallel Q$ to denote the concatenation $P$ and $Q$ (in the style of list concatenation). Importantly a pattern of the form $T \parallel S \parallel T'$ specifies

that $T$ and $T'$ are exactly two compartment layers away from each other, and that $S$ is one compartment layer away from both $T$ and $T'$, we will call this distance *projective* because it does not take the orientation of the compartment borders, that will separate the terms in the context, into account. We shall see that valid matches for a wide term $(T_1 \parallel \cdots \parallel T_n)\backslash V$ will correspond to those in which the distance between $T_i$ and $T_{i+k}$ is exactly $k$, for all $i \in \{1, \ldots, n-k\}$.

### *4.2 Pattern matching*

For any wide term $P$, say that $P$ has *width* $w(P) = n$ if $P \equiv (T_1 \parallel \cdots \parallel T_n)\backslash V$ for some local terms $T_i$.

**Definition 4.2** [Projective distance]

Let $P$ be a wide term and $T_i, T_j$ two disjoint term occurrences in $P$. The *projective distance* of $T_i, T_j$ in $P$, written $\Delta_{T_i,T_j}(P)$ is inductively defined as:

$$
\begin{array}{lll}
\Delta_{T_i,T_j}(T_i, T_j) & = 0 & \\
\Delta_{T_i,T_j}(T, S) & = \Delta_{T_i,T_j}(T) & \text{if } T_i, T_j \notin S \\
\Delta_{T_i,T_j}(P\backslash u) & = \Delta_{T_i,T_j}(P) & \\
\Delta_{T_i,T_j}(\mathsf{C}_m(T)) & = \Delta_{T_i,T_j}(T) & \\
\Delta_{T_i,T_j}(\mathsf{C}_m(T), S) & = \Delta_{T_i,T_j}(T, S) + 1 & \text{if } T_i \in T \text{ and } T_j \in S \\
\Delta_{T_i,T_j}(T \parallel P) & = \Delta_{T_i,T_j}(P) & \text{if } T_i, T_j \in P \\
\Delta_{T_i,T_j}(T \parallel P) & = \Delta_{T_i,T_j}(T) & \text{if } T_i, T_j \in T \\
\Delta_{T_i,T_j}(T \parallel S \parallel P) & = \Delta_{T_i,T_j}(T \parallel P) + 1 & \text{if } T_i, T_j \notin S \\
\Delta_{T_i,T_j}(T \parallel S) & = \Delta_{T_i,T_j}(T, S) + 1 & \text{if } T_i \in T, T_j \in S
\end{array}
$$

In other terms, the projective distance between $T_i$ and $T_j$ is equal to the number of wide compositions and compartment layers that separate $T_i$ from $T_j$.

Given a wide term $P = (T_1 \parallel \cdots \parallel T_n)\backslash V$, we need to define contexts $\mathbb{C}^n[\bullet, \ldots, \bullet]$ with exactly $n$ holes in which one may embed $P$ while preserving nesting distance. Let *generic contexts* (with an arbitrary number of holes) be inductively defined as:

$$T_\bullet, S_\bullet ::= \bullet \mid T \mid (T_\bullet, S_\bullet) \mid (T_\bullet)\backslash u \mid \mathsf{C}_m\,(T_\bullet) \quad u \in \mathcal{B} \cup \mathcal{S} \cup \mathcal{M} \;\&\; m \in \mathcal{M}$$

For any such context $T_\bullet$ with exactly $k$ holes, we write $T_\bullet = \mathbb{C}^k[\bullet, \ldots, \bullet]$ or simply $T = \mathbb{C}^k$. Importantly, not all contexts of the form $\mathbb{C}^1$ is a local context since $\mathsf{C}_m(\bullet)$ is not local. Furthermore, not all contexts of $k$ holes will be valid placeholders for wide terms of width $k$. Rather than trying to enumerate valid contexts with $n$ holes we use a procedure that generates valid matches for terms of arbitrary width. We will then prove that this procedure is both sound and complete in the sense that it finds only correct matches for wide terms, and finds them all.

$$\text{(ax.)}\quad \frac{}{T \hookrightarrow_\bullet \mathbb{C}[\bullet]} \qquad \frac{P \hookrightarrow_\pi T_\bullet \quad v \notin fn(T_\bullet) \cup bn(T_\bullet)}{P\backslash v \hookrightarrow_\pi T_\bullet}\quad \text{(rest)}$$

$$\frac{P \hookrightarrow_\pi T_\bullet \quad m \text{ fresh} \quad (\!|\cdot\pi\cdot|\!) \not\to \bot}{P \hookrightarrow_{(\!|\cdot\pi\cdot|\!)} \mathsf{C}_m(T_\bullet)}\quad \text{(wrap)}$$

$$\frac{P \hookrightarrow_{\pi_0} T_\bullet \quad Q \hookrightarrow_{\pi_1} S_\bullet \quad \pi_0 \cdot \pi_1 \not\to \bot}{P \parallel Q \hookrightarrow_{\pi_0\cdot\pi_1} \mathbb{C}[T_\bullet, S_\bullet]}\quad \text{(comp)}$$

Table 1
The extension relation. Contexts $\mathbb{C}[\bullet]$ are the local contexts of Definition 4.1.

