Modeling Biological Systems in Stochastic Concurrent Constraint Programming

Luca Bortolussi[1] and Alberto Policriti[1]

Dept. of Mathematics and Informatics
University of Udine, Udine, Italy
bortolussi|policriti@dimi.uniud.it

Abstract. We present an application of stochastic Concurrent Constraint Programming (sCCP) for modeling biological systems. We provide a library of sCCP processes that can be used to describe straightforwardly biological networks. In the meanwhile, we show that sCCP proves to be a general and extensible framework, allowing to describe a wide class of dynamical behaviours and kinetic laws.

1 Introduction

Computational Systems Biology is a extremely fertile field, where many different modeling techniques are used [7] in order to capture the intrinsic dynamics of biological systems. These techniques are very different both in spirit and in the mathematics they use. Some of them are based on the well known instrument of *Differential Equations*, mostly ordinary, and therefore they represent phenomena as *continuous and deterministic*, cf. [8] for a survey. On the other side we find *stochastic and discrete* models, that are usually simulated with *Gillespie's algorithm* [14], tailored for simulating (exactly) chemical reactions. In the middle, we find hybrid approaches like the *Chemical Langevin Equation* [11], a stochastic differential equation that bridges partially these two opposite formalisms.

In the last few years a compositional modeling approach based on *stochastic process algebras* (SPA) emerged [22], based on the inspiring parallel between molecules and reactions on one side and processes and communications on the other side. Stochastic process algebras, like stochastic π-calculus [20], have a simple and powerful syntax and a stochastic semantics expressed in terms of Continuous Time Markov Chains [19], that can be simulated with an algorithm equivalent to Gillespie's one. Since their introduction, SPA have been used to model, within the same framework, biological systems described at different level of abstractions, like biochemical reactions [21] and genetic regulatory networks [1].

Stochastic modeling of biological systems works by associating a rate to each active reaction (or, in general, interaction); rates are real numbers representing the frequency or propensity of interactions. All active reactions then undergo a (stochastic) race condition, and the fastest one is executed. Physical justification of this approach can be found in [13]. These rates encode all the quantitative information of the system, and simulations produce discrete temporal traces with variable delay between events.

In this work we show how stochastic Concurrent Constraint Programming [2] (sCCP), another SPA recently developed, can be used for modeling biological systems. sCCP is based on Concurrent Constraint Programming [23] (CCP), a process algebra where

agents interact by posting constraints on the variables of the system in the constraint store, cf. Section 2.

In order to underline the rationale behind the usage of sCCP, we take an high level point of view, providing a general framework connecting elements of biological systems with elements of the process algebra. Subsequently, we show how this general framework gets instantiated when focused on particular classes of biological system, like networks of biochemical reactions and gene regulatory networks.

In our opinion, the advantages of using sCCP are twofold: the presence of both quantitative information and computational capabilities at the level of the constraint systems and the presence of functional rates. This second feature, in particular, allows to encode in the system different forms of dynamical behaviours, in a very flexible way. Quantitative information, on the other hand, allows a more compact representation of models, as part of the details can be described in relations at the level of the store.

The paper is organized as follows: in Section 2 we review briefly sCCP, in Section 3 we describe a high level mapping between biological systems and sCCP, then we instantiate the framework for biochemical reactions (Section 3.1) and gene regulatory networks (Section 3.2). Finally, in Section 4, we draw final conclusions and suggest further directions of investigation.

2 Stochastic Concurrent Constraint Programming

In this section we present a stochastic version [2] of Concurrent Constraint Programming [23], which will be used in the following as a modeling language for biological systems.

2.1 Concurrent Constraint Programming

Concurrent Constraint Programming (CCP [23]) is a process algebra having two distinct entities: agents and constraints. Constraints are interpreted first-order logical formulae, stating relationships among variables (e.g. $X = 10$ or $X + Y < 7$). CCP-Agents compute by adding constraints (`tell`) into a "container" (the *constraint store*) and checking if certain relations are entailed by the current configuration of the constraint store (`ask`). The communication mechanism among agents is therefore asynchronous, as information is exchanged through global variables. In addition to `ask` and `tell`, the language has all the basic constructs of process algebras: non-deterministic choice, parallel composition, procedure call, plus the declaration of local variables. This dichotomy between agents and the constraint store can be seen as a form of separation between computing capabilities (pertaining to the constraint store) and the logic of interactions (pertaining to the agents). From a general point of view, the main difference between CCP and π-calculus resides really in the computational power of the former. π-calculus, in fact, has to describe everything in terms of communications only, a fact that may result in cumbersome programs in all those situations in which "classical" computations are directly or indirectly involved.

The *constraint store* is defined as an algebraic lattice structure, using the theory of cylindric algebras [15]. Essentially, we first choose a first-order language together with an interpretation, which defines a semantical entailment relation (required to be decidable). Then we fix a set of formulae, closed under finite conjunction, as the

$$Program = D.A$$

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

$$\pi = \mathrm{tell}_\lambda(c) \mid \mathrm{ask}_\lambda(c)$$
$$M = \pi.A \mid \pi.A.p(\mathbf{y}) \mid M + M$$
$$A = \mathbf{0} \mid \mathrm{tell}_\infty(c).A \mid \exists_x A \mid M \mid (A \parallel A)$$

Table 1. Syntax of of sCCP.

primitive constraints that the agents can add to the store. The algebraic lattice is obtained by considering subsets of these primitive constraints, closed by entailment and ordered by inclusion. The least upper bound operation in the lattice is denoted by $\sqcup$ and it basically represents the conjunction of constraints. In order to model local variables and parameter passing, the structure is enriched with cylindrification and diagonalization operators, typical of cylindric algebras [15]. These operators allow to define a sound notion of substitution of variables within constraints. In the following we denote the entailment relation by $\vdash$ and a generic constraint store by $\mathcal{C}$. We refer to [6, 24, 23] for a detailed explanation of the constraint store.

2.2 Syntax of sCCP

The stochastic version of CCP (sCCP [2]) is obtained by adding a stochastic duration to the instructions interacting with the constraint store $\mathcal{C}$, i.e. `ask` and `tell`. More precisely, each instruction is associated with a continuous random variable T, representing the time needed to perform the corrisponding operations in the store (i.e. adding or checking the entailment of a constraint). This random variable is exponentially distributed (cf. [19]), i.e. its probability function is

$$f(\tau) = \lambda e^{-\lambda\tau}, \tag{2.1}$$

where λ is a positive real number, called the rate of the exponential random variable, which can be intuitively seen as the expected frequency per unit of time.

In our framework, the rates associated to `ask` and `tell` are functions

$$\lambda : \mathcal{C} \to \mathbb{R}^+,$$

depending on the current configuration of the constraint store. This means that the speed of communications can vary according to the particular state of the system, though in every state of the store the random variables are perfectly defined (their rate is evaluated to a real number). This fact gives to the language a remarkable flexibility in modeling biological systems, see Section 3 for further material on this point.

The syntax of sCCP can be found in Table 1. An sCCP program consists in a list of procedures and in the starting configuration. Procedures are declared by specifying their name and their free variables, treated as formal parameters. Agents, on the other

$$(IR1)\quad \langle \text{tell}_\infty(c).A, d\rangle \longrightarrow \langle A, d \sqcup c\rangle$$

$$(IR2)\quad \langle p(\mathbf{x}), d\rangle \longrightarrow \langle A[\mathbf{x}/\mathbf{y}], d\rangle \quad \text{if } p(\mathbf{y}) :- A$$

$$(IR3)\quad \langle \exists_x A, d\rangle \longrightarrow \langle A[y/x], d\rangle \quad \text{with } y \text{ fresh}$$

$$(IR4)\quad \frac{\langle A_1, d\rangle \longrightarrow \langle A_1', d'\rangle}{\langle A_1 \parallel A_2, d\rangle \longrightarrow \langle A_1' \parallel A_2, d'\rangle}$$

Table 2. Instantaneous transition for stochastic CCP

hand, are defined by the grammar in the last three lines of Table 1. There are two different actions with temporal duration, i.e. `ask` and `tell`, identified by π. Their rate λ is a function as specified above. These actions can be combined together into a guarded choice M (actually, a mixed choice, as we allow both ask and tell to be combined with summation). In the definition of such choice, we force procedure calls to be always guarded. In fact, they are instantaneous operations, thus guarding them by a timed action allows to avoid instantaneous infinite recursive loops, like those possible in $p :- A \parallel p$. In summary, an agent A can choose between different actions (M), it can perform an instantaneous `tell`, it can declare a variable local ($\exists_x A$) or it can be combined in parallel with other agents.

