A Type System for a Stochastic CLS*

Mariangiola Dezani-Ciancaglini
Dipartimento di Informatica, Università di Torino
dezani@di.unito.it

Paola Giannini
Dipartimento di Informatica, Università del Piemonte Orientale
giannini@mfn.unipmn.it

Angelo Troina
Dipartimento di Informatica, Università di Torino
troina@di.unito.it

The Stochastic Calculus of Looping Sequences is suitable to describe the evolution of microbiological systems, taking into account the speed of the described activities. We propose a type system for this calculus that models how the presence of positive and negative catalysers can modify these speeds. We claim that types are the right abstraction in order to represent the interaction between elements without specifying exactly the element positions. Our claim is supported through an example modelling the lactose operon.

1 Introduction

The Calculus of Looping Sequences (CLS for short) [4, 5, 19], is a formalism for describing biological systems and their evolution. CLS is based on term rewriting, given a set of predefined rules modelling the activities one would like to describe. The model has been extended with several features, such as a commutative parallel composition operator, and some semantic means, such as bisimulations [5, 7], which are common in process calculi. This permits to combine the simplicity of notation of rewrite systems with the advantage of a form of compositionality. A Stochastic version of CLS (SCLS for short) is proposed in [6]. Rates are associated with rewrite rules in order to model the speed of the described activities. Therefore, transitions derived in SCLS are driven by a rate that models the parameter of an exponential distribution and characterizes the stochastic behaviour of the transition. The choice of the next rule to be applied and of the time of its application is based on the classical Gillespie's algorithm [15].

Defining a stochastic semantics for CLS requires a correct enumeration of all the possible and distinct ways to apply each rewrite rule within a term. A single pattern may have several, though isomorphic, matches within a CLS term. In this paper, we simplify the counting mechanism used in [6] by imposing some restrictions on the patterns modelling the rewrite rules. Each rewrite rule states explicitly the types of the elements whose occurrence are able to speed-up or slow-down a reaction. The occurrences of the elements of these types are then processed by a rate function (instead of a rate constant) which is used to compute the actual rate of a transition. We show how we can define patterns in our typed stochastic framework to model some common biological activities, and, in particular, we underline the possibility to combine the modelling of positive and negative catalysers within a single rule by reproducing a general case of osmosis.

Finally, as a complete modelling application, we show the expressiveness of our formalism by describing the lactose operon in *Escherichia Coli*.

*This work was partly funded by the project BioBIT of the Regione Piemonte.

This is a preliminary version of a paper
that will appear in Electronic Proceedings
in Theoretical Computer Science.

Summary The remainder of this paper is organized as follows. In Section 2 we formally recall the Calculus of Looping Sequence. In Section 3 we introduce our typed stochastic extension and we give some guidelines for the modelling of biological systems. In Sections 4 we use our framework to model the lactose operon of *Escherichia Coli*. Finally, in Section 5 we draw our conclusions and we present some related work.

2 The Calculus of Looping Sequences

In this section we recall the Calculus of Looping Sequences (CLS). CLS is essentially based on term rewriting, hence a CLS model consists of a term and a set of rewrite rules. The term is intended to represent the structure of the modelled system, and the rewrite rules to represent the events that may cause the system to evolve.

We start with defining the syntax of terms. We assume a possibly infinite alphabet $\mathscr{E}$ of symbols ranged over by $a, b, c, \ldots$.

Definition 2.1 (Terms) Terms T *and* sequences S *of* CLS *are given by the following grammar:*

$$\begin{array}{lcl} T & ::= & S \mid (S)^L \rfloor T \mid T \mid T \\ S & ::= & \varepsilon \mid a \mid S \cdot S \end{array}$$

where a is a generic element of $\mathscr{E}$, and ε represents the empty sequence. We denote with $\mathscr{T}$ the infinite set of terms, and with $\mathscr{S}$ the infinite set of sequences.

In CLS we have a sequencing operator $\_\cdot\_$, a looping operator $(\_)^L$, a parallel composition operator $\_|\_$ and a containment operator $\_\rfloor\_$. Sequencing can be used to concatenate elements of the alphabet $\mathscr{E}$. The empty sequence ε denotes the concatenation of zero symbols. A term can be either a sequence or a looping sequence (that is the application of the looping operator to a sequence) containing another term, or the parallel composition of two terms. By definition, looping and containment are always applied together, hence we can consider them as a single binary operator $(\_)^L \rfloor \_$ which applies to one sequence and one term.

The biological interpretation of the operators is the following: the main entities which occur in cells are DNA and RNA strands, proteins, membranes, and other macro–molecules. DNA strands (and similarly RNA strands) are sequences of nucleic acids, but they can be seen also at a higher level of abstraction as sequences of genes. Proteins are sequence of amino acids which usually have a very complex three–dimensional structure. In a protein there are usually (relatively) few subsequences, called domains, which actually are able to interact with other entities by means of chemical reactions. CLS sequences can model DNA/RNA strands and proteins by describing each gene or each domain with a symbol of the alphabet. Membranes are closed surfaces, often interspersed with proteins, which may contain something. A closed surface can be modelled by a looping sequence. The elements (or the subsequences) of the looping sequence may represent the proteins on the membrane, and by the containment operator it is possible to specify the content of the membrane. Other macro–molecules can be modelled as single alphabet symbols, or as short sequences. Finally, juxtaposition of entities can be described by the parallel composition of their representations.

Brackets can be used to indicate the order of application of the operators, and we assume $(\_)^L \rfloor \_$ to have precedence over $\_|\_$. In Figure 1 we show some examples of CLS terms and their visual representation, using $(S)^L$ as a short-cut for $(S)^L \rfloor \varepsilon$.

In CLS we may have syntactically different terms representing the same structure. We introduce a structural congruence relation to identify such terms.

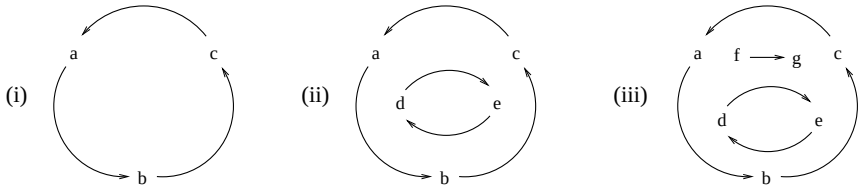

Figure 1: (i) represents $(a\cdot b\cdot c)^L$; (ii) represents $(a\cdot b\cdot c)^L \rfloor (d\cdot e)^L$; (iii) represents $(a\cdot b\cdot c)^L \rfloor ((d\cdot e)^L \mid f\cdot g)$.

