External Control in Process Algebra for Systems Biology

Filippo Del Tedesco and Carla Piazza[1,2]

Department of Mathematics and Computer Science
Università di Udine
Via delle Scienze, 206, 33100 Udine
Italy

Abstract

A critical aspect in the modeling of biological systems is the description view point. On the one hand, the Stochastic π-calculus formalism provides an intuitive and compact representation from an internal perspective. On the other hand, other proposed languages such as Hybrid Automata and Stochastic Concurrent Constraint Programming introduce in the system description an external control and provide more structured models.
This work aims at bridging the above discussed gap. In particular, we propose a different approach for the encoding of biological systems in Stochastic π-calculus in the direction of introducing an external control and comparing different formalisms. We show the effectiveness of our method on some examples.

Keywords: Stochastic pi-calculus, Simulation of Chemical Reactions, SPIM.

Introduction

Systems Biology exploits different languages and formalisms mainly inherited from mathematics and computer science with the aim of modeling and analyzing complex biological systems in terms of their components and their interactions. A formal modeling and understanding of the underlying laws of life are at the basis of all the scientific research in biology. We just mention here the impact that systems biology could produce on the development of new therapies and drugs.

The huge complexity of biological systems calls for the definition of a range of modeling languages operating at different levels of description (e.g., different space/time scales). However, it is also essential to compare and integrate such languages in a global framework in which information is shared and analyzed from many perspectives.

[1] This work is partially supported by PRIN 2006 BISCA project and by BioCheck regional project.
[2] Email: carla.piazza@dimi.uniud.it

Among the formalisms proposed for systems biology, the *Stochastic π-calculus*, proposed by Priami in [15], has received growing attention in the last years [16,14,13,6]. Many advantages come by the use of Stochastic π-calculus and more in general by the use of process calculi: if we consider the problem of describing a system, it allows to model biological entities (e.g., molecules, genes, proteins) as *processes* and entity interactions as *communications* between processes. This perspective has been taken also in other process algebras developed for systems biology. Such approach has some interesting peculiarities: for instance it is strongly compositional, and it explicitly models the biological meaning (functional activity) of each entity. In this sense we can say that it represents the system from an *internal point of view*, because all the terms inside the representation have a counterpart in the physical system, so there are not "external" contributions in the description. Stochastic π-calculus is also powerful for the analysis, because it inherits all the theoretical results, such as the notion of bisimulation, that are already available for the more traditional calculi (*CCS*, π-calculus, ...). Unfortunately, it also has few critical aspects, such as the fact that describing interactions by communications forces to consider only binary exchanges, since communications among three or more principals are not immediately encodable inside the language.

On the other hand, *Ordinary Differential Equations* (see, e.g., [17]) describe the system from a different perspective. If x_i is an entity and X_i is the quantity of x_i in the system, the differential equation $\dot{X}_i = f(X_1, \ldots, X_n)$ represents the time evolution of x_i, while the biological meaning of x_i (which is explicit in the internal point of view) has to be reconstructed by looking at all the right hand sides of the system equations. Roughly speaking we can say that the differential equation $\dot{X}_i = f(X_1, \ldots, X_n)$ allows to syntactically represent an *external view* of the system: from outside we can measure the global quantity of each entity; moreover, the analytic structure of the differential equation is a (not always immediate) translation in mathematical language of the interactions between entities. Such external point of view in the Stochastic π-calculus models is hidden at a semantic level, where the reaction rates depend on the total amount of reactants. Of course there are disadvantages also in the use of differential equations: it is very difficult to find their exact solutions and, at the same time, they lack the linguistic structure of process algebra and hence standard model-checking techniques are hard to apply.

Other formalisms known in literature, such as *Stochastic Concurrent Constraint Programming* (sCCP) [3,4] and *Hybrid Automata* [1,11,7], push even forward the discrepancy between the internal and the external perspectives by allowing at the syntactic level to model changes in dynamic laws according on global constraints (e.g., activation and reset conditions in the case of hybrid automata). In other words, while we talked about an "internal" perspective of the Stochastic π, here we have an "external" contribution of constraints and control structures that do not really exist in the physical system.

We propose a different use of Stochastic π-calculus in the modeling of biological systems with the aim of joining its positive aspects with the possibility of introducing the external control structure we have just mentioned for hybrid automata. To obtain this, we model the entities as messages on a "memory" channel and the interactions as processes. In particular, we focus on chemical reactions and we use a single process to coordinate all the reactions. The process is external with respect to the original system, since it "counts" the global amount of entities contained in it. In a sense our approach try to bridge the gap between the Stochastic π-calculus models described in [16,14,13,6] and other formalisms such as sCCP and hybrid automata, moving the control on the global state of the system from the semantics to the syntactic level. Our main aim is that of comparing / integrating different modeling languages. This should allow us to extend standard analysis techniques of process algebra to other formalisms. Interestingly, our approach also allows to easily model n-ary chemical reactions.

The paper is organized as follows. In Section 1 we recall the syntax and semantics of both Stochastic π-calculus and SPiM [13], while we present our proposal in Section 2. Some examples are discussed in Section 3. Section 4 ends the paper with some remarks and future work proposals. More technical details and semantics equivalences can be found in [9].

