Connecting process algebras and differential equations for Systems Biology

Luca Bortolussi* Alberto Policriti†

Abstract

We discuss two formalism widely used for modeling biological systems, i.e. stochastic process algebras and differential equations. The focus is in understanding relationships intervening between the two mathematical theories, discussing existing work, and suggesting methods to translate one formalism into the other.

1 Introduction

Computational systems biology is a very active research area, where many different modeling techniques are used to study and analyze a broad range of biological networks, cf. [1, 2]. In this paper we focus on two different modeling approaches, one based on stochastic process algebras (SPA) and the other using ordinary differential equations (ODE). Both are well established techniques that have been used extensively to model biochemical reactions [2, 3] and also biological networks at a higher level of abstraction, like genetic regulatory networks [4]. These two approaches, however, are radically different: the models built using SPA are discrete and stochastic, and they are generally used to infer pattern of dynamical evolution of the system. On the other hand, ODE's provide a continuous and deterministic model and can be used to formulate quantitative predictions.

Another fundamental difference between SPA and ODE's is the point of view taken while modeling a system. With process algebras, in fact, one has to describe the logical patterns of interaction between the different entities: a process is associated to each biological entity and the interaction capabilities are represented as communication links among processes. Moreover, modifications of biological entities due to interaction are mimicked by state changes of the processes associated to the corresponding entities. Quantitative information is introduced in the form of *communication rates*, encoding the speed at which each communication can take place. This approach is "internal", in the sense that it describes the system with an inner perspective, modeling each single entity and composing them together to obtain the global model. On the contrary,

*Dept. of Mathematics and Informatics University of Udine, Udine, Italy
Email: bortolussi@dimi.uniud.it

†Dept. of Mathematics and Informatics University of Udine, Udine, Italy
Email: policriti@dimi.uniud.it

ODE's take an external approach, in which what is actually modeled are the mutual relationships among variables expressing concentrations of substances. This corresponds to the point of view of an experimenter observing the temporal evolution of the quantities it is interested in, and relating such quantities through mathematical laws.

The aim of our work is to try to shed some light on the relationships intervening between these two approaches. There are two different levels where this relationship unwinds: one is concerned purely with mathematical connection between the two formalism, whilst the other takes into account also the phenomenon under study.

Considering the first level, the goal is to define procedures capable of converting one formalism into the other. Clearly, here there are two directions: associating ODE systems to SPA programs and SPA programs to ODEs systems. In both cases the translation should preserve the behaviour of the system it applies to, where for behaviour we mean the time trace for ODEs and the average evolution for stochastic systems. There are a lot of subtleties involved in this conversion: first of all, approximating a discrete variable with a continuous one is not always meaningful, especially when the former takes small values. In addition, the effect of stochastic fluctuations cannot always be neglected, as it can dominate the behaviour of the stochastic system: a classical example is the Lotka-Volterra chemical system presented in [5], where the system exhibits strong oscillatory behaviour, despite having a constant average value. If this is the case, the stochastic ingredient cannot be safely forgot; hence the need for defining clearly the conditions under which translating one formalism into the other is meaningful.

Motivations for this analysis are manyfold. First of all, there is the philosophical issue concerning the relationship between the internal description of SPA and the external one of ODE's. In some sense, this relationship is the model's counterpart of the connection between what is going on in a system and what is seen by an observer. Moreover, associating ODEs to SPA can lead to powerful analysis techniques that escape the state space explosion problem (cf. [6]). The other direction, instead, can clarify the behavioral expressivity of SPA models, showing what analytical behaviors (e.g. chaos) of ODEs systems can be captured by them. Finally, the determination of parameters may benefit from this analysis.

These two directions in the translation can be considered and validated also in the light a particular system to be modeled. In particular, given a SPA program and ODEs that model the same phenomenon, we may ask if they are equivalent w.r.t. the translation procedure and if their translation captures correctly the observed features of the real system. As we will argue in Section 6, considering the problem from this point of view may lead to the definition of different translation methods, aiming to be invariant w.r.t. the observed behaviour.

Several SPA have been used for biological modeling (see, for instance, [3]), but here we focus on two of them only. The first one is stochastic π-calculus [7], probably the most broadly used SPA for biological purposes [3, 8]. The second one, instead, is a stochastic version of Concurrent Constraint Programming [9] (CCP), called sCCP [10]. In the following we will show that this process algebra can be used for biological modeling purposes in a flexible way, similarly to π-calculus, and, moreover, we will strongly exploit one of its features: the

admissibility of non-constant stochastic rates (a feature already present in [11]). In particular, while discussing the mapping from ODE's into SPA in Section 5, we will show that having "variable" rates is not just syntactic sugar but it is the crucial ingredient to realize this translation. In other words, non-constant rates give the *right* burst to the expressivity of the language.

Similarly, there are several formats of ODE's used for modeling biochemical reactions and other biological networks; among them we recall Mass Action equations (sometimes referred also as Reaction Rate Equations), Michaelis-Menten Rate equations, S-Systems, Generalized Mass Action equations, Hill's equations. We refer to [2] for an overview. The class of ODE's chosen for our discussion is that of S-Systems [2], that will be described in more detail in Section 2.2. The reasons behind this choice are: the expressiveness of S-Systems in terms of analytical behaviors, the fact that all their parameters have a clear biological meaning, and the fact that they are good approximating functions, allowing parameters to be trained using biological data only and without previous knowledge on any kinetic rate.

A method for associating a set of differential equations to the stochastic process algebra (PEPA [12]) has been put forward by Hillston in [6]. The same method can be applied also to π-calculus (see Section 4), producing a set of Mass Action equations as a result. These equations relate in some way to the average value (over all possible traces) of variables under interest, approximated continuously. The relation between PEPA models and the corresponding ODEs is similar to the relation between stochastic models of chemical systems and the corresponding Mass Action equation models. In this case, it is well known that exact equality between average value of the stochastic system and the ODE system holds in few special cases [5, 13, 14], yet the approximation works well in many cases. In some cases, however, the difference is consistent. This phenomenon occurs also for process algebras, and in Section 6 we provide a simple but symptomatic example: a simple oscillatory genetic network called the repressilator. Specifically, we will show that, while the π-calculus model of the system exhibits an oscillating behavior when simulated, the corresponding equations associated to it by Hillston's method quickly converge to a stable value close to zero. However, this is coherent with the average behavior of the system (cf. Section 6).

