Stochastic concurrent constraint programming and differential equations

Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 190 (2007) 27–42
www.elsevier.com/locate/entcs

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.

Keywords: 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 under 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 activity a key role is played by the mathematical modeling of biological systems and the computational analysis of these models [6,17]. There are,

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

doi:10.1016/j.entcs.2007.07.003

broadly, two different classes of modeling formalisms used, the first based on 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 formally specified language, 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 analytically studied in detail, leading to the development of hybrid modeling approaches to chemical kinetics, using, for instance, stochastic differential equations (the Chemical Langevin Equation [12]).

In this paper we focus on the problem of defining procedures translating a stochastic model written using a process algebra, specifically stochastic Concurrent Constraint Programming (sCCP [2]), into a set of ordinary differential equations (ODEs), and viceversa. The definition of translations between models of biological systems built using stochastic process algebras ad models written with 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 ODEs 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.

What properties should we expect to hold between models obtained applying such translations? Ideally, we would like them to have an *equivalent dynamical behavior*. Equivalence can be *qualitative*, meaning that the dynamics in the two models has the same characterizing features, or even *quantitative*, with an agreement also in numerical values. However, even the precise definition of a notion of equivalence is a difficult task, because stochastic systems have a noisy dynamics, hence noise must be removed (or ignored) when comparing them with differential

systems. A simple possibility is to consider the average trace of the stochastic model as a representative of its characteristic dynamics; unfortunately, fluctuations cannot be ignored so easily, as sometimes they rule the behavior of the system. For instance, 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, for the stochastic model of the Lotka-Volterra prey-predator system, see [14,24] and the last section. Weaker notions of equivalence may be considered and defined; we give further comments on this problem in the conclusions. Defining translations preserving the behavior is also a difficult matter; in fact, the translations presented in this paper are not behaviorally invariant in general. However, when we restrict the attention to biochemical systems, they possess a simpler but interesting property.

Biochemical networks are modeled by biologists giving the list of reactions in which the molecules of the system are involved. Given such list, we can construct easily a model in sCCP; in addition, there is a canonical way to associate a set of ODEs to such list. It is thus reasonable to ask that the ODEs obtained from the sCCP model of a list of biochemical reactions coincide with the usual ODEs. This property has been called *preservation of rate semantics* in [9]. In the paper, we show that the translation methods for sCCP do satisfy this property. Unfortunately, preservation of rate semantics tells nothing about preservation of dynamical behavior, which can be different between the sCCP model and the associated ODEs.

Recently, there has been some work in this area, developing techniques associating sets of differential equations to programs written in PEPA [16] or in π-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 satisfy the coherency condition staten above: for instance, if we write a process algebra model of biochemical reactions using mass action kinetics (which is always the case for π-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. The inverse direction, i.e. associating a stochastic process algebra model to a set of ODE's, has received less attention in the 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 (Section 2), 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 study the relations between the defined translation, showing that they preserve rate semantics when applied 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. In the conclusions (Section 5) we comment more on behavioral invariance, giving a list of open research problems.

2 Stochastic Concurrent Constraint Programming

Concurrent Constraint Programming (CCP [23]) 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 π-calculus [20], and it will be a crucially used in the translation mechanisms, cf. below.

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

In the language we allow also `tell` instructions with infinite rate, which are 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 also 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, cf. [3] for further details. 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).

To simplify the task of modeling, in [4] we defined a library of agents corresponding to different types of biochemical reactions. Notably, the presence of non-constant rates allows to describe reactions that have a chemical kinetics different from the standard mass action one. This is not possible, for instance, in π-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 1, 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 an 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 biochemical reactions and genetic regulatory networks. After defining this translation, we show that it preserves the rate semantics, i.e. essentially the chemical kinetic, as defined in [9]. Practically, we take the sCCP agents used in modeling biochemical reactions (Table 1), 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.

