Dynamical Systems and Stochastic Programming: To Ordinary Differential Equations and Back

Luca Bortolussi[1] and Alberto Policriti[2,3]

[1] Dept. of Mathematics and Computer Science, University of Trieste, Italy.
luca@dmi.units.it
[2] Dept. of Mathematics and Computer Science, University of Udine, Italy
policriti@dimi.uniud.it
[3] Applied Genomics Institute (IGA), Udine, Italy
policriti@appliedgenomics.org

Abstract. In this paper we focus on the relation between models of biological systems consisting of ordinary differential equations (ODE) and models written in a stochastic and concurrent paradigm (sCCP stochastic Concurrent Constraint Programming). In particular, we define a method to associate a set of ODE's to an sCCP program and a method converting ODE's into sCCP programs. Then we study the properties of these two translations. Specifically, we show that the mapping from sCCP to ODE's preserves rate semantics for the class of biochemical models (i.e. chemical kinetics is maintained) and we investigate the invertibility properties of the two mappings. Finally, we concentrate on the question of behavioral preservation, i.e if the models obtained applying the mappings have the same dynamics. We give a convergence theorem in the direction from ODE's to sCCP and we provide several well-known examples in which this property fails in the inverse direction, discussing them in detail.

1 Introduction

The systemic approach to biology is nowadays a fertile and growing research area, considered by many as a promising track to the understanding of life [41, 1]. A key ingredient of systems biology resides in coupling wet lab experiments with mathematical modeling and analysis of bio-systems [33]. Many mathematical instruments have been used for this purpose, some concerned with qualitative analysis, others encapsulating also quantitative data [32]. Quantitative modeling is essentially dominated by two main mathematical tools: (ordinary) differential equations on one side and stochastic processes on the other [32]. Both these methods are concerned with the study of dynamical evolution of systems; however, they differ in the description of the quantities of interest: differential equations represent them as continuous variables, stochastic processes operate, instead, on discrete quantities. Modeling formalisms mixing discrete and continuous ingredients, like hybrid automata [29], have also been used in modeling bio-systems [2].

We will focus here mainly on the two former approaches, although commenting on the theme of the use of hybrid systems for restoring behavioral equivalence throughout the paper.

The theory of dynamical systems and differential equations (ODE's) is very attractive, being it a mature research area equipped with a huge set of analysis tools, ranging from static analysis of phase space topology to fast simulation via numerical integration [51, 43]. However, writing ODE's for a given system is generally a difficult task, requiring a considerable expertise. In addition, the representation of biological entities as continuous variables is an approximation that can sometimes be too rough, especially for low populations [26].

Stochastic processes like Continuous Time Markov Chains (CTMC) [39], on the other hand, do not suffer from these approximation limits, as they represent biological entities as discrete quantities, thus being more adherent to reality. On the other hand, analyzing a stochastic model is much more difficult, both from an analytical and from a computational point of view [53]. Regarding the description of stochastic models, recently we have seen the application of stochastic process algebras (SPA) [48, 45], a class of formal languages developed in theoretical computer science as formal tools to analyze (quantitatively) the performances of computing networks. These languages allow to build CTMC-based models following a simple, paradigmatic, identification of biological entities with (computing) processes. Moreover, they are *compositional*, allowing to build models by composing together sub-models.

Ideally, one would like to have a modeling technique that collects the advantages both of stochastic process algebras and differential equations, or, at least, to switch automatically between the two formalisms, depending on the particular task to be performed. In this direction, there are two related problems that must be faced: (a) studying the (mathematical) relation between the two modeling techniques and (b) finding automatic methods for converting one formalism into the other.

More specifically, we suggest the following workflow: first defining translation methods (for a specific process algebra), thus tackling (b), and then studying the mathematical relations intervening between the models obtained applying these translations. In this way we should be able to evaluate the appropriateness of conversion procedures between SPA and ODE's and to restrict the focus of the analysis required by (a).

There are two directions in the conversion between SPA and ODE: the first one associating a set of differential equations to a stochastic process algebra model, and the inverse one, mapping differential equations to stochastic process algebra programs. The first direction can be helpful for the analysis of SPA models, as ODE's can be solved and analyzed more efficiently. Associating SPA to ODE, instead, can help to clarify the logical pattern of interactions that are hidden in the mathematical structure of differential equations. Generally, as process algebra models can be written much more easily than differential equations, even by non-experts (possibly via a graphical interface), the first

direction, from SPA to ODE, looks potentially more fruitful, though having both mappings helps the study of the relationship between the two formalisms.

Supposing to have such transformations at our disposal, a crucial problem is to single out criteria to evaluate and validate them. The first possibility is to inspect the relationship intervening between a SPA program and the associated ODE's only from a mathematical point of view, forgetting any information about the system modeled. As both stochastic processes and differential equations are dynamical systems, this approach essentially corresponds to require that both models exhibit the same behavior, i.e. the same dynamical evolution. Of course, we may require agreement only from a qualitative point of view (so that the qualitative features of the dynamics are the same) or even from a quantitative one (numerical values agree). The difficulty with this approach is that stochastic processes have a noisy evolution, in contrast with the determinism characterizing differential equations. Hence, we need to remove the noise. One possibility is to look only at qualitative features of the dynamics, defining them in a precise way; we will go back to this problem in Section 3.2 below. Otherwise, we may average out noise from the stochastic models, thus considering the expected evolution of the system and requiring it to be described precisely (i.e. quantitatively) by the ODE's. Unfortunately, noise cannot be eliminated so easily, as sometimes it is the driving force of the dynamics [26, 52]. Therefore, this second form of equivalence is not completely justified; we will comment more on this point while discussing some examples in the following.

A different approach in comparing stochastic and differential models can be defined if we consider some additional information, which is external to the mathematics of the two models. The idea is to validate the translation w.r.t. this additional information. We explain this point with an example. Consider a model of a set of biochemical reactions; there are different chemical kinetic theories that can be used to describe such system, the most famous one being the principle of mass action. Using such a kinetic theory, we can build (in a canonical way) both a model based on differential equations and a model based on stochastic process algebras. If we are concerned with the principle of mass action more than with dynamical behavior, we may ask that our translation procedures preserves the former, meaning that the ODE's associated to a mass action SPA program are exactly the ODE's built according to mass action principle, and viceversa. Essentially, this corresponds to requiring that the translation procedures defined are *coherent with (some) principles of the system* modeled. For instance, in the case of mass action, coherency corresponds to preserve the meaning of rates (the so called *rate semantics* in [17]). Notice that in this case we are not requiring anything about dynamics, so coherent models may exhibit a divergent behavior, and this is indeed a well known issue, see, for instance, [26] or Section 3.2 below. Therefore, this comparison is essentially different from the behavioral-based one, and it is essentially syntactic, in the sense that it is concerned only with how models are written, not with their time evolution.

The operation of associating ODE's to SPA can be seen also as the definition of an ODE-based semantic for the stochastic processes, as opposed to the CTMC-

based one. Consequently, the comparison of the stochastic model with the derived ODE's can also be seen as an attempt to discover the mathematical relationship between these two semantics.

The problem of associating ODE's to stochastic process algebras has been tackled only recently in literature. The forefather is the work of Hillston [31], associating ODE's to models written in PEPA [30], a stochastic process algebra originally designed for performance modeling. Successively, similar methods have been developed for stochastic π-calculus [16, 11, 44] and for stochastic Concurrent Constraint Programming [7, 12]. All these methods build the ODE's performing a syntactic inspection and manipulation of the set of agents defining the SPA model. In fact, they all satisfy the coherency condition staten above, at least for mass action principle (a proof for stochastic π-calculus can be found in [17]). The inverse problem of associating SPA models to ODE's has received much less attention, the only example being [12], where we use stochastic Concurrent Constraint Programming as target SPA.

In this paper, we will retake the work previously done for sCCP in [12], presenting it in a more detailed and formal way. Basically, we will define two translation procedures: from sCCP to differential equations and viceversa. sCCP plays here a central role, thanks to some ingredients giving a noteworthy flexibility to it, the presence of functional rates above all. Therefore, in Section 2, we will recall the basics of sCCP and its application as a modeling language for biological systems, as presented in [15]. Further details on the language can be found in [8]. The translation procedure from sCCP to differential equations is presented formally in Section 3, while Section 4 is devoted to the presentation of the inverse mapping from general ODE's into sCCP. In Section 3, we will also show coherency conditions for a class of chemical kinetics and we will comment in detail the problem of behavioral equivalence in the conversion from sCCP to ODE's. This will be done mainly via examples, exhibiting biological systems for which the translation preservers also the behavior and other systems whose stochastic models show a different behavior than ODE's. The problem of behavioral equivalence is not new, and in fact some examples that we will give are famous ones [26]. However, the syntactic structure of process algebras in general, and sCCP specifically, give a new flavor to these classical examples, and brings the attention into new ones.
The issue of preservation of dynamic behavior in the mapping from ODE's to sCCP is tackled in Section 4. In this case we are able to exploit the structure of the mapping and thus to give a convergence theorem.

Throughout the paper, we will encounter several situations in which discreteness is a crucial ingredient for the dynamics of the system. This points to a third class of dynamical systems that is in the middle between SPA and ODE's and that can be used to approximate them, namely hybrid automata [29]. In [13, 14] we deal with the problem of mapping sCCP programs into hybrid automata, showing that in this case we are able to deal correctly from a behavioral viewpoint with a broader class of sCCP systems. The idea of such mapping is that of translating to ODE's *locally* while retaining some level of discreteness in the

$$
\begin{array}{c}
Program = D.A \\
D = \varepsilon \mid D.D \mid p(\mathbf{x}) : -A \\
\pi = \mathrm{tell}_\lambda(c) \mid \mathrm{ask}_\lambda(c) \\
M = \pi.G \mid M + M \\
G = \mathbf{0} \mid \mathrm{tell}_\infty(c).G \mid p(\mathbf{y}) \mid M \mid \exists_x G \mid G \parallel G \\
A = \mathbf{0} \mid \mathrm{tell}_\infty(c).A \mid M \mid \exists_x A \mid A \parallel A
\end{array}
$$

Table 1. Syntax of sCCP.

finite control of the hybrid automata. By the way, the method of [13] can be extended into a general framework encompassing also the mapping presented in this paper as a particular case.

2 Preliminaries

In this section we briefly recall the basics of stochastic Concurrent Constraint Programming (sCCP, Section 2.1) and its application as a modeling language for biological systems (Section 2.2). The interested reader is referred to [8] for further details. In Section 2.3, we introduce some restrictions on the language that greatly simplify the mapping from and to ODE's.

2.1 Stochastic Concurrent Constraint Programming

Concurrent Constraint Programming (CCP, [49]) 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 sort of global memory (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 [7, 15]) is obtained by adding a stochastic duration to all instructions interacting with the constraint store $\mathcal{C}$, i.e. `ask` and `tell`. Each instruction has an associated random variable representing time (thus taking values in the positive reals), 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 [30] or stochastic π-calculus [44] (although recently introduced in BioPEPA [18]), and it will be crucially used in the translation mechanisms. The syntax of sCCP can be found in Table 1.

$(IR1)$	$\text{tell}_\infty(c).A, dA, d \sqcup c$	
$(IR2)$	$p(\mathbf{x}), dA[\mathbf{x}/\mathbf{y}], d$	if $p(\mathbf{y}) :- A$
$(IR3)$	$\exists_x A, dA[y/x], d$	with y fresh
$(IR4)$	$\dfrac{A_1, dA_1', d'}{A_1 \parallel A_2, dA_1' \parallel A_2, d'}$	

Table 2. Instantaneous transition for stochastic CCP

Two different kind of actions are present in such table: stochastic actions, having a rate attached to them, and instantaneous actions, having an infinite rate. This second class of actions can be used to model the happening of complex atomic events, like a sequence of store updates happening instantaneously. However, only `tell` actions can happen instantaneously, and moreover they are always guarded by a stochastic action. The same restriction applies to recursive calls.

Operational Semantics. The definition of the operational semantics is given specifying two different kinds of transitions: one dealing with instantaneous actions and the other with stochastically timed ones. The basic idea of this operational semantics is to apply the two transitions in an interleaved fashion: first we apply the transitive closure of the instantaneous transition, then we do one step of the timed stochastic transition. To identify a state of the system, we need to take into account both the agents that are to be executed and the current state of the store. Therefore, a configuration will be a point in the space $\mathcal{P} \times \mathcal{C}$, where $\mathcal{P}$ is the space of agents and $\mathcal{C}$ is the space of all possible configurations of the constraint store.

The instantaneous transition $\longrightarrow \subseteq (\mathcal{P} \times \mathcal{C}) \times (\mathcal{P} \times \mathcal{C})$ and the stochastic transition $\Longrightarrow \subseteq (\mathcal{P} \times \mathcal{C}) \times [0,1] \times \mathbb{R}^+ \times (\mathcal{P} \times \mathcal{C})$ are defined according to the structural rules of Tables 2 and 3, respectively.

The fact that instantaneous actions and recursive calls are guarded by stochastic actions guarantees that $\longrightarrow$ can be applied only for a finite number of steps. Moreover, it can be proven to be confluent, see [8]. With the notation $\overrightarrow{\langle A, d \rangle}$ of Table 3, we denote by the configuration obtained by applying the transitions $\longrightarrow$ as long as it is possible (i.e., by applying the transitive closure of $\longrightarrow$). The confluence property of $\longrightarrow$ implies that $\overrightarrow{\langle A, d \rangle}$ is well defined.
The stochastic transition $\Longrightarrow$, instead, is labeled by two numbers: intuitively, the first one is the probability of the transition, while the second one is its global rate. Note that, after performing one step of the transition $\Longrightarrow$, we apply the transitive closure of $\longrightarrow$. This guarantees that all actions enabled after one $\Longrightarrow$ step are timed.

$$(SR1) \qquad \mathrm{tell}_\lambda(c).A, d_{(1,\lambda(d))} \overrightarrow{A, d \sqcup c}$$

$$(SR2) \qquad \mathrm{ask}_\lambda(c).A, d_{(1,\lambda(d))} \overrightarrow{A, d} \qquad \text{if } d \vdash c$$

$$(SR3) \qquad \frac{M_1, d_{(p,\lambda)} \overrightarrow{A_1', d'}}{M_1 + M_2, d_{(p',\lambda')} \overrightarrow{A_1', d'}}$$
$$\text{with } p' = \tfrac{p\lambda}{\lambda + \mathrm{rate}(M_2, d)} \text{ and } \lambda' = \lambda + \mathrm{rate}(M_2, d)$$

$$(SR4) \qquad \frac{A_1, d_{(p,\lambda)} \overrightarrow{A_1', d'}}{A_1 \parallel A_2, d_{(p',\lambda')} \overrightarrow{A_1' \parallel A_2, d'}}$$
$$\text{with } p' = \tfrac{p\lambda}{\lambda + \mathrm{rate}(A_2, d)} \text{ and } \lambda' = \lambda + \mathrm{rate}(A_2, d)$$

Table 3. Stochastic transition relation for stochastic CCP. The function rate : $\mathcal{P} \times \mathcal{C} \to \mathbb{R}$ assigns to each agent its global rate. Its effect is to recursively traverse the syntactic tree of agents, adding up the rates of active stochastic actions. Its formal definition can be found in [15].

Using relation $\Longrightarrow$, we can build a labeled transition system, whose nodes are configurations of the system and whose labeled edges correspond to derivable steps of $\Longrightarrow$. As a matter of fact, this is a multi-graph, as we can derive more than one transition connecting two nodes. Starting from this labeled graph, we can build a Continuous Time Markov Chain (cf. [39] and brlow) as follows: substitute each label (p, λ) with the real number $p\lambda$ and add up the numbers labeling edges connecting the same nodes.

Continuous Time Markov Chains. A Continuous Time Markov Chain (CTMC for short) [39] is a continuous-time stochastic process $(X_t)_{t \geq 0}$ taking values in a discrete set of states S and satisfying the memoryless property:

$$P\{X_{t_n} = s_n \mid X_{t_{n-1}} = s_{n-1}, \ldots, X_{t_1} = s_1\} = P\{X_{t_n} = s_n \mid X_{t_{n-1}} = s_{n-1}\}, \tag{1}$$

for each $n, t_1, \ldots, t_n, s_1, \ldots, s_n$.
A CTMC can be represented as a directed graph whose nodes correspond to the states of S and whose edges are labeled by real numbers, which are the rates of exponentially distributed random variables. In each state there are usually several exiting edges, competing in a race condition in such a way that the fastest one is executed. The time employed by each transition is drawn from the random variable associated to it. When the system changes state, it forgets its past activity and starts a new race condition (this is the *memoryless property*). Therefore, the traces of a CTMC are made by a sequence of states interleaved by variable time delays, needed to move from one state to another.

The time evolution of a CTMC can be characterized equivalently by computing, in each state, the normalized rates of the exit transitions and their sum (called the *exit rate*). The next state is then chosen according to the probability distribution defined by the normalized rates, while the time spent for the transition is drawn from an exponentially distributed random variable with parameter equal to the exit rate. This second characterization is at the basis of several stochastic simulation algorithms for CTMC, like the well-known Gillespie's one [26].

Stream Variables and Implementation. Some variables of the system, like those used in the definition of rate functions, need to store a single number that may vary over time. Such variables, for technical reasons, are conveniently modeled as variables of the constraint store, which, however, must be rigid (over time). To deal with this problem we store time varying parameters as growing lists with an unbounded tail variable. In order to avoid heavy symbolism, we will use a natural notation where $X' = X + 1$ has the intended meaning of[4]: "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*.

An interpreter for the language is available and 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.

2.2 Modeling Biological Systems in sCCP

In [8, 15] 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 [15], we showed that biochemical reactions and genetic regulatory networks are easily handled by sCCP. In [8] we added to this list also formation of protein complexes and the process of folding of a protein, whose description requires knowledge about spatial position of amino acids constituting the protein (a kind of information easily added building on expressive potential of the constraint store).
Finally, in [10] we showed how sCCP can be used to encode Kohn maps [34], a graphical formalism capable of describing implicitly biochemical networks subject to combinatorial explosion of the number of different kinds of protein complexes. In this case, the power of the constraint store is used to maintain a graph-based representation of complexes, allowing a *linear description* of Kohn Maps (i.e., the encoding requires a linear number of characters w.r.t. the ones needed to describe a Kohn map).

[4] The use of primed variables to denote values taken at the next time step is typical of model checking and is not to be confused with first derivatives (for which we will used dotted variables, as time is the only independent variable).

