Scalable simulation of cellular signaling networks

Vincent Danos[1,4★], Jérôme Feret[3], Walter Fontana[1,2], and Jean Krivine[5]

[1] Plectix Biosystems
[2] CNRS, Université Denis Diderot
[3] Harvard Medical School
[4] École Normale Supérieure
[5] École Polytechnique

Abstract. *Given the combinatorial nature of cellular signalling pathways, where biological agents can bind and modify each other in a large number of ways, concurrent or agent-based languages seem particularly suitable for their representation and simulation [1–4]. Graphical modelling languages such as κ [5–8], or the closely related BNG language [9–14], seem to afford particular ease of expression. It is unclear however how such models can be implemented.*[6] *Even a simple model of the EGF receptor signalling network can generate more than 10^{23} non-isomorphic species [5], and therefore no approach to simulation based on enumerating species (beforehand, or even on-the-fly) can handle such models without sampling down the number of potential generated species.*
We present in this paper a radically different method which does not attempt to count species. The proposed algorothm uses a representation of the system together with a super-approximation of its 'event horizon' (all events that may happen next), and a specific correction scheme to obtain exact timings. Being completely local and not based on any kind of enumeration, this algorithm has a per event time cost which is independent of (i) the size of the set of generable species (which can even be infinite), and (ii) independent of the size of the system (ie, the number of agent instances). We show how to refine this algorithm, using concepts derived from the classical notion of causality, so that in addition to the above one also has that the even cost is depending (iii) only logarithmically on the size of the model (ie, the number of rules). Such complexity properties reflect in our implementation which, on a current computer, generates about 10^6 events per minute in the case of the simple EGF receptor model mentioned above, using a system with 10^5 agents.

1 Introduction

An important thread of work in systems biology concerns the modelling of the intra-cellular signalling networks triggered by extra-cellular stimuli (such as hormones and growth factors). Such networks determine growth, differentiation, and

[★] This research was partly supported by the NIH/NIGMS grant R43GM81319-01.
[6] Eg, from Ref. [15, p. 4]: "programs implementing these methods include StochSim, BioNetGen, and Moleculizer. However, at the present time only a part of the entire EGFR network can be analyzed using these programs".

other cell responses. Many pathological states and diseases are now traced down to subtle dysfunctions of components in those noteworks. Accordingly there is a increasing need for fine-grained, executable, and quantitative descriptions of those pathways [16].

Early on, Regev et al. [1–3] have proposed to describe those complex networks using π-calculus [17], a minimal language for concurrent systems. Variants emphasizing different types of biological processes have been put forward since [4, 18–22]. While the syntactic choices differ, they share a same concern, namely to rescue the structure-less language of chemical reactions, and to convey the combinatorics of real biological networks in a natural and executable notation. We shall use here an agent-based language called κ [5–8]. The agents we consider have internal states, accommodating protein post-translational modifications. They can also bind each other at certain specific sites called 'domains', allowing for a direct representation of protein assembly into so-called 'complexes'. This simple graph-rewriting framework naturally captures the domain level description of protein-protein interactions [23].

An example of a signalling model written in κ is that of the EGF receptor signalling network presented in Ref. [5]. This simple model generates more than 10^{23} distinct species, and that places specific demands on a simulation algorithm. Any simulation method based on enumerating species beforehand, or on-the-fly, has to sample down the combinatorics of such models to make them amenable to species counting. This is the approach followed by the current implementations of the BNG language which attempts to generate species beforehand, as well as by the recent SPIM (an implementation of stochastic π-calculus) and beta-binders (another process language for representing biological systems) implementations which register species on-the-fly [24, 25]. It is also the route taken by differential models which ignore altogether the structure of agents and so don't have the advantage of a rule-based or contextual semantics in the first place.

We propose here a radically different method which does not attempt to count species, and works even if there is an infinite number of them. The obtained algorithm has a *per event cost* which does not depend on the number of distinct species, nor does it depend on the number of agent instances in the system. The next section gives a preliminary description of the algorithm. (Some of the relevant notions only find a complete definition later in the text.) We must hasten to say that our method is not unconditionally faster, and enumeration-based techniques, including differential equations, when they apply, that is to say when the combinatorial complexity is limited, will in general be more efficient. However for the particular application to signalling systems where the combinatorial complexity makes enumeration unfeasible, only such an approach can take the complexity upfront. The simulation algorithm was implemented and tested on the EGFR example model. Using 10^5 initial agent instances of various types, it takes 30' to run that model for a total of 10^7 events on an ordinary computer. Thus this methodological route, which can non doubt still be perfected, seems to make hitherto unfeasibly complex cellular signalling models amenable to simulation.

2 Preliminaries

We recall first the generic derivation of a continuous time Markov chain from a labelled transition system. In the particular case of flat chemical reactions (aka multiset rewriting, or equivalently Petri nets) this derivation has come to be known as Gillespie's algorithm [26–28]. This method is widely used to simulate the kinetics of coupled elementary chemical reactions. The idea is to assign to a reaction a probability which is proportional to the number of its instances (or matches), while the frequency at which events are produced is obtained from the total number of rule instances.

2.1 Exponential distributions

We start with a few definitions relevant to exponential distributions, which we will need when considering the temporal aspects of the simulation algorithm.

For $a > 0$, $n \in \mathbb{N}$, $t \in \mathbb{R}^+$ define:

$$\exp_{a,n}(t) = ae^{-at}(at)^n/n! \tag{1}$$

Lemma 1 *For all $n \in \mathbb{N}$, $\exp_{a,n}$ is a probability density on $\mathbb{R}^+$ with complementary cumulative distribution function $H_{a,n}(t) := (\sum_0^n (at)^i/i!)e^{-at}$.*

Proof. $H_{a,n}$ is clearly decreasing and continuous in t; $H_{a,n}(0) = 1$, $H_{a,n}(\infty) = 0$; so $H_{a,n}$ is a complementary distribution function, and since $\exp_{a,n} = -\frac{d}{dt}H_{a,n}(t)$, $\exp_{a,n}$ is the density associated to $H_{a,n}$.

Since $H_{a,n}$ is increasing with n, the associated probability shifts to the right when n increases (see Fig. 1).

Lemma 2 *Define inductively on f in $\mathbb{N}^R$ (equipped with the product ordering):*

$$H_{a,f}(t) = \textstyle\sum_{r\in R} \frac{f(r)}{F} H_{a,f-\mathbf{1}_r}(t) + e^{-at}\frac{(at)^F}{F!}$$

with $a > 0$, $\mathbf{1}_r \in \mathbb{N}^R$ the indicator function of $\{r\}$, and $F := \sum_{r\in R} f(r)$. (By convention all terms including an $f(r) < 0$ are supposed to be zero.)

One has $H_{a,f} = H_{a,F}$.