1 Stochastic Pi-Calculus, Biology, and Simulations

Since the beginning of our work, we have chosen to implement our approach using SPiM, an high level language based on Stochastic π-calculus. The reason for this is twofold: from the one hand, Stochastic π is well known in the literature, and it is already widely used in biologic descriptions. On the other hand, SPiM is, at the same time, rather intuitive (also for non-trained user) and equivalent to Stochastic π. So, in this section we would like to offer a short introduction to these two languages and to their use in biological modeling.

1.1 *Stochastic Pi-Calculus*

Stochastic π-calculus is an extension of the π-calculus process algebra. It has been introduced in [15] with the aim of modeling performances of dynamically reconfigurable or mobile networks. It inherits all the syntax of π-calculus and enriches this last with the possibility of associating to each action a probability distribution. As a consequence, we can associate to each prefix a time duration, represented by the value of a random variable, which follows the above mentioned probability distribution. In the case of memoryless processes exponential distributions can be used and actions take the form (a, r), also denoted by a_r, where a is the action name and r (*activity rate*) is the parameter characterizing the exponential distribution, i.e., the average duration of (a, r) is $1/r$. Notice that the language also allows instantaneous actions, corresponding to $r = \infty$, in which the rate is omitted.

Let us briefly recall the syntax of Stochastic π-calculus.

Definition 1.1 [Stochastic π-calculus – Syntax] Let $\mathcal{N} = \{a, b, .., x, y, \dots\}$ be an infinite set of names. A *Stochastic π-calculus process* is an expression of the following grammar:

$$P ::= 0 \mid (\pi, r).P \mid (\nu x)P \mid [x = y]P \mid P|P \mid P + P \mid P(y_1, .., y_n)$$

where $r \in \mathbb{R}^+ \cup \{\infty\}$.

The intuitive meaning of the operators is essentially the same as in π-calculus. In particular:

- π is either $x(y)$ or $\bar{x}y$ or τ. $x(y)$ denotes that we are waiting for a message on the channel x and y acts as a placeholder, which will be replaced with the received message. $\bar{x}y$ represents the output of the message y on the channel x. τ is the silent action. All the standard considerations about free and bound names hold.
- $P + Q$ stochastically behaves as either P or Q. The choice depends on the time durations of the actions occurring in P and Q. The fastest will win the race. This is one of the main differences with π-calculus, where + is a nondeterministic choice operator.
- In general the behavior of processes depends on *race conditions*: among all the executable activities we will activate the one which has the shortest duration.

A complete presentation of the operational semantics of the Stochastic π-calculus is outside the scope of this paper and can be found in [15]. We also implicitly adopt all the standard syntactic conventions.

Example 1.2 Consider the following toy example, representing the interaction between a man and a coffee/tea machine.

$$\begin{aligned} Man &\overset{def}{=} \overline{coin}\,am.\overline{choi}\,cof.drink(smthg) \\ Machine &\overset{def}{=} coin(m).choi(c).(([c = cof]\overline{drink}\,cof) + ([c = tea]\overline{drink}\,tea)) \\ System &\overset{def}{=} Man \mid Machine \end{aligned}$$

In this case all the actions are instantaneous and there are no stochastic effects.

The following example uses the stochastic features of the calculus on a mutual exclusion protocol.

$$\begin{aligned} P_1 &\overset{def}{=} \overline{down1}_{r1}\,pars1.0 \\ P_2 &\overset{def}{=} \overline{down2}_{r2}\,pars2.0 \\ mutex &\overset{def}{=} (down1_{r1}(p1).0 + down2_{r2}(p2).0) \\ race &\overset{def}{=} P_1|P_2|mutex \end{aligned}$$

The fastest process win.

1.2 Stochastic Pi-Calculus in Systems Biology

Many works have pointed out the usefulness of Stochastic π-calculus in the modeling of biological systems (see, e.g., [16,14,13,6]). We start focusing on [6]. Cardelli observes that the available description languages for biochemical system (e.g., state transition diagrams) are in a sense similar to process algebras, since in both cases labeled transition systems represent concurrent systems. This is not the case in differential equation models. Hence, the use of process algebras in systems biology is natural and provides double advantages: on the one hand, the representation is incremental and compositional; on the other hand, the models support formal verification techniques such as behavioral equivalences and model checking.

Then, the paper formally establishes connections between discrete and continuous descriptions of systems of chemical reactions. The discrete representations are given as processes of a fragment of the Stochastic π-calculus, while the continuous models are systems of ordinary differential equations.

Let us formally introduce the above mentioned representations:

- Chemical reactions are expression of the form $A \xrightarrow{k} B_1 + ... + B_n$ or $A_1 + A_2 \xrightarrow{k} B_1 + ... + B_n$ or $A + A \xrightarrow{k} B_1 + ... + B_n$. They represent the problem from a *macroscopic* point of view.
- Ordinary differential equations can be automatically inferred from chemical reactions and describe the dynamic of a reactant concentration in terms of the other concentrations.
- The *chemical ground form* (CGF) is a subset of the Stochastic π-calculus. It does not allow the use of the restriction operator (νx) and of the test operator $[x = y]$. It is expressive enough to represent each chemical entity involved in the system and hence it provides a *microscopic* description.