We recall now the modeling in sCCP of biochemical reactions. A general biochemical reaction has the form

$$R_1 + \ldots + R_n \rightarrow_{f(\mathbf{R},\mathbf{X};\mathbf{k})} P_1 + \ldots + P_m, \tag{2}$$

where $R_1, \ldots, R_n$ are the *reactants* and $P_1, \ldots, P_m$ are the *products*. The real-valued *kinetic function* of the reaction is $f(\mathbf{R}, \mathbf{X}; \mathbf{k})$, depending on the reactants $\mathbf{R}$, on other molecules $\mathbf{X}$ acting as modifiers, and on some parameters $\mathbf{k}$. This function can be one of the many used in biochemistry (cf. [20, 50]) and it is required to satisfy the following *boundary condition*: it must be zero whenever one reactant is less than its amount consumed by the reaction. For instance, if a reactant R appears two times in the left hand size of (2), then f must be zero for $R = 0, 1$. [5].
Biochemical networks can be easily modeled in sCCP taking a *reaction-centric* approach, where each reaction (or action capability) is associated to a process, while molecules, whose concentration varies over time, are represented by integer variables of the constraint store (actually, stream variables). Moreover, the presence of non-constant rates allows to describe reactions with arbitrary chemical kinetics. More specifically, to each reaction like (2), we associate the following sCCP agent:

$$\begin{aligned} &f\texttt{-reaction}(\mathbf{R}, \mathbf{X}, \mathbf{P}, \mathbf{k}) \text{ :-} \\ &\qquad \text{tell}_{f(\mathbf{R},\mathbf{X};\mathbf{k})}\big(\textstyle\bigwedge_{i=i}^{n}(R_i - 1) \wedge \bigwedge_{j=i}^{m}(P_j + 1)\big). \\ &\qquad f\texttt{-reaction}(\mathbf{R}, \mathbf{X}, \mathbf{P}, \mathbf{k}) \end{aligned}$$

This agent is a simple recursive loop, modifying the value of reactants' and products' variables at a speed given by the kinetic law. Note that the boundary conditions for the rate function f imply that no stream variable will ever become negative, as all reactions that may produce this effect have rate zero[6]. In Table 4 we give a list of some of the most common kinetics: mass action, Michaelis-Menten and Hill kinetics.

In order to describe genetic regulatory networks, instead, we use a modeling style mixing the *reaction-centric* point of view with the more classical *molecular-centric* one. Essentially, genes are described by sCCP agents, while proteins are associated to stream variables, like for biochemical reactions. An example of a genetic network can be found in Section 3.2. More information and examples on modeling biological systems in sCCP can be found in [15].

[5] In case of mass action kinetics, this condition means that the rate for $R + R \rightarrow P$ must be $kR(R-1)$ and not kR^2. This is, however, consistent with the definition of the mass action principle in the stochastic setting.

[6] Boundary conditions for f may be relaxed by checking explicitly with ask instructions that variables stay within their domain. For instance, for the reaction $R + R \rightarrow P$, we can precede `tell` by $\texttt{ask}(R > 1)$. This allows us to use the more common kR^2 as rate function.

$R \rightarrow_k P_1 + \ldots + P_m$	$f_{ma}(R;k) = kR$
$R_1 + R_2 \rightarrow_k P_1 + \ldots + P_m$	$f_{ma}(R_1, R_2; k) = kR_1R_2$
$R + R \rightarrow_k P_1 + \ldots + P_m$	$f_{ma}(R;k) = kR(R-1)$
$S \mapsto^E_{k,v} P$	$f_{MM}(S, E; k, v) = \frac{vES}{k+S}$
$S \mapsto^E_{K,V_0,h} P$	$f_{Hill}(S, E; h, k, v) = \frac{vES^h}{k+S^h}$

Table 4. List of some of the most common types of biochemical reaction, taken from [50]. The first three are first and second order mass-action-like reactions. The second arrow corresponds to a reaction with Michaelis-Menten kinetics. The last arrow replaces Michaelis-Menten kinetics with Hill's one (see [20]).

2.3 Restricted sCCP

The mapping between sCCP and ODE's is not defined for the whole sCCP language, but rather for a restricted version of it, which is, however, sufficient to describe biochemical reaction and genetic networks.

This restricted version of sCCP will be denoted in the following by RESTRICTED($sCCP$), and is formally specified by the following definition:

Definition 1. *A* RESTRICTED($sCCP$) *program is a tuple* $(Prog, \mathbf{X}, init(\mathbf{X}))$ *satisfying:*

1. *Prog is an sCCP-program respecting the grammar defined in Table 5.*
2. *The variables used in the definition of agents are taken from a finite set* $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ *of* global stream-variables*, each with the same domain* $\mathbb{D}$*, usually* $\mathbb{D} = \mathbb{N}$ *or, more generally,* $\mathbb{D} = \mathbb{Z}$*.*
3. *The only admissible updates for variables* $\{X_1, \ldots, X_n\}$ *are constraints of the form* $X_i = X_i + k$ *or* $X_i = X_i - k$*, with* $k \in \mathbb{D}$ *constant.*
4. *Constraints that can be checked by ask instructions are finite conjunctions of linear equalities and inequalities.*
5. *The initial configuration of the store is specified by the formula* $init(\mathbf{X})$*, consisting in the following conjunction of constraints:* $(X_1 = x_1^0) \wedge \ldots \wedge (X_n = x_n^0)$*, with the constants* $x_i^0 \in \mathbb{D}$ *referred to as the* initial values *of the sCCP-program.*

This definition can be justified looking at the sCCP-agent associated to a biochemical reaction and also at the sCCP-model of genes considered in [15]. In fact, in these cases all employed variables are numerical variables of the

stream-type[7], while all updates in the store add or subtract them a predefined constant quantity. Guards, instead, usually check if some molecules are present in the system ($X > 0$), though we consider here the more general case of linear equalities and inequalities. The use of global variables only, instead, can be justified noting that the existential operator $\exists_x$ is never used (neither in the reaction agent nor in gene models of [15]), as the scope of molecular interactions is system-wide. The suppression of the operator $\exists_x$, as a side consequence, guarantees that we can avoid to pass parameters to procedure calls: in fact, each procedure can be defined as operating on a specific subset of global variables. However, parameter passing is used in Section 2.2 to define parametrically the reaction agent. Therefore, we agree that each instance of a reaction agent, say $f\texttt{-reaction}(\mathbf{R}, \mathbf{X}, \mathbf{P}, \mathbf{k})$, is replaced with the corresponding ground form $f\texttt{-reaction}_{(\mathbf{R},\mathbf{X},\mathbf{P},\mathbf{k})}$. The same trick will be used for other agents. We demand further comments on the restrictions in Section 3.4.

In order to fix the notation in the rest of the paper, we give the following definition:

Definition 2. *A* RESTRICTED($sCCP$) *agent A not containing any occurrence of the parallel operator $\|$ is called a* sequential component *or a* sequential agent*. A* RESTRICTED($sCCP$) *agent N is called an* sCCP-network *if it is the parallel composition of sequential agents.*

Inspecting the grammar of Table 5, we can observe that the initial configuration of a RESTRICTED($sCCP$) program is indeed an sCCP-network. The following property is straightforward:

Lemma 1. *The number of sequential components forming an sCCP-program $(Prog, \mathbf{X}, init(\mathbf{X}))$ remains constant at run-time and equals the number of sequential agents in the sCCP-network of the initial configuration.*

Proof. As sequential components do not contain any parallel operator, no new agents can be forked at run-time.[8]

In the rest of the paper, for notational convenience, we usually identify an sCCP-program with the corresponding sCCP-network.
Moreover, forbidding the definition of local variables implies the following property:

Lemma 2. *The number of variables involved in the evolution of an sCCP-network*[9] *is a subset of $\{X_1, \ldots, X_n\}$, hence finite.*

[7] We do not need further types of variables, as we just need to count the number of different molecules in the system.

[8] We are counting also deadlocked agents.

[9] A variable is involved in the evolution of the network if one of the following things happen: it is updated in a tell instruction, it is part of a guard checked in an ask instruction, or it is used in the definition of a rate function.

$$
\begin{array}{cc}
Prog = Def.N & Def = \varepsilon \mid Def.Def \mid p : -A \\
\pi = \mathrm{tell}_\lambda(c) \mid \mathrm{ask}_\lambda(c) & M = \pi.G \mid M + M \\
G = \mathrm{tell}_\infty(c).G \mid p \mid M & A = \mathbf{0} \mid M \\
N = A \mid A \parallel N &
\end{array}
$$

Table 5. Syntax of the restricted version of sCCP.

The restrictions of RESTRICTED($sCCP$) are in the spirit of those introduced in [31]: we are forbidding an infinite unfolding of agents and we are considering global interactions only, forcing the speed of each action to depend on the whole state of the system. Indeed, also in [16] we find similar restrictions, though the comparison with sCCP is subtler. First of all, the version of π-calculus presented in [16] does not allow the use of the restriction operator, meaning that interactions have a global scope. However, agents in the π-calculus of [16] 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. What is finite in [16], however, is the number of syntactically different agents that can be present in a system.

On the Restrictions of the Language RESTRICTED($sCCP$) limits the full language in three main aspects: the allowance of sequential agents only, the suppression of local variables and the simplifications on the constraint store, cf. Definition 1. We will, however, comment here only on the first one, as the other two are forced by the translation to ODEs, hence their discussion will be postponed to Section 3.4.

As observed, in RESTRICTED($sCCP$) we constrain all the agents to be sequential, i.e. no occurrence of the parallel operator is allowed. Essentially, sequential agents are automata cooperating together, a property that will be exploited in the next section to represent them graphically in a simple way. Indeed, this restriction is only apparent: we can always convert a non-sequential agent into a network of sequential ones using additional (stream) variables of the constraint store. The idea is simply that of identifying all the syntactically different terms that are stochastic choices, associating a new variable to each of them. These variables are used to count the number of copies of each term that are in parallel. Each agents is modified consequently: all agents will only call recursively themselves, while the variations induced in the number of terms by transitions are dealt with by updating the new state variables. Finally, rates are corrected by multiplying them by the multiplicity variable associated to the agent executing the corresponding transition. This is justified by the fact that in Markovian models, the global rate of a set of actions is computed by adding all basic rates together—ultimately, a consequence of the properties of the exponential distribu-

tion [39]. For instance, consider the agents x and y, defined by x :- $\mathrm{tell}_1(true).(y \parallel y)$, y :- $\mathrm{tell}_1(true).x + \mathrm{tell}_1(true).\mathbf{0}$. They can be made sequential by introducing two variables, X and Y, counting the number of copies of x and y respectively and by replacing x by x':- $\mathrm{ask}_X(X > 0).\mathrm{tell}_\infty(X' = X - 1 \wedge Y' = Y + 2).x'$ and y by y' :- $\mathrm{ask}_Y(Y > 0).\mathrm{tell}_\infty(X' = X + 1 \wedge Y' = Y - 1).y' + \mathrm{ask}_Y(Y > 0).\mathrm{tell}_\infty(Y' = Y - 1).y'$.

3 From sCCP to ODE

In this section we define a translation method associating a set of ordinary differential equations to an sCCP program. This translation applies precisely to RESTRICTED($sCCP$), as defined above. The procedure is organized in several simple steps, illustrated in the following paragraphs. Essentially, we first associate a finite graph to each sequential component of an sCCP network and then, analyzing the graph, we define an interaction matrix similar to the one defined in [31] or to action matrices of (stochastic) Petri nets (see, for instance, [27]). Writing ODE's from this matrix is then almost straightforward.

After defining this translation, in Section 3.1 we investigate how it relates to biochemical kinetics and we show that the ODE's associated to an sCCP-model of a set of biochemical reactions are the ones generally considered in standard biochemical praxis [17, 20]. Some considerations on dynamical properties are then put forward in Section 3.2, while in Section 3.3 the focus is moved on the concept of behavioral equivalence. Finally, in Section 3.4 we reconsider the restrictions applied to sCCP in the light of the described transformation procedure.

Step 1: Reduced Transition Systems.

The first step consists in associating a labeled graph, called *reduced transition system* [8], to each sequential agent composing the network. As a working example, we consider the following simple sCCP agent:

$$
\begin{aligned}
\mathrm{RW}_X \text{ :-} & \\
& \mathrm{ask}_1(X > 0).\mathrm{tell}_\infty(X' = X - 1).\mathrm{RW}_X \\
& + \mathrm{tell}_1(X' = X + 2).\mathrm{RW}_X \\
& + \mathrm{ask}_{f(X)}(true).(\ \mathrm{ask}_1(X > 1).\mathrm{tell}_\infty(X' = X - 2).\mathrm{RW}_X \\
& \qquad + \mathrm{tell}_1(X' = X + 1).\mathrm{RW}_X
\end{aligned}
$$

$$
f(X) = \tfrac{1}{X^2+1}
$$

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.

Inspecting Table 5, where the syntax of RESTRICTED($sCCP$) is summarized, we observe that each branch of a stochastic choice starts with a stochastic timed instruction, i.e. an $\mathtt{ask}_\lambda(c)$ or a $\mathtt{tell}_\lambda(c)$, followed by zero or more $\mathtt{tell}_\infty(c)$,

$$
\begin{array}{c}
Prog = \hat{Def}.\hat{A} \\
\hat{Def} = \varepsilon \mid \hat{Def}.\hat{Def} \mid [\![p]\!] :- \hat{A} \\
\hat{\pi} = \text{action}(g, c, \lambda) \qquad \hat{M} = \hat{\pi};\hat{G} \mid \hat{M} \oplus \hat{M} \\
\hat{G} = [\![p]\!] \mid \hat{M} \qquad \hat{A} = [\![\mathbf{0}]\!] \mid \hat{M} \\
\hat{N} = \hat{A} \mid \hat{A} \parallel \hat{N}
\end{array}
$$

Table 6. Syntax of COMPACT($sCCP$).

followed by a procedure call or by another stochastic choice. The first operation that we need to perform, in order to simplify the structure of agents, is that of collapsing each timed instruction with all the instantaneous `tell` instructions following it and replacing everything with one "action" of the form

$$\texttt{action}(c, d, \lambda),$$

where c is a guard that must be entailed by the store for the branch to be entered, d is the constraint that will be posted to the store, and λ is the stochastic rate of the branch, i.e. a function $\lambda : \mathcal{C} \to \mathbb{R}^+ \cup \{\infty\}$. The presence of ∞ among possible values of λ is needed to simplify the treatment of instantaneous tells.

To achieve this goal, we formally proceed by defining a conversion function, named COMPACT, by structural induction on terms. The result of this function is that of transforming an agent written in RESTRICTED($sCCP$) into an agent of a simpler language, called COMPACT($sCCP$), where `ask` and `tell` are replaced by the instruction `action`. In order to distinguish between the two languages, we denote stochastic summation in COMPACT($sCCP$) by "$\oplus$", sequential composition by "**;**", and we surround procedure calls and nil agent occurrences by double square brackets "$[\![\cdot]\!]$". The syntax of COMPACT($sCCP$) is formally defined in Table 6; its constraint store, instead, follows the same prescriptions of Definition 1.

In defining the function COMPACT, we use a concatenation operator $\bowtie$ to merge instantaneous tells with the preceding stochastic action. Formally, COMPACT is defined as follows:

Definition 3. *The function* COMPACT: RESTRICTED*(sCCP)* $\to$ COMPACT*(sCCP) is defined by structural induction through the following rules:*

1. $\text{COMPACT}(\mathbf{0}) = [\![\mathbf{0}]\!]$;
2. $\text{COMPACT}(p) = [\![p]\!]$.
3. $\text{COMPACT}(\text{ask}_\lambda(c).G) = \text{action}(c, true, \lambda) \bowtie \text{COMPACT}(G)$.
4. $\text{COMPACT}(\text{tell}_\lambda(c).G) = \text{action}(true, c, \lambda) \bowtie \text{COMPACT}(G)$.
5. $\text{COMPACT}(\text{tell}_\infty(c).G) = \text{action}(true, c, \infty) \bowtie \text{COMPACT}(G)$.
6. $\text{COMPACT}(M + M) = \text{COMPACT}(M) \oplus \text{COMPACT}(M)$.

where $\infty : \mathcal{C} \to \mathbb{R}^+ \cup \{\infty\}$ *is defined by* $\infty(c) = \infty$*, for all* $c \in \mathcal{C}$.

We now define the concatenation operator $\bowtie$:

Definition 4. *The operator* $\bowtie$ *is defined by:*

1. $action(g, c, \lambda) \bowtie [\![p]\!] = action(g, c, \lambda); [\![p]\!]$.
2. $action(g, c, \lambda) \bowtie M = action(g, c, \lambda); M$.
3. $action(g_1, c_1, \lambda_1) \bowtie action(g_2, c_2, \lambda_2) = action(g_1 \wedge g_2, c_1 \wedge c_2, \min(\lambda_1, \lambda_2))$.

where $\min(\lambda_1, \lambda_2) : \mathcal{C} \to \mathbb{R}^+ \cup \{\infty\}$ *is defined by* $\min(\lambda_1, \lambda_2)(c) = \min\{\lambda_1(c), \lambda_2(c)\}$.

Going back to the agent RW_X previously defined; if we apply the function COMPACT to it, we obtain the following agent:

$$
\begin{array}{l}
\textsc{compact}(\mathrm{RW}_X) \text{ :-} \\
\qquad \mathrm{action}(X > 0, true, 1) \bowtie \mathrm{action}(true, X' = X - 1, \infty) \bowtie [\![\mathrm{RW}_X]\!] \\
\quad \oplus \quad \mathrm{action}(true, X' = X + 2, 1) \bowtie [\![\mathrm{RW}_X]\!] \\
\quad \oplus \quad \mathrm{action}(true, true, f(X)) \bowtie \\
\qquad (\quad \mathrm{action}(X > 1, true, 1) \bowtie \mathrm{action}(true, X' = X - 2, \infty) \bowtie [\![\mathrm{RW}_X]\!] \\
\qquad\ \oplus \mathrm{action}(true, X' = X + 1, 1) \bowtie [\![\mathrm{RW}_X]\!] \ )
\end{array}
$$

After the removal of $\bowtie$ operator according to the rules in Definition (4), the agent $\textsc{compact}(\mathrm{RW}_X)$ becomes a $\textsc{compact}(\mathrm{sCCP})$ agent:

$$
\begin{array}{l}
\textsc{compact}(\mathrm{RW}_X) = [\![\mathrm{RW}_X]\!] \text{ :-} \\
\qquad \mathrm{action}(X > 0, X' = X - 1, 1) \ ; \ [\![\mathrm{RW}_X]\!] \\
\quad \oplus \quad \mathrm{action}(true, X' = X + 2, 1) \ ; \ [\![\mathrm{RW}_X]\!] \\
\quad \oplus \quad \mathrm{action}(true, true, f(X)) \ ; \\
\qquad (\quad \mathrm{action}(X > 1, X' = X - 2, 1) \ ; \ [\![\mathrm{RW}_X]\!] \\
\qquad\ \oplus \mathrm{action}(true, X' = X + 1, 1) \ ; \ [\![\mathrm{RW}_X]\!] \ )
\end{array}
$$

The above example shows clearly that the function COMPACT simply collapses all the actions performed on the store after one execution of the stochastic transition, as defined in [7, 8]. It is a simple exercise to define a stochastic transition relation for $\textsc{compact}(sCCP)$ similar to the one for $\textsc{restricted}(sCCP)$ and to successively prove the strong equivalence [30] between agents A and $\textsc{compact}(A)$.[10] Hence, from a semantic point of view, the application of function COMPACT is safe, as stated in the following

Lemma 3. *For each sequential agent* A *of* $\textsc{restricted}(sCCP)$, A *and* $\hat{A} = \textsc{compact}(A)$ *are strongly equivalent.*

Let $\hat{A} = \textsc{compact}(A)$ be an agent of $\textsc{compact}(sCCP)$. We want to associate a graph to such agent, containing all possible actions that $\hat{A}$ may execute. Nodes in such graph will correspond to different internal states of $\hat{A}$, i.e. to different stochastic branching points. Edges, on the other hand, will be associated to actions: each edge will correspond to one $\mathrm{action}(g, c, \lambda)$ instruction and will be labeled consequently by the triple (g, c, λ).