Let *projection constraints* $\pi$ be words on the alphabet $\Pi \stackrel{def}{=} \{(\!|, |\!), \bullet, \bot\}$. We use these constraints during the construction of a wide context $\mathbb{C}^n$, as an abstraction of the context that retains only the positions of compartments borders, symbols $(\!|$ and $|\!)$, and holes, symbol $\bullet$. In order to check that $\mathbb{C}^n$ is a valid context, it will suffice to make sure that the projection constraint is well-formed. For instance, the constraint $\pi = \bullet \cdot (\!|\cdot \bullet \cdot|\!) \cdot \bullet$ is an abstraction of an invalid context with exactly three holes, that would place the term $T \parallel S \parallel T'$ in an environment where $T$ and $T'$ would be at (projective) distance 0 instead of 2. Invalid constraints are detected during the construction of a wide context (cf. Table 1), using the reduction relation of Table 1.

**Definition 4.3** [Valid constraints] Let $\pi \in \Pi^*$ be a projection constraint. Let $\cdot$ denote the concatenation of words over the alphabet $\Pi$. Say that $\pi$ is *valid* if $\pi \not\to \bot$ with $\to \subseteq \Pi^* \times \Pi^*$ the least reflexive, transitive, and compatible relation engendered by:

$$\bullet \cdot (\!|\cdot (\!| \to \bot \qquad |\!)\cdot |\!) \cdot \bullet \to \bot \qquad |\!) \cdot (\!| \to \bot$$
$$\bullet \cdot \bullet \to \bot \qquad \bullet \cdot (\!|\cdot \bullet \cdot |\!) \cdot \bullet \to \bot \qquad \bot \cdot \pi \to \bot \qquad \pi \cdot \bot \to \bot$$

The inductive construction of the extension relation is given in Table 1. Let $\mu, \mu', \ldots$ denote (possibly empty) lists of parameter assignation of the form $[X_1 \leftarrow T_1]; \ldots; [X_n \leftarrow T_n]$ with $\mathcal{V}(T_i) = \emptyset$. We use $|\mu|$ to denote the set of parameter names in $\mu$, and $P\mu$ to denote $P$ in which parameters have been substituted according to $\mu$.

**Definition 4.4** [Matches] A wide context $\mathbb{C}^n[\bullet, \ldots, \bullet]$ with exactly $n$ holes and a parameter assignation list $\mu$ form a *match* $\langle \mathbb{C}^n, \mu \rangle$ for a wide term $P = (T_1 \parallel \cdots \parallel T_n)\backslash V$ in $S$ if and only if:

$$P \hookrightarrow_\pi \mathbb{C}^n \quad \text{and} \quad |\mu| = \mathcal{V}(P) \quad \text{and} \quad ((\mathbb{C}^n[T_1, \ldots, T_n]\mu)\backslash V)\sigma \equiv S$$

for some name substitution $\sigma$.

Furthermore, a pair $r = \langle P, Q \rangle$ with $w(P) = w(Q) = n$ and $\mathcal{V}(P) = \mathcal{V}(Q)$

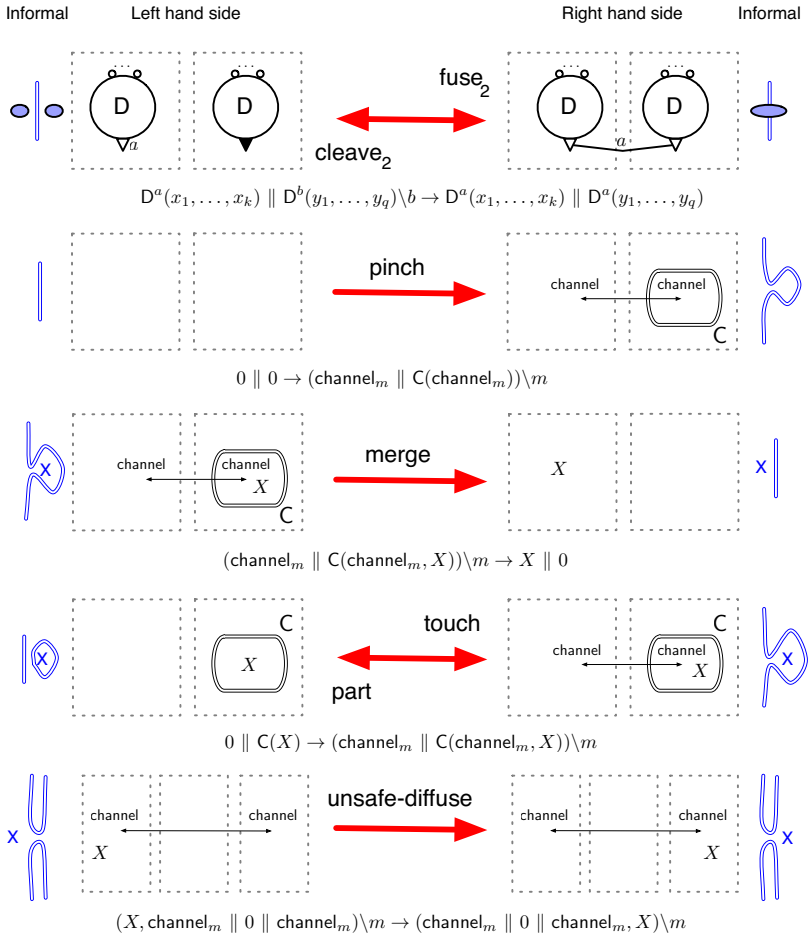



Fig. 4. Generators for $\mathcal{C}_2$.

