# Stochastic Concurrent Constraint Programming and Differential Equations

Luca Bortolussi [1]

*Department of Mathematics and Computer Science*
*University of Trieste, Italia*

Alberto Policriti [2]

*Department of Mathematics and Computer Science*
*University of Udine, Italia*

**Abstract**

We tackle the problem of relating models of systems (mainly biological systems) based on stochastic process algebras (SPA) with models based on differential equations. We define a syntactic procedure that translates programs written in stochastic Concurrent Constraint Programming (sCCP) into a set of Ordinary Differential Equations (ODE), and also the inverse procedure translating ODE's into sCCP programs. For the class of biochemical reactions, we show that the translation is correct w.r.t. the intended rate semantics of the models. Finally, we show that the translation does not generally preserve the dynamical behavior, giving a list of open research problems in this direction.

*Key words:* Stochastic Concurrent Constraint Programming, stochastic modeling ordinary differential equations, biological systems.

## 1 Introduction

In the last decade there has been a remarkable interest of the computer science community in systems biology [18], i.e. the branch of biological sciences concerned with the study of living beings in a systemic light. The key issue is that of understanding how the observable features of life emerge as the consequence of the complex interactions among its basic (molecular) constituents, like proteins and genes. In this comprehension activity a key role is played by mathematical modeling of biological systems and computational analysis of these models [6,17]. There are broadly two different classes of modeling formalisms used, the first based on

[1] luca@dmi.units.it
[2] policriti@dimi.uniud.it

the continuous and deterministic mathematics of differential equations, the second based on the theory of stochastic processes. Computer science enters this game at different levels: it provides the numerical routines for solving differential equations, it furnishes virtual environments where graphically building and analyzing models, and, above all, it lends to systems biology a class of formal languages, namely stochastic process algebras, that can be used to describe elegantly and precisely the systems of interest [22]. This is probably the most important contribution, as it goes towards the definition of a formal language specifically designed to tackle the intrinsic complexity of living systems. Process algebras used in biology are generally equipped with a stochastic semantics, resulting in a continuous-time Markov Chain (CTMC for short) [19]. Remarkably, when process algebras are used to describe biochemical reactions, the CTMC given by their semantics coincides with the one constructed when using classical stochastic simulation procedures, like the celebrated Gillespie algorithm [14].

When biochemical reactions are considered, the coexistence of Gillespie method and of a different modeling technique, based on ordinary differential equations (ODE), must be faced. Notably, both these methods are justified using the same fundamental principle, i.e. the law of mass action [14,24]. It is also important to recall that the relation between deterministic and stochastic models of biochemical reactions has been analitically studied in detail, leading to the development of an hybrid modeling approach via stochastic differential equations to chemical kinetics, i.e. the Chemical Langevin Equation [12].

The comparison between stochastic process and differential equations can be carried at two different levels: we can compare them either syntactically or semantically, looking at their dynamical behavior. In the case of stochastic and deterministic models of biochemical reactions, derived from the law of mass action, the syntactic comparison is somehow trivial: they should be equivalent, as they are derived from the same set of chemical equations. A different problem is whether these systems show an equivalent dynamical behavior. First of all, we must decide in what sense a stochastic system, whose traces show the characteristic fluctuations of noise, can be considered equivalent to a deterministic system. We may take a "minimalist" approach, considering two such systems equivalent if the deterministic system coincides with the average behavior of the stochastic one. With this approach we are dropping any information concerning the variance of the stochastic system, an assumption that cannot be made in general, see [24]. In fact, many stochastic systems exhibit an oscillatory behavior that is induced by stochastic fluctuations, while their average does not fluctuate at all; this is the case, for instance, of the stochastic model of the Lotka-Volterra prey-predator system, see [14,24] and the last section. Despite this simplification, however, only few special mass action deterministic models of biochemical reactions are equivalent in the behavioral sense to their stochastic counterpart, see again [12].

In this paper we focus on this intriguing problem in the more general context of stochastic models constructed using process algebras. Comparison of models of biological systems built using stochastic process algebras with models of the same

systems derived using differential equations can be motivated by different reasons. First, numerically solving differential equations is a computational task generally easier than the stochastic simulation or the state-based analysis of stochastic systems: if we can get a set of ODE's that describes the same behavior of a stochastic process, we may be able to analyze the model more efficiently. Viceversa, process algebra models are constructed taking into account the interaction patterns among entities of the model, while differential equations hide these interactions in numerical relations among variables: if, starting from a set of ODE's, we can build an equivalent SPA model, we may be able to recover part of the structure of these interaction patterns. Also SPA and ODE models can be compared at two levels, one syntactical and the other semantical, focussing on the dynamical behavior.

How can we compare syntactically stochastic process algebra programs and differential equations? A possibility is to define (sensible) translation procedures, operating at the syntactical level, that associate a set of differential equations to a stochastic model and, viceversa, a stochastic model to a set of differential equations. Therefore, a stochastic system and a set of ODE's may be considered equivalent whenever they can be derived one from the other using these translation methods.

