Extending the Calculus of Looping Sequences to Model Protein Interaction at the Domain Level

Roberto Barbuti, Andrea Maggiolo–Schettini, and
Paolo Milazzo

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

Abstract. In previous papers we introduced a formalism, called Calculus of Looping Sequences (CLS), for describing biological systems and their evolution. CLS is based on term rewriting. Terms can be constructed by composing symbols of a given alphabet in sequences, which could be closed (looping) and contain other terms. In this paper we extend CLS to represent protein interaction at the domain level. Such an extension, called Calculus of Linked Looping Sequences (LCLS), is obtained by labeling alphabet symbols used in terms. Two symbols with the same label are considered to be linked. We introduce a type system to express a concept of well–formedness of LCLS terms, we give an operational semantics of the new calculus, and we show the application of LCLS to the description of a biological system.

1 Introduction

Among the formalisms that either have been applied to or have been inspired by biological systems there are automata–based models [1, 8], rewrite systems [5, 9], and process calculi [11, 10, 4]. Automata have the advantage of allowing the direct use of many verification tools such as model checkers. Rewrite systems usually allow describing biological systems with a notation that can be easily understood by biologists. On the other hand, automata–like models and rewrite systems present, in general, problems of compositionality. Compositionality allows studying the behavior of a system componentwise, and it is in general ensured by process calculi, included those used to describe biological systems.

In [2, 3] we introduced a new formalism, called Calculus of Looping Sequences (CLS for short), for describing biological systems and their evolution. CLS is based on term rewriting with some features, such as a commutative parallel composition operator, and some semantic means, such as bisimulations, that are common in process calculi. This permits to combine the simplicity of notation of rewriting systems with the advantage of a form of compositionality. Actually, in [3] we have defined bisimulation relations which are congruences with respect to the operators. This is ensured by the assumption that the same set of rewrite rules is used for terms that are composed.

CLS terms are constructed by starting from basic constituent elements and composing them by means of operators of sequencing, looping, containment and parallel composition. Sequences may represent DNA fragments and proteins, looping sequences may represent membranes, and parallel composition may represent juxtaposition.

A formalism for modelling protein interactions was developed in the seminal paper by Danos and Laneve [5], and extended in [6]. This formalism allows expressing proteins by a node with a fixed number of domains; binding between domains allows complexating proteins. In this work we extend CLS to represent protein interaction at the domain level. Such an extension, called Calculus of Linked Looping Sequences (LCLS), is obtained by labelling elements of sequences. Two elements with the same label are considered to be linked.

We introduce a type system to express a concept of well–formedness of LCLS terms, we give an operational semantics of the new calculus, and, finally, we show the application of LCLS to the description of a biological system.

2 The Calculus of Looping Sequences

In this section we recall the Calculus of Looping Sequences (CLS). It is essentially based on term rewriting, hence a CLS model consists of a term and a set of rewrite rules. The term is intended to represent the structure of the modeled system, and the rewrite rules 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 *of CLS are given by the following grammar:*

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

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

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

The biological interpretation of the operators is the following: the main entities which occurs 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

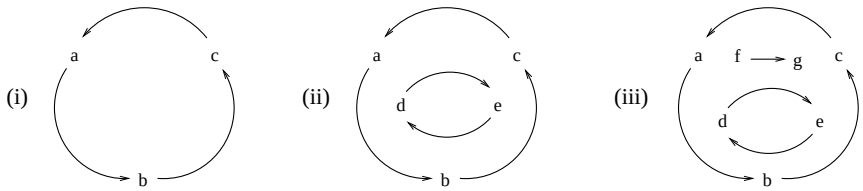

Fig. 1. (i) represents $\left(a \cdot b \cdot c\right)^L$; (ii) represents $\left(a \cdot b \cdot c\right)^L \rfloor \left(d \cdot e\right)^L$; (iii) represents $\left(a \cdot b \cdot c\right)^L \rfloor (\left(d \cdot e\right)^L \mid f \cdot g)$.

complex three–dimensional structure. In a protein there are usually (relatively) few subsequences, called domains, which actually are able to interact with other entities by means of chemical reactions. CLS sequences can model DNA/RNA strands and proteins by describing each gene or each domain with a symbol of the alphabet. Membranes are closed surfaces, often interspersed with proteins, which may contain something. A closed surface can be modeled by a looping sequence. The elements (or the subsequences) of the looping sequence may represent the proteins on the membrane, and by the containment operator it is possible to specify the content of the membrane. Other macro–molecules can be modeled as single alphabet symbols, or as short sequences. Finally, juxtaposition of entities can be described by the parallel composition of their representations.

