The Calculus of Looping Sequences

Roberto Barbuti, Giulio Caravagna, Andrea Maggiolo–Schettini,
Paolo Milazzo, and Giovanni Pardini

Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo 3, 56127 Pisa, Italy
{barbuti,caravagn,maggiolo,milazzo,pardinig}@di.unipi.it

Abstract. We describe the Calculus of Looping Sequences (CLS) which is suitable for modeling microbiological systems and their evolution. We present two extensions, CLS with links (LCLS) and Stochastic CLS. LCLS simplifies the description of protein interaction at a lower level of abstraction, namely at the domain level. Stochastic CLS allows us to describe quantitative aspects of the modeled systems, such as the frequency of chemical reactions. As examples of application to real biological systems, we show the simulation of the activity of the lactose operon in *E.coli* and the quorum sensing process in *P.aeruginosa*, both described with Stochastic CLS.

1 Introduction

Cell biology, the study of the morphological and functional organization of cells, is now an established field in biochemical research. Computer Science can help 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. 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 [29], Hybrid Systems [1], and the π-calculus [15, 42]. Moreover, new formalisms have been defined for describing biomolecular and membrane interactions [4, 11, 13, 17, 36, 40]. Others, such as P Systems [31, 32], 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 [36, 40] have been successful in the description of biological systems, as they allow systems to be described in a compositional way. However, these calculi offer very low–level interaction primitives, causing models to become very large and difficult to understand. Calculi such as those proposed in [11, 13, 17] 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 or protein interactions. Finally, P Systems 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 flexible enough to allow an easy description of new activities observed on membranes without extending the formalism itself.

Therefore, the need has arisen for a formalism with a simple notation, having the ability of describing biological systems at different levels of abstraction, having some notions of compositionality and being flexible enough to allow the description of new kinds of phenomena, without being specialized to the description of a particular class of systems. For this reason in [7] we have introduced the Calculus of Looping Sequences (CLS).

CLS is a formalism based on term rewriting with some features, such as a commutative parallel composition operator, and some semantic means, such as bisimulations, which are common in process calculi. All this permits us to combine the simplicity of notation of rewriting systems with the advantage of a form of compositionality.

Given an alphabet of symbols representing basic biological entities, such as genes, proteins and other macro–molecules, CLS terms are constructed by applying to these symbols operators of sequencing, looping, containment and parallel composition. Terms constructed by means of these operators represent biological structures such as DNA sequences and membranes. Rewrite rules can be used to model biological events that permit the system to evolve. In particular, they can be used to model biochemical reactions and structure rearragements such as membrane fusion and dissolution.

Some variants of CLS have been defined in [30]. Moreover, in [5,6], two extensions have been introduced. The first, CLS with links (LCLS), allows the description of protein interaction at a lower level of abstraction, namely at the domain level. The second, Stochastic CLS, allows the description of quantitative aspects of the modeled systems such as the frequency of chemical reactions. For Stochastic CLS a simulator has been developed [14] that allows simulating the evolution of biological systems over time.

In [8,9] we have defined bisimulations for CLS. Bisimulations may be used to compare the behaviour of two systems and as an alternative technique to verify a property of a system. This can be done by assessing the bisimilarity of a system with a system one knows to enjoy that property.

In this paper, we describe CLS and its two extensions, and we show two examples of application to real biological systems. While the formulations of CLS and LCLS differ slightly from those given in [7,5], the present formulation of Stochastic CLS is new and is more convenient than the one in [6] when dealing with biochemical systems. As examples of application to real biological systems, we show the simulation of the activity of the lactose operon in *E.coli* and the quorum sensing process in *P.aeruginosa*, both described with Stochastic CLS.

2 Setting the Context

Both the qualitative and the quantitative aspects of biological systems are interesting. The former are related to *state dependent properties*, such as reachability

of states or existence of equilibria and stable states; the latter are related to *time and probability dependent properties*, like the time needed to reach a certain state and the probability of reaching a certain state in a given time or in any time. We briefly describe some notable examples of formalisms that have been used in the last few years for modeling both aspects.

As regards qualitative aspects of biological systems, Lindenmayer systems (or L systems) [38] are one of the oldest formalisms. An L system is a formal grammar most famously used to model the growth processes of plant development.

In the tradition of automata and formal language theory, more recently Paun has proposed P Systems [31, 32]. P Systems introduce the idea of membrane computing in the subject of natural computing. They represent a new computational paradigm which allows NP-complete problems to be solved in polynomial time (but in exponential space), have originated a huge amount of work and recently have been also applied to the description of biological systems (see [39] for a complete list of references).

A pioneering formalism for the description of biological systems is the κ–calculus of Danos and Laneve [17]. It is a formal language for protein interactions, is enriched with a very intuitive visual notation and has been encoded into the π–calculus. The κ–calculus idealizes protein-protein interactions, essentially as a particular restricted kind of graph–rewriting operating on graphs with sites. A formal protein is a node with a fixed number of sites, and a complex (i.e. a bundle of proteins connected together by low energy bounds) is a connected graph built over such nodes, in which connections are established between sites. The κ–calculus has been recently extended to model also membranes [28].

An example of direct application of a model for concurrency to biochemical systems has been shown by Regev and Shapiro in [41] and by Regev, Silverman and Shapiro in [42]. Their idea is to describe biomolecular pathways as π–calculus processes. Chemical reactions between biological components are modeled as communications on channels whose names can be passed and sharing names of private channels allows the description of biological compartments. Regev, Panina, Silverman, Cardelli and Shapiro in [40] defined the BioAmbients calculus, a model inspired by both the π–calculus and the Mobile Ambients calculus [12], which can be used to describe biochemical systems with a notion of compartments (as, for instance, membranes). An extension of the π–calculus for the description of membranes and of biological interfaces is the Beta–binders calculus defined by Priami and Quaglia in [36]. More details of membrane interactions have been considered by Cardelli in the definition of Brane Calculi [11], which are elegant formalisms for describing intricate biological processes involving membranes. A refinement of Brane Calculi has been introduced by Danos and Pradalier in [18].

We mention also some works by Harel [23] and Kam et al.[26], in which the challenging idea is introduced of modelling a full multi–cellular animal as a reactive system. The multi–cellular animal should be, specifically, the *C.elegans* nematode worm, which is complex, but well defined in terms of anatomy and genetics. Moreover, Harel proposes to use the languages of Statecharts [22] and Live

Sequence Charts (LSC) [16], which are visual notations with a formal semantics and are commonly adopted in the specification of software projects. Harel et al. apply the same formalisms also to cellular and multi–cellular systems related to the immune systems of living organisms in [25] and [21].

As regards quantitative aspects of biological systems, they are usually described by biologists by means of differential equations. Each equation gives the transformation rate of one of the components of the described system. Hence, the dynamics of the system can be studied analitically and simulations can be performed by using a computer tool for solving differential equations. This technique has been successfully used in a huge number of cases, but it suffers from the following drawbacks: (i) the solution of a set of differential equations gives a unique "average" behavior of the system, and does not model stochastic fluctuations of the quantities of the involved components; (ii) when the size and the complexity of the modeled system increases, differential equations become difficult to manage; (iii) the approach assumes quantities to be expressed as continuous values, and this could lead to erroneous approximations, in particular when the number of components of the system is very small.

An alternative approach to the simulation of biological systems is the use of stochastic simulators. This kind of tools are usually based on simulation algorithms proved to be correct with respect to the kinetic theory of chemical reactions. The most used and well–established of such algorithms is the one introduced by Gillespie in [24]. In his paper, Gillespie shows that the quantity of time spent between the occurrences of two chemical reactions is exponentially distributed, with the sum of the kinetic rates of the possible reactions as the parameter of the exponential distribution. This allows him to give a very simple and exact stochastic algorithm for simulating chemical reactions.

Gillespie's algorithm is the *trait–d'union* between simulation of biological systems and stochastic process algebras, and permits the latter to be easily applied to the description of biological systems. In particular, the *stochastic π–calculus* [35, 37] has been successfully applied to the (quantitative) modeling of biological systems. In this extension of the π–calculus, kinetic constants are associated with communication channels and determine the stochastic behaviour of the model, in terms of communications, as in Gillespie's algorithm they determine occurrences of chemical reactions. Analogous stochastic extensions have been proposed for other formalisms such as P Systems [34], BioAmbients [10] and Beta–binders [19], and simulation tools have been developed.

The transition system obtained by the semantics of a stochastic formalism may be transformed into a Continuous Time Markov Chain (CTMC). If the set of states of the CTMC is manageable, a standard probabilistic model checker (such as PRISM [27]) can be used to verify properties of the described system.

3 The Calculus of Looping Sequences (CLS)

In the next sections we introduce the Calculus of Looping Sequences (CLS) and two variants, CLS with links (LCLS) and Stochastic CLS. The former is a variant

to model protein interaction at the domain level. The latter, equipped with a stochastic semantics, permits the description of quantitative aspects of biological systems. For all the formalisms presented here we show the modeling of some real biological systems.

3.1 The Basic Formalism

CLS is 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 modeled system, and the rewrite rules to represent the events that may cause the system to evolve.

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

Definition 1 (Terms). Terms T *and* sequences S *are given by the following grammar:*

$$
\begin{aligned}
T &::= S \quad \big| \quad \left(T\right)^L \rfloor T \quad \big| \quad T \mid T \\
S &::= \epsilon \quad \big| \quad a \quad \big| \quad S \cdot S
\end{aligned}
$$

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 parallel composition operator $\_ \mid \_$, a looping operator $\left(\_\right)^L$ 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. By definition, looping and containment are always applied together, hence we can consider them as a single binary operator $\left(\_\right)^L \rfloor \_$. Brackets can be used to indicate the order of application of the operators, and we assume $\left(\_\right)^L \rfloor \_$ to have precedence over $\_ \mid \_$.

The biological interpretation of the operators is the following: the main entities which occur in cells are DNA and RNA strands, proteins, membranes, and other macro–molecules. DNA strands (and similarly RNA strands) are sequences of nucleic acids, but they can be seen also at a higher level of abstraction as sequences of genes. Proteins are sequence of amino acids which usually have a very complex three–dimensional structure. In a protein there are usually (relatively) few subsequences, called domains, which actually are able to interact with other entities by means of chemical reactions. CLS sequences can model DNA/RNA strands and proteins by describing each gene or each domain with a symbol of the alphabet. Membranes are closed surfaces, often interspersed with proteins, which may have a content. Looping and containment allow the representation of membranes with their contents. For example, the term $\left(a \mid b\right)^L \rfloor c$ represents a membrane with the elements a and b on its surface and containing the element c. Other macro–molecules can be modeled as single alphabet symbols, or as short sequences. Finally, juxtaposition of entities can be described by the parallel composition of their representations.
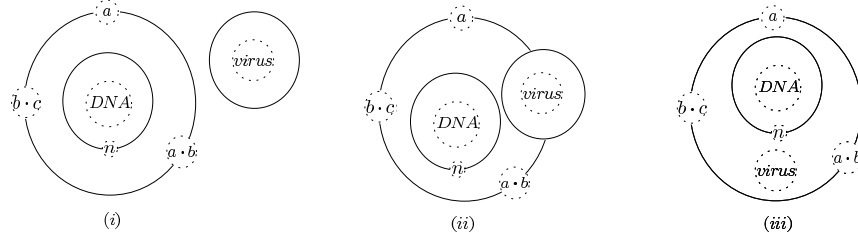
Fig. 1. A visual representation of some examples of CLS terms.

In Figure 1 we show the visual representation of some examples of CLS terms. Term (i), $\left(\epsilon\right)^L \rfloor virus \mid \left(a \mid b \cdot c \mid a \cdot b\right)^L \rfloor \left(n\right)^L \rfloor DNA$, represents an environment where there exist a virus, represented by $\left(\epsilon\right)^L \rfloor virus$, and a cell represented by $\left(a \mid b \cdot c \mid a \cdot b\right)^L \rfloor \left(n\right)^L \rfloor DNA$. The cell has three proteins on its external membrane, represented by the sequences a, $b \cdot c$ and $a \cdot b$, and contains a membrane with surface n, representing the nucleus and containing a DNA strand, represented by the sequence DNA. Term (ii), $\big(a \mid b \cdot c \mid a \cdot b \mid \left(\epsilon\right)^L \rfloor virus\big)^L \rfloor \left(n\right)^L \rfloor DNA$, represents the same cell of term (i) but with the virus attached to the surface of its external membrane. Finally, term (iii), $\left(a \mid b \cdot c \mid a \cdot b\right)^L \rfloor \left(virus \mid \left(n\right)^L \rfloor DNA\right)$, represents the state in which the virus, dissolving its external membrane, has entered the cell.

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

Definition 2 (Structural Congruence). *The structural congruence relations $\equiv_S$ and $\equiv$ are the least congruence relations on sequences and on terms, respectively, satisfying the following rules:*

$$S_1 \cdot (S_2 \cdot S_3) \equiv_S (S_1 \cdot S_2) \cdot S_3 \qquad S \cdot \epsilon \equiv_S \epsilon \cdot S \equiv_S S$$

$$S_1 \equiv_S S_2 \; implies \; S_1 \equiv S_2$$
$$T_1 \mid T_2 \equiv T_2 \mid T_1 \qquad T_1 \mid (T_2 \mid T_3) \equiv (T_1 \mid T_2) \mid T_3$$
$$T \mid \epsilon \equiv T \qquad \left(\epsilon\right)^L \rfloor \epsilon \equiv \epsilon$$

Rules of the structural congruence state the associativity of $\cdot$ and $\mid$, the commutativity of the latter and the neutral role of ϵ.

Note that there exist variants of the CLS formalism having a different syntax of terms and a different structural congruence relation. Among these variants it is worth mentioning those defined in [8] and in [30]. In the former, the looping operator can be applied only to a single sequence of alphabet symbols, and such a sequence can rotate by applying an axiom of the structural congruence. In the latter, called CLS+, the looping operator can be applied to a parallel composition of sequences, but not to other loopings.

