Typed Stochastic Semantics for the Calculus of Looping Sequences[☆]

Livio Bioglio[a], Mariangiola Dezani-Ciancaglini[a],
Paola Giannini[b], Angelo Troina[a]

[a] *Dipartimento di Informatica, Università di Torino*
[b] *Dipartimento di Informatica, Università del Piemonte Orientale*

Abstract

The Stochastic Calculus of Looping Sequences is a quantitative term rewrite formalism suitable to describe the evolution of microbiological systems, taking into account the speed of the described activities. In this paper we propose an operational semantics for this calculus that considers the types of the species to derive the stochastic evolution of the system. The presence of positive and negative catalysers can modify these speeds. We claim that types provide an abstraction suitable to represent the interaction between elements without specifying exactly the element positions. Our claim is supported through an example modelling the lactose operon.

Keywords: Calculus of Looping Sequences, Systems Biology, Stochastic Semantics

1. Introduction

The Calculus of Looping Sequences (CLS for short) [1, 2, 3], 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 [2, 4], which are common in process calculi, thus combining the simplicity of notation of rewrite systems with the advantage of a

[☆] This research is founded by the BioBITs Project (*Converging Technologies* 2007, area: Biotechnology-ICT), Regione Piemonte.

form of compositionality. A stochastic version of CLS (SCLS for short) is proposed in [5]. 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 characterises 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 [6].

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 [5] by imposing some restrictions on the patterns modelling the rewrite rules. Each rewrite rule states explicitly the types of the elements whose occurrences may speed-up or slow-down a reaction. The occurrences of the elements of these types are then processed by a rate function which is used to compute the actual rate of a transition. We show how we can define patterns in our 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.

While standard quantitative bio-inspired formalisms give stochastic semantics based on constant rates, we equip the rewrite rules of our calculus with a rate function. This makes possible the definition of a stochastic semantics that is more general than the classical one based on collision analysis (which is practical for very low level analyses, such as chemical interactions). In particular, we can define rules, whose evolutions follow different probability distributions. This is useful for higher level simulations, for example cellular or tissue interactions, or in other cases such as in the presence of enzymes (molecules that speed up the reaction) or inhibitors (molecules that slow down the reaction) where the reaction rate equation becomes complicated, and must be calculated using non-linear equations. We show how a particular interpretation of the rate function could be used to recover Gillespie's method.

More specifically we add to the reduction rules of CLS the information on the relevant objects and a function which computes the rate of the reduction starting from the number of relevant objects which can occur in the instantiations of the variables.

As a simple example consider a molecule a becoming a molecule b with

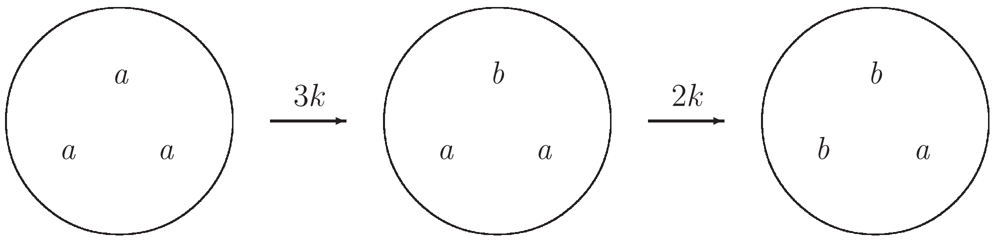

Figure 1: Application of $a \to b$ with kinetic constant k.

a kinetic constant k. The rate of this transformation is proportional to the concentration of a's. Figure 1 shows how a molecules contained inside a membrane evolve with a rate calculated by means of k and the concentration of a's. This transformation is modelled in our calculus by the rewrite rule:

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

where t is the type of molecule a, the overline means that a occurs in a parallel composition, and the function ϕ is $\lambda n.(n+1) \times k$. When reducing a term by means of this rule we compute the rate by applying the function ϕ to the number of parallel occurrences of molecules of type t, in the term matching the variable X (representing the environment in which the rule will be applied).

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

The present paper is an extended version of [7].

1.1. Summary

The remainder of this paper is organised 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. We propose an interpretation of the rate function able to cover Gillespie's collision based algorithm. In Section 4 we use our framework to model the lactose operon of *Escherichia Coli*. Finally, in Section 5 we draw our conclusions and we discuss 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 by defining the syntax of terms. We assume a possibly infinite alphabet $\mathcal{E}$ 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:*

$$\begin{array}{lcl} T & ::= & S \;\;\big|\;\; (S)^L \rfloor T \;\;\big|\;\; T \,|\, T \\ S & ::= & \epsilon \;\;\big|\;\; a \;\;\big|\;\; S \cdot S \end{array}$$

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

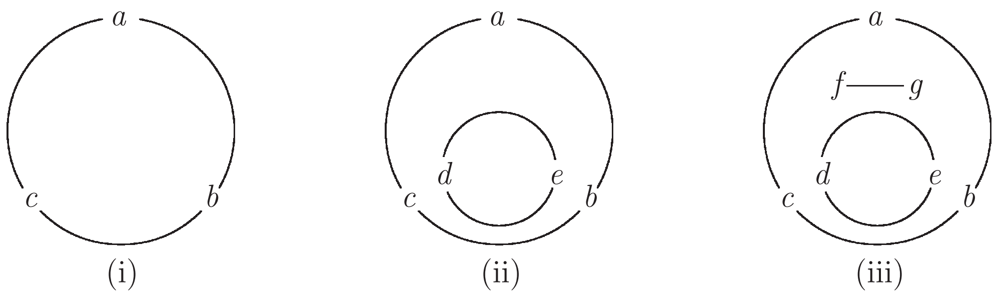

Figure 2: (i) represents $(a \cdot b \cdot c)^L$; (ii) represents $(a \cdot b \cdot c)^L \rfloor (d \cdot e)^L$; (iii) represents $(a \cdot b \cdot c)^L \rfloor ((d \cdot e)^L \mid f \cdot g)$.

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 $\_ \mid \_$. In Figure 2 we show some examples of CLS terms and their visual representation, using $(S)^L$ as a shortcut for $(S)^L \rfloor \epsilon$.

