Stochastic Calculus of Looping Sequences for the Modelling and Simulation of Cellular Pathways

Roberto Barbuti[1], Andrea Maggiolo–Schettini[1], Paolo Milazzo[1], Paolo Tiberi[1], and Angelo Troina[2]

[1] Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo 3, 56127 - Pisa, Italy
[2] Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 - Torino Italy

Abstract. The paper presents the Stochastic Calculus of Looping Sequences (SCLS) suitable to describe microbiological systems, such as cellular pathways, and their evolution. Systems are represented by terms. The terms of the calculus are constructed by basic constituent elements and operators of sequencing, looping, containment and parallel composition. The looping operator allows tying up the ends of a sequence, thus creating a circular sequence which can represent a membrane.
The evolution of a term is modelled by a set of rewrite rules enriched with stochastic rates representing the speed of the activities described by the rules, and can be simulated automatically.
As applications, we give SCLS representations of the regulation process of the lactose operon in *Escherichia coli* and of the quorum sensing in *Pseudomonas aeruginosa*.
A prototype simulator (SCLSm) has been implemented in F# and used to run the experiments. A public version of the tool is available at the url: `http://www.di.unipi.it/~milazzo/biosims/`.

1 Introduction

Biologists usually describe biological systems by mathematical means, such as differential equations. This allows them to reason on the behaviour of the described systems and to perform simulations. Mathematical modelling becomes more difficult both in specification and in analysis when the complexity of the system increases. This is one of the main motivations for the application of Computer Science formalisms to the description of biological systems [27]. Another motivation is that the use of formal means of Computer Science permits the application of analysis methods that are practically unknown to biologists, such as model checking.

Among the formalisms that either have been applied to or have been inspired by biological systems there are automata-based models [1, 19], rewrite systems [12, 21], and process calculi [27, 25, 7]. Automata have the advantage of allowing the direct use of many verification tools such as model checkers. Rewrite systems usually allow describing biological systems with a notation that can be easily

understood by biologists. On the other hand, automata-like models and rewrite systems present, in general, problems from the point of view of compositionality. Compositionality allows studying the behaviour of a system componentwise, and is in general ensured by process calculi, included those commonly used to describe biological systems.

In [4, 5, 20] we developed a new formalism, called Calculus of Looping Sequences (CLS for short), for describing biological systems and their evolution. CLS is based on term rewriting with some features, such as a commutative parallel composition operator, and some semantic means, such as bisimulations [5, 6], 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.

In this paper we focus on quantitative aspects of our formalism, in particular, to model speed of activities, we develop a stochastic extension of CLS (called SCLS). 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 [14].

We have developed a prototype simulator for SCLS. To show the expressiveness of our formalism, we model and simulate two examples: the regulation of the lactose operon in *Escherichia coli* and the quorum sensing in *Pseudomonas aeruginosa*. The first example shows all the features of SCLS used to describe a classical model. The second one shows the merit of the computational approach with respect to mathematical modelling when the complexity of the system increases.

We remark that the contribution of this paper is not in the simulation algorithm, inspired by the standard Gillespie's algorithm, but in the language proposed to describe systems: it allows describing cellular structures and compartments, and this simplifies the modelling of a cell as a system whose components are described individually.

1.1 Summary

The remainder of this paper is organized as follows. In Section 2 we formally recall the Calculus of Looping Sequence and we give some guidelines for the modelling of biological systems. In Section 3 we introduce our stochastic extension. In Sections 4 and 5 we use the stochastic framework to model and analyse two different applications; namely, we model the lactose operon of *Escherichia Coli* and a quorum sensing process in *Pseudomonas aeruginosa*. Finally, in Section 6 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). It is based on term rewriting, and hence a CLS model consists of a term and a set of rewrite

rules. The term represents the structure of the modelled system, and the rewrite rules represent its evolution.

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

Definition 1 (Terms). Terms T *and* Sequences S *of CLS are given by the grammars:*

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

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

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 the definition of terms, we have that looping and containment are always applied together, hence we can consider them as a single binary operator $(\_)^L \rfloor \_$ that 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 closed surfaces often interspersed with proteins, and may have a content. 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 what the membrane contains. Other macro-molecules can be modelled as single alphabet symbols, or as sequences of their components. Finally, juxtaposition of entities can be described by the parallel composition operator of their representations. A deeper description of the biological interpretation of CLS operators together with some modelling guidelines will be given in Section 2.1.

Brackets can be used to indicate the order of application of the operators, and $(\_)^L \rfloor \_$ has the precedence over $\_\,|\,\_$. An example of CLS term is $a \,|\, b \,|\, (m \cdot n)^L \rfloor (c \cdot d \,|\, e)$. It represents a membrane with two molecules m and n (for instance, two proteins) on its surface, and containing a sequence $c \cdot d$ (for instance, a DNA

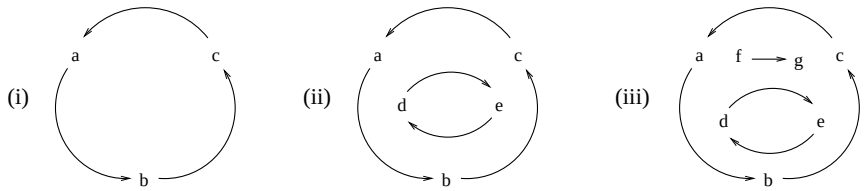


Fig. 1. (i) represents $(a{\cdot}b{\cdot}c)^L \rfloor \epsilon$; (ii) represents $(a{\cdot}b{\cdot}c)^L \rfloor (d{\cdot}e)^L \rfloor \epsilon$; (iii) represents $(a{\cdot}b{\cdot}c)^L \rfloor (((d{\cdot}e)^L \rfloor \epsilon) \,|\, f{\cdot}g)$.

strand) and a molecule e. Molecules a and b are outside the membrane. See Figure 1 for some graphical representations.

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

Definition 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:*

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

Rules of the structural congruence state the associativity of $\cdot$ and $|$, 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 we will use $\equiv$ in place of $\equiv_T$.

Rewrite rules are defined essentially as pairs of terms, in which the first term describes the portion of the system in which the event modelled by the rule may occur, and the second term describes 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 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 TV ranged over by $X, Y, Z, \ldots$, a set of sequence variables 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 pairwise disjoint and possibly infinite. We denote by $\mathcal{V}$ the set of all variables $TV \cup SV \cup \mathcal{X}$, and with ρ any variable in $\mathcal{V}$. A pattern is a term which may include variables.

Definition 3 (Patterns). Patterns P *and* sequence patterns SP *of CLS are given by the following grammar:*

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

where a is an element of $\mathcal{E}$, and $X, \widetilde{x}$ and x are elements of TV, 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 $X \in TV, \widetilde{x} \in SV$ 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 $\rho \in \mathcal{V}$ appearing in P with the corresponding term $\sigma(\rho)$. 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 can define rewrite rules.

Definition 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 $P_2\sigma$. We define the semantics of CLS as a transition system, in which states correspond to terms, and transitions correspond to rule applications. The semantics of CLS is defined by resorting to the notion of contexts.

Definition 5 (Contexts). Contexts C *are defined as:*

$$C ::= \Box \;\mid\; C \,|\, T \;\mid\; T \,|\, C \;\mid\; (S)^{L} \rfloor C$$

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

By definition, every context contains a single $\Box$. Let us assume $C, C' \in \mathcal{C}$. With $C[T]$ we denote the term obtained by replacing $\Box$ with T in C; with $C[C']$ we denote context composition, whose result is the context obtained by replacing $\Box$ with C' in C. The structural congruence relation can be easily extended to contexts, namely $C \equiv C'$ if and only if $C[\epsilon] \equiv C'[\epsilon]$.

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.

Biomolecular Entity	CLS Term
Elementary object (genes, domains, other molecules, etc...)	Alphabet symbol
DNA strand	Sequence of elements repr. genes
RNA strand	Sequence of elements repr. transcribed genes
Protein	Sequence of elements repr. domains or single alphabet symbol
Molecular population	Parallel composition of molecules
Membrane	Looping sequence

Table 1. Guidelines for the abstraction of biomolecular entities into CLS.

Definition 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 inference rule:*

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

2.1 Modelling Guidelines

We describe how CLS can be used to model biomolecular systems analogously to what done by Regev and Shapiro in [28] for the π-calculus. An abstraction is a mapping from a real-world domain to a mathematical domain, which may allow highlighting some essential properties of a system while ignoring other, complicating, ones. In [28], Regev and Shapiro show how to abstract biomolecular systems as concurrent computations by identifying the biomolecular entities and events of interest and by associating them with concepts of concurrent computations such as concurrent processes and communications. In particular, they give some guidelines for the abstraction of biomolecular systems to the π-calculus, and give some simple examples.