These considerations suggest that the behavior demonstrated by stochastic fluctuations cannot always be averaged away and that a translation from SPA to ODE's must take into account these effects, if it has to be behaviorally invariant with respect to the modeled system (the repressilator is expected to oscillate). The approximation property of S-Systems suggests a possible solution: we can use a set of traces produced by runs of simulations of SPA programs to train the parameters of the corresponding S-System's set of equations.

Also the passage from ODE's to SPA offers interesting problems. Specifically, we would like to see (1) if all the analytical behaviors of S-Systems are reproducible by stochastic simulations of SPA programs and (2) what are the relationships between the time evolution of concentrations and the logical description of interactions. In particular, (2) asks if and how we can deduce, starting from a set of observations, the logical structure of processes and interactions describing the systems, i.e. the fine topology of the biological network under exam.

In this direction, the use of sCCP becomes fundamental. In particular,

$Program = D.P$ $D \ ::= \ \varepsilon \mid D.D \mid X \ ::= \ P$ $\pi \ ::= \ \overline{x}y \mid x(z) \mid \tau$ $P \ ::= \ M \mid P \parallel P' \mid X$ $M \ ::= \ \mathbf{0} \mid \pi.P \mid M + M'$	$Program = D.A$ $D = \varepsilon \mid D.D \mid p(\mathbf{x}) : -A$ $\pi = \mathrm{tell}_\lambda(c) \mid \mathrm{ask}_\lambda(c)$ $M = \pi.A \mid M + M'$ $A = \mathbf{0} \mid [p(\mathbf{x})]_\lambda \mid M \mid \exists_x A \mid (A_1 \parallel A_2)$

Table 1: Syntax of restricted π-calculus (**left**) and of sCCP (**right**).

having at our disposal non-constant rates allows to define a simple automatic translation from S-Systems to sCCP programs, that exhibit the same behaviors and are coherent from a quantitative point of view. This translation suggests that a positive answer to (1) exists only if we extend usual process algebras with the non-trivial ingredient of functional rates.

The paper is organized as follows: in Section 2.1 we introduce a restricted version of stochastic π-calculus discussing, in the meanwhile, the SPA modeling approach. In Section 2.2 we introduce S-Systems. sCCP is presented in Section 3, while Hillston's method is introduced in Section 4. In Sections 6 and 5 we present the main contributions of the paper, analyzing in detail possible mechanisms for translating SPA to ODE's and viceversa. Finally, in Section 7 we draw some conclusions and outline our plan for future work.

2 Preliminaries

In this section we briefly describe the syntax of the two main modeling tools we are comparing: π-calculus from the process algebra side, and S-Systems as a differential equations format.

2.1 A paradigmatic Process Algebra: π-calculus

Stochastic π-calculus [7] has been used as a modeling technique for biochemical reactions in [3], where the interesting idea of a parallel between biological molecules that interact and agents that communicate has been exploited. Since then, several biological systems have been modeled [8, 4], showing the feasibility of the technique. One of its main appeals resides in the compositional nature of the modeling process, allowing to describe single biological entities with a concern on of the logic of their interactions only, while complex behaviors emerge as a system's property of the entire model. However, the main problem of modeling with π-calculus resides in the absence of any quantitative information, apart from stochastic rates, which, on the other hand, are parameters left to the user for determination.

In the following we introduce the syntax of a simplified version of π-calculus, where we forget some operators like restriction and the creation of fresh channel names. This choice is mainly determined by reasons that will become apparent in Section 4, where we discuss the approach developed by Hillston [6], associating a set of differential equations to a stochastic process algebra program. The

restricted π-calculus syntax used in this paper is stated in Table 1. The two basic actions are those devoted to send and receive names along a channel. In addition, we have a choice operator, allowing to select one among different alternatives, and a parallel composition, enabling agents to be executed together. Recursion is achieved through the use of constants within the syntactic specification of agents. We recall that the semantic model induced by operational semantics rules is a Continuous Time Markov Chain (CTMC) [15], where in every state we essentially have a race condition between all active communications.

At this point we introduce the repressilator, an artificial biological clock that will be used as the main example for comparing the different formalisms throughout the paper. Repressilator is a biochemical clock composed of three genes expressing three different proteins, **tetR**, λ**cI**, **LacI**, that have a regulatory function in each other gene expression. In particular, protein **tetR** inhibits the expression of protein λ**cI**, while protein λ**cI** represses the gene producing protein **LacI** and, finally, protein **LacI** is a repressor for protein **tetR**. The expected behavior is an oscillation of the concentrations of the tree proteins with a constant frequency.

In π-calculus, the model introduces an agent for each gene (called *neg gene gate* as it negatively regulates the next gene in the chain), and some agents for the proteins, cf. [4]. The number of these agents, considered in a single run of the simulation, is the quantitative counterpart in π-calculus of the protein's concentration. The subset of π-calculus presented here suffices for modeling the repressilator and the code is shown in Table 2 using a compact representation with parameter passing. We can see that the act of repression is represented by a communication between the protein and the gene gates, while the production of a protein is a silent (τ) action of the gene agent. A trace of a simulation of the model, provided using SPiM [16], is shown in Figure 1. The oscillating behavior is manifest.

2.2 S-Systems

Modeling biological systems by means of ordinary differential equations (ODE) means to identify a law representing the evolution over time of the concentration of each of the biological entities under study. This approach is radically different from the process algebraic one, as here we have a continuous, quantitative, and deterministic evolution of the flow of substances, as opposed to the discrete, qualitative, and stochastic evolution typical of process algebras.

In the literature there are different formats of ODE's used for modeling biological systems: mass action equations, generalized mass action (GMA) equations, Hill's equations, and S-systems just to cite a few. We refer to [2] for further details.

