Int J Software Informatics, Vol., No., , pp. 1–??
International Journal of Software and Informatics, ISSN 1673-7288

E-mail: ijsi@iscas.ac.cn
http://www.ijsi.org
Tel: +86-10-62661040

An Intuitive Modelling Interface for Systems Biology

Ozan Kahramanoğulları[1] and Luca Cardelli [2]

[1]The Microsoft Research – University of Trento Centre for Computational and Systems Biology

[2]Microsoft Research Cambridge

Abstract 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 give a formal semantics for this language and a translation algorithm into stochastic π calculus that delivers this semantics. 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, which can be used for simulation and analysis. [1] [2]

Key words: systems biology; stochastic π calculus; SPiM; modelling

O. Kahramanoğulları and L. Cardelli. An Intuitive Modelling Interface for Systems Biology. *Int J Software Informatics*, , (): 1–**??**.

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 [28, 29], 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 algebras [24, 5, 4, 27], term rewriting (see, e.g, [12, 9, 11, 2, 22, 1]) and Petri nets (see, e.g., [16, 32, 18]) 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

† This work is sponsored by UK Biotechnology and Biological Sciences Research Council through the Centre for Integrative Systems Biology at Imperial College (grant BB/C519670/1).
‡ Corresponding author: O. Kahramanoğulları and L. Cardelli
♯ Manuscript received 2009-03-08; revised 2009-10-10; accepted 2009-10-10; published online 2009-12-12

[1] A preliminary version of this paper, co-authored by Dr. Emmanuelle Caron, has been presented at the DCM'09 Workshop. We dedicate this paper to the memory of Emmanuelle, 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.

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.

Although languages based on Petri nets provide a representation scheme that is akin to chemical reactions, they enjoy a limited expressive power with respect to the biological phenomena that they capture [6]. Writing programs in more expressive 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 need to be represented in the model. Enumerating all these states by hand, without inserting typos, is a difficult task.

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. 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., [4, 5]).

Our focus here is on the stochastic π calculus [21, 26], which is a broadly studied process algebra because of its compactness, generality, and flexibility. Since biological systems are typically highly complex and massively parallel, 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 [6].

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 formal semantics for the language and a translation algorithm into stochastic π calculus that delivers this semantics. Based on this, we present the implementation of a tool for automated translation of models into Microsoft Research's stochastic simulation language SPiM [24, 23], 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. We then provide a discussion of the expressive power of the language. 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 adopt the abstraction of biochemical species as stateful entities with connectivity interfaces [10, 20]. A species can have a number of sites in its interface through which it interacts with other species, and may change its state as a result of the interactions. In Section 3, we use this idea to design a natural language-like syntax for building

[3] `https://sites.google.com/site/ozankahramanogullari/software/pim`

models. The models written in this language can be automatically translated into a SPiM program by using our tool, which implements the translation algorithm given in Section 4: with this algorithm, we map the sentences of the language into events constructed from 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 finite set of sites, denoted with $a, b, c, \ldots$ with which it can bind to other species or unbind from other species when they are already bound. ϵ denotes the 'dummy-site', which is a place holder. We write sentences that describe the 'behaviour' of each species with respect to their sites. There are three kinds of sentences: *associations*, *dissociations*, and *transformations*, defined as