Definition 2.2 (Structural Congruence) *The structural congruence relations $\equiv_S$ and $\equiv_T$ are the least congruence relations on sequences and on terms, respectively, satisfying the following rules:*

$$S_1\cdot(S_2\cdot S_3)\equiv_S (S_1\cdot S_2)\cdot S_3 \qquad S\cdot\varepsilon\equiv_S \varepsilon\cdot S\equiv_S S$$
$$S_1\equiv_S S_2 \text{ implies } S_1\equiv_T S_2 \text{ and } (S_1)^L\rfloor T\equiv_T (S_2)^L\rfloor T$$
$$T_1\mid T_2\equiv_T T_2\mid T_1 \qquad T_1\mid(T_2\mid T_3)\equiv_T (T_1\mid T_2)\mid T_3 \qquad T\mid\varepsilon\equiv_T T$$
$$(\varepsilon)^L\rfloor\varepsilon\equiv_T\varepsilon \qquad (S_1\cdot S_2)^L\rfloor T\equiv_T (S_2\cdot S_1)^L\rfloor T$$

Rules of the structural congruence state the associativity of $\cdot$ and $\mid$, the commutativity of the latter and the neutral role of ε. Moreover, axiom $(S_1\cdot S_2)^L\rfloor T\equiv_T (S_2\cdot S_1)^L\rfloor T$ says that looping sequences can rotate. In the following, for simplicity, we will use $\equiv$ in place of $\equiv_T$.

Rewrite rules will be defined essentially as pairs of terms, with the first term describing the portion of the system in which the event modelled by the rule may occur, and the second term describing how that portion of the system changes when the event occurs. In the terms of a rewrite rule we allow the use of variables. As a consequence, a rule will be applicable to all terms which can be obtained by properly instantiating its variables. Variables can be of three kinds: two of these are associated with the two different syntactic categories of terms and sequences, and one is associated with single alphabet elements. We assume a set of term variables $\mathscr{TV}$ ranged over by $X,Y,Z,\ldots$, a set of sequence variables $\mathscr{SV}$ ranged over by $\widetilde{x},\widetilde{y},\widetilde{z},\ldots$, and a set of element variables $\mathscr{X}$ ranged over by $x,y,z,\ldots$. All these sets are possibly infinite and pairwise disjoint. We denote by $\mathscr{V}$ the set of all variables, $\mathscr{V}=\mathscr{TV}\cup\mathscr{SV}\cup\mathscr{X}$, and with χ a generic variable of $\mathscr{V}$. Hence, a pattern is a term that may include variables.

Definition 2.3 (Patterns) Patterns *P and* sequence patterns *SP of* CLS *are given by the following grammar:*

$$\begin{array}{lcl} P & ::= & SP \mid (SP)^L\rfloor P \mid P\mid P \mid X \\ SP & ::= & \varepsilon \mid a \mid SP\cdot SP \mid \widetilde{x} \mid x \end{array}$$

where a is a generic element of $\mathscr{E}$, and $X,\widetilde{x}$ and x are generic elements of $\mathscr{TV},\mathscr{SV}$ and $\mathscr{X}$, respectively. We denote with $\mathscr{P}$ the infinite set of patterns.

We assume the structural congruence relation to be trivially extended to patterns. An *instantiation* is a partial function $\sigma:\mathscr{V}\to\mathscr{T}$. An instantiation must preserve the type of variables, thus for $X\in\mathscr{TV},\widetilde{x}\in\mathscr{SV}$ and $x\in\mathscr{X}$ we have $\sigma(X)\in\mathscr{T},\sigma(\widetilde{x})\in\mathscr{S}$ and $\sigma(x)\in\mathscr{E}$, respectively. Given $P\in\mathscr{P}$, with $P\sigma$ we denote the term obtained by replacing each occurrence of each variable $\chi\in\mathscr{V}$ appearing in P with the corresponding term $\sigma(\chi)$. With Σ we denote the set of all the possible instantiations and, given $P\in\mathscr{P}$, with $Var(P)$ we denote the set of variables appearing in P. Now we define rewrite rules.

Definition 2.4 (Rewrite Rules) *A rewrite rule is a pair of patterns (P_1,P_2), denoted with $P_1\mapsto P_2$, where $P_1,P_2\in\mathscr{P}$, $P_1\not\equiv\varepsilon$ and such that $Var(P_2)\subseteq Var(P_1)$.*

A rewrite rule $P_1 \mapsto P_2$ states that a term $P_1\sigma$, obtained by instantiating variables in P_1 by some instantiation function σ, can be transformed into the term $P_2\sigma$. We define the semantics of CLS as a transition system, in which states correspond to terms, and transitions correspond to rule applications.

We define the semantics of CLS by resorting to the notion of contexts.

Definition 2.5 (Contexts) Contexts *C are defined as:*

$$C ::= \Box \quad | \quad C \,|\, T \quad | \quad T \,|\, C \quad | \quad (S)^L \rfloor C$$

where $T \in \mathscr{T}$ and $S \in \mathscr{S}$. The context $\Box$ is called the empty context. *We denote with $\mathscr{C}$ the infinite set of contexts.*

By definition, every context contains a single hole $\Box$. Let us assume $C \in \mathscr{C}$, with $C[T]$ we denote the term obtained by replacing $\Box$ with T in C. The structural equivalence is extended to contexts in the natural way (i.e. by considering $\Box$ as a new and unique symbol of the alphabet $\mathscr{E}$).

Rewrite rules can be applied to terms only if they occur in a legal context. Note that the general form of rewrite rules does not permit to have sequences as contexts. A rewrite rule introducing a parallel composition on the right hand side (as $a \mapsto b \,|\, c$) applied to an element of a sequence (e.g., $m \cdot a \cdot m$) would result into a syntactically incorrect term (in this case $m \cdot (b \,|\, c) \cdot m$). To modify a sequence, a pattern representing the whole sequence must appear in the rule. For example, rule $a \cdot \widetilde{x} \mapsto a \,|\, \widetilde{x}$ can be applied to any sequence starting with element a, and, hence, the term $a \cdot b$ can be rewritten as $a \,|\, b$, and the term $a \cdot b \cdot c$ can be rewritten as $a \,|\, b \cdot c$.

The semantics of CLS is defined as follows.

Definition 2.6 (Semantics) *Given a finite set of rewrite rules $\mathscr{R}$, the* semantics *of CLS is the least relation closed with respect to $\equiv$ and satisfying the following rule:*

$$\frac{P_1 \mapsto P_2 \in \mathscr{R} \quad \sigma \in \Sigma \quad P_1\sigma \not\equiv \varepsilon \quad C \in \mathscr{C}}{C[P_1\sigma] \to C[P_2\sigma]}$$

As usual we denote with $\to^*$ the reflexive and transitive closure of $\to$.

Given a set of rewrite rules $\mathscr{R}$, the behaviour of a term T is the tree of terms to which T may reduce. Thus, a *model* in CLS is given by a term describing the initial state of the system and by a set of rewrite rules describing all the events that may occur.

3 Typed Stochastic CLS

In this section we show how types are used to enhance the expressivity of CLS. In particular, we use types to focus on quantitative aspects of CLS, by showing how to model the speeds of the biological activities.