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

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

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

where a is a generic element of $\mathcal{E}$, and $X, \widetilde{x}$ and x are generic 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. We can now 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 $Var(P_2) \subseteq Var(P_1)$. We denote with $\Re$ the infinite set of all the possible rewrite rules.*

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 $P_2\sigma$. For instance, the rule $a \cdot b \mid b \cdot \widetilde{x} \mapsto c \cdot \widetilde{x}$ prescribes the replacement, with respect to an instantiation function σ, of a sequence $a \cdot b$ and a sequence $b \cdot \sigma(\widetilde{x})$ with a sequence $c \cdot \sigma(\widetilde{x})$. If the term to which the rule is applied is $a \cdot b \mid b \cdot a \cdot b \mid c$, and we assume an instantiation function $\sigma = \{(\widetilde{x}, a \cdot b)\}$, the result of applying the rule to the term is $c \cdot a \cdot b \mid c$. The rule $p \mid (m \mid Y)^L \rfloor X \mapsto (m \mid Y \mid p)^L \rfloor X$ could describe the attachment of element p to a membrane containing on its surface at least an element m. As in the previous example, the membrane to which the element p gets attached, depends on the instantiation of variables X and Y.

With reference to Figure 1, the rewrite rule that transforms the state represented by term (i) into the state represented by term (ii) could be $(\epsilon)^L \rfloor virus \mid (X)^L \rfloor (Y \mid (n)^L \rfloor DNA) \mapsto (X \mid (\epsilon)^L \rfloor virus)^L \rfloor (Y \mid (n)^L \rfloor DNA)$. Analogously, the rule which transforms the state represented by term (ii) into the state represented by term (iii) could be $(X \mid (\epsilon)^L \rfloor virus)^L \rfloor (Y \mid (n)^L \rfloor DNA) \mapsto (X)^L \rfloor (Y \mid virus \mid (n)^L \rfloor DNA)$.

We now define the semantics of CLS as a transition system in which states correspond to terms and transitions correspond to rule applications.

Definition 5 (Semantics). *Given a finite set of rewrite rules $\mathcal{R} \subseteq \Re$, the semantics of CLS is the least transition relation $\rightarrow$ on terms closed under structural congruence $\equiv$ and satisfying the following inference rules:*

$$\frac{P_1 \mapsto P_2 \in \mathcal{R} \quad P_1 \sigma \not\equiv \epsilon \quad \sigma \in \Sigma}{P_1 \sigma \rightarrow P_2 \sigma} \qquad \frac{T_1 \rightarrow T_2}{T \mid T_1 \rightarrow T \mid T_2}$$

$$\frac{T_1 \rightarrow T_2}{(T)^L \rfloor T_1 \rightarrow (T)^L \rfloor T_2} \qquad \frac{T_1 \rightarrow T_2}{(T_1)^L \rfloor T \rightarrow (T_2)^L \rfloor T}$$

Note that, by the definition of the semantics, a rule cannot be applied to a portion of a sequence. This means that a rule such as $b \mapsto d$ cannot be applied to $a \cdot b \cdot c$ (so to obtain $a \cdot d \cdot c$). This constraint is necessary because the application of a rule to a portion of a sequence could lead to syntactically wrong terms. For instance, if it would be possible to apply $b \mapsto d \mid e$ to $a \cdot b \cdot c$, one would obtain $a \cdot (d \mid e) \cdot c$ as a result which is not a valid term. However, the constraint does not reduce the expressiveness of the formalism because, in order to modify a portion of a system, one can replace any rule such as $b \mapsto d$ with $\widetilde{x} \cdot b \cdot \widetilde{y} \mapsto \widetilde{x} \cdot d \cdot \widetilde{y}$.

Finally, 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.

Examples A well–known example of biomolecular system is the epidermal growth factor (EGF) signal transduction pathway [44, 33]. If EGF proteins are present in the environment of a cell, they should be interpreted as a proliferation signal from the environment, and hence the cell should react by synthesizing proteins which stimulate its proliferation. A cell recognizes the EGF signal because it has on its membrane some EGF receptor proteins (EGFR), which are transmembrane proteins (they have some intra–cellular and some extra–cellular domains). One of the extra–cellular domains binds to one EGF protein in the environment, forming a signal–receptor complex on the membrane. This causes a conformational change on the receptor protein that enables it to bind to another signal–receptor complex. The formation of the binding of the two signal–receptor complexes (called dimerization) causes the phosphorylation of some intra–cellular domains of the dimer. This, in turn, causes the internal domains of the dimer to be recognized by a protein that is inside the cell (in the cytoplasm), called SHC. The protein SHC binds to the dimer, enabling a long chain of protein–protein interactions, which finally activate some proteins, such

as one called ERK, which bind to the DNA and stimulate synthesis of proteins for cell proliferation.

Now, we use CLS to build a model of the first steps of the EGF signaling pathway up to the binding of the signal-receptor dimer to the SHC protein.

We model the EGFR,EGF and SHC proteins as the alphabet symbols $EGFR$, EGF and SHC, respectively. The cell is modeled as a looping applied to a parallel composition of sequences (representing the external membrane) initially composed only by symbols $EGFR$, containing symbols SHC and surrounded by symbols EGF. The rewrite rules modeling the first steps of the pathway are the following:

$$EGF \mid \left(EGFR \mid Y\right)^{L} \rfloor X \ \mapsto \ \left(CPL \mid Y\right)^{L} \rfloor X \qquad (R1)$$

$$\left(CPL \mid CPL \mid Y\right)^{L} \rfloor X \ \mapsto \ \left(CPL \cdot CPL \mid Y\right)^{L} \rfloor X \qquad (R2)$$

$$\left(CPL \cdot CPL \mid Y\right)^{L} \rfloor X \ \mapsto \ \left(CPLp \cdot CPLp \mid Y\right)^{L} \rfloor X \qquad (R3)$$

$$\left(CPLp \cdot CPLp \mid Y\right)^{L} \rfloor (SHC \mid X) \mapsto \left(CPLp \cdot CPLp \cdot SHC \mid Y\right)^{L} \rfloor X \quad (R4)$$

Rule R1 describes the binding of a EGF protein to a EGFR receptor protein on the membrane surface. The result of the binding is a signal-receptor complex denoted CPL. Rule R2 describes the dimerization of two signal-receptor complexes, the result is a sequence of two signal-receptor CPL symbols. Rule R3 describes the phosphorylation (and activation) of a signal-receptor dimer, that is the replacement of a $CPL \cdot CPL$ sequence with $CPLp \cdot CPLp$. Finally, rule R4 describes the binding of an active dimer $CPLp \cdot CPLp$ to a SHC protein contained in the cytoplasm. The result is a $CPLp \cdot CPLp \cdot SHC$ sequence placed on the membrane surface.

A possible initial term for the model in this example is given by a membrane having, on its surface, a parallel composition of symbols $EGFR$ and, inside, some symbols SHC and, outside, some symbols EGF. A possible evolution of such a term by means of application of the given rewrite rules is the following (we write on each transition the name of the applied rewrite rule):

$$EGF \mid EGF \mid \left(EGFR \mid EGFR \mid EGFR \mid EGFR\right)^{L} \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R1)} EGF \mid \left(EGFR \mid CPL \mid EGFR \mid EGFR\right)^{L} \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R1)} \left(EGFR \mid CPL \mid EGFR \mid CPL\right)^{L} \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R2)} \left(EGFR \mid CPL \cdot CPL \mid EGFR\right)^{L} \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R3)} \left(EGFR \mid CPLp \cdot CPLp \mid EGFR\right)^{L} \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R4)} \left(EGFR \mid CPLp \cdot CPLp \cdot SHC \mid EGFR\right)^{L} \rfloor SHC$$

We show another example of modeling of a biomolecular system with CLS, that is the modeling of a simple gene regulation process. This kind of processes are essential for cell life as they allow a cell to regulate the production of proteins

that may have important roles, for instance in metabolism, growth, proliferation and differentiation.

The example we consider is as follows: we have a simple DNA fragment consisting of a sequence of three genes. The first, denoted p, is called *promoter* and is the place where a *RNA polymerase* enzyme (responsible for translation of DNA into RNA) binds to the DNA. The second, denoted o, is called *operator* and it is the place where a *repressor* protein (responsible for regulating the activity of the RNA polymerase) binds to the DNA. The third, denoted g, is the gene that encodes for the protein whose production is regulated by this process.

When the repressor is not bound to the DNA, the RNA polymerase can scan the sequence of genes and transcribe gene g into a piece of RNA that will be later translated into the protein encoded by g. When the repressor is bound to the DNA, it becomes an obstacle for the RNA polymerase that cannot scan anymore the sequence of genes.

The CLS model of this simple regulation process is a follows. The sequence of genes is represented as the CLS sequence $p \cdot o \cdot g$, the RNA polymerase enzyme as $polym$, the repressor protein as $repr$, and the piece of RNA obtained by the translation of gene g as rna. The rewrite rules describing the process are the following:

$$polym \mid p \cdot \widetilde{x} \;\; \mapsto \;\; pp \cdot \widetilde{x} \tag{R1}$$

$$repr \mid \widetilde{x} \cdot o \cdot \widetilde{y} \;\; \mapsto \;\; \widetilde{x} \cdot ro \cdot \widetilde{y} \tag{R2}$$

$$pp \cdot o \cdot \widetilde{x} \;\; \mapsto \;\; p \cdot po \cdot \widetilde{x} \tag{R3}$$

$$\widetilde{x} \cdot po \cdot g \;\; \mapsto \;\; \widetilde{x} \cdot o \cdot pg \tag{R4}$$

$$\widetilde{x} \cdot pg \;\; \mapsto \;\; polym \mid rna \mid \widetilde{x} \cdot g \tag{R5}$$

Rules R1 and R2 describe the binding of the RNA polymerase and of the repressor to the corresponding genes in the DNA sequences. The results of these bindings are that the symbols representing the two genes are replaced by pp and ro, respectively. Rules R3, R4 and R5 describe the activity of the RNA polymerase enzyme in the absence of the repressor: it moves from gene p to gene o in rule R3, then it moves from gene o to gene g in rule R4, and finally it produces the RNA fragment and leaves the DNA in rule R5. Note that, in order to apply rule R3, the repressor must not be bound to the DNA.

The only possible evolution of a term representing an initial situation in which no repressors are present is

$$polym \mid p \cdot o \cdot g \;\; \xrightarrow{(R1)} \;\; pp \cdot o \cdot g \;\; \xrightarrow{(R3)} \;\; p \cdot po \cdot g$$
$$\xrightarrow{(R4)} \;\; p \cdot o \cdot pg \;\; \xrightarrow{(R5)} \;\; polym \mid rna \mid p \cdot o \cdot g$$

that represents the case in which the RNA polymerase enzyme can scan the DNA sequence and transcribe gene g into a piece of RNA. When the repressor is present, instead, a possible evolution is

$$repr \mid polym \mid p \cdot o \cdot g \;\; \xrightarrow{(R1)} \;\; repr \mid pp \cdot o \cdot g \;\; \xrightarrow{(R2)} \;\; pp \cdot ro \cdot g$$

that corresponds to a situation in which the repressor stops the transcription of the gene by hampering the activity of the RNA polymerase.

3.2 An Extension for the Modeling of Protein Interaction at the Domain Level

To model a protein at the domain level in CLS it would be natural to use a sequence with one symbol for each domain. However, the binding between two domains of two different proteins, that is the linking between two elements of two different sequences, cannot be expressed in CLS. To represent this, CLS has been extended in [5] by labels on basic symbols. If in a term two symbols have the same label, we intend that they represent domains that are bound to each other. If in a term there is a single symbol with a certain label, we intend that the term represents only a part of a system we model, and that the symbol will be linked to another symbol in another part of the term representing the full model.

As membranes create compartments, elements inside a looping sequence cannot be linked to elements outside. Elements inside a membrane can be linked either to other elements inside the membrane or to elements of the membrane itself. An element can be linked at most to another element. The partner to which an element is bound can be different at different times, and a domain able to bind to multiple partners simultaneously could be described by using more elements instead of a single one.

For the sake of simplicity, the syntax of terms of the CLS with links (LCLS) is defined as in [5], namely the looping operator can be applied only to a single sequence rather than a LCLS term.

Definition 6 (Terms). *Terms T and sequences S of LCLS are given by the following grammar:*

$$
\begin{aligned}
T &::= S \mid \left(S\right)^{L} \rfloor\, T \mid T \mid T \\
S &::= \epsilon \mid a \mid a^{n} \mid S \cdot S
\end{aligned}
$$

where a is a generic element of $\mathcal{E}$, and n is a natural number. We denote with $\mathcal{T}$ the infinite set of terms, and with $\mathcal{S}$ the infinite set of sequences.

Definition 7 (Structural Congruence). *The structural congruence relations $\equiv_S$ and $\equiv$ are the least congruence relations on sequences and on terms, respectively, satisfying the following rules:*

$$
S_1 \cdot (S_2 \cdot S_3) \equiv_S (S_1 \cdot S_2) \cdot S_3 \qquad S \cdot \epsilon \equiv_S \epsilon \cdot S \equiv_S S
$$

$$
S_1 \equiv_S S_2 \ \text{implies} \ S_1 \equiv S_2 \ \text{and} \ \left(S_1\right)^{L} \rfloor\, T \equiv \left(S_2\right)^{L} \rfloor\, T
$$

$$
T_1 \mid T_2 \equiv T_2 \mid T_1 \qquad T_1 \mid (T_2 \mid T_3) \equiv (T_1 \mid T_2) \mid T_3 \qquad T \mid \epsilon \equiv T
$$

$$
\left(\epsilon\right)^{L} \rfloor\, \epsilon \equiv \epsilon \qquad \left(S_1 \cdot S_2\right)^{L} \rfloor\, T \equiv \left(S_2 \cdot S_1\right)^{L} \rfloor\, T
$$

Note that, differently from the structural congruence relation of CLS, in this case the sequence to which the looping operator is applied can be rotated by axiom $\left(S_1 \cdot S_2\right)^L \rfloor T \equiv \left(S_2 \cdot S_1\right)^L \rfloor T$. For instance, the sequence $\left(a \cdot b^3 \cdot c\right)^L \rfloor d$ is equivalent to the sequence $\left(b^3 \cdot c \cdot a\right)^L \rfloor d$ and to the sequence $\left(c \cdot a \cdot b^3\right)^L \rfloor d$.