Proof. The definition above is a well-formed inductive definition on the product ordering on $\mathbb{N}^R$ and therefore uniquely defines $H_{a,f}$. Now, defining $G_{a,f}(t) := H_{a,f}(t) - (\sum_0^F (at)^i/i!)e^{-at}$, it is easy to see that

$$G_{a,f}(t) = \textstyle\sum_{r\in R} \frac{f(r)}{F} G_{a,f-\mathbf{1}_r}(t)$$

and since $G_{a,f}(t) \equiv 0$ is a (unique) solution the conclusion follows. □

So $H_{a,f}$ is just a complicated way to write $H_{a,F}$; this will be used later where f will map a reaction s to $f(s)$ the number of clashes on an s selection attempt as defined below (F is then the total number of clashes).

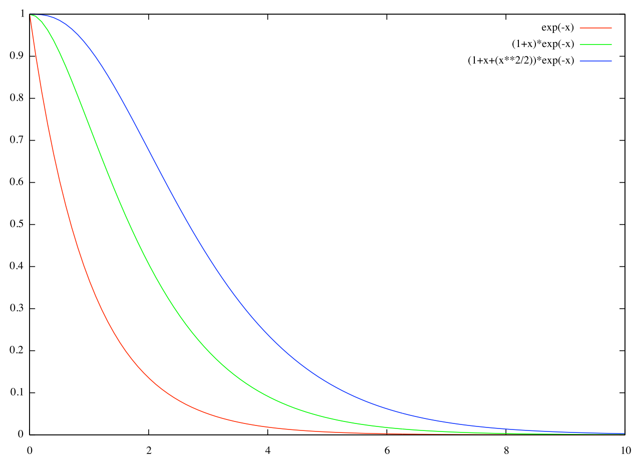

Fig. 1. *Shifted exponential distributions: $H_{1,0}$, $H_{1,1}$, and $H_{1,2}$. The base curve $H_{1,0}$ is the usual exponential distribution with density* $\exp_a(t) = ae^{-at} = \exp_{a,0}(t)$.

2.2 The basic CTMC construction

Usually one sees a labelled transition system (LTS) as an R-indexed family of binary relations $\rightarrow_r$ on the state space X. But in a quantitative setting it is important to know in how many distinct ways one can go from a state x to another one x', since the more, the likelier.[7] We will therefore start with a slight variant of LTSs that represent events explicitly and allows for counting them.

Suppose given a state space X, a finite set of labels R, a rate map τ from R to $\mathbb{R}^+$, and for each $r \in R$, and $x \in X$, a finite set of r-*events* $\mathcal{E}(x, r)$, and an *action* map $\_ \cdot x$ from $\mathcal{E}(x, r)$ to X. The action map specifies the effect of an event on the state x. We write $\mathcal{E}(x)$ for the (finite) disjoint sum $\sum_{r \in R} \mathcal{E}(x, r)$ which we will call the *event horizon*.

One can think of r as a reaction or a rewrite rule, of $\tau(r)$ as a relative measure of the rule rate, and of an event $e \in \mathcal{E}(x, r)$ as a particular application of r.

Define the *activity* of r at x as the quantity $\mathbf{a}(x, r) := \tau(r)|\mathcal{E}(x, r)|$, and the *global activity* at x as $\mathbf{a}(x) := \sum_{r \in R} \mathbf{a}(x, r) \geq 0$.

Supposing $\mathbf{a}(x) > 0$, the probability at x that the next event is $e \in \mathcal{E}(x, r)$, and the subsequent time advance are given by:

$$p(x, e) := \tau(r)/\mathbf{a}(x) \qquad (2)$$

$$p(\delta t(x) > t) := e^{-\mathbf{a}(x)t} \qquad (3)$$

[7] An example is $r_1 = A \rightarrow$, $r_2 = A, B \rightarrow A$, then $A, nB \rightarrow_{r_1} nB$ and $A, nB \rightarrow_{r_2} A, (n-1)B$, but the latter can happen in n different ways, whereas the former can happen in only one way. As a consequence A is protected from erasure by r_1 as long as there is a significant number of Bs.

The probability that the next event will be an r-event is $\mathbf{a}(x,r)/\mathbf{a}(x)$, which justifies calling $\mathbf{a}(x,r)$ the activity of r. The time advance $\delta t(x)$ is an exponential random variable with parameter $\mathbf{a}(x)$, ie has density $\exp_{\mathbf{a}(x)}$. Note that the time advance is independent from the actual event e that took place and only depends on x. Therefore, the lower the activity, the slower the system, and in the limit where the activity is zero, ie $\mathbf{a}(x) = 0$, the time advance is infinite, which means that the system is deadlocked. This implies among other things that the right unit of measure for performance of a simulation algorithm is the cost of an event, not the cost of a unit of simulation time. Indeed how many events are needed for a time unit to pass depends on the activity.

The above data $(X, R, \mathcal{E}, \tau, \cdot)$ defines a continuous time Markov chain (CTMC) with values in X, where the time advance is as in (3) above and:

$$p(x \to x') = \textstyle\sum_{r\in R} \sum_{\{e\in\mathcal{E}(x,r) \mid e\cdot x = x'\}} p(x, e)$$

2.3 Implementation by conditioning

Let us write $rand(A, f)$ for the random variable which returns an element of the set A according to the unique probability on A which has density f wrt to the uniform one. That definition will only be used for sets A with a canonical structure of measurable space that evidently carries a uniform distribution.

One gets a straightforward implementation $P(x)$ of the CTMC above as a random function that takes as an input x the current state, and returns a selected event e and a time advance δt:

$$\begin{aligned}&r := rand(R, \lambda r.\mathbf{a}(x,r)/\mathbf{a}(x));\\&e := rand(\mathcal{E}(x,r), 1);\\&\delta t := rand(\mathbb{R}^+, \exp_{\mathbf{a}(x)})\end{aligned}$$

The question one wishes to address is how to implement this Markov chain efficiently when the underlying labelled transition system is generated by a κ model. In that case x stands for the current system of agents, including their bindings and internal states, and an event $x \to_r x'$ corresponds to the application of a graph-rewriting rule r to x (which kind of graph rewriting we are using is not important at this stage of the discussion). That brings additional structure to the transition system. Specifically each rule r has a left hand side that decomposes as a multiset of connected components $C(r)$, and the set $\mathcal{E}(x,r)$, ie the set of the instances of r in x, can be naturally seen as a subset of the Cartesian product $\times_{c\in C(r)}[c, x]$, where $[c, x]$ is the set of matches for c in x. Depending on how a match is defined, $\mathcal{E}(x,r)$ may be a proper subset of the above product and therefore contain pseudo-events that do not correspond to the application of a rule.[8] Using this approximate decomposition of the event horizon $\mathcal{E}(x)$ makes it

[8] In our specific case one requires that two distinct connected components in $C(r)$ be matched to disjoint set of agents in x. For instance a rule $A, A \to B$ will mean that one must pick in x two distinct As in x. In categorical terms the 'disjoint sum' is only a weak sum in the category of graphs and graph embeddings we are considering.

possible to handle states and events locally, and at the same time preserves the CTMC semantics above as we will show now.

