An Intuitive Automated Modelling Interface for Systems Biology

Ozan Kahramanoğulları
The Microsoft Research – University of Trento
Centre for Computational and Systems Biology *

Luca Cardelli
Microsoft Research Cambridge
Centre for Integrative Systems Biology at Imperial College

Emmanuelle Caron
Centre for Integrative Systems Biology at Imperial College
Centre for Molecular Microbiology and Infection, Imperial College

We introduce a natural language interface for building stochastic π calculus models of biological systems. In this language, complex constructs describing biochemical events are built from basic primitives of association, dissociation and transformation. This language thus allows us to model biochemical systems modularly by describing their dynamics in a narrative-style language, while making amendments, refinements and extensions on the models easy. We demonstrate the language on a model of Fcγ receptor phosphorylation during phagocytosis. We provide a tool implementation of the translation into a stochastic π calculus language, Microsoft Research's SPiM. [1] [2]

1 Introduction

Modelling of biological systems by mathematical and computational techniques is becoming increasingly widespread in research on biological systems. In recent years, pioneered by Regev and Shapiro's seminal work [16, 17], there has been a considerable amount of research on applying computer science technologies to modelling biological systems. Along these lines, various languages with stochastic simulation capabilities based on, for example, process algebra, term rewriting (see, e.g, [5, 7]) and Petri nets (see, e.g., [19, 9]) have been proposed. However, expressing biological knowledge in specialised modelling languages often requires a simultaneous understanding of the biological system and expert knowledge of the modelling language. Isolating and communicating the biological knowledge to build models for simulation and analysis is a challenging task both for wet-lab biologists and modellers.

Writing programs in simulation languages requires specialised training, and it is difficult even for the experts when complex interactions between biochemical species in biological systems are considered: the representation of different states of a biochemical species with respect to all its interaction capabilities results in an exponential blow up in the number of states. For example, when a protein with n different interaction sites is being modelled, this results in 2^n states, which needs to be represented in the model. Enumerating all these states by hand, also without inserting typos is a difficult task.

*This work has been done during Kahramanoğulları's appointment at the Department of Computing, Imperial College and Centre for Integrative Systems Biology at Imperial College. Kahramanoğulları acknowledges support of the UK Biotechnology and Biological Sciences Research Council through the Centre for Integrative Systems Biology at Imperial College (grant BB/C519670/1).

[1] We dedicate this paper to the memory of Dr. Emmanuelle Caron, who unexpectedly passed away in July 2009. It has been an honour to have worked with Emmanuelle, a biologist of the highest calibre.

[2] This work has been presented as oral presentation at the BioSysBio'09 Conference and Noise in Life'09 Meeting, both held in Cambridge in March 2009.

S. Barry Cooper & Vincent Danos (eds.): Fifth Workshop on
Developments in Computational Models—Computational Models From Nature
EPTCS 9, 2009, pp. 73–86, doi:10.4204/EPTCS.9.9

To this end, we introduce an intuitive front-end interface language for building process algebra models of biological systems: process algebras are languages that have originally been designed to formally describe complex reactive computer systems. Due to the resemblance between these computer systems and biological systems, process algebra have been recently used to model biological systems. An important feature of the process algebra languages is the possibility to describe the components of a system separately and observe the emergent behaviour from the interactions of the components (see, e.g., [1, 2]).

Our focus here is on π calculus [12], which is a broadly studied process algebra, also because of its compactness, generality, and flexibility. Because biological systems are typically highly complex and massively parallel, the π calculus is well suited to describe their dynamics. In particular, it allows the components of a biological system to be modelled independently, rather than modelling individual reactions. This allows large models to be constructed by composition of simple components. π calculus also enjoys an expressive power in the setting of biological models that exceeds, e.g., Petri nets [3].

In the following, we present a language that consists of basic primitives of association, dissociation and transformation. We impose certain consistency constraints on these primitive expressions, which are required for the models that describe the dynamics of biochemical processes. We give a translation algorithm into stochastic π calculus. Based on this, we present the implementation of a tool for automated translation of models into Microsoft Research's stochastic simulation language SPiM [14, 13], which can be used to run stochastic simulations on π calculus models. We demonstrate the language on a model of Fcγ receptor phosphorylation during phagocytosis. The implementation of the translation tool as well as further information is available for download at our website [3].

2 Species, Sites, Sentences and Models

We follow the abstraction of biochemical species as stateful entities with connectivity interfaces to other species: a species can have a number of sites through which it interacts with other species, and changes its state as a result of the interactions [6, 11]. In Section 4, we use this idea to design a natural language-like syntax for building models. The models written in this language are then automatically translated into a SPiM program by a translation algorithm. This is done by mapping the sentences of the language into events constructed from the basic primitives, which are then compiled into executable process expressions in the SPiM language.

There is a countable set of species $A, B, C, \ldots$. Each species has a number of sites $a, b, c, \ldots$ with which it can bind to other species or unbind from other species when they are already bound. We write sentences that describe the 'behaviour' of each species with respect to their sites. There are three kinds of sentences, i.e., *associations*, *dissociations*, and *transformations*. We define the sentences as

$$\langle \mathsf{type}, (A,a), (B,b), Pos, Neg, r \rangle$$

where $\mathsf{type} \in \{association, dissociation, transformation\}$ is the type of the sentence. The pairs (A,a) and (B,b) are called the *body* of the sentence. The sets *Pos* and *Neg* are called the *conditions* of the sentences. (A,a) and (B,b) are pairs of species and sites, and *Pos* and *Neg* are sets of such pairs of species and sites. If the sentence is an *association*, it describes the event where the site a on species A associates to the site b on species B if the sites on species in *Pos* are already bound and those in *Neg* are already unbound. If it is a *dissociation* sentence, it describes the dissociation of the site a on species A from the site b on species B. A *transformation* sentence describes the event of species A transforming into

[3] http://www.doc.ic.ac.uk/∼ozank/pim.html

species B, where B can be empty. In this case, this describes the decay of species A. In transformation sentences, sites a and b must be empty, since transformations are site independent. $r \in \mathbb{R}^+$ denotes the rate of the event that the sentence describes. Then a model $\mathscr{M}$ is a set of such sentences. In Section 4, we give a representation of these sentences in natural-language. For example, a sentence of the form $\langle association, (A,a), (B,b), \{(A,c)\}, \{\}, 1.0 \rangle$ is given with the following English sentence.

```
site a on A associates site b on B with rate 1.0 if site c on A is bound
```

We denote with $\mathsf{species}(\mathscr{M})$ all the species occurring in the body of the sentences of $\mathscr{M}$. The function $\mathsf{sites}(\mathscr{M},A)$ denotes the sites of the species A that occur in the body of all the sentences of $\mathscr{M}$. $\mathsf{sites}(Pos,A)$ denotes the sites of the species A in Pos (similarly for Neg). For any set $\mathscr{A}$, $\wp(\mathscr{A})$ denotes the powerset of $\mathscr{A}$.