To define such graph, we proceed in two simple steps:

[10] Strong equivalence [30] is a form of bisimulation preserving probabilities: two agents are strongly equivalent if their exit rates are the same and transitions of one agent can be matched by transitions of the other having the same probability.

1. First we define an equivalence relation $\equiv_c$ over the set of COMPACT($sCCP$) agents, granting associativity and commutativity to $\oplus$ and reducing procedure calls to automatic "macro-like" substitutions (a reasonable move as we do not pass any parameter). We will then work on the set $\mathcal{A}$ of COMPACT($sCCP$) agents modulo $\equiv_c$, called the set of *states*; notably, all agents in $\mathcal{A}$ are stochastic summations.
2. Then, we define a structural operational semantics [42] on COMPACT($sCCP$), whose labeled transition system (LTS) will be exactly the target graph.

Definition 5. *The equivalence relation $\equiv_c$ between* COMPACT($sCCP$) *agents is defined as the minimal relation closed with respect to the following three rules:*

1. $\hat{M}_1 \oplus \hat{M}_2 \equiv_c \hat{M}_2 \oplus \hat{M}_1$;
2. $\hat{M}_1 \oplus (\hat{M}_2 \oplus \hat{M}_3) \equiv_c (\hat{M}_1 \oplus \hat{M}_2) \oplus \hat{M}_3$;
3. $[\![p]\!] \equiv_c \hat{A}$ *if* $[\![p]\!] : -\hat{A}$ *belongs to the declarations* $\hat{D}$.

The space of COMPACT($sCCP$) *agents modulo $\equiv_c$ is denoted by $\mathcal{A}$, and is referred to as the* space of states.

Definition 6. *The transition relation $\rightsquigarrow \subseteq \mathcal{A} \times (\mathcal{C} \times \mathcal{C} \times \mathbb{R}^{\mathcal{C}}) \times \mathcal{A}$ is defined in the SOS style as the minimal relation closed with respect to the following rule:*

$$action(g, c, \lambda); \hat{G} \oplus \hat{M} \overset{(g,c,\lambda)}{\rightsquigarrow} \hat{G}.$$

The transition relation $\rightsquigarrow$ encodes the possible actions that a COMPACT($sCCP$) agent can undertake. Notice that procedure calls are automatically solved as we are working modulo $\equiv_c$. The relation $\rightsquigarrow$ induces a labeled graph, its labeled transition system (LTS), whose nodes are agents in $\mathcal{A}$ and whose edges are labeled by triples (g, c, λ), where $g \in \mathcal{C}$ is a guard, $c \in \mathcal{C}$ is the update of the store, and λ the functional rate of the edge.

Definition 7. *Let $\hat{A}$ be an agent of* COMPACT($sCCP$)*; the portion of the labeled transition system reachable from the state $\hat{A}$ is denoted by $LTS(\hat{A})$.*

Theorem 1. *For any agent $\hat{A}$ of* COMPACT($sCCP$) *(modulo $\equiv_c$), $LTS(\hat{A})$ is finite.*

Proof. The agents reachable from $\hat{A}$ are subagents of $\hat{A}$ or subagents of $\hat{A}'$, where $[\![p]\!] : -\hat{A}'$ is a procedure called by an agent reachable from $\hat{A}$. The number of subagents of $\hat{A}$ (modulo $\equiv_c$) corresponds to the number of summations present in $\hat{A}$, and it is finite for any definable agent. The proposition follows because there is only a finite number of agents defined in the declarations $\hat{D}$.

We are finally ready to define the reduced transition system for an agent A of RESTRICTED(sCCP).

Definition 8. *Let A be an agent of* RESTRICTED*(sCCP). Its* reduced transition system *$RTS(A)$ is a finite labeled multigraph $(S(A), T(A), \ell_A)$ defined by*

$$RTS(A) = LTS(\text{COMPACT}(A)).$$

Given $RTS(A) = (S(A), T(A), \ell_A)$, $S(A) \subseteq \mathcal{A}$ is the set of *RTS-states* reachable from agent $\text{COMPACT}(A)$, finite for Theorem 1, $T(A)$ is the set of *RTS-edges* or *RTS-transitions* and $\ell_A : T(A) \to \mathcal{C} \times \mathcal{C} \times \mathbb{R}^{\mathcal{C}}$ is the label function assigning to each RTS-edge the triple (g, c, λ), $g, c \in \mathcal{C}$, $\lambda : \mathcal{C} \to \mathbb{R}^+$.

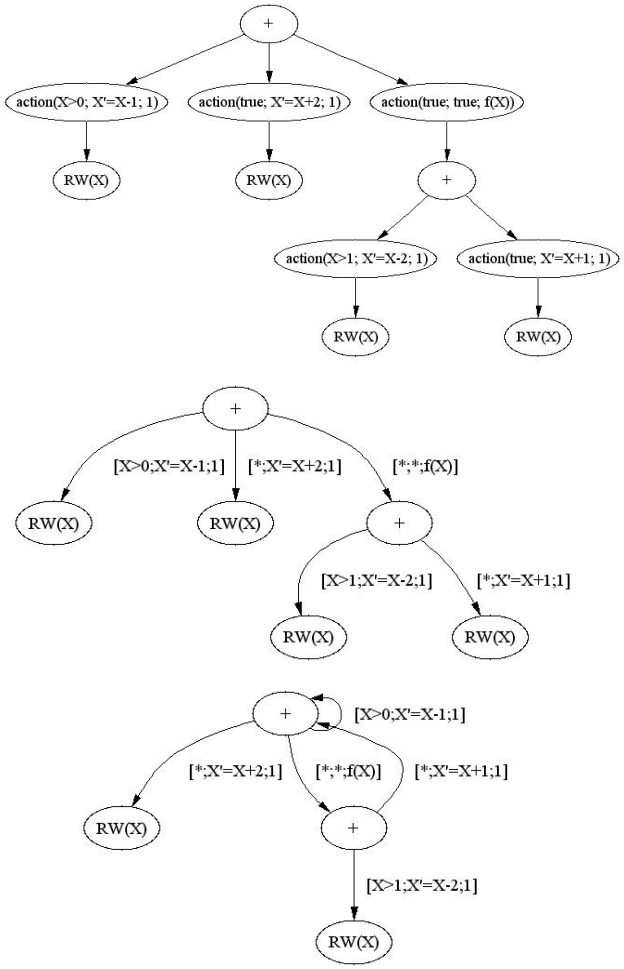


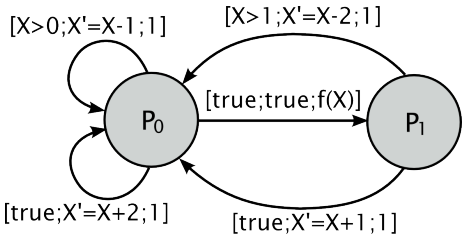
Fig. 1. Three intermediate steps of the construction of RTS for the agent $\text{RW}(X)$, according to the procedure sketched in the main body of the paper. The top one is the outcome of point 1, the middle one is the labeled tree obtained after step 2, while the bottom one is the result of applying twice the rule of step 3.

In order to effectively compute the RTS of an agent $\hat{A}$ of COMPACT($sCCP$), we can proceed as follows:

1. Given an COMPACT($sCCP$) program, write the syntactic tree of the agent $\hat{A}$ and of all the agents $\hat{A}'$ such that $[\![p]\!]$:-$\hat{A}'$ is in $\hat{Def}$.
2. Nodes corresponding to action(g, c, λ) instructions will have one single incoming edge and one single outgoing edge. Remove them, connecting the entering and exiting edges and labeling them by the triple c, g, λ).
3. Leaves of the obtained labeled tree correspond either to nil agents or to procedure calls. The latter are replaced according to the following rule: if the syntactic tree of the procedure $[\![p]\!]$ has not been added in the current tree, replace the leaf labeled with $[\![p]\!]$ with the corresponding syntactic tree; otherwise, remove the leaf and redirect the incoming edge to the root of the copy of the syntactic tree of $[\![p]\!]$. Iterate the application of the rule until no more leaves corresponding to procedure calls are available.

The previous procedure always terminates, as the number of different procedures $[\![p]\!]$ in $\hat{Def}$ is finite, hence the algorithm needs to process only a finite number of leaves.

Going back to our running example, $RTS(\mathrm{RW}_X) = LTS(\text{COMPACT}(\mathrm{RW}_X))$ is shown in the figure below. Note that it has one RTS-edge for every action that can be performed by COMPACT(RW_X), and just two RTS-states, corresponding to the two summations present in COMPACT(RW_X). Three intermediate steps in the construction of the RTS can be visualized in Figure 1.



Step 2: the interaction matrix.

Our next step consists in encoding all the information about the dynamics in a single *interaction matrix* and in a *rate vector*. Consider the initial sCCP-network $N = A_1 \parallel \ldots \parallel A_h$ of a RESTRICTED($sCCP$) program $(Prog, \mathbf{X}, init(\mathbf{X}))$, with sequential components $A_1, \ldots, A_h$. First of all, we construct the reduced transition system for all the components, i.e. $RTS(A_1) = (S(A_1), T(A_1), \ell_{A_1}), \ldots,$ $RTS(A_h) = (S(A_h), T(A_h), \ell_{A_h})$. Then we construct the set of RTS-states and RTS-transitions of the network (we agree that states and transitions belonging to different components $A_1, \ldots, A_h$ are distinct[11]), putting:

$$S(N) = S(A_1) \cup \ldots \cup S(A_h) \tag{3}$$

[11] If the same component is present in multiple copies, we distinguish among them by suitable labels.

and

$$T(N) = T(A_1) \cup \ldots \cup T(A_h).^{12} \tag{4}$$

Suppose now that there are m RTS-states in $S(N)$ and k RTS-transitions in $T(N)$. We conveniently fix an ordering of these two sets, say $S(N) = \{\sigma_1, \ldots, \sigma_m\}$ and $T(N) = \{t_1, \ldots, t_k\}$.

The variables $\mathbf{Y}$ of the differential equations are of two different kinds, $\mathbf{Y} = \mathbf{X} \cup \mathbf{P}$. The first type corresponds to the global stream variables of the store, i.e $\mathbf{X} = \{X_1, \ldots, X_n\}$ (see Definition 1). In addition, we associate a variable of the second type $P_{\sigma_i} = P_i$ to each RTS-state of $S(N) = \{\sigma_1, \ldots, \sigma_m\}$, so $\mathbf{P} = \{P_1, \ldots, P_m\}$. For the manipulations to follow, we assume the existence of a lexicographic ordering among all variables, so that vectors and matrices depending on this ordering are defined uniquely and manipulated consistently. Moreover, variables will be also used to index of these objects.

Consider now an RTS-transition $t_j \in T(N)$, connecting RTS-states σ_{j_1} and σ_{j_2}, and suppose $\ell_N(t_j) = (g_j, c_j, \lambda_j)$. We introduce the following notation:

- $\text{rate}_j^N(\mathbf{X}) = \lambda_j(\mathbf{X})$ is the rate function of t_j;
- $\text{guard}_j^N(\mathbf{X})$ is the indicator function of g_j (by Definition 1, g_j is a conjunction of linear equalities and inequalities), i.e.

$$\text{guard}_j^N(\mathbf{X}) = \begin{cases} 1 \text{ if } g_j \text{ is true for } \mathbf{X}, \\ 0 \text{ otherwise.} \end{cases} \tag{5}$$

We are now able to define the *rate vector*:

Definition 9. *The* rate vector r^N *for transitions* $T(N) = \{t_1, \ldots, t_k\}$ *is a* k*-dimensional vector of functions, whose components* r_j^N *are defined by*

$$r_j^N(\mathbf{Y}) = \text{rate}_j^N(\mathbf{X}) \cdot \text{guard}_j^N(\mathbf{X}) \cdot P_{j_1}, \tag{6}$$

where P_{j_1} *is the variable associated to the source state* σ_{j_1} *of transition* t_j*.*

Consider again a transition $t_j \in T(N)$, going from σ_{j_1} to σ_{j_2} and with label $\ell_N(t_j) = (g_j, c_j, \lambda_j)$, and consider the updates c_j, a conjunction of constraints of the form $X_i = X_i \pm k$, according to Definition 1. We can assume that each variable X_i appears in at most one conjunct of c_j.[13] We are now ready to define the *interaction matrix*

Definition 10. *The* interaction matrix $I_{\mathbf{Y}}^N$ *for an sCPP-network* N *with respect to variables* $\mathbf{Y}$ *is an integer-valued matrix with* $n+m$ *rows (one for each variable of* $\mathbf{Y}$*) and* k *columns (one for each RTS-transition* $T(N)$*), defined by:*

1. *If* $\sigma_{j_1} \neq \sigma_{j_2}$*, then* $I_{\mathbf{Y}}^N[P_{j_1}, t_j] = -1$ *and* $I_{\mathbf{Y}}^N[P_{j_2}, t_j] = 1$*.*
2. *If* $X_h = X_h \pm k$ *is a conjunct of* c_j*, then* $I_{\mathcal{Y}}^N[X_h, t_j] = \pm k$*.*

[12] The labeling function acting on $T(N)$ will be denoted consistently by ℓ_N.

[13] If, for instance, both $X_i = X_i + k_1$ and $X_i = X_i + k_2$ are in c_j. Then we can replace these two constraints with $X_i = X_i + (k_1 + k_2)$.

3. All entries not set by points 1,2 above are equal to zero.

For the agent RW_X, the interaction matrix $I_{\mathbf{Y}}^{\mathrm{RW}_X}$ for the variables $\mathbf{Y} = \{X, P_0, P_1\}$ is:

$$I_{\mathbf{Y}}^{\mathrm{RW}_X} = \begin{pmatrix} -1 & +2 & 0 & -2 & +1 \\ 0 & 0 & -1 & +1 & +1 \\ 0 & 0 & +1 & -1 & -1 \end{pmatrix} \begin{matrix} (X) \\ (P_0) \\ (P_1) \end{matrix} \tag{7}$$

Similarly, the rate vector r^{RW_X} is

$$r^{\mathrm{RW}_X} = \big(P_0\langle X > 0\rangle,\, P_0,\, f(X)P_0,\, P_1\langle X > 1\rangle,\, P_1, \big)^T, \tag{8}$$

where $\langle\cdot\rangle$ denotes the logical value of a formula (i.e., 1 if the formula is true, 0 otherwise).

Step 3: writing ODE's.

Once we have the interaction matrix, writing the set of ODE's is very simple: we just have to multiply matrix $I_{\mathbf{Y}}^N$ by the (column) rate vector r^N, in order to obtain the vector $ode_{\mathbf{Y}}^N$:

$$ode_{\mathbf{Y}}^N = I_{\mathbf{Y}}^N \cdot r^N. \tag{9}$$

Each row of the $ode_{\mathbf{Y}}^N$ vector gives the differential equation for the corresponding variable. Specifically, the equation for variable Y_i is

$$\begin{aligned} \dot{Y}_i = ode_{\mathbf{Y}}^N[Y_i] &= \sum_{j=1}^{k} I_{\mathbf{Y}}^N[Y_i, j] \cdot r_j^N(\mathbf{Y}) \\ &= \sum_{j=1}^{k} \big(I_{\mathbf{Y}}^N[Y_i, j] \cdot \mathrm{guard}_j(\mathbf{X}) \cdot \mathrm{rate}_j(\mathbf{X}) \cdot P_{j_1} \big) \end{aligned}$$

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

$$\begin{cases} \dot{X} = P_0(2 - \langle X > 0\rangle) + P_1(1 - 2\langle X > 1\rangle) \\ \dot{P}_0 = -\frac{1}{X^2+1} P_0 + P_1(1 + \langle X > 1\rangle) \\ \dot{P}_1 = \frac{1}{X^2+1} P_0 + P_1(1 + \langle X > 1\rangle) \end{cases}$$

In order to solve a set of ODE's, we need to fix the *initial conditions*. The variables $\mathbf{Y} = \mathbf{X} \cup \mathbf{P}$ of $ode_{\mathbf{Y}}^N$ are of two distinct types: $\mathbf{P}$, denoting states of the reduced transition systems of the components, and $\mathbf{X}$, representing stream variables of the store. The initial conditions for $\mathbf{P}$ are easily determined: we set to one all the variables corresponding to the initial states of RTS of each component, and to 0 all the others. Regarding $\mathbf{X}$, instead, initial conditions are given in the formula $init(\mathbf{X})$ of Definition 1, specifying the values assigned to stream variables before starting the execution of the sCCP program.

Elimination of redundant state variables.