Recently, there has been some work in this directions, developing techniques associating sets of differential equations to programs written in PEPA [16] or in $\pi$-calculus [8]. The basic idea of these methods is to approximate the number of syntactic terms of a certain kind present in parallel in the system with a real number, and then derive the variation of the number of such terms by a *syntactic* inspection of their structure and their communication patterns. Hence, transitions involving these terms will contribute with a negative flux, while transitions creating copies of these terms will give a positive flux. These translations are intuitively correct: for instance, if we write a process algebra model of biochemical reactions using mass action kinetics (which is always the case for $\pi$-calculus or PEPA), then the derived set of differential equations is exactly the set of mass action ODE associated to the biochemical reactions under examination, see [9] for a formal proof. On the other hand, the system of ODE's generated in this way is usually not behaviorally equivalent to the SPA model, in the sense of differential equations describing the average value of the stochastic system. The inverse direction, i.e. associating a stochastic process algebra model to a set of ODE's, has received less attention in literature, the only example known to us being [5], where the authors use as process algebra a stochastic version of concurrent constraint programming, sCCP [2].

In this paper, after recalling the basics of sCCP (Section2), we extend the work done in [5], showing a method to associate ordinary differential equations to sCCP programs written with a restricted syntax (Section 3) and a translation of ODE's into sCCP programs (Section 4). We then prove that these two translation are essentially one the inverse of the other, showing also that they behave well when applied, for instance, to sCCP agents describing biochemical reactions [4]. In practice, we show that the ODE's associated to sCCP agents modeling biochemical reactions, as defined in [4], are the usual ODE's describing the intended kinetics.

Unfortunately, syntactic techniques developed up to now are not invariant from

$$Program = D.A$$

$$D = \varepsilon \mid D.D \mid p(\vec{x}) :- A$$

$$\pi = \mathrm{tell}_\lambda(c) \mid \mathrm{ask}_\lambda(c)$$

$$M = \pi.G \mid M + M$$

$$G = \mathbf{0} \mid \mathrm{tell}_\infty(c) \mid p(\vec{y}) \mid M \mid \exists_x G \mid G.G \mid G \parallel G$$

$$A = \mathbf{0} \mid \mathrm{tell}_\infty(c) \mid M \mid \exists_x A \mid A.A \mid A \parallel A$$

Table 1
Syntax of sCCP.

a dynamical point of view. In the conclusions (Section 5) we comment more on this side of the problem, putting more emphasis on the need of finding translations linking process algebras and ODE's that preserve the observed behavior.

## 2 Stochastic Concurrent Constraint Programming

In this section we briefly recall the stochastic version of Concurrent Constraint Programming [23], as presented in [2,4].

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 $X + Y < 7$). 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 among agents 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 [2,4]) is obtained by adding a stochastic duration to all instructions interacting with the constraint store $\mathcal{C}$, i.e. `ask`, `tell`. Each instruction has an associated random variable, exponentially distributed with rate given by a function associating a real number to each configuration of the constraint store: $\lambda : \mathcal{C} \to \mathbb{R}^+$. This is a unusual feature in traditional stochastic process algebras like PEPA [15] or stochastic $\pi$-calculus [20], and it will be a crucially used in the translation mechanisms, cf. below. The syntax of sCCP can be found in Table 1.

The underlying semantic model of the language (defined via structural operational semantic, cf. [2]) is 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 eval-

uated. Therefore, as in stochastic $\pi$-calculus [20] or PEPA [15], we have a race condition between all active instructions such that the fastest one is executed.

As in [4], we allow also `tell` instructions with infinite rate, which will be executed instantaneously whenever encountered by an agent. To deal with this kind of instructions and with procedure calls, we need to define two transitions relations: one instantaneous and one stochastic. These transitions are applied in an interleaved fashion: the instantaneous relation is applied until possible, then one step of the stochastic one is executed. Restrictions on the syntax guarantee that the instantaneous transition is confluent and becomes quiescent after a finite number of steps, hence the stochastic semantics is well defined, see [3] for further details.

Variables used in the definition of rate functions need to store a single value that may vary over time. Such variables, for technical reasons, are conveniently modeled as variables of the constraint store, which are rigid (over time). To deal with this problem we store time varying parameters as growing lists with an unbounded tail variable. We will, however, use a natural notation where X=X+1 has the intended meaning of: "extract the last ground element $n$ in the list $X$, consider its successor $n+1$ and add it to the list (instantiating the old tail variable as a list containing the new ground element and a new tail variable)". We refer to such variables as *stream variables*.

We have developed an interpreter for the language that can be used for running simulations. This interpreter is written in Prolog and uses standard constraint solver on finite domains as manager for the constraint store. All simulations of sCCP shown in the paper are performed with it.