Example 1.3 Let us consider the *Na-Cl* ionization example. The chemical reaction describing the system is the following:

$$Na + Cl \xrightarrow{k} Na^{+} + Cl^{-}$$

The corresponding differential equations are:

$$\begin{aligned} d[Na^{+}]/dt &= d[Cl^{-}]/dt = k[Na][Cl] \\ d[Na]/dt &= d[Cl]/dt = -k[Na][Cl] \end{aligned}$$

In the CGF we have the parallel composition of the processes:

$$Cl = \bar{e}_{k/g}().0 \qquad Na = e_{k/g}().(Na^{+}|Cl^{-})$$

where g is a constant for the dimensional conversion (see [6]).

In [6] equivalences between the discrete and continuous semantics of chemical reactions, ordinary differential equations, and CGF are proved. This formally clarifies the connections between the microscopic and macroscopic points of view.

A graphical representation of the above described approach is presented in [14] on a variant of the Stochastic π-calculus. This is at the basis of the current implementation of the stochastic simulator SPiM [13]. In particular, in [14] the language is specialized for the biological context. A system has the form $E \vdash P$, where E is a *constant environment*, i.e., a library of entity definitions, and P is the *test tube* containing all the copies of the entities. Semantics equivalences between the language proposed in this paper and the original Stochastic π-calculus are proved.

Example 1.4 Let us consider a system consisting of a gene G which can synthesize a protein A in time τ_t. Moreover, protein B, produced by another gene, can either inhibit gene G for time τ_u or decay in time τ_d. The description of the system in Graphical Stochastic π-calculus is the following:

$$E \;:\; G \overset{def}{=} \tau_t.(G|A) + b_r().\tau_u.G \qquad P \;:\; G|B$$
$$B \overset{def}{=} \overline{b}_r.B + \tau_d$$

1.3 SPiM

The *Stochastic Pi-Machine* (SPiM) simulates the behavior of Stochastic π-calculus processes. The latest versions of SPiM have been optimized for the simulation of processes representing biological systems [13]. The input for the simulator is a process. SPiM simulates the time evolution of a given process determining the amount of time that each action requires and by selecting the next action. The selection is based on a random procedure. At the basis of SPiM computation there is a variant of Gillespie's algorithm [12], a Monte Carlo simulation procedure for Continuous Time Markov Chains (CTMC).

SPiM's input language is a high level translation of the Stochastic π-calculus. Since, we will use it in the remaining part of this work, we briefly introduce here part of its syntax. We address the reader to [13] for all the details.

Definition 1.5 [SPiM – Syntax]

$$Dec \;::=\; new\ x\{@r\} : chan\{(Type_1, ..., Type_n)\} \;\mid\; \text{run P}$$
$$\mid\; \text{let } D_1 \text{ and } ... \text{ and } D_N$$
$$P \;::=\; X(v_1, ..., v_n) \;\mid\; \textbf{if}\ b\ \textbf{then}\ P\ \{\textbf{else}\ P\} \;\mid\; () \;\mid\; \pi\{;P\}$$
$$\mid\; (P_1|...|P_M) \;\mid\; \textbf{do}\ \pi_1\{;P_1\}\ \textbf{or}\ ...\ \textbf{or}\ \pi_M\{;P_M\} \;\mid\; n \text{ of } P$$
$$D \;::=\; X(m_1, ..., m_N) = P$$
$$\pi \;::=\; !x\{(v_1, ..., v_N)\} \;\mid\; ?x\{(m_1, ..., m_N)\} \;\mid\; \textbf{delay}@r$$

Let us give some intuitions about the meaning of the operators with an example.

Example 1.6 Let us consider again Example 1.3 where we described the chemical reaction $Na + Cl \xrightarrow{k} Na^+ + Cl^-$. The SPiM program representing it can be written as follows.

```
new e@r:chan (*declaration of channel e with rate r=k/g*)
let Na()=?e;(Nap()|Clm()) (*declaration of Na molecule*)
 and Cl()=!e;()
 and Nap()=()
 and Clm()=()

run (Na()|Cl())
```

Similarly Example 1.4 can be translated in SPiM as follows.

```
new a@0.4:chan
new b@0.4:chan
let G()=do delay@1.0;(G()|A())
        or ?b;delay@0.5;G()
```

```
   and A()=do !a;A()
           or delay@0.3;()
   and B()=do !b;B()
           or delay@0.3;()
  run (G()|B())
```

1.4 Internal versus External

We already noticed that compositionality is one of the main advantages of the use of Stochastic π-calculus in the modeling of biological systems. As emerged from the examples, once we have defined the actors of the system and their rules (interactions, delays, . . .), we only have to let them play. This represents an internal perspective on the system, in the sense that each actor inside the representation exists also (or, better, has a correspondent entity) in the physical system. Moreover, the actions executed inside the model find an immediate counterpart in the reactions occurring in the real system.