Suppose given for each x, and r a finite $\mathcal{E}'(x,r) \supseteq \mathcal{E}(x,r)$ (and therefore $\mathbf{a}'(x,r) \geq \mathbf{a}(x,r)$). We can define an alternative implementation $Q(x)$:

```
[0] $f := 0$;
[1] $r := rand(R, \lambda r.\mathbf{a}'(x,r)/\mathbf{a}'(x))$;
[2] $e := rand(\mathcal{E}'(x,r), 1)$;
[3] $if(e \notin \mathcal{E}(x,r))(f := f+1; goto\ [1])$;
[4] $\delta t := rand(\mathbb{R}^+, \exp_{\mathbf{a}(x),f})$
```

Just as $P(x)$, $Q(x)$ defines a distribution on $\mathcal{E}(x,r)$ since pseudo-events are rejected at step [3]. We call such a rejection a *clash*. This new procedure also defines a time distribution at step [4], the choice of which depends on f the number of successive clashes.

The probability to fail at step [3] given that r was chosen at step [1] is given by $\epsilon_r(x) = |\mathcal{E}'(x,r) \setminus \mathcal{E}(x,r)|/|\mathcal{E}'(x,r)|$. Define $\epsilon(x) := \max_{s\in R} \epsilon_s(x)$. If $\epsilon(x) = 0$ then no clash can happen, and $P(x)$ and $Q(x)$ are clearly equivalent. In fact this is always true:

Proposition 1 *For all $x \in X$, $P(x)$ and $Q(x)$ generate the same probability distribution on $\mathcal{E}(x)$ (next event), and $\mathbb{R}^+$ (time advance). The expected number of clashes for $Q(x)$ is bounded by $\epsilon(x)/(1-\epsilon(x))^2$.*

Proof. The probability to draw a rule s at step [4] and then fail at step [3] is $(\mathbf{a}'(x,s)/\mathbf{a}'(x))\epsilon_s(x)$. Therefore the probability to eventually obtain an event in $\mathcal{E}(x,r)$ is:[9]

$$\begin{aligned}
&(1-\epsilon_r(x))\frac{\mathbf{a}'(x,r)}{\mathbf{a}'(x)} \cdot \frac{1}{(1-\sum_{s\in R}(\mathbf{a}'(x,s)/\mathbf{a}'(x))\epsilon_s(x))}\\
&= \frac{(1-\epsilon_r(x))\mathbf{a}'(x,r)}{\mathbf{a}'(x) - \sum_{s\in R}\mathbf{a}'(x,s)\epsilon_s(x)} = \frac{(1-\epsilon_r(x))\mathbf{a}'(x,r)}{\sum_{s\in R}\mathbf{a}'(x,s)(1-\epsilon_s(x))}
\end{aligned}$$

and since $\mathbf{a}'(x,s)(1-\epsilon_s(x)) = \mathbf{a}(x,s)$, the above probability is $\mathbf{a}(x,r)/\mathbf{a}(x)$ which is the same as the one defined by $P(x)$.[10]

Hence the $Q(x)$ selection scheme is equivalent to that of $P(x)$ for the next event, whatever the values of ϵ_r are. Of course its expected time of convergence

[9] the left term represent the successful drawing of r at step 1, and of an e in $\mathcal{E}(x,r)$ at step 2, and the right one includes all possible sequences of failures according to the usual formula $1/(1-x) = \sum x^n$.

[10] A limit case being when for all s, $\epsilon_s(x) = 1$, or equivalently $\mathbf{a}'(x) = \sum \mathbf{a}'(x,r)\epsilon_r(x)$ (which prevents the above computation to work, see second line above), or yet equivalently when the real activity $\mathbf{a}(x)$ is zero. In this case the protocol will loop forever never finding a legitimate event, since there is none. Concretely, one stops the simulation after a certain number of successive clashes, and it works well. Such precisions are necessary since this case will happen in practice.

will depend on those values. The probability of converging after exactly n clashes is:

$$\begin{aligned}&(\sum_s \epsilon_s(x)\mathbf{a}'(x,s)/\mathbf{a}'(x))^n(\sum_s(1-\epsilon_s(x))\mathbf{a}'(x,s)/\mathbf{a}'(x)) \leq\\ &\epsilon(x)^n(\sum_s(1-\epsilon_s(x))\mathbf{a}'(x,s)/\mathbf{a}'(x)) \leq \epsilon(x)^n\end{aligned}$$

So the expected number of clashes is bounded by $\sum_n n\epsilon(x)^n = \epsilon(x)/(1-\epsilon(x))^2$.

To see that $Q(x)$ has also the time advance right, let us start with the case of a single reaction with clash probability ϵ in a given state. In this case the real activity at x is $\mathbf{a}'(x)(1-\epsilon)$, so what we need to prove is:

$$\sum_n \epsilon^n(1-\epsilon)H_{\mathbf{a}'(x),n}(t) = e^{-\mathbf{a}'(x)t(1-\epsilon)}$$

or equivalently:

$$\begin{aligned}e^{\mathbf{a}'(x)t}\sum_n \epsilon^n H_{\mathbf{a}'(x),n}(t) &= e^{-\mathbf{a}'(x)t\epsilon}/(1-\epsilon)\\ \sum_n \epsilon^n \sum_{0\leq i\leq n}(\mathbf{a}'(x)t)^i/i! &= e^{-\mathbf{a}'(x)t\epsilon}/(1-\epsilon)\end{aligned}$$

Developing the right hand side as a power series of ϵ gives:

$$e^{-\mathbf{a}'(x)t\epsilon}/(1-\epsilon) = (\sum_i \frac{(\mathbf{a}'(x)t)^i}{i!}\epsilon^i)(\sum_j \epsilon^j) = \sum_n(\sum_{0\leq i\leq n}\frac{(\mathbf{a}'(x)t)^i}{i!})\epsilon^n$$

So the time advance is correct. The case of many reactions follows easily from the same computation and Lemma 2. □

We have obtained a flexible scheme that we will use to 'pad' the event horizon and make random selections and updates of events feasible. We now turn to a definition of κ including a description of its LTS semantics (from which the CTMC semantics follows as in the general case above); we will then proceed to the detailed definition of the simulation algorithm; and finish with a discussion of the complexity aspects of the algorithm.

3 κ

We have made a certain number of simplifications to the actual language to keep the notations and definitions simple.