The syntax presented here is slightly different from that of [2]. In fact, the class of instantaneous actions is expanded: in [2] it contained only the declaration of local variables, while here it contains also procedure call and a version of `tell`. Nevertheless, the congruence relation defined in [2], ascribing the usual properties to the operators of the language (e.g. associativity and commutativity to $+$ and $\parallel$), remains the same. The configurations of sCCP programs will vary in the quotient space modulo this congruence relation, denoted by $\mathcal{P}$.

2.3 Operational Semantics of sCCP

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. This is also a novelty w.r.t. [2], though in the previous version an instantaneous transition was implicitly defined in order to deal with local variables. The basic idea of this operational semantics is to apply the two transitions in an interleaved way: 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 configuration of the store. Therefore, a configuration will be a point in the space $\mathcal{P} \times \mathcal{C}$.

The recursive definition of the instantaneous transition $\longrightarrow \subseteq (\mathcal{P} \times \mathcal{C}) \times (\mathcal{P} \times \mathcal{C})$ is shown in Table 2. Rule ($IR1$) models the addition of a constraint in the store through the least upper bound operation of the lattice. Recursion corresponds to rule ($IR2$), which consists in substituting the actual variables to the formal parameters in the

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

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

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

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

Table 3. Stochastic transition relation for stochastic CCP

definition of the procedure called. In rule $(IR3)$, local variables are replaced by fresh global variables, while in $(IR4)$ the other rules are extended compositionally. Observe that we do not need to deal with summation operator at the level of instantaneous transition, as all the choices are guarded by (stochastically) timed actions. The syntactic restrictions imposed to instantaneous actions guarantee that $\longrightarrow$ can be applied only for a finite number of steps. Moreover it can be proven that it is confluent. Given a configuration $\langle A, d\rangle$ of the system, we denote by $\overrightarrow{\langle A, d\rangle}$ 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 $\Longrightarrow \subseteq (\mathcal{P} \times \mathcal{C}) \times [0,1] \times \mathbb{R}^+ \times (\mathcal{P} \times \mathcal{C})$ is defined in Table 3. This transition is labeled by two numbers: intuitively, the first one is the probability of the transition, while the second one is its global rate, see Section 2.4 for further details. Rule $(SR1)$ deals with timed tell action, and works similarly to rule $(IR1)$. Rule $(SR2)$, instead, defines the behaviour of the ask instruction: it is active only if the asked constraint is entailed by the current configuration of the constraint store. Rules (SR3) and (SR4), finally, deal with the choice and the parallel construct. 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. In Table 3 we use the function rate : $\mathcal{P} \times \mathcal{C} \to \mathbb{R}$, assigning to each agent its global rate. It is defined as follows:

Definition 1. *The function* rate : $\mathcal{P} \times \mathcal{C} \to \mathbb{R}$ *is defined by*

1. $\mathrm{rate}\,(\mathbf{0}, d) = 0$;
2. $\mathrm{rate}\,(\mathrm{tell}_\lambda(c).A, d) = \lambda(d)$;
3. $\mathrm{rate}\,(\mathrm{ask}_\lambda(c).A, d) = \lambda(d)$ *if* $d \vdash c$;
4. $\mathrm{rate}\,(\mathrm{ask}_\lambda(c).A, d) = 0$ *if* $d \nvdash c$;
5. $\mathrm{rate}\,(M_1 + M_2, d) = \mathrm{rate}\,(M_1, d) + \mathrm{rate}\,(M_2, d)$.
6. $\mathrm{rate}\,(A_1 \parallel A_2, d) = \mathrm{rate}\,(A_1, d) + \mathrm{rate}\,(A_2, d)$;

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 (consider the case of $\mathrm{tell}_\lambda(c) + \mathrm{tell}_\lambda(c)$). Starting from this labeled graph, we can build a Continuous Time Markov Chain (cf. [19] and next section) as follows: substitute each label (p, λ) with the real number $p\lambda$ and add up the numbers labeling edges connecting the same nodes. More details about the operational semantics can be found in [2].

2.4 Continuous Time Markov Chains and Gillespie's Algorithm

A Continuous Time Markov Chain (CTMC for short) 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, $\forall n, t_1, \ldots, t_n, s_1, \ldots, s_n$:

$$P\{X_{t_n} = s_n \mid X_{t_{n-1}} = s_{n-1}, \ldots, X_{t_1} = s_1\} = P\{X_{t_n} = s_n \mid X_{t_{n-1}} = s_{n-1}\}. \quad (2.2)$$

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 (defined by the probability density (2.1)). 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 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 can be used in a Monte-Carlo simulation algorithm. Suppose to be in state s; then draw two random numbers, one according to the probability given by the normalized rates, and the second according to an exponential probability distribution with parameter equal to the exit rate. Then choose the next state according to the first random number, and increase the time according to the second. The procedure sketched here is essentially the content of the Gillespie's algorithm [13, 14], originally derived in the context of stochastic simulation of chemical reactions. Indeed, the stochastic description of chemical reactions is exactly a Continuous Time Markov Chain [12].

2.5 Stream Variables

In the use of sCCP as a modeling language for biological systems, many variables will represent quantities that vary over time, like the number of molecules of certain chemical species. In addition, the functions returning the stochastic rate of communications will depend only on those variables. Unfortunately, the variables we have at our disposal in CCP are rigid, in the sense that, whenever they are instantiated, they keep that value forever. However, time-varying variables can be easily modeled as growing lists with an unbounded tail: $X = [a_1, \ldots, a_n|T]$. When the quantity changes, we simply need to add the new value, say b, at the end of the list by replacing the old tail

variable with a list containing b and a new tail variable: $T = [b|T']$. When we need to compute a function depending on the current value of the variable X, we need to extract from the list the value immediately preceding the unbounded tail. This can be done by defining the appropriate predicates in the first-order language over which the constraint store is built. As these variables have a special status in the presentation hereafter, we will refer to them as *stream variables*. In addition, we will use a simplified notation that hides all the details related to the list update. For instance, if we want to add 1 to the current value of the stream variable X, we will simply write $X = X + 1$. The intended meaning of this notation is clearly: "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)".

2.6 Implementation

We have developed an interpreter for the language that can be used for running simulations. The simulation engine is based on the Gillespie's Algorithm, therefore it performs a Monte-Carlo simulation of the underlying CTMC. The memoryless property of the CTMC guarantees that we do need to generate all its nodes to perform a simulation, but we need to store only the current state. By syntactic analysis of the current set of agents in execution, we can construct all the exit transitions and compute their rates, evaluating rate functions w.r.t. the current configuration of the store (actually, those functions depend only on stream variables, thus their computation has two steps: extract the current value of the variables and evaluate the function). Then we apply the Gillespie's procedure to determine the next state and the elapsed time, updating the system by modifying the current set of agents and the constraint store according to the chosen transition.

The interpreter is written in SICStus Prolog [10]. It is composed by a parser, accepting a program written in sCCP and converting it into an internal list-based representation. The main engine operates therefore by inspecting and manipulating the lists representing the program. The constraint store is managed using the constraint solver on finite domains of SICStus. Stream variables are not represented as lists, but rather as global variables using the meta-predicates `assert` and `retract` of Prolog. The choice of working with finite domains is mainly related to the fact that the biological systems analyzed can be described using only integer values[1].