On the opposite side, as we mentioned in the Introduction, we can find formalisms, such as hybrid automata, based on external perspectives, with their own advantages and disadvantages.

How can we compare and integrate internal and external perspectives?

In the next section we try to partially answer this question, proposing a different use of Stochastic π-calculus which is closer to formalisms with external perspectives.

2 Memory and Pi-Molecules

We start with an informal description of our approach for modeling biochemical systems in (Graphical) Stochastic π-calculus. The idea is that of introducing an external control mechanism in the model. We aim to: (1) keep the representation simple and compositional; (2) remain inside the Graphical Stochastic π-calculus language.

The approaches described in the previous section model the reactants of biochemical systems as *active* entities. Each of them plays a fundamental role in the system evolution.

A first idea could be that of introducing a control process in the system and asking the reactants to communicate only with the control process. The control process should acquire information from the reactants and give them back the orders. Unfortunately, this makes life very complex, since too many communications and stochastic rates are involved. As a matter of fact in this way both the reactants and the reactions would be *active* players.

So we propose to have *active* reactions and *passive* reactants.

Let us focus on the *passive* reactants. We model them as arguments of messages which are sent and received on a *memory* channel. In particular, inside a message of the form `!mem(react1,...,reactn)` the argument `react`*i* represents the number of molecules of the *i*th reactant inside the system.

Now we have to model the active reactions. Our idea is to consider them all together inside a unique abstract entity that we call π-Molecule. The role of the π-Molecule is to manage all the control flows that are related to the biochemical systems we are describing.

From a high level perspective, a π-Molecule can be thought as a cyclic and unbounded execution of the following operations: acquisition of memory, selection of *feasible* reactions, race condition among feasible reactions, and memory update.

The first instruction is easy to implement: since the memory is represented as a polyadic output message, we only have to synchronize the π-Molecule with it by performing an input command on the same channel, namely `?mem(x1,...,xn)`.

The selection of feasible reactions depends on their preconditions: for each reaction the π-Molecule checks if the system has enough reactants to trigger the reaction. In this way, if we consider a system composed of n reactions, the π-Molecule will provide all the 2^n possible combinations among the reactions (i.e., the powerset of the set of reactions), and this includes also the empty one that represents the termination condition for the simulation. Clearly, at each step the configuration of the memory will satisfy one (and only one) combination.

The third part of the π-Molecule is devoted to the race condition among all feasible reactions, and it has been built upon the results of Gillespie ([12]).

Let us intuitively argue about the soundness of our approach. Consider a biochemical system in which we can observe the execution of two chemical reactions exactly once: we expect to see (in the average case) first the products of the fastest one. But if the reactants for the fastest reaction are scarce, we may see the contrary, because the slowest reaction has more possibilities to win the race. For short, we have that the speed of the reaction is determined by the chemical rate of the reaction together with the quantity of each involved reactant. Gillespie formalized all these considerations by proving that the speed of a reaction depends on the reaction rate times the number of possible combinations of m_i molecules of reactant i in groups of c_i, where m_i is the number of i molecules in the system and c_i is the number of i molecules required by the reaction. To implement this property, we describe each reaction by a delay action (`delay@...`) with a rate that is the rate of the reaction times the number of possible combinations of its reactants, as explained before. In this way each reaction is simulated by a delay, no matter how many reactants it uses.

Finally, the π-Molecule updates the state, by consuming and producing reactants, and concludes its activities with a self recursive call.

Example 2.1 We propose a simple implementation of the above ideas. Let us enrich the chemical system of Example 1.3 by allowing the reaction to be bidirectional (ionization has rate 100.0, deionization has rate 10.0), namely $Na + Cl \rightleftharpoons Na^+ + Cl^-$. Let us suppose that the system starts from an initial state which has the following values for the chemical species: $Na = 100$, $Cl = 100$, $Na^+ = 0$, $Cl^- = 0$.

To encode such a system in SPiM we must begin with an easy syntactic adjustment, that splits the reaction into $Na + Cl \overset{100.0}{\to} Na^+ + Cl^-$ and $Na^+ + Cl^- \overset{10.0}{\to} Na + Cl$. Now, we can implement the following π-Molecule to properly simulate the system:

```
new mem:chan(float,float,float,float)
```

```
let PiM()=
 ?mem(na,cl,nap,clm);
 if(na>0.0)
  then (*ionization is feasible*)
   if(nap>0.0)
    then (*deionization is feasible*)
     do
      delay@100.0*na*cl;
      (!mem(na-1.0,cl-1.0,nap+1.0,clm+1.0)|PiM())
     or
      delay@10.0*nap*clm;
      (!mem(na+1.0,cl+1.0,nap-1.0,clm-1.0)|PiM())
    else (*deionization not feasible*)
     delay@100.0*na*cl;
     (!mem(na-1.0,cl-1.0,nap+1.0,clm+1.0)|PiM())
  else (*ionization not feasible*)
   if(nap>0.0)
    then (*deionization feasible*)
     delay@10.0*nap*clm;
     (!mem(na+1.0,cl+1.0,nap-1.0,clm-1.0)|PiM())
    else (*deionization not feasible*)
     println("end of simulation")

run (!mem(100.0,100.0,0.0,0.0)|PiM())
```

The first line is a declaration for a polyadic channel *mem*. Notice that the channel does not have any rate, so we have instantaneous exchanges on it. This agrees with our target, that is to demand the stochastic behavior to the execution phase.