Brackets can be used to indicate the order of application of the operators, and we assume $\left(\_\right)^L \rfloor \_$ to have precedence over $\_ \mid \_$. In Figure 1 we show some examples of CLS terms and their visual representation.

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

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

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

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

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

$$\left(\epsilon\right)^L \rfloor \epsilon \equiv_T \epsilon \qquad \left(S_1 \cdot S_2\right)^L \rfloor T \equiv_T \left(S_2 \cdot S_1\right)^L \rfloor T$$

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

Rewrite rules will be defined essentially as pairs of terms, in which the first term describes the portion of the system in which the event modeled by the rule may occur, and the second term describes how that portion of the system changes when the event occurs. In the terms of a rewrite rule we allow the use of variables. As a consequence, a rule will be applicable to all terms which can be

obtained by properly instantiating its variables. Variables can be of three kinds: two 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 which may include variables.

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

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

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. Now we 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 $P_2\sigma$. We 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 set of rewrite rules $\mathcal{R} \subseteq \Re$, the* semantics *of CLS is the least transition relation $\to$ on terms closed under $\equiv$, and satisfying the following inference rules:*

$$\frac{P_1 \mapsto P_2 \in \mathcal{R} \quad P_1\sigma \not\equiv \epsilon \quad \sigma \in \Sigma}{P_1\sigma \to P_2\sigma} \qquad \frac{T_1 \to T_2}{T \mid T_1 \to T \mid T_2} \qquad \frac{T_1 \to T_2}{\left(S\right)^{L} \rfloor T_1 \to \left(S\right)^{L} \rfloor T_2}$$

where the symmetric rule for the parallel composition is omitted.

A *model* in CLS is given by a term describing the initial state of the system and by a set of rewrite rules describing all the events that may occur.

3 The Calculus of Linked Looping Sequences

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, we extend CLS 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.

The syntax of terms of the Calculus of Linked Looping Sequences (LCLS) is defined as follows. We use as labels natural numbers.

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

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

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.

The structural congruence relation is the same as for CLS. Patterns of LCLS are similar to those of CLS, with the addition of the labels.

Definition 7 (Patterns). Patterns P *and* sequence patterns SP *of LCLS are given by the following grammar:*

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

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 content of a looping sequence and in which the contents of inner looping sequences are not considered. The top–level compartment is the portion of the pattern that is not inside

any looping sequence. For instance, the top–level compartment of a pattern $P = a \mid \left(b\right)^L \rfloor\, c \mid \left(d\right)^L \rfloor\, (X \mid \left(e\right)^L \rfloor\, f)$ 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 8 (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 variables occurring either four times ($N_1 \cap N_2 = \varnothing$) or three times ($N_1' \cap N_2 = N_1 \cap N_2' = \varnothing$). Variables occurring twice in $SP_1 \cdot SP_2$ are those which occur twice either in SP_1 or in SP_2 together with variables 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.

The following lemma states some simple properties of the type system.

Lemma 1. *Given $N, N' \subset \mathbb{N}$, and $P \in \mathcal{P}$, then $(N, N') \models P$ implies:*
(i) *both N and N' are finite;* (ii) *$N \cap N' = \varnothing$.*

Definition 9 (Well–Formedness of Patterns). *A pattern P is* well–formed *if and only if $\models P$ holds.*

Now we give two lemmas. The first relates the well–formedness of a pattern with the well–formedness of its subpatterns. The second states that well–formedness is preserved by structural congruence.