### *2.1 Modeling Biological Systems in sCCP*

In [3,4] we argued that sCCP can be conveniently used for modeling biological systems. In fact, while maintaining the compositionality of process algebras, the presence of a customizable constraint store and of variable rates gives a great flexibility to the modeler, so that different kinds of biological systems can be easily described within this framework. In [4], we showed that biochemical reactions and genetic regulatory networks are easily dealt by sCCP. In [3] we added to this list also formation of protein complexes and the process of folding of a protein, whose description requires the knowledge about spatial position of amino acids constituting the protein (a kind of information easily added exploiting the potentiality of the constraint store).

While modeling biochemical reactions, we take a point of view different from the classical usage of process algebras for this task: while in $\pi$-calculus the description is molecular-centric, describing each single molecule of the system as an independent process, we take a reaction-centric approach, where each reaction (or action capability) is associated to a process, while molecules are represented by variables of the constraint store (actually, stream variables). To simplify the task of modeling, in [4] we defined a library of agents corresponding to different types of biochemical reactions. Notably, the reaction-centric approach and the presence of non-constant rates allows to describe reactions that have a chemical kinetics dif-

| | |
|---|---|
| $R_1 + \ldots + R_n \rightarrow_k P_1 + \ldots + P_m$ | $\begin{array}{l}\text{reaction}(k,[R_1,\ldots,R_n],[P_1,\ldots,P_m]) :- \\ \quad \text{ask}_{r_{MA}(k,R_1,\ldots,R_n)}\,(\bigwedge_{i=1}^{n}(R_i>0))\,. \\ \quad \big(\,\|_{i=i}^{n}\,\text{tell}_\infty(R_i = R_i - 1)\,\| \\ \quad \|_{j=1}^{m}\,\text{tell}_\infty(P_j = P_j + 1)\big). \\ \quad \text{reaction}(k,[R_1,\ldots,R_n],[P_1,\ldots,P_m])\end{array}$ |
| $S \mapsto^{E}_{K,V_0} P$ | $\begin{array}{l}\text{mm\_reaction}(K,V_0,S,P) :- \\ \quad \text{ask}_{r_{MM}(K,V_0,S)}(S>0). \\ \quad (\text{tell}_\infty(S = S-1)\,\|\,tell_\infty(P = P+1))\,. \\ \quad \text{mm\_reaction}(K,V_0,S,P)\end{array}$ |

where

$$r_{MA}(k,X_1,\ldots,X_n) = k \cdot X_1 \cdots X_n;$$
$$r_{MM}(K,V_0,S) = \tfrac{V_0 S}{S+K};$$

Table 2
Translation into sCCP of different biochemical reaction types, taken from [4]. The reaction process models a mass-action-like reaction, while the second arrow corresponds to a reaction with Michaelis-Menten kinetics.

ferent from the standard mass action one. This is not possible, for instance, in $\pi$-calculus, given the fact that global rates are defined there using a mass action principle: essentially, the number of possible communications on a channel are multiplied by the basic rate of that communication. In Table 2, we present an extract of the library defined in [4], where two different typologies of reactions are considered: the first one has the classical mass action kinetics, while the second represents a catalyzed transformation of $S$ into $P$ (thanks to the action of enzyme $E$) and has a Michaelis-Menten kinetics (its rate is computed using the expression at the bottom of the table, corresponding to the format of the Michaelis-Menten differential equation for enzymatic kinetics [10]; formal justification of these rates in stochastic modeling can be found in [21]).

## 3 sCCP to Ordinary Differential Equations

In this section we define a translation machinery that associates a set of ordinary differential equations to a sCCP program. This translation applies to a restricted version of the language, both in the constraint store and in the syntax. Despite these restrictions, this sub-language is sufficient to deal with applications in modeling biochemical reactions and genetic regulatory networks. After defining this translation, we show that it preserves the chemical kinetic, or the rate semantics, as defined in [9]. Essentially, we take some sCCP agents used in modeling biochem-

ical reactions (Table 2), and show that the associated ODE is the one describing their kinetics [10], i.e mass action for simple reaction agents and Michaelis-Menten equations for agents with Michaelis-Menten rate.

We restrict the language both in the admissible constraints of the store and in the syntax of the agents. We restrict all the variables of the constraint store to be stream variables, and we allow only equalities and inequalities as constraints to be asked. In addition, we restrict the possible updates of variables in the store to a very special class of constraints, of the form $X = X+k$, where $k$ is a positive or negative constant. Note that these are the kind of updates we use in biological modeling, see Table 2. The syntactic restrictions are the following: we allow only sequential agents, fixing the number of agents in parallel in the initial configuration of the system. Sequential agents are simply agents not containing any parallel operator. In addition to this, we disallow the possibility of defining local variables: all variables used by agents must be global. Therefore, we can avoid to pass parameters in the procedure call, supposing that all procedures know the name of the global variables they must act on. [3]

The translation from restricted sCCP programs to ODE proceeds in several steps, illustrated in the following paragraphs.