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

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:*

$$\begin{gathered}
S_1 \cdot (S_2 \cdot S_3) \equiv_S (S_1 \cdot S_2) \cdot S_3 \qquad S \cdot \epsilon \equiv_S \epsilon \cdot S \equiv_S S \\
S_1 \equiv_S S_2 \ \textit{ implies } \ S_1 \equiv_T S_2 \ \textit{ and } \ (S_1)^L \rfloor T \equiv_T (S_2)^L \rfloor T \\
T_1 \mid T_2 \equiv_T T_2 \mid T_1 \qquad T_1 \mid (T_2 \mid T_3) \equiv_T (T_1 \mid T_2) \mid T_3 \qquad T \mid \epsilon \equiv_T T \\
(\epsilon)^L \rfloor \epsilon \equiv_T \epsilon \qquad (S_1 \cdot S_2)^L \rfloor T \equiv_T (S_2 \cdot S_1)^L \rfloor T
\end{gathered}$$

Rules of 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 may have 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 $\mathcal{TV}$ ranged over by $X, Y, Z, \ldots$, a set of sequence variables $\mathcal{SV}$ ranged over by $\widetilde{x}, \widetilde{y}, \widetilde{z}, \ldots$, and a set of element variables $\mathcal{X}$ ranged over by $x, y, z, \ldots$. All these sets are possibly infinite and pairwise disjoint. We denote by $\mathcal{V}$ the set of all variables, $\mathcal{V} = \mathcal{TV} \cup \mathcal{SV} \cup \mathcal{X}$, and with χ a generic variable of $\mathcal{V}$. 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:*

$$\begin{array}{lcl} P & ::= & SP \;\big|\; (SP)^L \rfloor P \;\big|\; P \,|\, P \;\big|\; X \\ SP & ::= & \epsilon \;\big|\; a \;\big|\; SP \cdot SP \;\big|\; \widetilde{x} \;\big|\; x \end{array}$$

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

We assume the structural congruence relation to be trivially extended to patterns. An *instantiation* is a partial function $\sigma : \mathcal{V} \to \mathcal{T}$. An instantiation must preserve the type of variables, thus for $X \in \mathcal{TV}, \widetilde{x} \in \mathcal{SV}$ and $x \in \mathcal{X}$ we have $\sigma(X) \in \mathcal{T}, \sigma(\widetilde{x}) \in \mathcal{S}$ and $\sigma(x) \in \mathcal{E}$, respectively. Given $P \in \mathcal{P}$, with $P\sigma$ we denote the term obtained by replacing each occurrence of each variable $\chi \in \mathcal{V}$ appearing in P with the corresponding term $\sigma(\chi)$. With Σ we denote the set of all the possible instantiations and, given $P \in \mathcal{P}$, 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 $P_1, P_2 \in \mathcal{P}$, $P_1 \not\equiv \epsilon$ 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 $P_2\sigma$. The semantics of CLS is 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:*

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

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

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

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[1]. 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* $\mathcal{R}$*, the* semantics *of* CLS *is the least relation closed with respect to* $\equiv$ *and satisfying the following rule:*

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

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

Given a set of rewrite rules $\mathcal{R}$, 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.

[1]This justifies the clause $S_1 \equiv_S S_2$ implies $(S_1)^L \rfloor T \equiv_T (S_2)^L \rfloor T$ in Definition 3.3.

3. Typed Stochastic CLS

In this section we show how an abstraction on the elements, that induces an abstraction on terms, may be used to enhance the expressiveness of CLS. In particular, we use this abstraction to focus on quantitative aspects of CLS, by showing how to model the speeds of the biological activities.

We consider a partition, $\mathfrak{T}$, of the sets of elements in $\mathcal{E}$. By t, with subscripts and superscript if needed, we denote the equivalence classes in $\mathfrak{T}$. In the following we assume a fixed $\mathfrak{T}$. Given a molecule represented by an element a in $\mathcal{E}$, we say that the *type of* a is the equivalence class $t \in \mathfrak{T}$ to which a belongs. For example, the elements a and b could represent different enzymes, say two isozymes, both catalysing a certain reaction. In this case, in the modelling of the reaction, we identify a and b assigning to them the same type t.

A term T is abstracted by saying that the *type of* T is the multiset of the types of the elements in its outermost parallel composition. We distinguish between occurrences of elements in parallel with other terms, and occurrences of elements within a sequence by having two names for each type. In particular the type of T will contain a $\overline{t}$ for each occurrence of $a \in t$ in parallel with some other term, and $\widetilde{t}$ for each occurrence of $a \in t$ which is either in a sequence or in the looping sequence of a compartment (in both cases we only consider the outermost parallel composition). We use $\overset{*}{t}$ to range over both $\overline{t}$ and $\widetilde{t}$, and τ to range over types of terms. By $\overset{*}{t} \in_n \tau$ we denote that $\overset{*}{t}$ occurs n times in τ, and $\uplus$ is the union on multisets. In the following when we say type we refer to either t's, or $\overset{*}{t}$'s, or τ's.

The following definition formalises the mappings *ptype* on terms and *stype* on sequences.

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

-
  - $\mathit{ptype}((S)^L \rfloor T) = \mathit{stype}(S)$
  - $\mathit{ptype}(T_1 \,|\, T_2) = \mathit{ptype}(T_1) \uplus \mathit{ptype}(T_2)$
  - $\mathit{ptype}(S_1 \cdot S_2) = \mathit{stype}(S_1 \cdot S_2)$
  - $\mathit{ptype}(a) = \{\overline{t}\}$ *if* $a \in t$
-
  - $\mathit{stype}(S_1 \cdot S_2) = \mathit{stype}(S_1) \uplus \mathit{stype}(S_2)$

– $\mathit{stype}(a) = \{\widetilde{t}\}$ *if* $a \in t$.

For example if $\mathfrak{T} = \{t_a, t_b, t_c\}$, where $t_a = \{a\}$, $t_b = \{b\}$, and $t_c = \{c\}$ we have $\mathit{ptype}(a \,|\, a \,|\, c) = \{\overline{t_a}, \overline{t_a}, \overline{t_c}\}$, $\mathit{ptype}(b{\cdot}c{\cdot}c) = \{\widetilde{t_b}, \widetilde{t_c}, \widetilde{t_c}\}$, $\mathit{ptype}(a \,|\, a \,|\, c \,|\, (b \cdot c \cdot c)^L \rfloor a) = \{\overline{t_a}, \overline{t_a}, \overline{t_c}, \widetilde{t_b}, \widetilde{t_c}, \widetilde{t_c}\}$ and $\mathit{ptype}((b \cdot c \cdot c)^L \rfloor (a \,|\, a \,|\, a \,|\, c)) = \{\widetilde{t_b}, \widetilde{t_c}, \widetilde{t_c}\}$.
Instead if $\mathfrak{T} = \{t, t'\}$, where $t = \{a, b\}$, and $t' = \{c\}$ we get $\mathit{ptype}(a \,|\, a \,|\, c) = \{\overline{t}, \overline{t}, \overline{t'}\}$, $\mathit{ptype}(b \cdot c \cdot c) = \{\widetilde{t}, \widetilde{t'}, \widetilde{t'}\}$, $\mathit{ptype}(a \,|\, a \,|\, c \,|\, (b \cdot c \cdot c)^L \rfloor a) = \{\overline{t}, \overline{t}, \overline{t'}, \widetilde{t}, \widetilde{t'}, \widetilde{t'}\}$ and $\mathit{ptype}((b \cdot c \cdot c)^L \rfloor (a \,|\, a \,|\, a \,|\, c)) = \{\widetilde{t}, \widetilde{t'}, \widetilde{t'}\}$.