Lemma 2. *Given $P \in \mathcal{P}$ and P' a subpattern of P, then $\models P$ implies $\models P'$.*

Lemma 3. *Given $P_1, P_2 \in \mathcal{P}$, $\models P_1$ and $P_1 \equiv P_2$ imply $\models P_2$.*

The use of labels to represent links is not new. In [5] 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 [5], make the treatment more complicated. In [6], where the concept of membrane is introduced, well–formedness of terms is given intuitively and not formally defined.

We say that a well–formed pattern P is *closed* if and only if $(N, \varnothing) \models P$ for some $N \subset \mathbb{N}$, and that it is *open* otherwise. Moreover, we say that P is *link–free* if and only if $(\varnothing, \varnothing) \models P$. Since patterns include terms, we use the same terminology also for terms. For example, $a \cdot b \cdot c \mid d \cdot x$ is a link–free pattern, $a \cdot b^1 \cdot c \mid d \cdot x^1$ is a closed pattern, and $a \cdot b^1 \cdot c^2 \mid d \cdot x^1$ is an open pattern.

In the following we shall use a notion of set of links of a pattern, namely the set of labels that occur twice in the top–level compartment of the pattern.

Definition 10. *The* set of links *of a pattern P is $L(P) = \{n | \#(n, L_M(P)) = 2\}$, where $L_M(P)$ is the* multiset of labels *of P, recursively defined as follows:*

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

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

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

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

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 $\alpha(n)$. It holds that α–renaming preserves well–formedness.

Lemma 4. *Given $P \in \mathcal{P}$, $\forall \alpha \in \mathcal{A}$ it holds $\models P \iff \models P\alpha$.*

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.

Definition 11 (α–equivalence). *The α–equivalence relation $=_\alpha$ on LCLS patterns is the least equivalence relation which satisfies the following rules:*

$$\nu^{n1} \mid \mu^{n1} =_\alpha \nu^{n2} \mid \mu^{n2} \qquad \frac{P_1 \mid P_2 =_\alpha P_3}{P_2 \mid P_1 =_\alpha P_3} \qquad \frac{SP_1 \mid SP_2 =_\alpha P_3}{SP_1 \cdot SP_2 =_\alpha P_3}$$

$$\frac{P_1 =_\alpha P_2 \quad P_3 =_\alpha P_4 \quad L(P_1) \cap L(P_3) = L(P_2) \cap L(P_4) = \varnothing}{P_1 \mid P_3 =_\alpha P_2 \mid P_4}$$

$$\frac{SP_1 =_\alpha SP_2 \quad P_1 =_\alpha P_2 \quad L(SP_1) \cap L(SP_2) = L(P_1) \cap L(P_2) = \varnothing}{\left(SP_1\right)^L \rfloor P_1 =_\alpha \left(SP_2\right)^L \rfloor P_2}$$

$$\frac{\left(SP_1 \cdot SP_1'\right)^L \rfloor P_1 =_\alpha \left(SP_2 \cdot SP_2'\right)^L \rfloor P_2 \quad ni \notin L_M(SP_i \cdot SP_i') \cup L_M(P_i)}{\left(SP_1 \cdot \nu^{n1} \cdot SP_1'\right)^L \rfloor (\mu^{n1} \mid P_1) =_\alpha \left(SP_2 \cdot \nu^{n2} \cdot SP_2'\right)^L \rfloor (\mu^{n2} \mid P_2)}$$

$$\frac{\left(SP_1 \cdot SP_1'\right)^L \rfloor P_1 =_\alpha \left(SP_2 \cdot SP_2'\right)^L \rfloor P_2 \quad ni \notin L_M(SP_i \cdot SP_i') \cup L_M(P_i)}{\nu^{n1} \mid \left(SP_1 \cdot \mu^{n1} \cdot SP_1'\right)^L \rfloor P_1 =_\alpha \nu^{n2} \mid \left(SP_2 \cdot \mu^{n2} \cdot SP_2'\right)^L \rfloor P_2}$$

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.*

It is easy to see that α–equivalence preserves well–formedness of patterns.

Lemma 5. *Given* $P_1, P_2 \in \mathcal{P}$, $\models P_1$ *and* $P_1 =_\alpha P_2$ *imply* $\models P_2$.