2.1 Conditions on Sentences

Given a model $\mathscr{M}$, we impose several conditions on its sentences.

1. **Sentences contain relevant species.** The species in the condition of each sentence must be a subset of those in the body of the sentence.
2. **Conditions of the sentences are consistent.** For every sentence of the form $\langle \mathsf{type}, (A,a), (B,b), Pos, Neg, r \rangle$, we have that $Pos \cap Neg = \emptyset$.
3. **All the sites in the conditions are declared in the model.** For every sentence of the form $\langle \mathsf{type}, (A,a), (B,b), Pos, Neg, r \rangle$, we have that $\mathsf{sites}(Pos,A) \subseteq \mathsf{sites}(\mathscr{M},A)$, $\mathsf{sites}(Neg,A) \subseteq \mathsf{sites}(\mathscr{M},A)$, $\mathsf{sites}(Pos,B) \subseteq \mathsf{sites}(\mathscr{M},B)$ and $\mathsf{sites}(Neg,B) \subseteq \mathsf{sites}(\mathscr{M},B)$.
4. **Association sentences associate unbound species.** For every association sentence $\langle association, (A,a), (B,b), Pos, Neg, r \rangle$, we have that $(A,a), (B,b) \in Neg$.
5. **Dissociation sentences dissociate bound species.** For every dissociation sentence $\langle dissociation, (A,a), (B,b), Pos, Neg, r \rangle$, we have that $(A,a), (B,b) \in Pos$.
6. **Transformation sentences are unbound at all sites.** For every transformation sentence $\langle transformation, A, B, Pos, Neg, r \rangle$, we have that $Pos = \emptyset$ and $Neg = \{(A,x) \mid x \in \mathsf{sites}(\mathscr{M},A)\}$.

When these conditions hold, we can map the sentences of a model to another representation where the role of the conditions become more explicit. In the following, for a model $\mathscr{M}$, we describe the states of its species as subsets of its sites, where bound sites are included in the set describing the state. For example, for a species A with binding sites $\mathsf{sites}(\mathscr{M},A) = \{a,b\}$, the set $\wp(\mathsf{sites}(\mathscr{M},A)) = \{\{\},\{a\},\{b\},\{a,b\}\}$ is the set of all its states. Then $\{a\}$ is the state where site a on A is bound and site b on A is unbound. We map each sentence $\langle \mathsf{type}, (A,a), (B,b), Pos, Neg, r \rangle$ to a sentence of the form

$$\langle \mathsf{type}, (A,a), (B,b), \mathsf{states}(A), \mathsf{states}(B), r \rangle$$

where $\mathsf{states}(A)$ and $\mathsf{states}(B)$ are obtained as follows.

$$\mathsf{states}(A) = \{\, \mathscr{S} \in \wp(\mathsf{sites}(\mathscr{M},A)) \mid ((A,x) \in Pos \Rightarrow x \in \mathscr{S}) \wedge (x \in \mathscr{S} \Rightarrow (A,x) \notin Neg) \,\}$$

This representation allows us to impose another condition on the sentences:

7. **There are no overlapping conditions in the sentences.** For any two sentences of a model $\mathscr{M}$ of the form $\langle \mathsf{type}_1, (A,a), (B,b), Pos_1, Neg_1, r\rangle$ and $\langle \mathsf{type}_2, (A,a), (B,b), Pos_2, Neg_2, r\rangle$ where $\mathsf{type}_1 = \mathsf{type}_2$, we obtain $\mathsf{states}(A)_1$ and $\mathsf{states}(B)_1$, for the first and $\mathsf{states}(A)_2$ and $\mathsf{states}(B)_2$, for the second sentence. Then we have that
   - if $\mathsf{states}(A)_1 = \mathsf{states}(A)_2$ then it must be that $\mathsf{states}(B)_1 \cap \mathsf{states}(B)_2 = \emptyset$;
   - if $\mathsf{states}(B)_1 = \mathsf{states}(B)_2$ then it must be that $\mathsf{states}(A)_1 \cap \mathsf{states}(A)_2 = \emptyset$;
   - if $\mathsf{states}(A)_1 \neq \mathsf{states}(A)_2$ and $\mathsf{states}(B)_1 \neq \mathsf{states}(B)_2$ then it must be that $\mathsf{states}(A)_1 \cap \mathsf{states}(A)_2 = \emptyset$ and $\mathsf{states}(B)_1 \cap \mathsf{states}(B)_2 = \emptyset$.

Example 1 *Consider the models $\mathscr{M}_1$.*

$$\begin{aligned}\mathscr{M}_1 = \{\, &\langle association, (A,a), (B,b), \{(B,f)\}, \{(C,c), (A,a), (B,f)\}, 1.0\rangle,\\ &\langle dissociation, (A,a), (B,b), \{(B,b)\}, \{\}, 1.0\rangle,\\ &\langle transformation, A, B, \{\}, \{\}, 1.0\rangle,\\ &\langle association, (D,d), (E,e), \{\}, \{(D,d), (E,e)\}, 2.0\rangle,\\ &\langle association, (D,d), (E,e), \{\}, \{(D,d), (E,e)\}, 4.0\rangle\,\}\end{aligned}$$

This model does not fulfill any of the conditions above: in the first sentence, (1.) $C \notin \{A,B\}$; (2.) $(B,f) \in Pos$ and $(B,f) \in Neg$; (3.) $f \notin \{b\}$; (4.) $(B,b) \notin \{(C,c), (A,a), (B,f)\}$. In the second sentence, (5.) $(A,a) \notin \{(B,b)\}$. In the third sentence, (6.) $\{\} \neq \{(A,a)\}$. (7.) In the fourth and fifth sentences, $\mathsf{states}(D)_1 = \{\{\}\} = \mathsf{states}(D)_2$ *and* $\mathsf{states}(E)_1 = \{\{\}\} = \mathsf{states}(E)_2$.