Patterns of LCLS are similar to those of CLS, with the addition of the labels.

Definition 8 (Patterns). *Patterns P and sequence patterns SP of LCLS are given by the following grammar:*

$$
\begin{aligned}
P &\ ::=\ SP \ \mid\ \left(SP\right)^L \rfloor P \ \mid\ P \mid P \ \mid\ X \\
SP &\ ::=\ \epsilon \ \mid\ a \ \mid\ a^n \ \mid\ SP \cdot SP \ \mid\ \widetilde{x} \ \mid\ x \ \mid\ x^n
\end{aligned}
$$

where a is an element of $\mathcal{E}$, n is a natural number 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.

Note that an LCLS term is also an LCLS pattern; everything we define for patterns will be immediately defined also for terms. Moreover, in what follows, we will often use the notions of *compartment* and of *top–level compartment* of a pattern. A compartment is a subpattern that is the second operand of a containment operator and in which the contents of inner containment operators are not considered. The top–level compartment is the portion of the pattern that is not the content of any containment operator. For instance, the top–level compartment of a pattern $P = a \mid \left(b\right)^L \rfloor c \mid \left(d\right)^L \rfloor \left(X \mid \left(e\right)^L \rfloor f\right)$ is $a \mid \left(b\right)^L \rfloor \epsilon \mid \left(d\right)^L \rfloor \epsilon$. Other compartments in P are c, $X \mid \left(e\right)^L \rfloor \epsilon$, and f.

An LCLS pattern is well–formed if and only if a label occurs no more than twice, and two occurrences of a label are always in the same compartment. The following type system will be used for deriving the well–formedness of patterns.

In each inference rule the conclusion has the form $(N, N') \models P$, where N and N' are sets of natural numbers with N the set of labels used twice and N' the set of labels used only once in the top–level compartment of P.

Definition 9 (Type System). *The typing algorithm for LCLS patterns is defined by the following inference rules:*

$$1.\ (\varnothing, \varnothing) \models \epsilon \qquad 2.\ (\varnothing, \varnothing) \models a \qquad 3.\ (\varnothing, \{n\}) \models a^n$$

$$4.\ (\varnothing, \varnothing) \models x \qquad 5.\ (\varnothing, \{n\}) \models x^n \qquad 6.\ (\varnothing, \varnothing) \models \widetilde{x} \qquad 7.\ (\varnothing, \varnothing) \models X$$

$$8.\ \frac{(N_1, N_1') \models SP_1 \quad (N_2, N_2') \models SP_2 \quad N_1 \cap N_2 = N_1' \cap N_2 = N_1 \cap N_2' = \varnothing}{\left(N_1 \cup N_2 \cup (N_1' \cap N_2'), (N_1' \cup N_2') \setminus (N_1' \cap N_2')\right) \models SP_1 \cdot SP_2}$$

$$9.\ \frac{(N_1, N_1') \models P_1 \quad (N_2, N_2') \models P_2 \quad N_1 \cap N_2 = N_1' \cap N_2 = N_1 \cap N_2' = \varnothing}{\left(N_1 \cup N_2 \cup (N_1' \cap N_2'), (N_1' \cup N_2') \setminus (N_1' \cap N_2')\right) \models P_1 \mid P_2}$$

$$10.\ \frac{(N_1, N_1') \models SP \quad (N_2, N_2') \models P \quad N_1 \cap N_2 = N_1' \cap N_2 = N_1 \cap N_2' = \varnothing \quad N_2' \subseteq N_1'}{\left(N_1 \cup N_2', N_1' \setminus N_2'\right) \models \left(SP\right)^L \rfloor P}$$

where a is a generic element of $\mathcal{E}$, n is a natural number, and $X, \widetilde{x}$ and x are generic elements of TV, SV and $\mathcal{X}$, respectively. We write $\models P$ if there exist $N, N' \subset \mathbb{N}$ such that $(N, N') \models P$, and $\not\models P$ otherwise.

Rules 1–7 are self explanatory. Rule 8 states that a sequence pattern $SP_1 \cdot SP_2$ is well–typed if there are no labels occurring either four times ($N_1 \cap N_2 = \varnothing$) or three times ($N_1' \cap N_2 = N_1 \cap N_2' = \varnothing$). Labels occurring twice in $SP_1 \cdot SP_2$ are those which occur twice either in SP_1 or in SP_2 together with labels occurring once both in SP_1 and in SP_2. Rule 9 for the parallel composition is analogous to rule 8. Rule 10 states that the only labels which can be used for typing $\left(SP\right)^L \rfloor P$ must be different from those used for typing P. Moreover the labels used once in P must be used once in SP, that is these labels are used to bind elements inside the membrane to elements on the membrane itself. As an example, the pattern $P \equiv a^1 \mid \left(b^1 \cdot b^2\right)^L \rfloor c \cdot c^2 \cdot c$ can be typed as follows.

$$(\emptyset, \{1\}) \models a^1 \qquad \text{(by rule 3)}$$
$$(\emptyset, \{1, 2\}) \models b^1 \cdot b^2 \qquad \text{(by rules 3 and 8)}$$
$$(\emptyset, \{2\}) \models c \cdot c^2 \cdot c \qquad \text{(by rules 2, 3 and 8)}$$
$$(\{2\}, \{1\}) \models \left(b^1 \cdot b^2\right)^L \rfloor c \cdot c^2 \cdot c \qquad \text{(by rule 10)}$$
$$(\{1, 2\}, \emptyset) \models P \qquad \text{(by rule 9)}$$

On the contrary, the pattern $a^1 \mid \left(b\right)^L \rfloor c^1$ cannot be typed because the premise $N_2' \subseteq N_1'$ of rule 10 of the type system is not satisfied. Similarly, $\not\models a^1 \mid b^1 \mid c^1$ and $\not\models a^1 \mid \left(b^1\right)^L \rfloor c^1$ hold since the premises of rules 9 and 10, respectively, are not satisfied.

The type system can be used to introduce a concept of well–formedness.

Definition 10 (Well–Formedness of Patterns). *A pattern P is* well–formed *if and only if $\models P$ holds.*

The use of labels to represent links is not new. In [17] well–formedness of terms is given by a concept of graph–likeness. We notice that in our case membranes, which are not present in the formalism of [17], make the treatment more complicated. In [28], where the concept of membrane is introduced, well–formedness of terms is given intuitively and not formally defined.

In the following we shall use a notion of multiset of labels and set of links of a pattern. The former represents the multiset of all the labels appearing in the top level compartment of a pattern, the latter represents the set of labels that occur twice in the top–level compartment of a pattern. We shall denote with $\#(e, M)$ the number of occurrences of element e in the multiset M.

Definition 11. *The* multiset of labels *of a pattern P is $L_M(P)$ where:*

$$L_M(\epsilon) = \varnothing \qquad L_M(\nu) = \varnothing \qquad L_M(\nu^n) = \{n\} \qquad L_M(\widetilde{x}) = \varnothing$$

$$L_M(SP_1 \cdot SP_2) = L_M(SP_1) \cup L_M(SP_2) \qquad L_M(P_1 \mid P_2) = L_M(P_1) \cup L_M(P_2)$$

$$L_M\left(\left(SP\right)^L \rfloor P\right) = L_M(SP) \cup (L_M(SP) \cap L_M(P)) \qquad L_M(X) = \varnothing$$

where $\nu \in \mathcal{E} \cup EV$, $n \in \mathbb{N}$, P_1, P_2 are any pattern, SP is any sequence pattern.

Definition 12. *The* set of links *of a pattern P is $L(P) = \{n | \#(n, L_M(P)) = 2\}$,*

As an example, given pattern $P \equiv a^1 \mid (b^1 \cdot b^2)^L \rfloor c \cdot c^2 \cdot c^3 \mid a^3$, we have that $L_M(P) = \{1\} \cup \{1, 2\} \cup (\{1, 2\} \cap \{2, 3\}) = \{1, 1, 2, 2\}$ and, consequently, $L(P) = \{1, 2\}$ because link labeled by number 3 is not in the top–level compartment of P. If P is a well–formed pattern, there exists $N \subset \mathbb{N}$ such that $(L(P), N) \models P$.

Let $\mathcal{A}$ be the set of all total injective functions $\alpha : \mathbb{N} \to \mathbb{N}$. Given $\alpha \in \mathcal{A}$, the α–*renaming* of an LCLS pattern P is the pattern $P\alpha$ obtained by replacing every label n in P by $\alpha(n)$. For example, let P be $a^1 \mid (b^2)^L \rfloor c^2$ and α be such that $\alpha(n) = n + 1$. We have that $P\alpha \equiv a^2 \mid (b^3)^L \rfloor c^3$. It is easy to note that, since α is injective, the application of α–renaming to well–formed patterns preserves the well-formedness.

Links in a term are placeholders: the natural number used in the two labels of a link has not a particular meaning. Hence, we can consider as equivalent patterns which differ only in the values of their links. This equivalence is formally defined as follows.

Definition 13 (α–equivalence). *The α–equivalence relation $=_\alpha$ on LCLS patterns is the least equivalence relation which satisfies the following rules:*

1. $\dfrac{P_1 \equiv P_2}{P_1 =_\alpha P_2}$

2. $\dfrac{ni \notin L_M(SP_j) \quad i = 1, 2 \quad j = 1, 2, 3}{SP_1 \cdot \nu^{n1} \cdot SP_2 \cdot \mu^{n1} \cdot SP_3 =_\alpha SP_1 \cdot \nu^{n2} \cdot SP_2 \cdot \mu^{n2} \cdot SP_3}$

3. $\dfrac{ni \notin L_M(SP_j) \quad i = 1, 2 \quad j = 1, 2, 3, 4}{SP_1 \cdot \nu^{n1} \cdot SP_2 \mid SP_3 \cdot \mu^{n1} \cdot SP_4 =_\alpha SP_1 \cdot \nu^{n2} \cdot SP_2 \mid SP_3 \cdot \mu^{n2} \cdot SP_4}$

4. $\dfrac{ni \notin L_M(SP_j) \cup L_M(P) \quad i = 1, 2 \quad j = 1, 2, 3, 4}{\left(SP_1 \cdot \nu^{n1} \cdot SP_2\right)^L \rfloor (SP_3 \cdot \mu^{n1} \cdot SP_4 \mid P) =_\alpha}$
$\left(SP_1 \cdot \nu^{n2} \cdot SP_2\right)^L \rfloor (SP_3 \cdot \mu^{n2} \cdot SP_4 \mid P)$

5. $\dfrac{ni \notin L_M(SP_j) \cup L_M(P) \quad i = 1, 2 \quad j = 1, 2, 3, 4}{SP_1 \cdot \mu^{n1} \cdot SP_2 \mid \left(SP_3 \cdot \nu^{n1} \cdot SP_4\right)^L \rfloor P =_\alpha}$
$SP_1 \cdot \mu^{n2} \cdot SP_2 \mid \left(SP_3 \cdot \nu^{n2} \cdot SP_4\right)^L \rfloor P$

6. $\dfrac{ni \notin L_M(SP_j) \cup L_M(P_k) \quad i = 1, 2 \quad j = 1, 2, 3, 4 \quad k = 1, 2}{\left(SP_1 \cdot \mu^{n1} \cdot SP_2\right)^L \rfloor P_1 \mid \left(SP_3 \cdot \nu^{n1} \cdot SP_4\right)^L \rfloor P_2 =_\alpha}$
$\left(SP_1 \cdot \mu^{n2} \cdot SP_2\right)^L \rfloor P_1 \mid \left(SP_3 \cdot \nu^{n2} \cdot SP_4\right)^L \rfloor P_2$

7. $\dfrac{ni \notin L_M(SP_j) \cup L_M(P_k) \quad i = 1, 2 \quad j = 1, 2, 3, 4 \quad k = 1, 2}{\left(SP_1 \cdot \mu^{n1} \cdot SP_2\right)^L \rfloor (P_1 \mid \left(SP_3 \cdot \nu^{n1} \cdot SP_4\right)^L \rfloor P_2) =_\alpha}$
$\left(SP_1 \cdot \mu^{n2} \cdot SP_2\right)^L \rfloor (P_1 \mid \left(SP_3 \cdot \nu^{n2} \cdot SP_4\right)^L \rfloor P_2)$

8. $\dfrac{\begin{array}{c} P_1 =_\alpha P_2 \quad P_3 =_\alpha P_4 \\ L_M(P_1) \cap L(P_3) = L(P_1) \cap L_M(P_3) = \varnothing \\ L_M(P_2) \cap L(P_4) = L(P_2) \cap L_M(P_4) = \varnothing \end{array}}{P_1 \mid P_3 =_\alpha P_2 \mid P_4}$

$$9. \quad \frac{\begin{array}{c} SP_1 =_\alpha SP_2 \quad P_1 =_\alpha P_2 \\ L_M(SP_1) \cap L(P_1) = L(SP_1) \cap L_M(P_1) = \varnothing \\ L_M(SP_2) \cap L(P_2) = L(SP_2) \cap L_M(P_2) = \varnothing \end{array}}{(SP_1)^L \rfloor P_1 =_\alpha (SP_2)^L \rfloor P_2}$$

*where $\nu, \mu \in \mathcal{E} \cup EV$, $n1, n2 \in \mathbb{N}$, P_1, P_2, P_3, P_4 are any pattern, $SP_1, SP_2, SP_3,$
SP_4 are any sequence pattern.*

Rule 1 says that structurally congruent patterns are also α–equivalent. rules from 2 to 7 describe the ridenomination of the two labels of a link with fresh labels. Each rule deals with a possible situation in which a link may occur. Rule 2 deals with links between two elements of the same sequence, rule 3 with links between two elements of two sequences composed in parallel, and so on. Rules 8 and 9 describe the closure of the relation with respect to parallel composition and containment.