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.

We say that an instantiation function σ is well–formed if it maps variables into well–formed closed terms and sequences. We denote with Σ_{wf} the set of all well–formed instantiation functions. Differently from CLS, the application of an instantiation function to a pattern does not correspond to the substitution of every variable in the pattern with the corresponding term given by the instantiation function, because this could lead to not well–formed terms. As an example, consider the well-formed pattern $P = a \cdot \widetilde{x} \mid X$ and a well–formed instantiation function σ such that $\sigma(\widetilde{x}) = b^1 \cdot c^1$ and $\sigma(X) = d^1 \mid e^1$. The application of σ to P would produce the term $P\sigma = a \cdot b^1 \cdot c^1 \mid d^1 \mid e^1$, which is not well–formed. Similarly, consider the well–formed pattern $P = a \cdot \widetilde{x} \cdot \widetilde{x}$ and the same well-formed instantiation function. We obtain $P\sigma = a \cdot b^1 \cdot c^1 \cdot b^1 \cdot c^1$, which is not well–formed. To avoid these situations, we define the application of an instantiation function to an LCLS pattern in a way such that the links in the instantiations of all occurrences of all variables are renamed, if necessary.

Definition 12 (Pattern Instantiation). *Given a pattern* $P \in \mathcal{P}$ *and an instantiation function* $\sigma \in \Sigma$, *the application of* σ *to* P *is an LCLS term* $P\sigma$ *given by the following inductive definition:*

$$\epsilon\sigma = \epsilon \quad a\sigma = a \quad a^n\sigma = a^n \quad \widetilde{x}\sigma = \sigma(\widetilde{x}) \quad x\sigma = \sigma(x) \quad x^n\sigma = \sigma(x)^n \quad X\sigma = \sigma(X)$$

$$\frac{SP_i\sigma =_\alpha S_i \quad L(S_1) \cap L(S_2) = \varnothing}{SP_1 \cdot SP_2\ \sigma = S_1 \cdot S_2} \qquad \frac{P_i\sigma =_\alpha T_i \quad L(T_1) \cap L(T_2) = \varnothing}{P_1 \mid P_2\ \sigma = T_1 \mid T_2}$$

$$\frac{SP\sigma =_\alpha S \quad P\sigma =_\alpha T \quad L(S) \cap L(T) = \varnothing}{\left(SP\right)^L \rfloor P\ \sigma = \left(S\right)^L \rfloor T}$$

where P_1, P_2, P *are any pattern,* SP_1, SP_2, SP *are any sequence pattern.*

Now, by applying a well–formed instantiation function to a well–formed pattern, we obtain a well–formed term.

Lemma 6. *Given $P \in \mathcal{P}, \sigma \in \Sigma_{wf}$, it holds that $\models P$ implies $\models P\sigma$.*

As in CLS, rewrite rules in LCLS are pairs of patterns.

Definition 13 (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.*

Our aim is to show that the application of a rewrite rule composed by well–formed patterns to a well–formed term produces another well–formed term. It is easy to see that, as a consequence of Lemma 6, this holds if variables of the rewrite rule are instantiated by a well–formed instantiation function. However, sometimes we would like to relax this constraint and allow a variable to be instantiated with an open term. For instance, we would permit the application of a rewrite rule $\widetilde{x} \cdot a \mapsto \widetilde{x} \cdot b$ to the term $c^1 \mid d^1 \cdot a$ (so to obtain $c^1 \mid d^1 \cdot b$), which requires that $\sigma(\widetilde{x}) = d^1$. Relaxing this constraint causes the introduction of constraints on the two patterns of the rewrite rules: they must not add or remove occurrences of variables, they cannot move variables from a compartment to another one, and they cannot add single occurrences of labels. To check these constraints we introduce a notion of compartment safety.