S-System equations (which are a special case of GMA equations) describe the evolution of a set of dependent variables $(X_1, \ldots, X_n)$, by taking into account also a set of independent variables $(X_{n+1}, \ldots, X_{n+m})$, representing quantities associated to the substrate and fixed by the experimenter. The general format for these equations is

$$\dot{X}_i = V_i^+(X_1, \ldots, X_{n+m}) - V_i^-(X_1, \ldots, X_{n+m}),$$

i.e. the speed of change of quantity X_i is determined by a production term (V_i^+) and a degradation term (V_i^-). Each V term contains all the variables directly

influencing the behavior of X_i and it has a particular format:

$$V(X_1, \ldots, X_{n+m}) = \alpha \prod_{j=1}^{n+m} X_j^{h_j},$$

where α and h_j are parameters to be determined. The general format for an S-System of differential equations is therefore:

$$\dot{X}_i = \alpha_i \prod_{j=1}^{n+m} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n+m} X_j^{h_{ij}}. \tag{1}$$

The parameters $\alpha_i, \beta_i \geq 0$ are called the *rate constants* and represent the basic production and degradation rates for each dependent variable. The parameters g_{ij} and h_{ij} are called kinetic orders and they represent the (kinetic) strength of the interaction of the biological entity X_j in the production or degradation of X_i. They can be positive (enhancing effect), negative (inhibition effect), or zero (no effect).

The main virtues of S-System reside in their analytical tractability (for certain specific properties, cf. [2]) and in the fact that all parameters involved have a clear biological meaning. Moreover, they can represent a huge set of behaviors, despite the rigidity of the analytical format. the reason for this is that each of the terms $V(X_1, \ldots, X_{n+m})$ is a first order power law approximation of a smooth function, thus S-systems can be seen as first-order power law approximations of general ODEs.

Going back to the repressilator, a possible model written in the S-System equations formalism is the following:

$$\begin{array}{lll} \dot{X}_1 = \alpha_1 X_3^{-1} - \beta_1 X_1^{0.5}, & \alpha_1 = 0.2, & \beta_1 = 1, \\ \dot{X}_2 = \alpha_2 X_1^{-1} - \beta_2 X_2^{0.578151}, & \alpha_2 = 0.2, & \beta_2 = 1, \\ \dot{X}_3 = \alpha_3 X_2^{-1} - \beta_3 X_3^{0.5}, & \alpha_3 = 0.2, & \beta_3 = 1. \end{array} \tag{2}$$

Here X_1, X_2, X_3 represent the concentration of the three proteins **tetR**, **λcI**, and **LacI**, while X_4, X_5, X_6 are three independent variables connected with the basic speed of production (the equivalent of gene gates in the π-calculus model). A numerical solution of these equations is shown in Figure 1.

We can see the dependence with a negative exponent from the repressor in the production term in each of the three equations and, moreover, we can see a strange exponent in the degradation term in the equation for X_2. This is due to an high sensitivity of these equations, both on initial conditions and on parameters: a slight change in that value destroys the oscillating behavior. This is in contrast with the model in π-calculus, whose oscillating behavior is pretty robust, cf. [4]. More comments on this sensitivity are postponed to Section 5.

3 Stochastic Concurrent Constraint Programming

In this section we briefly present a stochastic version [10] of Concurrent Constraint Programming [9], which will be used in the following in exploring further the relationships between process algebras and differential equations.

Concurrent Constraint Programming (CCP) is a process algebra having two distinct entities: agents and constraints. Constraints are interpreted first-order logical formulae, stating relationships among variables (e.g. $X = 10$ or $X + Y < 7$), and can be used to compute [17]. For instance, if the language contains the sum and product functions we can compute all polynomials; in general we can compute all usual analytical functions. Agents in CCP, instead, have the capability of adding constraints (`tell`) into a "container" (the *constraint store*) and checking if certain relations are entailed by the current configuration of the constraint store (`ask`). The communication mechanism between agent is therefore asynchronous, as information is exchanged through global variables. In addition to `ask` and `tell`, the language has all the basic constructs of process algebras: non-deterministic choice, parallel composition, procedure call, plus the declaration of local variables.

The stochastic version of CCP (sCCP [10]) is obtained by adding rates to all instruction interaction with the constraint store $\mathcal{C}$, i.e. `ask`, `tell`, and procedure call. Those rates, however, are not real numbers, but rather functions associating a real number to each configuration of the constraint store: $\lambda : \mathcal{C} \to \mathbb{R}^+$. This will be an important feature for the application of sCCP we have in mind, cf. Section 5. The syntax of sCCP can be found in Table 1.

The underlying semantic model of the language (defined via structural operational semantic, cf. [10]) is still a CTMC, as each configuration of the system in sCCP consists of the current set of processes and of the current configuration of the constraint store. Thus in every node of the transition graph all rate functions are evaluated. Therefore, as in stochastic π-calculus, we have a race condition between all active instructions and the fastest one is executed.

sCCP can be used for modeling biological processes in a way similar to π-calculus. The main difference is that instead of approximating the concentration of a biological molecule with a number of copies of the corresponding process, the concentration is directly represented as a (discretized) variable in the constraint store. If this variable is interpreted over integers we reproduce the same behavior of π-calculus, while if it varies over a grid of rationals (for instance, floating point numbers), we have the possibility of approximating continuous data.[1]

We reconsider here the repressilator's example and show one model for it in sCCP. The code is listed in Table 3. The aim of this model is to *reproduce the same stochastic behavior exhibited by the π-calculus description* of Section 2.1, in order to compare the two formalisms. The first point to underline is that the quantity (i.e. number of molecules) of the three proteins **tetR**, **λcI**, and **LacI** is stored in variables of the constraint store. Therefore, production and degradation mechanisms act on these variables modifying their values (cf. caption of Table 3). Production is managed by gene gate agents, which can produce a unit of protein or enter in a repressed state; degradation is realized in parallel by dedicated agents. Note that communication between agents is performed asynchronously by the modification of these variables. Another crucial difference with π-calculus resides in rates: here are no more necessarily constant but rather functions. Specifically, in this model of the repressilator, we use *rate functions expressing the way global rates of communication are computed*

[1] Computation in CCP is monotonic [9], meaning that once a variable is constrained to a value it will keep that value forever. To have variables changing values over time we have to represent them as growing lists of ground terms with an unbounded tail. We refer to these variables as *stream variables*.