Consider an sCCP component A whose reduced transition system $RTS(A)$ has just one RTS-state. Then, the $ode^N_{\mathbf{Y}}$ vector of an sCCP-network N having A as one of its components will contain a variable corresponding to this RTS-state, say P_i, with equation $\dot{P}_i = 0$ and initial value $P_i(0) = 1$. Clearly, such variable is redundant, and we can safely remove it by setting $P_i \equiv 1$ in all equations containing it and by eliminating its equation from the $ode^N_{\mathbf{Y}}$ vector. From now on, we assume that this simplification has always been carried out. As an example, consider the following agent

$$\begin{array}{ll} \text{A} :\text{-} & \text{tell}_{f_1(X)}(X = X + 1).\text{A} \\ + & \text{tell}_{f_2(X)}(X = X - 1).\text{A} \end{array}$$

Its RTS contains just one state, corresponding to the only summation present in it, with associated variable P. As the other variable of the agent is X, the vector $ode^A_{\{X,P\}}$ contains two equations, namely

$$\begin{pmatrix} \dot{X} \\ \dot{P} \end{pmatrix} = \begin{pmatrix} f_1(X)P - f_2(X)P \\ 0 \end{pmatrix}.$$

The simplification introduced above just prescribes to remove the equation for P, setting its value to 1 in the other equations; therefore we obtain

$$ode^A_{\{X\}} = (f_1(X) - f_2(X)).$$

Notice that the set of variables $\mathbf{Y}$ is updated consistently, i.e. removing the canceled variables from it.

We summarize the whole method just presented defining the following operator

Definition 11. *Let N be the sCCP-network of an* RESTRICTED($sCCP$) *program. With*

$$ODE(N)$$

we denote the vector $ode^N_{\mathbf{Y}}$ associated to N by the translation procedure previously defined, after applying the removal of state variables coming from network components with just one RTS-state.

Compositionality of the transformation operator

In order to clearly state formal properties of the transformation, we need a version of the $ODE(N)$ indicating explicitly the variables $\mathbf{X}$ for which the differential equations are given. In the following, the variables for the equations $ODE(N)$ are indicated by $VAR(ODE(N))$.

Definition 12. *Let N be the sCCP-network of an* RESTRICTED($sCCP$) *program and let $\mathbf{Y} = VAR(ODE(N))$ and $ODE(N) = ode^N_{\mathbf{Y}}$. The ordinary differential equations of N with respect to the set of variables $\mathbf{X}$, denoted by $ODE(N, \mathbf{X})$, is defined as*

$$ODE(N, \mathbf{X})[X_i] = \begin{cases} ode^N_{\mathbf{Y}}[Y_j] & \text{if} X_i = Y_j \in \mathbf{Y}, \\ 0 & \text{otherwise.} \end{cases}$$

The operations performed on $ODE(N)$ by $ODE(N, \mathbf{X})$ simply consist in the elimination of the equations of $ODE(N)$ for the variables not in $\mathbf{X}$, and in the addition of equations $\dot{X} = 0$ for all variables X in $\mathbf{X}$ but not in $\mathbf{Y}$. We can also associate a new interaction matrix $I^N_{\mathbf{X}}$ to $ODE(N, \mathbf{X})$, whose rows are derived according to Definition 12. As the set of RTS-transitions $T(N)$ is unaltered by $ODE(N, \mathbf{X})$, the equation $ODE(N, \mathbf{X}) = I^N_{\mathbf{X}} \cdot r^N$ continues to hold.

We can now prove the following theorem, stating compositionality of ODE operator.

Theorem 2. *Let N_1, N_2 be two sCCP-networks, and let $N = N_1 \parallel N_2$ be their parallel composition. If $\mathbf{Y_1} = VAR(ODE(N_1))$, $\mathbf{Y_2} = VAR(ODE(N_2))$, and $\mathbf{Y} = \mathbf{Y_1} \cup \mathbf{Y_2}$, then*[14]

$$ODE(N_1 \parallel N_2, \mathbf{Y}) = ODE(N_1, \mathbf{Y}) + ODE(N_2, \mathbf{Y}).$$

Proof. The components in $N_1 \parallel N_2$ are the components of N_1 plus the components of N_2. Therefore, the set of RTS-transitions (i.e. edges in the RTS of the components) $T(N_1 \parallel N_2)$ of $N_1 \parallel N_2$ is equal to $T(N_1) \cup T(N_2)$. As each column of $I^{N_1 \parallel N_2}_{\mathbf{Y}}$ is either a transition of $T(N_1)$ or a transition of $T(N_2)$, it clearly holds $I^{N_1 \parallel N_2}_{\mathbf{Y}}[Y_i, t_j] = I^{N_h}_{\mathbf{Y}}[Y_i, t_j]$ if $t_j \in T(N_h)$, $h = 1, 2$. The following chain of equalities then follows easily from the definitions:

$$\begin{aligned} ODE(N_1 \parallel N_2, \mathbf{Y})[Y_i] &= \sum_{t_j \in T(N_1 \parallel N_2)} I^{N_1 \parallel N_2}_{\mathbf{Y}}[Y_i, t_j] r_j^{N_1 \parallel N_2} \\ &= \sum_{t_j \in T(N_1)} I^{N_1}_{\mathbf{Y}}[Y_i, t_j] r_j^{N_1} + \sum_{t_j \in T(N_2)} I^{N_2}_{\mathbf{Y}}[Y_i, t_j] r_j^{N_2} \\ &= ODE(N_1, \mathbf{Y}) + ODE(N_2, \mathbf{Y}). \end{aligned}$$

3.1 Preservation of Rate Semantics

In Section 2.2 we discussed how biochemical reaction with general kinetic laws can be modeled in sCCP. Given a list of reactions, the standard praxis in computational chemistry is that of building a corresponding differential (set of ODE's) or stochastic (CTMC) model. The definition of such models is *canonical*, and it is fully specified by the reaction list, see [53] for further details.
sCCP models of biochemical reactions are generated by associating an agent,

[14] Here "+" denotes the usual sum of vectors.

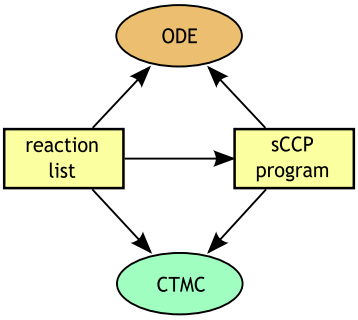


Fig. 2. Diagram of relations intervening between stochastic and ODE-based semantics of chemical reactions and sCCP agents.

call it *biochemical agent*, to each reaction in the list. These agents are rather simple: they can execute in one single way, namely an infinite loop consisting of activation steps, where the agents compete stochastically for execution, with rate given by the kinetic law specified in the reaction arrow, and update steps, in which the store is modified according to the prescriptions of the reaction. This bijective mapping between reactions and sCCP biochemical agents soon implies that the stochastic model for sCCP is identical to the continuous-time Markov chain generated in classical stochastic simulations with Gillespie algorithm [26], for instance like those obtainable with a program like Dizzy [46, 19].[15] A different question is wether the ODE's that are associated to an sCCP model of biochemical reactions coincide with the canonical ones. In the rest of the section, we show that this is indeed the case.

Following the approach by Cardelli in [17], we can then say that the translation from sCCP to ODE's *preserves the rate semantics*. The sense of this sentence is better visualized in Figure 2, graphically depicting the correspondence between stochastic and differential models of biochemical reactions and of the derived sCCP agents. Preservation of rate semantics essentially means that the arrows in the diagram commute.

As a matter of fact, in [17] the author deals only with mass action kinetics, due to the intrinsic properties of π-calculus (all definable rates are mass-action like). In our setting, instead, functional rates and the constraint store allow us to deal with arbitrary chemical kinetics, including also Michaelis-Menten and Hill ones (cf. [20]). In the following, we formally prove the equivalence of ODE's obtained from sCCP agents with the corresponding classical ones.

In general, the stochastic and the deterministic rate of a reaction are not the same, because ODE's variables measure concentration, while sCCP variables

[15] Stochastic simulations with Michaelis-Menten and Hill rate functions have been considered, for instance, in [47].

count the number of molecules. Therefore, in passing from one model to the other, we need to convert numbers to concentrations, dividing by the volume V times the Avogadro number N_A ($\gamma = VN_A$ will be referred as *system size*). Rates need also to be scaled consistently, see [53] for further details. In the rest of this section, however, we get rid of scaling problems simply by assuming $\gamma = 1$. In any case, system size can be reintroduced without difficulties, by change the scale of rates and variables after the derivation of ODE's.

We now put forward some notation, in order to specify how to formally derive a set of ODE's given a set of reactions $\mathcal{R} = \{\rho_1, \ldots, \rho_k\}$, where each ρ_i denotes a single reaction. Each reaction ρ has some attributes: a multiset of reactants (species can have a specific multiplicity), denoted by $REACT(\rho)$, a multiset of products, $PROD(\rho)$, and a real-valued rate function, $RATE(\rho)$, depending on the variables associated to the molecules involved in the reaction, $VAR(\rho)$. This last function, VAR, can be easily extended to sets of reactions by letting $VAR(\{\rho_1, \ldots, \rho_k\}) = VAR(\rho_1) \cup \ldots \cup VAR(\rho_k)$.

Now let $\mathcal{R}$ be a set of reactions and $\mathbf{X} = VAR(\mathcal{R})$. The (canonical) differential equations associated to $\mathcal{R}$ w.r.t. variables $\mathbf{X}$, denoted by $ODE(\mathcal{R}, \mathbf{X})$[16] are defined for each variable X_i as $\dot{X}_i = ODE(\mathcal{R}, \mathbf{X})[X_i]$, where:[17]

$$ODE(\mathcal{R}, \mathbf{X})[X_i] = \sum_{\substack{\rho \in \mathcal{R}: \\ X_i \in PROD(\rho)}} RATE(\rho) - \sum_{\substack{\rho \in \mathcal{R}: \\ X_i \in REACT(\rho)}} RATE(\rho). \quad (10)$$

If X_i is not involved in any reaction of $\mathcal{R}$, then $ODE(\mathcal{R}, \mathbf{X})[X_i] = 0$. We observe that the correct stoichiometry is automatically dealt with by the fact that we are using multisets to list reactants and products. Restricting this construction to a fixed variable ordering allows to state the following straightforward compositionality lemma.

Lemma 4. *Let $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ be a partition of $\mathcal{R}$ and let $\mathbf{X}_1 = VAR(\mathcal{R}_1)$, $\mathbf{X}_2 = VAR(\mathcal{R}_2)$, and $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2$. Then*

$$ODE(\mathcal{R}_1 \cup \mathcal{R}_2, \mathbf{X}) = ODE(\mathcal{R}_1, \mathbf{X}) + ODE(\mathcal{R}_2, \mathbf{X}).$$

We turn now to formally define the encoding of reactions into sCCP agents. For each reaction ρ, its sCCP agent $SCCP(\rho)$ is constructed according to Section 2.2. Operator $SCCP$ is extended compositionally to sets of reactions $\mathcal{R} = \{\rho_1, \ldots, \rho_k\}$ by letting $SCCP(\mathcal{R}) = SCCP(\rho_1) \parallel \ldots \parallel SCCP(\rho_k)$.

We are finally ready to state the theorem of preservation of rate semantics:

Theorem 3 (Preservation of rate semantics). *Let $\mathcal{R}$ be a set of biochemical reactions, with $\mathbf{X} = VAR(\mathcal{R})$. Then*

$$ODE(\mathcal{R}, \mathbf{X}) = ODE(SCCP(\mathcal{R}), \mathbf{X}) \quad (11)$$

[16] We overload here the symbol introduced in Definitions 11 and 12; however, the two cases can be easily distinguished looking at their first argument.

[17] We conveniently identify each variable with the molecule it represents.

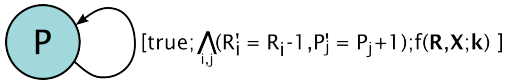
Proof. We prove the theorem by induction on the size k of the set of reactions $\mathcal{R}$. For the base case $k = 1$, consider a reaction ρ

$$R_1 + \ldots + R_n \rightarrow_{f(\mathbf{R},\mathbf{X};\mathbf{k})} P_1 + \ldots + P_m$$

and its associated sCCP agent $SCCP(\rho)$

$$SCCP(\rho) \text{ :- tell}_{f(\mathbf{R},\mathbf{X};\mathbf{k})} \left(\bigwedge_{i=i}^{n} (R_i - 1) \wedge \bigwedge_{j=i}^{m} (P_j + 1) \right) .SCCP(\rho).$$

Clearly, the reduced transition system of such agent is



Let $\mathbf{Y} = VAR(\rho)$, then the interaction matrix $I_{\mathbf{Y}}^{SCCP(\rho)}$ has $|\mathbf{Y}|$ rows and 1 column, with entries corresponding to the stoichiometry of the reaction:

$$I_{\mathbf{Y}}^{SCCP(\rho)}[Y_i] = \sum_{Y_i \in PROD(\rho)} 1 - \sum_{Y_i \in REACT(\rho)} 1.$$

The rate vector $r^{SCCP(\rho)}$, instead, is the scalar $f(\mathbf{R}, \mathbf{X}; \mathbf{k})$. Hence, the equation for variable Y_i is

$$\dot{Y}_i = \sum_{Y_i \in PROD(\rho)} f(\mathbf{R}, \mathbf{X}; \mathbf{k}) - \sum_{Y_i \in REACT(\rho)} f(\mathbf{R}, \mathbf{X}; \mathbf{k}),$$

which is equal to equation (10).

The inductive case follows easily from compositionality properties of ODE operators. Suppose the theorem holds for lists up to $k - 1$ reactions, and let $\mathcal{R} = \mathcal{R}_0 \cup \{\rho\}$ be a set of k chemical reactions (hence $|\mathcal{R}_0| = k - 1$). Then

$$\begin{aligned} ODE(SCCP(\mathcal{R}), \mathbf{X}) &= ODE(SCCP(\mathcal{R}_0 \cup \{\rho\}), \mathbf{X}) \\ &= ODE(SCCP(\mathcal{R}_0) \parallel SCCP(\rho), \mathbf{X}) \\ &= ODE(SCCP(\mathcal{R}_0), \mathbf{X}) + ODE(SCCP(\rho), \mathbf{X}) \\ &= ODE(\mathcal{R}_0, \mathbf{X}) + ODE(\rho, \mathbf{X}) \\ &= ODE(\mathcal{R}, \mathbf{X}), \end{aligned}$$

where the second equality follows from the definition of $SCCP$, the third follows from Theorem 2, the fourth is implied by the induction hypothesis on $SCCP(\mathcal{R}_0)$ and by the base case proof on $SCCP(\rho)$, while the last is a consequence of Lemma 4.

3.2 Preservation of dynamic behavior

In Theorem 3 we proved that the ODE map, when applied to sCCP-models of biochemical networks, satisfies a condition of coherence: it preserves the kinetic principles used in the construction of the model (i.e., the rate semantics).

A different question is whether an sCCP-network N (evolving stochastically according to the prescriptions of its semantic) shows a dynamic behavior equivalent to the one exhibited by the equations $ODE(N)$. This problem is the sCCP-counterpart of the famous mathematical issue concerning the relation between stochastic and differential models [35, 36], studied deeply also in the context of biochemical reactions [26, 23]. It is well-known that stochastic and differential models of biochemical reactions are behaviorally equivalent only in some cases.

These results are significant also for sCCP. Theorem 3, in fact, states that, when biochemical reactions are concerned, the stochastic process underlying the sCCP-models and the associated ODE's are exactly the classical ones. Therefore, in the mapping from sCCP to ODE's we have the same phenomenology as in the classical case. However, the logical structure of sCCP-agents makes the problem of behavioral preservation subtler.
In the following, we discuss this problem with different examples, especially of situations in which an sCCP-network and the corresponding ODE's show a different behavior. In particular, we are interested in sketching a brief, and plausibly incomplete, classification of the causes of behavioral divergence.

An important issue is the concept of behavioral equivalence itself, which is difficult to formalize, as already discussed in the introduction. We will return on this problem in the next section.

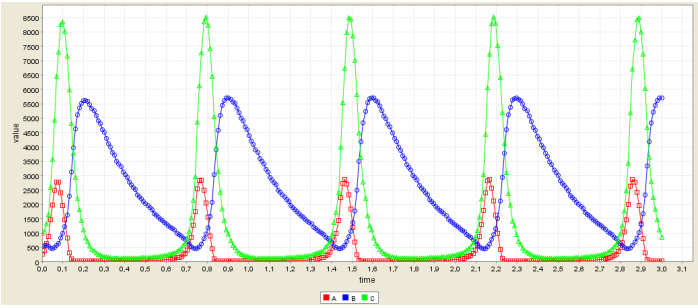
Oregonator The Oregonator is a chemical systems showing an oscillatory behavior, devised by Field and Noyes [40] as a simplified version of the Belousov-Zhabotinsky oscillator[18]. Essentially, Oregonator is composed of three chemical substances, call them A, B, C, subject to the following reactions:

$$\begin{aligned} B &\rightarrow_{k_1} A \\ A + B &\rightarrow_{k_2} \emptyset \\ A &\rightarrow_{k_3} 2A + C \\ 2A &\rightarrow_{k_4} \emptyset \\ C &\rightarrow_{k_5} B \end{aligned} \tag{12}$$

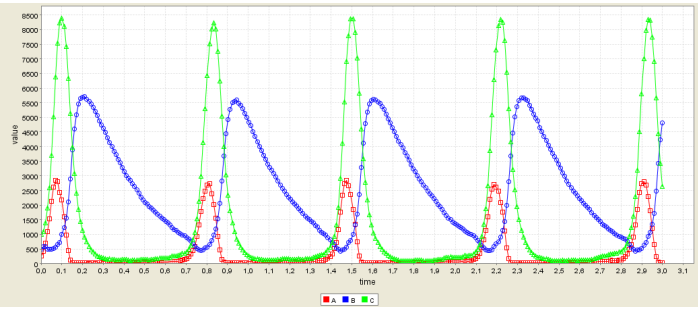
Actually, other chemical substances are involved, but they are kept constant in the experiment. The differential equations associated to (12) are known to possess a *stable limit cycle* for a wide range of parameter's values [28], containing an *unstable equilibrium*. The limit cycling behavior is clearly visible in Figure 3(a), where the numerical solution of Oregonator's ODE's is shown.

In Figure 3(b), instead, we plot a stochastic simulation of the sCCP model associated to (12) according to prescriptions of Section 2.2. In this case, the stochastic model shows the same pattern as the differential one. Theorem 3 guarantees that the graph in Figure 3 depicts the numerical solution of ODE's associated to the sCCP program by the transformation previously defined. In this case, the behavior is preserved.

[18] This chemical system is called "Oregonator" because its inventors where working at the University of Oregon.