The last line sets the simulator properly, by creating a system with exactly one instance of π-Molecule and one of memory.

Let us consider a set of chemical reactions C consisting of M reactions of the form "$i : c_{i,1}A_1 + ... + c_{i,n}A_n \xrightarrow{k_i} p_{i,1}A_1 + ... + p_{i,n}A_n$". Let us also consider an initial state for the system given by the values $m_1, ..., m_n$ for the species $A_1, ..., A_n$, respectively. C together with $\langle m_1, ..., m_n\rangle$ creates what we called a *general chemical system*.

The procedure π-MoleculeBuilder, described in Table 1, explains how to create the π-Molecule for $(C, \langle m_1, ..., m_n\rangle)$.

It is possible to prove an equivalence between the discrete semantics of a general chemical system and the discrete semantics of the corresponding program obtained using π-MoleculeBuilder (we called such program *virtual chemical system*, considering it as a couple composed by the program for the chemical reactions itself and the *mem* component representing the initial state).

The semantics of both the two systems is defined following [6], and it is based upon the notions of labelled transition graph ψ and Continuous Time Markov Chain (CTMC) $|\psi|$.

If we consider that the SPiM implementation of a system of chemical reactions is composed of proper chemical states (the execution point just before the acquisition

HEADER	
new mem:chan(float,...,float)	Declaration of an n-ary channel
DECLARATION AND MEMORY ACQUISITION	
let PiM()=	
?mem(a_1,...,a_n);	
PRECONDITION CHECK	
REACTION EXECUTION	
if(($a_1 \geq c_{1,1}$)*...*($a_n \geq c_{1,n}$))	Precondition for reaction 1
then if (($a_1 \geq c_{2,1}$'')*...*($a_n \geq c_{2,n}$))	Precondition for reaction 2
...	All other preconditions
then	Feasible reactions: 1, 3, 5
do	
delay@k_1*$\binom{a_1}{c_{1,1}}$*...*$\binom{a_n}{c_{1,n}}$;	Reaction simulation
(!mem($a_1 + p_{1,1}$,...,$a_n + p_{1,n}$) | PiM())	State update and recursion
or	
delay@k_3*$\binom{a_1}{c_{3,1}}$*...*$\binom{a_n}{c_{3,n}}$;	Reaction simulation
(!mem($a_1 + p_{3,1}$,...,$a_n + p_{3,n}$) | PiM())	State update and recursion
or	
delay@k_5*$\binom{a_1}{c_{5,1}}$*...*$\binom{a_n}{c_{5,n}}$;	Reaction simulation
(!mem($a_1 + p_{5,1}$,...,$a_n + p_{5,n}$) | PiM())	State update and recursion
...	
else println(‘‘end of simulation’’)	No feasible reactions
SIMULATION DIRECTIVE	
run (!mem(m_1,...,m_n) | PiM())	

Table 1
π-MoleculeBuilder

of memory) and improper chemical states (which have not any counterpart in the chemical behavior of the system) it is possible to demonstrate (refer to [9] for all the details) that the following theorem holds.

Theorem 2.2 *Let $|\psi_C|$ and $|\psi_V|$ be the two CTMCs associated to the general chemical system $(C, \langle m_1, ..., m_n \rangle)$ and to the virtual chemical system (M, mem), respectively. There is a weak bisimulation between $|\psi_C|$ and the proper chemical states of $|\psi_V|$.*

It is worth to point out that the version of the π-Molecule returned by π-MoleculeBuilder does not produce any visible interaction with the user, so if we are interested in this we must enrich the implementation with some instructions to expose the value of the *mem* component. For instance, in Example 2.1 we obtain this by declaring an instantaneous output channel, `new sinc:chan(float,float)`, that is used every time a reaction occurs. In particular, we add the command `!sinc(na-1.0,nap+1.0)` to each reaction branch of the π-Molecule. The command synchronizes with a printing process, namely `Printmem`, which uses the SPiM syntax to print the *mem* values:

```
let Printmem()=
 ?sinc(b,c); print(show b); println(show c); Printmem()
```

In order to properly instruct the simulator the last line has to be changed as follows:

```
run(!mem(100.0,100.0,0.0,0.0)|PiM()|Printmem())
```

Summarizing we can say that from the structural point of view our approach is compact and modular. Further, since in our approach we focus more on the chemical reactions, than on the reactants, the complexity of our models should not depend on the reactant quantities. Moreover, the kinetic behavior is syntactically (explicitly) declared in the delay rates, which depends also on the reactant concentrations.

3 Some Examples

In this section we present a set of case studies. Some case studies are taken from the SPiM test suite[3], where they are modeled and studied exploiting the approach described in [16,14,13,6]. Here we model them as defined in our approach and we compare the simulation results, always obtained using SPiM. Our last case study, namely the Brusselator, is taken from [12] and cannot be modeled with the classical *molecule centric* approach, since it also involves ternary reactions. It is immediate to model it with our approach.