In every execution cycle we need to inspect all terms in order to check if they enable a transition. Therefore, the complexity of each step is linear in the size of the (representation) of the program. This can be easily improved by observing that an enabled transition that is not executed remains enabled also in the future.

The correctness of the virtual machine can be proven by showing that it simulates exactly the same CTMC defined by the sCCP program. This can be done by showing that the exit rate and the probability distribution on exiting transitions are computed correctly, according to the operational semantics of sCCP.

Measurable Entities	Stream Variables
Logical Entities	Processes (Control Variables)
Interactions	Processes

Table 4. Schema of the mapping between elements of biological systems (left) and sCCP (right).

3 Modeling Biological Systems

Taking an high level point of view, biological systems can be seen as composed essentially by two ingredients: (biological) entities and interactions among those entities. For instance, in biochemical reaction networks, the molecules are the entities and the chemical reactions are the possible interactions, see [22] and Section 3.1. In gene regulatory networks, instead, the entities into play are genes and regulatory proteins, while the interactions are production and degradation of proteins, and repression and enhancement of gene's expression, cf. [1] and Section 3.2. In addition, entities fall into two separate classes: measurable and logical. Measurable entities are those present in a certain quantity in the system, like proteins or other molecules. Logical entities, instead, have a control function (like gene gates in [1]), hence they are neither produced nor degraded. Note that logical entities are not real world entities, but rather they are part of the models.

The translation scheme between the previously described elements and sCCP objects is summarized in Table 4. Measurable entities are associated exactly to stream variables introduced at the end of Section 2. Logical entities, instead, are represented as processes actively performing control activities. In addition, they can use variables of the constraint store either as control variables or to exchange information. Finally, each interaction is associated to a process modifying the value of certain measurable stream variables of the system.

Associating variables to measurable entities means that we are representing them as part of the environment, while the active agents are associated to the different actions capabilities of the system. These actions have a certain duration and a certain propensity to happen: a fact represented here in the standard way, i.e. associating to each action a stochastic rate. Actually, the speed of most of these actions depends on the quantity of the basic entities they act on. This fact shows clearly the need for having functional rates, which can be used to describe these dependencies explicitly.

In the next subsections we instantiate this general scheme, in order to deal with two classes of biological systems: networks of biochemical reactions and genetic regulatory networks.

[1] The real valued rates and the stochastic evolution are tight with the definition of the semantics and not with the syntax of the language, thus we do not need to represent them in the store.

3.1 Modeling Biochemical Reactions

Network of biochemical reactions are usually modeled through chemical equations of the form $R_1 + \ldots + R_n \rightarrow_k P_1 + \ldots + P_m$, where the n reactants R_i's (possibly in multiple copies) are transformed into the m products P_j's. In the equation above, either n or m can be equal to zero; the case $m = 0$ represents a degradation reaction, while the case $n = 0$ represents an external feeding of the products, performed by an experimenter. Actually, the latter is not a proper chemical reaction but rather a feature of the environmental setting, though it is convenient to represent it within the same scheme. Each reaction has an associated rate k, representing essentially its basic speed. The actual rate of the reaction is $k \cdot [R_1] \cdots [R_n]$, where $[R_i]$ denotes the number of molecules R_i present in the system. There are cases when a more complex expression for the rate of the reaction is needed, see [25] for further details. For instance, one may wish to describe an enzymatic reaction using a Michaelis-Menten kinetic law [8], rather than modeling explicitly the enzyme-substrate complex formation (as simple interaction/communication among molecules, cf. example below). A set of different *biochemical arrows* (corresponding to different biochemical laws) is shown in Table 5; this list is not exhaustive, but rather a subset of the one presented in [25]. Adding further arrows is almost always straightforward.

In Table 5, we also show how to translate biochemical reactions into sCCP processes. The basic reaction $R_1 + \ldots + R_n \rightarrow_k P_1 + \ldots P_m$ is associated to a process that first checks if all the reactants needed are present in the system (asking if all $[R_i]$ are greater than zero), then it modifies the variables associated to reactants and products, and finally it calls itself recursively. Note that all the `tell` instructions have infinite rate, hence they are instantaneous transitions. The rate governing the speed of the reaction is the one associated to `ask` instruction. This rate is nothing but the function $r_{MA}(k, X_1, \ldots, X_n) = k \cdot X_1 \cdots X_n$ representing mass action dynamics. Note that $\rightleftharpoons$ is a shorthand for the forward and the backward reactions. The arrow $\mapsto^E_{K,V_0}$ has a different dynamics, namely Michaelis-Menten kinetics: $r_{MM}(K, V_0, S) = \frac{V_0 S}{S+K}$. This reaction approximates the conversion of a substrate into a product due to the catalytic action of enzyme E when the substrate is much more abundant than the enzyme (quasi-steady state assumption, cf. [8]). The last arrow, instead, is associated to Hill's kinetics. The dynamics represented here is an improvement on the Michaelis-Menten law, where the exponent h encodes some information about the spatial behaviour of the reaction.

Comparing the encoding of biochemical reaction into sCCP with the encoding into other process algebras like π-calculus [22], we note that the presence of functional rates gives much more flexibility in the modeling phase. In fact, this form of rates allows to describe dynamics that are different from Mass Action. Notable examples are exactly Michaelis-Menten's and Hill's cases, represented by the last two arrows. This is not possible wherever only constant rates are present, as the definition of the operational semantics constrain the dynamics to be Mass-Action like. More comments about this fact can be found in [3].

Example: Enzymatic Reaction As a first and simple example, we show the model of an enzymatic reaction. We provide two different descriptions, one using a mass action kinetics, the other using a Michaelis-Menten one, see Table 5.

In the first case, we have the following set of reactions:

$$S + E \rightleftharpoons^{k_1}_{k_{-1}} ES \rightarrow_{k_2} P + E; \quad P \rightarrow_{k_{deg}}; \quad \rightarrow_{k_{prod}} S, \tag{3.1}$$

$R_1 + \ldots + R_n \to_k P_1 + \ldots + P_m$	$\text{reaction}(k, [R_1, \ldots, R_n], [P_1, \ldots, P_m]) :-$ $\quad \text{ask}_{r_{MA}(k,R_1,\ldots,R_n)} \left(\bigwedge_{i=1}^{n}(R_i > 0)\right).$ $\quad \Big(\|_{i=i}^{n} \text{tell}_\infty(R_i = R_i - 1) \parallel$ $\quad \|_{j=1}^{m} \text{tell}_\infty(P_j = P_j + 1)\Big).$ $\quad \text{reaction}(k, [R_1, \ldots, R_n], [P_1, \ldots, P_m])$
$R_1 + \ldots + R_n \rightleftharpoons_{k_2}^{k_1} P_1 + \ldots + P_m$	$\text{reaction}(k_1, [R_1, \ldots, R_n], [P_1, \ldots, P_m]) \parallel$ $\text{reaction}(k_2, [P_1, \ldots, P_m], [R_1, \ldots, R_n])$
$S \mapsto_{K,V_0}^{E} P$	$\text{mm\_reaction}(K, V_0, S, P) :-$ $\quad \text{ask}_{r_{MM}(K,V_0,S)}(S > 0).$ $\quad (\text{tell}_\infty(S = S - 1) \parallel tell_\infty(P = P + 1)).$ $\quad \text{mm\_reaction}(K, V_0, S, P)$
$S \mapsto_{K,V_0,h}^{E} P$	$\text{hill\_reaction}(K, V_0, h, S, P) :-$ $\quad \text{ask}_{r_{Hill}(K,V_0,h,S)}(S > 0).$ $\quad (\text{tell}_\infty(S = S - h) \parallel tell_\infty(P = P + h)).$ $\quad \text{Hill\_reaction}(K, V_0, h, S, P)$
$r_{MA}(k, X_1, \ldots, X_n) = k \cdot X_1 \cdots X_n;$	where $r_{MM}(K, V_0, S) = \frac{V_0 S}{S+K}; \quad r_{Hill}(k, V_0, h, S) = \frac{V_0 S^h}{S^h + K^h}$