We classify elements in $\mathscr{E}$ with types. Intuitively, given a molecule represented by an element a in $\mathscr{E}$, we associate a type to it which specifies the kind of the molecule. For an element a, we distinguish between occurrences of a single a in parallel with other terms, for which we use a *basic type* t, and occurrences of a within a sequence, for which we use a *sequence type* $\widetilde{t}$. So, types specify the kind of elements and their positioning. In the following, with type, we mean either a basic type or a sequence type and we use t to range over both basic and sequence types. The metavariable τ ranges over multi-sets of types. By $t \in_n \tau$ we denote that type t occurs n times in τ, and $\uplus$ is the union on multisets.

Let Γ be a type assignment such that for $a \in \mathscr{E}$, if $\Gamma(a) = t$, then t and $\widetilde{t}$ are the types for a. The type of a term (or sequence) is the multiset of types (or sequence types) of its outermost component. This is formalised in the following definition of *type* of a term and *stype* of a sequence.

Definition 3.1 (Mappings *type* and *stype*) *The mappings type and stype are defined by induction on terms and sequences as follows:*

- – $type((S)^L \rfloor T) = stype(S)$
  – $type(T_1 \,|\, T_2) = type(T_1) \uplus type(T_2)$
  – $type(S_1 \cdot S_2) = stype(S_1 \cdot S_2)$
  – $type(a) = \{\Gamma(a)\}$
- – $stype(S_1 \cdot S_2) = stype(S_1) \uplus stype(S_2)$
  – $stype(a) = \{\widetilde{\Gamma(a)}\}$

For example if $\Gamma(a) = t_a$, $\Gamma(b) = t_b$ and $\Gamma(c) = t_c$ we have $type(a\,|\,a\,|\,c) = \{t_a, t_a, t_c\}$, $type(b \cdot c \cdot c) = \{\widetilde{t_b}, \widetilde{t_c}, \widetilde{t_c}\}$, $type(a\,|\,a\,|\,c\,|\,(b \cdot c \cdot c)^L \rfloor a) = \{t_a, t_a, t_c, \widetilde{t_b}, \widetilde{t_c}, \widetilde{t_c}\}$ and $type((b \cdot c \cdot c)^L \rfloor a\,|\,a\,|\,a\,|\,c) = \{\widetilde{t_b}, \widetilde{t_c}, \widetilde{t_c}\}$.

Term transitions are labelled with a *rate* r, a real number, $T \xrightarrow{r} T'$, modelling the speed of the transition. The number r depends on the types and multiplicity of the elements interacting.

To compute the rate of transitions we associate to each rule, $P \mapsto P'$ the information which is relevant to the application of the rule. This is expressed by giving:

- for each variable χ in the pattern P, the types of the elements that influence the speed of the application of the rule,
- a weighting function that combines the multiplicity of types on single variables, producing the final rate.

We provide this information as follows. Given a pattern P, let $V(P) = \langle \chi_1, \ldots, \chi_m \rangle$ be the list of (sequence, term, and element) variables of P in left-to-right order of occurrence.

- To each χ_i we associate a list $\Pi_i = \langle t_1^{(i)}, \ldots, t_{p_i}^{(i)} \rangle$ of types,
- Moreover, let $\phi : \mathbf{N}^q \to \mathbf{R}$ be a function from a list of $q = \sum_{1 \le i \le m} p_i$ integers to a real.

The rewrite rules of our *Typed Stochastic* CLS (TSCLS for short) are of the shape

$$P \xrightarrow[\phi]{\overline{\Pi}} P'$$

where $\overline{\Pi} = \langle \Pi_1, \ldots, \Pi_m \rangle$.

For example as discussed in the following subsection the transformation of the element a into the element b inhibited by the presence of the element c can be described by the rule

$$a\,|\,X \xrightarrow[\phi]{\langle\langle t_a, t_c \rangle\rangle} b\,|\,X \tag{1}$$

where $\phi = \lambda n_1 n_2 . \frac{(n_1+1)\times k}{\text{if } n_2 = 0 \text{ then } 1 \text{ else } n_2 \times k'}$, and k, k' are the kinetic constant of the state change of a into b and the deceleration due to the presence of one inhibitor c, respectively.

We consider local interactions, that is interactions between elements in the same compartment. When applying a rule, to take into account a whole compartment, we redefine the notion of context by enforcing the property that the hole of a context embraces a whole compartment as follows:

Definition 3.2 (Stochastic Contexts) Stochastic Contexts C *are defined as:*

$$C ::= \Box \quad | \quad T \,|\, (S)^L \rfloor C$$

where $T \in \mathscr{T}$ *and* $S \in \mathscr{S}$*. We denote with* $\mathscr{SC}$ *the infinite set of stochastic contexts.*

We can now define the typed semantics.

Definition 3.3 (Typed Stochastic Semantics) *Given a finite set of rewrite rules* $\mathscr{R}$*, the* semantics *of* TSCLS *is the least relation closed with respect to* $\equiv$ *and satisfying the following rule:*

$$\frac{\begin{array}{c} P_1 \xrightarrow[\phi]{\overline{\Pi}} P_2 \in \mathscr{R} \quad \sigma \in \Sigma \quad P_1\sigma \not\equiv \varepsilon \quad C \in \mathscr{SC} \\ type(\sigma(\chi_i)) = \tau_i \qquad t_j^{(i)} \in_{n_j^{(i)}} \tau_i \quad (1 \le j \le p_i) \quad (1 \le i \le m) \\ r = \phi(n_1^{(1)}, \ldots, n_{p_1}^{(1)}, \cdots, n_1^{(m)}, \ldots, n_{p_m}^{(m)}) \end{array}}{C[P_1\sigma] \xrightarrow{r} C[P_2\sigma]}$$

For example, applying rule (1) with the empty context to the term $a\,|\,a\,|\,c$ we have:

$$a\,|\,a\,|\,c \xrightarrow{\frac{2\times k}{1\times k'}} a\,|\,b\,|\,c \xrightarrow{\frac{1\times k}{1\times k'}} b\,|\,b\,|\,c$$

and to the term $a\,|\,a\,|\,c\,|\,(b\cdot c\cdot c)^L \rfloor a$ we have:

$$a\,|\,a\,|\,c\,|\,(b\cdot c\cdot c)^L \rfloor a \xrightarrow{\frac{2\times k}{1\times k'}} a\,|\,b\,|\,c\,|\,(b\cdot c\cdot c)^L \rfloor a \xrightarrow{\frac{1\times k}{1\times k'}} b\,|\,b\,|\,c\,|\,(b\cdot c\cdot c)^L \rfloor a$$

Applying (1) with the context $\varepsilon\,|\,(b\cdot c\cdot c)^L \rfloor \Box$ to the term $(b\cdot c\cdot c)^L \rfloor a\,|\,a\,|\,a\,|\,c$ we get:

$$(b\cdot c\cdot c)^L \rfloor a\,|\,a\,|\,a\,|\,c \xrightarrow{\frac{3\times k}{1\times k'}} (b\cdot c\cdot c)^L \rfloor a\,|\,a\,|\,b\,|\,c \xrightarrow{\frac{2\times k}{1\times k'}} (b\cdot c\cdot c)^L \rfloor a\,|\,b\,|\,b\,|\,c \xrightarrow{\frac{1\times k}{1\times k'}} (b\cdot c\cdot c)^L \rfloor b\,|\,b\,|\,b\,|\,c$$

Note that we cannot use Definition 2.5 for contexts, since we would not count correctly the numbers of elements which influence the speed of transformations. For example, again rule (1) applied to the term $a\,|\,a\,|\,c$ with the context $\Box\,|\,a\,|\,c$ would produce the wrong transition:

$$a\,|\,a\,|\,c \xrightarrow{k} a\,|\,b\,|\,c.$$

Given the Continuous Time Markov Chain (CTMC) obtained from the transition system resulting from our typed stochastic semantics, we can follow a standard simulation procedure. Roughly speaking, the algorithm starts from the initial term (representing a state of the CTMC) and performs a sequence of steps by moving from state to state. At each step a global clock variable (initially set to zero) is incremented by a random quantity which is exponentially distributed with the exit rate of the current state as parameter, and the next state is randomly chosen with a probability proportional to the rates of the exit transitions.