Definition 14 (Compartment Safety). *The* compartment safety relation *cs on pairs of patterns is the least equivalence relation satisfying the following rules:*

$$cs(\epsilon, \epsilon) \qquad cs(\epsilon, \nu) \qquad cs(\nu^n, \mu^n) \qquad cs(\epsilon, \nu^n | \mu^n) \qquad cs(\widetilde{x}, \widetilde{x}) \qquad cs(X, X)$$

$$\frac{cs(P_1, P_2) \quad cs(P_3, P_4)}{cs(P_1|P_3, P_2|P_4)} \qquad \frac{cs(P_1|P_2, P_3)}{cs(P_2|P_1, P_3)} \qquad \frac{cs(SP_1|SP_2, P_3)}{cs(SP_1 \cdot SP_2, P_3)}$$

$$\frac{cs(SP_1, SP_2) \quad cs(P_1, P_2)}{cs(\left(SP_1\right)^L \rfloor P_1, \left(SP_2\right)^L \rfloor P_2)} \qquad \frac{cs(\left(SP_1 \cdot SP_2\right)^L \rfloor P_1, \left(SP_3\right)^L \rfloor P_2)}{cs(\left(SP_2 \cdot SP_1\right)^L \rfloor P_1, \left(SP_3\right)^L \rfloor P_2)}$$

$$\frac{cs(\left(SP_1\right)^L \rfloor P_1, \left(SP_2\right)^L \rfloor P_2)}{cs(\left(SP_1\right)^L \rfloor P_1, \left(SP_2 \cdot \nu^n\right)^L \rfloor (\mu^n \mid P_2))} \qquad \frac{cs(\left(SP_1\right)^L \rfloor P_1, \left(SP_2\right)^L \rfloor P_2)}{cs(\left(SP_1\right)^L \rfloor P_1, \nu^n \mid \left(SP_2 \cdot \mu^n\right)^L \rfloor P_2)}$$

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

Definition 15 (Compartment Safe Rewrite Rule). *A rewrite rule $P_1 \mapsto P_2$ is* compartment safe (CS) *if $cs(P_1, P_2)$ holds. It is* compartment unsafe (CU) *otherwise. We denote with $\Re^{CS} \subset \Re$ the infinite set of CS rewrite rules, and with $\Re^{CU} \subset \Re$ the infinite set of CU rewrite rules.*

Now, we can introduce well–formedness also for rewrite rules.

Definition 16 (Well–Formedness of Rewrite Rules). *Given a rewrite rule $P_1 \mapsto P_2 \in \Re$, it is* well–formed *if P_1 and P_2 are well–formed patterns, and either $P_1 \mapsto P_2 \in \Re^{CS}$ or both P_1 and P_2 are closed patterns.*

The application of a well–formed rule satisfying compartment safety to a well–formed term preserves the well–formedness of the term even if variables are instantiated by a non well–formed instantiation function.

Lemma 7. *Given $\sigma \in \Sigma$ and a well–formed rewrite rule $P_1 \mapsto P_2$ such that $P_1 \mapsto P_2 \in \Re^{CS}$, it holds that $\models P_1\sigma$ implies $\models P_2\sigma$.*

Now, we can define the semantics of LCLS.

Definition 17 (Semantics). *Given a set of rewrite rules $\mathcal{R} \subseteq \Re$, such that $\mathcal{R} = \mathcal{R}^{CS} \cup \mathcal{R}^{CU}$ with $\mathcal{R}^{CS} \subset \Re^{CS}$ and $\mathcal{R}^{CU} \subset \Re^{CU}$, the* semantics *of LCLS is the least transition relation $\rightarrow$ on terms closed under $\equiv$ and $=_\alpha$, and satisfying the following inference rules:*

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

$$\textit{(appCU)} \quad \frac{P_1 \mapsto P_2 \in \mathcal{R}^{CU} \quad P_1\sigma \not\equiv \epsilon \quad \sigma \in \Sigma_{wf} \quad \alpha \in \mathcal{A}}{P_1\alpha\sigma \rightarrow P_2\alpha\sigma}$$

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

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

where the symmetric rule for the parallel composition is omitted.