As an example, the term $a^1 \cdot b^2 \cdot c^1 \cdot d \mid (e^2 \cdot f^3 \cdot g)^L \rfloor h^3 \cdot i$ is α–equivalent to the term $a^4 \cdot b^5 \cdot c^4 \cdot d \mid (e^5 \cdot f^6 \cdot g)^L \rfloor h^6 \cdot i$ as rules 2 and 8 can be used to replace link 1 with link 4, rules 5 and 8 to replace link 2 with link 5 and rules 4 and 8 to replace link 3 with link 6.

Note that the labels occurring only once in a pattern P are not renamed by the α–equivalence relation. Instead, the application of an α–renaming function to P may change these labels. Moreover, labels which occur twice in more than one compartment of the pattern can be renamed differently in each compartment by the α–equivalence relation, while they are all renamed by the same value by applying some α–renaming function.

As in CLS, rewrite rules in LCLS are pairs of patterns, but in this case we require that the two patterns are well–formed.

Definition 14 (Rewrite Rules). *A rewrite rule is a pair of well–formed 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 $Var(P_2) \subseteq Var(P_1)$. We denote with $\Re$ the infinite set of all the possible rewrite rules.*

Now, we can define the semantics of LCLS.

Definition 15 (Semantics). *Given a set of rewrite rules $\mathcal{R} \subseteq \Re$, the semantics of LCLS is the least transition relation $\rightarrow$ on well–formed terms closed under $\equiv$ and $=_\alpha$, and satisfying the following inference rules:*

$$(app) \quad \frac{P_1 \mapsto P_2 \in \mathcal{R} \quad P_1\sigma \not\equiv \epsilon \quad \sigma \in \Sigma \quad \alpha \in \mathcal{A}}{P_1\alpha\sigma \rightarrow P_2\alpha\sigma}$$

$$(par) \quad \frac{T_1 \rightarrow T_1' \quad L(T_1) \cap L(T_2) = \{n_1, \ldots, n_M\} \quad n_1', \ldots, n_M' \text{ fresh}}{T_1 \mid T_2 \rightarrow T_1'\{n_1', \ldots, n_M'/n_1, \ldots, n_M\} \mid T_2}$$

$$(cont) \quad \frac{T \rightarrow T' \quad L(S) \cap L(T') = \{n_1, \ldots, n_M\} \quad n_1', \ldots, n_M' \text{ fresh}}{(S)^L \rfloor T \rightarrow (S)^L \rfloor T'\{n_1', \ldots, n_M'/n_1, \ldots, n_M\}}$$

where the symmetric rule for the parallel composition is omitted.

Rule *(app)* says that a rewrite rule $P_1 \mapsto P_2$ can be applied to a term that can be obtained from P_1 by some renaming α and instantiation σ. The result of the application is P_2 renamed and instantiated in the same manner. Rule *(par)* describes the derivation of a transition from a state represented by a parallel composition of terms. In such a state, for the set of links $\{n_1, \dots, n_M\}$ of both T_1 and T_2, a set of fresh links $\{n'_1, \dots, n'_M\}$ is assumed. The fresh links substitute in T'_1 the links with the same name T_1 and T_2. Rule *(cont)* describes the derivation of a transition from a state given by a looping. Also in this case fresh links are assumed and substituted into the term which describes the new content of the membrane T'. It is worth noting that, as the semantics is defined over well–formed terms, no transitions can be derived that lead to non well–formed terms. As a consequence, rewrite rules cannot be applied if they transform a term into another which is not well–formed. For instance, rewrite rule $a^1 \mapsto a$ cannot be applied to $(c)^L \rfloor (a^1 \mid b^1 \cdot b)$ as it would result in $(c)^L \rfloor (a \mid b^1 \cdot b)$ that is not well–formed. We have introduced this restriction on the the domain of the semantics for the sake of simplicity. In [5] we have given a more complex semantics which preserves well–formedness without the need of this restriction.

Example We model in LCLS the steps of the EGF pathway (see example in Section 3.1) up to the binding of the protein SHC to the dimer. We model the EGFR protein as the sequence $R_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2}$, where R_{E1} and R_{E2} are two extra–cellular domains and R_{I1} and R_{I2} are two intra–cellular domains. The membrane of the cell is modeled as a looping sequence which could contain EGFR proteins. Outside the looping sequence (i.e. in the environment) there could be EGF proteins, and inside (i.e. in the cytoplasm) there could be SHC proteins. The rewrite rules modeling the pathway are the following:

$$EGF \mid \left(R_{E1} \cdot \widetilde{x}\right)^L \rfloor X \;\; \mapsto \;\; \left(SR_{E1} \cdot \widetilde{x}\right)^L \rfloor X \tag{R1}$$

$$\left(SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot \widetilde{x} \cdot SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot \widetilde{y}\right)^L \rfloor X \;\; \mapsto$$
$$\left(SR_{E1} \cdot R_{E2}^1 \cdot R_{I1} \cdot R_{I2} \cdot SR_{E1} \cdot R_{E2}^1 \cdot R_{I1} \cdot R_{I2} \cdot \widetilde{x} \cdot \widetilde{y}\right)^L \rfloor X \tag{R2}$$

$$\left(R_{E2}^1 \cdot R_{I1} \cdot \widetilde{x} \cdot R_{E2}^1 \cdot R_{I1} \cdot \widetilde{y}\right)^L \rfloor X \;\; \mapsto \;\; \left(R_{E2}^1 \cdot PR_{I1} \cdot \widetilde{x} \cdot R_{E2}^1 \cdot R_{I1} \cdot \widetilde{y}\right)^L \rfloor X \tag{R3}$$

$$\left(R_{E2}^1 \cdot PR_{I1} \cdot \widetilde{x} \cdot R_{E2}^1 \cdot R_{I1} \cdot \widetilde{y}\right)^L \rfloor X \;\; \mapsto \;\; \left(R_{E2}^1 \cdot PR_{I1} \cdot \widetilde{x} \cdot R_{E2}^1 \cdot PR_{I1} \cdot \widetilde{y}\right)^L \rfloor X \tag{R4}$$

$$\left(R_{E2}^1 \cdot PR_{I1} \cdot R_{I2} \cdot \widetilde{x} \cdot R_{E2}^1 \cdot PR_{I1} \cdot R_{I2} \cdot \widetilde{y}\right)^L \rfloor (SHC \mid X) \;\; \mapsto$$
$$\left(R_{E2}^1 \cdot PR_{I1} \cdot R_{I2}^2 \cdot \widetilde{x} \cdot R_{E2}^1 \cdot PR_{I1} \cdot R_{I2} \cdot \widetilde{y}\right)^L \rfloor (SHC^2 \mid X) \tag{R5}$$

Rule R1 represents the binding of the EGF protein to the receptor domain R_{E1} with SR_{E1} as a result. Rule R2 represents that when two EGFR proteins activated by proteins EGF occur on the membrane, they may bind to each other to form a dimer (shown by the link 1). Rule R3 represents the phosphorylation of one of the internal domains R_{I1} of the dimer, and rule R4 represents the

phosphorylation of the other internal domain R_{I1} of the dimer. The result of each phosphorylation is PR_{I1}. Rule R5 represents the binding of the protein SHC in the cytoplasm to an internal domain R_{I2} of the dimer. Remark that the binding of SHC to the dimer is represented by the link 2, allowing the protein SHC to continue the interactions to stimulate cell proliferation.

Let us denote the $R_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2}$ by EGFR. By starting from a cell with some EGFR proteins on its membrane, some SHC proteins in the cytoplasm and some EGF proteins in the environment, a possible evolution is the following (we write on each transition the name of the rewrite rule applied):

$$EGF \mid EGF \mid \left(EGFR \cdot EGFR \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R1)} EGF \mid \left(SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R1)} \left(SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR \cdot SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR\right)^L \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R2)} \left(SR_{E1} \cdot R_{E2}^1 \cdot R_{I1} \cdot R_{I2} \cdot SR_{E1} \cdot R_{E2}^1 \cdot R_{I1} \cdot R_{I2} \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R3)} \left(SR_{E1} \cdot R_{E2}^1 \cdot PR_{I1} \cdot R_{I2} \cdot SR_{E1} \cdot R_{E2}^1 \cdot R_{I1} \cdot R_{I2} \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R4)} \left(SR_{E1} \cdot R_{E2}^1 \cdot PR_{I1} \cdot R_{I2} \cdot SR_{E1} \cdot R_{E2}^1 \cdot PR_{I1} \cdot R_{I2} \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R5)} \left(SR_{E1} \cdot R_{E2}^1 \cdot PR_{I1} \cdot R_{I2}^2 \cdot SR_{E1} \cdot R_{E2}^1 \cdot PR_{I1} \cdot R_{I2} \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC^2 \mid SHC)$$

3.3 A Stochastic Extension

In CLS only qualitative aspects of biological systems are considered, such as their structure and the presence (or the absence) of certain molecules. As a consequence, on CLS models it is only possible to verify properties such as the reachability of particular states or causality relationships between events. It would be interesting to verify also properties such as the time spent to reach a particular state, or the probability of reaching it. To face this problem, in [6] we have developed a stochastic extension of CLS, called *Stochastic CLS*, in which quantitative aspects, such as time and probability, are taken into account.

The standard way of extending a formalism to model quantitative aspects of biological systems is by incorporating the stochastic framework developed by Gillespie with its simulation algorithm for chemical reactions [24] in the semantics of the formalism. This has been done, for instance, for the π–calculus [35, 37]. The idea of Gillespie's algorithm is that a rate constant is associated with each chemical reaction that may occur in the system. 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 constant is then used as the parameter of an exponential distribution modeling 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 the description of the system as a Continuous Time Markov Chain (CTMC), and consequently it allows the verification of properties of the described system by means of analytic tools and by means of stochastic model checkers.

In Stochastic CLS, the incorporation of Gillespie's stochastic framework is not a simple exercise. The main difficulty is counting the number of possible reactant combinations of the chemical reaction described by a rewrite rule. Reactants are given by the left pattern in a rewrite rule, which may contain variables. A rewrite rule can be applied in different positions of the term which describes the state of the system. In order to compute the rate of application of the rule we have to count the number of different positions where the rewrite rule can be applied, by taking into account instantiation of variables.

We have defined the Stochastic CLS in [6], and showed how to derive a CTMC from the semantics of a system modeled in the formalism. This allows performing simulation and verification of properties of the described systems, for instance by using stochastic model checkers, such as PRISM [27].

In the present work the semantics given in [6] is slightly revised. In particular, the rewrite rules defined in [6] are enriched, with respect to those of CLS, with a rate function while, here, they are enriched with a kinetic constant. Rate functions, built over the domain of the instantiation functions σ, were assumed to be defined in order to correctly compute the rate of the modeled chemical reaction. Intuitively, a rule with left hand side $a \mid \left(c \mid X \right)^L \rfloor \epsilon$ modeling a reaction with reactants a and c, where c is placed on the surface of a membrane, should have a rate which depends on the number of c appearing on the surface itself, that is on the instantiation of X. In the semantics of [6], the computation of the number of c appearing at in the term $\sigma(X)$ is assumed to be done by the rate function associated with the rule. Differently, the semantics we define here, embeds such a computation. Obviously, the version of Stochastic CLS of [6] is more general as any function can be associated with a rule to specify how the rule rate has to be computed. However, the restrictions we introduce here simplify the modeling of biochemical systems.

Now we can define Stochastic CLS. The syntax of terms and the structural congruence relations are the same as those of CLS defined in Section 3.1.

As regards patterns, in Stochastic CLS we distinguish between the patterns that will be used on the left hand side of rewrite rules (called *left patterns*) and those that will be used on the right hand side of rewrite rules (called *right patterns*). In particular, we will impose restrictions on the use of term variables of left patterns in order to simplify the definition of the stochastic semantics.

Definition 16 (Patterns). Left patterns P_L and right patterns P_R of Stochastic CLS are given by the following grammar:

$$
\begin{aligned}
P_L &::= SP \quad \mid \quad \left(P_X \right)^L \rfloor P_X \quad \mid \quad P_L \mid P_L \\
P_X &::= P_L \quad \mid \quad P_L \mid X \\
P_R &::= SP \quad \mid \quad \left(P_R \right)^L \rfloor P_R \quad \mid \quad P_R \mid P_R \quad \mid \quad X \\
SP &::= \epsilon \quad \mid \quad a \quad \mid \quad SP \cdot SP \quad \mid \quad \widetilde{x} \quad \mid \quad x
\end{aligned}
$$

where $X, \widetilde{x}$ and x are generic elements of TV, SV and $\mathcal{X}$, respectively. The sets of all left and right patterns are denoted with $\mathcal{P}_L$ and $\mathcal{P}_R$, respectively.

We assume the structural congruence relation to be trivially extended to patterns. Furthermore, note that right patterns P_R are exactly the same as CLS patterns (see Definition 3) while left patterns P_L contain the following restrictions:

- term variables cannot appear in a parallel composition at the top–level of the pattern. In other words, they must always appear in an operand of some looping and containment operator. For example, $a \cdot b \mid X$ is not a syntactically correct left pattern, whereas $(a)^L \rfloor (a \cdot b \mid X)$ and $(a \cdot b \mid X)^L \rfloor a$ are correct;
- a parallel composition cannot have among its components more than one term variable which is not involved in the application of some looping and containment operator. For example, $(a)^L \rfloor (a \cdot b \mid X \mid Y)$ is not a syntactically correct term, whereas $(a)^L \rfloor (a \cdot b \mid X \mid (Y)^L \rfloor Z)$ is correct.