Table 5. Translation into sCCP of different biochemical reaction types, taken from the list of [25]. The reaction process models a mass-action-like reaction. It takes in input the basic rate of the reaction, the list of reactants, and the list of products. These list can be empty, corresponding to degradation and external feeding. The process has a blocking guard that checks if all the reactants are present in the system. The rate of the ask is exactly the global rate of the reaction. If the process overcomes the guard, it modifies the quantity of reactants and products and then it calls itself recursively. The reversible reaction is modeled as the combination of binding and unbinding. The third arrow corresponds to a reaction with Michaelis-Menten kinetics. The corresponding process works similarly to the reaction one, but the rate function is different. Here, in fact, the rate function is the one expressing Michaelis-Menten kinetics. See Section 3.1 for further details. The last arrow replaces Michaelis-Menten kinetics with Hill's one (see end of Section 3.1).

corresponding to a description of an enzymatic reaction that takes into account also the enzyme-substrate complex formation. Specifically, substrate S and enzyme E can bind and form the complex ES. This complex can either dissociate back into E and S, or be converted into the product P and again enzyme E. Moreover, in this particular system we added degradation of P and external feeding of S, in order to have continuous production of P. The sCCP model of this reaction can be found in Table 6. It is simply composed by 5 reaction agents, one for each arrow of the equations (3.2). The three reactions involving the enzyme are grouped together under the predicate `enz_reaction{k1,k-1,k2,S,E,ES,P}`, that will be used in following subsections.

$$\text{enz\_reaction}(k_1, k_{-1}, k_2, S, E, ES, P) :-$$
$$\text{reaction}(k_1, [S, E], [ES]) \parallel \text{reaction}(k_{-1}, [ES], [E, S]) \parallel \text{reaction}(k_2, [ES], [E, P]).$$

$$\text{enz\_reaction}(k_1, k_{-1}, k_2, S, E, ES, P) \parallel \text{reaction}(k_{prod}, [], [S]) \parallel \text{reaction}(k_{deg}, [P], [])$$

Table 6. sCCP program for an enzymatic reaction with mass action kinetics. The first block defines the predicate enz_reaction$(k_1, k_{-1}, k_2, S, E, ES, P)$, while the second block is the definition of the entire program. The predicate reaction has been defined in Table 5.

Simulations were performed with the simulator described in Section 2.6, and the trend of product P is plotted in Figure 1 (left). Parameters of the system were chosen in order to have, at regime, almost all the enzyme molecules in the complexed state, see caption of Figure 1 (top) for details.

For this simple enzymatic reaction, the quasi-steady state assumption holds [8], therefore replacing the substrate-enzyme complex formation with a Michaelis-Menten kinetics should leave the system behaviour unaltered. This intuition is confirmed by Figure 1 (bottom), showing the plot of the evolution over time of product P for the following system of reactions:

$$S \mapsto^{E}_{K,V_0} P; \quad P \to_{k_{deg}}; \quad \to_{k_{prod}} S,$$

whose sCCP can be derived easily from Table 5.

A slightly more complicated version of the above example is the case in which some level of cooperativity of the enzyme is to be modeled (Hill's case). The set of reactions in this case is an extension of the above one and can be written as:

$$n \times S + E \rightleftharpoons^{k_1}_{k_{-1}} ES_n \to_{k_2} n \times P + E; \quad P \to_{k_{deg}}; \quad \to_{k_{prod}} S. \tag{3.2}$$

In this case the sCCP program is a straightforward extension of the previous one:

$$\text{n\_enz\_reaction}(k_1, k_{-1}, k_2, S, E, ES, P) :-$$
$$\text{reaction}(k_1, [n \times S, E], [ES_n]) \parallel \text{reaction}(k_{-1}, [ES_n], [E, n \times S]) \parallel$$
$$\text{reaction}(k_2, [ES_n], [E, n \times P]).$$

while the rest of the coding is entirely similar to the previous case.

Also in this case a comparison with the reaction obtained with the computed Hill coefficient

$$S \mapsto^{E}_{K,V_0,n} P; \quad P \to_{k_{deg}}; \quad \to_{k_{prod}} S,$$

can be easily carried out. Notice that the Hill's exponent corresponds exactly to the degree of cooperativity of the enzyme.

Also a more refined approach to the case of Hill's kinetics is possible, decomposing the n-fold reaction in a series of n separated by Mass Action equation simulations.

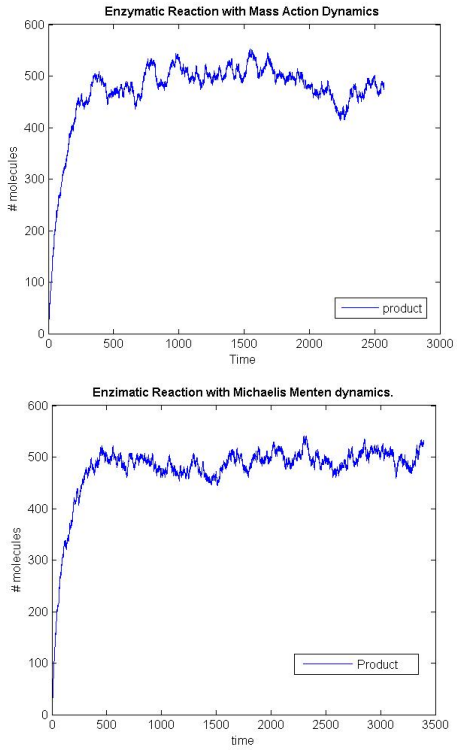


Fig. 1. (**top**) Mass Action dynamics for an enzymatic reaction. The graph shows the time evolution of the product P. Rates used in the simulation are $k_1 = 0.1$, $k_{-1} = 0.001$, $k_2 = 0.5$, $k_{deg} = 0.01$, $k_{prod} = 5$. Enzyme molecules E are never degraded (though they can be in the complex status), and initial value is set to $E = 10$. Starting value for S is 100, while for P is zero. Notice that the rate of complexation of E and S into ES and the dissociation rate of ES into E and P are much bigger than the dissociation rate of ES into E and S. This implies that almost all the molecules of E will be found in the complexed form. (**bottom**) Michaelis-Menten dynamics for an enzymatic reaction. The graph shows the time evolution of the product P. Rates k_{deg} and k_{prod} are the same as above, whilst $K = 5.01$ and $V_0 = 5$. These last values are derived from mass action rates in the standard way, i.e. $K = \frac{K_2 + k_{-1}}{k_1}$ and $V_0 = k_2 E_0$, where E_0 is the starting quantity of enzyme E, cf. [8] for a derivation of these expressions. Notice that the time spawn by this second temporal series is longer than the first one, despite the fact that simulations lasted the same number of elementary steps (of the labeled transition system of sCCP). This is because the product formation in the Michaelis-Menten dynamics model is a one step reaction, while in the other system it is a two step reaction (with a possible loop because of the dissociation of ES into E and S).

$$
\begin{array}{l}
\text{enz\_reaction}(k_a, k_d, k_r, KKK, E1, KKKE1, KKKS) \parallel \\
\text{enz\_reaction}(k_a, k_d, k_r, KKKS, E2, KKKSE2, KKK) \parallel \\
\text{enz\_reaction}(k_a, k_d, k_r, KK, KKKS, KKKKKS, KKP) \parallel \\
\text{enz\_reaction}(k_a, k_d, k_r, KKP, KKP1, KKPKKP1, KK) \parallel \\
\text{enz\_reaction}(k_a, k_d, k_r, KKP, KKKS, KKPKKKS, KKPP) \parallel \\
\text{enz\_reaction}(k_a, k_d, k_r, KKPP, KKP1, KKPPKKP1, KKP) \parallel \\
\text{enz\_reaction}(k_a, k_d, k_r, K, KKPP, KKKPP, KP) \parallel \\
\text{enz\_reaction}(k_a, k_d, k_r, KP, KP1, KPKP1, K) \parallel \\
\text{enz\_reaction}(k_a, k_d, k_r, KP, KKPP, KPKKPP, KPP) \parallel \\
\text{enz\_reaction}(k_a, k_d, k_r, KPP, KP1, KPPKP1, KP)
\end{array}
$$

Table 7. sCCP code for the MAP-Kinase signaling cascade. The enz_reaction predicate has been defined in Section 3.1. For this example, we set the complexation rates (k_a), the dissociation rates (k_d) and the product formation reaction rates (k_r) equal for all the reactions involved. For the actual values used in the simulation, refer to Figures 3 and 4.