The use of rewrite systems, such as CLS, to describe biological systems is founded on a different abstraction. Usually, entities (and their structures) are abstracted by terms of the rewrite system, and events by rewrite rules. We have already introduced the biological interpretation of CLS operators in the previous section. Here we want to give more general guidelines.

First of all, we should select the biomolecular entities of interest. Since we want to describe cells, we consider molecular populations and membranes. Molecular populations are groups of molecules that are in the same compartment of the cell. As we have said before, molecules can be of many types: we classify them as DNA and RNA strands, proteins, and other molecules. Membranes are considered as elementary objects, in the sense that we do not describe them at the level of the lipids they are made of. The only interesting properties of a membrane are that it may have a content (hence, create a compartment) and that it may have molecules on its surface.

Biomolecular Event	**Examples of CLS Rewrite Rule**
State change	$a \mapsto b$ $\widetilde{x} \cdot a \cdot \widetilde{y} \mapsto \widetilde{x} \cdot b \cdot \widetilde{y}$
Complexation	$a \mid b \mapsto c$ $\widetilde{x} \cdot a \cdot \widetilde{y} \mid b \mapsto \widetilde{x} \cdot c \cdot \widetilde{y}$
Decomplexation	$c \mapsto a \mid b$ $\widetilde{x} \cdot c \cdot \widetilde{y} \mapsto \widetilde{x} \cdot a \cdot \widetilde{y} \mid b$
Catalysis	$c \mid P_1 \mapsto c \mid P_2$ where $P_1 \mapsto P_2$ is the catalyzed event
State change on membrane	$(a \cdot \widetilde{x})^L \rfloor X \mapsto (b \cdot \widetilde{x})^L \rfloor X$
Complexation on membrane	$(a \cdot \widetilde{x} \cdot b \cdot \widetilde{y})^L \rfloor X \mapsto (c \cdot \widetilde{x} \cdot \widetilde{y})^L \rfloor X$ $a \mid (b \cdot \widetilde{x})^L \rfloor X \mapsto (c \cdot \widetilde{x})^L \rfloor X$ $(b \cdot \widetilde{x})^L \rfloor (a \mid X) \mapsto (c \cdot \widetilde{x})^L \rfloor X$
Decomplexation on membrane	$(c \cdot \widetilde{x})^L \rfloor X \mapsto (a \cdot b \cdot \widetilde{x})^L \rfloor X$ $(c \cdot \widetilde{x})^L \rfloor X \mapsto a \mid (b \cdot \widetilde{x})^L \rfloor X$ $(c \cdot \widetilde{x})^L \rfloor X \mapsto (b \cdot \widetilde{x})^L \rfloor (a \mid X)$
Catalysis on membrane	$(c \cdot \widetilde{x} \cdot SP_1 \cdot \widetilde{y})^L \rfloor X \mapsto (c \cdot \widetilde{x} \cdot SP_2 \cdot \widetilde{y})^L \rfloor X$ where $SP_1 \mapsto SP_2$ is the catalyzed event
Membrane crossing	$a \mid (\widetilde{x})^L \rfloor X \mapsto (\widetilde{x})^L \rfloor (a \mid X)$ $(\widetilde{x})^L \rfloor (a \mid X) \mapsto a \mid (\widetilde{x})^L \rfloor X$ $\widetilde{x} \cdot a \cdot \widetilde{y} \mid (\widetilde{z})^L \rfloor X \mapsto (\widetilde{z})^L \rfloor (\widetilde{x} \cdot a \cdot \widetilde{y} \mid X)$ $(\widetilde{z})^L \rfloor (\widetilde{x} \cdot a \cdot \widetilde{y} \mid X) \mapsto \widetilde{x} \cdot a \cdot \widetilde{y} \mid (\widetilde{z})^L \rfloor X$
Catalyzed membrane crossing	$a \mid (b \cdot \widetilde{x})^L \rfloor X \mapsto (b \cdot \widetilde{x})^L \rfloor (a \mid X)$ $(b \cdot \widetilde{x})^L \rfloor (a \mid X) \mapsto a \mid (b \cdot \widetilde{x})^L \rfloor X$ $\widetilde{x} \cdot a \cdot \widetilde{y} \mid (b \cdot \widetilde{z})^L \rfloor X \mapsto (b \cdot \widetilde{z})^L \rfloor (\widetilde{x} \cdot a \cdot \widetilde{y} \mid X)$ $(b \cdot \widetilde{z})^L \rfloor (\widetilde{x} \cdot a \cdot \widetilde{y} \mid X) \mapsto \widetilde{x} \cdot a \cdot \widetilde{y} \mid (b \cdot \widetilde{z})^L \rfloor X$
Membrane joining	$(\widetilde{x})^L \rfloor (a \mid X) \mapsto (a \cdot \widetilde{x})^L \rfloor X$ $(\widetilde{x})^L \rfloor (\widetilde{y} \cdot a \cdot \widetilde{z} \mid X) \mapsto (\widetilde{y} \cdot a \cdot \widetilde{z} \cdot \widetilde{x})^L \rfloor X$
Catalyzed membrane joining	$(b \cdot \widetilde{x})^L \rfloor (a \mid X) \mapsto (a \cdot b \cdot \widetilde{x})^L \rfloor X$ $(\widetilde{x})^L \rfloor (a \mid b \mid X) \mapsto (a \cdot \widetilde{x})^L \rfloor (b \mid X)$ $(b \cdot \widetilde{x})^L \rfloor (\widetilde{y} \cdot a \cdot \widetilde{z} \mid X) \mapsto (\widetilde{y} \cdot a \cdot \widetilde{z} \cdot \widetilde{x})^L \rfloor X$ $(\widetilde{x})^L \rfloor (\widetilde{y} \cdot a \cdot \widetilde{z} \mid b \mid X) \mapsto (\widetilde{y} \cdot a \cdot \widetilde{z} \cdot \widetilde{x})^L \rfloor (b \mid X)$
Membrane fusion	$(\widetilde{x})^L \rfloor (X) \mid (\widetilde{y})^L \rfloor (Y) \mapsto (\widetilde{x} \cdot \widetilde{y})^L \rfloor (X \mid Y)$
Catalyzed membrane fusion	$(a \cdot \widetilde{x})^L \rfloor (X) \mid (b \cdot \widetilde{y})^L \rfloor (Y) \mapsto$ $(a \cdot \widetilde{x} \cdot b \cdot \widetilde{y})^L \rfloor (X \mid Y)$
Membrane division	$(\widetilde{x} \cdot \widetilde{y})^L \rfloor (X \mid Y) \mapsto (\widetilde{x})^L \rfloor (X) \mid (\widetilde{y})^L \rfloor (Y)$
Catalyzed membrane division	$(a \cdot \widetilde{x} \cdot b \cdot \widetilde{y})^L \rfloor (X \mid Y) \mapsto$ $(a \cdot \widetilde{x})^L \rfloor (X) \mid (b \cdot \widetilde{y})^L \rfloor (Y)$

Table 2. Guidelines for the abstraction of biomolecular events into CLS.

Now, we select the biomolecular events of interest. The simplest kind of event is the change of state of an elementary object. Then, we may have interactions between molecules: in particular complexation, decomplexation and catalysis. These interactions may involve single elements of non-elementary molecules (DNA and RNA strands, and proteins). Moreover, we may have interactions between membranes and molecules: in particular a molecule may cross or join a membrane. Finally, we may have interactions between membranes: in this case there may be many kinds of interactions (fusion, division, etc. . .).

The guidelines for the abstraction of biomolecular entities and events into CLS are given in Table 1 and Table 2, respectively. Entities are associated with CLS terms: elementary objects are modelled as alphabet symbols, non-elementary objects as CLS sequences and membranes as looping sequences. Biomolecular events are associated with CLS rewrite rules. In the table we give some examples of rewrite rules for each type of event. The list of examples is not complete: one could imagine also rewrite rules for the description of complexation/decomplexation events involving more than two molecules, or catalysis events in which the catalyzing molecule is on a membrane and the catalyzed event occurs in its content, or more complex interactions between membranes. We remark that in the second example of rewrite rule associated with the complexation event we have that one of the two molecules which are involved should be either an elementary object or a protein modelled as a single alphabet symbol. As before, this is caused by the problem of modelling protein interaction at the domain level. This problem is solved by an extension of CLS where links are considered. Such a model, called LCLS, is formalized in [3].

3 The Stochastic Calculus of Looping Sequences

The standard way of extending a formalism to model quantitative aspects of biological systems is by incorporating a collision-based stochastic framework on the lines of the one presented by Gillespie in [14]. Following the law of mass action, we need to count the number of reactants that are present in a system in order to compute the exact rate of a reaction. This has been done, for instance, for the π-calculus [23, 25]. The idea of Gillespie's algorithm is that a rate constant is associated with each considered chemical reaction. Such a constant is obtained by multiplying the kinetic constant of the reaction by the number of possible combinations of reactants that may occur in the system. The resulting rate is then used as the parameter of an exponential distribution modelling the time spent between two occurrences of the considered chemical reaction.