Example 2 *The model $\mathscr{M}_2$ fulfills all the conditions above.*

$$\begin{aligned}\mathscr{M}_2 = \{\, &\langle association, (A,a_1), (B,b), \{\}, \{(A,a_1), (B,b)\}, 1.0\rangle,\\ &\langle association, (A,a_2), (C,c), \{\}, \{(A,a_2), (C,c)\}, 1.0\rangle,\\ &\langle dissociation, (A,a_1), (B,b), \{(A,a_1), (A,a_2), (B,b)\}, \{\}\}, 2.0\rangle,\\ &\langle dissociation, (A,a_1), (B,b), \{(A,a_1), (B,b)\}, \{(A,a_2)\}\}, 4.0\rangle\,\}\end{aligned}$$

3 Translation into Stochastic π calculus

We use the representation of the states of species as sets of their sites to map models to stochastic π calculus specifications. For this purpose, we first map each model to a *compile map*. Let us first recall some of the definitions of stochastic π calculus, implemented in SPiM, as they can be found in [13].

3.1 Stochastic π calculus

Definition 3 *Syntax of stochastic π calculus. Below $fn(P)$ denotes the set of names that are free in P.*

$E ::=$	$\emptyset$	*Empty*	$P,Q ::=$	Σ	*Summation*
$\mid$	$E, X(\tilde{m}) = P$	*Definition*	$\mid$	$X(\tilde{n})$	*Instance*
$\mid$		$fn(P) \subseteq \tilde{m}$	$\mid$	$P \mid Q$	*Parallel*
			$\mid$	$\mathtt{new}\, x\, P$	*Restriction*

$\Sigma ::=$	$\mathbf{0}$	*Null*	$\pi ::=$	$?x(\tilde{m})$	*Input*
$\mid$	$\pi\,; P + \Sigma$	*Action*	$\mid$	$!x(\tilde{n})$	*Output*
			$\mid$	τ_r	*Delay*

Expressions above are considered equivalent up to the least congruence relation given by the equivalence relation $\equiv$ defined as follows.

$$\begin{array}{rcl} P \mid \mathbf{0} & \equiv & P \\ P \mid Q & \equiv & Q \mid P \\ P \mid (Q \mid R) & \equiv & (P \mid Q) \mid R \\ X(\tilde{n}) & \equiv & P_{\{\tilde{n}/\tilde{m}\}} \text{ if } X(\tilde{m}) = P \end{array} \qquad \begin{array}{rcl} \texttt{new}\, x\, \mathbf{0} & \equiv & \mathbf{0} \\ \texttt{new}\, x\, \texttt{new}\, y\, P & \equiv & \texttt{new}\, y\, \texttt{new}\, x\, P \\ \texttt{new}\, x\, (P \mid Q) & \equiv & P \mid \texttt{new}\, x\, Q \text{ if } x \notin fn(P) \end{array}$$

3.2 Compile Maps

We map models into *compile maps*, denoted with $\mathscr{C}$. A compile map is a set of expressions that we call *process descriptions* for each species $A \in \mathsf{species}(\mathscr{M})$. For a model $\mathscr{M}$, the process description of species $A \in \mathsf{species}(\mathscr{M})$, denoted with $\mathsf{P}(A)$, is the pair $\langle A, \mathsf{actions}(A)\rangle$. Here, $\mathsf{actions}(A)$ is the set collecting $\mathsf{actions}(A,\mathscr{S})$ for every $\mathscr{S} \in \wp(\mathsf{sites}(\mathscr{M},A))$.

$$\mathsf{actions}(A,\mathscr{S}) = \langle \mathscr{S}, \mathsf{assoc}(A,\mathscr{S}), \mathsf{dissoc}(A,\mathscr{S}), \mathsf{transform}(A,\mathscr{S})\rangle$$

We define $\mathsf{assoc}(A,\mathscr{S})$ as the set of $\mathsf{assoc}(A,\mathscr{S},a)$ for every $a \in \mathsf{sites}(\mathscr{M},A)$.

$$\mathsf{assoc}(A,\mathscr{S},a) = \langle a, \mathsf{assocPartners}(A,\mathscr{S},a)\rangle$$

where $\mathsf{assocPartners}(A,\mathscr{S},a)$ is the set

$$\begin{array}{l} \{\langle B, b, \mathsf{states}(B), r\rangle \mid \\ \qquad (\langle \mathit{association}, (A,a), (B,b), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathscr{M} \wedge \mathscr{S} \in \mathsf{states}(A)) \\ \qquad \vee (\langle \mathit{association}, (B,b), (A,a), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathscr{M} \wedge \mathscr{S} \in \mathsf{states}(A))\}. \end{array}$$

We define $\mathsf{dissoc}(A,\mathscr{S})$, similarly, as the set of $\mathsf{dissoc}(A,\mathscr{S},a)$ for every $a \in \mathsf{sites}(\mathscr{M},A)$.

$$\mathsf{dissoc}(A,\mathscr{S},a) = \langle a, \mathsf{dissocPartners}(A,\mathscr{S},a)\rangle$$

where $\mathsf{dissocPartners}(A,\mathscr{S},a)$ is the set

$$\begin{array}{l} \{\langle B, b, \mathsf{states}(B), r\rangle \mid \\ \qquad (\langle \mathit{dissociation}, (A,a), (B,b), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathscr{M} \wedge \mathscr{S} \in \mathsf{states}(A)) \\ \qquad \vee (\langle \mathit{dissociation}, (B,b), (A,a), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathscr{M} \wedge \mathscr{S} \in \mathsf{states}(A))\}. \end{array}$$

If $\mathscr{S} = \emptyset$, the set $\mathsf{transform}(A,\mathscr{S})$ is defined as

$$\{\langle B, r\rangle \mid (\langle \mathit{transformation}, A, B, \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathscr{M}\}.$$

Otherwise, it is $\emptyset$.