Example: MAP-Kinase Cascade A cell is not an isolated system, but it communicates with the external environment using complex mechanisms. In particular, a cell is able to react to external signals, i.e. to signaling proteins (like hormones) present in the proximity of the external membrane. Roughly speaking, this membrane is filled with receptor proteins, that have a part exposed toward the external environment capable of binding with the signaling protein. This binding modifies the structure of the receptor protein, that can now trigger a chain of reactions inside the cell, transmitting the signal straight to the nucleus. In this signaling cascade a predominant part is performed by a family of proteins, called Kinase, that have the capability of phosphorylating other proteins. Phosphorylation is a modification of the protein fold by attaching a phosphorus molecule to a particular amino acid of the protein. One interesting feature of these cascades of reactions is that they are activated only if the external stimulus is strong enough. In addition, the activation of the protein at the end of the chain of reactions (usually an enzyme involved in other regulation activities) is very quick. This behaviour of the final enzyme goes under the name of ultra-sensitivity [17].

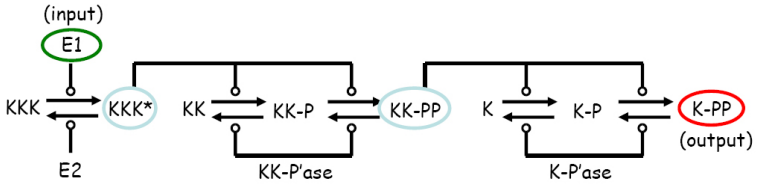


Fig. 2. Diagram of the MAP-Kinase cascade. The round-headed arrow schematically represents an enzymatic reaction, see Section 3.1 for further details. This diagram has been stolen from a presentation of Luca Cardelli, held in Dobbiaco, September 2005.

In Figure 2 a particular signaling cascade is shown, involving MAP-Kinase proteins. This cascade has been analyzed using differential equations in [17] and then modeled and simulated in stochastic Pi-Calculus in [4] (CONTROLLARE SE E' LA CITAZIONE GIUSTA). We can see that the external stimulus, here generically represented by the enzyme E_1, triggers a chain of enzymatic reactions. MAPKKK is converted into an active form, called MAPKKK*, that is capable of phosphorylating the protein MAPKK in two different sites. The diphosphorylated version MAPKK-PP of MAPKK is the enzyme stimulating the phosphorylation of another Kinase, i.e. MAPK. Finally, the diphosphorylated version MAPK-PP of MAPK is the output of the cascade.

The sCCP program describing MAP-Kinase cascade is shown in Table 7. The program itself is very simple, and it uses the mass action description of an enzymatic reaction (cf. Table 5). It basically consists in a list of the reactions involved, put in parallel. The real problem in studying such a system is in the determination of its 30 parameters, corresponding to the basic rates of the reactions involved. In addition, we need to fix a set of initial values for the proteins that respects their usual concentrations in the cell. Following [4], in Figure 3 we skip this problem and assign a value of 1.0 to all basic rates, while putting 100 copies of MAPKKK, MAPKK and MAPK, 5 copies of E2, MAPKK-P'ase, and MAPK-P'ase and just 1 copy of the input E1. This simple choice, however, is enough to predict correctly all the expected properties: the MAPK-PP time evolution, in fact, follows a sharp trend, jumping from zero to 100 in a short time. Remarkably, this property is not possessed by MAPKK-PP, the enzyme in the middle of the cascade. Therefore, this switching behaviour exhibited by MAPK-PP is intrinsically connected with the double chain of phosphorylations, and cannot be obtained by a simpler mechanism. Notice that the fact that the network works as expected using an arbitrary set of rates is a good argument in favor of its robustness and resistance to perturbations.

In Figure 4, instead, we choose a different set of parameters, as suggested in [17] (cf. its caption). We also let the input strength vary, in order to see if the activation effect is sensitive to its concentration. As we can see, this is the case: for a low value of the input, no relevant quantity of MAPK-PP is present in the system.

3.2 Modeling Gene Regulatory Networks

In a cell, only a subset of genes are expressed at a certain time. Therefore, an important mechanism of the cell is the regulation of gene expression. This is obtained by specific proteins, called *transcription factors*, that bind to the promoter region of genes (the portion of DNA preceding the coding region) in order to enhance or repress their transcription activity. These transcription factors are themselves produced by genes, thus the overall machinery is a networks of genes producing proteins that regulate other genes. The resulting system is highly complex, containing several positive and negative feedback loops, and usually very robust. This intrinsic complexity is a strong argument in favor of the use of a mathematical formalism to describe and analyze them. In the literature, different modeling techniques are used, see [7] for a Survey. However, we focus on a modeling formalism based on stochastic π-calculus [1].

In [1], the authors propose to model gene networks using a small set of "logical" gates, called *gene gates*, encoding the possible regulatory activities that can be performed on a gene. Specifically, there are three types of gene gates: *nullary gates*, *positive*

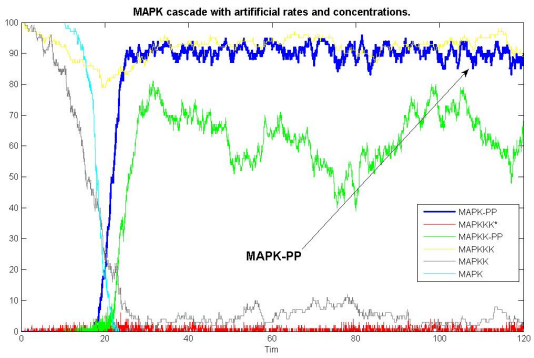


Fig. 3. Temporal trace for some proteins involved in the MAP-Kinase cascade. Traces were generated simulating the sCCP program of Table 7. In this simulation, the rates k_a, k_d, k_r were all set to one. We can notice the sharp increase in the concentration of the output enzyme, MAPK-PP, and its stability in the high expression level. The enzyme MAPKK-PP, the activator of MAPK phosphorylations, instead has a more unstable trend of expression.

gates and *negative gates*. Nullary gates represent genes with transcriptional activity, but with no regulation. Positive gates are genes whose transcription rate can be increased by a transcription factor. Finally, negative gates represent genes whose transcription can be inhibited by the binding of a specific protein. At the level of abstraction of [1], the product of a gene gate is not a mRNA molecule, but directly the coded protein. These product proteins are then involved in the regulation activity of the same or of other genes and can also be degraded.

We propose now an encoding of gene gates within sCCP framework, in the spirit of Table 4. Proteins are measurable entities, thus they are encoded as stream variables; gene gates, instead, are logical control entities and they are encoded as agents. The degradation of proteins is modeled by the reaction agent of Table 5. In Table 8 we present the sCCP agents associated to gene gates. A nullary gate simply increases the quantity of the protein it produces at a certain specified rate. Positive gates, instead, can produce their coded protein at the basic rate or they can enter in an enhanced state where production happens at an higher rate. Entrance in this excited state happens at a rate proportional to the quantity of transcription factors present in the system. Negative gates behave similarly to positive ones, with the only difference that they can enter an inhibited state instead of an enhanced one. After some time, the inhibited gate returns to its normal status. A specific gene, generally, can be regulated by more than transcription factor. This can be obtained by composing in parallel the different gene gates.