The use of exponential distributions to represent the (stochastic) time spent between two occurrences of chemical reactions allows describing the system as a Continuous Time Markov Chain (CTMC), and consequently allows verifying properties of the described system analytically and by means of stochastic model checkers.

We start by adding rates to rewrite rules.

Definition 7 (Stochastic Rewrite Rule). *A* stochastic rewrite rule *is a triple* (P_1, P_2, k)*, denoted with* $P_1 \overset{k}{\mapsto} P_2$*, where* $P_1, P_2 \in \mathcal{P}$*,* $P_1 \not\equiv \epsilon$ *and such that* $Var(P_2) \subseteq Var(P_1)$*;* $k \in \mathbb{R}^{\geq 0}$ *is the* rewrite rate*.*

To describe the evolution of a term, the stochastic semantics must consider, besides the rate of a rule, also the number of occurrences of subterms to which the rule can be applied and the terms produced. Subterms to which the rule can be applied correspond to reactants in a biological system. In what follows, a *subterm* of a term T will be a term $T' \not\equiv \epsilon$ for which a context C exists such that $T \equiv C[T']$, and a *reactant* will be an occurrence in T of a subterm.

Example 1. If $T = a \,|\, a \,|\, b \,|\, b$, then the set of subterms of T is

$$\{a\ ,\ b\ ,\ a \,|\, a\ ,\ a \,|\, b\ ,\ b \,|\, b\ ,\ a \,|\, a \,|\, b\ ,\ a \,|\, b \,|\, b\ ,\ T\}$$

while

$$\{a\ ,\ a\ ,\ b\ ,\ b\ ,\ a \,|\, a\ ,\ a \,|\, b\ ,\ a \,|\, b\ ,\ a \,|\, b\ ,\ a \,|\, b\ ,\ b \,|\, b\ ,\ a \,|\, a \,|\, b\ ,\ a \,|\, a \,|\, b\ ,\ a \,|\, b \,|\, b\ ,\ a \,|\, b \,|\, b\ ,\ T\}$$

is the multiset of reactants in T. □

Now, defining the stochastic semantics would be easy if rules would contain no variables. For instance, if we have the rewrite rule $a \,|\, b \overset{k}{\mapsto} c$, where k is the kinetic constant of the modelled chemical reaction, then its application rate is k multiplied by the number of possible combinations of occurrences of a and b in the term, namely the number of occurrences of $a \,|\, b$ in the multiset of reactants of the term. For example, given the term T in Example 1, we have two occurrences of a and two of b, hence the number of possible combinations of reactants is $2 \times 2 = 4$, and this holds also in the multiset of reactants of T, which contains four instances of $a \,|\, b$.

As we have variables, we have to take into account how they can be instantiated in order to compute the application rate of the rewrite rule. Variables allow a rewrite rule to stand for a family of ground rules, which represents a family of chemical reactions. Moreover, it often happens that the application rate of a rewrite rule depends on how many molecules of some kind are contained in the part of the system represented by a variable. For instance, consider a rule such as $a \,|\, (b{\cdot}\widetilde{x})^L \rfloor X \overset{k}{\mapsto} (c{\cdot}\widetilde{x})^L \rfloor X$, representing the binding of molecule a with an instance of b (resulting into the product molecule c) placed on the membrane represented by the looping sequence. We should have that the application rate of the derived reactions is proportional to the number of b which are present on the membrane, that is the number of b in the instantiation of the variable $\widetilde{x}$ plus one.

We remark that this problem has not been faced during the development of the stochastic extension of other formalisms such as the π-calculus, as those formalisms are not able to model chemical reactions with variables (as CLS patterns are). Also Gillespie's work does not deal with variables in the simulated

chemical reactions. As a consequence, we have to give a reasonable interpretation to rewrite rules with variables.

We follow an approach on the lines of the one used by Krivine et al. in [17] for defining a stochastic semantics for Bigraphical Reactive Systems. The technique they have developed to count the occurrences of a reactant is based on the definition of *abstract* and *concrete* bigraphs. Here we consider as *abstract* the CLS terms and patterns defined as in Definitions 1 and 3. In the remainder of this section, we denote abstract terms and patterns using a tilde, as in $\tilde{T}$ and $\tilde{P}$. Now, we give the definition of *concrete* CLS patterns. Since a term is a ground pattern, the analogous definition for concrete terms can be inherited from the one for patterns.

Definition 8 (Concrete patterns and terms). *If $\tilde{P}$ is an abstract pattern, then a* concrete *pattern P, called a* concretion *of $\tilde{P}$, is obtained by assigning to each alphabet symbol syntactically appearing in $\tilde{P}$ a unique identifier $v \in Id$, where Id is a finite set of identifiers. With $\grave{\mathcal{P}}$ and $\grave{\mathcal{T}}$ we denote the sets of concrete patterns and terms, respectively.*

Intuitively, each symbol of the alphabet $\mathcal{E}$ appearing in patterns and terms, becomes unique in the concretion by labelling it with a fresh identifier. Moreover, we equip concrete patterns and terms with a notion of *support*.

Definition 9 (Support). *Given a concrete pattern P, we call* support *the set of identifiers used to label its alphabet symbols and we denote it with $Supp(P)$.*

Two concrete patterns P and P' are support-equivalent*, written $P \doteq P'$, if they differ only by a bijection between their supports which preserves structure. Namely, $P \doteq P'$ if and only if $\tilde{P} \equiv \tilde{P}'$ and there exists a bijection between $Supp(P)$ and $Supp(P')$. We denote the $\doteq$-equivalence class of P by $[\![P]\!]$.*

As before, the analogous definitions of support and support-equivalence for concrete terms is inherited.

Given an abstract pattern $\tilde{P} = a\widetilde{x} \mid (a \cdot b)^L \rfloor X \in \mathcal{P}$ concretions of $\tilde{P}$ are $P_1 = a_{v_1} \cdot \widetilde{x} \mid (a_{v_2} \cdot b_{v_3})^L \rfloor X$ or $P_2 = a_{u_1} \cdot \widetilde{x} \mid (a_{u_2} \cdot b_{u_3})^L \rfloor X$ with supports $Supp(P_1) = \{v_1, v_2, v_3\}$ and $Supp(P_2) = \{u_1, u_2, u_3\}$. Note that for an abstract pattern $\tilde{P}$ and a set of identifiers Id there exist many different concretions. For the case above we have, however, $P_1 \doteq P_2$.

We can extend the definition of concrete patterns to stochastic rewrite rules.

Definition 10. *If $R = (\tilde{P}_1, \tilde{P}_2, k)$ is a stochastic rewrite rule, then (P_1, P_2, k) is called a concretion of R.*

The definition of contexts is extended to deal with concrete terms in the natural way, with $\grave{\mathcal{C}}$ we denote the set of concrete contexts. Without loss of generality, we assume instantiations to return abstract or concrete terms when applied to abstract or concrete patterns respectively. Namely, given $\tilde{P} \in \mathcal{P}$, $\tilde{P}\sigma \in \mathcal{T}$, while given $P \in \grave{\mathcal{P}}$, $P\sigma \in \grave{\mathcal{T}}$.

Since we would like to define rewrite rules in an abstract way, we should define a notion of occurrence of abstract patterns within a term.

Definition 11 (Occurrences). *If $\tilde{P} \in \mathcal{P}$ is an abstract pattern and $T \in \grave{\mathcal{T}}$ a concrete term, an* occurrence *of $\tilde{P}$ in T is a pair (C, P), where $P \in \grave{\mathcal{P}}$ is a concretion of $\tilde{P}$ and $C \in \grave{\mathcal{C}}$ is a context such that $T \equiv C[P\sigma]$ for some instantiation σ.*

An occurrence of a rule $R = (\tilde{P}_1, \tilde{P}_2, k)$ *in a concrete term T is a pair (C, P_1), where (P_1, P_2, k) is a concretion of R and $T \equiv C[P_1\sigma]$ for some instantiation σ. If also $T' \backsimeq C[P_2\sigma]$ we say that the occurrence of rule R in T results into a term support-equivalent to T'. With $\mathcal{O}(R, T, [\![T']\!])$ we define the set of occurrences of rule R in the term T resulting in a concrete term support-equivalent to T'.*

Example 2. Consider a concretion $T = a_{v_1} \,|\, a_{v_2} \,|\, b_{v_3} \,|\, b_{v_4}$ of the term in Example 1 and the abstract stochastic rewrite rule $R = a \,|\, b \overset{k}{\mapsto} c$. The occurrences of R in T are:

- $(a_{v_1} \,|\, b_{v_3} \,|\, \square \,,\, a_{v_2} \,|\, b_{v_4})$;
- $(a_{v_1} \,|\, b_{v_4} \,|\, \square \,,\, a_{v_2} \,|\, b_{v_3})$;
- $(a_{v_2} \,|\, b_{v_3} \,|\, \square \,,\, a_{v_1} \,|\, b_{v_4})$;
- $(a_{v_2} \,|\, b_{v_4} \,|\, \square \,,\, a_{v_1} \,|\, b_{v_3})$.