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

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

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

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

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

$$a \,|\, X \xrightarrow[\phi]{\langle\langle \overline{t}_a, \overline{t}_c \rangle\rangle} b \,|\, X \tag{1}$$

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.

Remark 3.2. *We consider local interactions, that is interactions between elements in the same compartment. As for the CLS semantics, given a term we match the left-hand-side pattern of a rule against the subterm contained in the hole of a context. However, in our case, when applying a rule we have to take into account a whole compartment, since our function depends only on the content of the subterm matching the pattern. For instance, for the previous example, matching the term* $(S)^L \rfloor (a \,|\, a \,|\, c \,|\, c)$, *we do not want a context* $C = (S)^L \rfloor (\Box \,|\, c)$ *that would cause a miscounting of the* c*'s present in the compartment.*

Given the above remark, we restrict Definition 2.5, to permit only a hole filling a compartment or the whole term.

Definition 3.3 (Stochastic Contexts). Stochastic Contexts C *are defined as:*

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

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

We can now define the typed semantics.

Definition 3.4 (Typed Stochastic Semantics). *Given a finite set* $\mathcal{R}$ *of rewrite rules, the* semantics *of* TSCLS *is the least relation closed with respect to* $\equiv$ *and satisfying the following rule:*

$$\frac{\begin{array}{c} P_1 \xrightarrow[\phi]{\langle \Pi_1,\ldots,\Pi_m \rangle} P_2 \in \mathcal{R} \quad \Pi_i = \langle \overset{*(i)}{t_1}, \ldots, \overset{*(i)}{t_{q_i}} \rangle \\ \sigma \in \Sigma \quad P_1\sigma \not\equiv \epsilon \quad C \in \mathcal{SC} \quad V(P_1) = \langle \chi_1, \ldots, \chi_m \rangle \\ ptype(\sigma(\chi_i)) = \tau_i \qquad t_j^{(i)} \in_{n_j^{(i)}} \tau_i \quad (1 \leq j \leq q_i) \quad (1 \leq i \leq m) \\ r = \phi\; n_1^{(1)} \ldots n_{q_1}^{(1)} \cdots n_1^{(m)} \ldots n_{q_m}^{(m)} \end{array}}{C[P_1\sigma] \xrightarrow{r} C[P_2\sigma]}$$

Example 3.5. *Applying rule (1) with the empty context to the term $a \,|\, a \,|\, c$ we have:*

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

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

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

Similarly, applying (1) to the term $(b \cdot c \cdot c)^L \rfloor (a \,|\, a \,|\, a \,|\, c)$ with the context $\epsilon \,|\, (b \cdot c \cdot c)^L \rfloor \square$ we get:

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

Note that we cannot simply use Definition 2.5 for the contexts in the stochastic framework, since we would not count correctly the numbers of elements which influence the speed of transformations (see Remark 3.2). For example, rule (1) applied to the term $a \,|\, a \,|\, c$ with the context $\square \,|\, a \,|\, c$ 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 when different reactions are competing with different rates, the ones which are not chosen should restart the competition at the following step.

From the transition rates of Definition 3.4, we can define an exponential probability distribution of the moment in which the next reaction will take place as follows. Given a term T, a global time Δ and all the transitions

$R_1, \ldots, R_M$ that can be applied to T, with rates, respectively, $r_1, \ldots, r_M$ such that $r = \sum_{i=1}^{M} r_i$, the standard simulation procedure consists of the following two steps:

- The time $\Delta + \delta$ at which the next stochastic reduction will occur is randomly chosen with δ exponentially distributed with parameter r;
- The reduction R_i that will occur at time $\Delta + \delta$ is randomly chosen with probability r_i / r.

3.1. Modelling Guidelines

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

(i) As discussed in the Introduction, 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:

$$a \,|\, X \xrightarrow[\phi]{\langle\langle \bar{t}_a \rangle\rangle} b \,|\, X$$

where $\phi = \lambda n.(n+1) \times k$. Notice that only occurrences of a in parallel can become b, so using this rule we get for example:

$$(m)^L \rfloor (a \,|\, a \,|\, a \cdot a) \xrightarrow{2k} (m)^L \rfloor (b \,|\, a \,|\, a \cdot a)$$

where m is any membrane.

(ii) 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:

$$a \,|\, b \,|\, X \xrightarrow[\phi]{\langle\langle \bar{t}_a, \bar{t}_b \rangle\rangle} c \,|\, X$$

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*:

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

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

(iii) 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 [8] 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 $\psi_s = -\mathsf{RT}c_s$ (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:

$$(\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y \xrightarrow[\phi]{\langle\langle\rangle, \langle \bar{t}_a, \bar{t}_b \rangle, \langle \bar{t}_a, \bar{t}_b \rangle\rangle} (\widetilde{x})^L \rfloor X \,|\, a \,|\, Y$$

$$(\widetilde{x})^L \rfloor X \,|\, a \,|\, Y \xrightarrow[\phi']{\langle\langle\rangle, \langle \bar{t}_a, \bar{t}_b \rangle, \langle \bar{t}_a, \bar{t}_b \rangle\rangle} (\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y$$

where $\begin{array}{l} \phi = \lambda n_1 n_2 n_3 n_4 . \frac{S}{V} \times (\frac{n_2}{(n_1+1)V_a + n_2 V_b} - \frac{n_4}{(n_3+1)V_a + n_4 V_b}) \times k \\ \phi' = \lambda n_1 n_2 n_3 n_4 . \frac{S}{V} \times (\frac{n_4}{(n_3+1)V_a + n_4 V_b} - \frac{n_2}{(n_1+1)V_a + n_2 V_b}) \times k \end{array}$

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:

$$(\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y \xrightarrow[\phi]{\langle\langle \widetilde{t}_c \rangle, \langle \bar{t}_a, \bar{t}_b \rangle, \langle \bar{t}_a, \bar{t}_b \rangle\rangle} (\widetilde{x})^L \rfloor X \,|\, a \,|\, Y$$

$$(\widetilde{x})^L \rfloor X \,|\, a \,|\, Y \xrightarrow[\phi']{\langle\langle \widetilde{t}_c \rangle, \langle \bar{t}_a, \bar{t}_b \rangle, \langle \bar{t}_a, \bar{t}_b \rangle\rangle} (\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y$$

where $\begin{array}{l}\phi = \lambda n_1 n_2 n_3 n_4 n_5.(n_1 \times k_c + 1) \times \frac{S}{V} \times \\ \quad (\frac{n_3}{(n_2+1)V_a+n_3V_b} - \frac{n_5}{(n_4+1)V_a+n_5V_b}) \times k \\ \phi' = \lambda n_1 n_2 n_3 n_4 n_5.(n_1 \times k_c + 1) \times \frac{S}{V} \times \\ \quad (\frac{n_5}{(n_4+1)V_a+n_5V_b} - \frac{n_3}{(n_2+1)V_a+n_3V_b}) \times k\end{array}$