The *race condition* described above implements the fact that, on the lines of Gillespie's algorithm [15], when different reactions are competing with different rates, the ones which are not chosen should restart the competition at the following step.

3.1 Modelling Guidelines

In the remain of this section we will put at work the TSCLS calculus in order to model biomolecular events of interest.

- The application rate in the case of the *change of state of an elementary object* is proportional to the number of objects which are present. For this reason if t_a is the type of the object a and k is the kinetic constant of the state change of a into b we can describe this chemical reaction by the following rewrite rule:

$$a \,|\, X \xrightarrow[\phi]{\langle\langle t_a \rangle\rangle} b \,|\, X$$

  where $\phi = \lambda n.(n+1) \times k$. Using this rule we get for example:

$$(m)^L \rfloor (a \,|\, a \,|\, a) \xrightarrow{3k} (m)^L \rfloor (b \,|\, a \,|\, a)$$

$$(m)^L \rfloor (a \,|\, a \cdot a) \xrightarrow{k} (m)^L \rfloor (b \,|\, a \cdot a)$$

  where m is any membrane.

- In the process of *complexation*, two elementary objects in the same compartment are combined to produce a new object. The application rate is then proportional to the product of the numbers of occurrences of the two objects. Assuming that t_a and t_b are the types of a and b we get:

$$a \,|\, b \,|\, X \xrightarrow[\phi]{\langle\langle t_a, t_b \rangle\rangle} c \,|\, X$$

  where $\phi = \lambda n_1 n_2.(n_1+1) \times (n_2+1) \times k$ and k is the kinetic constant of the modelled chemical reaction.

  Using the same conventions a similar and simpler rule describes *decomplexation*:

$$c \,|\, X \xrightarrow[\phi]{\langle\langle t_c \rangle\rangle} a \,|\, b \,|\, X$$

  where $\phi = \lambda n.(n+1) \times k$.

- Another phenomenon which can be easily rendered in our formalism is the *osmosis* regulating the quantity of water inside and outside a cell for a dilute solution of non-dissociating substances. In fact in this case according to [26] the total flow is $L_p \frac{S}{V} \Delta\psi_w$, where L_p is the hydraulic conductivity constant, which depends on the semi-permeability properties of the membrane, S is the surface of the cell, V is the volume of the cell, $\Delta\psi_w = \psi_{w(ext)} - \psi_{w(int)}$ is the difference between the water potentials outside and inside the cell. The water potential for non-dissociating substances is the sum of the solute potential $\psi_s = -\mathsf{R}\mathsf{T}c_s$ (where R is the gas constant, T is the absolute temperature and c_s is the solute concentration) and the pressure potential ψ_p (which depends on the elastic properties of the membrane and on the cell wall). We can therefore consider the rate of flow of water proportional (via a constant k) to $\frac{S}{V}(c_{s(ext)} - c_{s(int)})$, where the sign of this real gives the direction of the flow. The membrane crossing of the element a according to the concentration of the elements b inside and outside the cell is given by the pairs of rules:

$$(\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y \xrightarrow[\phi]{\langle\langle\rangle, \langle t_a, t_b \rangle, \langle t_a, t_b \rangle\rangle} (\widetilde{x})^L \rfloor X \,|\, a \,|\, Y$$

$$(\widetilde{x})^L \rfloor X \,|\, a \,|\, Y \xrightarrow[\phi']{\langle\langle\rangle, \langle t_a, t_b \rangle, \langle t_a, t_b \rangle\rangle} (\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y$$

where $\begin{array}{l}\phi = \lambda n_1 n_2 n_3 n_4.\frac{S}{V} \times (\frac{n_2}{(n_1+1)V_a+n_2V_b} - \frac{n_4}{(n_3+1)V_a+n_4V_b}) \times k \\ \phi' = \lambda n_1 n_2 n_3 n_4.\frac{S}{V} \times (\frac{n_4}{(n_3+1)V_a+n_4V_b} - \frac{n_2}{(n_1+1)V_a+n_2V_b}) \times k\end{array}$ and V_a, V_b are the volumes of the elements a and b, respectively.

The *positive catalysis* of osmosis by the presence of elements c on the membrane is rendered by:

$$(\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y \xrightarrow[\phi]{\langle\langle \widetilde{t_c}\rangle, \langle t_a, t_b\rangle, \langle t_a, t_b\rangle\rangle} (\widetilde{x})^L \rfloor X \,|\, a \,|\, Y$$
$$(\widetilde{x})^L \rfloor X \,|\, a \,|\, Y \xrightarrow[\phi']{\langle\langle \widetilde{t_c}\rangle, \langle t_a, t_b\rangle, \langle t_a, t_b\rangle\rangle} (\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y$$

where $\begin{array}{l}\phi = \lambda n_1 n_2 n_3 n_4 n_5.(n_1 \times k_c + 1) \times \frac{S}{V} \times (\frac{n_3}{(n_2+1)V_a+n_3V_b} - \frac{n_5}{(n_4+1)V_a+n_5V_b}) \times k \\ \phi' = \lambda n_1 n_2 n_3 n_4 n_5.(n_1 \times k_c + 1) \times \frac{S}{V} \times (\frac{n_5}{(n_4+1)V_a+n_5V_b} - \frac{n_3}{(n_2+1)V_a+n_3V_b}) \times k\end{array}$ and k_c is the acceleration due to the presence of one element c.