$$\text{neg(a,b)} ::= \tau_{\lambda_P}.(\text{tr(b)}|\text{neg(a,b)}) + \text{a}.\tau_{\lambda_U}.\text{neg(a,b)} \qquad \text{tr(x)} ::= \overline{\text{x}}.\text{tr(x)} + \tau_{\lambda_D}$$
$$\text{repressilator} ::= \text{neg(a,b)}|\text{neg(b,c)}|\text{neg(c,a)}$$

Table 2: π-calculus code for the repressilator. Neg gate neg(a,b) can spawn a new copy of protein b with rate λ_P, or be inhibited as the result of a communication on the repressor channel a, having rate λ_I. A repressed gene gate returns to normal state after a delay with rate λ_U. On the other hand, a protein tr(x) can perform a communication in its inhibition channel x, or be degraded at rate λ_D. Further details about gene gates can be found in [4].

$$\text{neg}(X,Y) \text{ :- } (\text{tell}_{[\lambda_P]}(X = X+1) + \text{ask}_{[\lambda_I \cdot Y]}(Y > 0).\text{ask}_{[\lambda_U]}(\text{true})).\text{neg}(X,Y)$$
$$\text{degr}(X) \text{ :- } \text{tell}_{[\lambda_D \cdot X]}(X = X-1).\text{degr}(X) \qquad \text{neg\_gate}(X,Y) \text{ :- } \text{neg}(X,Y) \parallel \text{degr}(X)$$
$$\text{repressilator :- neg\_gate}(A,C) \parallel \text{neg\_gate}(B,A) \parallel \text{neg\_gate}(C,B)$$

Table 3: Code of the repressilator in sCCP. The neg gate $\text{neg}(X,Y)$ is described by an agent that can produce one unit of protein X with constant rate $[\lambda_P]$, posting (tell) the constraint $X = X+1$, or enter in a repressed state. Repression is achieved by an ask instruction checking if there are repressor molecules ($Y > 0$); the rate of this instruction is $[\lambda_I \cdot Y]$, depending linearly on the repressor Y. Note that the condition $Y > 0$ forces this rate to be greater than zero. A repressed gate is converted back to its normal state after an unconditional time delay ($\text{ask}(true)$) with rate $[\lambda_U]$. Degradation of proteins is represented by a dedicated agent $\text{degr}(X)$, decreasing the value of X ($\text{tell}(X = X-1)$) with rate $[\lambda_D \cdot X]$, proportional to X.

in π-calculus: for instance, the degradation rate of protein A (**tetR**) has rate $r_D(A) = \lambda_D A$, depending linearly on A (A is a variable storing the number of molecules), while the inhibition rate of gene A is $r_I(C) = \lambda_I C$, depending on the concentration of the repressor C (**LacI**). Comparing the π-calculus model of the repressilator with sCCP one, it is straightforward to see that in each state we have the same actions that can happen (production of one unit of a protein, degradation, inhibition or disinhibition) and, moreover, with the same overall rate. Therefore, we expect to see the same stochastic behavior in the two systems, and this intuition is confirmed experimentally, cf. Figure 1.

4 A possible translation of process algebras into differential equations

In this section we describe the work of Hillston [18, 6], presenting a technique to associate a set of differential equations to stochastic process algebra programs. In [6] the method is described for PEPA [12], a stochastic process algebra with synchronization among n processes over shared actions. However, we describe the method for the restricted version of π-calculus presented in Section 2.1, as this is a straightforward adaption that goes more directly to our point.

Given a program in (restricted) π-calculus we consider the set of all different sequential syntactical terms that can appear. With "sequential term" we mean every agent that is not a parallel composition; i.e it is a summation as defined

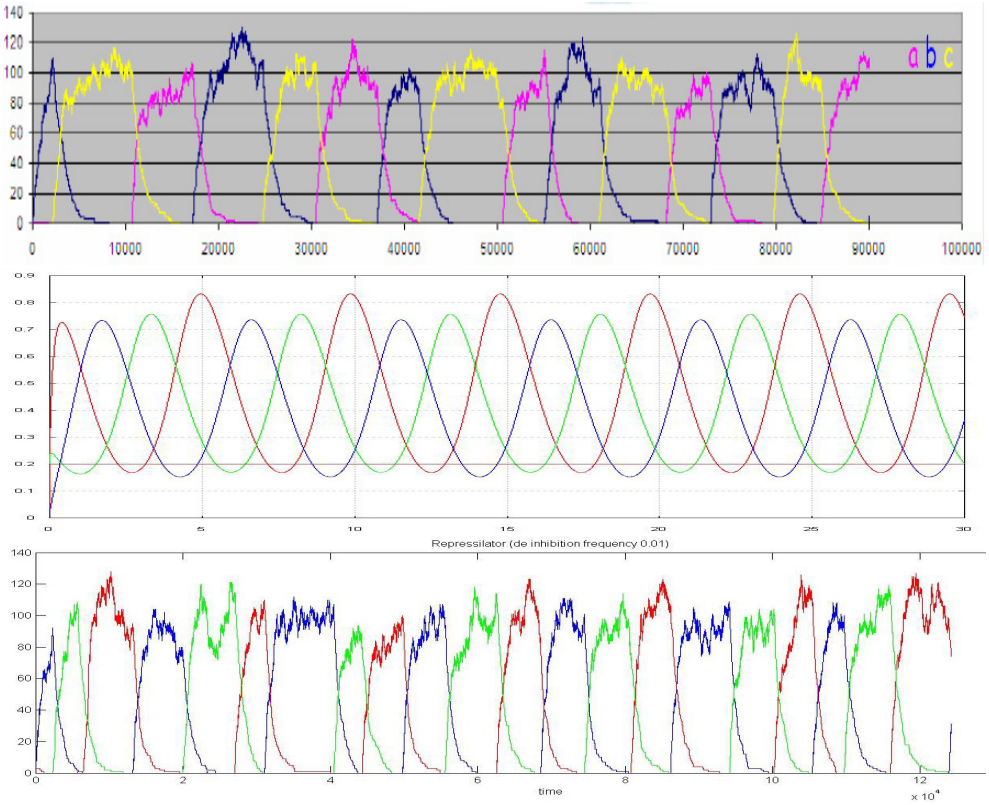

Figure 1: (**top**) Evolution of π-calculus model of repressilator, taken from [4]; the code is shown in Table 2. Numerical value of rates, whose meaning is defined in the caption of Table 2, are $\lambda_P = 1$, $\lambda_I = 1$, $\lambda_U = 0.0001$ and $\lambda_D = 0.01$ (**middle**) Numerical solution of S-System's equations (2) for repressilator. (**bottom**) An execution trace for repressilator in sCCP described in Table 3; numerical value of rates is the same as π-calculus' ones.