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

where $\mathsf{type} \in \{\mathit{association},\ \mathit{dissociation},\ \mathit{transformation}\}$ 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 if the sites on species in *Pos* are already bound and those in *Neg* are already unbound. A *transformation* sentence describes the event of species A transforming into species B, where B can be empty, in which case it describes the decay of species A. $r \in \mathbb{R}^+$ denotes the rate of the event that the sentence describes. A model $\mathcal{M}$ is a set of such sentences. In Section 3, we give a representation of these sentences in natural-language. For example, a sentence of the form $\langle\, \mathit{association}, (A, a),\ (B, b),\ \{(A, c)\},\ \{\,(A, a),\ (B, b)\,\},\ 1.0 \,\rangle$ 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
                                    and site a on A is unbound
                                    and site b on B is unbound
```

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

2.1 Conditions on Sentences

Given a model $\mathcal{M}$, 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 $\langle\, \mathsf{type}, (A, a),\ (B, b),\ \mathit{Pos},\ \mathit{Neg},\ r \,\rangle$, we have that $\mathit{Pos} \cap \mathit{Neg} = \emptyset$.

3. **All the sites in the conditions are declared in the model.** For every sentence of the form $\langle\, \mathsf{type}, (A, a), (B, b), \mathit{Pos}, \mathit{Neg}, r \,\rangle$, we have that $\mathsf{sites}(\mathit{Pos}, A) \subseteq \mathsf{sites}(\mathcal{M}, A)$, $\mathsf{sites}(\mathit{Neg}, A) \subseteq \mathsf{sites}(\mathcal{M}, A)$, $\mathsf{sites}(\mathit{Pos}, B) \subseteq \mathsf{sites}(\mathcal{M}, B)$ and $\mathsf{sites}(\mathit{Neg}, B) \subseteq \mathsf{sites}(\mathcal{M}, B)$.
4. **Association sentences associate unbound species.** For every association sentence $\langle\, \mathit{association}, (A, a), (B, b), \mathit{Pos}, \mathit{Neg}, r \,\rangle$, we have that $(A, a), (B, b) \in \mathit{Neg}$.
5. **Dissociation sentences dissociate bound species.** For every dissociation sentence $\langle\, \mathit{dissociation}, (A, a), (B, b), \mathit{Pos}, \mathit{Neg}, r \,\rangle$, we have that $(A, a), (B, b) \in \mathit{Pos}$.
6. **Transformation sentences are unbound at all sites.** For every transformation sentence $\langle\, \mathit{transformation}, (A, \epsilon), (B, \epsilon), \mathit{Pos}, \mathit{Neg}, r \,\rangle$, we have that $\mathit{Pos} = \emptyset$ and $\mathit{Neg} = \{(A, x) \mid x \in \mathsf{sites}(\mathcal{M}, A)\}$.

Condition 3 allows only those sites to appear in the conditions of the model that are relevant to the association and dissociations in the body of the sentences. Although this condition could be lifted without hampering the correctness of the models, it is useful for avoiding redundancies.

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 $\mathcal{M}$, 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 $\mathsf{sites}(\mathcal{M}, A) = \{a_1, a_2\}$, the set $\wp(\mathsf{sites}(\mathcal{M}, A)) = \{\{\}, \{a_1\}, \{a_2\}, \{a_1, a_2\}\}$ is the set of all its states. Then $\{a_1\}$ is the state where site a_1 on A is bound and site a_2 on A is unbound.

We map each sentence $\langle\, \mathsf{type}, (A, a), (B, b), \mathit{Pos}, \mathit{Neg}, r \,\rangle$ 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.

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

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 $\mathcal{M}$ of the form $\langle\, \mathsf{type}_1, (A, a), (B, b), \mathit{Pos}_1, \mathit{Neg}_1, r \,\rangle$ and $\langle\, \mathsf{type}_2, (A, a), (B, b), \mathit{Pos}_2, \mathit{Neg}_2, r \,\rangle$ 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 $\mathsf{states}(B)_1 \cap \mathsf{states}(B)_2 = \emptyset$.

This condition does not only make the translation that we present below easier, but also helps to prevent ambiguities in the models by not allowing the definition of two different interaction rates for any two model species for their same states.

Example 1. Consider the model $\mathcal{M}_1$.

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