**Step 1: Reduced Transition Systems.**

The first step consists in manipulating the syntactic trees of the sequential agents composing the network, in order to rewrite them in a simpler form, called *reduced transition systems*, cf. [3]. In order to illustrate this procedure, we show its functioning on the following simple sCCP agent:

$$
\begin{array}{l}
\mathrm{RW}_X \text{ :-} \\
\quad \mathrm{tell}_1(X = X - 1).\mathrm{RW}_X \\
+\ \mathrm{tell}_1(X = X + 2).\mathrm{RW}_X) \qquad\qquad f(X) = \frac{1}{X^2+1} \\
+\ \mathrm{ask}_{f(X)}(true).(\ \ \mathrm{tell}_1(X = X - 2).\mathrm{RW}_X \\
\qquad\qquad +\mathrm{tell}_1(X = X + 1).\mathrm{RW}_X\ )
\end{array}
$$

This agent performs a sort of random walk in one variable, increasing or decreasing its value by 1 or 2 units, depending on its inner state.
The syntactic tree of this agent is a tree where each node either contains an instruction or is a summation node (see Figure 1(1)). No parallel composition nodes exist, as the agent is sequential. The basic idea of the translation is to collapse all transitions with infinite rate following a stochastic-timed one; to do this, we first move the information about guards, updates and rates of transitions from nodes to edges, and then collapse the edges. We deal with recursive calls introducing cycles in the syntactic tree, effectively constructing a compact form of the labeled transition system of the agent.

[3] Sometimes parameter passing is used to reutilize the same code on different global variables. In this case, we need to define different procedures, one for each set of global variables we are interested in.

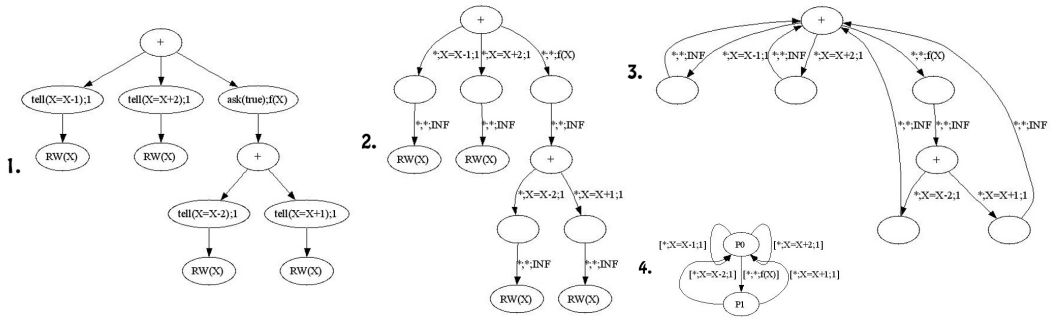



Fig. 1. Steps in the manipulation of the syntactic tree of agent $\mathrm{RW}_X$ (1), transforming it into a reduced transition system (4). In step (2) we moved information from nodes to edges, and in (3) we removed procedure calls. In the graphs, empty labels are denoted withan asterisk *.

In the first step of the procedure we label the incoming edge of a node with a triple where the first element is a finite set of guards, the second element is a set of assignments, and the third element is the rate function of that transition. Nodes containing ask instructions contribute just with guards and rates, while tell agents contribute with updates and rates (consistency check for assignments is trivial). We also remove labels from nodes corresponding to ask or tell. Summation nodes and procedure calls are left untouched, though incoming arcs of procedure calls are labeled by $(\emptyset, \emptyset, \infty)$ (see Figure 1(2)).

The following step consists in removing the nodes corresponding to recursive calls. This can be achieved in two ways, depending on the presence or absence of a copy of the syntactic tree of the called procedure in the graph under manipulation. If no copies of the tree of the called procedure are present in the graph, we replace the calling node with one copy of such tree (with the transformations of first step already performed). If, instead, there already exists a copy of the tree of called procedure in the graph we are managing, we simply redirect the incoming edge of the node calling the procedure to the root of the copy of such tree. In this way, we can introduce cycles in the graph (see Figure 1(3)).

The next sequence of steps removes all nodes having a single outgoing edge with infinite rate and a single incoming edge. We are guaranteed that all nodes with more than one incoming edge have just stochastic edges (edges whose rate labeling them is not $\infty$) exiting from them (see [3]). Therefore, we are removing only nodes defining a sequence of instantaneous and deterministic steps. When these nodes are removed, their entering and exiting edge are merged, and the corresponding labels are joined together, taking the conjunction of guards and updates. As rate of the new edge we put the (only) one less that $\infty$, if any, $\infty$ otherwise. At the end of this procedure, all edges will have a stochastic rate less than infinity. After merging all possible edges, we clean all labels of remaining nodes (see Figure 1(4)).