3.1 Agents and Interfaces

Atomic elements of the calculus are called *agents* $(a, a', \dots)$ and represent basic Lego pieces of the system. The grammar describing an agent is given Fig. 2. Each agent has a *name* and an *interface*, that is to say a set of *interaction sites* $(x, y, z, \dots)$ where each site is equipped with an *internal state* ι, and a *link state* λ. The former is used to denote post-translational modifications and sometimes cellular locations.

A site may have an unknown link state ($\lambda = ?$), or be connected to an undetermined site ($\lambda = \_$), or be connected via a particular edge ($\lambda = \alpha \in \mathcal{L}$), or be free ($\lambda = \epsilon$). The associated ordering is given Fig. 3.

$$
\begin{array}{lll}
a & ::= N(\sigma) & \text{(Agent)} \\
N & ::= A, B, \cdots \in \mathcal{N} & \text{(Name)} \\
\sigma & ::= \varnothing \mid x^{\iota,\lambda}, \sigma & \text{(Interface)} \\
\iota & ::= \epsilon & \text{(Any state)} \\
 & \quad \mid m \in \mathcal{I} & \text{(Internal state)} \\
\lambda & ::= \epsilon & \text{(Free site)} \\
 & \quad \mid ? & \text{(Bound or free site)} \\
 & \quad \mid \_ & \text{(Semi link)} \\
 & \quad \mid \alpha, \beta, \cdots \in \mathcal{L} & \text{(link)}
\end{array}
$$

Fig. 2. *Syntax of agents, assuming 3 disjoint sets of agent names $\mathcal{N}$, link names $\mathcal{L}$, and internal states $\mathcal{I}$.*

$$\begin{tikzcd}
m \in \mathcal{I} & \epsilon & & \alpha \in \mathcal{L} & {} \\
 & & & \_ \arrow[u] & \\
\epsilon \arrow[uu] & & ? \arrow[uul] \arrow[ur] & & {} \arrow[uu, "\text{Information}"']
\end{tikzcd}$$

Fig. 3. *Ordering internal and link state values.*

Let $Site(a)$ denote the sites of the agent a, $Intf(a)$ its interface, and $Name(a)$ its name. We suppose given a *signature* function Σ which maps an agent's name to the set of sites its interface may contain and we assume that $Site(a) \subseteq \Sigma(Name(a))$.

An agent a is said *partial* if its interface is partial, ie:
- there exists $x^{\iota,\lambda} \in Intf(a)$ such that $\iota = \epsilon$ or $\lambda \in \{?, \_\}$.
- or $Site(a) \subset \Sigma(Name(a))$.

Note that the form $A(x^{\epsilon,?}, \sigma)$ is equivalent (in terms of potential interactions) to the simpler form $A(\sigma)$ since no information is required concerning the states of site x. We shall thus consider agents up to the following equivalence:

$$
\begin{array}{ll}
A(x^{\iota,\lambda}, y^{\iota',\lambda'}, \sigma) & \equiv A(y^{\iota',\lambda'}, x^{\iota,\lambda}, \sigma) \\
A(x^{\epsilon,?}, \sigma) & \equiv A(\sigma)
\end{array}
$$

for x, y in $\Sigma(A)$.

3.2 Solutions and Embeddings

We use the chemical term *solution* to denote a syntactical term of the form:

$$S ::= \varnothing \mid a, S \text{ (Solution)}$$

Solutions are considered as multisets of agents and are thus taken up to congruence $a, S \equiv S, a$. In the following we will consider them as sets of occurrences of agents and by convention we use $a, b, \dots \in S$ to denote occurrences of agents in a solution S. In particular $a \neq a'$ indicates different occurrences of agents even though a may be syntactically equal to a'. We will write $(a, x) \in S$ to mean $a \in S$ with $x \in \mathit{Intf}(a)$.

Say a solution S is *well formed* if eack link name in S occurs exactly twice. Say a well formed solution is *partial* if it contains partial agents, and *complete* otherwise.

Link names α, β, ... are implicitly bound in all solutions, and we extend the equivalence on agents, and consider two solutions differing only in the names of their edges and in the position of their agents to be equivalent. As a result solutions may be seen as (site) graphs, and we shall use graph-theoretic terminology freely. We give Fig. 4 an example of the graphical notation we commonly use.

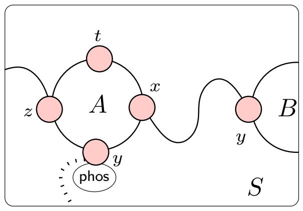

Fig. 4. *Graphical representation of the solution* $S = A(x^{\alpha}, y^{\mathsf{phos},?}, z^{-}, t), B(y^{\alpha})$. *The dotted semi edge indicates that the link state of site* y *is unknown, while the solid semi edge shows that site* z *is bound in the context. Internal state* phos *denotes a phosphorylated site.*

A map ϕ between solutions S and T is an *embedding* if for all $a, b \in S$:

$$\begin{array}{l} \phi(a) = \phi(b) \Rightarrow a = b \\ \mathit{Name}(a) = \mathit{Name}(\phi(a)) \\ \mathit{Site}(a) \subseteq \mathit{Site}(\phi(a)) \\ x^{\iota,\lambda} \in \mathit{Intf}(a) \Rightarrow x^{\iota',\lambda'} \in \mathit{Intf}(\phi(a)) \text{ with } \iota \leq \iota', \lambda \leq \lambda' \end{array}$$

where $\leq$ denotes the partial order induced by the semi lattices given Fig. 3.

Given a possibly partial map between solutions S and T, we write $\mathit{cod}(\phi)$ for the sets of occurrences of sites in the image or codomain of ϕ in T, and $\mathit{dom}(\phi)$ for those in its domain.

We say an embedding ϕ is an *iso* if it is bijective on nodes, and ϕ^{-1} is also an embedding. Two embeddings ϕ_1, ϕ_2 between S and T are said to be equivalent if there is an iso ψ from S to S such that $\phi_1 = \psi\phi_2$, and one writes $[\phi]$ for ϕ's equivalence class. Finally we write $[S, T]$ for all embedding of S in T.

We give an example of an embedding Fig. 5. Contrary to the usual notion of graph morphism, one asks embeddings to 'reflect' edges, ie a free site can only be mapped to a free one. Another unusual fact is the following simple rigidity lemma which is key for the control of the simulation complexity:

Lemma 3 (rigidity) *If S is connected, a non-empty partial map $\phi : S \to T$ extends to at most one embedding of S into T.*

So if S is connected, the number of embeddings of S in T is linear in $|T|$, and so is the cost of verifying the existence of an embedding, given a particular 'anchor' agent or site in T.