in Table 1. Each of these terms will be associated to a component of a vector of integers, storing the number of different copies of that term present in a particular configuration. This is, in fact, a compact way of representing the different states of the π-calculus program. For example, consider the terms $P_1 ::= \tau_r.P_2$ and $P_2 ::= \tau_r.P_1$, and the following program: $P_1 \parallel P_1 \parallel P_1 \parallel P_2 \parallel P_2 \parallel P_2$. The only two syntactical terms of this program are P_1 and P_2, thus we associate to it a vector with two components, the first representing the number of instances of P_1 and the second the number of copies of P_2. Therefore, the initial configuration of this simple program is represented by the vector $(3, 3)$. Note that these vector con be constructed because in the restricted version of π-calculus each program has always a finite number of different syntactical terms, as new channels cannot be generated. In addition, the absence of the restriction operator guarantees that we do not have to distinguish between copies of a syntactical term that are inside or outside the scope of a restriction. These facts imply that these vectors correctly represent each state of the system.

Once we have this vector representation, we need to consider all transitions that can take place and the modifications in the components of the vector they induce. If we focus on a single term P there are transitions that, once executed, create new copies of P. We call these *enter transitions*, while other transitions, named *exit transitions*, remove occurrences of P. For instance, in the example above, the transition $P_1 \longrightarrow_\tau P_2$ is an exit transition for P_1 and an enter transition for P_2. The idea for the construction of ODE's is to approximate the discrete variation in the number of instances of syntactical terms with a continuous variable. This is meaningful especially when the number of copies of a term is high. The rate of change for the variable corresponding to a term is given by the sum of the rates of all enter transition into it minus the sum of the rates of all exit transition from it. To automatically generate ODE's we construct a bipartite graph, called the *action graph* in [6], with two sets of nodes, one corresponding to terms and the other to channel names. We have an edge from the node associated to term P to a node for name a if there is an exit transition out of P involving channel a. Similarly, incoming edges in P represent enter transitions. Each edge is labeled with the terms involved in the transition (or with their corresponding position in the vector), taking into account the correct multiplicity of send and receive instruction on channel a.[2] Moreover, in case of enter transitions, we store also the number of copies of the term generated by that transition.[3] The enter rate for term P, with associated variable X_i, is obtained by summing for all enter transitions the product of the rate of the channel, times the variables associated to terms labeling the edge, times their multiplicity, times the number of copies spawned by the transitions. Exit rates are computed similarly. The equations for the toy example involving terms P_1 and P_2 are:

$$\dot{X}_1 = rX_2 - rX_1$$
$$\dot{X}_2 = rX_1 - rX_2$$

Note that this mechanism of associating ODE's to π-calculus can be extended also to sCCP, at least in the case where we put in the store only variables changing value over time. There is a remarkable difference in that the vector representing a configuration will have to store both variables associated to syntactical terms and variables of the constraint store.

5 From Differential Equations to Process Algebras

In this section we discuss the inverse problem of passing from differential equations to process algebra programs. This situation is the following: we start with a set of differential equations, possibly the model of a biological system, and we want to find a process in a suitable process algebra capable of reproducing the same behaviors, at least from a qualitative point of view.

This problem looks complicated, especially if we start from S-Systems or similar formats. In S-Systems the logic underlying the interactions among biological entities is partly hidden in the numerical values of the exponents of variables. On the other hand, in the transformation from process algebras to

[2] This is necessary for summations of the form $\overline{a} + \overline{a} + \ldots + \overline{a}$ or $a + a + \ldots + a$

[3] This is necessary if we have an enter transition into P from a term like $\tau.(P \parallel \ldots \parallel P)$.

differential equations put forward by Hillston, we expect to find several control variables, somehow encapsulating the logical structure of interactions. In addition, as we said, the format of the resulting equations is the one of mass action ODE's; this implies that, in general, the formalism is less expressive than S-Systems [2].

Basically, this means that we need a target process algebraic formalism that is more expressive than π-calculus, in order to have any hope to achieve this goal. In fact, if we could associate a π-calculus process with a similar behavior to each S-System, then we would have a correspondence between S-Systems and mass action equations simply associating to the π-calculus process the corresponding mass action system.

Analyzing the method of [6] in more detail, we can observe that the form of the terms in a mass action equation corresponds to the way of composing basic rates of communication in π-calculus. For instance, if we have n_1 copies of a process P_1, and n_2 copies of a process P_2 ready to communicate on a channel a with rate λ, then the overall communication rate of P_1 and P_2 is $\lambda n_1 n_2$, corresponding to a term in the ODE's of the form $\lambda X_1 X_2$. Therefore, this limitation of π-calculus seems to depend on the rigidity of composition of rates.

In sCCP, instead, we do have arbitrary rates as they are general functions from the constraint store to positive real numbers. We claim that this difference is able to bridge the gap of expressivity.

In the following we present a general method to map S-System equations into sCCP programs. The general format of an S-System equation is

$$\dot{X}_i = V_i^+(X_1, \ldots, X_{n+m}) - V_i^-(X_1, \ldots, X_{n+m}),$$

where V_i^+ is the term expressing the speed of production and V_i^- represents the speed of degradation. Since in stochastic process algebras the rate associated to an action is interpreted as the speed of execution, we can use V_i^+ and V_i^- as rate functions to produce and degrade the quantity X_i.