Similarly the *inhibition* of osmosis by the presence of elements c on the membrane is rendered by:

$$(\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y \xrightarrow[\phi]{\langle\langle \widetilde{t_c}\rangle, \langle t_a, t_b\rangle, \langle t_a, t_b\rangle\rangle} (\widetilde{x})^L \rfloor X \,|\, a \,|\, Y$$
$$(\widetilde{x})^L \rfloor X \,|\, a \,|\, Y \xrightarrow[\phi']{\langle\langle \widetilde{t_c}\rangle, \langle t_a, t_b\rangle, \langle t_a, t_b\rangle\rangle} (\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y$$

where $\begin{array}{l}\phi = \lambda n_1 n_2 n_3 n_4 n_5.\frac{1}{\text{if } n_1=0 \text{ then 1 else } n_1 \times k_c} \times \frac{S}{V} \times (\frac{n_3}{(n_2+1)V_a+n_3V_b} - \frac{n_5}{(n_4+1)V_a+n_5V_b}) \times k \\ \phi' = \lambda n_1 n_2 n_3 n_4 n_5.\frac{1}{\text{if } n_1=0 \text{ then 1 else } n_1 \times k_c} \times \frac{S}{V} \times (\frac{n_5}{(n_4+1)V_a+n_5V_b} - \frac{n_3}{(n_2+1)V_a+n_3V_b}) \times k\end{array}$ and k_c is the deceleration due to the presence of one element c.

- If the rule
$$P_1 \xrightarrow[\phi]{\overline{\Pi}} P_2$$
describes an event, in order to express that this event is *positively catalysed* by an element c we can modify the rewrite rule as follows.

  If $P_1 \equiv P_1' \,|\, X$, the type list of X is Π_X and the weighting function ϕ is $\lambda \overline{n n_X}.e$, where $\overline{n}$ takes into account the elements occurring in P_1' and $\overline{n_X}$ takes into account the elements occurring in X, we define:

  - Π_X' as the list whose head is t_c and whose tail is Π_X,
  - $\phi' = \lambda \overline{n} n_c \overline{n_X}.e \times (n_c \times k + 1)$,

  where k is the acceleration due to the presence of one positive catalyser c. The new rule is obtained from the old one by replacing Π_X' and ϕ' to Π_X and ϕ, respectively.

  Otherwise if $P_1 \not\equiv P_1' \,|\, X$, the new rule is:
$$P_1 \,|\, X \xrightarrow[\phi']{\overline{\Pi} \frown \langle\langle t_c\rangle\rangle} P_2 \,|\, X$$
where $\frown$ represents list concatenation and if $\phi = \lambda \overline{n}.e$, then $\phi' = \lambda \overline{n} n_c.e \times (n_c \times k + 1)$.

  Similarly we can represent the effect of an *inhibitor* just replacing the inserted multiplications by divisions. We can also represent in one rule both positive and negative catalysers. For example to add the effect of a positive catalyser c and an inhibitor d to the rule $P_1 \xrightarrow[\phi]{\overline{\Pi}} P_2$ if $P_1 \equiv P_1' \,|\, X$ and Π_X, ϕ are as above we define:

– $\Pi'_X = \langle t_c, t_d \rangle^\frown \Pi_X$,
– $\phi' = \lambda \overline{n} n_c n_d \overline{n_X}.e \times \frac{n_c \times k+1}{\text{if } n_d=0 \text{ then 1 else } n_d \times k'}$,

where k is the acceleration due to the presence of one positive catalyser c and k' is the deceleration due to the presence of one inhibitor d.

Otherwise if $P_1 \not\equiv P'_1 \,|\, X$, the new rule is:

$$P_1 \,|\, X \xrightarrow[\phi']{\overline{\Pi}^\frown \langle\langle t_c, t_d \rangle\rangle} P_2 \,|\, X$$

where if $\phi = \lambda \overline{n}.e$, then $\phi' = \lambda \overline{n} n_c n_d.e \frac{n_c \times k+1}{\text{if } n_d=0 \text{ then 1 else } n_d \times k'}$.

Looking at the previous examples, we claim that our formalism enlightens better than other formalisms the duality between the roles of positive and negative catalysers.

4 An Application: The Lactose Operon

To show that our framework can be easily used to model and simulate cellular pathways, we give a model of the well-known regulation process of the lactose operon in *Escherichia coli*.

E. coli is a bacterium often present in the intestine of many animals. It is one of the most completely studied of all living things and it is a favourite organism for genetic engineering. Cultures of E. coli can be made to produce unlimited quantities of the product of an introduced gene. As most bacteria, E.coli is often exposed to a constantly changing physical and chemical environment, and reacts to changes in its environment through changes in the kinds of enzymes it produces. In order to save energy, bacteria do not synthesize degradative enzymes unless the substrates for these enzymes are present in the environment. For example, E. coli does not synthesize the enzymes that degrade lactose unless lactose is in the environment. This result is obtained by controlling the transcription of some genes into the corresponding enzymes.

Two enzymes are involved in lactose degradation: the *lactose permease*, which is incorporated in the membrane of the bacterium and actively transports the sugar into the cell, and the *beta galactosidase*, which splits lactose into glucose and galactose. The bacterium produces also the *transacetylase* enzyme, whose role in the lactose degradation is marginal.

The sequence of genes in the DNA of E. coli which produces the described enzymes, is known as the *lactose operon*.

The first three genes of the operon (i, p and o) regulate the production of the enzymes, and the last three (z, y and a), called *structural genes*, are transcribed (when allowed) into the mRNA for beta galactosidase, lactose permease and transacetylase, respectively.

The regulation process is as follows (see Figure 2): gene i encodes the *lac Repressor*, which, in the absence of lactose, binds to gene o (the *operator*). Transcription of structural genes into mRNA is performed by the RNA polymerase enzyme, which usually binds to gene p (the *promoter*) and scans the operon from left to right by transcribing the three structural genes z, y and a into a single mRNA fragment. When the lac Repressor is bound to gene o, it becomes an obstacle for the RNA polymerase, and the transcription of the structural genes is not performed. On the other hand, when lactose is present inside the bacterium, it binds to the Repressor and this cannot stop anymore the activity of the RNA polymerase. In this case the transcription is performed and the three enzymes for lactose degradation are synthesized.

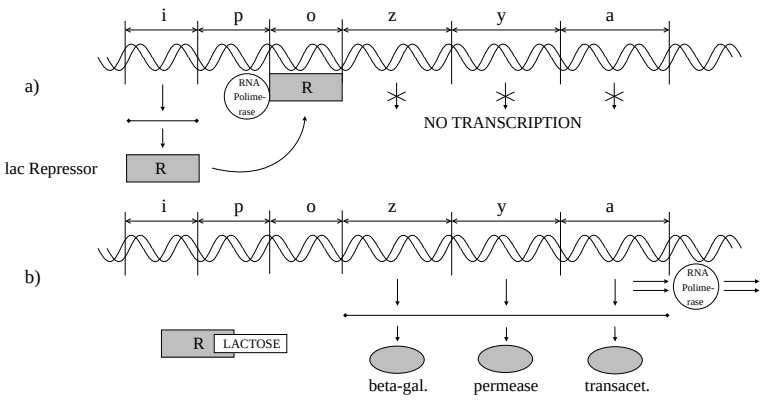

Figure 2: The regulation process in the Lac Operon.

4.1 Typed Stochastic CLS Model

A detailed mathematical model of the regulation process can be found in [28]. It includes information on the influence of lactose degradation on the growth of the bacterium.