generates a transition $T \to_r S$ if the match $\langle \mathbb{C}^n[\bullet, \ldots, \bullet], \mu \rangle$ for $P$ in $T$ is a match for $Q$ in $S$.

We conclude this section with the expected soundness and completeness results for our extension relation with respect to projective distance.

**Theorem 4.5 (Soundness)** *Let $\langle \mathbb{C}^n[\bullet, \ldots, \bullet], \mu \rangle$ be a match for a wide term $P$ in a local term $T$. For all disjoint local term occurrences $S, S' \in P$, we have $\Delta_{S,S'}(P) = \Delta_{S,S'}(T)$.*

**Theorem 4.6 (Completeness)** *Let $P = (T_1 \parallel \cdots \parallel T_n)\backslash V$ be a wide term and $\mathbb{C}^n[\bullet, \ldots, \bullet]$ be a generic context with exactly $n$ holes. Let also $T \equiv ((\mathbb{C}^n[T_1, \ldots, T_n]\mu)\backslash V)\sigma$ for some parameter assignation $\mu$ and name substitution $\sigma$.*

*If for all $i, j \leq n$ one has $\Delta_{T_i,T_j}(P) = \Delta_{T_i,T_j}(T)$, then $P \hookrightarrow_\pi \mathbb{C}^n$ is derivable, for some $\pi \in (\Pi \backslash \{\bot\})^*$.*

### *4.3 Generators*

The generators are presented in Fig. 4, keeping with the graphical convention introduced earlier. We add here compartments, represented as nodes with double line boundaries, and variables. Wide terms are simply represented next to each other. Crucially, the possibility to express compartment patches independently of their general curvature allows us to maintain a minimal set of generators. Rules specifying curvature are then obtained as refinements of these generators. The wide versions of the fuse and cleave generators now allow for the representation of transmembrane proteins (aka receptors). Note that we do not generalize the connect and disconnect generators to keep with the fact that protein-protein interactions are local.

The other generators rely on the intuition, sketched in an earlier work on bigraphs [15], that dynamic molecular compartments can be modeled using an intermediate step where two compartments are connected by a "neck". This neck, visible in generators pinch, merge, touch and unsafe-diffuse, is represented by two connected channel nodes, which are particular *info* nodes. In the unsafe-diffuse rule, they are used to indicate that molecules can translocate from one location to another, along the channel edge. This rule can be applied in order to populate a vesicle after pinch or touch, and until part or merge is applied.

At this stage our language is equipped with ways to model dynamic compartments and diffusion. Yet, consistency of the biological interpretation of $\mathcal{C}_2$ terms relies on a careful usage of the unsafe-diffuse rule. Indeed, nothing prevents modelers from using this generator to stretch a protein across several membranes by diffusing only a part of it, violating the desired invariant that only a backbone edge may cross a compartment (in the case of a receptor). In order to correct for this, we need to restrict diffusion to instances that will preserve biological soundness of terms. The final step in the design of our language is aimed at solving this question.

## 5 $\mathcal{C}_3$: moving molecules

### *5.1 Terms*

Let $\mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}$ be a family of $\mathsf{B}$ and $\mathsf{S}$ indexed terminal symbols (distinct from all others) with $\mathsf{B} \subseteq \mathcal{B}$ and $\mathsf{S} \subseteq \mathcal{S} \cup \mathcal{M}$. The grammar generating terms of $\mathcal{C}_3$ extends the previous one in the following way:

$$\begin{array}{lll} T, S & ::= \ldots & \text{(local terms)} \\ G, H & ::= T \mid \mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(T) \mid (G, H) & \text{(global terms)} \\ P, Q & ::= G \mid (P \parallel Q) \mid \ldots & \text{(wide terms)} \end{array}$$

where $\mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(T)$ denotes the fact that term $T$ describes a partial species, *i.e.* is either a connected component or a pattern that should be placed in a context that will make it connected. The sets $\mathsf{B}$ and $\mathsf{S}$ denote respectively the free backbone names

of the species and its free site and meta names. These names are kept separated for convenience because backbones will be allowed to cross membranes while *meta* and *site* names will not be shared by nodes that are not co-located in the same compartment. For instance, the expression $\mathsf{spec}^{\{a\}}_{\emptyset}((D^b_m(x), X)\backslash b, x, m)$ denotes a partial species that contains a domain $D^b_m(x)$ and that may only have a connection with other nodes outside the species boundaries by sharing the backbone name $a$.