Moreover, all these occurrences result into a concrete term which is support-equivalent to $T' = a_{v_1} \,|\, c_{t_1} \,|\, b_{v_3}$. Thus, $\mathcal{O}(R, T, [\![T']\!])$ contains exactly the four occurrences listed above.

If we consider a term $T_o = a_{u_0} \,|\, a_{u_1} \,|\, (b_{u_2} \cdot c_{u_3} \cdot b_{u_4} \cdot a_{u_5})^L \rfloor \epsilon$ and the abstract rule $R' = a \,|\, (b \cdot \widetilde{x})^L \rfloor X \overset{k'}{\mapsto} (c \cdot \widetilde{x})^L \rfloor X$ (which contains variables), then the occurrences of R' in T_o are:

- $o_1 = (a_{u_1} \,|\, \square \,,\, a_{u_0} \,|\, (b_{u_2} \cdot \widetilde{x})^L \rfloor X)$, for $\sigma(\widetilde{x}) = c_{u_3} \cdot b_{u_4} \cdot a_{u_5}$ and $\sigma(X) = \epsilon$;
- $o_2 = (a_{u_1} \,|\, \square \,,\, a_{u_0} \,|\, (b_{u_4} \cdot \widetilde{x})^L \rfloor X)$, for $\sigma(\widetilde{x}) = a_{u_5} \cdot b_{u_2} \cdot c_{u_3}$ and $\sigma(X) = \epsilon$;
- $o_3 = (a_{u_0} \,|\, \square \,,\, a_{u_1} \,|\, (b_{u_2} \cdot \widetilde{x})^L \rfloor X)$, for $\sigma(\widetilde{x}) = c_{u_3} \cdot b_{u_4} \cdot a_{u_5}$ and $\sigma(X) = \epsilon$;
- $o_4 = (a_{u_0} \,|\, \square \,,\, a_{u_1} \,|\, (b_{u_4} \cdot \widetilde{x})^L \rfloor X)$, for $\sigma(\widetilde{x}) = a_{u_5} \cdot b_{u_2} \cdot c_{u_3}$ and $\sigma(X) = \epsilon$.

Note that these occurrences take into account all the possible combinations of any molecule a (outside the looping sequence) with any molecule b (in the looping sequence).

In this case, the different occurrences of the rule produce terms which are also structurally different. For example, by applying the first occurrence with the concretion of the right hand side of R' given by $P_2 = (c_{t_1} \cdot \widetilde{x})^L \rfloor X$, we get $T'_{o_1} = a_{u_1} \,|\, (c_{t_1} \cdot c_{u_3} \cdot b_{u_4} \cdot a_{u_5})^L \rfloor \epsilon$. Differently, if we apply the second occurrence, again with the same concretion $P_2 = (c_{t_1} \cdot \widetilde{x})^L \rfloor X$, we get $T'_{o_2} = a_{u_1} \,|\, (c_{t_1} \cdot a_{u_5} \cdot b_{u_2} \cdot c_{u_3})^L \rfloor \epsilon$. Note that T'_{o_1} and T'_{o_2} are structurally different: in T'_{o_1}, the a-molecule remaining in the looping sequence is followed by a c-molecule; in T'_{o_2}, the a-molecule in the looping sequence is followed by a b-molecule. Thus $T'_{o_1} \neq T'_{o_2}$. However, if we compute the terms T'_{o_3} and T'_{o_4}, by applying the third and fourth occurrence respectively, with the same P_2, then we get $T'_{o_1} \backsimeq T'_{o_3}$ and $T'_{o_2} \backsimeq T'_{o_4}$. Thus, we obtain the following sets: $\mathcal{O}(R', T_o, [\![T'_{o_1}]\!]) = \{o_1, o_3\}$ and $\mathcal{O}(R', T_o, [\![T'_{o_2}]\!]) = \{o_2, o_4\}$.

The use of support-equivalence in the definition of $\mathcal{O}(R, T, [\![T']\!])$ allows us to consider as a single occurrence the occurrences which differ only for the support in P_2 (thus producing different, but support-equivalent, T'). As an example, in the case of T_o, the first occurrence $(a_{u_1} \mid \Box\ ,\ a_{u_0} \mid (b_{u_2} \cdot \widetilde{x})^L \rfloor X)$ can be produced by several concretions of the rule R' differing in their P_2 parts. In particular, admissible concretions for $\tilde{P}_2$ could be $P_2^1 = (c_{t_1} \cdot \widetilde{x})^L \rfloor X$, $P_2^2 = (c_{t_2} \cdot \widetilde{x})^L \rfloor X$, ... However, all of them produce a single occurrence in $\mathcal{O}(R, T_o, [\![T'_{o_1}]\!])$ since, in this case, $C[P_2^i\sigma] \simeq C[P_2^j\sigma]$ for any i and j. □

The following proposition holds.

Proposition 1. *Given an abstract rule $R = (\tilde{P}_1, \tilde{P}_2, k)$ and a concrete term T, let (C, P_1) be an occurrence of R in T, where (P_1, P_2, k) is a concrete rule generated by R. Then:*

(a) C is determined uniquely by P_1;
(b) P_2 is determined uniquely by P_1 up to support-equivalence.

With $T \xrightarrow{R} T'$ we denote a transition, driven by the rule $R = (\tilde{P}_1, \tilde{P}_2, k)$, from the concrete term T to the concrete term T'. We now associate a rate with transitions between concrete terms. The rate is obtained as the product of the rate k of the stochastic rewrite rule and the number of distinct occurrences of the rule within the term T resulting in T'.

Definition 12 (Rate of concrete transitions). *Given T, T' concrete, and an abstract reaction rule $R = (\tilde{P}_1, \tilde{P}_2, k)$, then $|\mathcal{O}(R, T, [\![T']\!])|$ is the number of distinct occurrences (C, P_1) of R in T resulting in T'. Each such occurrence is also called a* contribution *of R to the rate of $T \xrightarrow{R} T'$. The* transition rate *for $T \xrightarrow{R} T'$ is defined formally by*

$$rate_R[T, T'] \stackrel{\text{def}}{=} k \cdot |\mathcal{O}(R, T, [\![T']\!])| \ .$$

To compute the rate of an abstract transition $\tilde{T} \xrightarrow{R} \tilde{T}'$, we can just compute the rate for arbitrary concretions $T \xrightarrow{R} T'$ of that transition, because the rate is independent of the chosen concretions, i.e.:

Proposition 2. *If $T_1 \simeq T_2$ and $T'_1 \simeq T'_2$, all concrete, then, for any stochastic rewrite rule R:*

$$rate_R[T_1, T'_1] = rate_R[T_2, T'_2] \ .$$

This justifies the following definition of the abstract reaction rate.

Definition 13 (Rate of abstract transitions). *Given a stochastic rewrite rule R, the rate of an abstract transition $\tilde{T} \xrightarrow{R} \tilde{T}'$ is defined by*

$$rate_R[\tilde{T}, \tilde{T}'] \stackrel{\text{def}}{=} rate_R[T, T']$$

where T and T' are arbitrary concretions of $\tilde{T}$ and $\tilde{T}'$, respectively.

Again, an example will be helpful.

Example 3. Consider again the abstract term $\tilde{T} = a \,|\, a \,|\, b \,|\, b$, its concretion $T = a_{v_1} \,|\, a_{v_2} \,|\, b_{v_3} \,|\, b_{v_4}$ and the rule $R = a \,|\, b \overset{k}{\mapsto} c$. In Example 2 we have defined the set of occurrences $\mathcal{O}(R, T, [\![T']\!])$ which contains exactly four elements, thus $rate_R[T, T'] = k \cdot 4$. As a consequence, for the abstract term $\tilde{T}' = a \,|\, c \,|\, b$ we obtain $rate_R[\tilde{T}, \tilde{T}'] = k \cdot 4$.

Similarly, for the abstract term $\tilde{T}_o = a \,|\, a \,|\, (b{\cdot}c{\cdot}b{\cdot}a)^L \rfloor \epsilon$, given its concretion T_o and the stochastic rewrite rule R' defined in Example 2, we get the following:

- $\tilde{T}'_{o_1} = a \,|\, (c{\cdot}c{\cdot}b{\cdot}a)^L \rfloor \epsilon$ is the abstraction of T'_{o_1};
- $\tilde{T}'_{o_2} = a \,|\, (c{\cdot}a{\cdot}b{\cdot}c)^L \rfloor \epsilon$ is the abstraction of T'_{o_2}.