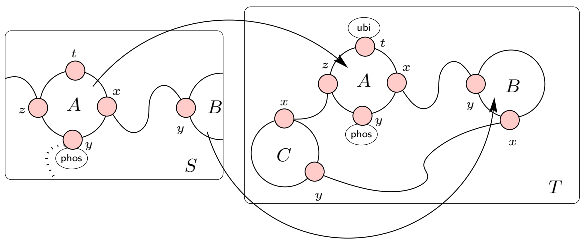

Fig. 5. *Solution S embeds into T: note that site t on A has to stay free in the codomain of the embedding.*

3.3 Rules and transitions

In contrast with process algebras where rules are simple and behaviours are mostly encoded in the processes, the dynamics of solutions in κ is expressed in rewriting rules. Rules can test the immediate environment of an agent, whereas in a process approach one would have to encode that exploration in the participating processes, (although a translation from κ to π-calculus is possible [6, 7]).

One could use double push-out methods to describe our rewrite rules, but we have found more convenient to define a rule as a pair $\langle S, act\rangle$ where S is a solution, and act is a map from agents in S to sets of *actions* subject to certain conditions explained below.

The actions one may perform on agents are:[11]
- $\mathsf{set}(x, m)$ to set the internal state of site x to $m \in \mathcal{I}$,
- $\mathsf{bnd}(x, \alpha)$ to set the link state of site x to α,
- and $\mathsf{brk}(x, \alpha)$ to set the link state of site x to ϵ.

Given a rule $r = \langle S, act\rangle$, one says:
- $(a, x) \in S$ is *ι-modified* by r if $\mathsf{set}(x, m) \in act(a)$ for some m;

[11] The full language also allows the deletion and creation of agents, but that complicates the presentation of the operational semantics. Eg if one erases an agent then one has to erase all the links it shares with its neighbours. We have refrained from presenting the full set of actions since the simulation strategy can be discussed just as well in this simpler 'mass-preserving' fragment.

- $(a,x) \in S$ is λ-*modified* by r if $\mathsf{bnd}(x,\alpha)$ or $\mathsf{brk}(x,\alpha) \in act(a)$ for some α.
One says $(a,x) \in S$ is modified by r if it is either ι-modified, or λ-modified.

An action map *act* on S is said to be *valid* if:
- every $(a,x) \in S$ is ι-modified at most once and

$$\mathsf{set}(x,m) \in act(a) \Rightarrow x^{\iota,\lambda} \in \mathit{Intf}(a),\ \iota \neq \epsilon$$

- every $(a,x) \in S$ is λ-modified at most once and

$$\begin{array}{l} \mathsf{bnd}(x,\alpha) \in act(a) \Rightarrow x^{\iota,\epsilon} \in \mathit{Intf}(a)\ ,\ \exists!(b,y) \in S : \mathsf{bnd}(y,\alpha) \in act(b)\ ,\ \alpha \notin S,\ a \neq b \\ \mathsf{brk}(x,\alpha) \in act(a)\ \Rightarrow x^{\iota,\alpha} \in \mathit{Intf}(a)\ ,\ \exists!(b,y) \in S : \mathsf{brk}(y,\alpha) \in act(b),\ a \neq b \end{array}$$

Well formedness of solutions is evidently preserved by valid actions.

Whenever *act* is an action map over S, we write $act \cdot S$ for the solution obtained by applying *act* to agents of S, with the obvious definition. Given an embedding $\phi : S' \to S$, one writes $\phi(act) \cdot S$ for the result of *act* on S along ϕ, again with the obvious definition.

Definition 1 (Transition system) *Let R be a set of rules, S a complete solution, $r = \langle S_r, act_r \rangle$ a rule in R, and $\phi : S_r \to S$ an embedding. One defines the transition relation over complete solutions associated to R as:*

$$S \to^r_\phi \phi(act_r) \cdot S$$

That definition of the LTS of a rule set fits in the framework of the preceding section:
- the state space X is the set of all complete solutions,
- the set R is the set of rules of interest,
- the r-event horizon $\mathcal{E}(x,r)$ is $\{[\phi] \mid \phi \in [S_r, x]\}$ (instances of r),
- and $[\phi] \cdot x = \phi(act_r) \cdot x$.

Thus one obtains from any κ rule set a CTMC as in Subsection 2.2.

3.4 Rule activation and inhibition

We need one last preparatory step pertaining to a well studied notion in concurrency theory namely causality [29–31]. In the particular framework of process algebra numerous notions of causality have been studied [32–35] and some were used to study dependencies among events in biological systems [36,37]. Causality is a relation among computation *events*, and we wish to define here an analog notion between rules.

Consider for instance a solution composed of a thousand As and a thousand Bs together with two rules $r_1 = A \to B$, and $r_2 = B \to C$. Then it is always the case that the application of r_1 increases the probability to trigger r_2. Thus, we may say that r_1 *activates* r_2 although it is not always the case that an instance of r_2 will use a B created by an instance of r_1 (B could be created in another way). In Section 4, activation and inhibition will allow us to bound the cost of updating various data structures after the application of a given rule in the stochastic simulation, and obtain a neat statement of its complexity properties.

A rule $r_1 = \langle S_1, act_1\rangle$ *activates* a rule $r_2 = \langle S_2, act_2\rangle$, written $r_1 \prec r_2$ if there exists S, $\phi : act_1 \cdot S_1 \to S$, and $\psi : S_2 \to S$ such that $cod(\phi) \cap cod(\psi)$ contains at least one site modified by r_1.

Similarly, r_1 *inhibits* r_2, written $r_1 \# r_2$, if there exists S, $\phi : S_1 \to S$, and $\psi : S_2 \to S$ such that $cod(\phi) \cap cod(\psi)$ contains at least one site modified by r_1.

Note that neither inhibition nor activation is *a priori* a symmetric relation. Fig. 6 shows an example of an activation.

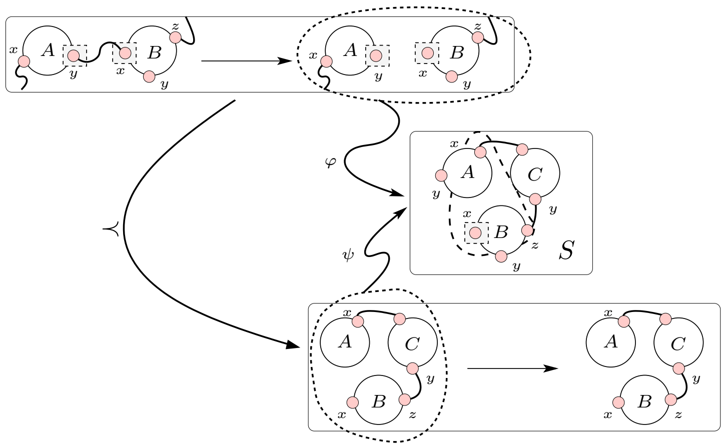

Fig. 6. *Activation relation: the image by the upper embedding ϕ of B's modified site x is also in the image of the lower embedding ψ in S; therefore the upper reaction activates the lower one.*

4 The simulation algorithm

There are three ingredients to the algorithm. The first is to introduce in the state of the simulation an explicit representation of the event horizon $\mathcal{E}(x)$. The second is to use a product approximation of $\mathcal{E}(x)$, and maintain separately a representation of the embeddings of each component of a given rule. The last ingredient is to correct for that approximation by using the time advance corrections introduced in Section 2.