The idea behind $\mathcal{C}_3$ is that although connectivity, *i.e.* transitive closure of name sharing, is a property one may not want to consider in general, it becomes relevant for some particular interactions including diffusion. We will come back to this in the section describing the new generators.

Structural congruence allows us to form $\mathsf{spec}$ nodes on demand. To do so, we extend previous structural laws with the following ones:

$$\frac{}{\mathsf{D}^a_m(x_1,\ldots,x_k) \equiv \mathsf{spec}^{\{a\}}_{\{m,x_1,\ldots,x_k\}}(\mathsf{D}^a_m(x_1,\ldots,x_k))}(\mathsf{init})$$

$$\frac{fn(A)\cap(\mathsf{B}\cup\mathsf{S})\neq\emptyset \quad \mathsf{B}'=\mathsf{B}\cup(fn(A)\cap\mathcal{B}) \quad \mathsf{S}'=\mathsf{S}\cup(fn(A)\cap\mathcal{S})}{\mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(T), A \equiv \mathsf{spec}^{\mathsf{B}'}_{\mathsf{S}'}(T, A)}(\mathsf{grow})$$

$$\frac{u\in\mathcal{B}\cup\mathcal{S} \quad \mathsf{B}'\stackrel{def}{=}\mathsf{B}-\{u\} \quad \mathsf{S}'\stackrel{def}{=}\mathsf{S}-\{u\}}{\mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(T)\backslash u \equiv \mathsf{spec}^{\mathsf{B}'}_{\mathsf{S}'}(T\backslash u)} \qquad \frac{T\equiv T'}{\mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(T) \equiv \mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(T')}$$

Where $A$ is either a $\mathsf{domain}$ node or an $\mathsf{info}$ node. Intuitively, the left-to-right orientation of the above first three equations allows one to capture more knowledge about connectivity, while the other direction is forgetful. If one wishes to consider diffusion of vesicles, one needs the additional rule:

$$\frac{fn(T')\cap(\mathsf{B}\cup\mathsf{S})\neq\emptyset \quad \mathsf{B}'=\mathsf{B}\cup(fn(T')\cap\mathcal{B}) \quad \mathsf{S}'=\mathsf{S}\cup(fn(T')\cap\mathcal{S})}{\mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(T), \mathsf{C}_m(T') \equiv \mathsf{spec}^{\mathsf{B}'}_{\mathsf{S}'}(T, \mathsf{C}_m(T'))}$$

that allows one to encompass compartments in the recognition of molecular species.

In order to ease the understanding of the generators presented in the next section, let us give a simple example of the usage of a species term in a pattern. Consider the term $P = (\mathsf{spec}^a_{\emptyset}(X) \parallel \mathsf{spec}^a_{\emptyset}(Y))\backslash a$ which denotes a transmembrane complex split in two parts $X$ and $Y$ on both sides of a membrane. We wish to find a match for $P$ in the term:

$$T = \big(\mathsf{D}^a_{m_1}(x), \mathsf{SH2}_{m_1}, \mathsf{C}_{m_2}(\mathsf{D}^a_{m_3}(y), \mathsf{D}^b_{m_4}(y))\big) \setminus \{a, b, x, y, m_i\}$$

To do so, we first need to turn $T$ into a form that makes the desired connectivity apparent:

$$\begin{aligned} T \equiv\ & \big(\mathsf{spec}^a_{\emptyset}(\mathsf{D}^a_{m_1}(x), \mathsf{SH2}_{m_1}\setminus\{x, m_1\}), \\ & \mathsf{C}_{m_2}(\mathsf{spec}^a_{\emptyset}((\mathsf{D}^a_{m_3}(y), \mathsf{D}^b_{m_4}(y))\setminus\{b, y, m_3, m_4\})\backslash m_2)\big)\backslash a \end{aligned}$$