**Step 2: the interaction matrix.**

Consider an sCCP program, composed by several agents in parallel. Once we have converted all of them into their reduced transition system representation, we associate a unique number to each state and each transition of all such graphs. Successively, we need to identify the variables of the system of differential equations. All variables of the store used by the agents will have a syntactic counterpart in a variable of the ODE system. In the following, we denote such variables by $X_1, \ldots, X_n$. In addition, we associate a variable to each state of the reduced transition systems, of the form $P_i$, where $P$ is a name never used for a variable of the store, and $i$ is the index assigned to the node. Consider a transition indexed by $j$; we indicate with $\mathrm{exit}_j$ the variable labeling the exiting node of edge $j$, with $\mathrm{rate}_j(X_1, \ldots, X_n)$ its rate function (labeling the corresponding edge in the RTS) and with $\mathrm{guard}_j(X_1, \ldots, X_n)$ the function constructed by taking the product of the indicator functions of the guards of each edge. An indicator function of a guard returns 1 if the guard is satisfied, 0 otherwise.

We are now ready to define the *interaction matrix*. This matrix has one row for each variable $X_1, \ldots, X_n, P_1, \ldots, P_m$ (we suppose to have $m$ states in total in the RTS of sCCP agents), and one column for each transition. Column $j$ is constructed in the following way: if edge $j$ goes from the node identified by $P_i$ to a node identified by $P_h$, we put a -1 in correspondence to row $P_i$ and a +1 in correspondence of row $P_h$ (if $i = h$, we simply put a zero). Then, for each update instruction of edge $j$ of the form $X_l = X_l + \tau$, we put $\tau$ in correspondence of row $X_l$. All other entries of the column are set to 0. We denote the interaction matrix by $Im$ and the element corresponding to the row associated to variable $Y$ and to column $j$ by $Im(Y, j)$.

For the example introduced above, the resulting interaction matrix is:

(1)

| $X$ | $-1$ | $+2$ | $0$ | $-2$ | $+1$ |
|---|---|---|---|---|---|
| $P_0$ | $0$ | $0$ | $-1$ | $+1$ | $+1$ |
| $P_1$ | $0$ | $0$ | $+1$ | $-1$ | $-1$ |

**Writing ODE's.**

Once we have the interaction matrix, writing the set of ODE's is very simple. To each row of the matrix we associate an equation expressing the variation of the corresponding variable. The equation for a variable $Y$ is the following ($k$ indicates the number of columns in the matrix):

$$\dot{Y} = \sum_{j=1}^{k} \left( Im(Y, j) \cdot \mathrm{guard}_j(X_1, \ldots, X_n) \cdot \mathrm{rate}_j(X_1, \ldots, X_n) \cdot \mathrm{exit}_j \right) \tag{2}$$

For instance, the set of ODE's associated to the agent of our example is

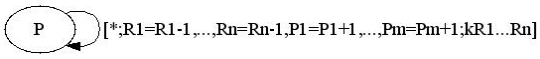
$$\begin{cases} \dot{X} = P_0 - P_1 \\ \dot{P}_0 = -\frac{1}{X^2+1} P_0 + 2P_1 \\ \dot{P}_1 = \frac{1}{X^2+1} P_0 - 2P_1 \end{cases}$$

### *3.1 ODE's for Biochemical sCCP Agents*

In Table 2 we have presented part of the library of sCCP agents describing the main biochemical reactions. Each agent in that diagram corresponds to a reaction having a specific kinetics. We considered here only mass action kinetics and Michaelis-Menten kinetics, theories usually presented by means of differential equations [10]. In this section we show that if we apply the translation just defined to these agents, we obtain exactly the differential equations corresponding to their kinetics. Therefore, we can say that our translation preserves the rate semantics, in the sense of [9].

**Mass action kinetics.**

Consider the sCCP encoding (Table 2) of a biochemical reaction with mass action kinetics, indicated by the arrow $R_1 + \ldots + R_n \rightarrow_k P_1 + \ldots + P_m$. First of all, note that the expression of the rate function allows us to remove the condition in the ask guard. In fact, whenever one of the $R_i$ variables is zero, the function $r_{MA}$ is also zero, hence the term in the ODE will give no contribution. The reduced transition system for the agent reaction$_{k,R_1,\ldots,R_n,P_1,\ldots,P_m}$ is the following




Applying the translation method defined, we obtain the following set of ODE:

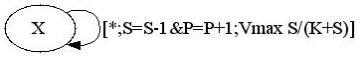
$$\begin{array}{lll} \dot{R}_1 = -kR_1 \cdots R_n & \dot{P}_1 = kR_1 \cdots R_n & \\ \vdots & \vdots & \dot{P} = 0 \\ \dot{R}_n = -kR_1 \cdots R_n & \dot{P}_m = kR_1 \cdots R_n & \end{array}$$

This is exactly the form of Mass Action ODE for this reaction.

**Michaelis-Menten kinetics.**

Let's consider a reaction based on Michaelis-Menten kinetics, represented in Table 2 by the chemical arrow $S \mapsto^{E}_{K,V_0} P$.