4.1 The state

Given a fixed set of rules R, the *simulation state* consists in:
- a complete solution S
- a *matching map* which associates a connected component c of a rule r to the

set of its possible embeddings in S:

$$\Phi(r,c) := [c,S]$$

- an *(overestimated) activity map*, with $aut(S_r)$ is the set of automorphisms of the left hand side of rule r:[12]

$$\mathbf{a}'(r) = \tau(r)/|aut(S_r)| \cdot \times_{c\in C(r)}|\Phi(r,c)|$$

- a *lift map* which maps $(a,x) \in S$ to the set of embeddings in $\Phi(r,c)$ that have (a,x) in their codomain, for some r, and $c \in C(r)$:

$$\ell(a,x) := \{\langle r, c, \phi_c\rangle \mid \phi_c \in \Phi(r,c),\ (a,x) \in cod(\phi_c)\}$$

The maps Φ and $\mathbf{a}'$ track all rule applications and their activities. Both are computed once during an initialization phase and then updated with local cost at each simulation step. The associated data structure has a size which is controlled as follows:

Proposition 2 *The size of the matching map is linear in the size of S and bounded by $a_{max}(R) \cdot |R| \cdot |S|$ where $a_{max}(R)$ is the maximum arity in R.*

Proof: By Lemma 3, each component in $\Phi(r,c)$ is uniquely defined by the image of any agent of c in S. Therefore, $|\Phi(r,c)| \leq |S|$, and the size of the injection map is bounded by $a_{max}(R) \cdot |R| \cdot |S|\square$

4.2 The event loop

The event generating loop naturally decomposes into a *drawing phase* and an *update phase* described below (See Fig 7).

The drawing phase:

1. set $clash := 0$
2. draw some r with probability $\mathbf{a}'(r)/\mathbf{a}'(R)$
3. for $c \in C(r)$ draw uniformly $\phi_c \in \Phi(r,c)$
4. if $\sum_{C(r)} \phi_c$ is not injective increment *clash* and go to 2
5. draw time advance δt with $H_{\mathbf{a}'(S),clash}$ and increase global time
6. do $S \to^r_\phi S'$ with $\phi := \langle \phi_c; c \in C(r)\rangle$

The drawing phase is a straightforward specialisation of the protocol $Q(x)$ of Section 2 and is therefore correct. Note that the criterion for a clash is the lack of joint injectivity of the component embeddings ϕ_c. It remains now to see how to perform the updates to the event horizon that the application of the selected event made necessary.

The negative update phase:
for all pairs $(a,x) \in S$ modified by r, ϕ and $\langle r', c, \phi_c\rangle \in \ell(a,x)$ do:

[12] Recall from the preceding section that an event is isomorphism class of embeddings; the term $aut(S_r)$ makes sure that one is counting events and not embeddings.

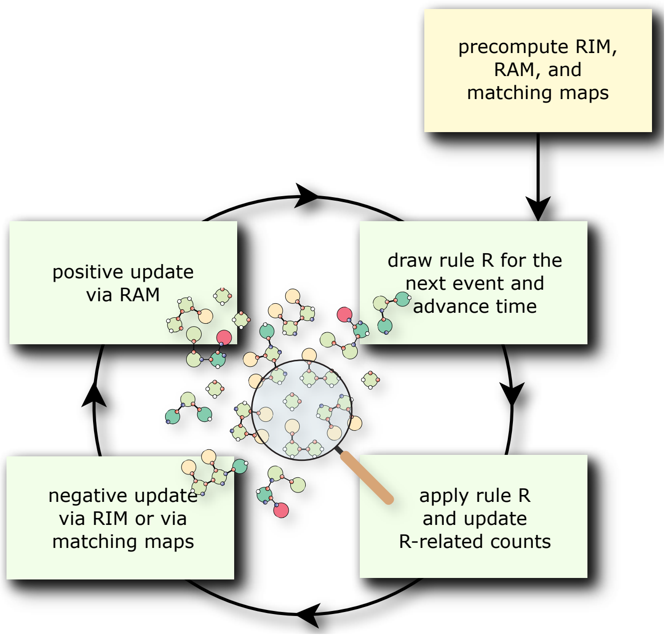
precompute RIM, RAM, and matching maps
draw rule R for the next event and advance time
apply rule R and update R-related counts
negative update via RIM or via matching maps
positive update via RAM

Fig. 7. *The event generating loop; the RIM is the rule inhibition map, the RAM is the rule activation map.*

1. remove ϕ_c from $\Phi(r', c)$ and decrease $\mathbf{a}'(r')$ accordingly
2. for all pairs $(b, y) \in cod(\phi_c)$ remove $\langle r', c, \phi_c\rangle$ from $\ell(b, y)$
3. set $\ell(a, x) := \varnothing$

The positive update phase:
for all pairs $(a, x) \in S$ modified by r, ϕ and r' such that $r \prec r'$ do:

1. for every $c \in C(r')$ try to find a (unique) embedding extension $\phi_c \in [c, S']$ of the injection $c \mapsto \{a\}$
2. for all obtained ϕ_cs add ϕ_c to $\Phi(r', c)$, increase $\mathbf{a}'(r')$ accordingly, and add $\langle r', c, \phi_c\rangle$ to $\ell(b, y)$ for all pairs $(b, y) \in cod(\phi_c)$.

The negative update consists in deleting all embeddings using sites which were modified by the application of r (and deleting associations in the lift map accordingly). It results in a decrease of the (strictly positive) activities of all the rules which were using those embeddings. In particular the activity of r decreases at this step. During the positive update one first proceed by "waking-up" all the rules which are activated by r in the sense defined in Subsection 3.4. (This is essential to control the dependency in $|R|$, but otherwise not related to the other complexity properties). Then one tries to apply those rules using the modified agent as an anchor to build new embeddings. For each of the obtained new embeddings, one updates the matching map (and the lift map accordingly) which results in a potential increase of the activities of the rules which in turn may use those embeddings.

4.3 Complexity

We bound the cost of an event loop in terms of the following parameters of the rule set:
- $s_{max}(R)$ the maximal number of sites modified by a rule,
- $c_{max}(R)$ the maximal size of a rule connected component, and
- $a_{max}(R)$ for the maximal rule arity (usually 2).
- $\delta_{\prec}(R)$ (resp. $\delta_{\#}(R)$) the maximum out-degree of the activation (resp. inhibition) map (see Subsection 3.4).