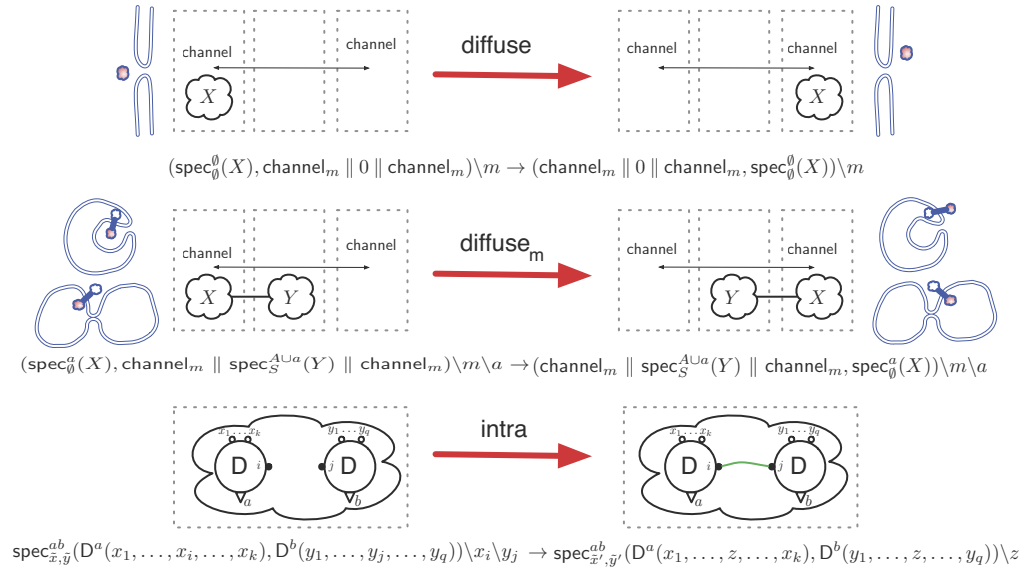



Fig. 5. $\mathcal{C}_3$ generators. In the intra generator, let $\tilde{x} \stackrel{def}{=} \{x_1,\ldots,x_k\}$ and $\tilde{y} \stackrel{def}{=} \{y_1,\ldots,y_q\}$, $\tilde{x}' \stackrel{def}{=} \tilde{x}\{z/x_i\}$ and $\tilde{y}' \stackrel{def}{=} \tilde{y}\{z/y_j\}$

Then, using the extension relation, we generate a context for $P$

$$P \hookrightarrow_{\bullet\cdot(\!|\cdot\bullet\cdot|\!)} (\bullet, \mathsf{C}_m(\bullet)) = \mathbb{C}^2[\bullet,\bullet]$$

which, together with a list of parameter assignations

$$\mu \stackrel{def}{=} [X \leftarrow (\mathsf{D}^a_{m_1}(x), \mathsf{SH2}_{m_1})\backslash\{x, m_1\}]; [Y \leftarrow (\mathsf{D}^a_{m_3}(y), \mathsf{D}^b_{m_4}(y))\backslash\{b, y, m_3, m_4\}]$$

defines a valid match for $P$ in $T$. One verifies that, indeed:

$$(\mathbb{C}[\mathsf{spec}^a_{\emptyset}(X), \mathsf{spec}^a_{\emptyset}(Y)]\mu)\{m_2/m\}\backslash a \equiv T$$

### *5.2 Generators*

Generators are given in Fig. 5. They extend the generators of all previous stages, to the exception of the unsafe-diffuse rule that is replaced by its safe counterparts. We keep with the graphical conventions introduced earlier, and use cloud nodes to denote (partial or total) species.

As one may see in Fig. 5, we now have two generators for diffusion. The first one models classical diffusion: a total species may move from one compartment connected to another *via* a channel. The second generator models diffusion of transmembrane species: two partial and parametric species denote, respectively, both sides of a transmembrane complex. The side of the complex whose content is $X$ may translocate while the other side stays in its current location. The result of this operation in the two possible projections, is informally depicted on both

sides of the generator and corresponds to the diffusion of a transmembrane complex along the neck. Finally, the intra generator stands for intra-molecular complex formation [2].

**Definition 5.1** [Mixture] Say that a term $P$ is a *mixture* if:

- $w(P) = 1$, $fn(P) = \emptyset$ and $P$ is parameter free
- Site edges have exactly two sites and do not cross compartments
- Backbone hyper edges cross at most one compartment
- $P$ is structurally equivalent to a term that contains no species node.

The last condition essentially states that species nodes that are present in a mixture are derivable from a species free mixture to which the above structural congruence rules have been applied. To ensure this one simply needs to verify the simple syntactical condition:

**Proposition 5.2** *A global term of the form $G = \mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(T)$ is a mixture if and only if $T$:*

- *$T$ is a mixture.*
- *$fn(T) = \mathsf{B} \cup \mathsf{S}$.*
- *$T$ is a connected component.*

The above proposition guarantees that one may always eliminate species nodes of the form $\mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(T)$ from a mixture, provided the sets $\mathsf{B}$ and $\mathsf{S}$ capture the free names of $T$ and provided $T$ defines a single connected set of agents. Note however, that general global terms need not be mixtures and one may have occurrences of species node in rules that do not satisfy this condition as it is for instance the case in the $\mathsf{diffuse}_{\mathsf{m}}$ generator. Yet not all species node make sense in a $\mathcal{C}_3$ expression. For instance $\mathsf{spec}^{\emptyset}_{\emptyset}(\mathsf{D}^{a}_{m}(x), X)$ will never have a match in any mixture since the structural congruence for species node introduction will always insure that the free names $a, m$ and $x$ will appear in the superscript and subscript of spec. The following proposition defines *well-formed* expressions with species nodes:

**Proposition 5.3** *For any term $G = \mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(T)$, there exists a mixture $M$ such that $G$ has a match in $M$ if and only if:*

- *$fn(T) \subseteq \mathsf{B} \cup \mathsf{S}$*
- *and either:*
  - *$T$ is connected*
  - *$\mathcal{V}(T) \neq \emptyset$ and $fn(T) \neq \emptyset$*
  - *$fn(T) = \emptyset$ and $T = X_1, \ldots, X_n$ for some parameters $X_i$ .*

Note that the second condition says that either $T$ needs to be already connected in the expression or leave "room enough" so that the context will make $T$ connected. It is easy to check that all the generators introduced in Fig. 5 satisfy this condition.