To generate the corresponding set of ODE's, we first have to build its reduced transition system, having the form:



Note that also in this case we dropped the guard condition in the ask instruction, as the form of the rate function subsumes it. Building the interaction matrix, we can derive the corresponding set of ODE, taking the desired form of classic Michaelis-Menten equation:

$$\dot{P} = \frac{V_{max}S}{K+S} \qquad \dot{S} = -\frac{V_{max}S}{K+S} \qquad \dot{X} = 0$$

## 4 Ordinary Differential Equations to sCCP

In this section we define a transformation that associates a sCCP process to a generic set of ordinary differential equation. Then, we show that the transformation behaves well, in the sense that the set of ODE associated to the derived sCCP agent, using the method of the previous section, is exactly the initial set of ODE. Finally, we give an example.

Consider a system of first order ODE with $n$ variables $X_1, \ldots, X_n$; we write it separating positive and negative addends in each equation:

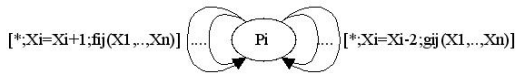
$$\text{(3)} \qquad \begin{cases} \dot{X}_1 = \sum_{j=1}^{h_1} f_{1j}(X_1, \ldots, X_n) - \sum_{j=1}^{k_1} g_{1j}(X_1, \ldots, X_n) \\ \vdots \\ \dot{X}_n = \sum_{j=1}^{h_1} f_{1j}(X_1, \ldots, X_n) - \sum_{j=1}^{k_1} g_{1j}(X_1, \ldots, X_n) \end{cases}$$

The translation to sCCP simply proceeds associating an agent to each differential equation of (3), defined by

$$\begin{aligned} \text{man}_{X_i}\text{:-} \\ & \sum_{j=1}^{h_i} \text{tell}_{f_{ij}(X_1,\ldots,X_n)}(X_i = X_i + \tau).\text{man}_{X_i} \\ + \ & \sum_{j=1}^{h_i} \text{tell}_{g_{ij}(X_1,\ldots,X_n)}(X_i = X_i - \tau).\text{man}_{X_i} \end{aligned}$$

Here $\tau$ denotes the basic increment, that we consider as unitary. Notably, if we apply the transformation defined in the previous section, associating a set of ODE to our sCCP agent, we can easily see that we obtain exactly the initial set of ODE. Before showing this in more detail, we want to spend some words on the functional rates. This is a feature different from common process algebras, where rates are real numbers and the final speed of an action is determined from this basic rate in a mass action style, i.e. summing all rates of enabled transitions of a certain type. As a result, the ODE format that can be generated from these process algebras coincide with the set of mass action equations, like those of Section 3.1. On the contrary, functional rates are somehow more expressive, as they allow to encode, at least syntactically, every possible ODE, without restrictions. In fact, we are using non-constant rates to hide the logical interaction mechanism that is usually modeled explicitly in common process algebras.

To generate a set of ODE from the agents $\text{man}_{X_i}$, we have first to obtain their reduced transition system. It is easy to see that it has the form




If we consider the interaction matrix that is derived from these RTS, we observe that the row corresponding to variable $X_i$ has non-zero entries only relatively to the transitions of the RTS for $\text{man}_{X_i}$, each entry being equal to $\tau$ or $-\tau$. The corresponding ODE therefore is

$$\dot{X}_i = \sum_{j=1}^{h_i} \tau f_{ij}(X_1, \ldots, X_n) P_i - \sum_{j=1}^{k_i} \tau g_{ij}(X_1, \ldots, X_n) P_i. \tag{4}$$

Now, we can perform two simplifications: first, $\tau$ is the unitary increment, and we can set it equal to 1. Secondly, the equation for $P_i$, the variable denoting the only state of agent $\text{man}_{X_i}$, has equation $\dot{P}_i = 0$, hence $P_i$ is constant. As we have just one agent $\text{man}_{X_i}$, $P_i$ is equal to one. Therefore, equation (4) boils down to

$$\dot{X}_i = \sum_{j=1}^{h_i} f_{ij}(X_1, \ldots, X_n) - \sum_{j=1}^{k_i} g_{ij}(X_1, \ldots, X_n), \tag{5}$$

which is exactly the starting equation for $X_i$.

Note that as basic step in the translation from ODE to sCCP we are using a generic $\tau$. $\tau$ determines the size of the basic increment or decrement of variables of the system. In sCCP, we are not forced to use integer variables, but we can let them vary, for instance, on a grid of rational numbers, where the basic distance can be set equal to $\tau$. Varying the size of $\tau$, we can calibrate the effect of the stochastic fluctuations, reducing or increasing it. This is evident in the following example, where we compare solutions of ODE's and the simulation of the corresponding sCCP processes.