Example 4 *Consider the model $\mathscr{M}_2$ in Example 2. We have that the compile map $\mathscr{C}_2$ for this model is as follows.*

$$\begin{array}{l} \{\ \langle A, \{\ \langle \{\}, \{(a_1, \{(B,b,\{\{\}\},1.0)\}), (a_2, \{(C,c,\{\{\}\},1.0)\}, \{\}, \{\}\rangle, \\ \qquad \langle \{a_1\}, \{(a_2, \{(C,c,\{\{\}\},1.0)\}, \{(B,b,\{\{b\}\},4.0)\}, \{\}\rangle, \\ \qquad \langle \{a_2\}, \{(a_1, \{(B,b,\{\{\}\},1.0)\}, \{\}, \{\}\rangle, \\ \qquad \langle \{a_1,a_2\}, \{\}, \{(B,b,\{\{b\}\},2.0)\}, \{\}\rangle\ \}\rangle, \\ \quad \langle B, \{\ \langle \{\}, \{(b, \{(A,a_1,\{\{\}, \{a_2\}\},1.0)\}, \{\}, \{\}\rangle, \\ \qquad \langle \{b\}, \{\}, \{(b, \{(A,a_1,\{\{a_1\}\},4.0), (A,a_1,\{\{a_1,a_2\}\},2.0)\}, \{\}\rangle\ \}\rangle, \\ \quad \langle C, \{\ \langle \{\}, \{(a_1, \{(A,a_2,\{\{\}, \{a_1\}\},1.0)\ \}, \{\}, \{\}\rangle, \\ \qquad \langle \{c\}, \{\}, \{\}, \{\}\rangle\ \}\rangle\} \end{array}$$

3.3 From Compile Maps to Stochastic π calculus

We construct a π calculus specification from the compile map $\mathscr{C}$ of a model $\mathscr{M}$. For each species $A \in \mathsf{species}(\mathscr{M})$, we map the process description $\mathsf{P}(A)$ to a process specification in stochastic π calculus. Let

$$\mathsf{P}(A) = \langle A, \{\, \mathsf{actions}(A,\mathscr{S}_1), \ldots, \mathsf{actions}(A,\mathscr{S}_n) \,\} \rangle$$

where $\wp(\mathsf{sites}(\mathscr{M},A)) = \{\mathscr{S}_1, \ldots, \mathscr{S}_n\}$, that is, the powerset of set of sites of A. Thus, there are n process specifications for the species A, some of which may be empty. Each process specification for each state $\mathscr{S}$ of A is of the following syntactic form.

process declaration	"= ("	*local channel declarations*
		association specifications
	"+"	*dissociation specifications*
	"+"	*transformation specifications* ")"

The idea here is that each set of sites of a species A denotes the state where the sites in the set are bound. Thus the powerset of the set of sites of a species denotes the set of all its states. Now, let us obtain the process expression for each state $\mathscr{S}_i$ with respect to $\mathsf{actions}(A,\mathscr{S}_i)$ where $1 \leq i \leq n$. Let us consider $\mathscr{S}_i = \{a_1, \ldots, a_k\}$ of A with

$$\mathsf{actions}(A,\mathscr{S}_i) = \langle \mathscr{S}_i, \mathsf{assoc}(A,\mathscr{S}_i), \mathsf{dissoc}(A,\mathscr{S}_i), \mathsf{transform}(A,\mathscr{S}_i) \rangle\,.$$

Process declaration

The expression for *process declaration* is a process name with its list of parameters. It is delivered by the dissociation sentences in $\mathscr{M}$ and $\mathscr{S}_i = \{a_1, \ldots, a_k\}$. For every $a_j \in \mathscr{S}_i$, consider the set

$$\begin{array}{l}
\mathscr{R}(A,a_j) = \\
\{(a_j,\,(r/2)) \mid \langle \mathit{dissociation}, (A,a_j), (B,b), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathscr{M}\,\} \cup \\
\quad \{(a_j,\,(r/2)) \mid \langle \mathit{dissociation}, (B,b), (A,a_j), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathscr{M}\,\} \cup \\
\qquad \{(a_j,\,1.0) \mid \langle \mathit{dissociation}, (B,b), (A,a_j), \mathit{Pos}, \mathit{Neg}, r\rangle \notin \mathscr{M} \wedge \\
\qquad\qquad \langle \mathit{dissociation}, (A,a_j), (B,b), \mathit{Pos}, \mathit{Neg}, r\rangle \notin \mathscr{M}\}\,.
\end{array}$$

We associate each element of the set $\mathscr{R}(A,a_j)$ a unique label $s \in \mathbb{N}^+$ and obtain $\mathscr{R}'(A,a_j)$. Then if $\mathscr{R}'(A,a_j) = \{(a_j, r_1, 1), \ldots, (a_j, r_\ell, \ell)\}$ we write the process declaration for A at state $\mathscr{S}_i = \{a_1, \ldots, a_k\}$ as follows.