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:

$$(\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y \xrightarrow[\phi]{\langle\langle\widetilde{t_c}\rangle,\langle\overline{t}_a,\overline{t}_b\rangle,\langle\overline{t}_a,\overline{t}_b\rangle\rangle} (\widetilde{x})^L \rfloor X \,|\, a \,|\, Y$$
$$(\widetilde{x})^L \rfloor X \,|\, a \,|\, Y \xrightarrow[\phi']{\langle\langle\widetilde{t_c}\rangle,\langle\overline{t}_a,\overline{t}_b\rangle,\langle\overline{t}_a,\overline{t}_b\rangle\rangle} (\widetilde{x})^L \rfloor (X \,|\, a) \,|\, Y$$

where $\begin{array}{l}\phi = \lambda n_1 n_2 n_3 n_4 n_5.\frac{1}{\text{if } n_1{=}0 \text{ then } 1 \text{ else } n_1 \times k_c} \times \\ \quad \frac{S}{V} \times (\frac{n_3}{(n_2+1)V_a+n_3V_b} - \frac{n_5}{(n_4+1)V_a+n_5V_b}) \times k \\ \phi' = \lambda n_1 n_2 n_3 n_4 n_5.\frac{1}{\text{if } n_1{=}0 \text{ then } 1 \text{ else } n_1 \times k_c} \times \frac{S}{V} \times \\ \quad (\frac{n_5}{(n_4+1)V_a+n_5V_b} - \frac{n_3}{(n_2+1)V_a+n_3V_b}) \times k\end{array}$

and k_c is the deceleration due to the presence of one element c.

(iv) If the rule

$$P_1 \xrightarrow[\phi]{\overrightarrow{\Pi}} P_2$$

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 $\lambda \overrightarrow{n}\, \overrightarrow{n_X}.e$, where $\overrightarrow{n}$ takes into account the types of the elements occurring in P_1' and $\overrightarrow{n_X}$ takes into account the types of the elements occurring in X, we define:

- Π'_X as the list whose head is $\overline{t}_c$ and whose tail is Π_X,
- $\phi' = \lambda \overrightarrow{n}\, n_c \overrightarrow{n_X}.e \times (n_c \times k + 1)$,

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:

$$P_1 \,|\, X \xrightarrow[\phi']{\overrightarrow{\Pi}^\frown\langle\langle \bar{t}_c\rangle\rangle} P_2 \,|\, X$$

where $^\frown$ represents list concatenation and if $\phi = \lambda\overrightarrow{n}.e$, then $\phi' = \lambda\overrightarrow{n} n_c.e \times (n_c \times k + 1)$.

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 $P_1 \xrightarrow[\phi]{\overrightarrow{\Pi}} P_2$ if $P_1 \equiv P'_1 \,|\, X$ and Π_X, ϕ are as above we define:

- $\Pi'_X = \langle \bar{t}_c, \bar{t}_d\rangle^\frown \Pi_X$,
- $\phi' = \lambda\overrightarrow{n} n_c n_d \overrightarrow{n_X}.e \times \frac{n_c \times k+1}{\text{if } n_d=0 \text{ then } 1 \text{ else } n_d \times k'}$,

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:

$$P_1 \,|\, X \xrightarrow[\phi']{\overrightarrow{\Pi}^\frown\langle\langle \bar{t}_c, \bar{t}_d\rangle\rangle} P_2 \,|\, X$$

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

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.

3.1.1. Recovering Gillespie's Framework

Gillespie's approach (see [6]) simulates the time evolution of a chemically reacting system by determining when the next reaction will occur and what kind of reaction it will be. Kind and time of the next reaction are computed on the basis of a stochastic reaction constant.

Gillespie's stochastic simulation algorithm is defined for populations of a well-stirred mixture of N molecular species $\{s_1, ..., s_N\}$ interacting through M chemical reactions $\{R_1, ..., R_M\}$ under the conditions that the molecules are confined to a fixed volume and kept at constant temperature.

We might restrict TSCLS in order to match Gillespie's framework. Since we just need to deal with simple molecular populations, we restrict our calculus eliminating the sequencing and the looping operators. We denote with $\mathcal{T}_G$ the infinite set of terms representing Gillespie's molecular populations.

Definition 3.6. *$\mathcal{T}_G$ is the infinite set of TSCLS terms built as the parallel composition of atomic elements.*

The usual notation for chemical reactions can be expressed by:

$$\ell_1 s_1 + \ldots + \ell_m s_m \xrightarrow{k} \ell'_1 p_1 + \ldots + \ell'_n p_n \tag{2}$$

where s_i and p_i are the reagents and product molecules, respectively, ℓ_i, ℓ'_i are the stoichiometric coefficients and k is the kinetic constant.

We denote with $\mathcal{P}_G$ the infinite set of patterns built as the parallel composition of atomic elements and exactly one variable. In particular, we restrict to rewrite rules modelling chemical reactions of the shape of rule (2). Namely, assuming that each species of the molecular population has a different basic type, and, in particular, $t_1, \ldots, t_m$ are the basic types of $s_1, \ldots, s_m$, a chemical reaction of the form described by rule (2) can be expressed by the following TSCLS rewrite rule:

$$\ell_1 \times s_1 \,|\, \ldots \,|\, \ell_m \times s_m \,|\, X \xrightarrow[\phi]{\langle\langle \bar{t}_1, \ldots, \bar{t}_m \rangle\rangle} \ell'_1 \times p_1 \,|\, \ldots \,|\, \ell'_n \times p_n \,|\, X \tag{3}$$

where $\ell \times s$ stands for a parallel composition $s \,|\, \ldots \,|\, s$ of length ℓ and similarly for $\ell \times p$.

We now need to define the weighting function ϕ of rule (3) used to model Gillespie's collision based stochastic simulation algorithm. Intuitively, items (i) and (ii) of Section 3.1 already go in this direction (they actually define particular subcases of general chemical reactions expressed by rule (3)).

In particular, the collision based framework defined by Gillespie, when the stoichiometry ℓ of a reagent is greater than 1, picks one of all the possible combinations of ℓ reagents. This leads to binomial distributions of the reagents involved. Namely, we define the weighting function ϕ as:

$$\phi = \lambda n_1 \ldots n_m . \binom{n_1 + \ell_1}{\ell_1} \times \ldots \times \binom{n_m + \ell_m}{\ell_m} \times k \tag{4}$$

where k is the kinetic constant of the modelled chemical reaction.

By construction, the following holds.