3.1 NaCl

Let us consider again the two reactions $Na+Cl \overset{100}{\rightarrow} Na^{+}+Cl^{-}$ and $Na^{+}+Cl^{-} \overset{10}{\rightarrow} Na+Cl$.

We already saw in Section 2 how to model them with our approach, while the classical SPiM model is available in the SPiM test suite. We show the results of the simulation of our model on the left part of Figure 1, while on the right side we have the results obtained with the classical approach. In both cases the initial state has 100 molecules of Na and 100 molecules of Cl.

Fig. 1. $Na + Cl \rightleftharpoons Na^{+} + Cl^{-}$

Notice that, since in our approach we print the configuration of memory after each reaction, while in the classical approach the number of processes is measured on a given time window, there are some discrepancies in the time scales of our graphs.

3.2 Mg2Cl

In this case we consider another bidirectional reaction of ionization, namely

$$Mg + 2Cl \rightleftharpoons Mg^{2+} + 2Cl^{-}$$

Using the classical approach one can only describe binary reactions, so the encoding of the system is based on the use of a short term molecule, namely Mg^{+}. With this trick, the reactions become $Mg + Cl \rightleftharpoons Mg^{+} + Cl^{-}$ and $Mg^{+} + Cl \rightleftharpoons Mg^{2+} + Cl^{-}$, so they can be translated in $Mg+Cl \overset{10}{\rightarrow} Mg^{+}+Cl^{-}$, $Mg^{+}+Cl^{-} \overset{50}{\rightarrow} Mg+Cl$,

[3] Available at `http://research.microsoft.com/~aphillip/spim/Examples.pdf`

$Mg^+ + Cl \overset{100}{\rightarrow} Mg^{2+} + Cl^-$ and $Mg^{2+} + Cl^- \overset{5}{\rightarrow} Mg^+ + Cl$. In Figure 2 we show our results (left) and the classical ones (right). The initial state is the same for both simulations and it has 100 molecules of Mg and 100 molecules of Cl.

Fig. 2. $Mg + 2Cl \rightleftharpoons Mg^{2+} + 2Cl^-$

We would like to stress that our approach does not need the introduction of the new molecule Mg^+, since inside our framework one can immediately implement ternary reactions. The results in Figure 3 are indeed obtained by directly implementing the reactions $Mg + 2Cl \overset{500}{\rightarrow} Mg^{2+} + 2Cl^-$ and $Mg^{2+} + 2Cl^- \overset{250}{\rightarrow} Mg + Cl$ with our approach and confirm the expressive power of our approach.

Fig. 3. $Mg + 2Cl \rightleftharpoons Mg^{2+} + 2Cl^-$ - π-Molecule (1 step)

3.3 Repressilator

Let us show how our approach can be easily applied also in the case of genetic regulatory networks.

We consider the Repressilator example which is described also in the SPiM distribution. The Repressilator is a biological system containing three genes, namely A, B, and C. Each gene produces a protein, in particular b is produced by A, c by B, and a by C. Each gene can also be blocked by its suppressant protein (each gene is suppressed by the protein with the same letter, e.g., A is blocked by a).

The implementation of the system in the classical approach is as follows.

```
new a@1.0:chan
new b@1.0:chan
new c@1.0:chan

let Gene(a:chan,b:chan)=
  do delay@0.1; (Protein(b) | Gene(a,b))
  or ?a; delay@0.0001; Gene(a,b)
and Protein(b:chan) =
  do !b; Protein(b)
  or delay@0.001
run ( Gene(a,b) | Gene(b,c) | Gene(c,a) )
```

One can see that the program actually contains three genes ($Gene(a, b)$, $Gene(b, c)$ e $Gene(c, a)$) that can synthesize a protein (by executing the `delay` statement) or can do an exchange with the suppressor, after which they enter in an idle state for a while (`delay@0.001`).

In this situation we can consider the following set of "reactions":

- $A \overset{0.1}{\rightarrow} A + b$, $B \overset{0.1}{\rightarrow} B + c$ and $C \overset{0.1}{\rightarrow} C + a$ represent the synthesis of the proteins;
- $a \overset{0.001}{\rightarrow} \tau$, $b \overset{0.001}{\rightarrow} \tau$ and $c \overset{0.001}{\rightarrow} \tau$ represent the decays of proteins;

- $A + a \overset{0.001}{\rightarrow} A' + a$, $B + b \overset{0.001}{\rightarrow} B' + b$ and $C + c \overset{0.001}{\rightarrow} C' + c$ represent the idle phase of gene X through a conversion in a temporary gene X';
- $A' \overset{0.0001}{\rightarrow} A$, $B' \overset{0.0001}{\rightarrow} B$ and $C' \overset{0.0001}{\rightarrow} C$ represent the waking up of the genes.

We can use our method to translate these reactions in a π-Molecule for the Repressilator. In Figure 4 we compare our results (on the left) with the traditional approach (on the right).