In more detail, consider the system of equations (1), where $X_1, \ldots, X_n$ are the dependent variables and $X_{n+1}, \ldots, X_{n+m}$ are the independent ones. We associate a variable in the constraint store to each dependent variable of the system of equations, i.e. to $X_1, \ldots, X_n$, while the independent ones are fixed by posting the appropriate constraints (for instance, if the value of X_k equals 0.2, we post $X_k = 0.2$). These variables take values over a discretized set of real numbers; for instance, fixing a precision of three digits, we can let them vary over the set $\{n + \frac{m}{1000} \mid n, m \in \mathbb{N}\}$. We indicate with σ the basic precision, e.g. $\sigma = \frac{1}{1000}$ in the example above. Finally, we associate an agent $\mathrm{subst}(X_i)$ to each equation of the system, defined as:

$$\left(\mathrm{tell}(X_i = X_i + \sigma)_{[\alpha_i \prod_{j=1}^{n+m} X_j^{g_{ij}}]} + \mathrm{tell}(X_i = X_i - \sigma)_{[\beta_i \prod_{j=1}^{n+m} X_j^{h_{ij}}]}\right).\mathrm{subst}(X_i).$$

Such an agent produces one "basic unit" of X_i (for instance, is adds $\frac{1}{1000}$ to the current value of X_i) with rate $\alpha_i \prod_{j=1}^{n+m} X_j^{g_{ij}}$ and removes such a unit (subtracting $\frac{1}{1000}$) with rate $\beta_i \prod_{j=1}^{n+m} X_j^{h_{ij}}$. All these agents are then put in parallel to generate the desired sCCP program.

Before giving an example of the method, we observe that with this transformation we are exploiting the versatility introduced by non-constant rates to

hide there the logical interaction mechanism that is usually modeled explicitly in process algebras. Moreover, if we apply Hillston method to the sCCP program associated to an S-System, following the extension suggested in Section 4, we will obtain exactly the original system of equations.

Let's consider again the S-System model for repressilator, with a slight change in parameters (the strange degradation exponent for the equation of X_2 is replaced with 0.5), with respect to the one shown in equation (2):

$$\begin{array}{ll} \dot{X}_1 = \alpha_1 X_3^{-1} - \beta_1 X_1^{0.5}, & \alpha_1 = 0.2 \\ \dot{X}_2 = \alpha_2 X_1^{-1} - \beta_2 X_2^{0.5}, & \alpha_2 = 0.2 \\ \dot{X}_3 = \alpha_3 X_2^{-1} - \beta_3 X_3^{0.5}, & \alpha_3 = 0.2. \end{array} \tag{3}$$

This slight modification, however, has a severe impact in the behavior of the system, that for some values of β does not oscillate any longer (see Figure 2). The corresponding sCCP process is

$$\text{subst}(X_1) \parallel \text{subst}(X_2) \parallel \text{subst}(X_3), \tag{4}$$

where

$$\begin{array}{l} \text{subst}(X_1) ::= \big(\text{tell}(X_1 = X_1 + 1)_{[\alpha_1 X_3^{-1}]} + \text{tell}(X_1 = X_1 - 1)_{[\beta_1 X_1^{0.5}]}\big).\text{subst}(X_1) \\ \text{subst}(X_2) ::= \big(\text{tell}(X_2 = X_2 + 1)_{[\alpha_2 X_1^{-1}]} + \text{tell}(X_2 = X_2 - 1)_{[\beta_2 X_2^{0.5}]}\big).\text{subst}(X_2) \\ \text{subst}(X_3) ::= \big(\text{tell}(X_3 = X_3 + 1)_{[\alpha_3 X_2^{-1}]} + \text{tell}(X_3 = X_3 - 1)_{[\beta_3 X_3^{0.5}]}\big).\text{subst}(X_3) \end{array}$$

In Figure 2 we compare numerical solutions of S-Systems, for different values of β (αs have the value defined in equation(3)), and numerical simulations of the corresponding sCCP process. As we can see, not only the general qualitative behavior is respected, but also the quantitative information about concentrations is preserved. Note that also "wild" behaviors of S-Systems are perfectly reproduced, see again Figure 2. Probably, one of the main ingredients guaranteeing this reproducibility is the fact that variables take values in a finer grid than integers, meaning that the effect of stochastic fluctuations is less remarkable, as they relative magnitude is smaller. This is essentially the same as working with a sufficiently high number of molecules in Gillespie's algorithm [5, 14].

6 From Process Algebras to Differential Equations

Stochastic simulation and deterministic description of chemical reactions via mass action equations are not equivalent. This fact is well known in computational chemistry, where the relationship between the average behavior of a stochastic description (analyzed by means of the stochastic master equation) and the solution of mass action equations has been extensively studied, cf. [5, 13, 14]. Specifically, these two descriptions coincide only in the thermodynamic limit of molecular population's sizes and volumes going to infinity with constant ratio; usually the deterministic equations are only an an *approximate description* of the average stochastic behavior. Moreover, in [5, 19], the author suggested that sometimes, in stochastic simulations, there is a discrepancy between the average value of a quantity and its temporal evolution observed in a run of the simulation algorithm. This is the case, for instance, for the Lotka-Volterra chemical