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

where

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

Table 1
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. In the code, $X-1$ stands for $X = X-1$, while $X+1$ stands for $X = X+1$.

The language is restricted both in the admissible constraints of the store and in the syntax of the agents. All the variables of the constraint store are required to be stream variables, and only equalities and inequalities constraints can be asked. In addition, the possible updates of variables in the store are confined 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 1. The syntax of the restricted language is given in Table 2. First, only sequential agents are allowed, and the number of agents in parallel in a global configuration of the system, called also a *network*, is constant and fixed from the beginning. *Sequential agents* are agents not containing any parallel operator, hence they cannot fork new agents at run-time. In addition, the possibility of defining local variables is disallowed: all variables used by agents must be global. Therefore, there is no need to pass parameters in the procedure call, supposing that all procedures know the name of the global variables they must act on.[3] These restrictions are in the spirit of those introduced in [16]: we are forbidding an infinite unfolding of agents and we are considering only global interactions, forcing the speed of each action to depend on the whole state of the system. Indeed, also in [8] we find similar restrictions, though the comparison with sCCP is subtler. First of all, the version of π-calculus presented in [8] does not allow the use of the restriction operator, meaning that interactions have a global scope. However, agents in the π-calculus of [8] are not sequential, as each process is associated to a single molecule, and the production of new molecules is essentially achieved by forking processes at run-time. This is not necessary in sCCP, as sCCP-agents model reactions, while molecules are identified by variables of the system. Finiteness in the π-calculus of [8], however, corresponds to the property that the number of different syntactical terms that can be present in a system is always finite.

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

[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.

$$Program = D.N$$
$$D = \varepsilon \mid D.D \mid p :- A$$
$$\pi = \mathrm{tell}_\lambda(c) \mid \mathrm{ask}_\lambda(c) \qquad M = \pi.G \mid M + M$$
$$G = \mathbf{0} \mid \mathrm{tell}_\infty(c).G \mid p \mid M \qquad A = \mathbf{0} \mid M$$
$$N = A \mid A \parallel N$$

Table 2
Syntax of the restricted sCCP.

Step 1: Reduced Transition Systems.

The first step consists in associating a labeled graph to each sequential agent composing the network, the so called *reduced transition system* [3]. In order to illustrate this procedure, we show its functioning on the following simple sCCP agent:

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

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.
Each sequential agent, as defined in Table 2, can be of three different types: a stochastic choice, an instantaneous tell or a procedure call. Specifically, each branch of a stochastic choice begins with a timed `ask` or `tell`, followed by zero o more instantaneous `tell`, followed again by a procedure call or by another stochastic choice. The first operation to perform is to collapse the timed instruction opening a stochastic branch (i.e. $\mathtt{ask}_\lambda$ or $\mathtt{tell}_\lambda$) with all the instantaneous $\mathtt{tell}_\infty$ following it, replacing them with a predicate of the form $\mathtt{action}(c, d, \lambda)$, where c is a guard that must be entailed by the store for the branch to be enabled, d is the constraint that will be posted to the store and λ is the stochastic rate of the branch. After this replacement, there are only two possible agent types left: stochastic branches and procedure calls. We denote by $\mathtt{collapsed}(A)$ the agent A after this replacement. For example, the agent $\mathtt{collapsed}(RW_X)$ simply is:

$$\begin{aligned} \mathtt{collapsed}(\mathrm{RW}_X) :- & \\ & \mathrm{action}(true, X = X - 1, 1).\mathrm{RW}_X \\ + \ & \mathrm{action}(true, X = X + 2, 1).\mathrm{RW}_X \\ + \ & \mathrm{action}(true, true, f(X)).(\quad \mathrm{action}(true, X = X - 2, 1).\mathrm{RW}_X \\ & \qquad\qquad + \mathrm{action}(true, X = X + 1, 1).\mathrm{RW}_X \ ) \end{aligned}$$