Example: Bistable Circuit The first example, taken from [1], is a gene network composed by two negative gates repressing each other, see Figure 5. The sCCP model for this simple network comprehends two negative gates: the first producing protein A and repressed by protein B, the second producing protein B and repressed by protein A. In addition, there are the degradation reactions for proteins A and B. This network

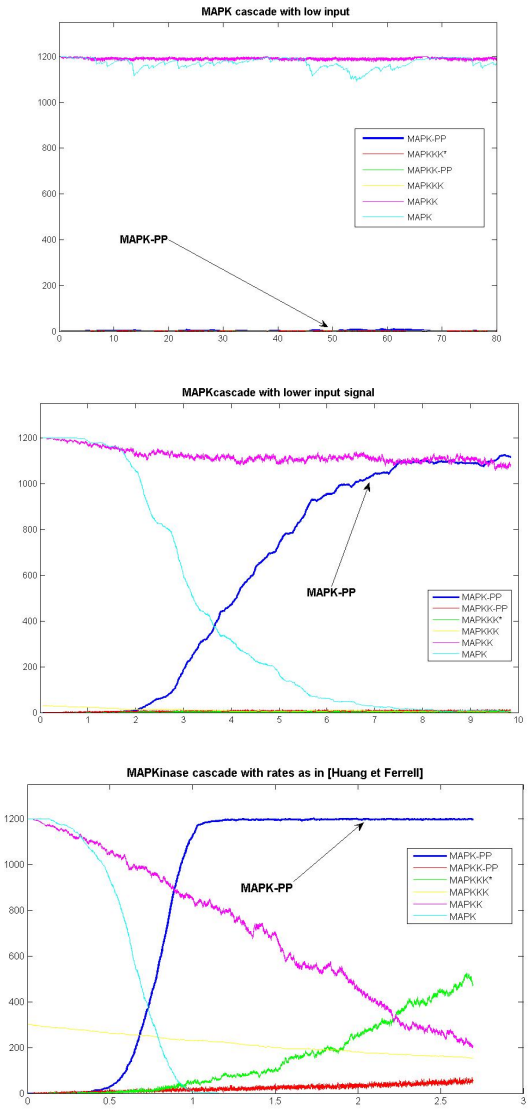
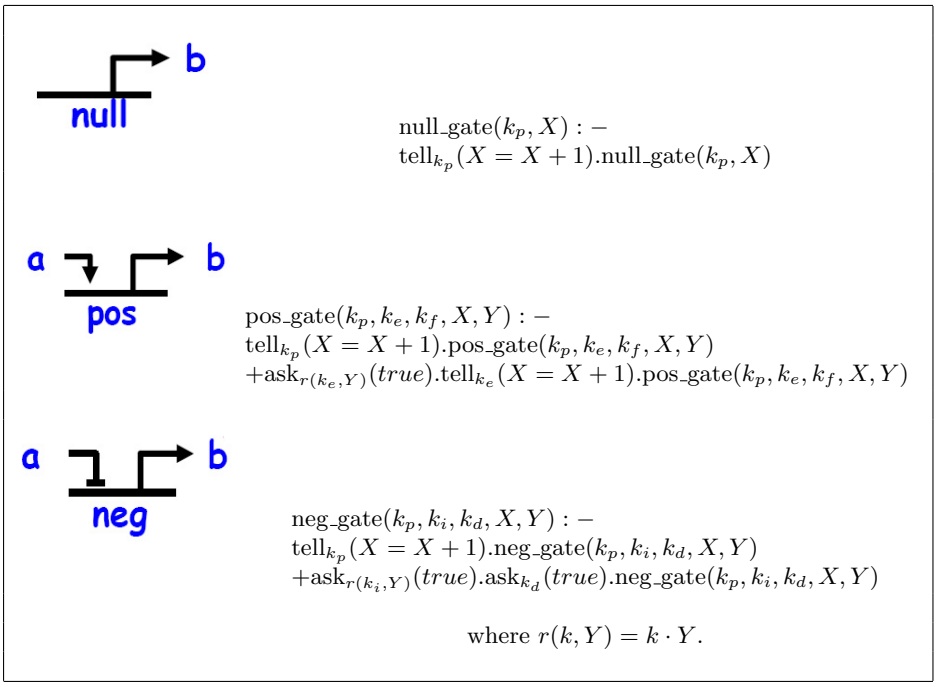


Fig. 4. Comparison of the temporal evolution of the MAP-Kinase cascade for different concentrations of the enzyme MAPKKK. As argued in [17], this is equivalent to the variation of the input signal E1. Rates are equal for all reactions, and have the following values: $k_a = 1$, $k_d = 150$, $k_r = 150$. This corresponds to a Michaelis-Menten rate of 300 for all the enzymatic reactions. The initial quantity of MAPKK and MAPK is set to 1200, the initial quantity of phosphatase MAPK-P'ase is set to 120, the initial quantity of other phosphatase and the enzyme E2 is set to 5, and the initial quantity of E1 is 1. (**top**) The initial quantity of MAPKKK is 3. We can see that there is no sensible production of MAPK-PP. (**middle**) The initial quantity of MAPKKK is 30. Enzyme MAPK-PP is produced but its trend is not sharp, as expected. (**bottom**) The initial quantity of MAPKKK is 300. The system behaves as expected. We can see that the increase in the concentration of MAPK-PP is very sharp, while MAPKK-PP grows very slowly in comparison.



$$\text{null\_gate}(k_p, X) :-$$
$$\text{tell}_{k_p}(X = X + 1).\text{null\_gate}(k_p, X)$$

$$\text{pos\_gate}(k_p, k_e, k_f, X, Y) :-$$
$$\text{tell}_{k_p}(X = X + 1).\text{pos\_gate}(k_p, k_e, k_f, X, Y)$$
$$+\text{ask}_{r(k_e,Y)}(true).\text{tell}_{k_e}(X = X + 1).\text{pos\_gate}(k_p, k_e, k_f, X, Y)$$

$$\text{neg\_gate}(k_p, k_i, k_d, X, Y) :-$$
$$\text{tell}_{k_p}(X = X + 1).\text{neg\_gate}(k_p, k_i, k_d, X, Y)$$
$$+\text{ask}_{r(k_i,Y)}(true).\text{ask}_{k_d}(true).\text{neg\_gate}(k_p, k_i, k_d, X, Y)$$

where $r(k, Y) = k \cdot Y$.

Table 8. Scheme of the translation of gene gates into sCCP programs. The null gate is modeled as a process continuously producing new copies of the associated protein, at a fixed rate k_p. The negative gate is modeled as a process that can either produce a new protein or enter in an repressed state due to the binding of the repressor. This binding can happen at a rate proportional to the concentration of the repressor. After some time, the repressor unbinds and the gate return in the normal state. The enhancing of activators in the pos gate, instead, is modeled here in an "hit and go" fashion. The enhancer can hit the gate and make it produce a protein at an higher rate than usual. The hitting rate is proportional to the number of molecules of the stimulating protein.

is bistable: only one of the two proteins is expressed. If the initial concentrations of A and B are zero, then the stochastic fluctuations happening at the beginning of the simulations decide which of the two fix points will be chosen. In Figure 5 we show one possible outcome of the system, starting with zero molecules of A and B. In this case, protein A wins the competition. Notice that the high sensitivity of this system makes it unsuitable for biological system.

Example: Repressilator The repressilator [9] is a synthetic biochemical clock composed of three genes expressing three different proteins, **tetR**, **λcI**, **LacI**, that have a regulatory function in each other's gene expression. In particular, protein **tetR** inhibits the expression of protein **λcI**, while 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 tree proteins with a constant frequency.