We can then derive the following rates: $rate_{R'}[\tilde{T}_o, \tilde{T}'_{o_1}] = rate_{R'}[\tilde{T}_o, \tilde{T}'_{o_2}] = k' \cdot 2$ since $|\mathcal{O}(R', T_o, [\![T'_{o_1}]\!])| = |\mathcal{O}(R', T_o, [\![T'_{o_2}]\!])| = 2$. □

We can now unroll the definitions of occurrences and transition rates to get the semantics of SCLS. The stochastic transition system for abstract terms is defined as follows.

Definition 14 (Semantics). *Given a finite set $\mathcal{R}$ of stochastic rewrite rules, the* semantics *of SCLS is the least labelled transition relation satisfying the following rule:*

$$\frac{R = \tilde{P}_1 \overset{k}{\mapsto} \tilde{P}_2 \in \mathcal{R} \quad (C, P_1) \in \mathcal{O}(R, T, [\![T']\!]) \quad T \equiv C[P_1\sigma] \quad T' \backsimeq C[P_2\sigma]}{\tilde{T} \xrightarrow{R, k \cdot |\mathcal{O}(R,T,[\![T']\!])|} \tilde{T}'}$$

The stochastic reduction semantics associates with each transition a rate which is the parameter of an exponential distribution that characterizes the stochastic behaviour of the activity corresponding to the applied rewrite rule. As we have already noticed, the rate is obtained as the product of the rewrite rate constant and the number of occurrences of the rule within the starting term (thus counting the exact number of reactants to which the rule can be applied and which produce the same result).

Our stochastic semantics is essentially a *Continuous Time Markov Chain* (CTMC). We can follow a standard simulation procedure that corresponds to Gillespie's simulation algorithm [14]. We have developed a prototype simulator (called SCLSm) for SCLS in the language F#.

In the next two sections we report some experimental results. We used concrete terms and patterns just for defining a consistent methodology for counting the occurrences of a rule within a term. When modelling, however, we would like to reason in an abstract way and we are not interested in concrete patterns or terms anymore. Thus, to lighten the notation in the next two sections, we resort again to the plain convention (without the tilde) to denote abstract patterns and terms.

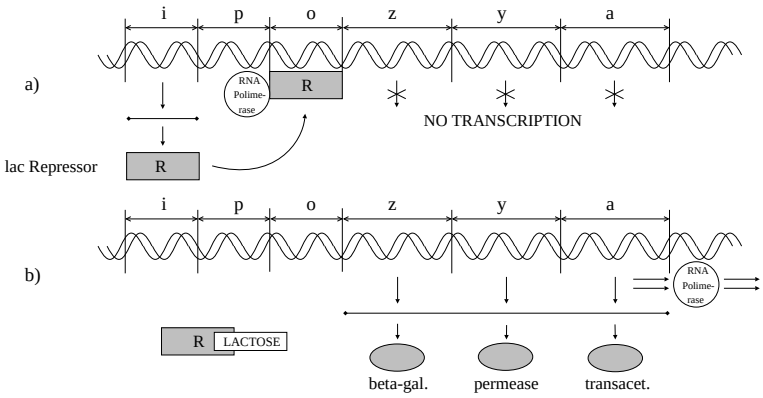


Fig. 2. The regulation process in the Lac Operon.

4 Modelling the Lactose Operon

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

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 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 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.

4.1 Stochastic CLS Model

A detailed mathematical model of the regulation process can be found in [34]. It includes information on the influence of lactose degradation on the growth of the bacterium.

We give a SCLS model of the gene regulation process, with stochastic rates taken from [33]. 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 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 $T \mid \ldots \mid T$ of length n):

$$Ecoli \;::=\; (m)^L \rfloor (lacI\text{–}A \mid 30 \times polym \mid 100 \times repr) \tag{1}$$

$$EcoliLact \;::=\; Ecoli \mid 100 \times LACT \tag{2}$$

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 rewrite rules:

$$lacI\cdot\widetilde{x} \overset{0.02}{\longmapsto} lacI\cdot\widetilde{x} \,|\, Irna \qquad Irna \overset{0.1}{\longmapsto} Irna \,|\, repr \tag{R1-R2}$$

$$polym \,|\, \widetilde{x}\cdot lacP\cdot\widetilde{y} \overset{0.1}{\longmapsto} \widetilde{x}\cdot PP\cdot\widetilde{y} \tag{R3}$$

$$\widetilde{x}\cdot PP\cdot\widetilde{y} \overset{0.01}{\longmapsto} polym \,|\, \widetilde{x}\cdot lacP\cdot\widetilde{y} \tag{R4}$$

$$\widetilde{x}\cdot PP\cdot lacO\cdot\widetilde{y} \overset{20.0}{\longmapsto} polym \,|\, Rna \,|\, \widetilde{x}\cdot lacP\cdot lacO\cdot\widetilde{y} \tag{R5}$$

$$Rna \overset{0.1}{\longmapsto} Rna \,|\, betagal \,|\, perm \,|\, transac \tag{R6}$$

$$repr \,|\, \widetilde{x}\cdot lacO\cdot\widetilde{y} \overset{1.0}{\mapsto} \widetilde{x}\cdot RO\cdot\widetilde{y} \qquad \widetilde{x}\cdot RO\cdot\widetilde{y} \overset{0.01}{\mapsto} repr \,|\, \widetilde{x}\cdot lacO\cdot\widetilde{y} \tag{R7-R8}$$

$$repr \,|\, LACT \overset{0.005}{\mapsto} RLACT \qquad RLACT \overset{0.1}{\mapsto} repr \,|\, LACT \tag{R9-R10}$$

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) and (R8) describe binding and unbinding of the lac Repressor to gene o. Finally, rules (R9) and (R10) 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:

$$(\widetilde{x})^L \rfloor (perm \,|\, X) \overset{0.1}{\mapsto} (perm\cdot\widetilde{x})^L \rfloor X \tag{R11}$$

$$LACT \,|\, (perm\cdot\widetilde{x})^L \rfloor X \overset{0.001}{\mapsto} (perm\cdot\widetilde{x})^L \rfloor (LACT|X) \tag{R12}$$

$$betagal \,|\, LACT \overset{0.001}{\mapsto} betagal \,|\, GLU \,|\, GAL \tag{R13}$$

Rule (R11) describes the incorporation of the lactose permease in the membrane of the bacterium, rule (R12) the transportation of lactose from the environment to the interior performed by the lactose permease, and rule (R13) the decomposition of the lactose into glucose (denoted GLU) and galactose (denoted GAL) performed by the beta galactosidase.

The following rules describe the degradation of all the proteins and pieces of mRNA involved in the process:

$$perm \overset{0.001}{\mapsto} \epsilon \qquad Irna \overset{0.001}{\mapsto} \epsilon \qquad transac \overset{0.001}{\mapsto} \epsilon \tag{R14-R16}$$

$$repr \overset{0.002}{\mapsto} \epsilon \qquad betagal \overset{0.01}{\mapsto} \epsilon \qquad Rna \overset{0.01}{\mapsto} \epsilon \tag{R17-R19}$$

$$RLACT \overset{0.002}{\mapsto} LACT \qquad (perm\cdot\widetilde{x})^L \rfloor X \overset{0.001}{\mapsto} (\widetilde{x})^L \rfloor X \tag{R20-R21}$$

We recall that sequences are not allowed as context of application of rules, hence rule (R14) cannot be applied to *perm* when this is an element of the looping sequence representing the membrane of the bacterium. This motivates the presence of the rule (R21).

4.2 Simulation Results

We simulated the evolution of the bacterium in the absence of lactose (modelled by the term *Ecoli* of Equation (1)) and in the presence of 100 molecules of lactose in the environment (modelled by the term *EcoliLact* of Equation (2)).

In Figure 3 we show the results of the two simulations. The first graph shows that in the absence of lactose the production of the beta galactosidase and lactose permease enzymes starts after more than 750 seconds, and that the number of such enzymes is always smaller than 20. Moreover, this graph shows that the lactose permeases, once produced, become immediately part of the membrane of the bacterium, because the number of such enzymes not on the membrane remains always small.

The second and the third graphs show the results of the simulation when the lactose is present in the environment. In this case the production of the enzymes starts almost immediately (as shown by the second graph), but we remark that the fact that the starting times in the production of enzymes in the two simulations are different is not relevant. The amount of time elapsed before the production of these enzymes does not depend on the presence of the lactose in the environment, as the lactose cannot enter the bacterium until some molecules of permease have joint the membrane.