Let's consider the following system of equations, representing a model of the repressilator, a synthetic genetic network having an oscillatory behavior (see [11]):

$$\begin{aligned} \dot{X}_1 &= \alpha_1 X_3^{-1} - \beta_1 X_1^{0.5}, \quad \alpha_1 = 0.2 \\ \dot{X}_2 &= \alpha_2 X_1^{-1} - \beta_2 X_2^{0.5}, \quad \alpha_2 = 0.2 \\ \dot{X}_3 &= \alpha_3 X_2^{-1} - \beta_3 X_3^{0.5}, \quad \alpha_3 = 0.2. \end{aligned} \tag{6}$$

The values of $\beta$ are here parameters; their value has a severe impact on the behavior of the system, that for some values of $\beta$ oscillates (as expected from repressilator) while for some other values does not oscillate at all (see Figure 2). The corresponding sCCP process is

$$\text{man}_{X_1} \parallel \text{man}_{X_2} \parallel \text{man}_{X_3}, \tag{7}$$

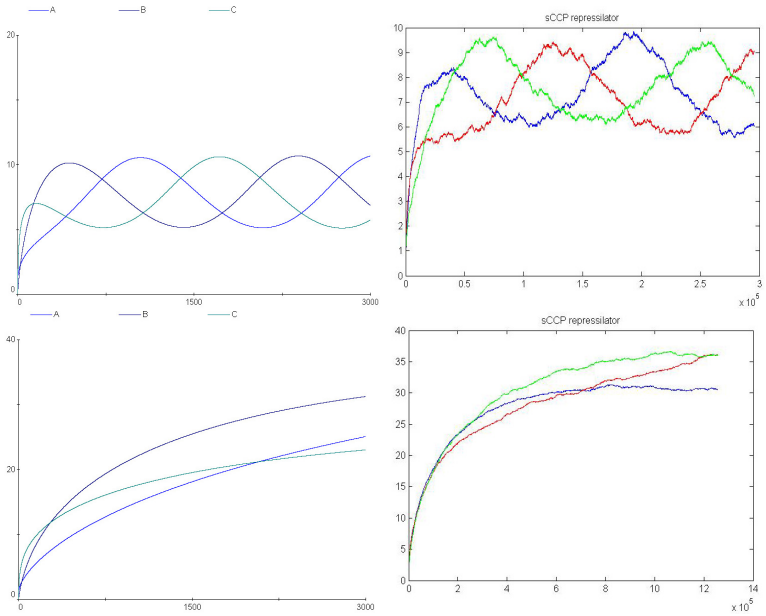



Fig. 2. Numerical solutions of the system of equations for the repressilator (left column), and the numerical simulation of the corresponding sCCP program (right column), for $\beta = 0.01$ (top) and $\beta = 0.001$ (bottom)

where

$$\begin{aligned}
\mathrm{man}_{X_1} &: -(\mathrm{tell}_{[\alpha_1 X_3^{-1}]}(X_1 = X_1 + \tau) + \mathrm{tell}_{[\beta_1 X_1^{0.5}]}(X_1 = X_1 - \tau)).\mathrm{man}_{X_1} \\
\mathrm{man}_{X_2} &: -(\mathrm{tell}_{[\alpha_2 X_1^{-1}]}(X_2 = X_2 + \tau) + \mathrm{tell}_{[\beta_2 X_2^{0.5}]}(X_2 = X_2 - \tau)).\mathrm{man}_{X_2} \\
\mathrm{man}_{X_3} &: -(\mathrm{tell}_{[\alpha_3 X_2^{-1}]}(X_3 = X_3 + \tau) + \mathrm{tell}_{[\beta_3 X_3^{0.5}]}(X_3 = X_3 - \tau)).\mathrm{man}_{X_3}
\end{aligned}$$

In Figure 2 we compare numerical solutions of S-Systems and simulations of the corresponding sCCP process, for different values of $\beta$ ($\alpha$s have the value defined in equation (6)) and fixed $\tau$, equal to 0.01. 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 [14,13]. Interestingly, though there is a strong qualitative accordance between the deterministic and stochastic kinetics, the graphs differ in the time scale. In the stochastic simulation, in fact, time is no more an independent variable, but the temporal distance between consecutive events is governed by the sum of the fluxes of the differential equations.

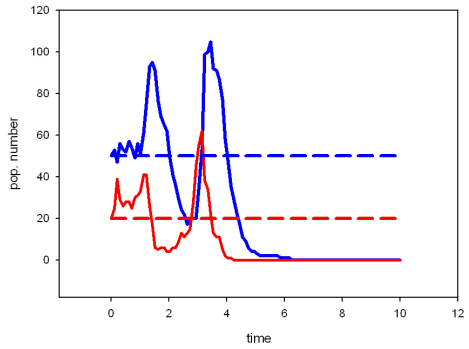



Fig. 3. Stochastic simulation of the Lotka Volterra model in sCCP (solid line) compared with the solution of associated ODE's (dotted line), for initial conditions $E_0 = 20$ and $C_0 = 50$. Predators are shown in blue and preys are shown in red.