These restrictions simplify the definition of the stochastic semantics and the development of a simulator of Stochastic CLS models. In fact, the restrictions allow us to prove that for all $P \in \mathcal{P}_L$ and $\sigma, \sigma' \in \Sigma$, $P\sigma \equiv P\sigma'$ implies that for all $X \in TV$ it holds $\sigma(X) \equiv \sigma'(X)$. This means that the instantiation of term variables in the application of a rewrite rule to a specific portion of a term is always unique. This result is important in the definition of the stochastic semantics because, as we shall see, to compute the application rate of a rule we will need to count the number of occurrences of some sequences in the instantiation of the term variables of the rule.

The restrictions we impose do not reduce significantly the expressiveness of the formalism in the modeling of biological systems. As regards the use of term variables at top–level of a left pattern of a rule, by the definition of the semantics, such a rule could rewrite any portion of the term representing the state of the system. From a biological perspective, this would correspond to modeling a reaction with an uncertain number of reactants. Analogously, two term variables inside a looping or a containment operator of a rule could be instantiated in many different ways correspondingly to different partitions of the term. From a biological perspective, this would represent a reaction between, for instance, two arbitrary portions of the content of a membrane. Consequently, as chemical reactions happen usually between a fixed and small number of reactants, the restriction appears to be reasonable.

We now introduce stochastic rewrite rules.

Definition 17 (Stochastic Rewrite Rules). *A stochastic rewrite rule is a pair of patterns and a kinetic constant (P_1, P_2, k), denoted with $P_1 \overset{k}{\mapsto} P_2$, where $P_1 \in \mathcal{P}_L$, $P_2 \in \mathcal{P}_R$, $P_1 \not\equiv \epsilon$, $k \in \mathbb{R}$ and such that $Var(P_2) \subseteq Var(P_1)$. We denote with $\Re$ the infinite set of all the possible rewrite rules.*

As said above, we distinguish between the patterns on the left and on the right hand sides of a (stochastic) rewrite rule, and we assume some restrictions on the use of term variables on the left hand side. As examples of forbidden rules, consider $a \mid X \mapsto a$ and $(b)^L \rfloor (X \mid Y) \mapsto (b)^L \rfloor X$. The former removes

from the term to which it is applied an arbitrary number of components of a parallel composition where a appears. The latter removes an arbitrary portion of the content of any $(b)^L$ occurring in the term to which it is applied. The left hand sides of both of these rules violate the restrictions we have imposed on left patterns.

Stochastic rewrite rules will be applied with a frequency that corresponds to the rate of occurrence of chemical reactions computed by Gillespie's algorithm [24]. This means that a rule such as $a \mid b \overset{k}{\mapsto} c$, modeling the formation of a complex c as result of the binding of two molecules a and b, will be applied with a frequency proportional to k and to the number of possible combinations of elements a and b in the term to which the rule is applied. For instance, if the rule is applied to term $a \mid a \mid b \mid b \mid b$, then its application rate will be $6k$. Instead, if the rule is applied to $a \mid a \mid b \mid (c)^L \rfloor (a \mid a \mid b \mid b)$, then its application rate will be $2k$ if the rule is applied to a pair a, b at the top–level of the term, and will be $4k$ if the rule is applied to a pair a, b contained in the looping.

More complex is the case of a rule such as $a \mid (b \cdot \widetilde{x} \mid X)^L \rfloor Y \overset{k}{\mapsto} (c \cdot \widetilde{x} \mid X)^L \rfloor Y$, modeling the binding of a molecule a with a portion b of a molecule on the surface of some membrane. In this case, the frequency of application of the rule should be proportional to the number of symbols a in the term and to the number of sequences starting with b in the instantiation of X plus one (the instantiation of $b \cdot \widetilde{x}$). For instance, the rate of application of the rule to the term $a \mid a \mid a \mid (b \cdot a \mid b \cdot a)^L \rfloor c$ should be $6k$.

The application rate of a rewrite rule will be computed by the semantics of Stochastic CLS. Before defining it, we need to introduce some auxiliary functions. Let $\mathbf{n} : \mathcal{T} \times \mathcal{T} \to \mathbb{N}$ be a function such that $\mathbf{n}(T, T')$ computes the *number of occurrences* of the term T', assumed either to be a sequence or to have the form $(T_1)^L \rfloor T_2$, as a component of the parallel composition at the top–level of T. Formally,

$$\mathbf{n}(T_1 \mid T_2, T') = \mathbf{n}(T_1, T') + \mathbf{n}(T_2, T')$$

$$\mathbf{n}(S, S') = \begin{cases} 1, & \text{if } S \equiv S' \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{n}((T_1)^L \rfloor T_2, (T_1')^L \rfloor T_2') = \begin{cases} 1, & \text{if } T_1 \equiv T_1' \text{ and } T_2 \equiv T_2' \\ 0, & \text{otherwise} \end{cases}$$

For instance, $\mathbf{n}(a \cdot b \mid a \cdot b, a \cdot b) = 2$ and $\mathbf{n}((m)^L \rfloor a \mid (m)^L \rfloor b, (m)^L \rfloor a) = 1$. Furthermore, let us denote with $\overline{T}$ the set containing all the sequences and all the membranes (with their content) appearing at top–level in T. For instance, if $T \equiv a \mid a \mid (a)^L \rfloor b \mid (a \mid a)^L \rfloor b$, then $\overline{T} = \{a, (a)^L \rfloor b, (a \mid a)^L \rfloor b\}$. Let $comb : \mathcal{P}_L \times \Sigma \to \mathbb{N}$ be a function which, given a left pattern P_L and an instantiation function σ, computes the number of *combinations of reactants* of

P_L in $P_L\sigma$. The function *comb* can be recursively defined as follows:

$$comb(SP, \sigma) = 1$$
$$comb(P_{L1} \mid P_{L2}, \sigma) = comb(P_{L1}, \sigma) \cdot comb(P_{L2}, \sigma)$$
$$comb\big((P_{X1})^L \rfloor P_{X2}, \sigma\big) = comb'(P_{X1}, \sigma) \cdot comb'(P_{X2}, \sigma)$$

where *comb'* is defined as follows:

$$comb'(P_L, \sigma) = comb(P_L, \sigma)$$
$$comb'(P_L \mid X, \sigma) = \prod_{T \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid \sigma(X), T)}{\mathbf{n}(P_L\sigma, T)} \cdot comb(P_L, \sigma) \, .$$

The reactants of a left pattern are the sequence patterns it contains. As a consequence, for a single sequence pattern, *comb* is 1. For the parallel composition of two left patterns *comb* is the product of the values of *comb* for the two components. For looping and containment of two patterns, note that the restriction we have imposed on left patterns ensures that in each pattern there may be at most one term variable. Therefore, one must count the occurrences of reactant combinations in the instantiations of such variables. This implies the computation of a binomial coefficient for each distinct reactant, and the result of this computation is given by *comb'*. If in the considered pattern there are no term variables, then *comb'* is equal to *comb*. If a term variable is present, then *comb'* gives the product of all binomial coefficients for each distinct reactant, namely for each element of the set $P_L\sigma$, multiplied by the number of combinations of reactants which may result from P_L.

For instance, given $\sigma = \{(X, b \mid c), (Y, a \mid c \mid c)\}$ and $P_L \equiv a \mid a \mid \big(b \mid X\big)^L \rfloor (c \mid Y)$, we have $comb(P_L, \sigma) = 1 \cdot 1 \cdot comb'(b \mid X, \sigma) \cdot comb'(c \mid Y, \sigma) = 2 \cdot 3 = 6$.

Now we must also take into account the fact that in the context in which a rewrite rule applies there may be other reactants, and therefore the rate of application of the rule should be consequentially increased.

If we have a rewrite rule $P_L \overset{k}{\mapsto} P_R$, we have seen that we can compute the number of combinations of reactants of such a rule in $P_L\sigma$ as $comb(P_L, \sigma)$. However, if the term to which the rule is applied is $P_L\sigma \mid T$, then the number of combinations of reactans should become

$$comb(P_L, \sigma) \cdot \prod_{T' \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T, T')}{\mathbf{n}(P_L\sigma, T')} \, .$$

In fact, $\prod_{T' \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T, T')}{\mathbf{n}(P_L\sigma, T')}$ is the number of combinations of components of $P_L\sigma$ in the extended term $P_L\sigma \mid T$, that is the number of positions in $P_L\sigma \mid T$ where a rule with left hand side P_L can be applied.

As an example, if $P_L \equiv (a)^L \rfloor (b \mid X), \sigma(X) = b$ and $T \equiv (a)^L \rfloor (b \mid b) \mid c$, then $P_L\sigma \mid T$ is $(a)^L \rfloor (b \mid b) \mid (a)^L \rfloor (b \mid b) \mid c$ and the number of combinations

of reactants of P_L in $P_L\sigma \mid T$ is $comb(P_L, \sigma) \cdot \prod_{T' \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T, T')}{\mathbf{n}(P_L\sigma, T')} = 2 \cdot \binom{2}{1} = 4$. In fact, in $P_L \mid T$ there are four pairs of reactants a and b to which a rule with P_L on the left hand side could be applied.

Our aim is to compute compositionally this increase of the rate of application of a rule due to some reactant that is in the context of the portion of the term to which the rule applies. To this purpose we define, as follows, an auxiliary function $binom : \mathcal{T} \times \mathcal{T} \times \mathcal{T} \to \mathbb{Q}$:

$$binom(T_1, T_2, T_3) = \prod_{T \in \overline{T_1}} \prod_{i=1}^{\mathbf{n}(T_3, T)} \frac{\mathbf{n}(T_2, T) + i}{\mathbf{n}(T_2, T) - \mathbf{n}(T_1, T) + i}.$$

The following two propositions show that $binom$ can be used to compute the increase of the application rate we are interested in, and that such and increase can be computed compositionally.

Proposition 1. *Given $P_L \in \mathcal{P}_L, T \in \mathcal{T}$ and $\sigma \in \Sigma$ it holds*

$$binom(P_L\sigma, P_L\sigma, T) = \prod_{T' \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T, T')}{\mathbf{n}(P_L\sigma, T')}.$$

Proof. By the definition of *binom* it follows

$$binom(P_L\sigma, P_L\sigma, T) = \prod_{T' \in \overline{P_L\sigma}} \prod_{i=1}^{\mathbf{n}(T,T')} \frac{\mathbf{n}(P_L\sigma, T') + i}{\mathbf{n}(P_L\sigma, T') - \mathbf{n}(P_L\sigma, T) + i} =$$

$$= \prod_{T' \in \overline{P_L\sigma}} \frac{(\mathbf{n}(P\sigma, T') + \mathbf{n}(T, T')) \cdot (\mathbf{n}(P\sigma, T') + \mathbf{n}(T, T') - 1) \cdots (\mathbf{n}(P\sigma, T') + 1)}{\mathbf{n}(T, T')!}$$

since $\mathbf{n}(P_L\sigma, T') + \mathbf{n}(T, T') = \mathbf{n}(P_L\sigma \mid T, T')$ this is equal to

$$\prod_{T' \in \overline{P_L\sigma}} \frac{\mathbf{n}(P_L\sigma \mid T, T') \cdot (\mathbf{n}(P_L\sigma \mid T, T') - 1) \cdots (\mathbf{n}(P_L\sigma \mid T, T') - \mathbf{n}(T, T') + 1)}{\mathbf{n}(T, T')!} =$$

$$= \prod_{T' \in \overline{P_L\sigma}} \frac{\mathbf{n}(P_L\sigma \mid T, T')!}{\mathbf{n}(T, T')! \cdot (\mathbf{n}(P_L\sigma \mid T, T') - \mathbf{n}(T, T'))!} = \prod_{T' \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T, T')}{\mathbf{n}(T, T')}$$

and, by the well–known property of binomial coefficients such that $\binom{n}{k} = \binom{n}{n-k}$, this is equal to

$$\prod_{T' \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T, T')}{\mathbf{n}(P_L\sigma, T')}.$$

$\square$

Proposition 2. *Given $P_L \in \mathcal{P}_L, T_1, T_2 \in \mathcal{T}$ and $\sigma \in \Sigma$ it holds*

$$binom(P_L\sigma, P_L\sigma, T_1 \mid T_2) = binom(P_L\sigma, P_L\sigma, T_1) \cdot binom(P_L\sigma, P_L\sigma \mid T_1, T_2).$$

Proof. By Proposition 1 the property we have to prove can be rewritten as

$$\prod_{T \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T_1 \mid T_2, T)}{\mathbf{n}(P_L\sigma, T)} = \prod_{T \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T_1, T)}{\mathbf{n}(P_L\sigma, T)} \cdot binom(P_L\sigma, P_L\sigma \mid T_1, T_2) =$$

$$= \prod_{T \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T_1, T)}{\mathbf{n}(P_L\sigma, T)} \cdot \prod_{T \in \overline{P_L\sigma \mid T_1}} \prod_{i=1}^{\mathbf{n}(T_2, T)} \frac{\mathbf{n}(P_L\sigma \mid T_1, T) + i}{\mathbf{n}(P_L\sigma \mid T_1, T) - \mathbf{n}(P_L\sigma, T) + i}$$

Now, let $n, m, k \in \mathbb{N}$ be such that $k \leq n$. The following equation holds:

$$\binom{n+m}{k} = \frac{(n+m)!}{k!(n+m-k)!} = \frac{(n+m)\cdots(n+1)\cdot n \cdots (k+1)}{(n+m-k)!} =$$

$$= \frac{(n+m)\cdots(n+1)}{(n+m-k)\cdots(n+1-k)} \cdot \frac{n \cdots (k+1)}{(n-k)!} = \left(\prod_{i=1}^{m} \frac{n+i}{n+i-k}\right) \cdot \binom{n}{k}.$$