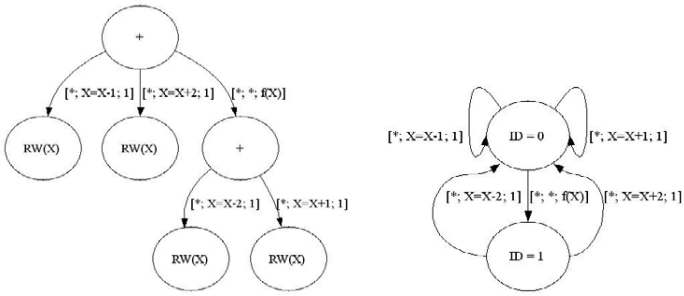


Fig. 1. Steps in the creation of the reduced transition system for RW_X. The graph on the left is the one obtained from the agent $\texttt{collapsed}(RW_X)$, while the graph on the right is its reduced transition system. In these graphs, the asterisk * denotes the constraint *true*.

We can now associate to each agent $\texttt{collapsed}(A)$ a graph where nodes correspond to stochastic choices (labeled with "+") and procedure calls (labeled by the name of the called procedure) of $\texttt{collapsed}(A)$, while edges correspond to instructions $\texttt{action}(c, d, \lambda)$, and are labeled by the triple (c, d, λ). The structure of the graph mirrors the structure of the agent $\texttt{collapsed}(A)$. The *entering node* of such graph is the node corresponding to the first instruction of $\texttt{collapsed}(A)$ (which is always a summation). The graph for the agent $\texttt{collapsed}(RW_X)$ is shown in Figure 1 (left).

We can now process the initial configuration of the network, $A_1 \parallel \ldots \parallel A_n$, acting separately on each sequential component A_i. Consider the graph associated to $\texttt{collapsed}(A_i)$: we want to remove the procedure call nodes. We do this by a kind of unfolding, using at most one copy of the graph of each different procedure (recall that we have a finite number of them). Consider a node corresponding to a procedure, say `p`, with `p` :- A. If the graph associated to $\texttt{collapsed}(A)$ is not present in the graph we are manipulating, then we substitute the node for `p` with the graph of $\texttt{collapsed}(A)$, otherwise we remove the node for `p` and we redirect its incoming edges to the entering node of the graph of $\texttt{collapsed}(A)$. In Figure 1 (right) we show the result for the agent $\texttt{collapsed}(RW_X)$.

The resulting graph for each component of the network is called *reduced transition system* (RTS), as it contains all possible actions and all possible states of that component. We remark that an RTS for an agent of restricted sCCP is always finite. This property is a consequence of the fact that we disallowed the generation of local variables and the passing of parameters in procedure calls. Therefore, each called procedure executes always the same operations on the same (global) variables, hence its graph needs to be added in the RTS of a component only once. As the number of definable procedures is finite, so are the possible nodes of an RTS.

Step 2: the interaction matrix.

Consider a restricted sCCP program, composed by several agents in parallel. Once we have converted all of them into their RTS representation, we number 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 exit_j the variable labeling the exiting node of edge j, with $\text{rate}_j(X_1, \ldots, X_n)$ its rate function (labeling the corresponding edge in the RTS) and with $\text{guard}_j(X_1, \ldots, X_n)$ the indicator function of its guard, returning 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 $X_l = X_l + \delta$, we put δ in correspondence of row X_l. All other entries of the column are set to 0. We denote the interaction matrix by I and the element corresponding to the row associated to variable Y and to column j by $I(Y, j)$.

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. We associate an equation to each row of the matrix, 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):

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

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

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

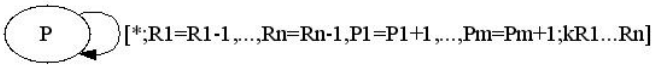
3.1 ODE's for Biochemical sCCP Agents

In Table 1 we have presented part of the library of sCCP agents describing the main biochemical reactions. Each agent in that list 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]. This result is presented here for the sCCP-encoding of one single reaction; extending it to a set of reactions is simply achieved exploiting compositionality, see [3] for further details.