Fig. 4. Repressilator - SPiM

In both cases we find a periodic behavior. In particular, only one gene is active in a certain moment. The curves are different because of the stochastic mechanism that drives the systems.

3.4 Brusselator

The following example does not belong to the test suite of SPiM because it contains ternary reactions. The system is an abstraction of the well-known Belusov - Zhabotinsky (BZ) system, that appears in literature with the name of Brusselator.

The chemical reactions inside the Brusselator are the following: $X_1 \overset{c_1}{\rightarrow} Y_1$, $X_2 + Y_1 \overset{c_2}{\rightarrow} Y_2 + Z_1$, $2Y_1 + Y_2 \overset{c3}{\rightarrow} 3Y_1$ e $Y_1 \overset{c_4}{\rightarrow} Z_2$, and the overall result of the system is an oscillation of quantities of the species Y_1 and Y_2. Further details about Brusselator are in [2].

Here we show the π-Molecule which implements the system, and then the comparisons between our results and Gillespie's ones [12].

```
new mem:chan(float,float) (*memory content: Y1 Y2*)
new sinc:chan(float,float)


let PiM()=
 ?mem(y1,y2);
 if(y1>0.0)
  then (*ok 1,2,4*)
   if(y1>1.0*y2>0.0)
    then (*feasible: 3*)
     do (*1*)
      delay@5000.0;(!mem(y1+1.0,y2)|!sinc(y1+1.0,y2)|PiM())
     or (*2*)
      delay@50.0*y1;(!mem(y1-1.0,y2+1.0)|!sinc(y1-1.0,y2+1.0)|PiM())
     or (*3*)
      delay@ 0.00005*y1*(y1-1.0)*0.5*y2;
       (!mem(y1+1.0,y2-1.0)|!sinc(y1+1.0,y2-1.0)|PiM())
     or (*4*)
      delay@ 5.0*y1;(!mem(y1-1.0,y2)| !sinc(y1-1.0,y2)|PiM())
    else (*not feasible: 3*)
     do (*1*)
      delay@5000.0;(!mem(y1+1.0,y2)|!sinc(y1+1.0,y2)|PiM())
     or (*2*)
      delay@50.0*y1;(!mem(y1-1.0,y2+1.0)| !sinc(y1-1.0,y2+1.0)|PiM())
     or (*4*)
      delay@ 5.0*y1;(!mem(y1-1.0,y2)| !sinc(y1-1.0,y2)|PiM())
  else (*only 1 is feasible*)
   delay@5000.0;(!mem(y1+1.0,y2)|!sinc(y1+1.0,y2)|PiM())

let Printmem(n:int)=
 ?sinc(b,c);
 if(n>200)
  then
   print(show b);println(show c);Printmem(0)
  else
   Printmem(n+1)

run (!mem(1000.0,2000.0)|PiM()|Printmem(0))
```

In Figure 5 (Figure 6) we plot the time evolution of Y_1 (Y_2, respectively); Figure 7, instead, contains the phase portrait of both the two species. As always, we put our results on the left, while on the right you can find the results from Gillespie's paper. There are small differences between our results and Gillespie's ones (e.g., in our results Y_1 spends less time around the minimum) which are still under investigation.

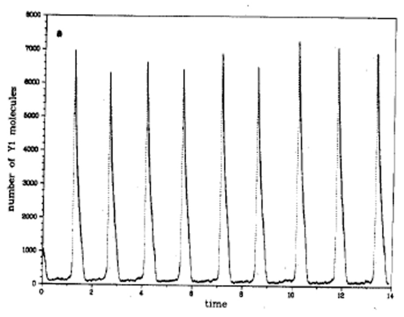

Fig. 5. Brusselator, Y_1

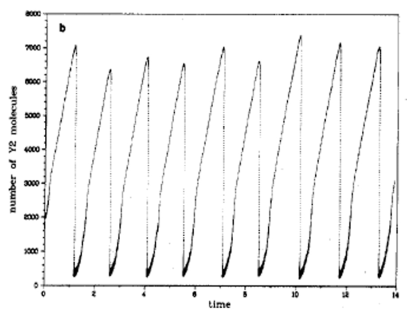

Fig. 6. Brusselator, Y_2

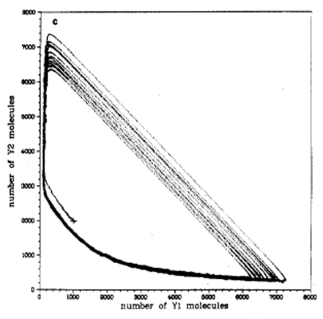

Fig. 7. Brusselator, Y_1 and Y_2

As already said it is not possible to encode and simulate the Brusselator in SPiM using the classical *molecule centric* approach, since it involves the ternary reaction $2Y_1 + Y_2 \overset{c3}{\rightarrow} 3Y_1$. We tried to implement a simplified version of the system (by using two consecutive reactions to simulate the ternary one), but it does not behave like the original one, and this is probably due to the nontrivial problem of assigning stochastic rates to the coupled reactions.

4 Conclusions

We described an alternative *reaction centric* use of Stochastic π-calculus.