This resuls and the fact that $\mathbf{n}(P_L\sigma \mid T_1 \mid T_2, P_L\sigma) = \mathbf{n}(P_L\sigma \mid T_1, P_L\sigma) + \mathbf{n}(T_2, P_L\sigma)$, are used in what follows:

$$\prod_{T \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T_1 \mid T_2, T)}{\mathbf{n}(P_L\sigma, T)} = \prod_{T \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T_1, T) + \mathbf{n}(T_2, T)}{\mathbf{n}(P_L\sigma, T)} =$$

$$= \prod_{T \in \overline{P_L\sigma}} \left(\binom{\mathbf{n}(P_L\sigma \mid T_1, T)}{\mathbf{n}(P_L\sigma, T)} \cdot \prod_{i=1}^{\mathbf{n}(T_2, T)} \frac{\mathbf{n}(P_L\sigma \mid T_1, T) + i}{\mathbf{n}(P_L\sigma \mid T_1, T) - \mathbf{n}(P_L\sigma, T) + i} \right) =$$

$$= \left(\prod_{T \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T_1, T)}{\mathbf{n}(P_L\sigma, T)} \right) \cdot \left(\prod_{T \in \overline{P_L\sigma}} \prod_{i=1}^{\mathbf{n}(T_2, T)} \frac{\mathbf{n}(P_L\sigma \mid T_1, T) + i}{\mathbf{n}(P_L\sigma \mid T_1, T) - \mathbf{n}(P_L\sigma, T) + i} \right) =$$

$$= \left(\prod_{T \in \overline{P_L\sigma}} \binom{\mathbf{n}(P_L\sigma \mid T_1, T)}{\mathbf{n}(P_L\sigma, T)} \right) \cdot binom(P_L\sigma, P_L\sigma \mid T_1, T_2).$$

$\square$

As an example, if we assume $P_L \equiv a \mid b$ and $\sigma = \varnothing$, we have $comb(P_L, \sigma) = 1$. Let us consider the terms $T_1 = a \mid c$ and $T_2 = a \mid b$. We have that the number of combinations of reactants of P_L in $P_L\sigma \mid T_1 \mid T_2$ is $comb(P_L, \sigma) \cdot \prod_{T \in \{a,b\}} \binom{\mathbf{n}(a|b|a|c|a|b, T)}{\mathbf{n}(a|b, T)} = \binom{3}{1}\binom{2}{1} = 6$. By applying *binom* compositionally, we obtain $comb(P_L, \sigma) \cdot binom(a \mid b, a \mid b, a \mid c) \cdot binom(a \mid b, a \mid b \mid a \mid c, a \mid b) = 1 \cdot 2 \cdot 3 = 6$.

The stochastic semantics of Stochastic CLS can be defined as follows.

Definition 18 (Semantics). *Given a finite set of stochastic rewrite rules $\mathcal{R}$, let $\xrightarrow{R,T,r,b}$, with $R \in \mathcal{R}, T \in \mathcal{T}, r \in \mathcal{R}$ and $b \in \mathbb{Q}$, be the least labeled transition relation on terms closed with respect to $\equiv$ and satisfying the following inference*

rules:

1. $$\dfrac{R : P_L \overset{k}{\mapsto} P_R \in \mathcal{R} \quad \sigma \in \Sigma \quad P_L\sigma \not\equiv P_R\sigma}{P_L\sigma \xrightarrow{R, P_L\sigma, k \cdot comb(P_L, \sigma), 1} P_R\sigma}$$

2. $$\dfrac{T_1 \xrightarrow{R,T,r,b} T_2}{T_1 \mid T_3 \xrightarrow{R,T,r,b \cdot binom(T,T_1,T_3)} T_2 \mid T_3}$$

3. $$\dfrac{T_1 \xrightarrow{R,T,r,b} T_2}{\left(T_1\right)^L \rfloor T_3 \xrightarrow{R,\left(T_1\right)^L \rfloor T_3, r \cdot b, 1} \left(T_2\right)^L \rfloor T_3}$$

4. $$\dfrac{T_1 \xrightarrow{R,T,r,b} T_2}{\left(T_3\right)^L \rfloor T_1 \xrightarrow{R,\left(T_3\right)^L \rfloor T_1, r \cdot b, 1} \left(T_3\right)^L \rfloor T_2}$$

The semantics of Stochastic CLS is the least labeled transition relation on terms
$\xrightarrow{R,r}$, *with $R \in \mathcal{R}$ and $r \in \mathbb{R}$, satisfying*

$$\dfrac{T_1 \xrightarrow{R,T,r,b} T_2}{T_1 \xrightarrow{R, r \cdot b} T_2}$$

The semantics is defined as a labeled transition system where each transition $T_1 \xrightarrow{R,r} T_2$ represents the application of the stochastic rewrite rule R to a subterm of T_1 yielding term T_2, and where $r \in \mathbb{R}$ specifies the application rate.

The definition uses an auxiliary transition relation, whose transitions are of the form $T_1 \xrightarrow{R,T,r,b} T_2$. The following labels appear on transitions:

- R, that is the rewrite rule applied and which is used to distinguish between two transitions corresponding to the application of different rules, which have the same application rate and take to the same result;
- T, that is either the instantiation of the left hand side of rule R or a term consisting of a looping and containment operator applied to a term which contains the instantiation of the left hand side of R;
- r, that is the application rate of R to T;
- b, that is the number of occurrences of T in the term which is transformed, with the exclusion of occurrences inside looping and containment operators.

The auxiliary transition relation $\xrightarrow{R,T,r,b}$ is defined by four inference rules. Rule 1 allows to derive a transition corresponding to the instantiation of a rewrite rule. Its rate is obtained by multiplying the kinetic constant of the rule, k, with the number of different reactant combinations appearing in the instantiation of the left pattern, namely the result of applying the function *comb*. Transitions which correspond to rule applications that do not modify terms are excluded by the condition $P_L\sigma \not\equiv P_R\sigma$. Rule 2 uses the function *binom* to compute the number of different reactant combinations for the state $T_1 \mid T_3$, by assuming

that b is the number of different reactant combinations for the only term T_1. Finally, rules 3 and 4 deal with looping and containment. In both the rules, the second label of the premises, namely T, is substituted by the label $(T_1)^L \rfloor T_3$ or $(T_3)^L \rfloor T_1$, respectively. This because, if the term is furtherly composed in parallel with other terms, *binom* should count the occurrences of $(T_1)^L \rfloor T_3$ or $(T_3)^L \rfloor T_1$ rather than of the sole T.

The main transition relation $\xrightarrow{R,r}$ of the semantics is obtained by from the auxiliary relation by removing from all transitions the second label T, used only to compositionally compute the application rate of the rule, and by multiplying, and by multiplying their third and fourth labels r and b, so to obtain the total application rate.

As a simple example of application of the semantics, let us consider a Stochastic CLS model consisting of the term

$$ T \equiv (m)^L \rfloor (a \mid a) \mid (m)^L \rfloor (a \mid a) \mid a \cdot b \mid a $$

and of the rewrite rules

$$ R_1 : (m)^L \rfloor X \overset{k_1}{\mapsto} \epsilon \qquad R_2; a \cdot b \cdot \widetilde{x} \overset{k_2}{\mapsto} a \mid b \mid \widetilde{x} \qquad R_3 : a \overset{k_3}{\mapsto} b \,. $$

As regards rule R_1, it is possible to derive, given an instantiation function $\sigma = \{(X, a \mid a)\}$, the following transition of the auxiliary relation of the semantics:

$$ \cfrac{\cfrac{R_1 : (m)^L \rfloor X \overset{k_1}{\mapsto} \epsilon \quad \sigma = \{(X, a \mid a)\}}{(m)^L \rfloor (a \mid a) \xrightarrow{R_1, (m)^L \rfloor (a\mid a), k_1, 1} \epsilon}}{T \xrightarrow{R_1, (m)^L \rfloor (a\mid a), k_1, 2} (m)^L \rfloor (a \mid a) \mid a \cdot b \mid a} $$

Note that the transition from state T has the forth label equal to 2 because $binom((m)^L \rfloor (a \mid a), (m)^L \rfloor (a \mid a), (m)^L \rfloor (a \mid a) \mid a \cdot b \mid a) = 2$.

Finally, the corresponding main transition is

$$ T \xrightarrow{R_1, 2 \cdot k_1} (m)^L \rfloor (a \mid a) \mid a \cdot b \mid a \,. $$

Note that, as term T contains two occurrences of $(m)^L \rfloor (a \mid a)$, then the rule is correctly applied with rate $2 \cdot k_1$.

Differently, as regards rule R_2, given an instantiation function $\sigma = \{(\widetilde{x}, \epsilon)\}$, it is possible to derive, after the application of the auxiliary transition relation $\xrightarrow{R,T,r,b}$, the transition

$$ T \xrightarrow{R_2, k_2} (m)^L \rfloor (a \mid a) \mid (m)^L \rfloor (a \mid a) \mid a \mid b \mid a \,. $$

Also in this case the value computed as rate is correct because T contains just one sequence $a \cdot b$.
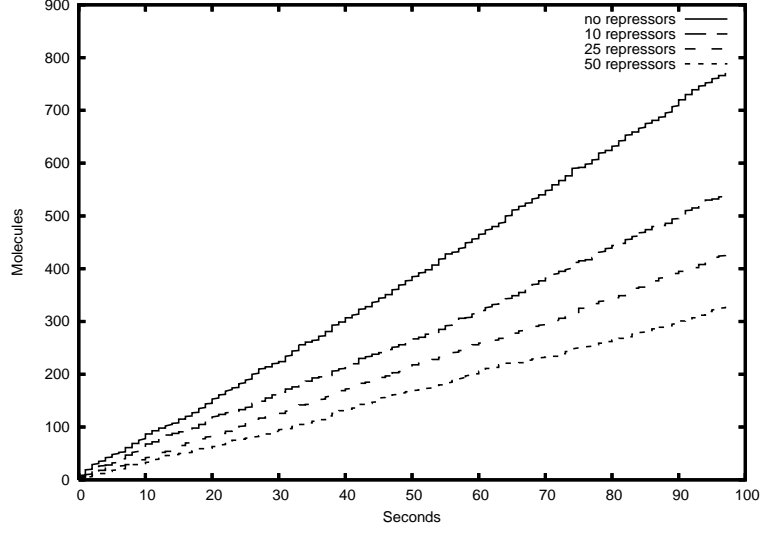
Fig. 2. Simulation result of the regulation process: number of RNA molecules over time.

Finally, as regards rule R_3, it is possible to derive the following transition:

$$\dfrac{\dfrac{\dfrac{\dfrac{R_3 : a \overset{k_3}{\mapsto} b \quad \sigma = \emptyset}{a \xrightarrow{R_3,a,k_3,1} b}}{a \mid a \xrightarrow{R_3,a,k_3,2} a \mid b}}{\big(m\big)^L \rfloor (a \mid a) \xrightarrow{R_3,\big(m\big)^L \rfloor (a|a),2\cdot k_3,1} \big(m\big)^L \rfloor (a \mid b)}}{T \xrightarrow{R_3,\big(m\big)^L \rfloor (a|a),2\cdot k_3,2} \big(m\big)^L \rfloor (a \mid b) \mid \big(m\big)^L \rfloor (a \mid a) \mid a \cdot b \mid a}$$
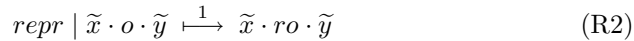
As expected, the semantics of Stochastic CLS gives the transition

$$T \xrightarrow{R_3,4\cdot k_3} \big(m\big)^L \rfloor (a \mid b) \mid \big(m\big)^L \rfloor (a \mid a) \mid a \cdot b \mid a$$

where the rate is $4 \cdot k_3$ because there are 4 different sequences a to which the rule R_3 can be applied.

Example Let us consider the simple regulation process we modeled with CLS in Section 3.1. We now extend the CLS model by including a kinetic constant in each rewrite rule. The result is a Stochastic CLS model. In order to make the model a little more realistic we add two rewrite rules describing the unbinding of the RNA polymerase and of the repressor from the DNA. Hence, the rewrite

rules of the Stochastic CLS model are the following:

$$polym \mid p \cdot \widetilde{x} \overset{0.1}{\longmapsto} pp \cdot \widetilde{x} \tag{R1}$$

$$pp \cdot \widetilde{x} \overset{2}{\longmapsto} polym \mid p \cdot \widetilde{x} \tag{R1'}$$

$$repr \mid \widetilde{x} \cdot o \cdot \widetilde{y} \overset{1}{\longmapsto} \widetilde{x} \cdot ro \cdot \widetilde{y} \tag{R2}$$

$$\widetilde{x} \cdot ro \cdot \widetilde{y} \overset{10}{\longmapsto} repr \mid \widetilde{x} \cdot o \cdot \widetilde{y} \tag{R2'}$$

$$pp \cdot o \cdot \widetilde{x} \overset{100}{\longmapsto} p \cdot po \cdot \widetilde{x} \tag{R3}$$

$$\widetilde{x} \cdot po \cdot g \overset{100}{\longmapsto} \widetilde{x} \cdot o \cdot pg \tag{R4}$$

$$\widetilde{x} \cdot pg \overset{30}{\longmapsto} polym \mid rna \mid \widetilde{x} \cdot g \tag{R5}$$
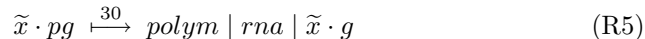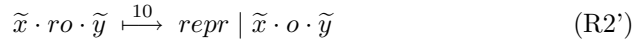
We have developed a simulator based on Stochastic CLS, and we used it to study the behaviour of the regulation process. In particular, we performed simulations by varying the quantity of repressors and we observed the production of RNA fragments in each case. The initial configuration of the system is given by the following term

$$\underbrace{repr \mid \ldots \mid repr}_{n} \mid \underbrace{polym \mid \ldots \mid polym}_{100} \mid p \cdot o \cdot g$$

and we performed simulations with $n = 0, 10, 25$ and 50. The results of the simulations are shown in Figure 2. By varying the number of repressors from 0 to 50 the rate of transcription of the DNA into RNA molecules decreases.