[2] This generator cannot be obtained as a refinement of connect since $\mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(\bullet)$ is not a valid local context.

**Lemma 5.4 (Preservation)** *Let $\mathcal{R}$ be a set of generated rules and let $P$ be a mixture. If $P \to_r Q$ with $r \in \mathcal{R}$ then $Q$ is a mixture.*

As a corollary of the above lemma and Proposition 5.2, one has that a term containing $\mathsf{spec}^{\mathsf{B}}_{\mathsf{S}}(T, S)$ can only have a match in a mixture where $T$ and $S$ are part of the same connected component, which is a guarantee of the soundness of the intra generator.

# 6 Conclusion

The idea that models of signaling pathways or protein assembly should be considered as programs is now wending its way through the systems biology crowd. This is an appealing fact to language theoreticians, because it implies that one needs to accomplish in Systems Biology the same mutation that was accomplished in software engineering, when programs became too cumbersome and unwieldy to be developed in a non uniform way. This suggests that systems biology will soon require the development of high level languages, debuggers, and IDEs to compensate for the increasing gap between accumulation of data and its representation in executable models. The work we have presented here is an attempt to comply with Fontana's requirement that *"a model should be a data structure that contains a transparent, formal, and executable representation of the facts it rests upon"* [13]. In order to do so, we have structured our language in order to be able to tune the resolution level of the entities we wanted to describe: from anonymous domains, to molecular species, and from membrane patches to full fledged compartments.

We have already mentioned several approaches that were conducted with similar motivations, some of which we took inspiration from. Yet, we believe that the presented language offers a level of expressivity that was not accessible before in a single formalism. In particular we should mention that our language strictly contains the $\kappa$-calculus and corresponds to a particular class of bigraphical reactive systems that is yet to be defined formally [3]. Obviously, expressiveness and relative ease of use is not enough and future work should aim at developing quantitative simulation and analysis techniques. Here again, previous works have paved the way for such developments. In particular, proximity with the $\kappa$-calculus for which such analysis and simulation technique have been defined [8,7] and the stochastic semantics for bigraphs [15], should be of great help.

# References

[1] Andrei, O. and H. Kirchner, *Graph rewriting and strategies for modeling biochemical networks*, in: *Proc. SYNASC*, 2007, pp. 407–414.

[2] Bacci, G., D. Grohmann and M. Miculan, *A framework for protein and membrane interactions*, in: *Proc. MeCBIC'09*, 2009, pp. 19–33.

[3] Cardelli, L., *Brane calculi - interactions of biological membranes*, in: *Computational Methods in Systems Biology*, Springer, 2004 pp. 257–278.

[3] This may prove to be a complex task, since projectivity is not a trivial concept to capture with the standard definition of bigraphs. See Ref. [6] for some hints on how to do this.

[4] Cardelli, L., *Bitonal membrane systems - interactions of biological membranes*, Theoretical Computer Science **404** (2008).

[5] Chabrier, N. and F. Fages, *Symbolic model checking of biochemical networks*, in: *Proc. CMSB'03*, LNCS **2602**, 2003, pp. 146–162.

[6] Damgaard, T. C. and J. Krivine, *A generic language for biological systems based on bigraphs*, Technical Report 115, IT University of Copenhagen (2009).

[7] Danos, V., J. Feret, W. Fontana, R. Harmer and J. Krivine, *Abstracting the differential semantics of rule-based models: exact and automated model reduction*, in: *IEEE Symposium LICS*, 2010, pp. 362–381.

[8] Danos, V., J. Féret, W. Fontana and J. Krivine, *Scalable simulation of cellular signaling networks*, in: *Proc. APLAS'07*, LNCS **4807**, 2007, pp. 139–157.

[9] Danos, V. and C. Laneve, *Core formal molecular biology*, in: *Proc. ESOP'03*, LNCS **2618**, 2003, pp. 302–318.

[10] Danos, V. and C. Laneve, *Graphs for formal molecular biology*, in: *Proc. CMSB'03*, LNCS **2602**, 2003, pp. 34–46.

[11] Danos, V. and S. Pradalier, *Projective brane calculus*, in: *Proc. CMSB'04*, 2004, pp. 134–148.

[12] Faeder, J. R., M. L. Blinov and W. S. Hlavacek, *Rule based modeling of biochemical networks*, Complexity (2005), pp. 22–41.

[13] Fontana, W., *Systems biology, models, and concurrency*, in: *Proc. POPL'08*, 2008, pp. 1–2.

[14] John, M., C. Lhoussaine, J. Niehren and C. Versari, *Biochemical reaction rules with constraints*, in: *Proc. ESOP 2011*, LNCS **6602**, 2011, pp. 338–357.

[15] Krivine, J., R. Milner and A. Troina, *Stochastic bigraphs*, in: *Proceedings of MFPS XXIV*, ENTCS **218**, 2008, p. 7396.

[16] Milner, R., "The Space and Motion of Communicating Agents," Cambridge University Press, 2009.