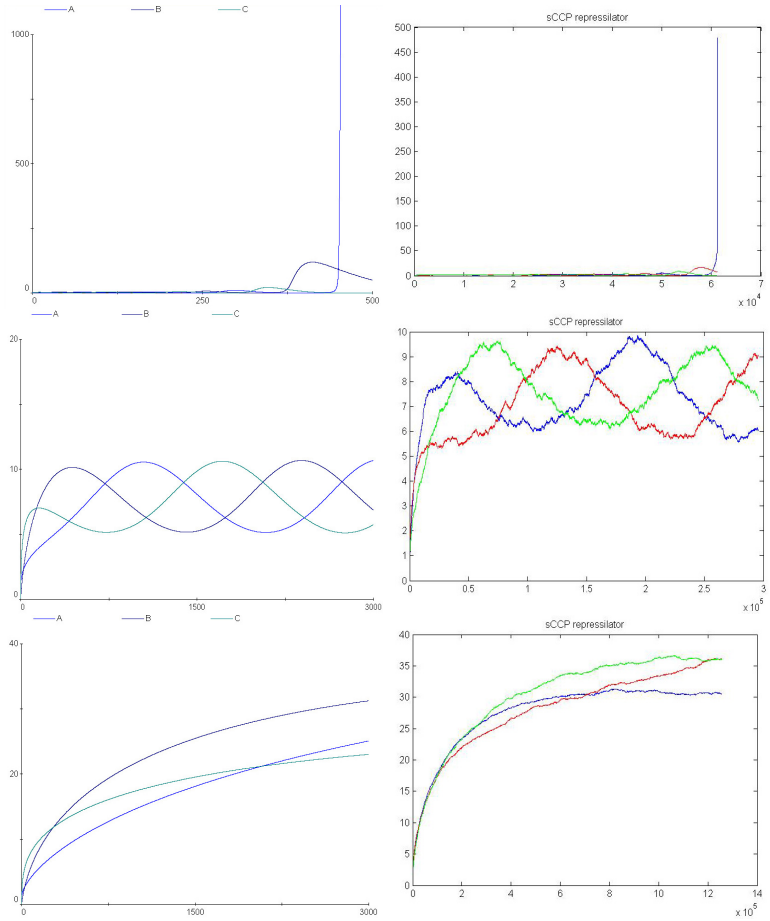

Figure 2: Numerical solutions of S-Systems' equations for the repressilator (left column), and the numerical simulation of the corresponding sCCP program (right column), for $\beta = 0.1$ (top), $\beta = 0.01$ (middle), and $\beta = 0.001$ (bottom)

system, whose average behavior (starting from specific initial conditions) is predicted to be constant; nevertheless, the traces of simulations oscillate far away from this constant value. This means that stochastic fluctuations cannot always be safely neglected, as there are cases where they rule the dynamical evolution of the system.

In the following we show that this strange behavior happens also when associating ODEs to SPA programs with Hillston's method (see Section 4). A paradigmatic example is the repressilator system. The first step to observe this phenomenon is to associate to the repressilator system, described in π-calculus (see Table 2), the set of ODE's that are the outcome of the Hillston's method. The equations are shown in Table 4. We have 9 equations corresponding to 9 dependent variables. Three variables (X_1, X_2, X_3) express the number of proteins **tetR**, λ**cI**, **LacI**, while the other 6 variables are connected with gene gates: Y_1, Y_2, Y_3 represent the normal status of the gene gate, while Z_1, Z_2, Z_3 are as-

$$\begin{array}{lll} \dot{X}_1 = \lambda_P Y_1 - \lambda_D X_1 & \dot{Y}_1 = \lambda_U Z_1 - \lambda_I Y_1 X_3 & \dot{Z}_1 = \lambda_I Y_1 X_3 - \lambda_U Z_1 \\ \dot{X}_2 = \lambda_P Y_2 - \lambda_D X_2 & \dot{Y}_2 = \lambda_U Z_2 - \lambda_I Y_2 X_1 & \dot{Z}_2 = \lambda_I Y_2 X_1 - \lambda_U Z_2 \\ \dot{X}_3 = \lambda_P Y_3 - \lambda_D X_3 & \dot{Y}_3 = \lambda_U Z_3 - \lambda_I Y_3 X_2 & \dot{Z}_3 = \lambda_I Y_3 X_2 - \lambda_U Z_3 \end{array}$$

Table 4: ODE's for the repressilator, generated by the Hillston's method. In this case value of the parameters is: $\lambda_P = 1$ for production, $\lambda_I = 1$ for inhibition, $\lambda_D = 0.125$ for degradation, $\lambda_U = 0.001$ for terminating inhibition.

sociated to the repressed status. The numerical solution of this set of equations is displayed in Figure 3. As we can see there is no oscillation at all and, in addition, X_1, X_2, X_3 have exactly the same temporal evolution. However, following the considerations in [14], we expect a relation with the average behavior of the π-calculus model, more than with a single run of the simulation. To have an idea of the average value of the three proteins, we mapped the π-calculus description of the repressilator into the language of the probabilistic model checker PRISM [20], which can perform transient and steady state analysis of CTMC, and then we analyzed the average quantities of proteins as time varies. Results are shown in Figure 3. As we can see, oscillations are not present also in the average case, though there is a huge difference (almost two orders of magnitude) between the limit value for the ODE's and the limit value of the average.

Therefore, the average analysis and the related Hillston's method do not capture the oscillatory behaviour that the repressilator system is supposed to exhibit. In some sense, they do not satisfy the property of behavioral invariance that we believe to be central in the translation mechanism. Thus, the point is to find a translation into a set of differential equations manifesting an oscillatory behavior.

S-Systems provide an interesting framework in this sense. First of all, the format of the terms appearing in S-System's equations correspond to the terms obtained while performing a first-order power law approximation of an n-variable function (this approximation is a first-order Taylor expansion in logarithmic coordinated, cf. [2] for further details). In this perspective, we can think of S-System's production and degradation terms as approximations of the "real" production and degradation terms. Moreover, all exponents of this approximation have a clear biological meaning, therefore they can be estimated from a set of biological measurements (for instance, numerical traces coming from wet experiments), with standard numerical optimization techniques. In addition, note that, as opposed to mass action equations deriving from Hillston's technique, S-Systems have no "control variables", but all the control mechanisms are hidden in the structure of equations and in the value of the exponents.

In the following we sketch a possible solution to realize a direct transformation from stochastic π-calculus (or a generic process algebra) to S-Systems. The starting point is the biological system under study, for instance available in the form of a diagram, like Kohn's maps [21]. From these diagrams we can extract all the information needed to describe the logical structure of the model, like inhibitions, enhancements, complexes' formation, and so on. This means that we can derive both the structure of processes in π-calculus and the structure of S-System equations. If the maps are provided also with kinetic rates of all the