We give a TSCLS model of the gene regulation process, with stochastic rates taken from [27]. We model the membrane of the bacterium as the looping sequence $(m)^L$, where the alphabet symbol m generically denotes the whole membrane surface in normal conditions. Moreover, we model the lactose operon as the sequence $lacI \cdot lacP \cdot lacO \cdot lacZ \cdot lacY \cdot lacA$ ($lacI$–A for short), in which each symbol corresponds to a gene. We replace $lacO$ with RO in the sequence when the lac Repressor is bound to gene o, and $lacP$ with PP when the RNA polymerase is bound to gene p. When the lac Repressor and the RNA polymerase are unbound, they are modelled by the symbols $repr$ and $polym$, respectively. We model the mRNA of the lac Repressor as the symbol $Irna$, a molecule of lactose as the symbol $LACT$, and beta galactosidase, lactose permease and transacetylase enzymes as symbols $betagal$, $perm$ and $transac$, respectively. Finally, since the three structural genes are transcribed into a single mRNA fragment, we model such mRNA as a single symbol Rna.

The transcription of the DNA, the binding of the lac Repressor to gene o, and the interaction between lactose and the lac Repressor are modelled by the following set of stochastic typed rewrite rules:

$$lacI\text{–}A \,|\, X \xrightarrow[\phi]{\langle\langle\rangle\rangle} lacI\text{–}A \,|\, Irna \,|\, X \tag{R1}$$

where $\phi = 0.02$.

$$Irna \,|\, X \xrightarrow[\phi]{\langle\langle t\rangle\rangle} Irna \,|\, repr \,|\, X \tag{R2}$$

where t is the type of $Irna$ and $\phi = \lambda n.(n+1) \times 0.1$.

$$lacI\text{–}A \,|\, polym \,|\, X \xrightarrow[\phi]{\langle\langle t\rangle\rangle} lacI \cdot PP \cdot lacO \cdot lacZ \cdot lacY \cdot lacA \,|\, X \tag{R3}$$

where t is the type of $polym$ and $\phi = \lambda n.(n+1) \times 0.1$.

$$lacI \cdot PP \cdot lacO \cdot lacZ \cdot lacY \cdot lacA \,|\, X \xrightarrow[\phi]{\langle\langle\rangle\rangle} lacI\text{–}A \,|\, polym \,|\, X \tag{R4}$$

where $\phi = 0.01$.

$$lacI \cdot PP \cdot lacO \cdot lacZ \cdot lacY \cdot lacA \,|\, X \xrightarrow[\phi]{\langle\langle\rangle\rangle} lacI\text{–}A \,|\, polym \,|\, Rna \,|\, X \tag{R5}$$

where $\phi = 20$.

$$Rna \,|\, X \xrightarrow[\phi]{\langle\langle t\rangle\rangle} Rna \,|\, betagal \,|\, perm \,|\, transac \,|\, X \tag{R6}$$

where t is the type of *Rna* and $\phi = \lambda n.(n+1) \times 0.1$.

$$lacI\text{–}A \,|\, repr \,|\, X \xrightarrow[\phi]{\langle\langle t\rangle\rangle} lacI \cdot lacP \cdot RO \cdot lacZ \cdot lacY \cdot lacA \,|\, X \tag{R7}$$

where t is the type of *repr* and $\phi = \lambda n.(n+1) \times 1$.

$$lacI \cdot PP \cdot lacO \cdot lacZ \cdot lacY \cdot lacA \,|\, repr \,|\, X \xrightarrow[\phi]{\langle\langle t\rangle\rangle} lacI \cdot PP \cdot RO \cdot lacZ \cdot lacY \cdot lacA \,|\, X \tag{R8}$$

where t is the type of *repr* and $\phi = \lambda n.(n+1) \times 1$.

$$lacI \cdot lacP \cdot RO \cdot lacZ \cdot lacY \cdot lacA \,|\, X \xrightarrow[\phi]{\langle\langle\rangle\rangle} lacI\text{–}A \,|\, repr \,|\, X \tag{R9}$$

where $\phi = 0.01$.

$$lacI \cdot PP \cdot RO \cdot lacZ \cdot lacY \cdot lacA \,|\, X \xrightarrow[\phi]{\langle\langle\rangle\rangle} lacI \cdot PP \cdot lacO \cdot lacZ \cdot lacY \cdot lacA \,|\, repr \,|\, X \tag{R10}$$

where $\phi = 0.01$.

$$repr \,|\, LACT \,|\, X \xrightarrow[\phi]{\langle\langle t_r, t_l\rangle\rangle} RLACT \,|\, X \tag{R11}$$

where t_r and t_l are the types of *repr* and *LACT* and $\phi = \lambda n_1 n_2.(n_1+1) \times (n_2+1) \times 0.005$.

$$RLACT \,|\, X \xrightarrow[\phi]{\langle\langle t\rangle\rangle} repr \,|\, LACT \,|\, X \tag{R12}$$

where t is the type of *RLACT* and $\phi = \lambda n.(n+1) \times 0.1$.

Rules (R1) and (R2) describe the transcription and translation of gene i into the lac Repressor (assumed for simplicity to be performed without the intervention of the RNA polymerase). Rules (R3) and (R4) describe binding and unbinding of the RNA polymerase to gene p. Rules (R5) and (R6) describe the transcription and translation of the three structural genes. Transcription of such genes can be performed only when the sequence contains *lacO* instead of *RO*, that is when the lac Repressor is not bound to gene o. Rules (R7)-(R10) describe binding and unbinding of the lac Repressor to gene o. Finally, rules (R11) and (R12) describe the binding and unbinding, respectively, of the lactose to the lac Repressor.

The following rules describe the behaviour of the three enzymes for lactose degradation:

$$(\widetilde{x})^L \rfloor (perm \,|\, X) \,|\, Y \xrightarrow[\phi]{\langle\langle\rangle, \langle t\rangle, \langle\rangle\rangle} (perm \cdot \widetilde{x})^L \rfloor X \,|\, Y \tag{R13}$$

where t is the type of *perm* and $\phi = \lambda n.(n+1) \times 0.1$.

$$(\widetilde{x})^L \rfloor X \,|\, LACT \,|\, Y \xrightarrow[\phi]{\langle\langle \widetilde{t_p}\rangle,\langle\rangle,\langle t_l\rangle\rangle} (\widetilde{x})^L \rfloor (LACT \,|\, X) \,|\, Y \tag{R14}$$

where t_p and t_l are the types of *perm* and *LACT*, respectively, and $\phi = \lambda n_1 n_2 . n_1 \times (n_2 + 1) \times 0.001$.

$$LACT \,|\, X \xrightarrow[\phi]{\langle\langle t_l, t_b\rangle\rangle} GLU \,|\, GAL \,|\, X \tag{R15}$$

where t_l and t_b are the types of *LACT* and *betagal*, and $\phi = \lambda n_1 n_2 . (n_1 + 1) \times n_2 \times 0.001$.

Rule (R13) describes the incorporation of the lactose permease in the membrane of the bacterium, rule (R14) the transportation of lactose from the environment to the interior performed by the lactose permease, and rule (R15) the decomposition of the lactose into glucose (denoted GLU) and galactose (denoted GAL) performed by the beta galactosidase.