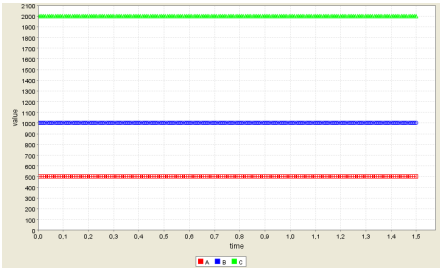
(a) ODE model of Oregonator



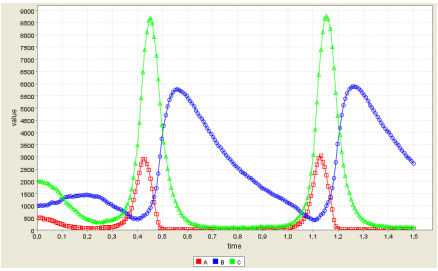
(b) sCCP model of Oregonator

Fig. 3. 3(a): numerical simulation of the differential equation model of the Oregonator, with parameters determined according to the method presented in [26]. Specifically, let $A_s = 500$, $B_s = 1000$ and $C_s = 2000$ be an equilibrium of the system of equations, and let $R_1 = 2000$, $R_2 = 50000$. Then parameters are equal to $k_1 = R_1/B_s = 2$, $k_2 = R_2/(A_s B_s) = 0.1$, $k_3 = (R_1 + R_2)/A_s = 104$, $k_4 = ((2R_1)/(A_s^2))/2 = 4e^{-7}$, and $k5 = (R_1 + R_2)/C_s = 26$. The starting point is $A_0 = A_s/2$, $B_0 = B_s/2$, $C_0 = C_s/2$. The system soon approaches an attractive limit cycle. **3(b)**: stochastic simulation with Gillespie's method of the sCCP network associated to reactions (12). Parameters and initial conditions are those specified above. The effect of stochastic fluctuations is negligible, and the plot essentially coincide with its deterministic counterpart.

We remark two things regarding Oregonator. First, the size of each molecular species is of the order of thousands, hence the relative variation induced by one reaction in the stochastic model is small. Under this condition, stochastic and deterministic models of biochemical reactions usually coincide [26]. Another property of the Oregonator that can be important for behavioral preservation is that the limit cycle is an *attractor* in the *phase space*: nearby trajectories asymptotically converge to it (see [51]). This means that a relatively small perturbation is not willing to change the overall dynamics: stochastic fluctuations have a negligible effect. Things are different if we start from the unstable equilibrium of the system. The numerical solution of ODE's shows a constant evolution (Figure 4(a)), while the stochastic simulation (Figure 4(b)) essentially evolves as the limit cycle of Figure 3. In fact, stochastic fluctuations, in this case, make the sCCP system move away from the instable equilibrium into the basin of attraction of the limit cycle. This shows another well known fact: *stochastic and differential models usually differ near instabilities* [26].



(a) ODE model of Oregonator from unstable equilibrium



(b) sCCP model of Oregonator from unstable equilibrium

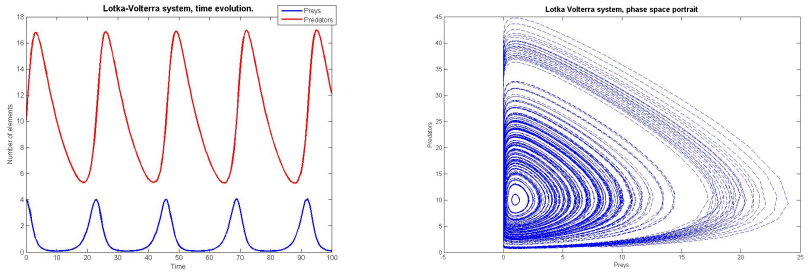
Fig. 4. 4(a): numerical simulations of ODE's derived from reactions (12), with parameters given in caption of Figure 3, starting from an unstable equilibrium of the system. **4(b)**: stochastic simulation of sCCP model associated to reactions (12), with the same parameters and initial conditions than the differential counterpart. As we can see, stochastic fluctuations drive the system away from the unstable equilibrium, so that its surrounding limit cycle is approached.

Lotka-Volterra system The Lotka-Volterra system is a famous simple model of population dynamics, see for example [26] and references therein. There are two species: preys and predators. Preys eat some natural resource, supposed unbounded, and reproduce at a rate depending only on their number. Predators, instead, can reproduce only if they eat preys, otherwise they die. To keep the model simple, we admit predation as the only source of prey's death. The previous hypotheses can be summarized in the following set of reactions, where E

refer to preys and C to predators:

$$\begin{aligned} E &\rightarrow_{k_b} 2E \\ C &\rightarrow_{k_d} \emptyset \\ E + C &\rightarrow_{k_p} 2C \end{aligned} \tag{13}$$

If we consider the standard mass action ODE's (they coincide with the equations derived from the sCCP model due to Theorem 3), a typical solution shows oscillations in which high values of preys and predators alternate. An example of such a solution is given in Figure 5(a). Inspecting equations, it can be shown that the point $E_s = k_d/k_p$, $C_s = k_b/k_p$ is an equilibrium of a rather special kind: it is stable (trajectories starting nearby it stay close) but not asymptotically stable (trajectories starting nearby do not converge to it as time approaches infinity). This behavior is easily understood looking at the phase space (Figure 5(b)), in which we can see that trajectories form closed orbits around the equilibrium, whose amplitude increases with distance from equilibrium. More details can be found, for instance, in [51].



(a) ODE solution of Lotka-Volterra model

(b) Phase space of Lotka-Volterra model

Fig. 5. 5(a): numerical solution of ODE's associated to reactions (13), with parameters $k_b = 1$, $k_p = 0.1$, $k_d = 0.1$ and initial conditions $E_0 = 4$ and $C_0 = 10$. **5(b)**: phase portrait of the Lotka Volterra system, for the same value of parameters as above. As we can see, all the solutions show an oscillating behaviour. The system has an (instable) equilibrium for $E = 1$, $C = 10$, at the center of the circles.

What kind of behavior can we expect from the stochastic evolution of the sCCP model for (13)? Stochastic fluctuations will make the system jump from one trajectory to nearby ones, without any force pulling it towards the equilibrium. Therefore, fluctuations can, in the long run, make the system wander in the phase plane, eventually reaching a borderline trajectory (corresponding to E or C axis in the phase plane). Whenever this happens, then both preys and predators go extinct (C-axis trajectory), or just predators do, while preys go to infinity (E-axis trajectory). This intuition is confirmed in Figure 6, where we

compare the ODE solution starting from equilibrium (dotted lines), and a trace of the sCCP model, starting from the same initial configuration. As we can see, the stochastic system starts oscillating until both species go extinct.

This is another well known case in which stochastic and differential dynamics differ, again induced by properties of the phase space [26].

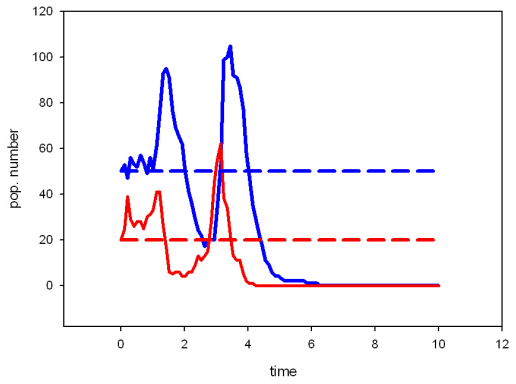


Fig. 6. Effect of stochastic fluctuations for the Lotka-Volterra system. The dotted lines are an equilibrium solution for the ODE model (parameters are as in caption of Figure 5). A stochastic trace of the sCCP model is drawn with solid lines: both species fluctuate around the equilibrium values until they both get extinct.

A negatively auto-regulated system The effect of stochastic fluctuations is mostly remarkable in biological phenomena where gene expression is involved. This is because the transcription of a gene is usually a slower process than protein-protein interaction, and often the number of mRNA strands for a given gene present in the cell is very small, of the order of some units. As the production of one single mRNA is a rare event (compared to other cellular events), stochastic variability in its happening can induce behaviors difficult to capture if mRNA is approximated with its concentration. Stochasticity in gene expression is indeed a phenomenon that has received a lot of attention, see for instance [38, 6].

We present here a simple, artificial example taken from [53] and depicted in Figure 7. The biological network shown represents a simple autoregulatory mechanism in gene expression of a procaryotic cell. Gene g produces, via mRNA r, a protein P that, as a dimer, can bind to a promoter region of gene g, preventing RNA-polymerase activity and thus inhibiting its own production.

Following the approach of [5], genes can be modeled as logical gates having a fixed output (the produced mRNA or protein), and several inputs, corresponding to different proteins of the system, exerting a positive or negative regulatory

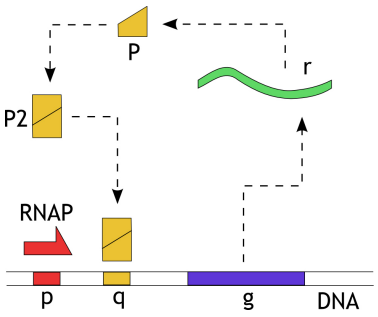


Fig. 7. Diagram of a simple self-regulated gene network. Gene g produces mRNA r and, from it, protein P. P can dimerise and its dimer can bind to a promoter region of gene g, downstream of RNA polymerase binding site. The P_2-binding blocks polymerase activity, thus inhibiting gene expression.

function. A gene gate with one inhibitory input is called in [5] *neg gate*, and can be modeled in RESTRICTED(sCCP) simply as:

$$\begin{aligned} \text{neg\_gate}_{P,I} :- & \\ & \text{tell}_{k_p}(P = P + 1).\text{neg\_gate}_{P,I} \\ + \quad & \text{ask}_{k_b \cdot I}(I \geq 1).\text{ask}_{k_u}(\mathit{true}).\text{neg\_gate}_{P,I}, \end{aligned}$$

where k_p is the basic production rate, k_b is the binding rate of the repressor to the promoter region of the gene and k_u is its unbinding rate.

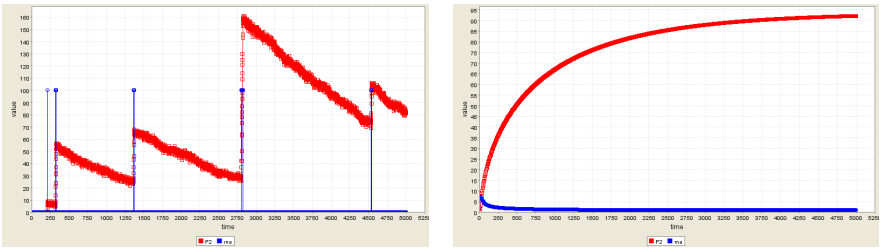
In order to model the system of Figure 7 we can combine one neg gate with some reactions. This is an example of the modeling style mixing the reaction-centric and the molecular-centric point of view, see Section 2.2. The model is the following:

$$\begin{aligned} & \text{neg\_gate}_{r,P_2} \\ & r \rightarrow_{k_t} r + P \\ & P \rightarrow_{k_{dim_1}} P_2 \\ & P_2 \rightarrow_{k_{dim_2}} P \\ & r \rightarrow_{k_{d_1}} \emptyset \\ & P \rightarrow_{k_{d_2}} \emptyset \end{aligned} \tag{14}$$

In Figure 8 we compare a stochastic simulation of the sCCP model of reactions (14) with the numerical solution of the associated ODE's. As we can readily see, the two plots are completely different. In particular, in the stochastic simulation, P_2 is produced in short bursts; normally it is slowly degraded. The bursts correspond to mRNA production events, shown in Figure 8(a) as blue peaks. The ODE's system, however, presents a much simpler pattern of evolution, in which the quantity of P_2 converges to an asymptotic value. This divergence is caused by the fact that, approximating continuously the number of RNA molecules, we lose the discrete information that seems to characterize

its dynamics, i.e. the fact that mRNA can be present in one unit of completely absent from the system.

Staten otherwise, continuously approximating molecular species present in low quantities may lead to errors inducing a completely divergent observable behavior.



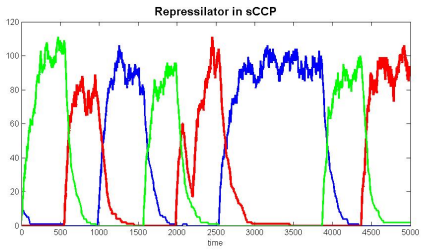
(a) sCCP model of system (14) (b) ODE model of system (14)

Fig. 8. 8(a): simulation of the sCCP model of (14). The red line corresponds to P_2, while the blue line shows the evolution of r, multiplied for a factor 100 (for visualization purposes). Note that the increases in P_2 expression immediately follow mRNA production events. Parameters of the models are the following: $k_p = 0.01$, $k_b = 1$, $k_u = 10$, $k_t = 10$, $k_{dim_1} = 1$, $k_{dim_2} = 1$, $k_{d_1} = 0.1$, and $k_{d_2} = 0.01$. All molecules are set initially equal to 0. **8(b)**: numerical simulation of ODE's associated to the sCCP model of (14), for the same parameters just given. The evolution of P_2 is tamer than in the stochastic counterpart, as it converges quickly to an asymptotic value.

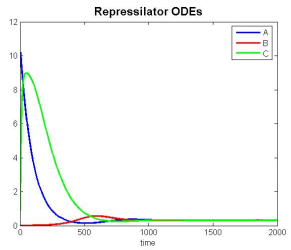
Repressilator The *Repressilator* [21] is an artificial biochemical clock composed of three genes expressing three different proteins, **tetR**, **λcI**, **LacI**, exerting a regulatory function on each other's gene expression. In particular, protein **tetR** represses the expression of protein **λcI**, protein **λcI** 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 three proteins. A simple stochastic model of Repressilator can be found in [5], where the authors describe it with three neg gates (see the previous paragraph) cyclically connected, in such a way that the product of one gate inhibits the successive gene gate in the cycle. In addition, they introduce degradation mechanisms for the three repressors. More formally, the model is the following

$$\begin{array}{l} \text{neg\_gate}_{A,C} \\ \text{neg\_gate}_{B,A} \\ \text{neg\_gate}_{C,B} \\ A \rightarrow_{k_d} \emptyset \\ B \rightarrow_{k_d} \emptyset \\ C \rightarrow_{k_d} \emptyset \end{array} \tag{15}$$

In Figure 9(a) we show a trace of the stochastic model generated by a simulator of sCCP based on Gillespie algorithm. The oscillatory behavior is manifest.



(a) sCCP model of repressilator



(b) sCCP model of repressilator

Fig. 9. 9(a): stochastic time trace for the Repressilator system of described by reactions 15. Parameters are $k_p = 1$, $k_d = 0.01$, $k_b = 1$, $k_u = 0.01$. **9(b)**: solution of the differential equations of Table 7, automatically derived from sCCP program associated to reactions 15. Parameters are the same as in stochastic simulation. The stochastic simulation lasts longer than the ODE one in order to better underline its oscillatory behavior.

If we apply the translation procedure discussed in Section 3 to this particular model, we obtain the ODE's shown in Table 7, while their numerical integration is shown in Figure 9(b). As we can readily see there is no oscillation at all, but rather the three proteins converge to an asymptotic value, after an initial adjustment.

Inspecting the ODE's, we note the presence of six variables (Y_A, Y_B, Y_C and Z_A, Z_B, Z_C) in addition to those representing the quantity of repressors in the system (A, B, C). Such variables correspond to states of genes gates, and they are used to model the change of configuration of the gates, from active to repressed and vice versa.

This scenario seems rather unjustified here: there is no argument to support the introduction of these variables, especially because we are continuously approximating boolean quantities.

An interesting point regarding Repressilator is the relation between the solution of the ODE's and the average trace of the stochastic system (i.e. $\mathbb{E}[\mathbf{X}(t)]$, returning the average value of system variables as a function of time). In fact, we may expect that the behavior preserved by the differential equations is the average dynamics of the stochastic system, rather than that shown by one of its traces. Interestingly, also the average value of the Repressilator model does not oscillate, as can be seen from Figure 10. This can be explained by noticing that the oscillations' period in the stochastic model is not constant, but it varies considerably. Hence, for every instant (when the Markov chain is at the stationary regime), we will observe one of the proteins at its peak value approximatively only in one third of the traces. Hence its average value will tend to stabilize at

$$\begin{array}{lll} \dot{A} = k_p Y_A - k_d A & \dot{Y}_1 = k_u Z_A - k_b Y_A C & \dot{Z}_1 = k_b Y_A C - k_u Z_A \\ \dot{B} = k_p Y_B - k_d B & \dot{Y}_2 = k_u Z_B - k_b Y_B A & \dot{Z}_2 = k_b Y_B A - k_u Z_B \\ \dot{C} = k_p Y_C - k_d C & \dot{Y}_3 = k_u Z_C - k_b Y_C B & \dot{Z}_3 = k_b Y_C B - k_u Z_C \end{array}$$

Table 7. ODE's derived for the Repressilator, generated by the method of Section 3.1.

one third of the peak value, as confirmed by Figure 10. In fact, when we average Repressilator, we measure the fraction of traces in which a certain gene is active and the fraction of traces in which it is inactive, for every time instant. In this way, however, we lose any information regarding the sequence of gene gate's state changing. The different behavior existing between a trace of a stochastic system and its average trace suggests that the switching dynamics of genes can be the driving force behind oscillations. This implies that another source of non-equivalence between sCCP models and the associated ODE's can appear due to the representation of RTS-states with continuous RTS-state variables.
Indeed, this example suggested us to preserve part of the discrete dynamics, mapping the sCCP Repressilator into an hybrid automaton. The work put forward in [**?**] shows that this move is enough to maintain oscillations. The translation to hybrid automata opens an entire range of possibilities to combine discreteness and continuity. These will be investigated in detail in the planned second part of this paper.

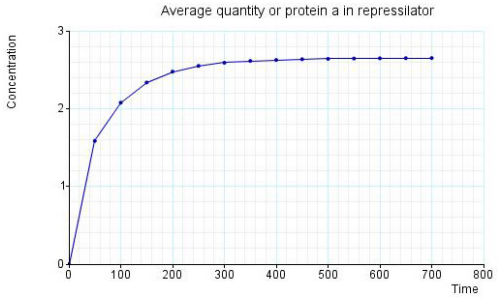


Fig. 10. Average value of the sCCP model for Repressilator, computed using model checker PRISM [37]. See [8] for further details.

Sources of non-equivalence In the previous examples we outlined different cases in which an sCCP model and its associated ODE's fail to be equivalent from a dynamical viewpoint. We remark that most of these examples are well known, as they have been studied in detail in theoretical and applicative contexts, like biochemical reactions [26, 53] and our main interest here is in their connection

with the sCCP translation machinery. For sake of clarity, we summarize the different sources of non-equivalence.