$$A_i(a_1 1, \ldots, a_1 \ell_1, \ldots\ldots, a_k 1, \ldots, a_k \ell_k)$$

Example 5 *For the state $\mathscr{S}_2 = \{a_1\}$ of species A of Example 2, we have the process declaration below, since we have that $\mathscr{R}'(A,a_1) = \{(a_1, 2.0, 1), (a_1, 1.0, 2)\}$.*

$$A_2(a_1 1, a_1 2)$$

Local channel declarations

These expressions are delivered by the dissociation sentences in $\mathscr{M}$ and the $\mathsf{assoc}(A,\mathscr{S}_i)$. That is, for every

$$\mathsf{assoc}(A,\mathscr{S}_i,a_j) = \langle\, a_j,\, \mathsf{assocPartners}(A,\mathscr{S}_i,a_j)\,\rangle \in \mathsf{assoc}(A,\mathscr{S}_i)\,,$$

and for every $\langle B, b, \mathsf{states}(B), r\rangle \in \mathsf{assocPartners}(A, \mathscr{S}_i, a_j)$ consider the set

$$\begin{aligned}&\mathscr{U}(A, a_j, B, b) = \\ &\{(a_j, (r/2)) \mid \langle \mathit{dissociation}, (A, a_j), (B, b), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathscr{M} \wedge a_j \prec b\} \cup \\ &\quad \{(a_j, (r/2)) \mid \langle \mathit{dissociation}, (B, b), (A, a_j), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathscr{M} \wedge a_j \prec b\} \cup \\ &\quad\quad \{(a_j, 1.0) \mid \langle \mathit{dissociation}, (B, b), (A, a_j), \mathit{Pos}, \mathit{Neg}, r\rangle \notin \mathscr{M} \wedge \\ &\quad\quad\quad \langle \mathit{dissociation}, (A, a_j), (B, b), \mathit{Pos}, \mathit{Neg}, r\rangle \notin \mathscr{M} \wedge a_j \prec b\}\end{aligned}$$

where $\prec$ denotes a lexicographic order on sites. We associate each element of the set $\mathscr{U}(A, a_j, B, b)$ a unique label $s \in \mathbb{N}^+$ to obtain $\mathscr{U}'(A, a_j, B, b)$. Then if

$$\mathscr{U}'(A, a_j, B, b) = \{(a_j, r_1, 1), \ldots, (a_j, r_\ell, \ell)\}$$

then we write the channel declarations for $\mathsf{assoc}(A, \mathscr{S}_i, a_j)$ as follows.

$$\texttt{new } a_j 1 \texttt{@} r_1 \quad \ldots \quad \texttt{new } a_j \ell \texttt{@} r_\ell$$

Example 6 *For the state $\mathscr{S}_2 = \{a_1\}$ of species A of Example 2, we have the channel declarations below, since we have that $\mathscr{U}'(A, a_2, B, b) = \{(a_2, 1.0, 1)\}$.*

$$\texttt{new } a_2 1 \texttt{@} 1.0$$

Association specifications

The expression for *association specifications* for species A at state $\mathsf{assoc}(A, \mathscr{S}_i)$ is delivered by $\mathsf{assoc}(A, \mathscr{S}_i)$. For every

$$\langle a_j, \mathsf{assocPartners}(A, \mathscr{S}_i, a_j)\rangle \in \mathsf{assoc}(A, \mathscr{S}_i),$$

and for every $\langle B, b, \mathsf{states}(B), r\rangle \in \mathsf{assocPartners}(A, \mathscr{S}_i, a_j)$ consider the set

$$\begin{aligned}&\mathscr{B}(A, a_j, B, b) = \\ &\quad \{(!a_j b, r) \mid \langle (B, b), \mathsf{states}(B), r\rangle \in \mathsf{assocPartners}(A, \mathscr{S}_i, a_j) \wedge a_j \prec b\} \cup \\ &\quad\quad \{(?b a_j, r) \mid \langle (B, b), \mathsf{states}(B), r\rangle \in \mathsf{assocPartners}(A, \mathscr{S}_i, a_j) \wedge b \prec a_j\}.\end{aligned}$$

We associate each element of the set $\mathscr{B}(A, a_j, B, b)$ a unique label $s \in \mathbb{N}^+$ and obtain $\mathscr{B}'(A, a_j, B, b)$. Association of site a_j on A results in the state $\mathscr{S}_{i'} = \mathscr{S}_i \cup \{a_j\}$. For each element of $(!a_j b, r_s, s) \in \mathscr{B}'(A, a_j, B, b)$ we write the following, composed by +.

$$!a_j bs(a_j 1, \ldots, a_j \ell)\,; \mathit{continuation}$$

The association channel names, such as $a_j bs$ here, are also declared as *global channel declarations*, preceding all the process declarations. The *continuation* is written for A in $\mathscr{S}_{i'}$ as for *process declarations* above, however we write `nil` for the channel names for those associations of site a_j on A with some site $b' \neq b$. Here, `nil` is the nil-dissociation channel with rate 0. We obtain $a_j 1, \ldots, a_j \ell$ from the set $\mathscr{U}(A, a_j, B, b)$ as in *channel declarations*.

Example 7 *For the state $\mathscr{S}_2 = \{a_1\}$ of species A of Example 2, we have the following association specifications.*

$$!a2c1(a_2)\,; A3(a_1 1, a_1 2, a_2)$$

Dissociation specifications

The expression for *dissociation specifications* for species A at state $\mathsf{assoc}(A,\mathscr{S}_i)$ is delivered by $\mathsf{dissoc}(A,\mathscr{S}_i)$. For every

$$\langle a_j, \mathsf{dissocPartners}(A,\mathscr{S}_i,a_j)\rangle \in \mathsf{dissoc}(A,\mathscr{S}_i),$$

and for every $\langle B,b,\mathsf{states}(B),r\rangle \in \mathsf{dissocPartners}(A,\mathscr{S}_i,a_j)$ consider the set

$$\begin{aligned}\mathscr{G}(A,a_j,B,b) = \\ \{(!a_j,r) \mid \langle (B,b),\mathsf{states}(B),r\rangle \in \mathsf{dissocPartners}(A,\mathscr{S}_i,a_j) \wedge a_j \prec b\} \cup \\ \{(?b,r) \mid \langle (B,b),\mathsf{states}(B),r\rangle \in \mathsf{dissocPartners}(A,\mathscr{S}_i,a_j) \wedge b \prec a_j\}.\end{aligned}$$

We associate each element of the set $\mathscr{G}(A,a_j,B,b)$ a unique label $s \in \mathbb{N}^+$ and obtain $\mathscr{G}'(A,a_j,B,b)$. Dissociation of a_j on A results in the state $\mathscr{S}_{i'} = \mathscr{S}_i \setminus \{a_j\}$. For each $(!a_j,r_s,s) \in \mathscr{G}'(A,a_j,B,b)$ we write the following, composed by "+".

$$!a_j s\,;\mathit{continuation} \;\texttt{+}\; ?a_j s\,;\mathit{continuation}$$

The *continuation* is written for A in $\mathscr{S}_{i'}$ as for *process declarations* above.