This model does not fulfill any of the conditions above: in the first sentence, (1.) $C \notin \{A,B\}$; (2.) $(B,f) \in \mathit{Pos}$ and $(B,f) \in \mathit{Neg}$; (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 $\mathcal{M}_2$ fulfills all the conditions above.

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

3 The Narrative Language

We are now ready to define a natural-language-like narrative language for describing molecular events that are typically modelled in systems biology. For this purpose, we resort to the data structures given above. Let us first define the syntax of the language.

3.1 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 a 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. In our implementation of the translation, we allow the association and dissociation sites not to be included in the conditions. We thus include these sites in the conditions automatically during the compilation if all the other conditions are met.

3.2 Semantics of the Language

A narrative given in the syntax defined above describes the dynamics of the system that it models: a narrative given in this syntax can be translated into a stochastic π calculus model by mapping each narrative sentence to the data structures of Section 2,

and then by applying the algorithm given in Section 4. Then the reduction semantics of the stochastic π calculus can be applied to the model (see Section 4).

We give a reduction semantics directly on the narrative sentences, which corresponds to the reduction semantics of the stochastic π calculus. For this purpose, we define a *solution*, denoted with $\mathcal{Z}$, as a multiset[4] of species. For each species in the solution, we give a representation of its state with respect to its bound binding sites as in Section 2: for a species $A \in \mathsf{species}(\mathcal{M})$ of a model $\mathcal{M}$, consider the set $\mathsf{sites}(\mathcal{M}, A)$ of all the sites of A in $\mathcal{M}$. Every instance of a species A in the solution is equipped with a subset of the set $\mathsf{sites}(\mathcal{M}, A)$, which denotes the state of A where these sites are bound. Moreover, we borrow from the κ calculus [9, 11] the notation of bonds as superscripts: we decorate each site with a natural number as a superscript to denote an explicit bond. This natural number appears strictly twice in the solution, once as the superscript of the site of A and once as the superscript of another site of a species with which A is bound.

Example 3. Consider the model $\mathcal{M}_2$ of Example 2, and the solution $\mathcal{Z}$ below for this model.

$$\mathcal{Z} = \dot{\{}\, A\{a_1^1, a_2^2\}, B\{b^1\}, C\{c^2\}, A\{a_1^3\}, B\{b^3\}, A\{a_2^4\}, C\{c^4\}, A\{\}, A\{\}, B\{\}, C\{\} \,\dot{\}}$$

In solution $\mathcal{Z}$, there is an instance of the species A that has bonds with instances of the species B and C; there is an instance of A that has a bond with an instance of B; and an instance of A that has a bond with an instance of C. There are two unbound instances of A, an unbound instance of B, and an unbound instance of C.

We are now ready to define the reduction semantics of the narrative language.

Definition 4. Consider a model $\mathcal{M}$ that fulfils the conditions given in Subsection 2.1. Let A, B be species in $\mathcal{M}$; and X, Y be sets of sites such that $X \subseteq \mathsf{sites}(\mathcal{M}, A)$ and $Y \subseteq \mathsf{sites}(\mathcal{M}, B)$. We define the reduction in the narrative language as follows.

$$\textsf{Association}: \quad \mathcal{M} \Rightarrow \dot{\{}\, A(X), B(Y) \,\dot{\}} \,\dot{\cup}\, \mathcal{Z} \xrightarrow{r}_{\mathsf{pim}} \dot{\{}\, A(\{a^k\} \cup X), B(\{b^k\} \cup Y) \,\dot{\}} \,\dot{\cup}\, \mathcal{Z}$$

if and only if there is a sentence in $\mathcal{M}$ of the form

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

such that $\{x \,|\, (x, A) \in \mathit{Pos}\} \subseteq X$, $X \cap \{x \,|\, (x, A) \in \mathit{Neg}\} = \emptyset$, $\{y \,|\, (y, B) \in \mathit{Pos}\} \subseteq Y$, $Y \cap \{y \,|\, (y, B) \in \mathit{Neg}\} = \emptyset$, $a \notin X$, $b \notin Y$, and $k \in \mathbb{N}^+$ does not appear anywhere in $\mathcal{Z}$.

$$\textsf{Dissociation}: \quad \mathcal{M} \Rightarrow \dot{\{}\, A(\{a^k\} \cup X), B(\{b^k\} \cup Y) \,\dot{\}} \,\dot{\cup}\, \mathcal{Z} \xrightarrow{r}_{\mathsf{pim}} \dot{\{}\, A(X), B(Y) \,\dot{\}} \,\dot{\cup}\, \mathcal{Z}$$