The model we present here is extracted from [1], and it is constituted by three negative gene gates repressing each other in cycle (see Figure 6). The result of a simulation

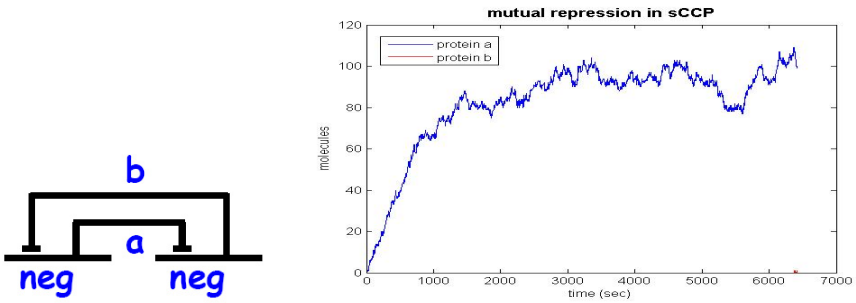


Fig. 5. Bistable circuit. (**right**) Diagram of gene gates involved. (**left**) Time evolution of the circuit. The negative gates have the same rates, set as follows: basic production rate is 0.1 (k_p in Table 8), degradation rate of proteins is 0.0001, inhibition rate (k_i) is 1 and inhibition delay rate (k_d) is 0.0001. Both proteins have an initial value of zero. This graph is one of the two possible outcomes of this bistable network. In the other the roles of the two proteins are inverted.

of the sCCP program is shown in Figure 6, where the oscillatory behaviour is manifest. In [1] it is shown that the oscillatory behaviour is stable w.r.t. changes in parameters. Interestingly, some models of the repressilator using differential equations do not show this form of stability. More comments on the differences between continuous and discrete models of repressilator can be found in [3].

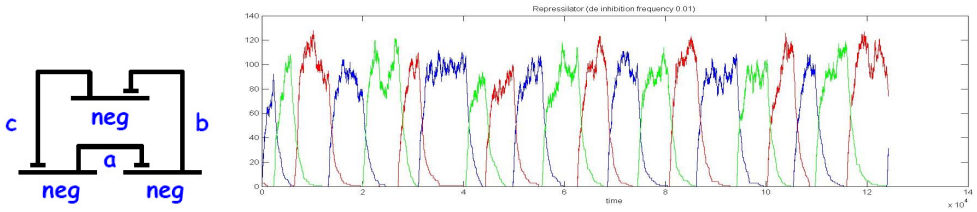


Fig. 6. Repressilator. (**right**) Diagram of gene gates involved. (**left**) Time evolution of the circuit. The negative gates have the same rates, set as follows: basic production rate is 0.1 (k_p in Table 8), degradation rate of proteins is 0.0001, inhibition rate (k_i) is 1 and inhibition delay rate (k_d) is 0.0001. All proteins have an initial value of zero. The time evolution of the repressilator is stable: all simulation traces show this oscillatory behaviour. However, the oscillations among different traces usually are out of phase, and the frequency of the oscillatory pattern varies within the same trace. Remarkably, the average trend of the three proteins shows no oscillation at all, see [3] for further details.

3.3 Modeling the Circadian Clock

In this section we provide as a final example the model of a system containing regulatory mechanism both at the level of genes and at the level of proteins. The system is schematically shown in Figure 7. It is a simplified model of the machinery involved in the circadian rhythm of living beings. In fact, this simple network is present in a wide range of species, from bacteria to humans. The circadian rhythm is a typical mechanism responding to environmental stimuli, in this case the periodic change between light and dark during a day. Basically, it is a clock, expressing a protein periodically with a stable period. This periodic behaviour, to be of some use, must be stable and resistant to both external and internal noise. Here with internal noise we refer to the stochastic fluctuations observable in the concentrations of proteins. The model presented here is taken from [26], a paper focused on the study of the resistance to noise of this system. Interestingly, they showed that the stochastic fluctuations make the oscillatory behaviour even more resistant. Our aim, instead, is that of showing how a system like this can be modeled in an extremely compact way, once we have at disposal the libraries of Sections 3.1 and 3.2.

]

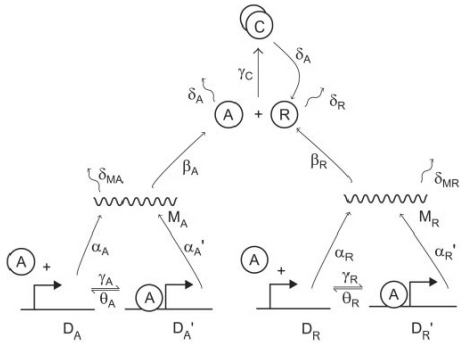


Fig. 7. Biochemical network for the circadian rhythm regulatory system. The figure is taken from [26], like numerical values of rates. Rates are set as follows: $\alpha_A = 50$, $\alpha'_A = 500$, $\alpha_R = 0.01$, $\alpha'_R = 50$, $\beta_A = 50$, $\beta_R = 5$, $\delta_{MA} = 10$, $\delta_{MR} = 0.5$, $\delta_A = 1$, $\delta_R = 0.2$, $\gamma_A = 1$, $\gamma_R = 1$, $\gamma_C = 2$, $\theta_A = 50$, $\theta_R = 100$.

The system is composed by two genes, one expressing an activator protein A, the other producing a repressor protein R. The generation of a protein is depicted here in more detail than in Section 3.2, as the transcription phase of DNA into mRNA and the traduction phase of mRNA into the protein are both modeled explicitly. Protein A is an enhancer for both genes, meaning that it regulates positively their expression. Repressor R, instead, can capture protein A, forming the complex AR and making A inactive. Proteins A and R are degraded at a specific rate (see the caption of Figure 7 for more details about the numerical values), but R can be degraded only if it is not in the complexed form, while A can be degraded in any form. Notice that the regulation

$$
\begin{array}{c}
\text{pos\_gate}(\alpha_A, \alpha'_A, \gamma_A, \theta_A, M_A, A) \parallel \\
\text{pos\_gate}(\alpha_R, \alpha'_R, \gamma_R, \theta_R, M_R, A) \parallel \\
\text{reaction}(\beta_A, [M_A], [A]) \parallel \\
\text{reaction}(\delta_{MA}, [M_A], []) \parallel \\
\text{reaction}(\beta_R, [M_R], [R]) \parallel \\
\text{reaction}(\delta_{MR}, [M_R], []) \parallel \\
\text{reaction}(\gamma_C, [A, R], [AR]) \parallel \\
\text{reaction}(\delta_A, [AR], [R]) \parallel \\
\text{reaction}(\delta_A, [A], []) \parallel \\
\text{reaction}(\delta_R, [R], [])
\end{array}
$$

Table 9. sCCP program for the circadian rhythm regulation system of Figure 7. The agents used have been defined in the previous sections. The first four reaction agents model the translation of mRNA into the coded protein and its degradation. Then we have complex formation, and the degradation of R and A. The pos_gate agent has been redefined as follows, in order to take into account the binding/unbinding of the enhancer: $\text{pos\_gate}(K_p, K_e, K_b, K_u, P, E)$:- $\text{pos\_gate\_off}(K_p, K_e, K_b, K_u, P, E)$; $\text{pos\_gate\_off}(K_p, K_e, K_b, K_u, P, E)$:- $\text{tell}_{K_p}(P = P + 1).\text{pos\_gate\_off}(K_p, K_e, K_b, K_u, P, E) + \text{ask}_{r_{ma}(K_b,E)}(E > 0).\text{pos\_gate\_on}(K_p, K_e, K_b, K_u, P, E)$; $\text{pos\_gate\_on}(K_p, K_e, K_b, K_u, P, E)$:- $\text{tell}_{K_e}(P = P + 1).\text{pos\_gate\_on}(K_p, K_e, K_b, K_u, P, E) + \text{ask}_{K_u}(\text{true}).\text{pos\_gate\_off}(K_p, K_e, K_b, K_u, P, E)$.

activity of A is modeled by an explicit binding to the gene, which remains stimulated until A unbinds. This mechanism is slightly different from the positive gate described in Section 3.2, but the code can be adapted in a straightforward manner (we simply need to define two states for the gene: bound and free, see caption of Table 9).