4 Application Examples

We present two biological systems modeled with Stochastic CLS. The first describes the regulation process of the lactose operon in *Escherichia coli*. The second describes an example of *quorum sensing*, namely the ability of bacteria of monitoring their population density and modulating their gene expressions according to this density. We show the results of simulations of both models.

4.1 Application to the Modeling of Metabolic Pathways

We give a Stochastic CLS model of the well–known regulation process of the lactose operon in *Escherichia coli*.

The lactose operon is a sequence of genes that are responsible for producing three enzymes for lactose degradation, namely the *lactose permease*, which is incorporated in the membrane of the bacterium and actively transports the sugar into the cell, the *beta galactosidase*, which splits lactose into glucose and galactose, and the *transacetylase*, whose role is marginal.

The first three genes of the operon (i,p,o) regulate the production of the enzymes, and the last three (z, y, a), called *structural genes*, are transcribed
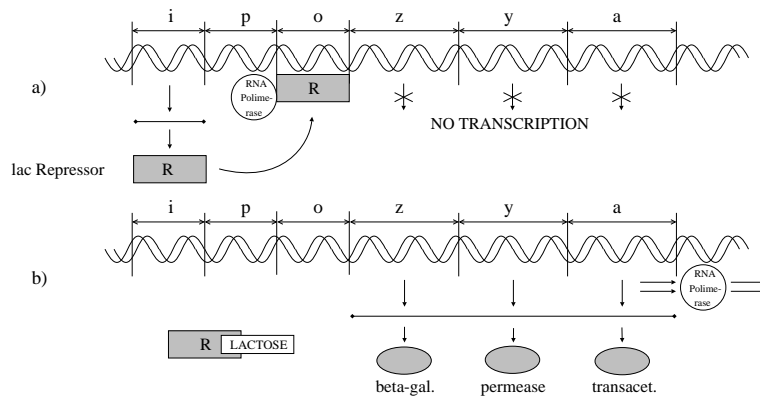
Fig. 3. The regulation process in the Lac Operon.

(when allowed) 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 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 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.

A detailed mathematical model of the regulation process can be found in [46]. It includes information on the influence of lactose degradation on the growth of the bacterium.

We give a Stochastic CLS model of the gene regulation process, with stochastic rates taken from [45]. 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 modeled 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
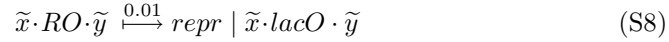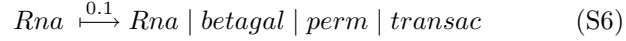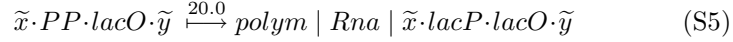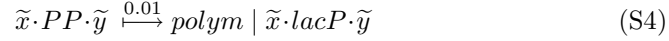
the three structural genes are transcribed into a single mRNA fragment, we model such mRNA as a single symbol Rna.

The initial state of the bacterium when no lactose is present in the environment and when 100 molecules of lactose are present are modeled by the following terms (where $n \times T$ stands for a parallel composition $T \mid \ldots \mid T$ of length n):
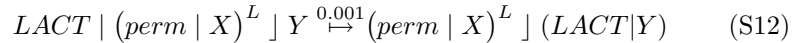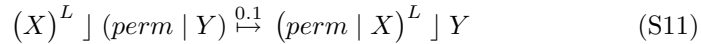
$$Ecoli \ ::= \ \big(m\big)^L \rfloor (lacI{-}A \mid 30 \times polym \mid 100 \times repr) \tag{1}$$

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

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

$$lacI{\cdot}\widetilde{x} \ \overset{0.02}{\longmapsto} \ lacI{\cdot}\widetilde{x} \mid Irna \tag{S1}$$

$$Irna \ \overset{0.1}{\longmapsto} \ Irna \mid repr \tag{S2}$$

$$polym \mid \widetilde{x}{\cdot}lacP{\cdot}\widetilde{y} \ \overset{0.1}{\longmapsto} \ \widetilde{x}{\cdot}PP{\cdot}\widetilde{y} \tag{S3}$$

$$\widetilde{x}{\cdot}PP{\cdot}\widetilde{y} \ \overset{0.01}{\longmapsto} \ polym \mid \widetilde{x}{\cdot}lacP{\cdot}\widetilde{y} \tag{S4}$$

$$\widetilde{x}{\cdot}PP{\cdot}lacO{\cdot}\widetilde{y} \ \overset{20.0}{\longmapsto} \ polym \mid Rna \mid \widetilde{x}{\cdot}lacP{\cdot}lacO{\cdot}\widetilde{y} \tag{S5}$$

$$Rna \ \overset{0.1}{\longmapsto} \ Rna \mid betagal \mid perm \mid transac \tag{S6}$$

$$repr \mid \widetilde{x}{\cdot}lacO{\cdot}\widetilde{y} \ \overset{1.0}{\longmapsto} \ \widetilde{x}{\cdot}RO{\cdot}\widetilde{y} \tag{S7}$$

$$\widetilde{x}{\cdot}RO{\cdot}\widetilde{y} \ \overset{0.01}{\longmapsto} \ repr \mid \widetilde{x}{\cdot}lacO \cdot \widetilde{y} \tag{S8}$$

$$repr \mid LACT \ \overset{0.005}{\longmapsto} \ RLACT \tag{S9}$$

$$RLACT \ \overset{0.1}{\longmapsto} \ repr \mid LACT \tag{S10}$$

Rules (S1) and (S2) 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 (S3) and (S4) describe binding and unbinding of the RNA polymerase to gene p. Rules (S5) and (S6) 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 (S7) and (S8) describe binding and unbinding of the lac Repressor to gene o. Finally, rules (S9) and (S10) 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:
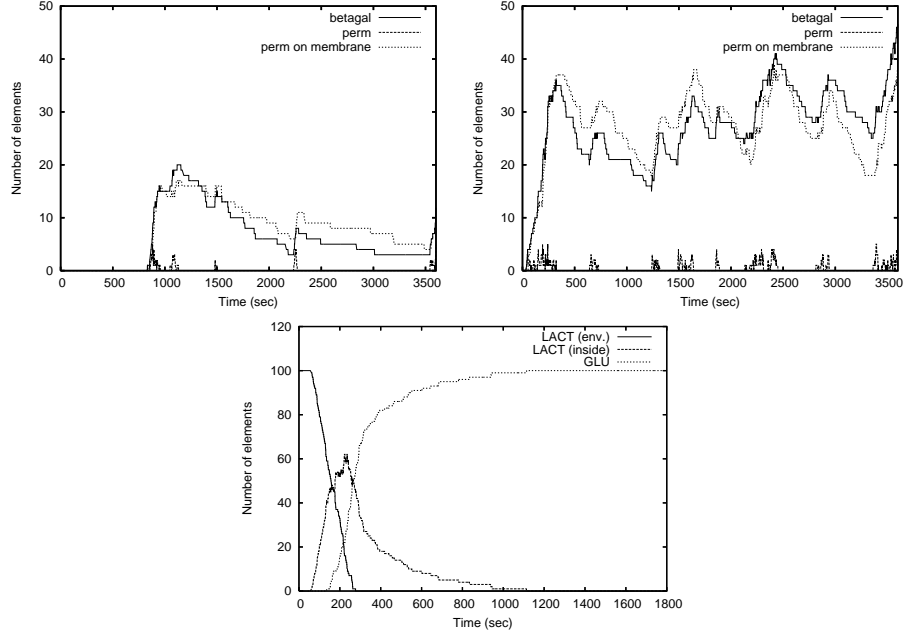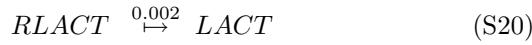
$$\big(X\big)^L \rfloor (perm \mid Y) \overset{0.1}{\mapsto} \big(perm \mid X\big)^L \rfloor Y \tag{S11}$$

$$LACT \mid \big(perm \mid X\big)^L \rfloor Y \overset{0.001}{\mapsto} \big(perm \mid X\big)^L \rfloor (LACT|Y) \tag{S12}$$

$$betagal \mid LACT \overset{0.001}{\mapsto} betagal \mid GLU \mid GAL \tag{S13}$$

Fig. 4. Simulation results: production of enzymes in the absence (left) and presence (center) of lactose, and degradation of lactose into glucose (right).

Rule (S11) describes the incorporation of the lactose permease in the membrane of the bacterium, rule (S12) the transportation of lactose from the environment to the interior performed by the lactose permease, and rule (S13) 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:

$$perm \overset{0.001}{\mapsto} \epsilon \qquad lrna \overset{0.001}{\mapsto} \epsilon \qquad transac \overset{0.001}{\mapsto} \epsilon \qquad \text{(S14-S16)}$$

$$repr \overset{0.002}{\mapsto} \epsilon \qquad betagal \overset{0.01}{\mapsto} \epsilon \qquad Rna \overset{0.01}{\mapsto} \epsilon \qquad \text{(S17-S19)}$$

$$RLACT \overset{0.002}{\mapsto} LACT \qquad \text{(S20)}$$

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

In Figure 4 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. 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. Once some molecules of lactose permease join the membrane, the lactose starts entering the bacterium and being transformed into glucose (see the third graph).

4.2 Application to the Modeling of 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 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 [43]. Another example is given by the bacterium *Pseudomonas aeruginosa*, a prevalent human pathogen [20]. 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. In this paper we are interested in the one regulating the expression of elastase LasB, named the *las* system. The two enzymes, LasB elastase and LasA elastase, are responsible for pulmonary hemorrhages associated with *P. aeruginosa* infections.
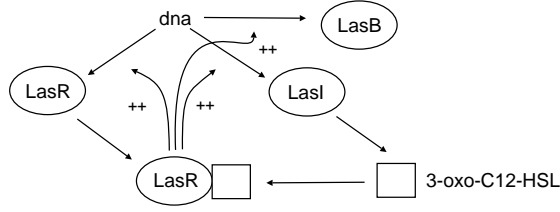
Fig. 5. A schematic description of the las system in *P. aeruginosa*.

A schematic description of the *las system* is shown in Fig.5. The autoinducer 3-oxo-C12-HSL and the transcriptional activator protein LasR are produced at basal rates. The LasR/3-oxo-C12-HSL dimer is the activated form of LasR. It 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.
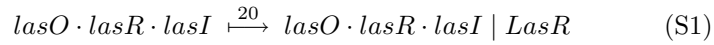
We now give the Stochastic CLS model of the quorum sensing process. We do not model the production of the LasB as it has no active role in the regulation process. The initial state of each bacterium is:

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

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, respectively.

This model shows one of the advantages of using terms for describing the structure of biological systems in Stochastic CLS. 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 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.
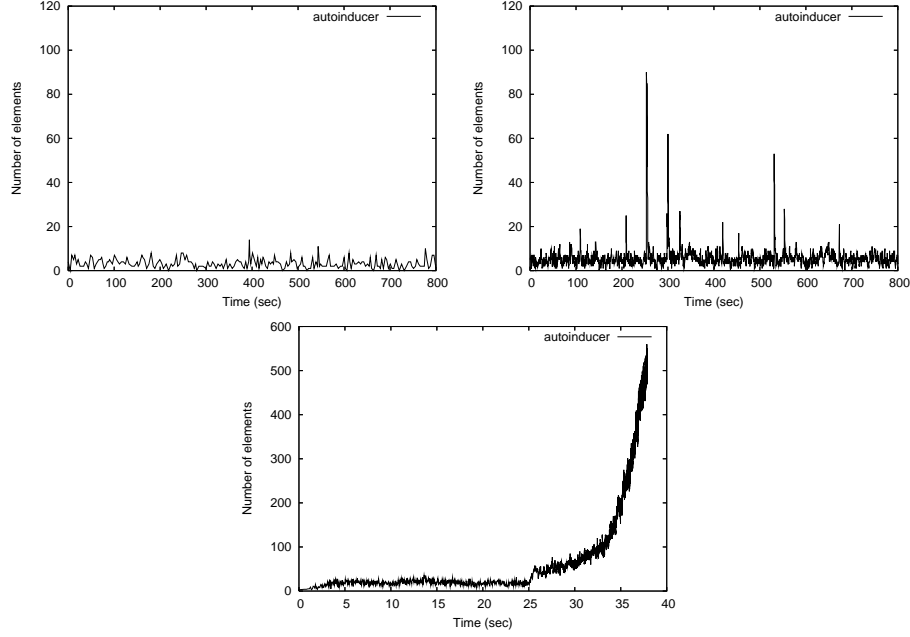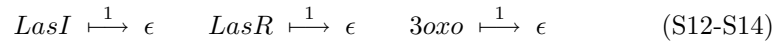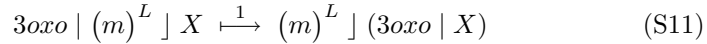
$$lasO \cdot lasR \cdot lasI \ \overset{20}{\longmapsto} \ lasO \cdot lasR \cdot lasI \mid LasR \qquad \text{(S1)}$$

$$lasO \cdot lasR \cdot lasI \ \overset{5}{\longmapsto} \ lasO \cdot lasR \cdot lasI \mid LasI \qquad \text{(S2)}$$

$$LasI \ \overset{8}{\longmapsto} \ LasI \mid 3oxo \qquad \text{(S3)}$$

$$3oxo \mid LasR \ \overset{0.25}{\longmapsto} \ 3R \qquad \text{(S4)}$$

$$3R \ \overset{400}{\longmapsto} \ 3oxo \mid LasR \qquad \text{(S5)}$$

$$3R \mid lasO \cdot lasR \cdot lasI \ \overset{0.25}{\longmapsto} \ 3RO \cdot lasR \cdot lasI \qquad \text{(S6)}$$
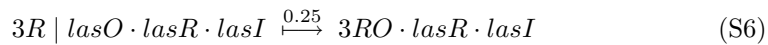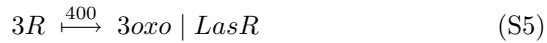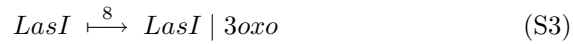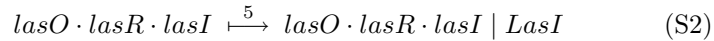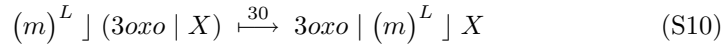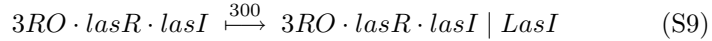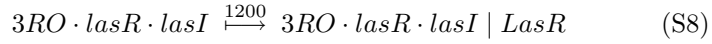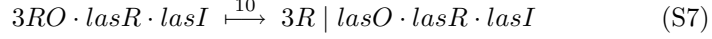
Fig. 6. Simulation results: quantity of autoinducer inside one bacterium in a population of one (left), five (center) and twenty (right) bacteria.