Example 8 *For the state $\mathscr{S}_2 = \{a_1\}$ of species A of Example 2, we have the following dissociation specifications.*

$$!a_1 1\,;A_1() \;\texttt{+}\; ?a_1 1;A_1()$$

Transformation specifications

The expression for *transformation specifications* for species A is given only if the state $\mathscr{S} = \{\}$. In that case, for $\mathsf{transfrom}(A,\{\}) = \{(B_1,r_1),\ldots,(B_k,r_k)\}$ we write

$$\texttt{delay@}r_1\,;B_1() \;\texttt{+}\; \ldots \;\texttt{+}\; \texttt{delay@}r_k\,;B_k()$$

4 Syntax of the Language

The syntax of the language is defined in BNF notation, where optional elements are enclosed in braces as {Optional}. A model (description) consists of sentences of the following form.

Model ::= *Sentence*$_1$ ... *Sentence*$_m$ $m \geq 1$

Sentence ::= *Association*
| *Dissociation*
| *Transformation*
| *Decay*
| *Phosphorylation*
| *Dephosphorylation*

Association ::= *Site* `on` *Species* `associates` *Site* `on` *Species* {`with rate` *Float*} {`if` *Conditions*}

Dissociation ::= *Site* `on` *Species* `dissociates` *Site* `on` *Species* {`with rate` *Float*} {`if` *Conditions*}

Phosphorylation ::= *Site* `on` *Species* `gets phosphorylated` {`with rate` *Float*} {`if` *Conditions*}

Dephosphorylation ::= *Site* `on` *Species* `gets dephosphorylated` {`with rate` *Float*} {`if` *Conditions*}

Transformation ::= *Species* `becomes` *Species* {`with rate` *Float*}

Decay ::= *Species* `decays` {`with rate` *Float*}

Conditions ::= *Condition*
| *Condition* `and` *Conditions*

Condition ::= *Site* `on` *Species* `is bound`
| *Site* `on` *Species* `is unbound`

Site ::= *String*

Species ::= *String*

In our implementation of the translation algorithm, each sentence of a model given in this syntax is mapped by a lexer and parser to a data structure of the form given in Section 2 in the obvious way. Phosphorylation sentences are treated as association sentences where the second species is by default `Phosph` with the binding site `phosph`. The dephosphorylation sentences are mapped similarly to dissociation sentences. If not given, a default rate (1.0) is assigned to sentences.

4.1 A Model of Fcγ Receptor-mediated Phagocytosis

We demonstrate the use of the language on a model of Fcγ receptor (FcγR) phosphorylation during phagocytosis, where the binding of complexed immunoglobulins G (IgG) to FcγR triggers a signalling cascade that leads to actin-driven particle engulfment [8, 18, 4]. When a small particle is coated (opsonised) with IgG, the Fc regions of the IgG molecules can bind to FcγRs in the plasma membrane and initiate a phagocytic response: a signalling cascade then drives the remodelling of the actin cytoskeleton close to the membrane. This results in cup-shaped folds of plasma membrane that extend outwards around the internalised particle and eventually close into a plasma membrane-derived phagosome.

FcγR contains within its cytoplasmic tail an immunoreceptor tyrosine-based activation motif (ITAM). The association of FcγR with an IgG induces the phosphorylation of two tyrosine residues within the ITAM domain by Src-family kinases. The phosphorylated ITAM domain then recruits Syk kinase, which propagates the signal further to downstream effectors (see Figure 1). In our language, we can describe

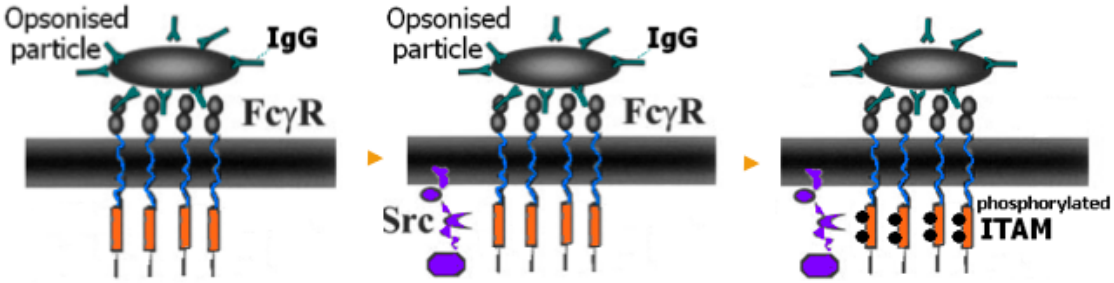

Figure 1: A simple model of the phosphorylation of the ITAM domain on the Fcγ receptor during phagocytosis. Adapted from [8].

the initial phases of this cascade as follows:

```
site f on FcR associates site i on IgG with rate 2.0
site y on FcR gets phosphorylated if site f on FcR is bound
site z on FcR gets phosphorylated if site f on FcR is bound
```

The first sentence above describes the binding of FcγR to IgG. The second and third sentences describe the phosphorylation of the two tyrosine residues on ITAM (association of a `Phosph0()` molecule). This is automatically translated by our tool into the SPiM program given in Appendix A. We can then run stochastic simulations on the model given by these sentences.

By using this language and our translation tool, we can build models of different size and complexity, and modify and extend these models with respect to the knowledge in hand on the different sites and interaction capabilities of the FcγR, as well as other biological systems. For example, the model above abstracts away from the role played by the Src kinases in the phosphorylation of the FcγR as depicted in Figure 1. The sentences above can be easily modified and extended to capture this aspect in the model as follows: here, the shaded part demonstrates the modifications with respect to the model given above.

```
site f on FcR associates site i on IgG with rate 2.0
site y on FcR gets phosphorylated if site s on FcR is bound
site z on FcR gets phosphorylated if site s on FcR is bound
site s on FcR associates site sr on Src if site f on FcR is bound
site s on FcR dissociates site sr on Src
```

The SPiM program resulting from automated translation of this model is given in Appendix B. It is important to note that, because FcR has 4 binding sites in the model above, in the SPiM code resulting from the translation, there are 16 species for FcR, denoting its different possible states. However, in the code given in Appendix A, there are 8 species for FcR denoting its states that result from its 3 binding sites in that model.