Proposition 3.7. *Molecular populations defined as $\mathcal{T}_G$ terms with a fixed set of rules of the shape of rule (3) interpret Gillespie's framework for the evolution of chemically reacting systems into TSCLS.*

Even if Gillespie's method is defined for simple populations of species, it has been greatly reused in more complex frameworks, e.g. in calculi where compartmentalisation and linked structures where taken into account (see, for example, [9, 10, 5, 11, 12, 13]). It is debatable whether such an extension of the usage of Gillespie's method is still correct: the assumptions behind this method are quite strict and considering as same collisions happening between free molecules and molecules bound on a membrane or a protein structure could not always result in a faithful model. We can manage this kind of situations with ad-hoc instantiations of the weight function, allowing us to define more general evolutions than the ones ruled by the law of mass action, (see, e.g., items (iii) and (iv) of Section 3.1 and the example about cell division in Section 3.1.2). There are also some cases in which Gillepie's method appears to be reasonably applicable also when its strict assumptions are not fully satisfied. As an example, see the lactose operon case study in Section 4, in which, on the lines of [5], we apply a Gillespie based analysis also to rules involving compartments (rules R13 and R14).

3.1.2. Cell Division: an Example of Multi-Match Patterns

We have just seen how the weighting function introduced in our stochastic semantics can be used to interpret the classical Gillespie's model. We have also seen how more complex interactions can be modelled (see items (iii) and (iv) of Section 3.1). In this subsection we consider a more complicated but intriguing case.

In [13], interesting considerations about the structure of rewrite rules are raised. Actually, different conditions can be placed on the structure of the patterns, some of them might be natural when modelling biological systems, some might be not.

Consider the following rule modelling the splitting of a cell and the distribution of its content to the newly produced cells (abstract away, for the moment, from the type list and the weighting function):

$$(\widetilde{x} \cdot \widetilde{y})^L \rfloor (X \mid Y) \to (\widetilde{x})^L \rfloor X \mid (\widetilde{y})^L \rfloor Y$$

The left pattern contains multiple variables within the same sub-term giving rise to several different variables instantiations for a same term. For example some possible matchings for the term $(a \cdot b \cdot c)^L \rfloor (d \mid e \mid f)$ are:

- $\widetilde{x} = a$, $\widetilde{y} = b \cdot c$, $X = d$, $Y = e \,|\, f$;
- $\widetilde{x} = a \cdot b$, $\widetilde{y} = c$, $X = d \,|\, e$, $Y = f$;
- $\widetilde{x} = b$, $\widetilde{y} = c \cdot a$, $X = e$, $Y = d \,|\, f$.

In such cases it is not clear how the stochastic rate should be parcelled out among the possible matches of the four variables. Gillespie's method, which does not deal with compartments, non linear and multi-match rules, could not be used in this kind of situations. In [13, 11] this kind of patterns are prevented and such a rule could not be used to directly model a natural biological phenomenon such as cell division. In our framework, we might resort to the weighting function to deal with this kind of situations.

To add some detail, we refer to the splitting example proposed in [13]. In [14], Rosenfeld et al. propose a methodology to analyse the gene regulation function of a particular protein. They start by considering a high concentration of a repressor protein within a single cell. During cell division, each daughter cell receives approximately one half the population of the repressor.[2] As a consequence, after a few divisions, the concentration of the repressor becomes low enough to trigger the production of the target protein.

Abstracting away the set of reactions occurring inside the cell and leading the gene expression, we focus on the rule modelling cell division. Consider a simple cell containing a certain number, say n of repressor proteins: $(cell)^L \rfloor (n \times rep)$. Let t_r be the basic type of the repressor protein, a TSCLS rule modelling a split distributing about $n/2$ repressor proteins to the two freshly produced cells could be defined as follows:

$$(cell)^L \rfloor (X \,|\, Y) \xrightarrow[\phi]{\langle\langle \bar{t}_r \rangle \langle \bar{t}_r \rangle\rangle} (cell)^L \rfloor X \,|\, (cell)^L \rfloor Y$$

with the weighting function:

$$\phi = \lambda n_1 n_2 . \frac{k}{1 + |n_1 - n_2| \times k'}$$

where k and k' are used to weight the distribution of the repressor proteins to the new cells (the more far is the partition from the ideal half, the lower the value returned by ϕ).

[2]For simplicity we do not consider a detailed volumetrical analysis. We just suppose the volume of the cell increases during the mitosis phase and that the two resulting daughter cells have the same volume of the mother cell.

The example could be extended in the natural way to take into account any other species within the dividing cell. For the sake of simplicity, we presented here a very naive example of partitioning, more complex functions could be defined to randomly distribute the population of each species of a cell between its two children.

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 deeply studied of all living things and it is a favorite 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 synthesise degradative enzymes unless the substrates for these enzymes are present in the environment. For example, E. coli does not synthesise 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 permitted) into the mRNA for beta galactosidase, lactose permease and transacetylase, respectively.

The regulation process is as follows (see Figure 3): 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

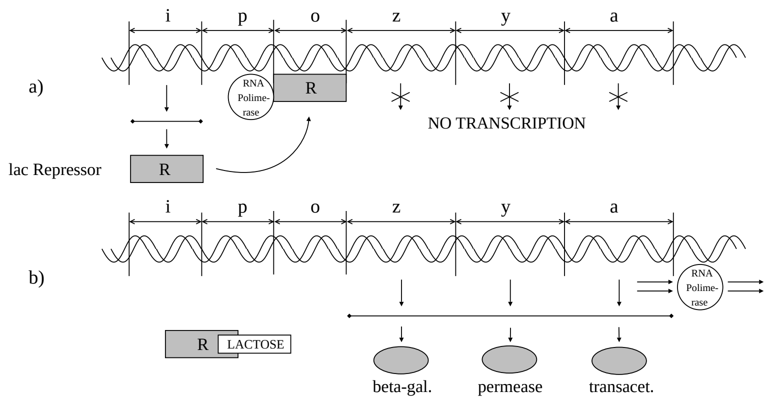

Figure 3: The regulation process in the Lac Operon.

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 synthesised.

4.1. Typed Stochastic CLS Model

A detailed mathematical model of the regulation process can be found in [15]. 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 [16]. 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:

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

where $\phi = 0.02$.

$$\text{(R2)} \qquad Irna \,|\, X \xrightarrow[\phi]{\langle\langle\bar{t}\rangle\rangle} Irna \,|\, repr \,|\, X$$

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

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

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

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

where $\phi = 0.01$.

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

where $\phi = 20$.

$$\text{(R6)} \qquad Rna \,|\, X \xrightarrow[\phi]{\langle\langle\bar{t}\rangle\rangle} Rna \,|\, betagal \,|\, perm \,|\, transac \,|\, X$$

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

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

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

(R8)

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

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

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

where $\phi = 0.01$.

(R10)

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

where $\phi = 0.01$.