Mass action kinetics.

Consider the sCCP encoding (Table 1) of a biochemical reaction with mass action kinetics, indicated by the arrow $X_1 + \ldots + X_n \rightarrow_k Y_1 + \ldots + Y_m$. 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 X_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$_{k,X_1,\ldots,X_n,Y_1,\ldots,Y_m}$ is the following



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

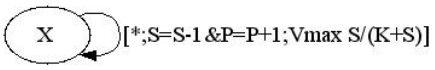
$$\begin{array}{lll} \dot{X}_1 = -kX_1 \cdots X_n & \dot{Y}_1 = kX_1 \cdots X_n & \\ \vdots & \vdots & \dot{P} = 0 \\ \dot{X}_n = -kX_1 \cdots X_n & \dot{Y}_m = kX_1 \cdots X_n & \end{array}$$

This is exactly the form of Mass Action ODE for this reaction: its speed is proportional to the concentration of the reagents, via the constant k.

Michaelis-Menten kinetics.

Let's consider a reaction based on Michaelis-Menten kinetics, represented in Table 1 by the chemical arrow $X \mapsto^{E}_{K,V_0} Y$.

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 [10]:

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

4 Ordinary Differential Equations to sCCP

In this section we define a transformation that associates an 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:

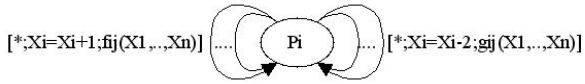
$$\begin{cases} \dot{X}_1 = \sum_{j=1}^{h_1} f_{1j}(X_1, \ldots, X_n) - \sum_{j=1}^{k_1} g_{1j}(X_1, \ldots, X_n) \\ \vdots \\ \dot{X}_n = \sum_{j=1}^{h_1} f_{1j}(X_1, \ldots, X_n) - \sum_{j=1}^{k_1} g_{1j}(X_1, \ldots, X_n) \end{cases} \tag{3}$$

In order to keep the presentation simple, we impose that $f_{ij}(X_1, \ldots, X_n) \geq 0$ and $g_{ij}(X_1, \ldots, X_n) \geq 0$.[4] The translation to sCCP simply proceeds associating an agent to each differential equation of (3), defined by

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

Here δ denotes the unitary basic increment. 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. Essentially, 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 man_{X_i}, we have first to obtain their reduced transition system. It is easy to see that it has the form



[4] The definition of the sCCP agents in the general case is straightforward: we simply need to check if the value of a function, say f is positive or negative and act accordingly, using its absolute value as the rate.

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 man_{X_i}, each entry being equal to δ or $-\delta$. The corresponding ODE therefore is

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

Now, we can perform two simplifications: first, δ 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 man_{X_i}, has equation $\dot{P}_i = 0$, hence P_i is constant. As we have just one agent 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 δ, determining 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 δ. Varying the size of δ, 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 β are here parameters; their value has a severe impact on the behavior of the system, that for some values of β oscillates (as expected from repressilator) while for some other values does not oscillate at all (see Figure 2). The corresponding sCCP process is

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

where

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

In Figure 2 we compare numerical solutions of S-Systems and simulations of the corresponding sCCP process, for different values of β (αs have the value defined