if and only if there is a sentence in $\mathcal{M}$ of the form

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

such that $\{x \,|\, (x, A) \in \mathit{Pos}\} \subseteq X \cup \{a\}$, $X \cap \{x \,|\, (x, A) \in \mathit{Neg}\} = \emptyset$, $\{y \,|\, (y, B) \in$

[4] Multisets are denoted by the curly brackets "$\dot{\{}\ \ \dot{\}}$". $\dot{\cup}$, $\dot{-}$ and $\dot{\subseteq}$ denote the multiset operations corresponding to the usual set operations $\cup$, $-$ and $\subseteq$, respectively.

$Pos\} \subseteq Y \cup \{b\}$, $Y \cap \{y \,|\, (y, B) \in Neg\} = \emptyset$, $a \notin X$, $b \notin Y$, and $k \in \mathbb{N}^+$ does not appear anywhere in $\mathcal{Z}$.

$$\mathsf{Transformation}: \quad \mathcal{M} \Rightarrow \dot{\{}\, A\{\}\, \dot{\}} \,\dot{\cup}\, \mathcal{Z} \xrightarrow{r}_{\mathsf{pim}} \dot{\{}\, B\{\}\, \dot{\}} \,\dot{\cup}\, \mathcal{Z}$$

if and only if there is a sentence in $\mathcal{M}$ of the form

$$\langle\, \mathit{transformation}, A, B, \mathit{Pos}, \mathit{Neg}, r \,\rangle\,.$$

4 Translation into Stochastic π calculus

In this section, we give an algorithm for translating models written in the narrative language into processes of the stochastic π calculus. Here we use a version of the stochastic π calculus, where each action can be associated with a stochastic weight [5]. The availability of this extension allows us to regulate the creation of channels and improves the modularity in our translation. For the representation of the states of species in the stochastic π calculus specifications, we use sets of the sites of each species.

The translation algorithm maps each model to an intermediate data structure that we call *compile map*, which is then translated into a π calculus specification. Let us first recall some of the definitions of the stochastic π calculus, implemented in SPiM [23]. Here we adapt the SPiM syntax as in [5].

4.1 Stochastic π calculus

In stochastic π calculus, the basic building blocks are processes which are defined as follows.

Definition 5. [5] Syntax of the stochastic π calculus: *processes* range over $P, Q, \ldots$ Below $\mathrm{fn}(P)$ denotes the set of names that are free in P.

`P,Q` ::=	`M`	Choice	`M` ::=	`()`	Null
|	`X(n)`	Instance	|	π`; P`	Action
|	`P | Q`	Parallel	|	`do` π`1;P1 or...or` π`N;PN`	Actions
|	`new x P`	Restriction			

`E` ::=	`{}`	Empty	π ::=	`?x(m)*r`	Input
|	`E,X(m) = P`	Definition, $\mathrm{fn}(P) \subseteq m$	|	`!x(n)*r`	Output
			|	`delay@r`	Delay

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

	`P | ()`	$\equiv$	`P`
	`P | Q`	$\equiv$	`Q | P`
	`P | (Q | R)`	$\equiv$	`(P | Q) | R`
`X(m) = P`	`X(n)`	$\equiv$	`P{m:=n}`
	`new x ()`	$\equiv$	`()`
	`new x new y P`	$\equiv$	`new y new x P`
`x`$\notin$`fn(P)`	`new x (P | Q)`	$\equiv$	`P | new x Q`

The reduction rules of the calculus are given below. Each rule is labelled with a corresponding rate that denotes the rate of a single reaction, which can be either a communication or a delay. The rules are standard except for the communication rule (2), where the rate of the communication is given by the weights of the input and output actions.

Definition 6. [5] Reduction in the stochastic π calculus.

(1)		`do delay@r; P or ...`	$\xrightarrow{r}$	`P`
(2)		`(do !x(n)*r1; P1 or...)`		
		`| (do ?x(m)*r2; P2 or...)`	$\xrightarrow{\rho(x)\cdot r1\cdot r2}$	`P1 | P2{m:=n}`
(3)	`P`$\xrightarrow{r}$`P'`	`new x P`	$\xrightarrow{r}$	`new x P'`
(4)	`P`$\xrightarrow{r}$`P'`	`P | Q`	$\xrightarrow{r}$	`P' | Q`
(5)	`Q`$\equiv$`P`$\xrightarrow{r}$`P'`$\equiv$`Q'`	`Q`	$\xrightarrow{r}$	`Q'`

A process can send a value n on channel x with weight r_1 and then do P_1, written `!x(n)*r1;P1`, or it can receive a value m on channel x with weight r_2 and then do P_2, written `?x(m)*r2;P2`. With respect to the reduction semantics above, if these complementary send and receive actions are running in parallel, they can synchronise on the common channel x and evolve to `P1 | P2{m:=n}`, where m is replaced by n in process P_2. This allows messages to be exchanged from one process to another. Each channel name x is associated with an underlying rate given by $\rho(x)$. The resulting rate of the interaction is given by $\rho(x)$ times the weights r_1 and r_2. These weights decouple the ability of two processes to interact on a given channel x from the rate of the interaction, which can change over time depending on the evolution of the processes. If no weight is given then a default weight of 1 is used.