5 Discussion

We have introduced a natural language interface for building stochastic π calculus models of biological systems. The κ-calculus [5, 6, 7] and the work on Beta-binders in [11] have been source of inspiration

for this language.

In [11], Guerriero et al. give a narrative style interface for the process algebra Beta-binders for a rich biological language. In our language, we build complex events such as phosphorylation and dephosphorylation of sites as instances of basic primitives of association, dissociation and transformation. We give a functional translation algorithm for our translation into stochastic π calculus. The conditions that we impose on the models are automatically verified in the implementation of our tool. These conditions should be instrumental for 'debugging' purposes while building increasingly large models.

The implicit semantics of our language, which is implemented in the translation algorithm into π calculus, can be seen as a translation of a fragment of the κ calculus into π calculus. Another approach similar to the one in this paper is the work by Laneve et al. in [15], where the authors give an encoding of nano-κ-calculus in SPiM. In comparison with our algorithm, the encoding in [15] covers a larger part of nano-κ by using the SPiM language as a programming language for implementing a notion of term rewriting, where there is an explicit function for matching. The algorithm gives the different states of a species in the SPiM encoding with respect to the parameters of that species as in κ-calculus.

Topics of future work include an exploration of the expressive power of the association, dissociation and transformation primitives with respect to Kohn diagram representation [10] of biological models.

References

[1] L. Cardelli, E. Caron, P. Gardner, O. Kahramanoğulları, and A. Phillips. A process model of actin polymerisation. In *FBTC'08*, volume 229 of *ENTCS*, pages 127–144. Elsevier, 2008.

[2] L. Cardelli, E. Caron, P. Gardner, O. Kahramanoğulları, and A. Phillips. A process model of Rho GTP-binding proteins. *Theoretical Computer Science*, 410/33-34:3166–3185, 2009.

[3] L. Cardelli and G. Zavattaro. On the computational power of biochemistry. In *AB'08*, volume 5147 of *LNCS*, pages 65–80. Springer, 2008.

[4] C. Cougoule, S. Hoshino, A. Dart, J. Lim, and E. Caron. Dissociation of recruitment and activation of the small G-protein Rac during Fc gamma receptor-mediated phagocytosis. *J. Bio. Chem.*, 281:8756–8764, 2006.

[5] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling of cellular signalling. In *CONCUR'07*, volume 4703 of *LNCS*, pages 17–41. Springer, 2007.

[6] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling, symmetries, refinements. In *FMSB'08*, volume 5054 of *LNCS*, pages 103–122. Springer, 2008.

[7] V. Danos, J. Feret, W. Fontana, and J. Krivine. Abstract interpretation of cellular signalling networks. In *VMCAI'08*, volume 4905 of *LNBI*, pages 83–97. Springer, 2008.

[8] E. Garcia-Garcia and C. Rosales. Signal transduction during Fc receptor-mediated phagocytosis. *Journal of Leukocyte Biology*, 72:1092–1108, 2002.

[9] M. Heiner, D. Gilbert, and R. Donaldson. Petri nets for systems and synthetic biology. In *SFM'08*, volume 5016 of *LNCS*, pages 215–264. Springer, 2008.

[10] K. W. Kohn, M. I. Aladjem, S. Kim, J. N. Weinstein, and Y. Pommier. Depicting combinatorial complexity with the molecular interaction map notation. *Molecular Systems Biology*, 2:51, 2006.

[11] C. Priami M. L. Guerriero, J. K. Heath. An automated translation from a narrative language for biological modelling into process algebra. In *CMSB'07*, volume 4695 of *LNCS*, pages 136–151. Springer, 2007.

[12] R. Milner. *Communication and Mobile Systems: the π-calculus*. Cambridge University Press, 1999.

[13] A. Phillips and L. Cardelli. Efficient, correct simulation of biological processes in stochastic Pi-calculus. In *CMSB'07*, volume 4695 of *LNBI*. Springer, 2007.

[14] A. Phillips, L. Cardelli, and G. Castagna. A graphical representation for biological processes in the stochastic pi-calculus. In *Transactions on Computational Systems Biology VII*, volume 4230 of *LNCS*, pages 123–152. Springer, 2006.

[15] S. Pradalier, C. Laneve, and G. Zavattaro. From biochemistry to stochastic processes. In *QALP'09*, ENTCS. Elsevier, 2009. to appear.

[16] C. Priami, A. Regev, E. Shapiro, and W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80:25–31, 2001.

[17] A. Regev and E. Shapiro. Cellular abstractions: Cells as computation. *Nature*, 419:343, 2002.

[18] J. A. Swanson and A. D. Hoppe. The coordination of signaling during Fc receptor-mediated phagocytosis. *Journal of Leukocyte Biology*, 76:1093–1103, 2004.

[19] A. Tiwari, C. Talcott, M. Knapp, P. Lincoln, and K. Laderoute. Analyzing pathways using sat-based approaches. In Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen, editors, *Second International Conference, Algebraic Biology 2007*, volume 4545 of *LNCS*, pages 155–169. Springer, 2007.