1. In some cases, non-equivalence is a direct consequence of *properties of the phase space*. For instance, instable trajectories are destroyed by small fluctuations, like the equilibrium trajectory of the Oregonator. Also stable but not asymptotically stable trajectories can be troublesome, as stochastic fluctuations are not counterbalanced by any attracting force, and so they can bring the stochastic system far away from the initial trajectory. This is the case of the Lotka-Volterra system.
2. Another well-known problem is related to the *approximation by continuous quantities of integer variables having small (absolute) values*. In this case, in fact, the effect of a single stochastic fluctuation has a relative magnitude that is relevant, so the dynamics can change quite dramatically. A typical example appearing in Biology is related to the transcription of genes, as shown in the simple example of a self-regulated gene.
3. A final source of non-equivalence is, instead, characteristic of the translation procedure defined for sCCP. In fact, in this case we *represent each RTS-state of a component of the system with a continuous variable*, which can take values in the real interval $[0, 1]$. RTS-states represent, in some sense, logical structures that control the activity of the system, while a change of state is an event triggered by some condition of the system. Moreover, in each sCCP trace, each component can be in only one state, hence RTS-state variables are boolean quantities. Continuous approximation, in this case, can have dramatic consequences, as the example of Repressilator seems to suggest.

3.3 Behavioral Equivalence

Comparing the dynamical evolution of a deterministic and a stochastic system is a delicate issue, because stochastic processes have a noisy evolution, hence we need to remove noise from their traces, before attempting any comparison with time traces evolving deterministically. In the previous discussion, in fact, we appealed to the concept of "behavioral equivalence" always in a vague sense, essentially leaving to the reader the task of visually comparing plots and recognizing similarities and differences. Clearly, a mathematical definition is needed in order to prove theorems and automatize comparisons.

We first consider the comparison of traces generated by ODE's with the average trace of the stochastic system, taken as the representative of its whole ensemble of traces. In practice, for each time instant t we need to compute the average value $\mathbb{E}(X(t))$ of each stream variable X w.r.t. the probability distribution on states of the system at time t. This probability can be obtained as the solution of the *Chapman-Kolmogorov forward equation* [39], a system of differential equations of the size of the state space. This equation, known in biochemical literature as the *chemical master equation* [25], can rarely be solved analytically, and it is also very difficult to integrate numerically [26]. A more efficient

approach to compute an estimate of the average consists in generating several (thousands of) stochastic traces and in computing pointwise their sample mean. Alternatively, the average value of one or more variables can be computed for a small sample of time points $\{t_1, \ldots, t_k\}$ using numerical techniques, as those implemented in the model checker PRISM [37].
Whatever the method chosen, the computation (even approximate) of the average trace of a stochastic system is a difficult matter. Whenever such trace is known, we can compare it with the trace of the ODE's, generated using standard numerical techniques [43], using quantitative measures (essentially computing a distance between the two curves). Indeed, in [9] it is shown that the ODE associated to a sCCP program is a *first order approximation* of the true equation for the average.
However, the average trace of a stochastic system is not necessarily a good representative of its evolution. A paradigmatic example is the Repressilator, whose average trace (sampled with PRISM, see caption of Figure 10) converges to an asymptotic value, while all its stochastic traces show persistent oscillations. Hence, *even when averaging a stochastic system, we may lose the characterizing qualitative features of its dynamics.*

The example of Repressilator suggests that the notion of behavioral equivalence is probably better captured in a qualitative setting. Qualitative comparison requires a formal definition of the *features* of dynamical evolution, like oscillations, convergence to a stable value, and so on. A possibility we suggest in this direction is to describe these features as logical formulae of a suitable logical language $\mathcal{L}$, for instance temporal logic, as done in Simpathica [3]. The concepts below are just sketched; this subject is currently under investigation and we will deal with it in detail in future works. Let Φ denote the set of formulae describing all dynamical features of interest. Associating a Kripke structure K_1 to the trace of an ODE and another structure K_2 to a stochastic trace, then we may declare these traces equivalent whenever their Kripke structures satisfy the same subset of formulae of Φ (possibly restricting the attention to formulae of degree $\leq n$).

Below we give three examples of temporal logic formulas expressing infinite oscillations:

$$G(Z = z_m \rightarrow F(Z = z_M)) \ \wedge \ G(Z = z_M \rightarrow F(Z = z_m)) \ \wedge \\ G(z_m \leqslant Z \leqslant z_M) \ \wedge \ F(Z = z_m);$$

$$G(Z = z_m \rightarrow X(Z > z_m \ U \ Z = z_M)) \ \wedge \ G(Z = z_M \rightarrow X(Z < z_M \ U \ Z = z_m)) \ \wedge \\ G(z_m \leqslant Z \leqslant z_M) \ \wedge \ F(Z = z_m);$$

$$\left(\neg G\left(\frac{dZ}{dt} > 0\right)\right) \ \wedge \ \left(\neg G\left(\tfrac{dZ}{dt} = 0\right)\right) \ \wedge \ \left(\neg G\left(\frac{dZ}{dt} < 0\right)\right).$$

In the above formulas X stands for *next*, G stands for *always (globally)*, F stands for *sometimes (in the future)*, U stands for *until*, z_m and z_M are minimum

and maximum values, and the thirds formula uses propositional formulas taking values according to the sign of the first derivative.[19]

This idea seems promising, as it gives a considerable freedom in the definition of formulae Φ, hence allowing to privilege some aspects of dynamical evolution more than others. However, the real problem is in the definition of a reasonable Kripke structure for a stochastic trace (and for sets of traces). In fact, Kripke structures for ODE's can be constructed starting from one or more traces, as done in Simpathica [3], in the following way: the bounded (product) domain of all variables is divided in small, compact regions; a state of the Kripke automaton consists of one of such regions; edges connect two states if a trajectory crosses the corresponding regions consecutively. This construction, however, is not reasonable for stochastic traces, as noise would force the addition of many edges that may introduce spurious behaviors. Of course, it is possible to model check directly on CTMC formulae written in CSL [4, 37]. However, the complexity of this latter approach makes the definition of non-deterministic Kripke structures interesting also for stochastic traces. We are currently investigating this direction, considering the introduction of a bounded form of memory to tame noise.

3.4 More on the restrictions of the language

RESTRICTED($sCCP$) restricts the full language in several aspects, see Section 2.3. Actually, these restrictions have been introduced in order to define in a reasonably simple way the mapping to ODE's. We discuss them in detail in the following.

First, all agents must be sequential, i.e. not containing any occurrence of the parallel operator. As already remarked at the end of Section 2.3, this does not constitute a real limitation, as each non-sequential agent can be transformed into a network of sequential ones. Here we note that the same trick of Section 2.3 can be used to transform each sCCP-network into an equivalent network where each sequential agent has an RTS with one single state; indeed, this is done implicitly by the transformation to ODE's itself. However, writing programs in this form is less natural.

Another syntactic restriction regards the definition of local variables. Actually, variables in ODE's have a global scope. Of course, any local variable can be made global by suitably renaming it. There is a problem, however, concerning the fact that at run-time we may generate an unbounded number of local variables. This implies that their use may lead to a set of ODE's with an infinite number of variables (although each equation will depend only on a finite number of them). The uprising of an infinite number of variables requires more complex mathematical techniques, and it prevents the use of standard numerical solvers.

Finally, the third class of restrictions regards the constraint store. The restriction to numeric variables is obviously necessary, as we are mapping to ODE's.

[19] In order to use meaningfully the notion of "next" for ODE's we need to consider a discretization of the time and of the state space, such as that performed in [3].

The restriction on the admissible constraints for the updating of variables, on the other hand, is related to the fact that each update in a sCCP program needs to be considered as a flux acting on some variables. Indeed, even a simple update like $X' = 0$ is difficult to render within ODE's framework, as it is inherently discrete. A possible way out is to mix the continuous ingredient of ODE's with discreteness, mapping sCCP programs to hybrid automata [29]. Within this formalism, updates like $X' = 0$ are perfectly admissible: they are *resets* associated to discrete transitions.

4 From ODE's to sCCP

In this section we define a transformation $SCCP$, associating an sCCP network to a generic set of ordinary differential equations, analyzing both its mathematical properties and the relation with the the map ODE defined in the previous section.

Before entering into the mathematical details, we need to make a preliminary remark. Essentially, the main obstacle we have to face in defining the map $SCCP$ is the fact that ODE's are an *aggregate* description of a system. To be more precise, if a system can be described by a set of fluxes acting on the different entities into play (i.e. on the system variables), then the ODE's hide part of the logical structure of such fluxes by combining them into the equations. To clarify the concept, consider the following two sCCP agents:

$$\text{A :- } (\text{tell}_1(X = X + 1) + \text{tell}_1(Y = Y + 1)).\text{A}$$

$$\text{B :- } (\text{tell}_1(X = X + 1 \wedge Y = Y + 1)).\text{B}$$

When we apply the ODE operator to the networks $N_1 = A$ and $N_2 = B$, we obtain, in both cases, the following equations:

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Therefore, two different sCCP agents can be mapped into the same set of ODE's. Note that A and B are "semantically" different, as they induce two different CTMC. The chain associated to A has edges connecting a state (i, j) to $(i+1, j)$ and $(i, j + 1)$ (hence the exit rate from (i, j) is 2), while the chain of B has transitions only from (i, j) to $(i + 1, j + 1)$ (with exit rate 1).
This information pertains the logical structure of the system, which is manifest in the sCCP program, but irremediably lost in the associated ODE's.

An even worse situation happens for the following agent, implementing a one-dimensional random walk [39]:

$$\text{C :- } (\text{tell}_1(X = X + 1) + \text{tell}_1(X = X - 1)).\text{C}$$

The equation associated to C by ODE is $\dot{X} = 0$, as the production and degradation rate cancel out when summed together. This equation predicts a constant

evolution for X, thus failing to capture its erratic behavior. Note, however, that the average value of X is constant also in the stochastic model for C.

Therefore, the structural information lost in passing from sCCP agents to ODE's makes impossible to recover the original sCCP network; stated otherwise, the map $ODE(\cdot)$ is not injective. Indeed, the lack of injectivity of $ODE(\cdot)$ means that an sCCP program is more informative than a set of ODE's: it defines not only the fluxes, but also their logical relation.

The previous discussion suggests that the map $SCCP$ must be defined with care. Given an ODE set $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, there are many sCCP networks that can be associated to it, i.e. at least all those belonging to the set $ODE^{-1}(\mathbf{f}(\mathbf{x}))$. In order to choose one among them, additional discriminating information is required, essentially related to the structure of the fluxes, hence to the logic of the system modeled.

As suggested by the previous discussion, we will therefore define not a single $SCCP$ map, but rather a class of maps, parametric w.r.t. the additional information required to sort out the logical structure of fluxes. We will then show that, independently of this additional information, the transformation scheme satisfies properties guaranteeing a form of coherence w.r.t. the ODE mapping and also a form of behavioral equivalence. Finally, we will provide two instantiations of such scheme, assuming specific conditions on the system modeled.

4.1 The translation to sCCP

In the conversion from ODE's to sCCP, we approximate continuous quantities by discrete variables. Therefore, this mapping will depend on an additional parameter, the *step* δ, specifying the *granularity* of the approximation of continuous variables. The magnitude of δ has a strong impact on the preservation of dynamical behavior; this point will be the content of Section 4.3.

Consider a system of first order ODE's with n variables $\mathbf{x} = (x_1, \ldots, x_n)$:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}).$$

We will now define the notion of *set of covering functions*, which captures the idea of external knowledge required to solve the ambiguity about the logical structure inherent in the ODE's. Essentially, a set of covering functions corresponds to a plausible choice of a set of fluxes, generating the given ODE's.

Definition 13. *A* set of covering functions $\mathcal{G}$ *for the ODE* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ *is a set of pairs* $\{(g_i, \mathbf{h_i}) \mid i = 1, \ldots, k\}$*, such that each* g_i *is a function* $g_i : \mathbb{R}^n \to \mathbb{R}$*, each* $\mathbf{h_i}$ *is a vector of* $\mathbb{Z}^n$*, and, for each* $\mathbf{x} \in \mathbb{R}^n$*,*

$$\sum_{i=1}^{k} \mathbf{h_i} g_i(\mathbf{x}) = f(\mathbf{x}).$$

Example 1. Consider the following simple system of ODE's with two variables:

$$\begin{cases} \dot{x} = a - by \\ \dot{y} = c + dx \end{cases} \tag{16}$$

One possible covering set is the following:

$$\mathcal{G} = \{(a,(1,0)),(by,(-1,0)),(c,(0,1)),(dx,(0,1))\},$$

which corresponds to the choice of disentangling all addends of the equations. Another possibility, instead, is the following: $\mathcal{G}' = \{(-a+by,(-1,1)),(a-by,(0,1,)),(c+dx,(0,1,))\}$, as easily verified.

In the sCCP program associated to the ODE's $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, we approximate the continuous variables $\mathbf{x}$ with discrete stream variables $\mathbf{X}$. Definition 1, however, requires variables $\mathbf{X}$ to have integer values. In order to set the size of the basic increment to an arbitrary step δ, we can change variables, setting $\mathbf{x} = \delta\mathbf{X}$ and expressing $\mathbf{f}$ with respect to $\mathbf{X}$ (in this way, a unit increment of X_i corresponds to an increment of δ of x_i). The equation for $\mathbf{X}$ thus becomes

$$\dot{\mathbf{X}} = \frac{1}{\delta}\mathbf{f}(\delta\mathbf{X}) = \mathbf{F}(\mathbf{X};\delta).$$

If we are given a set of covering functions $\mathcal{G}$ for $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, we can apply the same variable's substitution to each g_i, obtaining new covering functions $G_i(\mathbf{X};\delta) = \frac{1}{\delta}g_i(\delta\mathbf{X})$ such that

$$\dot{\mathbf{X}} = \mathbf{F}(\mathbf{X};\delta) = \frac{1}{\delta}\mathbf{f}(\delta\mathbf{X}) = \frac{1}{\delta}\sum_{i=1}^{k} g_i(\delta\mathbf{X}) = \sum_{i=1}^{k} G_i(\mathbf{X};\delta).$$

The translation to sCCP simply proceeds associating an agent to each element of the set of covering functions $\mathcal{G}$:

Definition 14. *Let $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ be a set of ODE's, and $\mathcal{G}$ be a set of covering functions for it. Let $g_i \in \mathcal{G}$ and $\delta \in \mathbb{R}^+$. The agent $\mathrm{man}_{G_i,\delta}$ is defined as*[20]

$$\begin{aligned}\mathrm{man}_{G_i,\delta} \;\coloneq\; &\mathrm{ask}_{|G_i(\mathbf{X};\delta)|}(G_i(\mathbf{X};\delta) > 0).\ \mathrm{tell}_{\infty}(\mathbf{X}' = \mathbf{X} + \mathbf{h_i}).\mathrm{man}_{G_i,\delta}\\ &+ \mathrm{ask}_{|G_i(\mathbf{X};\delta)|}(G_i(\mathbf{X};\delta) < 0).\ \mathrm{tell}_{\infty}(\mathbf{X}' = \mathbf{X} - \mathbf{h_i}).\mathrm{man}_{G_i,\delta}\end{aligned}$$

The agent $\mathrm{man}_{G_i,\delta}$ is a summation with two branches: both have rate equal to the modulus of function G_i, but one is active when $G_i > 0$, and it increments the value of $\mathbf{X}$ according to the vector $\mathbf{h_i}$, while the other is active when $G_i < 0$, decrementing $\mathbf{X}$ by $\mathbf{h_i}$.

In order to construct the sCCP network associated to a set of ODE $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, we simply need to define an agent $\mathrm{man}_{G_i,\delta}$ for each function G_i of the covering set $\mathcal{G}$, putting these agents in parallel. We can render this procedure in the following $SCCP$ operator:

[20] The name "man" stands for manager.

Definition 15. *Let* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ *be a set of ODE's,* $\mathcal{G}$ *be a set of covering functions for* $\mathbf{f}$*, and* $\delta \in \mathbb{R}$*,* $\delta > 0$*.* The sCCP-network associated to $\mathbf{f}(\mathbf{x})$, with respect to the set of covering functions $\mathcal{G}$ and the increment's step δ*, indicated by* $SCCP(\mathbf{f}(\mathbf{x}), \mathcal{G}, \delta)$*, is*

$$SCCP(\mathbf{f}(\mathbf{x}), \mathcal{G}, \delta) = \mathrm{man}_{G_1,\delta} \parallel \ldots \parallel \mathrm{man}_{G_k,\delta}, \tag{17}$$

with $\mathbf{x} = \delta\mathbf{X}$*.*

The initial conditions of the sCCP program, given by $init(\mathbf{X})$, are $\mathbf{X}(0) = \frac{1}{\delta}\mathbf{x_0}$, where $\mathbf{x_0}$ are the initial conditions of the ODE's.

Functional rates of sCCP are central in the definition of this translation: *each function of the covering set becomes a rate in a branch of an sCCP summation.* This is made possible only due to the freedom in the definition of rates, because differential equations and covering functions considered here are general.
The possibility of having general rates in sCCP is intimately connected with the presence of the constraint store, which contains information external to the agents. This means that part of the description of interactions can be moved from the logical structure of agents to the functional form of rates. Common stochastic process algebras like stochastic π-calculus [44] or PEPA [30], on the other hand, have simple numerical rates and they rely just on the structure of agents (and on additivity of the exponential distribution [39]) to compute the global rate. This restricts severely the class of functional rates that they can model. Indeed, in a recent work [18] Hillston introduces general rates in PEPA essentially through the addition of information external to the model, an approach similar in spirit to sCCP.

4.2 Invertibility

We turn now to study the relation between the two translations defined, i.e. ODE and $SCCP$. Specifically, we will show that $(ODE \circ SCCP)$ returns the original differential equations, independently from the covering set $\mathcal{G}$. The other direction, instead, cannot hold, as pointed out at the beginning of this section. In fact, we have seen that several sCCP agents can be mapped by ODE to the same equations, hence the map ODE cannot be inverted.