4.2 Compile Maps

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

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

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

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

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

$$\{\langle B, b, \mathsf{states}(B), r\rangle \mid (\langle \mathit{association}, (A, a), (B, b), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathcal{M} \wedge \mathcal{S} \in \mathsf{states}(A)) \vee (\langle \mathit{association}, (B, b), (A, a), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathcal{M} \wedge \mathcal{S} \in \mathsf{states}(A))\}.$$

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

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

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

$$\{\langle B, b, \mathsf{states}(B), r\rangle \mid (\langle \mathit{dissociation}, (A,a), (B,b), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathcal{M} \wedge \mathcal{S} \in \mathsf{states}(A)) \vee (\langle \mathit{dissociation}, (B,b), (A,a), \mathit{Pos}, \mathit{Neg}, r\rangle \in \mathcal{M} \wedge \mathcal{S} \in \mathsf{states}(A))\}.$$

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

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

Otherwise, it is $\emptyset$.

Example 7. Consider the model $\mathcal{M}_2$ in Example 2. We have that the compile map $\mathcal{C}_2$ for this model, which is as follows.

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

4.3 From Compile Maps to Stochastic π calculus

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

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

where $\wp(\mathsf{sites}(\mathcal{M}, A)) = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$, 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 $\mathcal{S}$ of A is of the following syntactic form.

process declaration	"= ("	*local channel declarations*
	"do"	*association specifications*
	"or"	*dissociation specifications*
	"or"	*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 $\mathcal{S}_i$ with respect to $\mathsf{actions}(A, \mathcal{S}_i)$ where $1 \leq i \leq n$. Let us consider $\mathcal{S}_i = \{a_1, \ldots, a_k\}$ of A with

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

Process declaration

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

$a_j \in \mathcal{S}_i$, consider the set

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

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

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

Example 8. For the state $\mathcal{S}_2 = \{a_1\}$ of species A of Example 2, we have the process declaration below, since we have that $\mathcal{R}'(A, a_1) = \{(a_1, 2.0, 1), (a_1, 1.0, 2)\}$.

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

Local channel declarations

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

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

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

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

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

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

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

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

Example 9. For the state $\mathcal{S}_2 = \{a_1\}$ of species A of Example 2, we have the channel declarations below, since we have that $\mathcal{U}'(A, a_2, B, b) = \{(a_2, 1.0, 1)\}$.

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

Association specifications

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

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

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

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

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

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

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 $\mathcal{S}_{i'}$ 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 $\mathcal{U}(A, a_j, B, b)$ as in *channel declarations*.