Finally, the third graph shows that if lactose is present in the environment, it starts entering the bacterium after some molecules of lactose permease join the membrane and it is decomposed into glucose and galactose. Once some molecules of lactose permease join the membrane, the lactose starts entering the bacterium (see the third graph). In fact, the third graph shows that the number of molecules in the environment rapidly decreases.

Once entered the bacterium, the lactose interacts with the lac Repressor, and this favors the production of more enzymes.

Once all the molecules of lactose have been decomposed, the number of lac Repressors increases, reaching the same values of the first simulation. The number of beta galactosidase and lactose permease enzymes, instead, does not decrease, and hence does not reach the values of the first simulation. This happens because the degradation of such enzymes, and of the mRNA from which they are translated, is a very slow process, which would take much more time than the time of the simulations we performed.

Notice that the curves proposed in the figures (and the ones in the next section) result from a single experiment chosen among several we have performed with the same initial conditions. Even though the outputs of probabilistic simulations are inherently approximations of the overall behaviour of a system, in our cases the experiments gave always the same results, apart for negligible fluctuations. In tens of simulations no rare events manifested. Thus, instead of computing an average of the different simulations we decided to choose just one of them as a good representative of the standard behaviour of the system. Thus, we are confident that the proposed simulations reflect a realistic behaviour of the overall system.

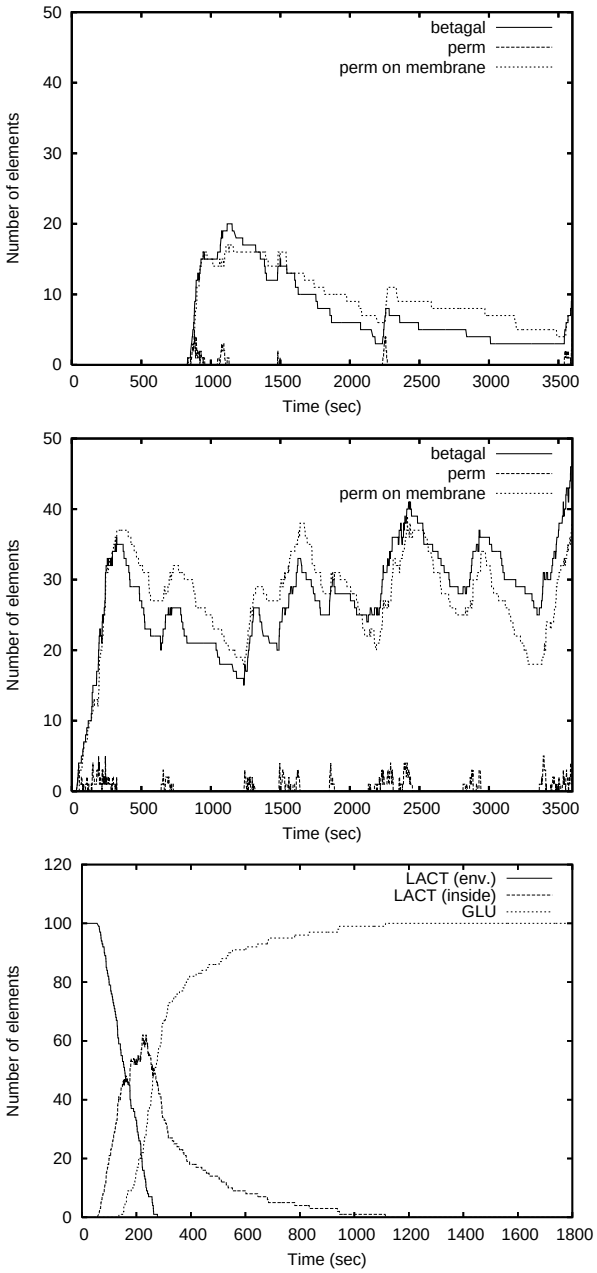


Fig. 3. Simulation results: production of enzymes in the absence (top) and presence (middle) of lactose, and degradation of lactose into glucose (bottom).

5 Modelling Quorum Sensing

Traditionally, bacteria have been studied as independent individuals. Now, it is recognised that many bacteria have the ability of monitoring their population density and modulating their gene expressions according to this density. This process is called *quorum sensing*.

The process of quorum sensing consists in two activities, one involving one or more diffusible small molecules (called *autoinducers*) and the other involving one or more transcriptional activator proteins (*R-proteins*) located within the cell. The autoinducer can cross the cellular membrane, and thus it can diffuse either out or in bacteria.

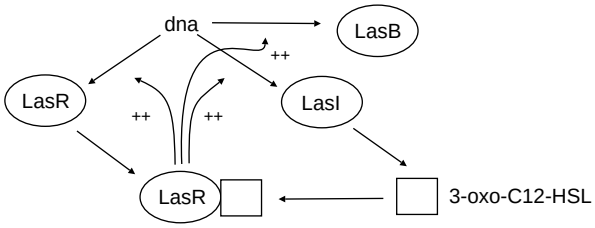
The production of the autoinducer is regulated by the R-protein. The R-protein by itself is not active without the corresponding autoinducer. The autoinducer molecule can bind to the R-protein to form an *autoinducer/R-protein* complex, which binds to a target of the DNA sequence enhancing the transcription of specific genes. Usually, these genes regulate both the production of specific behavioural traits (as we will show in the following) and the production of the autoinducer and of the R-protein.

At low cell density, the autoinducer is synthesized at basal levels and diffuse in the environment where it is diluted. With high cell density both the extracellular and intracellular concentrations of the autoinducer increase until they reach thresholds beyond which the autoinducer is produced autocatalytically. The autocatalytic production results in a dramatic increase of product concentration.

Quorum sensing behaviour is very widespread in bacteria. An example is the regulation of the bioluminescence in the symbiotic marine bacterium *Vibrio fischeri*, which colonizes the light organs of marine fishes and squids. The bacteria only luminesce when they are found in high concentrations in the light organs, while they do not emit light when they are free swimming [30]. Another example is given by the bacterium *Pseudomonas aeruginosa*, a prevalent human pathogen [31]. The ability of *P. aeruginosa* to infect a host mainly is based on controlling its virulence by quorum sensing. The level of virulence expressed by isolated bacteria is very low, thus avoiding host response. When a colony has reached a certain density, the production of virulence factors is autoinduced by quorum sensing, and it is generally sufficient to overcome the defenses of the host.

The quorum sensing system of *P. aeruginosa* has two regulatory systems regulating the expression of elastase LasA and elastase LasB, respectively. The two enzymes are responsible for pulmonary hemorrhages associated with *P. aeruginosa* infections. In this paper we are interested in the regulatory system of elastase LasB, named the *las* system.

A schematic description of the *las system* is as follows (arrows from an element to another one represent the production of the second element starting form the first one, arrows with the ++ label represent catalysed faster productions):



The autoinducer 3-oxo-C12-HSL and the transcriptional activator protein LasR are produced at basal rates starting from the dna and LasI, respectively. The LasR/3-oxo-C12-HSL dimer, which is the activated form of LasR, promotes the production of itself, of the autoinducer and of the LasB enzyme. The formation of the dimer is controlled mainly by the concentration of the autoinducer, which is influenced by the number of bacteria.

5.1 Stochastic CLS Model

We now give the SCLS model of the quorum sensing process. We do not model the production of the LasB as it has not an active role in the regulation process. The initial state of each bacterium is:

$$Bact \ ::= \ (m)^L \rfloor (lasO \cdot lasR \cdot lasI)$$

where the looping sequence $(m)^L$ represents the bacterium membrane, $lasO$ the target of the DNA sequence where LasR/3-oxo-C12-HSL complex binds to for promoting DNA transcription, and $lasR$ and $lasI$ the genes that encode $LasR$ and the autoinducer.

This model shows one of the advantages of using terms for describing the structure of biological systems in SCLS. In fact, in order to model a population of n bacteria we have to describe only one bacterium, and then compose n copies of such a description by using the parallel composition operator. In other words, we model a population of n bacteria simply as $n \times Bact$.