The initial state of the bacterium when no lactose is present in the environment and when 100 molecules of lactose are present are modelled, respectively, by the following terms (where $n \times T$ stands for a parallel composition $T \,|\, \ldots \,|\, T$ of length n):

$$Ecoli \;::=\; (m)^L \rfloor (lacI\text{–}A \,|\, 30 \times polym \,|\, 100 \times repr) \tag{2}$$

$$EcoliLact \;::=\; Ecoli \,|\, 100 \times LACT \tag{3}$$

Now, starting from the term $EcoliLact$, a possible stochastic trace generated by our semantics, given the rules above, is[1]:

$$\begin{aligned}
EcoliLact &\xrightarrow{\text{R3, } 30\times 0.1} 100 \times LACT \,|\, (m)^L \rfloor (lacI \cdot PP \cdot lacO \cdot lacZ \cdot lacY \cdot lacA \,|\, 29 \times polym \,|\, 100 \times repr) \\
&\xrightarrow{\text{R5, } 20} 100 \times LACT \,|\, (m)^L \rfloor (lacI\text{–}A \,|\, 30 \times polym \,|\, 100 \times repr \,|\, Rna) \\
&\xrightarrow{\text{R6, } 0.1} 100 \times LACT \,|\, (m)^L \rfloor (lacI\text{–}A \,|\, 30 \times polym \,|\, 100 \times repr \,|\, Rna \,|\, betagal \,|\, perm \,|\, transac) \\
&\xrightarrow{\text{R13, } 0.1} 100 \times LACT \,|\, (perm \cdot m)^L \rfloor (lacI\text{–}A \,|\, 30 \times polym \,|\, 100 \times repr \,|\, Rna \,|\, betagal \,|\, transac) \\
&\xrightarrow{\text{R14, } 100\times 0.001} 99 \times LACT \,|\, (perm \cdot m)^L \rfloor (lacI\text{–}A \,|\, 30 \times polym \,|\, 100 \times repr \,|\, Rna \,|\, betagal \,|\, transac \,|\, LACT) \\
&\xrightarrow{\text{R15, } 0.001} 99 \times LACT \,|\, (perm \cdot m)^L \rfloor (lacI\text{–}A \,|\, 30 \times polym \,|\, 100 \times repr \,|\, Rna \,|\, betagal \,|\, transac \,|\, GLU \,|\, GAL)
\end{aligned}$$

5 Conclusions and Related Work

This paper is a first proposal for using types in describing quantitative aspects of biological systems. Types for qualitative properties of the CSL calculus have been studied in [2] and [14]. We plan to develop a prototype simulator for our calculus TSCLS in order to experimentally test the expressiveness of our formalism. This would make possible to compare quantitatively the approach presented in this paper, with the one of [6].

In the remaining of this section we will put our paper in the framework of qualitative and quantitative models of biological systems.

[1] For simplicity we just show the rate of the transition reaching the target state considered in the trace. We avoid to report explicitly the whole exit rate from a given term, which should be computed, following the standard simulation algorithm, by summing up the rates for all the possible target states. For the sake of readability, we also show, on the transitions, the labels of the rules leading the state change.

Qualitative Models. In the last few years many formalisms originally developed by computer scientists to model systems of interacting components have been applied to Biology. Among these, there are Petri Nets [18], Hybrid Systems [1], and the π-calculus [11, 25]. Moreover, new formalisms have been defined for describing biomolecular and membrane interactions [4, 9, 10, 13, 22, 24]. Others, such as P-Systems [21], have been proposed as biologically inspired computational models and have been later applied to the description of biological systems.

The π-calculus and new calculi based on it [22, 24] have been particularly successful in the description of biological systems, as they allow describing systems in a compositional manner. Interactions of biological components are modelled as communications on channels whose names can be passed; sharing names of private channels allows describing biological compartments.

These calculi offer very low-level interaction primitives, but may cause the description models to become very large and difficult to read. Calculi such as those proposed in [9, 10, 13] give a more abstract description of systems and offer special biologically motivated operators. However, they are often specialized to the description of some particular kinds of phenomena such as membrane interactions or protein interactions.

P-Systems [21] have a simple notation and are not specialized to the description of a particular class of systems, but they are still not completely general. For instance, it is possible to describe biological membranes and the movement of molecules across membranes, and there are some variants able to describe also more complex membrane activities. However, the formalism is not so flexible to allow describing easily new activities observed on membranes without extending the formalism to model such activities.

Danos and Laneve [13] proposed the κ-calculus. This formalism is based on graph rewriting where the behaviour of processes (compounds) and of set of processes (solutions) is given by a set of rewrite rules which account for, e.g., activation, synthesis and complexation by explicitly modelling the binding sites of a protein.

The Calculus of Looping Sequences [4] has no explicit way to model protein domains (however they can be encoded, and a variant with explicit binding has been defined in [3]), but accounts for an explicit mechanism (the *looping sequences*) to deal with compartments and membranes. Thus, while the κ-calculus seems more suitable to model protein interactions, CLS allows for a more natural description of membrane interactions. Another feature lacking in other formalisms is the capacity to express ordered sequences of elements. To the best of our knowledge, CLS is the first formalism offering such a feature in an explicit way, thus allowing to naturally operate over proteins or DNA fragments which should be frequently defined as ordered sequences of elements.

Stochastic Models. Among stochastic process algebras we would like to mention the stochastic extension of the π-calculus, given by Priami et al. in [23], and the PEPA framework proposed by Hillston in [16]. We also would like to compare our work with two closer ones, namely [6] and [8].

The stochastic engine behind PEPA and the Stochastic π-calculus is constructed on the intuition of cooperating agents under different bandwidth limits. If two agents are interacting, the time spent for a communication is given by the slowest of the agents involved. Differently, our stochastic semantics is defined in terms of the collision-based paradigm introduced by Gillespie. A similar approach is taken in the quantitative variant of the κ-calculus ([12]) and in BioSPi ([23]). Motivated by the law of mass action, here we need to count the number of the reactants present in a system in order to compute the exact rate of a reaction. In [17], a stochastic semantics for bigraphs has been developed. An application in the field of systems biology has been provided by modelling a process of membrane budding.

A stochastic semantics for CLS (SCLS) has been defined in [6]. Such a semantics computes the transition rates by resorting to a complete counting mechanism to detect all the possible occurrences of patterns within a term. In our framework, the set of rule schemata that can be defined is limited with respect to SCLS, however, our counting mechanism, based on types, is quite more simple in practice. This would simplify, for example, the development of automatic simulators. As another advantage, our rules, similar to what happens in [8] for a variant of the ambient calculus, are equipped with rate functions, rather than with rate constants. Such functions may allow the definition of kinetics that are more complex than the standard mass-action ones.