The code of the sCCP program modeling the system is shown in Table 9. It makes use of the basic agents defined previously, and it is very compact and very easy and quick to write. In Figure 8 (top) we show the evolution of proteins A and R in a numerical simulation performed with the interpreter of the language. As we can see, they oscillate periodically and the length of the period is remarkably stable. Figure 8 (bottom), instead, shows what happens if we replace the bind/unbind model of the gene gate with the "hit and go" code of Section 3.2 (where the enhancer do not bind to the gene, but rather puts it into a stimulated state that makes the gene produce *only the next protein* quicker). The result is dramatic, the periodic behaviour is lost and the system behaves in a chaotic way.

4 Conclusion and future work

In this paper we presented an application of stochastic concurrent constraint programming for modeling of biological systems. We dealt with two main classes of biological networks: biochemical reactions and gene regulation. The main theme is the use of constraints in order to store information about the biological entities into play; this lead straightforwardly to the definition of a general purpose library of processes that can be used in the modeling phase (see Sections 3.1 and 3.2). However, this is only a part of the general picture, as there are more complex classes of biological systems that need to be modeled, like transport networks and membranes. In addition, all these systems are

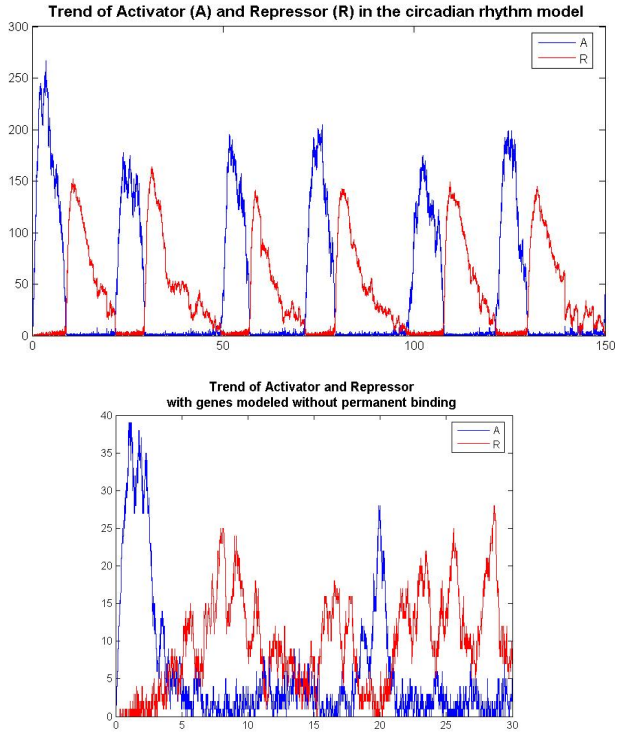


Fig. 8. Time evolution for circadian rhythm model in sCCP. (top) The figure refers to the system described in Figure 7, with parameters described in the caption of the figure. We can see the regularity of the period of the oscillations. (bottom) The graph shows the time evolution for the model where the process governing the gene is the pos_gate described in Table 8. The periodic behaviour, with this simple modification, is irremediably lost.

strongly interconnected, and they must be modeled altogether in order to extract deep information about living beings. We believe that the flexibility of constraints makes sCCP a powerful general purpose language that can be simply programmed, extended with libraries, and used to model all these different classes of systems in a compact way. For instance, different kinds of spatial information, like exact position of molecules or the compartment they are in, can be easily represented using suitable constraints.

Biochemical reactions can be challenging to model, because proteins can form very big complexes that are built incrementally. Therefore, we can find in the cell a huge number of sub-complexes. Usually, these networks are described by biologists with diagrams, like Kohn maps [18], that are very compact, because they represent complexes and sub-complexes implicitly. Modeling these networks explicitly, instead, can be extremely difficult, due to the blow up of the number of different molecules of the system. A calculus having complexation as a primitive operation, the κ-calculus, has been developed in [5]. It offers a compact way to represent formally these diagrams. Constraints can be used to encode this calculus elegantly, by representing complexes implicitly, i.e. as lists of basic constituents.

Another interesting feature that sCCP offers are functional rates. As shown in Section 3.1, they can be used to represent more complex kinetic dynamics, allowing a more compact description of the networks. In this direction, we need to make deeper analysis of the relation between these different kinetics in the context of stochastic simulation, in order to characterize the cases where these different kinetics can be used equivalently. Notice that the use of complex rates can be seen as an operation on the Markov Chain, replacing a subgraph with a smaller one, hiding part of its complexity in the expression of rates. This seems to be a sort of non-trivial lumpability relation [19], though further studies are necessary.

In [3], the authors investigate the expressivity gained by the addition of functional rates to the language. They suggest that there is an increase of power in terms of dynamical behaviours that can be reproduced, after encoding in sCCP a wide class of differential equations. This problem, together with the inverse one of describing sCCP programs by differential equations, is an interesting direction of research, which may lead to an integration of these different techniques, see [16, 3] for further comments.

Finally, we plan to implement a more powerful and fast interpreter for the language, using also all available tricks to increase the speed of stochastic simulations [12]. Moreover, we plan to tackle also the problem of distributing efficiently the stochastic simulations of programs written in sCCP.

References

1. 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.
2. L. Bortolussi. Stochastic concurrent constraint programming. In *Proceedings of 4th International Workshop on Quantitative Aspects of Programming Languages, QAPL 2006*, 2006.
3. L. Bortolussi and A. Policriti. Relating stochastic process algebras and differential equations for biological modeling. *Proceedings of PASTA 2006*, 2006.
4. L. Cardelli and A. Phillips. A correct abstract machine for the stochastic pi-calculus. In *Proceeding of Bioconcur 2004*, 2004.
5. V. Danos and C. Laneve. Formal molecular biology. *Theor. Comput. Sci.*, 325(1):69–110, 2004.
6. F.S. de Boer, A. Di Pierro, and C. Palamidessi. Nondeterminism and infinite computations in constraint programming. *Theoretical Computer Science*, 151(1), 1995.
7. H. De Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.
8. L. Edelstein-Keshet. *Mathematical Models in Biology*. SIAM, 2005.
9. M.B. Elowitz and S. Leibler. A syntetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.
10. Swedish Institute for Computer Science. Sicstus prolog home page.
11. D. Gillespie. The chemical langevin equation. *Journal of Chemical Physics*, 113(1):297–306, 2000.
12. D. Gillespie and L. Petzold. *System Modelling in Cellular Biology*, chapter Numerical Simulation for Biochemical Kinetics. MIT Press, 2006.
13. D.T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. of Computational Physics*, 22, 1976.
14. D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. of Physical Chemistry*, 81(25), 1977.

15. L. Henkin, J.D. Monk, and A. Tarski. *Cylindric Algebras, Part I.* North-Holland, Amsterdam, 1971.
16. J. Hillston. Fluid flow approximation of pepa models. In *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QEST05)*, 2005.
17. C.F. Huang and J.T. Ferrell. Ultrasensitivity in the mitogen-activated protein kinase cascade. *PNAS, Biochemistry*, 151:10078–10083, 1996.
18. K. W. Kohn. Molecular interaction map of the mammalian cell cycle control and dna repair systems. *Molecular Biology of the Cell*, 10:2703–2734, August 1999.
19. J. R. Norris. *Markov Chains.* Cambridge University Press, 1997.
20. C. Priami. Stochastic π-calculus. *The Computer Journal*, 38(6):578–589, 1995.
21. C. Priami and P. Quaglia. Stochastic π-calculus. *Briefings in Bioinformatics*, 5(3):259–269, 2004.
22. 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.
23. V. A. Saraswat. *Concurrent Constraint Programming.* MIT press, 1993.
24. V. A. Saraswat, M. Rinard, and P. Panangaden. Semantics foundations of concurrent constraint programming. In *Proceedings of POPL*, 1991.
25. 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.
26. J. M. G. Vilar, H. Yuan Kueh, N. Barkai, and S. Leibler. Mechanisms of noise resistance in genetic oscillators. *PNAS*, 99(9):5991, 2002.