Our idea is based on the use of passive reactants and active reactions: this choice injects an external point of view inside a typical internal point of view framework, but retains the structure of the language and its available theoretical tools. We got this result by translating chemical reactions in delay transactions, following the chemical meaning of the formers. The pure chemical based models (such as ODEs) are not usable in the context of formal analysis (e.g., model checking techniques). Our approach should allows us to get around this problem.

As a byproduct of our approach we notice that reactions involving more than two inputs can be directly encoded and simulated. This potential has been exploited in the Brusselator example.

Moreover, we have briefly discussed how our method can be used to model genetic regulatory networks on the Repressilator case study.

In the long period our main aim is that of providing a deeper understanding and integration between different formalisms. In particular, we are interested in translations from ODEs and hybrid automata into processes algebra. In this sense the external point of view seems necessary for the translation of the global constraints (invariant, activations, and resets). A continuos semantics in terms of ODEs has been provided for Biocham (see, e.g., [10]), while translations from sCCP programs into hybrid automata have been considered in [5].

For the moment we focused on Stochastic π-calculus, since its notion of channels seems to be useful to naturally implement the memory component. In a sense our results demonstrate the effectiveness of Stochastic π-calculus for the description of nontrivial biochemical systems, by showing that, even if the classical use of Stochastic π-calculus in systems biology is *molecule centric*, it is easy to use it from a *reaction centric* perspective. However, since we are interested in the integration of different formalisms, in the future we plan to consider also other languages. In particular, in this context the recent work done on Bio-PEPA (see, e.g., [8]) is very relevant: it allows to model different kind of reactions without limitations on the number of reactants; it has been used from both a *reactant centric* and a pathway centric point of view.

References

[1] Alur, R., C. Belta, F. Ivancic, V. Kumar, M. Mintz, G. J. Pappas, H. Rubin and J. Schug, *Hybrid Modeling and Simulation of Biomolecular Networks*, in: *Proc. of Hybrid Systems: Computation and Control (HSCC'01)*, LNCS **2034** (2001), pp. 19–32.

[2] Besozzi, D., G. Mauri, D. Pescini and C. Zandron, *Membrane Systems in Systems Biology*, in: *Proc. of Workshop of Discrete Event Systems (WODES08)* (2008), pp. 275–280.

[3] Bortolussi, L., *Stochastic Concurrent Constraint Programming*, Electronic Notes in Theoretical Computer Science **164** (2006), pp. 65–80.

[4] Bortolussi, L. and A. Policriti, *Stochastic Concurrent Constraint Programming and Differential Equations*, Electronic Notes in Theoretical Computer Science **190** (2007), pp. 27–42.

[5] Bortolussi, L. and A. Policriti, *Hybrid approximation of Stochastic Concurrent Constraint Programming*, in: *International Federation of Automatic Control World Congress, (IFAC'08)*, 2008, to appear.

[6] Cardelli, L., *On process rate semantics*, Theoretical Computer Science **391** (2008), pp. 190–215.

[7] Casagrande, A., C. Piazza and B. Mishra, *Semi-Algebraic Constant Reset Hybrid Automata - SACoRe*, in: *Proc. of the 44rd Conference on Decision and Control (CDC'05)* (2005), pp. 678–683.

[8] Ciocchetta, F. and J. Hillston, *Bio-pepa: a framework for the modelling and analysis of biological systems* (2008), Theoretical Computer Science.

[9] Del Tedesco, F. and C. Piazza, *External Control in Process Algebra for Systems Biology - Extendend Version*, available at `http://www.dimi.uniud.it/piazza/mecbic.pdf`.

[10] Fages, F. and S. Soliman, *Formal Cell Biology in BIOCHAM*, in: M. Bernardo, P. Degano and G. Zavattaro, editors, *Formal Methods for Computational Systems Biology, Int. School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM'08)*, LNCS **5016** (2008), pp. 54–80.

[11] Ghosh, R., A. Tiwari and C. Tomlin, *Automated Symbolic Reachability Analysis; with Application to Delta-Notch Signaling Automata*, in: O. Maler and A. Pnueli, editors, *Proc. of Hybrid Systems: Computation and Control (HSCC'03)*, LNCS **2623** (2003), pp. 233–248.

[12] Gillespie, D., *Exact Stochastic Simulation of Coupled Chemical Reactions*, Journal of Physical Chemistry **81** (1977), pp. 2340–2361.

[13] Phillips, A. and L. Cardelli, *Efficient, Correct Simulation of Biological Processes in the Stochastic Pi-calculus*, in: M. Calder and S. Gilmore, editors, *Proc. of Int. Conf. on Computational Methods in Systems Biology (CMSB'07)*, LNCS **4695** (2007), pp. 184–199.

[14] Phillips, A., L. Cardelli and G. Castagna, *A Graphical Representation for Biological Processes in the Stochastic pi-Calculus* (2006), pp. 123–152.

[15] Priami, C., *Stochastic pi-Calculus*, The Computer Journal **38** (1995), pp. 578–589.

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

[17] Voit, E. O., "Computational Analysis of Biochemical Systems. A Pratical Guide for Biochemists and Molecular Biologists," Cambridge University Press, 2000.