Rules *(appCS)* and *(appCU)* describe the application of compartment safe and compartment unsafe rewrite rules, respectively. In the latter case we require that the instantiation function used to apply the rule is well–formed. In both cases, an α–renaming function is used to rename the labels in the pattern, in particular those appearing only once in the top–level compartment. The *(par)* and *(cont)* rules propagate the effect of a rewrite rule application to contexts by resolving conflicts in the use of labels.

Finally, we can give a theorem which states that the application of well–formed rewrite rules to well–formed terms produces new well–formed terms.

Theorem 1 (Subject Reduction). *Given a set of well–formed rewrite rules $\mathcal{R}$ and $T \in \mathcal{T}$, it holds that $\models T$ and $T \rightarrow T'$ imply $\models T'$.*

4 An Example: The EGF Signalling Pathway

A cell recognizes the EGF signal from the environment 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 one 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 causes the internal domains of the dimer to be recognized by a protein that is in the cytoplasm, called SHC. The protein SHC binds to the dimer, enabling a chain of protein–protein interactions inside the cell.

We model in LCLS the steps of the EGF pathway 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:

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

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

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

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

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

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):

$$\begin{aligned}
& EGF \mid EGF \mid \left(EGFR \cdot EGFR \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC) \\
\xrightarrow{(R1)} \;\; & EGF \mid \left(SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC) \\
\xrightarrow{(R1)} \;\; & \left(SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR \cdot SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR\right)^L \rfloor (SHC \mid SHC) \\
\xrightarrow{(R2)} \;\; & \left(SR_{E1} \cdot R^1_{E2} \cdot R_{I1} \cdot R_{I2} \cdot SR_{E1} \cdot R^1_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC)
\end{aligned}$$

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

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

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

5 Conclusions

In previous papers we introduced the formalism Calculus of Looping Sequences (CLS) suitable to describe biological systems and their evolution.

In the present paper we have presented LCLS, an extension of CLS suitable to describe protein interaction at the domain level. A type system allows expressing well–formedness of terms and rewrite rules of the calculus, and an operational semantics is given which preserves well–formedness. We have shown an example of application of the calculus to the description of a classical biological system, namely the protein interactions of the EGF signalling pathway.

The relationship between CLS/LCLS and similar formalisms are studied in detail in [7]. Further work includes developing concepts of bisimulations for the new calculus in the line of what done for CLS.

References

1. R. Alur, C. Belta, F. Ivancic, V. Kumar, M. Mintz, G.J. Pappas, H. Rubin, and J. Schug. "Hybrid Modeling and Simulation of Biomolecular Networks". Hybrid Systems: Computation and Control, LNCS 2034, pages 19–32, Springer, 2001.
2. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. "A Calculus of Looping Sequences for Modelling Microbiological Systems". Fundamenta Informaticae, volume 72, number 1–3, pages 21–35, 2006.
3. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. "Bisimulation Congruences in the Calculus of Looping Sequences". Proc. of ICTAC'06, LNCS 4281, pages 93–107, 2006.
4. L. Cardelli. "Brane Calculi. Interactions of Biological Membranes". Proc. of CMSB'04, LNCS 3082, pages 257–280, Springer, 2005.
5. V. Danos and C. Laneve. "Formal Molecular Biology". Theoretical Computer Science, volume 325, number 1, pages 69–110, 2004.
6. C. Laneve and F. Tarissan. "A Simple Calculus for Proteins and Cells". Proc. of MeCBIC'06, ENTCS, to appear.
7. P. Milazzo. "Qualitative and Quantitative Formal Modeling of Biological Systems". PhD Thesis, University of Pisa, 2007.
8. H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano."Hybrid Petri Net Representation of Gene Regulatory Network". Proc. of PSB'00,World Scientific Press, pages 341–352, 2000.
9. G. Păun. "Membrane Computing. An Introduction". Springer, 2002.
10. A. Regev, E.M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. "BioAmbients: An Abstraction for Biological Compartments". Theoretical Computer Science, volume 325, number 1, pages 141–167, 2004.
11. A. Regev, W. Silverman, and E.Y. Shapiro. "Representation and Simulation of Biochemical Processes Using the Pi-Calculus Process Algebra". Proc. of PSB'01, World Scientific Press, pages 459–470, 2001.