[17] Phillips, A. and L. Cardelli, *Efficient, correct simulation of biological processes in the stochastic pi-calculus*, in: *CMSB*, 2007, pp. 184–199.

[18] Priami, C. and P. Quaglia, *Beta binders for biological interactions*, in: *Computational Methods in Systems Biology*, LNCS **3082**, 2005, pp. 20–33.

[19] Păun, G. and F. J. Romero-Campero, *Membrane computing as a modeling framework. cellular systems case studies*, in: *Formal Methods for Computational Systems Biology*, LNCS **5016**, 2008, pp. 168–214.

[20] R.Barbuti, A.Maggiolo-Schettini, P.Milazzo and A.Troina, *A calculus of looping sequences for modelling microbiological systems*, Fundamenta Informaticæ**72** (2006), pp. 21–35.

[21] Regev, A., E. M. Panina, W. Silverman, L. Cardelli and E. Shapiro, *Bioambients: An abstraction for biological compartments*, Theoretical Computer Science **325** (2004), pp. 141–167.

[22] Regev, A., W. Silverman and E. Y. Shapiro, *Representation and simulation of biochemical processes using the $\pi$-calculus process algebra*, in: *Pacific Symposium on Biocomputing*, 2001, pp. 459–470.

# A Retrieving the $\kappa$-calculus.

In this section we show how one may naturally represent any $\kappa$-calculus model at the $\mathcal{C}_1$ level of our language. As the encoding is rather straightforward from a technical point of view, we shall simply describe here the translation of a particular example. We then show how $\mathcal{C}_3$ enables us to go beyond what one can express in $\kappa$.

## *A.1 The $\kappa$-calculus*

We consider here the definition of $\kappa$ that is implemented in the $\kappa$-simulator KaSim [4] . Terms of the $\kappa$-calculus are built on the following grammar:

**Definition A.1 ($\kappa$-Agents)**

| | | | | |
|---|---|---|---|---|
| *(i)* | *agent* | $a$ | $::= N(\sigma)$ | |
| *(ii)* | *agent name* | $N$ | $::= A \in \mathcal{A}$ | |
| *(iii)* | *interface* | $\sigma$ | $::= \emptyset \mid s, \sigma$ | |
| *(iv)* | *site* | $s$ | $::= n_\iota^\lambda$ | |
| *(v)* | *site name* | $n$ | $::= x \in \mathcal{S}$ | |
| *(vi)* | *internal state* | $\iota$ | $::= \epsilon$ | *(any state)* |
| | | | $\mid m \in \mathbb{V}$ | |
| *(vii)* | *binding state* | $\lambda$ | $::= \epsilon$ | *(free)* |
| | | | $\mid -$ | *(semi-link)* |
| | | | $\mid ?$ | *(wild-card)* |
| | | | $\mid i \in \mathbb{N}$ | |

Expressions are simply formed by concatenation of agents $E ::= a, E \mid \emptyset$. Every agent represents a molecular entity (such as a protein) that has *sites* that can be used for complex formation (*i.e.* binding with other sites). For instance the expression:

$$\texttt{EGF(r}^1\texttt{),ErbB1(l}^1\texttt{,CR}^3\texttt{,Y1016}_p\texttt{,Y1092}_p^?\texttt{),EGF(r}^2\texttt{),ErbB1(l}^2\texttt{,CR}^3\texttt{,Y1092}_u^-\texttt{)}$$

corresponds to a molecular soup containing two instances of the agent `ErbB1` (a membrane receptor for the *epidermial growth factor* protein) and two instances of the agent `EGF` (the growth factor signal). In $\kappa$, each agent name comes with a fixed *signature* $\Sigma : N \to \mathcal{P}(\mathcal{S})$ that specifies the names of the sites each instance has. For instance $\Sigma(\texttt{ErbB1}) \stackrel{def}{=} \{\texttt{l}, \texttt{CR}, \texttt{Y1016}, \texttt{Y1092}, \dots\}$. Note that the protein `ErbB1` has in fact numerous tyrosine domains (whose name are of the form `Yxxx` where `xxx` corresponds to some amino acid position in the chain) that we do not list here. As

[4] http://kappalanguage.org

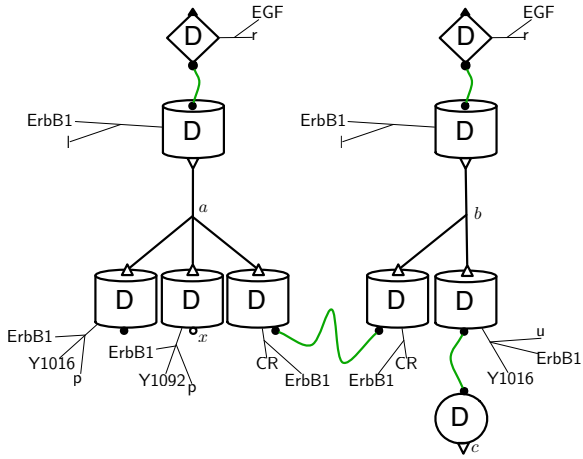



Fig. A.1. Representation of the $\kappa$ expression into $\mathcal{C}_1$.

a convention in $\kappa$, one does not represent sites that take no part in a given rule. In the example above, the site `Y1092` is left aside in one of the instances of `ErbB1`.