Bioambients, [24], is a calculus in which biological systems are modelled using a variant of the ambient calculus. In Bioambients both membranes and elements are modelled by ambients, and activities by capabilities (enter, exit, expel, etc.). In [8], Bioambients are extended by allowing the rates associated with rules to be context dependent. Dependency is realized by associating to a rule a function which is evaluated when applying the rule, and depends on the context of the application. The context contains (as for our stochastic contexts) the state of the sibling ambients, that is the ambients in parallel in the innermost enclosing ambient (membrane). The property of the context used to determine the value of the function is its volume that synthesizes (with a real number) the elements present in the context. In Section 3 we sketched the representation of osmosis in our framework: the same example is presented with all details in [8]. However, our modelling is more general allowing to focus more selectively on context, and specifying functions that may also cause inhibition.

Finally MGS, `http://mgs.spatial-computing.org/`, is a domain specific language for simulation of biological processes. The state of a dynamical system is represented by a collection. The elements in the collection represent either entities (a subsystem or an atomic part of the dynamical system) or messages (signal, command, information, action, etc.) addressed to an entity. The dynamics is defined by rewrite rules specifying the collection to be substituted through a pattern language based on the neighborhood relationship induced by the topology of the collection. It is possible to specify stochastic rewrite strategies. In [20], this feature is used to provide the description of various models of the genetic switch of the λ phage, from a very simple biochemical description of the process to an individual-based model on a Delaunay graph topology. Note that, in MSG, the topological changes are programmed in some external language, whereas in CLS they are specified directly by the rewrite rules.

Acknowledgements. We thank the referees for their helpful comments. The final version of the paper improved due to their suggestions.

References

[1] Alur, R., Belta, C., Ivancic, F., Kumar, V., Mintz, M., Pappas, G.J., Rubin, H. and Schug, J. (2001) Hybrid Modelling and Simulation of Biomolecular Networks. *Proc. of Hybrid Systems: Computation and Control*, LNCS 2034, Springer, 19-32.

[2] Aman, B., Dezani-Ciancaglini, M., Troina, A. (2008) Type Disciplines for Analysing Biologically Relevant Properties. *Proc. of MeCBIC'08*, ENTCS 227, Elsevier, 97-111.

[3] Barbuti, R., Maggiolo-Schettini, A. and Milazzo, P. (2007) Extending the Calculus of Looping Sequences to Model Protein Interaction at the Domain Level. *Proc. of ISBRA'07*, LNBI 4463, Springer, 638-649.

[4] Barbuti, R., Maggiolo-Schettini, A., Milazzo, P. and Troina, A. (2006) A Calculus of Looping Sequences for Modelling Microbiological Systems. *Fund. Inform.*, **72**, 21-35.

[5] Barbuti, R., Maggiolo-Schettini, A., Milazzo, P. and Troina, A. (2006) Bisimulation Congruences in the Calculus of Looping Sequences. *Proc. of ICTAC'06*, LNCS 4281, Springer, 93-107.

[6] Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Tiberi, P.,Troina, A. (2008) Stochastic Calculus of Looping Sequences for the Modelling and Simulation of Cellular Pathways. *Transactions on Computational Systems Biology*, **vol. IX**, 86-113.

[7] Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., and Troina, A. (2008) Bisimulations in Calculi Modelling Membranes. *Formal Aspects of Computing*, **20**, 351-377.

[8] Bortolussi, L. and Vigliotti, M.G. EnVy: a Context-dependent Calculus for Biological Systems. *Proc. of QAPL'09*, ENTCS, Elsevier, to appear.

[9] Cardelli, L. (2005) Brane Calculi. Interactions of Biological Membranes. *Proc. of CMSB'04*, LNCS 3082, Springer, 257-280.

[10] Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F. and Schachter, V. (2004) Modeling and Querying Biomolecular Interaction Networks. *Theor. Comput. Sci.*, **325**, 25-44.

[11] Curti, M., Degano, P., Priami, C., and Baldari, C. T. (2004) Modelling Biochemical Pathways through Enhanced pi-calculus. *Theor. Comput. Sci.*, **325**, 111-140.

[12] Danos, V., Feret, J., Fontana, W., and Krivine, J. (2007) Scalable Modelling of Biological Pathways. *Proc. of APLAS'07*, LNCS 4807, Springer, 139-157.

[13] Danos, V. and Laneve, C. (2004) Formal Molecular Biology. *Theor. Comput. Sci.*, **325**, 69-110.

[14] Dezani-Ciancaglini, M., Giannini, P. and Troina, A. (2009) A Type System for Required/Excluded Elements in CLS. *Proc. of DCM'09*, EPTCS, to appear.

[15] Gillespie, D. (1977) Exact Stochastic Simulation of Coupled Chemical Reactions. *J. Phys. Chem.*, **81**, 2340-2361.

[16] Hillston. J. (1996) *A Compositional Approach to Performance Modelling*. Cambridge University Press.

[17] Krivine, J., Milner, R. and Troina, A. (2008) Stochastic Bigraphs. *Proc. of MFPS'08*, ENTCS 218, Elsevier, 73-96.

[18] Matsuno, H., Doi, A., Nagasaki, M. and Miyano, S. (2000) Hybrid Petri Net Representation of Gene Regulatory Network. *Proc. of Pacific Symposium on Biocomputing*, World Scientific Press, 341-352.

[19] Milazzo, P. (2007) *Qualitative and Quantitative Formal Modelling of Biological Systems*. Ph.D. Thesis, University of Pisa.

[20] Michel, O., Spicher, A., Giavitto, J.L. (2009) Rule-based Programming for Integrative Biological Modelling – Application to the Modelling of the Lambda Phage Genetic Switch. *Natural Computing*, Springer, to appear.

[21] Păun, G. (2002) *Membrane Computing. An Introduction.* Springer, 2002.

[22] Priami, C. and Quaglia, P. (2005) Beta Binders for Biological Interactions. *Proc. of CMSB'04*, LNCS 3082, Springer, 20-33.

[23] Priami, C., Regev, A., Shapiro, E. and Silverman, W. (2001) Application of a Stochastic Name-passing Calculus to Representation and Simulation of Molecular Processes. *Inform. Process. Lett.*, **80**, 25-31.

[24] Regev, A., Panina, E. M., Silverman, W., Cardelli, L. and Shapiro, E. (2004) BioAmbients: an Abstraction for Biological Compartments. *Theor. Comput. Sci.,* **325**, 141-167.

[25] Regev, A., Silverman, W. and Shapiro, E. Y. (2001) Representation and Simulation of Biochemical Processes Using the Pi-calculus Process Algebra. *Proc. of Pacific Symposium on Biocomputing*, World Scientific Press, 459-470.

[26] Taiz, L. and Zeiger, E. (2006) *Plant Physiology.* Sinauer Associated Inc.

[27] Wilkinson, D. (2006) *Stochastic Modelling for Systems Biology*. Chapman & Hall/CRC.

[28] Wong, P., Gladney, S. and Keasling, J. D. (1997) Mathematical Model of the Lac Operon: Inducer Exclusion, Catabolite Repression, and Diauxic Growth on Glucose and Lactose. *Biotechnology Progress*, **13**, 132-143.