We neglect the cost induced by clashes as they only have an impact on small solutions which are not the target of our algorithm. Indeed the simulation of the EGFR example [5] for 10^6 events, with a total of 3000 agents produced only 4 clashes. The algorithm uses extensible arrays whose size is bound according to Prop. 2, so that the deletion (negative update) and insertion (positive update) or uniform selection of a component in the matching map takes a constant time.

Proposition 3 *For any rule set R, there exists constants C_1 and C_2 such that the event loop cost is bounded above by:*

$$C_1 \cdot \log(|R|) \ + C_2 \cdot a_{max}(R) \cdot c_{max}(R) \cdot s_{max}(R) \cdot (\delta_{\#}(R) + \delta_{\prec}(R))$$

Proof: The dominant cost in the drawing phase is step 2 which can be done in $C_1 \cdot \log(|R|)$ for some constant C_1 using an appropriate tree representation.[13]

Applying r at step 6 is linear in $s_{max}(R)$ since rules perform at most one modification per site. The complexity of the negative update is the following: the number of pairs (a, x) in S modified by r is bounded by $s_{max}(R)$ and for any such pair (a, x), the number of triple $\langle r', c, \phi_c \rangle$ in $\ell(a, x)$ is bounded by $a_{max}(R) \cdot \delta_{\#}(R)$. Indeed suppose (a, x) is modified by r, ϕ. Then if there is $\psi \neq \phi$ such that $\langle r', c, \psi \rangle \in \ell(a, x)$, by definition $r \# r'$. And for any rule r' there are at most $a_{max}(R)$ embeddings having (a, x) in their codomain. Steps 1 and 3 are performed in negligible time and step 2 takes a time at most proportional to $c_{max}(R)$. Hence the overall cost of the negative update is proportional to $a_{max}(R) \cdot c_{max}(R) \cdot s_{max}(R) \cdot \delta_{\#}(R)$.

The cost of the positive update is straightforward. The number of pairs (a, x) modified by r is bounded by $s_{max}(R)$ and the number of rules to wake up is bounded by $\delta_{\prec}(R)$. For each of these rules one has to look for $a_{max}(R)$ new

[13] The rule set can be represented as a tree of size $|R|$ whose nodes are triples $\langle r_i, \mathbf{a}'(r_i), \mathbf{a}'(sub_i) \rangle$ where $\mathbf{a}'(sub_i)$ is the sum of the activities of the rules contained in the left and right subtrees. Drawing a random rule according to its activity consists in generating a random number $0 < n \leq \mathbf{a}'(R)$ and, at node i, either returning r_i if $n < \mathbf{a}'(r_i)$ or doing one of the following alternatives: either going to the left subtree j whenever $n < sub_j$ or to the right subtree k and it that case set $n := n - sub_j$. This drawing scheme is in logarithmic time.

injections each of them being constructed in a time proportional to $c_{max}(R)$. So the overall positive update phase takes a time proportional to $a_{max}(R) \cdot c_{max}(R) \cdot s_{max}(R) \cdot \delta_{\prec}(R)$, and the overall time of the update phase is proportional to $a_{max}(R) \cdot c_{max}(R) \cdot s_{max}(R) \cdot (\delta_{\#}(R) + \delta_{\prec}(\mathcal{R}))$. $\square$

Note that the rule inhibition map is not used in the algorithm above, but is used in giving an upper bound on the per event cost.

5 Conclusion

We have presented a low event cost stochastic simulation algorithm for κ. This algorithm generalises the Gillespie algorithm. The key insight is to keep a representation of the next events which is linear in the size of the state, and does not present unfeasible space requirements, while being locally updatable. Although this representation introduces event clashes, it can be made to coincide with the intended stochastic semantics, by skewing the next reaction and time advance distributions in a suitable way.

In practice, as one would expect from the complexity analysis, the algorithm indeed scales well. We were able to run simulations involving a million agents, with about 50 rules, and about 10000 non-isomorphic reachable configurations, resulting in a simulation time of about 15 minutes for a million events. So even in conditions where agents far exceed in number the possible combinations, which are a priori not the best for our dimension-insensitive method, the algorithm still works. It also scales well with respect to the number of rules, because it is using a static approximate causality structure to determine whether a rule should be activated, and we ran simulations on (machine-generated) systems comprising thousands of rules, with no detectable impact on event costs.

Previous simulation methods include the traditional species-sensitive procedures, working on a ground rewriting system where every configuration is identified beforehand. This is the way the current BNG implementation works, although it is rule-based (hence the name biological network generator). The simulation then boils down to the simulation of stochastic Petri nets, and a natural implementation is to partition events into the ground reactions they correspond to, and count each class, which is an efficient thing to do for small dimensions. The fact is that all such methods have to sample the dimensionality of the rule set (since the generated network could be infinite), either explicitly, as in the current BNG implementation, or implicitly as in traditional ODE modelling, whereas as said above, the method we present here does not.

An intermediate approach one might think is worth pursuing, as in the recent betaWB implementation [25] of an extension of the beta-binders language to enable the description of complexes [38], or the latest SPIM implementation [24], is that of computing the species produced during a single trajectory on-the-fly. This will certainly fare better than a prior enumeration as in the current BNG implementation, however it is still showing a dependency in the size of that increasing set of species, because one has to scan it at each step to identify (up to isomorphism) the species just produced. In signalling systems where the set of on-the-fly species becomes large, this dependency could slow down the simulation.

The StochSim [39] simulation is based on a different agent-centric scheme, whereby one picks two agents A, B (supposing all rule are binary to simplify), and apply a reaction if any does. It shares an interesting feature with ours, namely that it behaves well with respect to the number of reactions $|R|$, an effect obtained in our case by resorting to the activation relation. However, it generates as many unproductive steps on average as there are non-reacting pairs of agents, and that number is typically $O(N^2)$ where N is the number of distinct species. This can be efficient only if the number of reactions $|R| \gg N$, ie if the reaction network is dense, which is not the expected case for signalling. Specifically, the probability of success, meaning of picking two reacting agents, is about d_m/N where d_m is the mean number of co-reactants; so the mean time for success will be N/d_m, which is increasingly bad if d_m is constant or logarithmic in N (a reasonable assumption for signalling). So the Stochsim method cost may be independent of $|R|$, but it is getting slower linearly in N (supposing d_m to be constant in N) the dimension of the system, so is highly species sensitive.

There are various attempts at general simulation engines for grammars of various sorts. An interesting one is in Ref. [40], where the authors develop a formal semantics in terms of operator algebras; another is MGS [41]. Those generic engines address a much more general situation than we have done in this paper. It should be instructive however to see to which extent the event horizon methods we have developed here apply.