Example 10. For the state $\mathcal{S}_2 = \{a_1\}$ 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 $\mathsf{assoc}(A, \mathcal{S}_i)$ is delivered by $\mathsf{dissoc}(A, \mathcal{S}_i)$. For every

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

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

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

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

$$!\,a_j s\,;\, \textit{continuation} \quad \texttt{or} \quad ?a_j s\,;\, \textit{continuation}$$

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

Example 11. For the state $\mathcal{S}_2 = \{a_1\}$ of species A of Example 2, we have the following dissociation specifications.

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

Transformation specifications

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

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

4.4 Translating Solutions

A solution consisting of a multiset of species is translated as the parallel composition of the processes, which are given by the translation of the instances of the species in that solution. This expression is preceded with the declaration of the private names, which denote the bonds between species, and are obtained with respect to the superscripts of the species' sites in the solution. For every superscript k that connects the site a of species A with the site b of species B, let us consider the set

$$\begin{aligned}
&\mathcal{Q}(k, A, a, B, b) = \\
&\{(\mathtt{e}k,\ (r/2)) \mid \langle \mathit{dissociation}, (A, a), (B, b),\ \mathit{Pos},\ \mathit{Neg},\ r\rangle \in \mathcal{M}\,\} \cup \\
&\quad \{(\mathtt{e}k,\ (r/2)) \mid \langle \mathit{dissociation}, (B, b), (A, a),\ \mathit{Pos},\ \mathit{Neg},\ r\rangle \in \mathcal{M}\,\} \cup \\
&\qquad \{(\mathtt{e}k,\ 1.0) \mid \langle \mathit{dissociation}, (B, b), (A, a),\ \mathit{Pos},\ \mathit{Neg},\ r\rangle \notin \mathcal{M} \wedge \\
&\qquad\qquad \langle \mathit{dissociation}, (A, a), (B, b),\ \mathit{Pos},\ \mathit{Neg},\ r\rangle \notin \mathcal{M}\}\,.
\end{aligned}$$

We associate each element of the set $\mathcal{Q}(k, A, a, B, b)$ a unique label $s \in \mathbb{N}^+$ and obtain $\mathcal{Q}'(k, A, a, B, b)$. Then if $\mathcal{Q}'(k, A, a, B, b) = \{(\mathtt{e}k,\ r_1,\ 1), \ldots, (\mathtt{e}k,\ r_\ell,\ \ell)\}$ then we write the following expression.

$$\texttt{new } ek\_1\texttt{@}r_1 \quad \ldots \quad \texttt{new } ek\_1\ell\texttt{@}r_\ell$$

Then, for each species A with the state $\mathcal{S}_i = \{a_1, \ldots, a_n\}$in the solution, we write

$$A_i(ek_1\_1, \ldots, ek_1\_\ell_1, \ldots\ldots, ek_n\_1, \ldots, ek_n\_\ell_n)$$

where each $ek_j\_j, \ldots, ek_j\_\ell_j$ is obtained from the set $\mathcal{Q}(k, A, a_j, B, b)$.

4.5 Correctness

Let us denote with the function Π the translation algorithm given in Subsections 4.2, 4.3 and 4.4 as a function from models to processes. We can now state the following proposition.

Proposition 12. For any model $\mathcal{M}$ that fulfils the conditions of Section 2, solutions $\mathcal{Z}$ and $\mathcal{Z}'$, and the stochastic π calculus specification $\Pi(\mathcal{M})$ obtained from $\mathcal{M}$, we have that $\mathcal{M} \Rightarrow \mathcal{Z} \xrightarrow{r}_{\mathsf{pim}} \mathcal{Z}'$ if and only if $\Pi(\mathcal{Z}) \xrightarrow{r} \Pi(\mathcal{Z}')$.

Proof. Proof by case analysis on the sentence used in the reduction for the 'if' direction, and on the rules of reduction given in Definition 6 for the 'only if' direction. Each step of the algorithm in Section 4 is a bijective function modulo the ordering of the sentences and their conditions. The conditions given in Subsection 2 establish the uniqueness of the reduction steps.

5 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

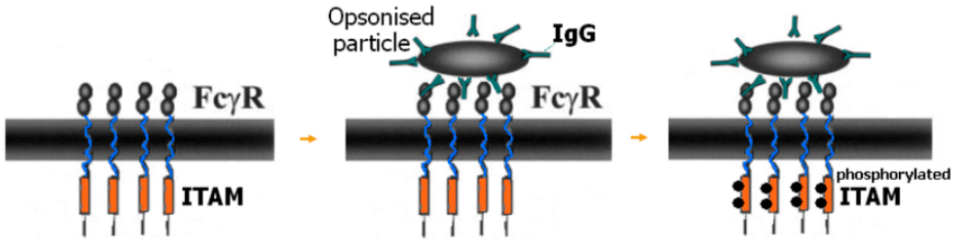

Fig. 1. A simple model of the phosphorylation of the ITAM domain on the Fcγ receptor during phagocytosis. Adapted from [14].

G (IgG) to FcγR triggers a signalling cascade that leads to actin-driven particle engulfment [14, 31, 7]. 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 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 2, which plays a role in the phosphorylation of ITAM domain. The sentences above can be easily modified and extended to capture this aspect in the model as follows.