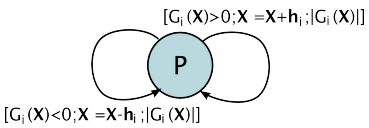
Theorem 4. *Let* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ *be a set of differential equations, with* $\mathbf{x} = (x_1, \ldots, x_n)$ *and* $\mathbf{X} = \frac{1}{\delta}\mathbf{x}$*, and let* $\mathcal{G}$ *be a set of covering functions for* $\mathbf{f}$*.*
Then

$$ODE(SCCP(\mathbf{f}(\mathbf{x}), \mathcal{G}, \delta), \mathbf{X}) = \mathbf{f}(\mathbf{x}).$$

Proof. From Definition 15 we know that $SCCP(\mathbf{f}(\mathbf{x}), \delta) = \mathrm{man}_{G_1,\delta} \parallel \ldots \parallel \mathrm{man}_{G_n,\delta}$, and by Theorem 2,

$$ODE(\mathrm{man}_{G_1,\delta} \parallel \ldots \parallel \mathrm{man}_{G_n,\delta}, \mathbf{X}) = ODE(\mathrm{man}_{G_1,\delta}, \mathbf{X}) + \ldots + ODE(\mathrm{man}_{G_n,\delta}, \mathbf{X}).$$

Now, the agent $\mathrm{man}_{G_i,\delta}$ can modify several variables X_i, according to the vector $\mathbf{h_i}$ coupled with the function G_i. The RTS of $\mathrm{man}_{G_i,\delta}$ is easily seen to have the following form



Therefore, the interaction matrix associated to $\mathrm{man}_{G_i,\delta}$ has just two columns, corresponding to the vectors $\mathbf{h_i}$ and $-\mathbf{h_i}$, and $ODE(\mathrm{man}_{G_i,\delta}, \mathbf{X})$ is equal to

$$\mathbf{h_i}|G_i(\mathbf{X};\delta)|\,\langle G_i(\mathbf{X};\delta) > 0\rangle - \mathbf{h_i}|G_i(\mathbf{X};\delta)|\,\langle G_i(\mathbf{X};\delta) < 0\rangle\,.$$

In the previous equation, $\langle\cdot\rangle$ denotes, as in Section 3, the logical value of a formula. The previous equation can be simplified by noting that

$$|G_i(\mathbf{X};\delta)|\,\langle G_i(\mathbf{X};\delta) > 0\rangle - |G_i(\mathbf{X};\delta)|\,\langle G_i(\mathbf{X};\delta) < 0\rangle = G_i(\mathbf{X};\delta),$$

hence

$$ODE(\mathrm{man}_{G_i,\delta}, \mathbf{X}) = \mathbf{h_i} G_i(\mathbf{X};\delta).$$

By applying Theorem 2 we then obtain

$$ODE(SCCP\,(\mathbf{f}(\mathbf{x}), \mathcal{G}, \delta)\,, \mathbf{X}) = \sum_{i=1}^{k} \mathbf{h_i} G_i(\mathbf{X};\delta) = F(\mathbf{X}),$$

which is equal to $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ when changing the variables back to $\mathbf{x}$.

4.3 Behavioral equivalence

We start this section by presenting an example showing how the translation from ODE's to sCCP works. In particular we will be concerned with the behavior exhibited by both systems and with the dependence on the step size δ, governing the size of the basic increment or decrement of variables. Intuitively, δ controls the "precision" of the sCCP agents w.r.t. the original ODE's. Hence, 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 another model of the Repressilator (see Section 3.2), a synthetic genetic network having an oscillatory behavior (see [21, 3]):

$$\begin{array}{llll} \dot{x}_1 = \alpha_1 x_3^{-1} - \beta_1 x_1^{0.5}, & \alpha_1 = 0.2, & \beta_1 = 0.01 \\ \dot{x}_2 = \alpha_2 x_1^{-1} - \beta_2 x_2^{0.5}, & \alpha_2 = 0.2, & \beta_2 = 0.01 \\ \dot{x}_3 = \alpha_3 x_2^{-1} - \beta_3 x_3^{0.5}, & \alpha_3 = 0.2, & \beta_3 = 0.01. \end{array} \tag{18}$$

We fix the following set $\mathcal{G}$ of covering functions: $g_1 = \alpha_1 x_3^{-1}$, $\mathbf{h_1} = (1,0,0)$, $g_2 = \alpha_2 x_1^{-1}$, $\mathbf{h_2} = (0,1,0)$, $g_3 = \alpha_3 x_2^{-1}$, $\mathbf{h_3} = (0,0,1)$, $g_4 = \beta_1 x_1^{0.5}$, $\mathbf{h_4} = (-1,0,0)$, $g_5 = \beta_2 x_2^{0.5}$, $\mathbf{h_5} = (0,-1,0)$, $g_6 = \beta_3 x_3^{0.5}$, $\mathbf{h_6} = (0,0,-1)$.

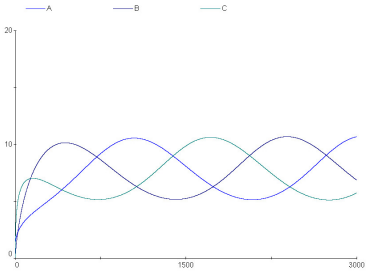
The corresponding sCCP process, after changing variables according to $X_i = \frac{x_i}{\delta}$, is:

$$\mathrm{man}_{G_1,\delta} \parallel \mathrm{man}_{G_2,\delta} \parallel \mathrm{man}_{G_3,\delta} \parallel \mathrm{man}_{G_4,\delta} \parallel \mathrm{man}_{G_5,\delta} \parallel \mathrm{man}_{G_6,\delta}, \tag{19}$$

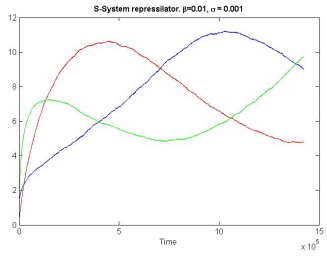
where, for instance, the agent $\text{man}_{G_1,\delta}$ is

$$\begin{aligned}\text{man}_{G_1,\delta} \;:\!-\; &\text{ask}_{\lfloor \frac{\alpha_1}{\delta}(\delta X_3)^{-1} \rfloor}(\tfrac{\alpha_1}{\delta}(\delta X_3)^{-1} > 0).\text{tell}_\infty(X_1' = X_1 + 1).\text{man}_{G_1,\delta} \\ + &\text{ask}_{\lfloor \frac{\alpha_1}{\delta}(\delta X_3)^{-1} \rfloor}(\tfrac{\alpha_1}{\delta}(\delta X_3)^{-1} < 0).\text{tell}_\infty(X_1' = X_1 - 1).\text{man}_{G_1,\delta}\end{aligned}$$

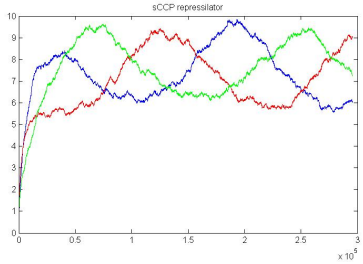
In Figure 11, we study the dependence on δ of the sCCP network obtained from equations (19). From the plots, we note that the smaller the δ, the closer the stochastic trace is to the solution of ODE's. However, increasing δ, the effect of stochastic perturbations gets stronger and stronger, making the system change dynamics radically.



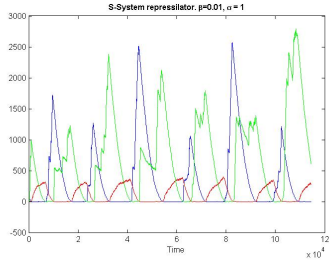
(a) Solution of ODE's (19)



(b) SCCP simulation, $\delta = 0.001$



(c) SCCP simulation, $\delta = 0.01$



(d) SCCP simulation, $\delta = 1$

Fig. 11. Different simulations of sCCP agent obtained from S-Systems equations of repressilator (19), as basic step δ varies. Specifically, in Figure 11(a) we show the solution of ODE's (19), while in Figures 11(b), 11(c), 11(d) we present three simulations of the sCCP agent corresponding to ODE's (19), for $\delta = 0.001, 0.01, 1$ respectively. In the last diagram, the behavior of S-System's equations is destroyed. Note that in Figure 11(c) the time axis is stretched by a factor of 100, while in Figure 11(b) the time axis is stretched by a factor of 1000, consistently with the rescaling of variables by $\frac{1}{\delta}$ performed in the translation from ODE to sCCP.

Reducing the value of δ seems to be essentially the same as working with a sufficiently high number of molecules in standard biochemical networks, see [26, 24] and the discussion in Section 3.2. It is thus reasonable to expect that, by

taking δ smaller and smaller, the deterministic and the stochastic dynamics will eventually coincide. In fact, reducing δ we are diminishing the magnitude of stochastic fluctuations, hence their perturbation effects.
This conjecture is indeed true: in the rest of the section we prove that, under mild conditions on the ODE's and of the functions of $\mathcal{G}$, the trajectories of the stochastic simulation converge to the solution of the ODE's, independently of the choice of the covering set $\mathcal{G}$. In fact, the set of stochastic traces whose distance from the solution of the ODE's is greater than a fixed arbitrary constant has zero probability in the limit $\delta \to 0$.

Kurtz theorem In 1970 Thomas Kurtz proved a theorem giving conditions for a family of density dependent Continuous Time Markov Chains to converge to a solution of a system of ODE's [35, 36]. In fact, under mild assumptions on the smoothness of functions into play, the trajectories of the CTMC remain, in the limit, close to the solution of a particular set of ODE's with probability one. Our mapping *SCCP* easily fits into Kurtz's framework, with the step δ playing the role of the density. We start by recalling the Kurtz's theorem.

Let V be a positive parameter, playing the role of the "size" of the system, and $\mathbf{X}_V(t)$ be a family of CTMC with state space $\mathbb{Z}^n$, depending on the parameter V. Suppose that there exist a continuous positive real function $\varphi : \mathbb{R}^n \times \mathbb{Z}^n \to \mathbb{R}$, such that the infinitesimal generator matrix [39] $Q = (q_{\mathbf{X},\mathbf{Y}})$ for $\mathbf{X}_V(t)$ is given by

$$q_{\mathbf{X},\mathbf{X}+\mathbf{h}} = V\varphi(\frac{1}{V}\mathbf{X}, \mathbf{h}), \quad \mathbf{h} \neq \mathbf{0}.$$

In addition, let $\Phi(\mathbf{x}) = \sum_{\mathbf{h}\in\mathbb{Z}^n} \mathbf{h}\varphi(\mathbf{x}, \mathbf{h})$.

Theorem 5 (Kurtz [35]). *Fix a bounded time interval $[0, T]$. Suppose there exists an open set $E \subseteq \mathbb{R}^n$ and a constant $M_E \in \mathbb{R}^+$ such that*

1. *$|\Phi(\mathbf{x}) - \Phi(\mathbf{y})| < M_E|\mathbf{x} - \mathbf{y}|$, $\forall \mathbf{x}, \mathbf{y} \in E$ (i.e. Φ satisfies the Lipschitz condition);*
2. *$\sup_{\mathbf{x}\in E} \sum_{\mathbf{h}\in\mathbb{Z}^n} |\mathbf{h}|\varphi(\mathbf{x}, \mathbf{h}) < \infty$;*
3. *$\lim_{d\to\infty} \sup_{\mathbf{x}\in E} \sum_{|\mathbf{h}|>d} |\mathbf{h}|\varphi(\mathbf{x}, \mathbf{h}) = 0$.*

Then, for every trajectory $\mathbf{x}(t)$ that is a solution of $\dot{\mathbf{x}} = \Phi(\mathbf{x})$ satisfying $\mathbf{x}(0) = \mathbf{x_0}$ and $\mathbf{x}(t) \in E$, $t \in [0, T]$, if

$$\lim_{V\to\infty} \frac{1}{V}\mathbf{X}_V(0) = \mathbf{x_0},$$

then for every $\varepsilon > 0$,

$$\lim_{V\to\infty} \mathbb{P}\left\{\sup_{t\leq T} \left|\frac{1}{V}\mathbf{X}_V(t) - \mathbf{x}(t)\right| > \varepsilon\right\} = 0.$$

This theorem states that the trajectories of $\mathbf{X}_V(t)$ converge, in a bounded time interval, to the solution of $\dot{\mathbf{x}} = \Phi(\mathbf{x})$, when $V \to \infty$. The function Φ is essentially the sum of all fluxes of the system.

Convergence for *SCCP*

Our framework can be easily adapted to fit this theorem. Consider a system of ODE's $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ and a set of covering functions $\mathcal{G}$. Denote by $\mathbf{X}_\delta(t)$ the CTMC associated to the sCCP-network $SCCP(\mathbf{f}(\mathbf{x}), \mathcal{G}, \delta)$.

Theorem 6. *Let $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ be a system of ODE's, with $\mathbf{x} \in \mathbb{R}^n$, and $[0, T]$ a bounded time interval. Let $\mathcal{G} = \{(g_i, \mathbf{h_i}) \mid i = 1 \ldots, \mu\}$ be a set of covering functions for $\mathbf{f}$.*
If there exists an open set $E \subseteq \mathbb{R}^n$ such that $\mathbf{f}$ satisfies the Lipschitz condition in E and $\sup_{\mathbf{x}\in E} |g_i(\mathbf{x})| < \infty$, for each $i = 1 \ldots, \mu$, then for every $\varepsilon > 0$

$$\lim_{\delta \to 0} \mathbb{P}\left\{ \sup_{t \leq T} |\delta \mathbf{X}_\delta(t) - \mathbf{x}(t)| > \varepsilon \right\} = 0,$$

where $\mathbf{x}(t)$ is the solution of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ with initial condition $\mathbf{x}(0) = \mathbf{x_0}$ and $\delta \mathbf{X}_\delta(0) = \mathbf{x_0}$.

Proof. In order to prove the theorem, we simply need to show that we satisfy all the hypothesis of the Kurtz's theorem. First of all, in this setting the density V is equal to $\frac{1}{\delta}$, so that $\frac{1}{\delta} \to \infty$ when $\delta \to 0$.
Consider now the function $g_i(\mathbf{x})$, and define as customary $g_i^+(\mathbf{x}) = g_i(\mathbf{x})\langle g_i(\mathbf{x}) \geq 0\rangle$ and $g_i^-(\mathbf{x}) = g_i(\mathbf{x})\langle g_i(\mathbf{x}) \leq 0\rangle$, so that $g_i(\mathbf{x}) = g_i^+(\mathbf{x}) - g_i^-(\mathbf{x})$ and $|g_i(\mathbf{x})| = g_i^+(\mathbf{x}) + g_i^-(\mathbf{x})$, where $\langle\cdot\rangle$ denotes the logical value as before.
Consider now the infinitesimal generator matrix $Q^\delta = (q^\delta_{\mathbf{X},\mathbf{Y}})$ of the CTMC $\mathbf{X}_\delta(t)$. It is straightforward to prove that, for each $\mathbf{h} \in \mathbb{Z}^n$,

$$q^\delta_{\mathbf{X},\mathbf{X}+\mathbf{h}} = \frac{1}{\delta} \sum_{i \mid \mathbf{h_i} = \mathbf{h}} g_i^+(\delta \mathbf{X}) + \frac{1}{\delta} \sum_{i \mid \mathbf{h_i} = -\mathbf{h}} g_i^-(\delta \mathbf{X}),$$

where the sum must be intended equal to zero if the index set is empty. Clearly, these are density dependent rates, with density $\frac{1}{\delta}$. Note that there is a finite number of vectors for which $q^\delta_{\mathbf{X},\mathbf{X}+\mathbf{h}}$ is different from zero, as the set $\{\mathbf{h_1}, \ldots, \mathbf{h}_\mu\}$ is finite.
Therefore, the function φ of the Kurtz theorem is simply defined as

$$\varphi(\mathbf{x}, \mathbf{h}) = \sum_{i \mid \mathbf{h_i} = \mathbf{h}} g_i^+(\mathbf{x}) + \sum_{i \mid \mathbf{h_i} = -\mathbf{h}} g_i^-(\mathbf{x}).$$

Then, the function $\Phi(\mathbf{x})$ is

$$\Phi(\mathbf{x}) = \sum_{\mathbf{h}} \mathbf{h}\varphi(\mathbf{x}, \mathbf{h}) = \sum_{j=1}^{\mu} \mathbf{h_j}(g_j^+(\mathbf{x}) - g_j^-(\mathbf{x})) = \sum_{j=1}^{\mu} \mathbf{h_j} g_j(\mathbf{x}) = \mathbf{f}(\mathbf{x}).$$

It only remains to prove that conditions 1–3 of Theorem 5 are satisfied. Condition 1 is obvious because $\Phi = \mathbf{f}$ is Lipschitz by hypothesis, while condition 3

hold because $|\mathbf{h}| > M$ implies $\varphi(\mathbf{x}, \mathbf{h}) = 0$, where $M = \max_{1 \leq i \leq \mu} |\mathbf{h_i}|$. Finally, condition 2 follows because

$$\sum_{\mathbf{h}} |\mathbf{h}| \varphi(\mathbf{x}, \mathbf{h}) \leq M \sum_{j=1}^{\mu} (g_j^+(\mathbf{x}) + g_j^-(\mathbf{x})) = M \sum_{j=1}^{\mu} |g_j(\mathbf{x})|,$$

hence

$$\sup_{\mathbf{x} \in E} \sum_{\mathbf{h}} |\mathbf{h}| \varphi(\mathbf{x}, \mathbf{h}) \leq \sup_{\mathbf{x} \in E} M \sum_{j=1}^{\mu} |g_j(\mathbf{x})| \leq \sum_{j=1}^{\mu} \sup_{\mathbf{x} \in E} M |g_j(\mathbf{x})| \leq \infty,$$

due to the condition on g_i functions.

Comments and examples on Theorem 6 Theorem 6 states that sCCP networks are able to simulate ODE's with an arbitrary precision. The cost of an exact stochastic simulation of the sCCP-network of Definition 15, however, is proportional to $\frac{1}{\delta}$, hence accurate stochastic simulations of ODEs are computationally impractical. On the other hand, there is no apparent reason to generate stochastic trajectories indistinguishable from the solution of the ODE's, as the latter can be generally obtained with much less computational effort.
In a work related to ours [22], Hillston et al. used the same Kurtz theorem to prove an analogous result for the equations that can be obtained from a PEPA program. Theorem 6 can be seen as a generalization of their result. Moreover, in [22] the authors suggest that a stochastic approximation of ODE's can be used together with analysis techniques typical of CTMC, like steady state analysis. This is a promising direction, but extreme care must be used.

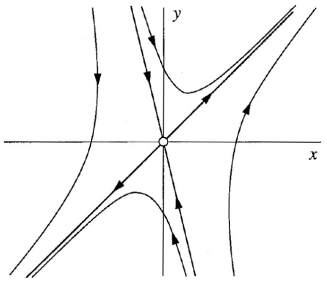
Kurtz theorem, in fact, guarantees convergence only in a fixed and bounded time interval $[0, T]$, hence it does say nothing about asymptotic convergence of stochastic trajectories to ODE's.
Intuitively, the step δ may not be the only responsible for asymptotic convergence; an important role should also be played by initial conditions through topological properties of the phase space. If the ODE-trajectory we are considering is stable, i.e. resistant to small perturbations, then we can expect it to be reproduced in sCCP along the whole time axis, given a step δ small enough. On the other hand, if the trajectory is unstable, then even small perturbations can drive the dynamics far away from it; stochasticity, in this case, will unavoidably produce a trace dramatically different from the one of ODE's. Of course, by taking the interval $[0, T]$ of the theorem big enough (hence δ small enough), we can postpone arbitrarily far away in time the moment in which a stochastic and an unstable deterministic trajectories will diverge.