References

1. A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 6, pages 459–470, Singapore, 2001. World Scientific Press.
2. Corrado Priami, Aviv Regev, Ehud Shapiro, and William Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 2001.
3. Aviv Regev and Ehud Shapiro. Cells as computation. *Nature*, 419, September 2002.
4. A. Regev, E.M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. BioAmbients: an abstraction for biological compartments. *Theoretical Computer Science*, 325(1):141–167, 2004.
5. Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, and Jean Krivine. Rule-based modelling of cellular signalling. In Luis Caires and Vasco Vasconcelos, editors, *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07)*, Lecture Notes in Computer Science, Sep 2007.
6. Pierre-Louis Curien, Vincent Danos, Jean Krivine, and Min Zhang. Computational self-assembly. Submitted, Feb 2007.
7. Vincent Danos and Cosimo Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, September 2004.

8. Vincent Danos and Cosimo Laneve. Core formal molecular biology. In *Proceedings of the 12th European Symposium on Programming, ESOP'03*, volume 2618 of *LNCS*, pages 302–318. Springer-Verlag, April 2003.
9. J.R. Faeder, M.L. Blinov, and W.S. Hlavacek. Graphical rule-based representation of signal-transduction networks. *Proc. ACM Symp. Appl. Computing*, pages 133–140, 2005.
10. J.R. Faeder, M.L. Blinov, Goldstein B., and W.S. Hlavacek. BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Complexity*, 10:22–41, 2005.
11. ML Blinov, J. Yang, JR Faeder, and WS Hlavacek. Graph theory for rule-based modeling of biochemical networks. *Proc. BioCONCUR 2005*, 2005.
12. James R. Faeder, Michael L. Blinov, Byron Goldstein, and William S. Hlavacek. Combinatorial complexity and dynamical restriction of network flows in signal transduction. *Systems Biology*, 2(1):5–15, March 2005.
13. Michael L. Blinov, James R. Faeder, Byron Goldstein, and William S. Hlavacek. A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity. *BioSystems*, 83:136–151, January 2006.
14. W.S. Hlavacek, J.R. Faeder, M.L. Blinov, R.G. Posner, M. Hucka, and W. Fontana. Rules for Modeling Signal-Transduction Systems. *Science's STKE*, 2006(344), 2006.
15. A. Kiyatkin, E. Aksamitiene, N.I. Markevich, N.M. Borisov, J.B. Hoek, and B.N. Kholodenko. Scaffolding Protein Grb2-associated Binder 1 Sustains Epidermal Growth Factor-induced Mitogenic and Survival Signaling by Multiple Positive Feedback Loops. *Journal of Biological Chemistry*, 281(29):19925, 2006.
16. B.B. Aldridge, J.M. Burke, D.A. Lauffenburger, and P.K. Sorger. Physicochemical modelling of cell signalling pathways. *Nat Cell Biol*, 8:1195–1203, 2006.
17. Robin Milner. *Communicating and mobile systems: the π-calculus.* Cambridge University Press, Cambridge, 1999.
18. Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln, José Meseguer, and Kemal Sonmez. Pathway logic: Symbolic analysis of biological signaling. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 400–412, January 2002.
19. C. Priami and P. Quaglia. Beta binders for biological interactions. *Proceedings of CMSB*, 3082:20–33, 2004.
20. Vincent Danos and Jean Krivine. Formal molecular biology done in CCS. In *Proceedings of BIO-CONCUR'03, Marseille, France*, volume 180 of *Electronic Notes in Theoretical Computer Science*, pages 31–49. Elsevier, 2003.
21. Luca Cardelli. Brane calculi. In *Proceedings of BIO-CONCUR'03, Marseille, France*, volume 180 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2003.
22. M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. *Transactions on Computational Systems Biology*, 4230:1–23, 2006.
23. T. Pawson and P. Nash. Assembly of Cell Regulatory Systems Through Protein Interaction Domains. *Science*, 300(5618):445–452, 2003.
24. Andrew Phillips and Luca Cardelli. Efficient, correct simulation of biological processes in the stochastic pi-calculus. In *Proceedings of CMSB'07*, 2007. To appear.
25. Pierpaolo Degano, Davide Prandi, Corrado Priami, and Paola Quaglia. Beta-binders for biological quantitative experiments. In *Proceedings of QAPL 2006*, volume 164 of *ENTCS*, pages 101–117, 2006.

26. A. B. Bortz, M. H. Kalos, and J. L. Lebowitz. A new algorithm for Monte Carlo simulation of Ising spin systems. *J. Comp. Phys.*, 17:10–18, 1975.
27. Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem*, 81:2340–2361, 1977.
28. Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Phys.*, 22:403–434, 1976.
29. Gérard Berry and Jean-Jacques Lévy. Minimal and optimal computation of recursive programs. *JACM*, 26:148–175, 1979.
30. Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. Petri nets, event structures and domains. *Theoretical Computer Science*, 13:85–108, 1981.
31. Philippe Darondeau and Pierpaolo Degano. Causal trees. In *Proceedings of ICALP'89*, volume 372 of *LNCS*, pages 234–248, 1989.
32. Glynn Winskel. Event structure semantics for CCS and related languages. In *Proceedings of 9th ICALP*, volume 140, pages 561–576, 1982.
33. Gérard Boudol and Ilaria Castellani. Permutation of transitions: An event structure semantics for CCS and SCCS. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 411–427, 1989.
34. Michele Boreale and Davide Sangiorgi. A fully abstract semantics for causality in the π-calculus. *Acta Inf.*, 35:353–400, 1998.
35. Pierpaolo Degano and Corrado Priami. Non-interleaving semantics for mobile processes. *Theoretical Computer Science*, 216(1–2):237–270, 1999.
36. C. Baldi, Pierpaolo Degano, and Corrado Priami. Causal π-calculus for biochemical modeling. In *Proceedings of the AI*IA Workshop on BioInformatics 2002*, pages 69–72, 2002.
37. M. Curti, P. Degano, C. Priami, and C. T. Baldari. Modelling biochemical pathways through enhanced π-calculus. *Theor. Comp. Sci.*, 325(1):111–140, 2004. Online.
38. Corrado Priami and Paola Quaglia. Beta binders for biological interactions. In *Computational Methods in Systems Biology*, volume 3082, pages 20–33, 2005.
39. C. J. Morton-Firth. *Stochastic simulation of cell signalling pathways.* PhD thesis, Cambridge, 1998.
40. Eric Mjolsness and Guy Yosiphon. Stochastic process semantics for dynamical grammars. *Annals of Mathematics and Artificial Intelligence*, 2007.
41. Jean-Louis Giavitto and Olivier Michel. MGS: a programming language for the transformations of topological collections. Technical Report 61-2001, LaMI, 2001.