(R11) $$repr \,|\, LACT \,|\, X \xrightarrow[\phi]{\langle\langle \bar{t}_r, \bar{t}_l \rangle\rangle} RLACT \,|\, X$$

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

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

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:

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

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

$$\text{(R14)} \qquad (\widetilde{x})^L \rfloor X \,|\, LACT \,|\, Y \xrightarrow[\phi]{\langle\langle\widetilde{t}_p\rangle,\langle\rangle,\langle\overline{t}_l\rangle\rangle} (\widetilde{x})^L \rfloor (LACT \,|\, X) \,|\, Y$$

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

$$\text{(R15)} \qquad LACT \,|\, X \xrightarrow[\phi]{\langle\langle\overline{t}_l,\overline{t}_b\rangle\rangle} GLU \,|\, GAL \,|\, X$$

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):

$$(5) \qquad Ecoli \ ::= \ (m)^L \rfloor (lacI\text{–}A \,|\, 30 \times polym \,|\, 100 \times repr)$$

$$(6) \qquad EcoliLact \ ::= \ Ecoli \,|\, 10000 \times LACT$$

Now, starting from the term *EcoliLact*, a possible stochastic trace gen-

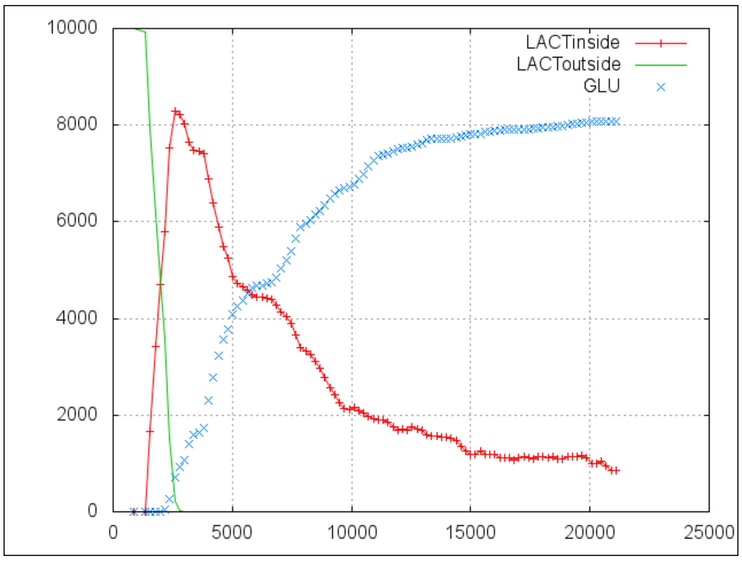

Figure 4: Simulation results: absorption and degradation of lactose into glucose.

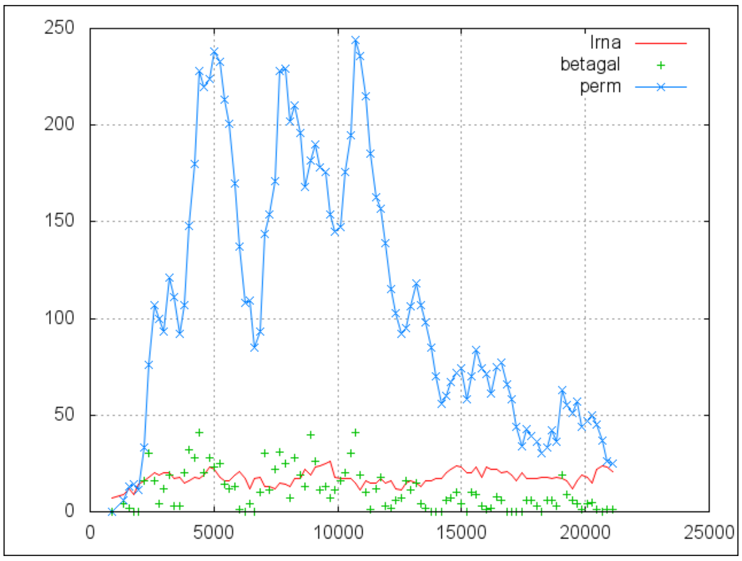

Figure 5: Simulation results: production of enzymes.

erated by our semantics, given the rules above, is[3]:

$$
\begin{array}{ll}
 & EcoliLact \\
\xrightarrow{\text{R3, }30\times 0.1} & 10000 \times LACT \mid (m)^L \rfloor (lacI \cdot PP \cdot lacO \cdot lacZ \cdot lacY \cdot lacA \mid \\
 & 29 \times polym \mid 100 \times repr) \\
\xrightarrow{\text{R5, }20} & 10000 \times LACT \mid (m)^L \rfloor (lacI\text{–}A \mid 30 \times polym \mid 100 \times repr \mid \\
 & Rna) \\
\xrightarrow{\text{R6, }0.1} & 10000 \times LACT \mid (m)^L \rfloor (lacI\text{–}A \mid 30 \times polym \mid 100 \times repr \mid \\
 & Rna \mid betagal \mid perm \mid transac) \\
\xrightarrow{\text{R13, }0.1} & 10000 \times LACT \mid (perm \cdot m)^L \rfloor (lacI\text{–}A \mid 30 \times polym \mid \\
 & 100 \times repr \mid Rna \mid betagal \mid transac) \\
\xrightarrow{\text{R14, }10000\times 0.001} & 9999 \times LACT \mid (perm \cdot m)^L \rfloor (lacI\text{–}A \mid 30 \times polym \mid \\
 & 100 \times repr \mid Rna \mid betagal \mid transac \mid LACT) \\
\xrightarrow{\text{R15, }0.001} & 9999 \times LACT \mid (perm \cdot m)^L \rfloor (lacI\text{–}A \mid 30 \times polym \mid \\
 & 100 \times repr \mid Rna \mid betagal \mid transac \mid GLU \mid GAL).
\end{array}
$$

In Figure 4 and Figure 5 we show the results of a TSCLS simulation of the term *EcoliLact* obtained with a prototype simulator for TSCLS written in JAVA. For a more realistic simulation we added to the model also the rules describing the spontaneous degradation of the elements involved in the model (omitted here for simplicity, a complete description of the simulated model is available at: `http://www.di.unito.it/∼giannini/TSCLSim/`). In particular, in Figure 4, we show the absorption of lactose showing the concentrations of lactose outside and inside the bacterium and, inside the bacterium, the degradation of lactose into glucose. In Figure 5, we show the concentrations of the enzymes *Irna*, *betaga* and *perm* (notice how the production of the *perm* enzyme inside the bacterium is activated after the absorption of the lactose).

[3]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.

5. Conclusions and Related Work

This paper is a first proposal for using a type abstraction in describing quantitative aspects of biological systems. Types for qualitative properties of the CSL calculus have been studied in [17, 18, 19].

In the remaining of this section we will put our paper in the framework of qualitative and quantitative models of biological systems, and of the literature about type system analysis of biological properties.

5.1. 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 [20], Hybrid Systems [21], and the π-calculus [22, 23]. Moreover, new formalisms have been defined for describing biomolecular and membrane interactions [1, 24, 25, 26, 27, 28]. Others, such as P-Systems [29], 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 [27, 28] have been particularly successful in modelling biological systems, as they permit a compositional description. Interactions of biological components are modelled as communications on channels whose names can be passed; sharing names of private channels makes possible to model 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 [24, 25, 26] give a more abstract description of systems and offer special biologically motivated operators. However, they are often specialised to the description of some particular kinds of phenomena such as membrane interactions or protein interactions.