```
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
```

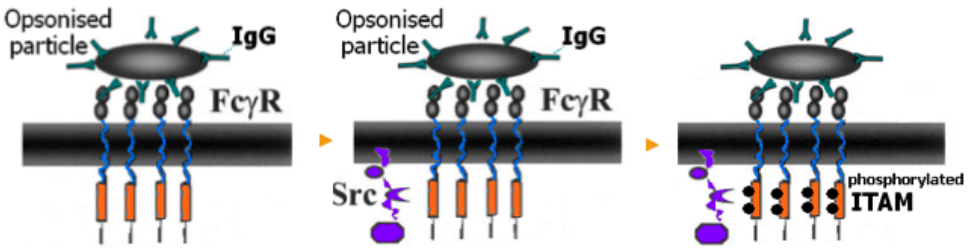

Fig. 2. A refinement of the model depicted in Figure 1 with the role of Src kinase. Adapted from [14].

```
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.

6 Expressivity

The aim of the language presented here is to provide a high level interface to stochastic π calculus models for describing phenomena that are addressed in systems biology, especially those in cellular signalling. In this respect, the level of expressivity of our narrative language suffices to address common patterns in cellular signal transduction [15].

The Kohn diagram graphical representation [19] of biological pathways provides a formal notation for the patterns that occur in cellular pathways. Indeed, the primitives presented here cover the reaction symbols of Kohn diagrams with respect to their 'combinatorial' interpretation given in [19]: association and dissociation primitives permits the representation of covalent and non-covalent bonds, including dimerisations, and their cleavage. These interactions can also be composed to represent enzymatic catalysis as illustrated in the examples above. The transformation primitive is useful in representing a limited form of stoichometric conversion and degradation of species.

The narrative language presented here does not include a primitive for the transcription symbol of Kohn diagrams, however an extension along these lines can be introduced as in, e.g., [3]. Moreover, the language here remains within the boundaries of representation to permit the analysis of models by using other techniques, e.g., the algorithm developed for κ calculus for obtaining ordinary differential equation models [13] remain within the setting of the language presented here.

Our narrative language and the translation algorithm given in Section 4 can be seen as a compilation of a biologically meaningful minimal rule-based language fragment [12, 9, 11, 2, 22, 1] into the stochastic π calculus. Such an algorithm provides a

stochastic semantics for rule based languages with respect to the stochastic π calculus. Despite the exponential blow up in the number of states during the translation, this is an advantage due to the ease in implementing an interaction-base engine in comparison to a rewrite-based engine, in particular for stochastic behaviors. This is because extending pi-calculus (or Petri nets) with a stochastic semantics and a Gillespie style implementation is now quite standard, while the stochastic semantics and implementation of rewrite rules is much more subtle due to the necessary enumeration of all the possible rewrites.