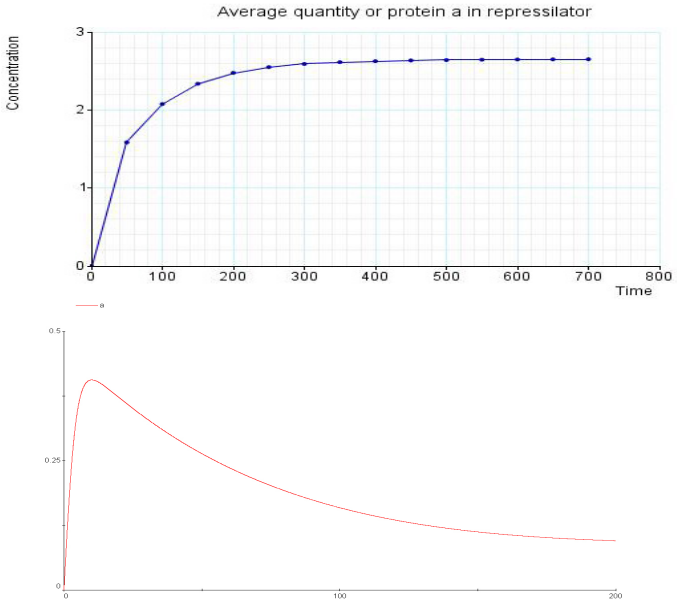

Figure 3: (**top**) Average value of protein X_i in the π-calculus model of Table 2, computed with PRISM model checker. Value of parameters are as in caption of Table 4 (**bottom**) Numerical solution of the system of equations of Table 4. Only the graphs for X_i are shown.

interactions, we have all the information to complete the π-calculus model, i.e. we have also stochastic rates.

We can thus proceed as follows: we simulate the π-calculus model several times, generating a set of traces only for the observables we are interested in. The idea is to use this data for training the parameters of the S-System. Note that the temporal series generated with π-calculus are noisy, therefore we need a number of them in order to be able to remove part of this noise. However, this data is cheap, being generated by computer simulations. In addition, if we have experimental data, we can use also these traces in the process of parameter determination, thus reducing the modeling errors. In this case we have to scale the traces coming from different sources, in order to have them coherent with basic concentrations.

This training part is not trivial: interpolating by a simple average we again lose the oscillatory behaviour. Hence, more refined techniques from numerical analysis are necessary to manage these temporal traces and remove noise without altering their information.

The main disadvantage of the method we proposed is the fact that it is an approximate procedure, prone to errors (especially in the training phase). This is in contrast with Hillston's technique, which is exact, in the sense that it is independent from particular runs of the simulation. On the other hand, its advantages mainly reside in producing a set of equations depending only on biological observables, which are more expressive than mass law equations, i.e. S-Systems, and that are free from the loss of behavioral details of the averaging

process. In addition, the values of the exponents of S-System's equations are revealing of the strength of interactions between observable variables at the particular level of abstraction of the model.

7 Conclusion and future work

What we have presented here is a step toward a clarification of the correspondence between two formalisms used for representing biological interactions. The two formalisms have profoundly different histories and foundations and further study is necessary. One focal point is tied to the method described in Section 6, which suffers from the drawback of hiding all the logical controlling structure inside the function spelling out rates. A much more intriguing question is whether it is possible to recover this logical structure, representing it explicitly in the structure of agents. Even though using non-constant rates remains a necessary ingredient to reproduce all the non-linear behaviors possibly encoded in S-Systems, establishing the link between the mathematical structure of rate functions and the logical structure of process algebra terms is considered by us the central open problem in this area.

References

[1] H. De Jong, "Modeling and simulation of genetic regulatory systems: A literature review," *Journal of Computational Biology*, vol. 9, no. 1, pp. 67–103, 2002.

[2] E. O. Voit, *Computational Analysis of Biochemical Systems*. Cambridge University Press, 2000.

[3] C. Priami, A. Regev, E. Y. Shapiro, and W. Silverman, "Application of a stochastic name-passing calculus to representation and simulation of molecular processes," *Inf. Process. Lett.*, vol. 80, no. 1, pp. 25–31, 2001.

[4] R. Blossey, L. Cardelli, and A. Phillips, "A compositional approach to the stochastic dynamics of gene networks," *T. Comp. Sys. Biology*, pp. 99–122, 2006.

[5] D. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *J. of Physical Chemistry*, vol. 81, no. 25, 1977.

[6] J. Hillston, "Fluid flow approximation of pepa models," in *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QEST05)*, 2005.

[7] C. Priami, "Stochastic π-calculus," *The Computer Journal*, vol. 38, no. 6, pp. 578–589, 1995.

[8] C. Priami and P. Quaglia, "Stochastic π-calculus," *Briefings in Bioinformatics*, vol. 5, no. 3, pp. 259–269, 2004.

[9] V. A. Saraswat, *Concurrent Constraint Programming*. MIT press, 1993.

[10] L. Bortolussi, "Stochastic concurrent constraint programming," in *Proceedings of 4th International Workshop on Quantitative Aspects of Programming Languages, QAPL 2006*, 2006.

[11] M. Calder, V. Vyshemirsky, D. Gilbert, and R. Orton, "Analysis of signalling pathways using the prism model checker," in *Proceedings of CMSB 2005*, 2005.

[12] J. Hillston, "Pepa: Performance enhanced process algebra," tech. rep., University of Edimburgh, 1993.

[13] D. Gillespie, "The chemical langevin equation," *Journal of Chemical Physics*, vol. 113, no. 1, pp. 297–306, 2000.

[14] D. Gillespie and L. Petzold, *System Modelling in Cellular Biology*, ch. Numerical Simulation for Biochemical Kinetics. MIT Press, 2006.

[15] J. R. Norris, *Markov Chains*. Cambridge University Press, 1997.

[16] L. Cardelli and A. Phillips, "A correct abstract machine for the stochastic pi-calculus," in *Proceeding of Bioconcur 2004*, 2004.

[17] K. Apt, *Principles of Constraint Programming*. Cambridge University Press, 2003.

[18] M. Calder, S. Gilmore, and J. Hillston, "Automatically deriving odes from process algebra models of signalling pathways," in *Proceedings of CMSB 2005*, 2005.

[19] J. M. G. Vilar, H. Yuan Kueh, N. Barkai, and S. Leibler, "Mechanisms of noise resistance in genetic oscillators," *PNAS*, vol. 99, no. 9, p. 5991, 2002.

[20] M. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic symbolic model checking with prism: A hybrid approach," *International Journal on Software Tools for Technology Transfer*, vol. 6, pp. 128–142, September 2004.

[21] K. W. Kohn, "Molecular interaction map of the mammalian cell cycle control and dna repair systems," *Molecular Biology of the Cell*, vol. 10, pp. 2703–2734, August 1999.