P-Systems [29] are a biologically inspired computational model. Later on, through the introduction of ad-hoc features, they have been applied to describe and analyse biological systems [30, 31]. They have a simple notation and are not specialised 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 in the description of new activities observed on membranes without extending the formalism to model such activities.

Danos and Laneve [26] 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 [1] has no explicit way to model protein domains (however they can be encoded, and a variant with explicit binding has been defined in [32]), 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 permits a more natural description of membrane interactions. Another feature lacking in other formalisms is the capability 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 the modeller to naturally operate over proteins or DNA fragments which should be frequently defined as ordered sequences of elements.

5.2. Stochastic Models

Among stochastic process algebras we would like to mention the stochastic extension of the π-calculus, given by Priami et al. in [33], and the PEPA framework proposed by Hillston in [34]. We also would like to compare our work with two closer ones, namely [5] and [35].

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 [9] and in BioSPi [33]. 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 [10], 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. Other models based on bigraphs and specifically designed for biological systems can be found in [36, 37].

A stochastic semantics for CLS (SCLS) has been defined in [5]. Such a semantics, generalises the "mass-action law" to patterns containing variables and sequences. The rate of a transition is computed by resorting to a complete counting mechanism to detect all the possible occurrences of patterns within a term that, once the rule is applied, produce the same term. E.g., consider the case in which we want to apply with the generalisation of the "mass-action law" given in [5] the rule $(a \cdot \widetilde{x})^L \rfloor X \mapsto (b \cdot \widetilde{x})^L \rfloor X$ with rate k:

1. if the rule is applied to the term $(a \cdot c \cdot a \cdot c)^L \rfloor \epsilon$ the kinetic constant of the rule should be $2 \times k$ since the 2 matches of the pattern in the left-hand-side of the rule with the term are such that the corresponding reductions produce terms congruent to $(b \cdot c \cdot a \cdot c)^L \rfloor \epsilon$, instead

2. if the rule is applied to the term $(a \cdot c \cdot a \cdot d)^L \rfloor \epsilon$ the kinetic constant of the rule should be k, since in this case the two reductions produce the two terms $(b \cdot c \cdot a \cdot d)^L \rfloor \epsilon$ and $(a \cdot c \cdot b \cdot d)^L \rfloor \epsilon$ that are not congruent, and therefore do not express the same reaction.

Comparing the SCLS calculus with our calculus we note that, with our counting mechanism based on types, we abstract sequences with the multiset of the types of their elements, and loose the information on the ordering of the elements. Therefore, we cannot define a function computing correctly the kinetic constant for this example, since the function should depend on the number of a's that in both cases is the same. However, as shown in Section 3.1.1 for a restricted set of terms we can correctly realise the "mass-action law". A final difference between SCLS and our calculus is in the definition of the patterns, and therefore of rule schemata. As discussed in Remark 3.2, the hole of our contexts encompass a whole compartment, so as we can see from the examples, the patterns in the left-hand-side and right-hand-side of a rule have a subterm $\rfloor X$ which is not needed in the SCLS calculus since the hole of a context may be a subterm of a compartment.

However, there are several advantages to our framework. Firstly our counting mechanism, based on types, is simpler then the one of SCLS in practice: while a single pattern may have several, though isomorphic, matches within a CLS term, in Typed Stochastic CLS we state explicitly the types of the elements whose occurrences affect the speed of a reduction. This has simplified the development of our automatic simulation tool. Observe that, the simulator available at `http://www.di.unipi.it/msvbio/wiki/sclsm`, which was developed for the SCLS calculus, for efficiency reasons, does not implement the complex counting of matches defined in [5], but computes the kinetic constant of a reduction by counting the number of matches based on the occurrences of the elements of the pattern present in the term, as we described in Section 3.1.1. The differences between the functionalities of the semantics presented in [5] and the one implemented in the SCLS simulator make not significant a comparison of the efficiency of the two simulators. As another advantage, our rules, similar to what happens in [35] for a variant of the ambient calculus, are equipped with rate functions, rather than with rate constants. Such functions allow us to define kinetics that are more complex than the standard mass-action ones.

Bioambients [28] 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 [35], Bioambients are extended by permitting the rates associated with rules to be context dependent. Dependency is realised 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 synthesises (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 [35]. However, our modelling is more general allowing us 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 neighbourhood relationship induced by the topology of the collection. It is possible to specify stochastic rewrite strategies. In [38], 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 MGS, the topological changes are programmed in some external language, whereas in CLS they are specified directly by the rewrite rules.

5.3. Type Systems

In the last few years there has been a growing interest on the use of type disciplines to enforce biological properties. In [17, 39] a type system has been defined to ensure the well-formedness of links between protein sites within the Linked Calculus of Looping Sequences (see [32]). In [40] three type systems are defined for the Biochemical Abstract Machine, BIOCHAM (see [41]). The first one is used to infer the functions of proteins in a reaction model, the second one to infer activation and inhibition effects of proteins, and the last one to infer the topology of compartments. In [18] we have defined a type system for CLS to guarantee the soundness of reduction rules with respect to the requirement of certain elements, and the repellency of others. In [19], Bioglio generalises the previous type discipline by considering the minimum and the maximum requested numbers of elements. Finally, in [42], group types are used to regulate compartment crossing in the BioAmbients framework [28].

Acknowledgements. We thank the referees of MecBic'09 and of the present submission for their helpful comments, the present version of the paper improved due to their suggestions.

References

[1] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, A. Troina, A calculus of looping sequences for modelling microbiological systems, Fundamenta Informaticæ 72 (1–3) (2006) 21–35.

[2] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, A. Troina, Bisimulation congruences in the calculus of looping sequences, in: International Colloquium on Theoretical Aspects of Computing (ICTAC'06), Vol. 4281 of LNCS, Springer, 2006, pp. 93–107.

[3] P. Milazzo, Qualitative and quantitative formal modeling of biological systems, Ph.D. thesis, University of Pisa (2007).

[4] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, A. Troina, Bisimulations in calculi modelling membranes., Formal Aspects of Computing 20 (4-5) (2008) 351–377.

[5] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, P. Tiberi, A. Troina, Stochastic calculus of looping sequences for the modelling and simulation

of cellular pathways, Transactions on Computational Systems Biology IX (2008) 86–113.

[6] D. T. Gillespie, Exact stochastic simulation of coupled chemical reactions, Journal of Physical Chemistry 81 (25) (1977) 2340–2361.

[7] M. Dezani-Ciancaglini, P. Giannini, A. Troina, A type system for a stochastic CLS, in: Membrane Computing and Biologically Inspired Process Calculi (MeCBIC'09), Vol. 11 of EPTCS, 2009, pp. 91–106.