Appendix A

```
site f on FcR associates site i on IgG with rate 2.0
site y on FcR gets phosphorylated if site f on FcR is bound
site z on FcR gets phosphorylated if site f on FcR is bound
```

The SPiM code resulting from the automated translation of this model.

```
directive sample 10.0
directive plot FcR7(); FcR6();
        FcR5();  FcR4(); FcR3();
        FcR2(); FcR1();
        FcR0(); IgG1(); IgG0();
        Phosph1(); Phosph0()

new fi1@1.0:chan(chan)
new phosphy2@1.0:chan(chan)
new phosphz3@1.0:chan(chan)
new nil@0.0:chan

let FcR0() =
     ( new f@1.0:chan
       !fi1(f)*2.0; FcR1(f) )


and FcR1(f:chan) =
     ( do ?phosphy2(y); FcR4(f,y)
       or ?phosphz3(z); FcR5(f,z) )

and FcR2(y:chan) =
     ( new f@1.0:chan
       !fi1(f)*2.0; FcR4(f,y) )

and FcR3(z:chan) =
     ( new f@1.0:chan
       !fi1(f)*2.0; FcR5(f,z) )

and FcR4(f:chan,y:chan) =
     ( ?phosphz3(z); FcR7(f,y,z) )

and FcR5(f:chan,z:chan) =
     ( ?phosphy2(y); FcR7(f,y,z) )

and FcR6(y:chan,z:chan) =
     ( new f@1.0:chan
       !fi1(f)*2.0; FcR7(f,y,z) )

and FcR7(f:chan,y:chan,z:chan) =
     ()

let IgG0() =
     ( ?fi1(i); IgG1(i) )

and IgG1(i:chan) =
     ()

let Phosph0() =
     ( new phosph@1.0:chan
       do !phosphy2(phosph)*1.0;
            Phosph1(phosph)
       or !phosphz3(phosph)*1.0;
            Phosph1(phosph) )

and Phosph1(phosph:chan) =
     ()


run 1000 of  FcR0()
run 1000 of  IgG0()
run 1000 of  Phosph0()
```

6 Appendix B

```
site f on FcR associates site i on IgG with rate 2.0
site y on FcR gets phosphorylated if site s on FcR is bound
site z on FcR gets phosphorylated if site s on FcR is bound
site s on FcR associates site sr on Src if site f on FcR is bound
site s on FcR dissociates site sr on Src
```

The SPiM code resulting from the automated translation of this model.

```
directive sample 10.0
directive plot FcR15();
                                        FcR14(); FcR13(); FcR12();
                                        FcR11(); FcR10();
                                        FcR9(); FcR8(); FcR7();
```

```
      FcR6(); FcR5();
      FcR4(); FcR3(); FcR2();
      FcR1(); FcR0();
      IgG1(); IgG0();
      Phosph1(); Phosph0();
      Src1(); Src0()

new fi1@1.0:chan(chan)
new phosphx2@1.0:chan(chan)
new phosphy3@1.0:chan(chan)
new ssr4@1.0:chan(chan)
new nil@0.0:chan


let FcR0() =
 ( new f@1.0:chan
    !fi1(f)*2.0; FcR1(f) )

and FcR1(f:chan) =
 ( new s1@0.50:chan
   !ssr4(s1)*1.0; FcR5(f,s1) )

and FcR2(s1:chan) =
 ( new f@1.0:chan
   do !fi1(f)*2.0; FcR5(f,s1)
   or ?phosphx2(x); FcR8(s1,x)
   or ?phosphy3(y); FcR9(s1,y)
   or !s1; FcR0() or ?s1; FcR0() )

and FcR3(x:chan) =
 ( new f@1.0:chan
   !fi1(f)*2.0; FcR6(f,x) )

and FcR4(y:chan) =
 ( new f@1.0:chan
   !fi1(f)*2.0; FcR7(f,y) )

and FcR5(f:chan,s1:chan) =
 ( do ?phosphx2(x); FcR11(f,s1,x)
   or ?phosphy3(y); FcR12(f,s1,y)
   or !s1; FcR1(f) or ?s1; FcR1(f) )

and FcR6(f:chan,x:chan) =
 ( new s1@0.50:chan
   !ssr4(s1)*1.0; FcR11(f,s1,x) )

and FcR7(f:chan,y:chan) =
 ( new s1@0.50:chan
   !ssr4(s1)*1.0; FcR12(f,s1,y) )

and FcR8(s1:chan,x:chan) =
 ( new f@1.0:chan
   do !fi1(f)*2.0; FcR11(f,s1,x)
   or ?phosphy3(y); FcR14(s1,x,y)
   or !s1; FcR3(x) or ?s1; FcR3(x) )

and FcR9(s1:chan,y:chan) =
 ( new f@1.0:chan
   do !fi1(f)*2.0; FcR12(f,s1,y)
   or ?phosphx2(x); FcR14(s1,x,y)
   or !s1; FcR4(y) or ?s1; FcR4(y) )

and FcR10(x:chan,y:chan) =
 ( new f@1.0:chan
   !fi1(f)*2.0; FcR13(f,x,y) )

and FcR11(f:chan,s1:chan,x:chan) =
 ( do ?phosphy3(y); FcR15(f,s1,x,y)
   or !s1; FcR6(f,x) or ?s1; FcR6(f,x) )

and FcR12(f:chan,s1:chan,y:chan) =
 ( do ?phosphx2(x); FcR15(f,s1,x,y)
   or !s1; FcR7(f,y) or ?s1; FcR7(f,y) )

and FcR13(f:chan,x:chan,y:chan) =
 ( new s1@0.50:chan
   !ssr4(s1)*1.0; FcR15(f,s1,x,y) )

and FcR14(s1:chan,x:chan,y:chan) =
 ( new f@1.0:chan
   do !fi1(f)*2.0; FcR15(f,s1,x,y)
   or !s1; FcR10(x,y) or ?s1; FcR10(x,y) )

and FcR15(f:chan,s1:chan,x:chan,y:chan) =
 ( do !s1; FcR13(f,x,y) or ?s1; FcR13(f,x,y) )


 let IgG0() =
   ( ?fi1(i); IgG1(i) )

 and IgG1(i:chan) =
   ()


 let Phosph0() =
   ( new phosph@1.0:chan
     do !phosphx2(phosph)*1.0;
     Phosph1(phosph)
     or !phosphy3(phosph)*1.0;
     Phosph1(phosph) )

 and Phosph1(phosph:chan) =
   ()


 let Src0() =
   ( ?ssr4(sr1); Src1(sr1) )

 and Src1(sr1:chan) =
   ( do !sr1; Src0() or ?sr1; Src0() )


 (* run  1000 of ... *)
```