The following example that we borrow from [8] is instrumental in illustrating the language with respect to the κ calculus. Consider a biomolecular species T with a phosphorylation site x that gets phosphorylated by a kinase K and gets dephosphorylated by a phosphatase P. We can describe the interaction of these species in the narrative language with the following model.

```
site a on T associates site k on K
site a on T dissociates site k on K
site x on T gets phosphorylated if site a on T is bound

site b on T associates site p on P
site b on T dissociates site p on P
site x on T gets dephosphorylated if site b on T is bound
```

As it is illustrated in this example, the conditions of the sentences allow us to constraint the association and dissociation of the species with respect to the state of their other binding sites. This allows us to write models that capture the idea that species can change their conformation as a result of an interaction with another species, and as a result of this, they can gain or lose other interaction capabilities. This makes it possible, for instance, to describe interactions where the rate depends on the state of the species that are bound with the species that interact. However, when the states of the sites of a species do not affect the interactions they can be left unspecified. For example, to the first sentence of the model above, we can add the condition '`if site b on T is unbound`'. This would then restrict the binding of the kinase K to the states of T where the phosphatase P is not bound.

Our representation of models provides an abstraction for the stochastic π calculus models of biological systems. However, the level of abstraction that we have chosen still requires the association and dissociation sites of the species to be explicitly declared in the models. This level of abstraction can be easily lifted by automatically populating a model with sites, if these sites would be chosen not to be specified.

7 Discussion

We have introduced a natural language interface for building stochastic π calculus models of biological systems. The κ-calculus [12, 9, 10, 11] and the work on Beta-binders in [20, 17] have been a source of inspiration for this language.

In [20], 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 work presented in this paper can be seen as a translation of a fragment of the κ calculus into the stochastic π calculus. Another approach similar to the one in this paper is the work by Laneve et al. in [25], where the authors give an encoding of nano-κ-calculus in SPiM. In comparison with our algorithm, the encoding in [25] 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 in [25] gives the different states of a species in the SPiM encoding with respect to the parameters of that species as in κ-calculus.

Our narrative language has been applied to genetic programming to automate the construction of models [30]. In Ross' work, given a description of desired time-course data, generic programming explores a search space to construct a model in our language that might generate this data.

References

[1] J. A. Bachman and P. Sorger. New approaches to modeling complex biochemistry. *Nature Methods*, 8(2):130–131, 2011.

[2] M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek. BioNetGen: Software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20:3289–3292, 2004.

[3] R. Blossey, L. Cardelli, and A. Phillips. Compositionality, stochasticity and cooperativity in dynamic models of gene regulation. *HFSP Journal*, 2(1):17–28, 2008.

[4] 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.

[5] 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.

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

[7] 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.

[8] V. Danos. Agile modelling of cellular signalling. SOS'08 invited paper, 2008.

[9] 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.

[10] 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.

[11] 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.

[12] V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[13] J. Feret, V. Danos, J. Krivine, R. Harmer, and W. Fontana. Internal coarse-

graining of molecular systems. *Proceedings of the National Academy of Sciences*, 106(16):6453–6458, 2008.

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

[15] A. Goldbeter and J. E. Koshland. An amplified sensitivity arising from covalent modification in biological systems. *Proceedings of the National Academy of Sciences*, 78(11):6840–6844, 1981.

[16] P. J. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic petri nets. *PNAS*, 95(12):6750–5, 1998.

[17] M. L. Guerriero, A. Dudka, N. Underhill-Day, J. K. Heath, and C. Priami. Narrative-based computational modelling of the gp130/jak/stat signalling pathway. *BMC Systems Biology*, 3:40, 2009.

[18] 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.

[19] 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.

[20] 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.

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

[22] J. F. Ollivier, V. Shahrezaei, and P. S. Swain. Scalable rule-based modelling of allosteric proteins and biochemical networks. *PLoS Computational Biology*, 6(11):6750–5, 2010.

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

[24] 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.

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

[26] C. Priami. Stochastic pi-calculus. *The Computer Journal*, 38(7):578–589, 1995.

[27] C. Priami, P. Quaglia, and A. Romanel. BlenX static and dynamic semantics. In *CONCUR'09*, volume 5710 of *LNCS*, pages 37–52. Springer, 2009.

[28] 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.

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

[30] B. J. Ross. The evolution of higher-level biochemical reaction models. *Genetic Programming and Evolvable Machines*, 2011. in press, Technical Report at Brock University, Department of Computer Science, CS-10-02.

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

[32] 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()
```

8 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 ... *)
```