[8] L. Taiz, E. Zeiger, Plant Physiology, Fourth Edition, Sinauer Associated Inc., 2006.

[9] V. Danos, J. Feret, W. Fontana, J. Krivine, Scalable modelling of biological pathways, in: ASIAN Symposium on Programming Languages and Systems (APLAS'07), Vol. 4807 of LNCS, 2007, pp. 139–157.

[10] J. Krivine, R. Milner, A. Troina, Stochastic bigraphs, in: Mathematical Foundations of Programming Semantic (MFPS'08), Vol. 218 of ENTCS, Elsevier, 2008, pp. 73–96.

[11] M. Coppo, F. Damiani, M. Drocco, E. Grassi, E. Sciacca, S. Spinella, A. Troina, Hybrid calculus of wrapped compartments, in: Membrane Computing and Biologically Inspired Process Calculi (MeCBIC'10), Vol. 40 of EPTCS, 2010, pp. 102–120.

[12] M. Coppo, F. Damiani, M. Drocco, E. Grassi, M. Guether, A. Troina, Modelling ammonium transporters in arbuscular mycorrhiza symbiosis, Transactions on Computational Systems Biology XIII (2011) 85–109.

[13] N. Oury, G. Plotkin, Multi-level modelling via stochastic multi-level multiset rewriting, draft submitted to Mathematical Structures in Computer Science (2011).

[14] N. Rosenfeld, J. W. Young, U. Alon, P. S. Swain, M. B. Elowitz, Gene regulation at the single-cell level, Science 307(5717) (2007) 1962–1965.

[15] P. Wong, S. Gladney, J. D. Keasling, Mathematical model of the lac operon: Inducer exclusion, catabolite repression, and diauxic growth on glucose and lactose, Biotechnology Progress 13 (1997) 132–143.

[16] D. Wilkinson, Stochastic Modelling for Systems Biology, Chapman & Hall/CRC, 2006.

[17] B. Aman, M. Dezani-Ciancaglini, A. Troina, Type disciplines for analysing biologically relevant properties, in: Membrane Computing and Biologically Inspired Process Calculi (MeCBIC'08), Vol. 227 of ENTCS, Elsevier, 2009, pp. 97–111.

[18] M. Dezani-Ciancaglini, P. Giannini, A. Troina, A type system for required/excluded elements in CLS, in: Developments in Computational Models (DCM'09), Vol. 9 of EPTCS, 2009, pp. 38–48.

[19] L. Bioglio, Enumerated type semantics for the calculus of looping sequences, RAIRO - Theoretical Informatics and Applications 45 (2011) 35 –58.

[20] H. Matsuno, A. Doi, M. Nagasaki, S. Miyano, Hybrid Petri net representation of gene regulatory networks, in: Pacific Symposium on Biocomputing (PSB'00), World Scientific Press, 2000, pp. 341–352.

[21] R. Alur, C. Belta, V. Kumar, M. Mintz, Hybrid modeling and simulation of biomolecular networks, in: Hybrid Systems: Computation and Control, Vol. 2034 of LNCS, Springer, 2001, pp. 19–32.

[22] M. Curti, P. Degano, C. Priami, C. T. Baldari, Modelling biochemical pathways through enhanced π-calculus, Theoretical Computer Science 325 (1) (2004) 111–140.

[23] A. Regev, W. Silverman, E. Y. Shapiro, Representation and simulation of biochemical processes using the pi-calculus process algebra, in: Pacific Symposium on Biocomputing (PCB'01), Vol. 6, World Scientific Press, 2001, pp. 459–470.

[24] L. Cardelli, Brane calculi. Interactions of biological membranes, in: Computational Methods in Systems Biology (CMSB'04), Vol. 3082 of LNCS, Springer, 2005, pp. 257–280.

[25] N. Chabrier-Rivier, M. Chiaverini, V. Danos, F. Fages, V. Schachter, Modeling and querying biomolecular interaction networks, Theoretical Computer Science 325 (2004) 25–44.

[26] V. Danos, C. Laneve, Formal molecular biology, Theoretical Computer Science 325 (2004) 69–110.

[27] C. Priami, P. Quaglia, Beta binders for biological interactions, in: Computational Methods in Systems Biology (CMSB'04), Vol. 3082 of LNCS, 2005, pp. 20–33.

[28] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, E. Shapiro, BioAmbients: An abstraction for biological compartments, Theoretical Computer Science 325 (2004) 141–167.

[29] G. Păun, Membrane Computing. An Introduction, Springer, 2002.

[30] D. Besozzi, G. Ciobanu, A P system description of the sodium-potassium pump, in: Workshop on Membrane Computing (WMC'04), Vol. 3365 of LNCS, Springer, 2005, pp. 210–223.

[31] D. Pescini, D. Besozzi, G. Mauri, C. Zandron, Dynamical probabilistic P systems, International Journal of Foundations of Computer Science 17 (1) (2006) 183–204.

[32] R. Barbuti, A. Maggiolo-schettini, P. Milazzo, Extending the calculus of looping sequences to model protein interaction at the domain level, in: International Symposium on Bioinformatics Research and Applications (ISBRA'07), Vol. 4463 of LNBI, Springer, 2006, pp. 638–649.

[33] C. Priami, A. Regev, W. Silverman, E. Shapiro, Application of stochastic name-passing calculus to representation and simulation of molecular processes, Infomation Processing Letters 80 (1) (2001) 25–31.

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

[35] L. Bortolussi, M. Vigliotti, CoBiC: Context-dependent BioAmbient calculus, in: Quantitative Aspects of Programming Languages (QAPL'09), Vol. 253 of ENTCS, Elsevier, 2009, pp. 187–201.

[36] T. C. Damgaard, J. Krivine, A generic language for biological systems based on bigraphs, Tech. rep., IT University Technical Report Series, TR-2008-115 (2008).

[37] G. Bacci, D. Grohmann, M. Miculan, A framework for protein and membrane interactions, in: Membrane Computing and Biologically Inspired Process Calculi (MeCBIC'09), Vol. 11 of EPTCS, 2009, pp. 3–18.

[38] O. Michel, A. Spicher, J. Giavitto, Rule-based programming for integrative biological modelling – application to the modelling of the lambda phage genetic switch, Natural Computing 8 (2009) 865–889.

[39] R. Barbuti, M. Dezani-Ciancaglini, A. Maggiolo-Schettini, P. Milazzo, A. Troina, A formalism for the description of protein interaction, Fundamenta Informaticae 103 (2010) 1–29.

[40] F. Fages, S. Soliman, Abstract interpretation and types for systems biology, Theoretical Computer Science 403 (1) (2008) 52–70. doi:http://dx.doi.org/10.1016/j.tcs.2008.04.024.

[41] Biocham. available at `http://contraintes.inria.fr/BIOCHAM/`.

[42] S. Capecchi, A. Troina, Types for BioAmbients, in: From Biology To Concurrency and Back, Vol. 19 of EPTCS, 2010, pp. 103–115.