As an example of instability, let's consider a simple linear system of differential equations:

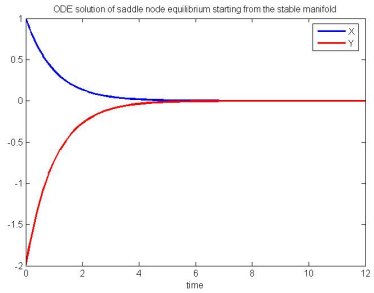
$$\begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} = \begin{pmatrix} X + Y \\ 4X + Y \end{pmatrix} \tag{20}$$

The theory of dynamical systems [51] tells us that the point $(0, 0)$ is a *saddle*

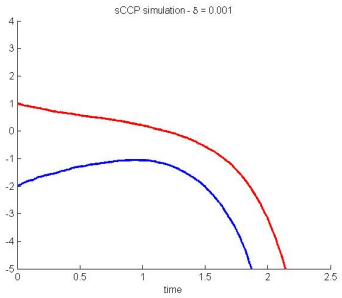
node, i.e. an unstable equilibrium whose phase space resembles the one depicted in Figure 12(a). The two straight lines are the directions spanned by the eigenvectors of the matrix of coefficients $\begin{pmatrix} 1 & 1 \\ 4 & 1 \end{pmatrix}$, and are called *stable* and *unstable manifolds*. Motion in the stable manifold converges to the equilibrium $(0, 0)$, while the unstable manifold and all other trajectories diverge to infinity. However, small perturbations applied to the stable manifold can bring the system on a divergent hyperbolic trajectory, so we expect that ODE's and the associated sCCP agent, when starting from the stable manifold (say from point $(1, -2)$), will eventually jump on a divergent trajectory. Moreover, we expect that the smaller δ the later this event will happen. This intuition is confirmed in Figures 12(b), 12(c), 12(d).



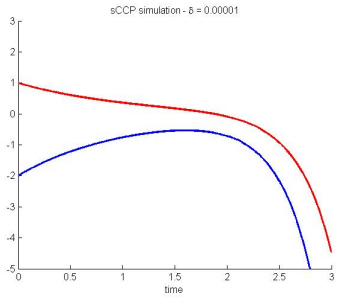
(a) Phase portrait of a saddle node



(b) ODE solution



(c) sCCP simulation — $\delta = 0.001$



(d) sCCP simulation — $\delta = 0.00001$

Fig. 12. 12(a): phase space of the linear system (20). The origin is a saddle node; the stable manifold is displayed with arrows pointing towards the origin, while the unstable manifold has arrows diverging from it. **12(b)**: solution of the ODE's (20), starting from $(1, -2)$, a point belonging to the stable manifold. **12(c),12(d)**: simulation of the sCCP agent associated to the linear system (20), w.r.t. the set of convering functions $\{(X, (1, 0)), (Y, (1, 0)), (4X, (0, 1)), (Y, (0, 1))\}$ with initial conditions $(1, -2)$. The step δ is equal respectively to 0.001 and 0.00001. The time in which these trajectories diverge from the solution of the ODE's increases as δ becomes smaller.

This example shows that convergence issues need to be investigate further. In particular, conditions taking into account the topology of the phase space of the ODE's are required in order to guarantee also asymptotic convergence. Another interesting direction to investigate is to exploit other results by Kurtz [36] in order to state error bounds in the approximation.

Examples of Sets of Covering Functions All the results of this Section have been given parametrically w.r.t. a set of covering functions $\mathcal{G}$. When we motivated the introduction of such concept, we stated that a choice of a specific $\mathcal{G}$ corresponds to a specific logical structure of the fluxes generating the ODEs. We discuss now two possible choices of $\mathcal{G}$, one natural in absence of information, and the other tailored for ODEs coming from sets of biochemical reactions for which it is possible to reconstruct the reactions from the ODEs.

Example 2. The simplest choice of a covering set is the one in which all the addends of ODEs are treated as independent flux sources. To be more specific, consider a set of ODE $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$, with $f_i(\mathbf{x}) = \sum_{j=1}^{k_i} f_{ij}(\mathbf{x})$, where f_{ij} are the single addends of the ODE. The idea is to treat independently each such f_{ij}, so that $\mathcal{G}_D = \{(f_{ij}, \mathbf{e_i}) \mid 1 \leq i \leq k, 1 \leq j \leq k_i\}$. Such covering set $\mathcal{G}_D$ will be called in the following the *disentangled covering set*. This was the choice adopted, for instance, when discussing the Repressilator example in Section 4.3. This choice is reasonable in absence of any further information on the system modeled by the ODE's, and it does not preserve structural properties of the system, like mass conservation. For instance, consider the system defined by the single reaction $A \rightarrow_{k \cdot A} B$, which preserves the total mass $A+B$. With the disentangled covering set, however, we would reconstruct the logical structure of the following system of biochemical reactions: $A \rightarrow_{k \cdot A}$ and $\rightarrow_{k \cdot A} B$, which does not preserve the total mass $A + B$.

Example 3. Assume now we have a system generated by a set of mass action reactions such that the left hand side (i.e., the list of reactants) of each such reaction is unique. This has the consequence that each reaction is uniquely identified by the algebraic structure as a monomial of its rate function. For instance, the only reaction with A and B as reagents, $A+B \rightarrow_k ?$, is uniquely identified by the signature as a monomial of its rate function kAB, i.e. by the monomial AB with coefficient 1. This property has the immediate consequence that, whenever we find two addends of the ODE with the same signature as a monomial, we are guaranteed that they are two instances of the same flux. That is to say, if the condition is satisfied, we know how to reconstruct the set of reactions that originated the ODE's.

Formally, given $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$, with $f_i(\mathbf{x}) = \sum_{j=1}^{k_i} f_{ij}(\mathbf{x})$, we can construct the covering set $\mathcal{G}_R$ as follows: list all the different support monomials $\{p_1, \ldots, p_s\}$ in the set $\{f_{ij} \mid 1 \leq i \leq k, 1 \leq j \leq k_i\}$, and, for each l, define the terms $\alpha_{l,j} \in \mathbb{Z}$ and $\beta_l \in \mathbb{R}^+$ such that:

1. if p_l occurs in the equation for variable i, then its occurrence is equal to $\alpha_{l,i}\beta_l p_l$;

2. if p_l does not occur in the equation for variable i, then $\alpha_{l,i} = 0$;
3. all $\alpha_{l,j} \neq 0$ are prime among them[21];

Then, letting $\alpha_l = (\alpha_{l,1}, \ldots, \alpha_{l,k})$, the covering set $\mathcal{G}_R$ can be defined as $\mathcal{G}_R = \{(\beta_l p_l, \alpha_l) \mid l = 1, \ldots, s\}$.

For instance, consider the ODEs

$$\begin{cases} \dot{X} = -2kX^2Y \\ \dot{Y} = -kX^2Y \\ \dot{Z} = 3kX^2Y \end{cases}$$

The associated covering set $\mathcal{G}_R$ is simply $\mathcal{G}_R = \{(kX^2Y, (-2, -1, 3))\}$, corresponding to the single reaction $2X + Y \rightarrow_k 3Z$.

5 Final discussion

In this paper we presented a method to associate ordinary differential equations to sCCP programs (written with a restricted syntax), and also a method that generates an sCCP-network from a set of ODE's. The translation from sCCP to ODE's is based on the construction of a graph, called RTS, whose edges represent all possible actions performable by sCCP-agents. Properties of RESTRICTED($sCCP$) guarantee that the graph is always finite. From an RTS, we can construct an interaction matrix containing the modifications that each action makes to each variable. Writing the corresponding ODE's is simply a matter of combining the interaction matrix with the rate of each action. The inverse translation, from ODE's to sCCP, exploits the functional form that rates have in sCCP. In this way, we can associate sCCP-agents to general ODE's. An important feature of this method is that it is parametric w.r.t. the basic increment of variables, meaning that we can reduce the effect of stochastic fluctuations in the sCCP-model. Actually, we proved in Theorem 6 that, in the limit of an infinitesimal increment, the trajectories of the ODE's and of the corresponding sCCP-system coincide.

In Section 3.1, we showed that the translation from sCCP to ODE's, when applied to models of biochemical reactions, preserves the rate semantics in the sense of [17]. This condition, however, is not sufficient to guarantee that the translation maintains also the dynamical behavior of the sCCP-model. In fact, in Section 3.2, we provided several examples where an sCCP-network and the associated ODE's manifest a different behavior. This divergence can be caused by many factors, all qualitatively different.

Preserving dynamical behavior, however, is not just a mathematical game, but is is a central property that a translation from sCCP to ODE should have in order to be used as an analysis technique for stochastic process algebras. In this light, also the mapping from ODE to sCCP can be seen as a tool to investigate behavioral preservation.

[21] This condition guarantees that α_{lj} are uniquely defined

In Section 4, when we introduced the notion of set of covering functions, we noted that in the passage to ODE's we unavoidably lose something of the logic of the sCCP model. This also suggest that the preservation of behavior may be reasonable only from a qualitative point of view. Indeed, this weaker approach fits better with the management of stochastic noise, see the discussion at the end of Section 3.2. The loss of precision in passing to ODE's is, however, counterbalances by the computational gain: simulating stochastic processes is undoubtedly much more expensive than numerically solving ODE's [24].

There are several open problems related to the question of behavioral equivalence. We list hereafter some of the most important ones, according to us.

- We need to identify the class of sCCP models (and their regions of parameter space/initial conditions) for which the mapping ODE preserves dynamics. Intuitively, according to discussion of Section 3.2, this may happen if all variables have big absolute values and if the phase space of the ODE's has asymptotically stable trajectories with ample basins of attraction.
- The repressilator and the simple self-inhibited genetic network of Section 3.2 suggest that the discrete ingredient cannot be continuously approximated so easily. In particular, associating continuous variables to RTS-states seems rather arbitrary. A possible solution can be that of transforming an sCCP network into a hybrid system, in which continuous and discrete dynamics coexist. In this way, we may be able to preserve part of the discrete structure of an sCCP-network, possibly just that fundamental for maintaining the behavior. We are investigating this direction, mapping sCCP-programs to hybrid automata [29, 2]. The first results are encouraging, see [13, 14]
- The notion of behavioral equivalence needs to be specified formally. At the end of Section 3.2, we suggested an approach based on a suitable temporal logic, in which equivalence would mean equi-satisfiability of the same set of formulae.

As a final remark, we would like to consider this work under the perspective of the study of systemic properties. In fact, when we model a biological system, we are concerned mainly with the understanding of its systemic properties, especially what they are and how they emerge from basic interactions. In this direction, a modeler needs a formal language to specify biological systems, possibly provided with different semantics, related to one another and stratified in several layers of increasing approximation and abstraction. For example, sCCP has a natural CTMC-based semantics, but an ODE-based one can be assigned to it via the ODE operator. A possible layer in the middle consists in a semantic based, for instance, on hybrid automata. Finally, we need also a language to specify system's properties, automatically verifying them on the different semantics, or better, on the simpler semantic where answers are correct (i.e., on the simpler semantic showing the same dynamical behavior of the most general one). All these features must clearly be part of the same operative framework (and of the same software tool), hence all the open questions presented above can be seen as steps in this direction.

References

1. Converging sciences. Trento, 2004. http://www.unitn.it/events/consci/.
2. R. Alur, C. Belta, F. Ivancic, V. Kumar, M. Mintz, G. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biomolecular networks. In *Proceedings of Fourth International Workshop on Hybrid Systems: Computation and Control*, volume LNCS 2034, pages 19–32, 2001.
3. M. Antoniotti, A. Policriti, N. Ugel, and B. Mishra. Model building and model checking for biochemical processes. *Cell Biochemistry and Biophysics*, 38(3):271–286, 2003.
4. A. Aziz, V. Singhal, F. Balarin, R. Brayton, and A. Sangiovanni-Vincentelli. Verifying continuous time markov chains. In *Proceedings of CAV96*, 1996.
5. R. Blossey, L. Cardelli, and A. Phillips. A compositional approach to the stochastic dynamics of gene networks. *T. Comp. Sys. Biology*, pages 99–122, 2006.
6. R. Blossey, L. Cardelli, and A. Phillips. Compositionality, stochasticity and cooperativity in dynamic models of gene regulation. *HFPS Journal, in print*, 2007.
7. L. Bortolussi. Stochastic concurrent constraint programming. In *Proceedings of 4th International Workshop on Quantitative Aspects of Programming Languages (QAPL 2006)*, volume 164 of *ENTCS*, pages 65–80, 2006.
8. L. Bortolussi. *Constraint-based approaches to stochastic dynamics of biological systems.* PhD thesis, PhD in Computer Science, University of Udine, 2007. Available at http://www.dmi.units.it/ bortolu/files/reps/Bortolussi-PhDThesis.pdf.
9. L. Bortolussi. A master equation approach to differential approximations of stochastic concurrent constraint programming. In *Proceedings of QAPL 2008, to appear in ENTCS*, 2008.
10. L. Bortolussi, S. Fonda, and A. Policriti. Constraint-based simulation of biological systems described by molecular interaction maps. In *Proceedings of IEEE conference on Bioinformatics and Biomedicine, BIBM 2007*, 2007.
11. L. Bortolussi and A. Policriti. Relating stochastic process algebras and differential equations for biological modeling. *Proceedings of PASTA 2006*, 2006.
12. L. Bortolussi and A. Policriti. Stochastic concurrent constraint programming and differential equations. In *Proceedings of Fifth Workshop on Quantitative Aspects of Programming Languages, QAPL 2007*, volume 16713 of ENTCS, 2007.
13. L. Bortolussi and A. Policriti. Hybrid approximation of stochastic concurrent constraint programming. In *Proceedings of IFAC 2008*, 2008.
14. L. Bortolussi and A. Policriti. The importance of being (a little bit) discrete. In *Proceedings of FBTC'08, to appear in ENTCS*, 2008.
15. L. Bortolussi and A. Policriti. Modeling biological systems in concurrent constraint programming. *Constraints*, 13(1), 2008.
16. L. Cardelli. From processes to odes by chemistry. *downloadable from http://lucacardelli.name/*, 2006.
17. L. Cardelli. On process rate semantics. *Theoretical Computer Science*, 391(3) 190-215, 2008.
18. F. Ciocchetta and J. Hillston. Bio-PEPA: an extension of the process algebra PEPA for biochemical networks. In *Proceeding of FBTC 2007*, 2007. Workshop of CONCUR 2007.
19. Seattle CompBio Group, Institute for Systems Biology. Dizzy home page.
20. A. Cornish-Bowden. *Fundamentals of Chemical Kinetics.* Portland Press, 3rd edition, 2004.

21. M.B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.
22. N. Geisweiller, J. Hillston, and M. Stenico. Relating continuous and discrete pepa models of signalling pathways. *Theoretical Computer Science*, 2008. in print.
23. D. Gillespie. The chemical langevin equation. *Journal of Chemical Physics*, 113(1):297–306, 2000.
24. D. Gillespie and L. Petzold. *System Modelling in Cellular Biology*, chapter Numerical Simulation for Biochemical Kinetics. MIT Press, 2006.
25. D.T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. of Computational Physics*, 22, 1976.
26. D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. of Physical Chemistry*, 81(25), 1977.
27. P. J. Haas. *Stochastic Petri Nets*. Springer Verlag, 2002.
28. S. P. Hastings and J. D. Murray. The existence of oscillatory solutions in the field-noyes model for the belousov-zhabotinskii reaction. *SIAM Journal on Applied Mathematics*, 28(3):678–688, 1975.
29. T. A. Henzinger. The theory of hybrid automata. In *LICS '96: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, 1996.
30. J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
31. J. Hillston. Fluid flow approximation of PEPA models. In *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QEST05)*, 2005.
32. H. Kitano. *Foundations of Systems Biology*. MIT Press, 2001.
33. H. Kitano. Computational systems biology. *Nature*, 420:206–210, 2002.
34. K. W. Kohn, M. I. Aladjem, J. N. Weinstein, and Y. Pommier. Molecular interaction maps of bioregulatory networks: A general rubric for systems biology. *Molecular Biology of the Cell*, 17(1):1–13, 2006.
35. T. G. Kurtz. Solutions of ordinary differential equations as limits of pure jump markov processes. *Journal of Applied Probability*, 7:49–58, 1970.
36. T. G. Kurtz. Limit theorems for sequences of jump markov processes approximating ordinary differential processes. *Journal of Applied Probability*, 8:244–356, 1971.
37. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with prism: A hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2):128–142, September 2004.
38. H. H. Mcadams and A. Arkin. Stochastic mechanisms in gene expression. *PNAS USA*, 94:814–819, 1997.
39. J. R. Norris. *Markov Chains*. Cambridge University Press, 1997.
40. R. M. Noyes and R. J. Field. Oscillatory chemical reactions. *Annual Review of Physical Chemistry*, 25:95–119, 1974.
41. P. Nurse. Understanding cells. *Nature*, 24, 2003.
42. G.D. Plotkin. A structural approach to operational semantics. *J. Log. Algebr. Program.*, 60-61:17–139, 2004.
43. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C++ : The Art of Scientific Computing*. Cambridge University Press, 2002.
44. C. Priami. Stochastic π-calculus. *The Computer Journal*, 38(6):578–589, 1995.
45. 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.*, 80(1):25–31, 2001.

46. S. Ramsey, D. Orrell, and H. Bolouri. Dizzy: stochastic simulation of large-scale genetic regulatory networks. *Journal of Bioinformatics and Computational Biology*, 3(2):415–436, 2005.
47. 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.
48. A. Regev and E. Shapiro. Cellular abstractions: Cells as computation. *Nature*, 419, 2002.
49. V. A. Saraswat. *Concurrent Constraint Programming*. MIT press, 1993.
50. B. E. Shapiro, A. Levchenko, E. M. Meyerowitz, Wold B. J., and E. D. Mjolsness. Cellerator: extending a computer algebra system to include biochemical arrows for signal transduction simulations. *Bioinformatics*, 19(5):677–678, 2003.
51. S. H. Strogatz. *Non-Linear Dynamics and Chaos, with Applications to Physics, Biology, Chemistry and Engeneering*. Perseus books, 1994.
52. J. M. G. Vilar, H. Yuan Kueh, N. Barkai, and S. Leibler. Mechanisms of noise resistance in genetic oscillators. *PNAS*, 99(9):5991, 2002.
53. D. J. Wilkinson. *Stochastic Modelling for Systems Biology*. Chapman & Hall, 2006.