We now give the stochastic rewrite rules describing the protein/protein and protein/DNA interactions in the described systems. Again, we have only to give the rules for one bacterium, and they will be applicable in all the n bacteria of the considered population.

$$lasO \cdot lasR \cdot lasI \overset{20}{\longmapsto} lasO \cdot lasR \cdot lasI \,|\, LasR \qquad \text{(R1)}$$

$$lasO \cdot lasR \cdot lasI \overset{5}{\longmapsto} lasO \cdot lasR \cdot lasI \,|\, LasI \qquad \text{(R2)}$$

$$LasI \overset{8}{\longmapsto} LasI \,|\, 3oxo \qquad \text{(R3)}$$

$$3oxo \,|\, LasR \overset{0.25}{\longmapsto} 3R \qquad 3R \overset{400}{\longmapsto} 3oxo \,|\, LasR \qquad \text{(R4-R5)}$$

$$3R \,|\, lasO \cdot lasR \cdot lasI \overset{0.25}{\longmapsto} 3RO \cdot lasR \cdot lasI \qquad \text{(R6)}$$

$$3RO \cdot lasR \cdot lasI \overset{10}{\longmapsto} 3R \,|\, lasO \cdot lasR \cdot lasI \qquad \text{(R7)}$$

$$3RO \cdot lasR \cdot lasI \overset{1200}{\longmapsto} 3RO \cdot lasR \cdot lasI \,|\, LasR \qquad \text{(R8)}$$

$$3RO \cdot lasR \cdot lasI \overset{300}{\longmapsto} 3RO \cdot lasR \cdot lasI \,|\, LasI \qquad \text{(R9)}$$

$$(m)^L \rfloor (3oxo \,|\, X) \overset{30}{\longmapsto} 3oxo \,|\, (m)^L \rfloor X \qquad \text{(R10)}$$

$$3oxo \,|\, (m)^L \rfloor X \overset{1}{\longmapsto} (m)^L \rfloor (3oxo \,|\, X) \qquad \text{(R11)}$$

$$LasI \overset{1}{\longmapsto} \epsilon \qquad LasR \overset{1}{\longmapsto} \epsilon \qquad 3oxo \overset{1}{\longmapsto} \epsilon \qquad \text{(R12-S14)}$$

Rules (R1) and (R2) describe the production from the DNA of proteins LasR and LasI, respectively. For the sake of simplicity we do not model the transcription of the DNA into mRNA. Rule (R3) describes the production of the autoinducer 3-oxo-C12-HSL, denoted $3oxo$, performed by the LasI enzyme. Rules (R4) and (R5) describe the complexation and decomplexation of the autoinducer and the LasR protein, where the complex is denoted $3R$. Rules from (R7) to (R9) describe the binding of the activated autoinducer to the DNA and its influence in the production of LasR and LasI. Rules (R10) and (R11) describe the autoinducer exiting and entering the bacterium. The kinetic constants associated with these two rules give a measure of the autoinducer dilution. Finally, rules from (R12) to (R14) describe the degradation of proteins.

5.2 Experimental Results

We simulated the behaviour of a population of *P. aeruginosa* by varying the number of individuals. In Figure 4 we show how the concentration of the autoinducer varies inside bacteria when the population is composed by one, five and twenty individuals. In the last two cases we show the autoinducer concentration inside one only bacterium (the concentrations inside the others are analogous).

When the number of bacteria increases also the concentration of the autoinducer in the extracellular space increases. As a consequence the concentration of the autoinducer in the intracellular spaces increases as well and the quorum sensing process starts. Note that the kinetic constants of rules (R10) and (R11), regulating the autoinducer exiting and entering the membrane, cause the bacteria to maintain the autoinducer production mostly at a basal rate when the population size is one or five. When the population size is twenty the quorum sensing starts after a few seconds thus causing a very high autocatalytic autoinducer production. Increasing the ratio between the kinetic constants of (R10) and (R11) would cause the quorum sensing to be triggered when the number of individuals is bigger.

6 Conclusions

As we have seen, SCLS allows representing membranes and operations on them. Other formalisms were developed to describe membrane systems. Among them we cite Brane Calculi [7] and P-Systems [21].

SCLS can describe situations that cannot be easily captured by the above mentioned formalisms, which consider membranes as atomic objects. An example of this is given by the representation of the membrane of *Escherichia coli*, shown in Section 4. Representing the membrane as a sequence of elements permits the definition of different functionalities depending on the type and the number of

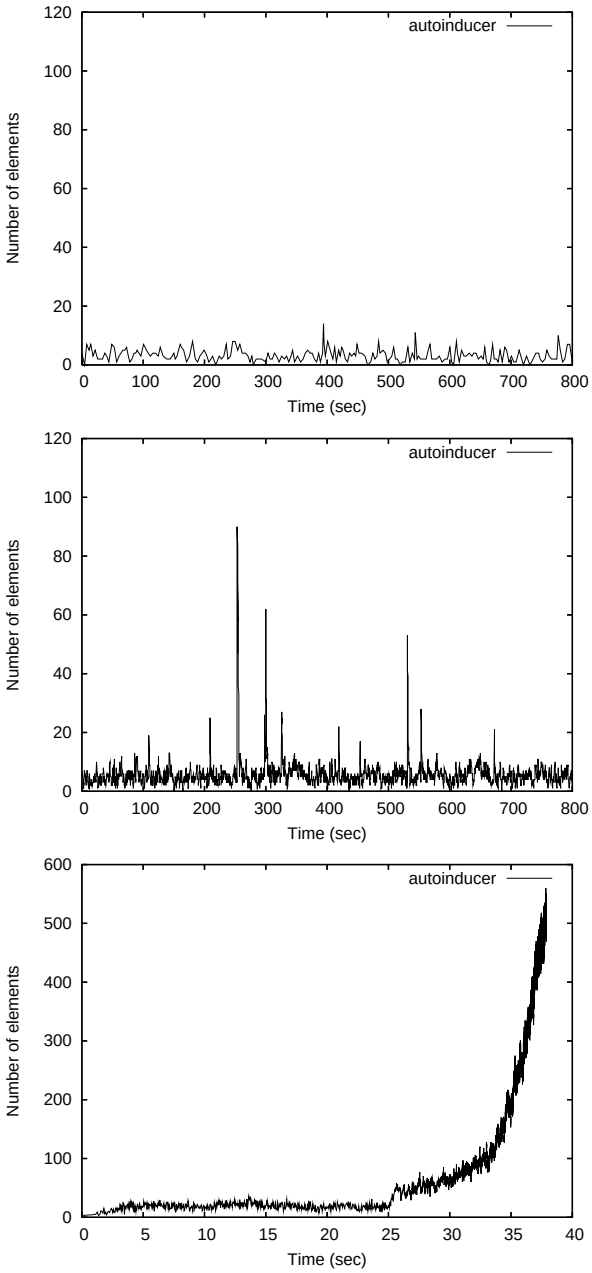


Fig. 4. Simulation results: quantity of autoinducer inside one bacterium in a population of one (top), five (middle) and twenty (bottom) bacteria.

elements on the membrane itself. In the example, the presence and the number of lactose permeases on the bacterium membrane regulates the transportation of lactose inside the bacterium. Moreover, as no restrictions are imposed on the format of rewrite rules, SCLS seems to be suitable for the description of a wider class of systems than the one the formalisms mentioned above easily handle.

Quorum sensing is a complex biological process, which is not based on signals and receptors but only on the concentration of a protein freely crossing bacteria membranes. Many mathematical models have been developed for describing this challenging phenomenon [13, 16, 32]. These models consider various aspects of the problem: the diffusion of the autoinducer, its degradation, the percentage of "up-regulated" bacteria (the ones with an enhanced production of the autoinducer), the density of bacteria, their size, etc. However, all these models describe the process at a very abstract level. They consider that the intracellular concentration of the autoinducer is a function of the density of the bacteria, although modulated by other factors. Thus they start from this assumption to study the behaviour of the system with different values of the parameters.

The SCLS model is based on a different approach. A single bacterium is described by means of a set of rewrite rules modelling its internal processes. These rules also model the autoinducer crossing (in both directions) the cellular membranes and the autoinducer degrading at the same rate both inside and outside cells.

Differently from the mathematical models mentioned above, the SCLS model describes the elementary processes each bacterium performs, and the quorum sensing results from the activity of a sufficient number of bacteria. The stochastic nature of SCLS allows observing fluctuations of the autoinducer concentration which in the first two cases considered are not sufficient to trigger quorum sensing. Moreover, our model shows the discrete behaviour of the binding between the autoinducer/R-protein complex and DNA. In Figures 5 and 6 we show the dynamics of the binding between the autoinducer/R-protein complex and the DNA in one of the bacteria of the populations of five and twenty bacteria, respectively. In Figure 5, since the autoinducer and the R protein are produced at basal levels, the concentration of the autoinducer/R-protein complex inside the bacterium is low. As a consequence, the complex does not bind to DNA, except in the few cases of stochastic peaks in the autoinducer concentration. In Figure 6, when the quorum sensing process starts (in this case approximately after 30 seconds), the concentration of the autoinducer/R-protein complex increases sharply causing the complex to be constantly bound to the DNA. As a consequence, the autoinducer is produced autocatalitically.

6.1 Related Work

Cell biology, the study of the morphological and functional organization of cells, is now an established field in biochemical research. Computer Science can help the research in cell biology in several ways. For instance, it can provide biologists with models and formalisms capable of describing and analyzing complex systems such as cells.