## 5 Future Work and Conclusions

In the paper we focused on the problem of relating stochastic models written with process algebras and models with differential equations. Despite our interest is mainly in biological systems, and examples are drawn from this field, the problem is general, and these methods can be applied to a broad range of systems, like computer networks. Here we focused in the definition of translation procedures associating sets of differential equations to process algebraic models and stochastic process algebra programs to differential equations. The process algebra we used is sCCP, a stochastic version of CCP; some of its features, functional rates above all, play an important role in this translation procedure. In particular, they enable to define a translation from general differential equations that is well-behaved, in the sense that applying the transformation from sCCP programs we obtain again the starting equation.

These translation procedures work at the syntactic level, and there is no theoretical guarantee that the transformed model shows a behavior that is equivalent to the one of the initial model. For instance, consider the sCCP program for the set of reactions describing a simple population dynamics, the so-called Lotka-Volterra model ($C$ is the predator and $E$ is the prey), $C \rightarrow_2$, $E \rightarrow_5 2E$ and $C + E \rightarrow_{0.1} 2C$. The set of ODE obtained with the translation method defined above are $\dot{C} = 0.1EC - 2C$ and $\dot{E} = 5E - 0.1EC$. If we set the initial conditions of the system to $E_0 = 20$ and $C_0 = 50$, the ODE's are in equilibrium, and their solution is a straight line. Instead, the stochastic system shows no equilibrium at all: the number of preys and predators oscillates until they both go extinct, see Figure 3.

In other cases, the ODE's associated to a sCCP program perfectly capture the behavior of the system, see [7] for examples in this sense. However, the Lotka-Volterra example is one of many where associated ODE fail to capture the behavior of the system, so we are far away from having defined a procedure that translated stochastic process algebra programs into ODE's in a *semantically* correct way. This

is the main open problem in this research topic. Indeed, there are other questions worth considering. First of all, we lack a reasonable definition of *behavioral equivalence* between stochastic processes and differential equations. Considering the average behavior of the stochastic process is not the best solution, as there are cases where the stochastic model exhibits strong oscillations, though the average value of the systems does not oscillate at all [**?**,5]. Therefore, a different approach, aiming to capture qualitative aspects more than quantitative information, should be adopted.
Another open question concerns the characterization of a class of sCCP programs for which the associated ODE's works well also from a behavioral viewpoint. For these systems, the syntactic transformation we have defined is safe. Finally, an interesting point is that of seeing if and where hybrid systems, like hybrid automaton [1], can enter the picture. In fact, while passing to ODE's we are dropping any form of non-determinism (which is, instead, present in stochastic systems under the form of race conditions); hybrid automaton, on the other hand, while having a continuous time evolution governed by ODE's, have also discrete states and non-deterministic transitions among them.
Finally, we need to characterize in a mathematically more precise way what happens in the translation from ODE's to sCCP. Specifically, we want to understand how the variation in the increment step $\tau$ influences the behavior of the stochastic process w.r.t. the one of the ODE. Our conjecture is that in the limit of $\tau \to 0$, the average of the stochastic behavior coincides with the solution of the ODE (modulo a suitable time rescaling).

## References


[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

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

[3] L. Bortolussi. *Constraint-based approaches to stochastic dynamics of biological systems*. PhD thesis, PhD in Computer Science, University of Udine, 2007. In preparation. Available on request from the author.

[4] L. Bortolussi and A. Policriti. Modeling biological systems in concurrent constraint programming. In *Proceedings of Second International Workshop on Constraint-based Methods in Bioinformatics, WCB 2006*, 2006.

[5] L. Bortolussi and A. Policriti. Relating stochastic process algebras and differential equations for biological modeling. *Proceedings of PASTA 2006*, 2006.

[6] J. M. Bower and H. Bolouri eds. *Computational Modeling of Genetic and Biochemical Networks*. MIT Press, Cambridge, 2000.

[7] M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of rkip on the erk signalling pathway using the stochastic process algebra pepa. *Transactions on Computational Systems Biology*, 4230:1–23, 2006.

[8] L. Cardelli. Chemical $\pi$-calculus. *Draft*, 2006.

[9] L. Cardelli. On process rate semantics. *draft*, 2006.

[10] A. Cornish-Bowden. *Fundamentals of Chemical Kinetics*. Portland Press, 3rd edition, 2004.

[11] M.B. Elowitz and S. Leibler. A syntetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.

[12] D. Gillespie. The chemical langevin equation. *Journal of Chemical Physics*, 113(1):297–306, 2000.

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

[14] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. of Physical Chemistry*, 81(25), 1977.

[15] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

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

[17] H. Kitano. *Foundations of Systems Biology*. MIT Press, 2001.

[18] H. Kitano. Computational systems biology. *Nature*, 420:206–210, 2002.

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

[20] C. Priami. Stochastic $\pi$-calculus. *The Computer Journal*, 38(6):578–589, 1995.

[21] C. V. Rao and A. P. Arkin. Stochastic chemical kinetics and the quasi-steady state assumption: Application to the gillespie algorithm. *Journal of Chemical Physics*, 118(11):4999–5010, March 2003.

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

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

[24] Darren J. Wilkinson. *Stochastic Modelling for Systems Biology*. Chapman & Hall, 2006.