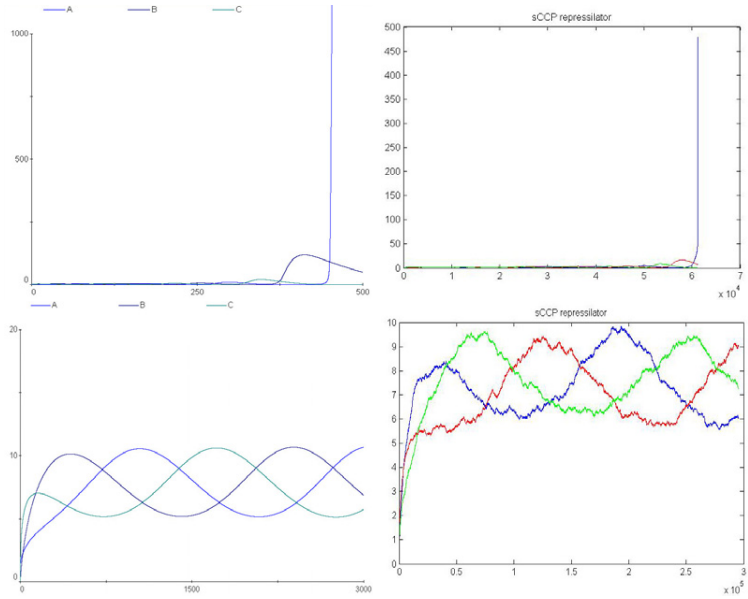


Fig. 2. Numerical solutions of the system of equations for the repressilator (left column), and the numerical simulation of the corresponding sCCP program (right column), for $\beta = 0.01$ (top) and $\beta = 0.001$ (bottom)

in equation (6)) and fixed δ, 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. 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. This can be explained by noting that in (4) δ plays the role of a scaling factor in the differential equation, resulting in a stretching of the temporal axis of a factor $\frac{1}{\delta}$.

5 Future Work and Conclusions

In the paper we focused on the problem of relating stochastic models written with process algebras and with differential equations. Even though 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 enabled us to define a "well-behaving" translation from general differential equations, in the sense that applying the (inverse) transformation from sCCP programs we obtain again

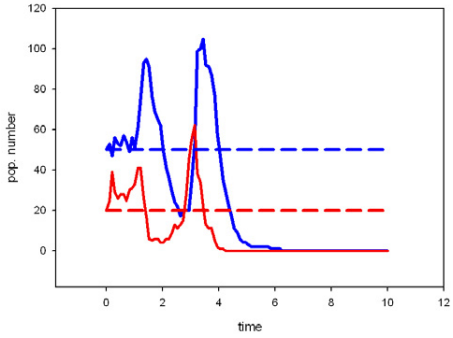


Fig. 3. Stochastic simulation of the Lotka Volterra model in sCCP (solid line) compared with the solution of associated ODE's (dotted line), for initial conditions $E_0 = 20$ and $C_0 = 50$. Predators are shown in blue and preys are shown in red.

our initial equations.

These translation procedures work at the syntactic level and in the paper we showed that, at least when applied to models of biochemical reactions, they preserve the intended chemical kinetics. Staten otherwise, the models obtained with this translation procedure are coherent with the notion of chemical kinetics used. However, there is no theoretical guarantee that the transformed model shows a *dynamical 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 in doing this: we are far from having defined a procedure translating stochastic process algebra programs into ODE's in a *semantically* correct way. We consider this as the main open problem in this research area. 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 [3,5]. Therefore, a different approach, aiming at capturing qualitative aspects more than quantitative information, should be adopted. A direction we are currently investigating is to use formulae written in a suitable temporal logic, expressing qualitative features of the dynamics, stating two models equivalent whenever they satisfy the same set of formulae.

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. A further interesting point is that of seeing if and where hybrid systems, like hybrid automata [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 sense what happens in the translation from ODE's to sCCP. Specifically, it would be important to understand how a variation of the step's value δ influences the behavior of the stochastic process w.r.t. the one of the ODE. Our conjecture is that in the limit of $\delta \to 0$, the average of the stochastic behavior coincides with the solution of the ODE (modulo a suitable time re-scaling), at least if the ODE's trajectories are not too sensitive from initial conditions, for the particular initial values chosen.

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. From Processes to ODEs by Chemistry. *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 π-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.