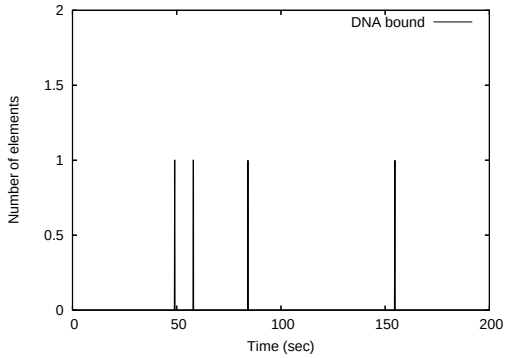


Fig. 5. The binding of the autoinducer/R-protein complex inside a bacterium in the population of five.

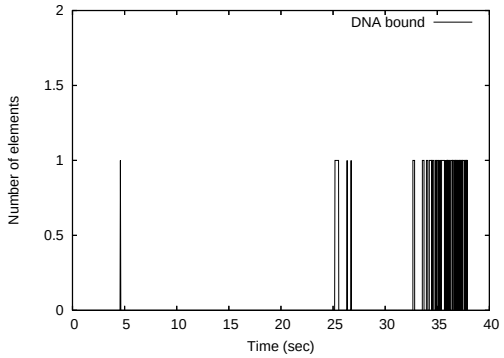


Fig. 6. The binding of the autoinducer/R-protein complex inside a bacterium in the population of twenty.

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 [19], Hybrid Systems [1], and the π-calculus [10, 29]. Moreover, new formalisms have been defined for describing biomolecular and membrane interactions [4, 7, 8, 12, 24, 26]. 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 [24, 26] 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 [7, 8, 12] 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 [12] 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 presented in the present paper, 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 [25], and the PEPA framework proposed by Hillston in [15].

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 ([11]) and in BioSPi ([25]). 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. We already mentioned the work by Krivine et al. [17], in which a stochastic semantics for bigraphs has been developed. The intuitive and natural methodology they have developed to count rule occurrences has been adapted,

in the present paper, to count the number of occurrences of stochastic rewrite rules within CLS terms.

An alternative stochastic semantics for CLS has been defined in [2]. Such a semantics computes the transition rates in a compositional way and it is rather complicated. Moreover, in [20] and in preliminary versions of the present paper we defined the stochastic extension of CLS (upon which the current version of our simulator is based) by enriching rewrite rules with rate functions rather than rate constants. Such functions allow the definition of kinetics that are more complex than the standard mass-action ones, but are rather difficult to be used when modelling systems. By following the approach of [17], in this paper we have been able to give a natural and simpler definition of SCLS.

We would also like to mention that a computational model describing the quorum sensing process in *Vibrio fischeri* [30] by modelling single bacteria is presented in [22]. The model is defined with a variant of P-Systems.

Tools and Applications Among the simulation tools based on other stochastic formalisms we mention the following ones:

- SPiM (`http://research.microsoft.com/~aphillip/spim/`) and Cytosim (`http://www.cosbi.eu/Rpty_Soft_CytoSim.php`) are constructed on the Stochastic π-calculus;
- the Beta Workbench (`http://www.cosbi.eu/Rpty_Soft_BetaWB.php`) provides a collection of tools based on Beta-binders [24];
- Bio-PEPA (`http://www.dcs.ed.ac.uk/home/stg/software/biopepa`) is a biologically inspired extension of PEPA;
- PSym (`http://psystems.disco.unimib.it/`) is a simulation tool developed for P-Systems.

These tools, as our prototype simulator for SCLS, can be used to perform stochastic simulations. Other tools often used to study biological systems, such as GEPASI (`http://www.gepasi.org/`), simulate systems by solving differential equations. This kind of simulation is well-suited for chemical solutions with big quantities of reactants, and becomes less precise when the number of reactants decreases (as in cells) as it is based on a continuous representation of the quantities of reactants.

In conclusion, we also point out that the translation of SBML descriptions of cellular pathways (`http://sbml.org`) into SCLS is trivial, provided the reactions in the pathways are based on standard mass action kinetics.

References

1. Alur, R., Belta,C., Ivancic,F., Kumar,V., Mintz,M., Pappas,G.J., Rubin,H. and Schug,J. (2001) Hybrid modeling and simulation of biomolecular networks. *Proc. of Hybrid Systems: Computation and Control*, LNCS 2034, Springer, 19-32.

2. Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P. and Pardini, G. (2008) The Calculus of Looping Sequences. *In Formal Methods for Computational Systems Biology (SFM 2008)*, LNCS 5016, Springer, 387-423.
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 International Symposium on Bioinformatics Research and Applications (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 International Colloquium on Theoretical Aspects of Computing (ICTAC'06)*, LNCS 4281, Springer, 93-107.
6. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., and Troina, A. (2008) Bisimulations in Calculi Modelling Membranes. *Formal Aspects of Computing*, to appear.
7. Cardelli, L. (2005) Brane calculi. Interactions of biological membranes. *Proc. of Comput. Methods in Systems Biology (CMSB'04)*, LNCS 3082, Springer, 257-280.
8. 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.
9. Ciocchetta, F. and Hillston, J. (2008) Bio-PEPA: a framework for the modelling and analysis of biological systems. *Theor. Comput. Sci.*, to appear.
10. Curti, M., Degano, P., Priami, C., and Baldari, C. T. (2004) Modelling Biochemical Pathways through Enhanced pi-calculus. *Theor. Comput. Sci.*, **325**, 111-140.
11. Danos, V., Feret, J., Fontana, W., and Krivine, J. (2007) Scalable modelling of biological pathways. *Proc. of ASIAN Symposium on Programming Languages and Systems (APLAS'07)*, LNCS 4807, Springer, 139-157.
12. Danos, V. and Laneve, C. (2004) Formal molecular biology. *Theor. Comput. Sci.*, **325**, 69-110.
13. Dockery, J.D. and Keener, J.P. (2001) A mathematical model for quorum sensing in Pseudomonas aeruginosa. *Bulletin of Mathematical Biology*, **63**, 95-116.
14. Gillespie, D. (1977) Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, **81**, 2340-2361.
15. Hillston. J. (1996) *A Compositional Approach to Performance Modelling*. Cambridge University Press.
16. James, S., Nilsson, P., James, G., Kjelleberg, S. and Fagerstroem, T. (2000) Luminescence Control in the Marine Bacterium Vibrio fischeri: An Analysis of the Dynamics of lux Regulation. *Journal of Molecular Biology*, **296**, 1127-1137.
17. Krivine, J., Milner, R. and Troina, A. (2008) Stochastic Bigraphs. *Proc. of the 24th Conference on Mathematical Foundations of Programming Semantics (MFPS'08)*, ENTCS, Elsevier, to appear.
18. Kwiatkowska, M., Norman, G. and Parker, D. (2004) Probabilistic symbolic model checking with PRISM: a hybrid approach. *Int. J. on Software Tools for Technology Transfer*, **6** 128-142.
19. Matsuno, H., Doi, A., Nagasaki, M. and Miyano, S. (2000) Hybrid Petri net representation of gene regulatory network. *Prooceedings of Pacific Symposium on Biocomputing*, World Scientific Press, 341-352.
20. Milazzo, P. (2007) *Qualitative and quantitative formal modeling of biological systems*. Ph.D. Thesis, University of Pisa.

21. Păun, G. (2002) *Membrane computing.* An introduction. Springer, 2002.
22. Pérez-Jiménez, M. J. and Romero-Campero, F. J. (2005) Modelling Vibrio fischeri's behaviour using P systems. *Proc. of the 8th European Conference on Artificial Life, Systems Biology Workshop.*
23. Priami, C. (1995) Stochastic π-calculus. *The Computer Journal*, **38**, 578-589.
24. Priami, C. and Quaglia, P. (2005) Beta Binders for Biological Interactions. *Proc. of Computational Methods in Systems Biology (CMSB'04)*, LNCS 3082, Springer, 20-33.
25. 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.
26. 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.
27. Regev, A. and Shapiro, E. (2002) Cells as computation. *Nature*, **419**, 343.
28. Regev, A. and Shapiro, E. (2004) The π-calculus as an abstraction for biomolecular systems. *Modelling in Molecular Biology*, Natural Computing Series, Springer, 219-266.
29. 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.
30. Stevens, A. M. and Greenberg, E. P. (1997) Quorum sensing in Vibrio fischeri: essential elements for activation of the luminescence genes. *J. of Bacteriology*, **179**, 557-562.
31. Van Delden, C. and Iglewski, B. H. (1998) Cell-to-cell signaling and Pseudomonas aeruginosa infections. *Emerg Infect Dis*, **4**, 551-560.
32. Ward, J. P., King, J. R., Koerber, A. J., Croft, J. M., Sockett, R. E. and Williams, P. (2003) Early development and quorum sensing in bacterial biofilms. *Journal of Mathematical Biology*, **47**, 23-55.
33. Wilkinson, D. (2006) *Stochastic modelling for Systems Biology.* Chapman & Hall/CRC.
34. 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.