The superscript on a site indicate its *binding state*. The empty superscript $\epsilon$ marks a site that is free of any connection, _ indicates that the site is bound to an unspecified partner, ? indicates a site that is either free or bound and an integer is used to denote an explicit edge, as the one that connects the site `r` of the leftmost `EGF` to the site `l` of `ErbB1`.

The subscript on a site indicate its *internal state*. This is essentially a placeholder for a tag that serves to identify sites that have been chemically modified. Note that the absence of tag indicates that one does not care about its internal state in the expression.

### *A.2 The $\kappa$-calculus in $\mathcal{C}_1$*

Fig. A.1 shows the $\mathcal{C}_1$ representation of the above $\kappa$-expression.

The convention we adopt for the representation of $\kappa$-terms is the following: we use a domain node for each site of the kappa expression to translate. Internal states, site and agent names are represented by *info* nodes. Sites that belong to the same $\kappa$-agents will share the same backbone. Sites that are connected in the $\kappa$ expression will be bound in the $\mathcal{C}_1$-term. Notice that the backbone of both instances of `ErbB1` are both open. This captures the fact that not all sites of the signature of `ErbB1` are present in the expression.

### *A.3 Expressiveness of $\mathcal{C}_3$*

As said, `ErbB1` proteins are in fact membrane receptors. `ErbB1` protein is composed of an extra cellular domain that holds the ligand binding site `l` and an intra cellular domain that bears the other interaction sites. Now that we have represented our expression in a richer language, it becomes natural to represent these facts as we show in Fig. A.2.

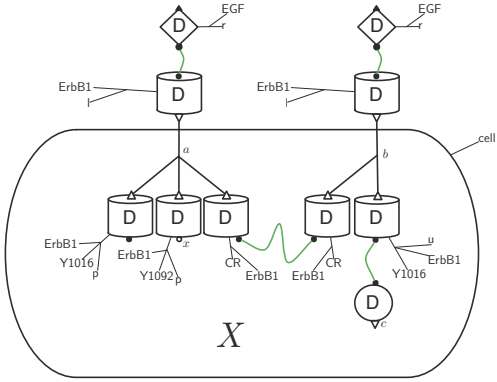



Fig. A.2. Adding compartments to the the $\kappa$ expression.

A key regulatory mechanism of the EGF pathway is called *receptor internalization*. It is a mechanism by which receptors become trapped in inner vesicles that may eventually bubble down to the cytoplasm of the cell. This prevents the receptor from binding to new incoming signals. It is not possible to represent this behavior in $\kappa$ for two reasons. The first reason, which we have already solved, is that there is no way to represent compartments in $\kappa$. The second reason is more subtle. Indeed, during receptor internalization, not only will `ErbB1` get trapped inside the vesicle, but along with it will be any protein complex attached to its extra cellular domain. In the example of Fig. A.2, one should capture also the `EGF` ligand that is bound to it. This is what we do in Fig. A.3 by defining an internalization rule that utilizes a `species` node.

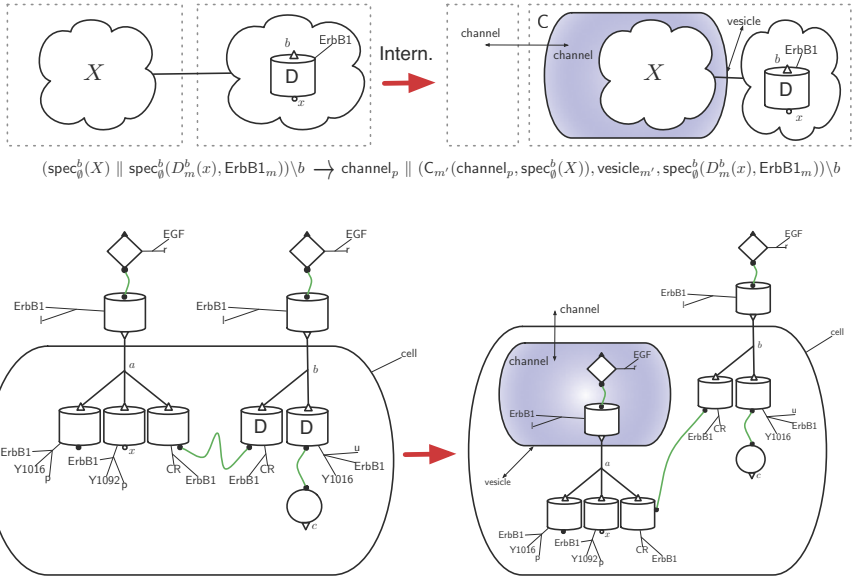



Fig. A.3. The Intern. rule is obtained by composition of the pinch and diffuse$_m$ generators and invoking the species node where it is needed. Below is the result of the application of this rule to our example. Notice that the receptor gets internalized together with its ligand protein `EGF`.