$$3RO \cdot lasR \cdot lasI \xmapsto{10} 3R \mid lasO \cdot lasR \cdot lasI \qquad \text{(S7)}$$

$$3RO \cdot lasR \cdot lasI \xmapsto{1200} 3RO \cdot lasR \cdot lasI \mid LasR \qquad \text{(S8)}$$

$$3RO \cdot lasR \cdot lasI \xmapsto{300} 3RO \cdot lasR \cdot lasI \mid LasI \qquad \text{(S9)}$$

$$\left(m\right)^{L} \rfloor (3oxo \mid X) \xmapsto{30} 3oxo \mid \left(m\right)^{L} \rfloor X \qquad \text{(S10)}$$

$$3oxo \mid \left(m\right)^{L} \rfloor X \xmapsto{1} \left(m\right)^{L} \rfloor (3oxo \mid X) \qquad \text{(S11)}$$

$$LasI \xmapsto{1} \epsilon \qquad LasR \xmapsto{1} \epsilon \qquad 3oxo \xmapsto{1} \epsilon \qquad \text{(S12-S14)}$$

Rules (S1) and (S2) 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 (S3) describes the production of the autoinducer 3-oxo-C12-HSL, denoted $3oxo$, performed by the LasI enzyme. Rules (S4) and (S5) describe the complexation and decomplexation of the autoinducer and the LasR protein, where the complex is denoted $3R$. Rules from (S7) to (S9) describe the binding of the activated autoinducer to the DNA and its influence in the production of LasR and LasI. Rules (S10) and (S11) 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 (S12) to (S14) describe the degradation of proteins.

We have simulated the behavior of a population of *P. aeruginosa* by varying the number of individuals. In Figure 6 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 (S10) and (S11) 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 kinetic constants of (S10) and (S11) would cause the quorum sensing to be triggered when the number of individuals is bigger.

5   Conclusion and Future Perspectives

We have presented the Calculus of Looping Sequences (CLS) suitable to describe microbiological systems and their evolution. Terms of the calculus are constructed by basic constituent elements and operators of sequencing, looping, containment and parallel composition. The looping operator, together with the containment one, permits the description of arbitrarily nested membranes and of sequences on the surface and inside the membranes themselves. The evolution of a term is modeled by a set of rewrite rules and the semantics is a transition system, in which states correspond to terms, and transitions correspond to rules applications. We have presented two extensions, CLS with links (LCLS) and Stochastic CLS. LCLS allows the description of protein interaction at a lower level of abstraction, namely at the domain level. Stochastic CLS allows the description of quantitative aspects of the modeled systems, such as the frequency of chemical reactions. As examples of application to real biological systems, we have shown the simulation of the activity of the lactose operon in *E.coli* and the quorum sensing process in *P.aeruginosa*, both described with Stochastic CLS.

The CLS formalism, with its extensions LCLS and Stochastic CLS, fulfill the wanted request of having a simple notation, having the ability of describing biolgical systems at different abstraction levels, and being flexible enough to allow the description of new kinds of phenomena. In fact rewrite rules as those of CLS are more similar to the reaction notation employed by biologists and a model may be understood easier than models based, for instance, on process calculi. CLS basic elements and operators allow us to choose the level of detail of a model. As an example a cell may be described either as a single alphabet symbol or as the application of looping and containment operator to a term describing the membrane and to a term describing the cell content. Another

example may be the model of a DNA strand which can be described either as a single alphabet symbol or as a sequence of symbols representing genes or as a sequence of symbols representing nucleic acids. Finally, the absence of constraints on rewrite rules allows new phenomena to be described easily.

We are presently working on another extension of CLS, called Topological CLS (TCLS), that integrates space and time into CLS. The aim of TCLS is to enable a more accurate description of those biological processes whose behaviour depends on the exact position of the elements. In particular, TCLS allows the description of the position of biological elements, and of space they take up in a 2D/3D space. The elements may move autonomously during the passage of time, and may interact when constraints on their positions are met. In particular, rewrite rules are extended with a function that constrains application of the rule depending on the exact positions of the elements involved. Similarly to Stochastic CLS, rewrite rules are also endowed with a reaction rate. Moreover, we model the space occupied by each object as a hard sphere, hence space conflicts may arise during the evolution. These conflicts are resolved by an appropriate algorithm, which rearranges the position of the elements by assuming that they push each other when they are too close.

The development of a simulator based on the Stochastic CLS has suggested the introduction of an intermediate language for the simulation of biological systems. Such a language, the Stochastic String MultiSet Rewriting (sSMSR) [2, 3], is based on multiset rewriting. Multiset elements are strings built over a given alphabet and the state of a sSMSR system is represented by a multiset of strings. The evolution of a multiset is modeled by rewrite rules which rewrite multisets. Rules may contain variables that can be used to match either individual symbols or portions of the strings which are involved in the application of a rule. Furthermore rules can contain two different maching operators. The first allows a rule to be applicable to a multiset of strings only if such a multiset contains a single string with a certain prefix (unique matching), the second applies only if all the strings with the same given prefix are involved in the rule application (maximal matching). Finally, we have that fresh symbols, namely symbols that are present neither in the multiset of strings to which the rule is applied nor in the applied rewrite rule, can be generated when a rewrite rule is applied. The use of strings as multiset elements and of operations on strings in rewrite rules allows the development of a simulator for sSMSR based on efficient data structures and pattern matching algoritms. The features of sSMSR can ease the translation of high level languages. In [3] we defined both the encoding of the Stochastic CLS and of the stochastic $\pi-$calculus into sSMSR.

References

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

2. Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P.: An intermediate language for the simulation of biological systems. In: From Biology to Concurrency and Back (FBTC'07). Volume 194 of ENTCS., Elsevier (2007) 19–34
3. Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P.: An intermediate language for the stochastic simulation of biological systems. Submitted for publication. Draft available at: `http://www.di.unipi.it/~milazzo/` (2008)
4. Barbuti, R., Cataudella, S., Maggiolo-Schettini, A., Milazzo, P., Troina, A.: A probabilistic model for molecular systems. Fundamenta Informaticae **67** (2005) 13–27
5. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P.: Extending the calculus of looping sequences to model protein interaction at the domain level. In: Int. Symposium on Bioinformatics Research and Applications (ISBRA'07). Volume 4463 of LNBI., Springer (2006) 638–649
6. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Tiberi, P., Troina, A.: Stochastic cls for the modeling and simulation of biological systems. Submitted for publication. Draft available at: `http://www.di.unipi.it/~milazzo/` (2008)
7. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Troina, A.: A calculus of looping sequences for modelling microbiological systems. Fundamenta Informaticae **72** (2006) 21–35
8. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Troina, A.: Bisimulation congruences in the calculus of looping sequences. In: International Colloquium on Theoretical Aspects of Computing (ICTAC'06). Volume 4281 of LNCS., Springer (2006) 93–107
9. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Troina, A.: Bisimulations in Calculi Modelling Membranes. Formal Aspects of Computing, in press
10. Brodo, L., Degano, P., Priami, C.: A stochastic semantics for bioambients. In: Parallel Computing Technologies (PaCT'07). Volume 4671 of LNCS., Springer (2007) 22–34
11. Cardelli, L.: Brane calculi. interactions of biological membranes. In: Computational Methods in Systems Biology (CMSB'04). Volume 3082 of LNCS., Springer (2005) 257–280
12. Cardelli, L., Gordon, A.: Mobile ambients. Theoretical Compututer Scence. **240**(1) (2000) 177–213
13. Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., Schachter, V.: Modeling and querying biomolecular interaction networks. Theoretical Computer Science **325**(1) (2004) 25–44
14. CLSm: simulation tool (web page). `http://www.di.unipi.it/~milazzo/biosims/`
15. Curti, M., Degano, P., Priami, C., Baldari, C.: Modelling biochemical pathways through enhanced pi-calculus. Theoretical Computer Science **325**(1) (2004) 111–140
16. Damm, W., Harel, D.: LSCs: Breathing life into message sequence charts. Formal Methods in System Design **19**(1) (2001)
17. Danos, V., Laneve, C.: Formal molecular biology. Theoretical Computer Science **325**(1) (2004) 69–110
18. Danos, V., Pradalier, S.: Projective brane calculus. In: Computational Methods in Systems Biology (CMSB'04). Volume 3082 of LNCS., Springer (2005) 134–148
19. Degano, P., Prandi, D., Priami, C., Quaglia, P.: Beta-binders for biological quantitative experiments. In: Quantitative Aspects of Programming Languages (QAPL 2006). Volume 164 of ENTCS., Elsevier (2006) 101–117
20. Delden, C.V., Iglewski, B.: Cell-to-cell signaling and Pseudomonas aeruginosa infections. Emerg. Infect. Dis. **4** (1998) 551–560

21. Efroni, S., Choen, I., Harel, D.: Toward rigorous comprehension of biological complexity: Modeling, execution and visualization of thymic t–cell maturation. Genome Research **13** (2003) 2485–2497
22. Harel, D.: Statecharts: A visual formalism for complex systems. Science of Computer Programming **8**(3) (1987) 231–274
23. Harel, D.: A grand challenge: Full reactive modeling of a multi-cellular animal. Bulletin of the EATCS **81** (2003) 226–235
24. Gillespie, D.: Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry **81** (1977) 2340–2361
25. Kam, N., Cohen, I., Harel, D.: The immune system as a reactive system: Modeling t–cell activation with Statecharts. In: Symposia on Human Centric Computing Languages and Environments (HCC'01), IEEE Computer Society (2001) 15
26. Kam, N., Harel, D., Kugler, H., Marelly, R., Pnueli, A., Hubbard, E., Stern, M.: Formal modeling of c. elegans development: A scenario-based approach. In: Computational Methods in Systems Biology (CMSB'03). Volume 2602 of LNCS., Springer (2003) 4–20
27. Kwiatkowska, M., Norman, G., Parker, D.: Probabilistic symbolic model checking with prism: a hybrid approach. Int. Journal on Software Tools for Technology Transfer **6**(2) (2004) 128–142
28. Laneve, C., Tarissan, F.: A simple calculus for proteins and cells. In: Membrane Computing and Biological Inspired Process Calculi (MeCBIC'06). Volume 171 of ENTCS., Elsevier (2007) 139–154
29. Matsuno, H., Doi, A., Nagasaki, M., Miyano, S.: Hybrid petri net representation of gene regulatory network. In: Pacific Symposium on Biocomputing, World Scientific Press (2000) 341–352
30. Milazzo, P.: Qualitative and Quantitative Formal Modeling of Biological Systems. PhD thesis, Università di Pisa (2007)
31. Păun, G.: Computing with membranes. Journal of Computer and System Sciences **61**(1) (2000) 108–143
32. Păun, G.: Membrane Computing. An Introduction. Springer (2002)
33. Pérez-Jiménez, M., Romero-Campero, F.: A study of the robustness of the egfr signalling cascade using continuous membrane systems. In: IWINAC'05. Volume 3561 of LNCS., Springer (2005) 268–278
34. Pérez-Jiménez, M., Romero-Campero, F.: P systems, a new computational modelling tool for systems biology. In: Transactions on Computational Systems Biology VI. Volume 4220 of LNBI. Springer (2006) 176–197
35. Priami, C.: Stochastic π–calculus. The Computer Journal **38**(7) (1995) 578–589
36. Priami, C., Quaglia, P.: Beta binders for biological interactions. In: Computational Methods in Systems Biology (CMSB'04). Volume 3082 of LNCS., Springer (2005) 20–33
37. Priami, C., Regev, A., Silverman, W., Shapiro, E.: Application of a stochastic name–passing calculus to representation and simulation of molecular processes. Information Processing Letters **80** (2001) 25–31
38. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer (1990)
39. P Systems: web page. `http://psystems.disco.unimib.it/`
40. Regev, A., Panina, E., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: An abstraction for biological compartments. Theoretical Computer Science **325**(1) (2004) 141–167
41. Regev, A., Shapiro, E.: Cells as computation. Nature **419** (2002) 343

42. Regev, A., Silverman, W., Shapiro, E.: Representation and simulation of biochemical processes using the pi-calculus process algebra. In: Pacific Symposium on Biocomputing, World Scientific Press (2001) 459–470
43. Stevens, A., Greenberg, E.: Quorum sensing in Vibrio fischeri: essential elements for activation of the luminescence genes. J. of Bacteriology **179** (1997) 557–562
44. Wiley, H., Shvartsman, S., Lauffenburger, D.: Computational modeling of the egf–receptor system: a paradigm for systems biology. Trends in Cell Biology **13**(1) (2003) 43–50
45. Wilkinson, D.: Stochastic Modelling for Systems Biology. Chapman & Hall/CRC (2006)
46. Wong, P., Gladney, S., Keasling, J.: Mathematical model of the lac operon: Inducer exclusion, catabolite repression, and diauxic growth on glucose and lactose. Biotechnology Progress **13** (1997) 132–143