IOS Press

A Formalism for the Description of Protein Interaction[*] Dedicated to Jerzy Tiuryn on the Occasion of His Sixty Birthday

Roberto Barbuti,
Dipartimento di Informatica, Università di Pisa,
Largo Bruno Pontecorvo 3, 56127 Pisa, Italy
barbuti@di.unipi.it

Mariangiola Dezani-Ciancaglini,
Dipartimento di Informatica, Università di Torino,
Corso Svizzera 185, 10149 Torino, Italy
dezani@di.unito.it

Andrea Maggiolo-Schettini,
Dipartimento di Informatica, Università di Pisa,
Largo Bruno Pontecorvo 3, 56127 Pisa, Italy
maggiolo@di.unipi.it

Paolo Milazzo,
Dipartimento di Informatica, Università di Pisa,
Largo Bruno Pontecorvo 3, 56127 Pisa, Italy
milazzo@di.unipi.it

Angelo Troina,
Dipartimento di Informatica, Università di Torino,
Corso Svizzera 185, 10149 Torino, Italy
troina@di.unito.it

Abstract. The Calculus of Looping Sequences is a formalism for describing evolution of biological systems by means of term rewriting rules. We propose to enrich this calculus by labelling elements of sequences. Since two elements with the same label are considered to be linked, this allows us to represent protein interaction at the domain level. Well-formedness of terms are ensured by both a syntactic constraint and a type system: we discuss the differences between these approaches through the description of a biological system, namely the EGF pathway.

Keywords: Systems Biology, Predictive Modelling, Term Rewriting, Type Theories.

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

1. Introduction

The most common approach of biologists to describe biological systems is based on the use of deterministic mathematical means (like, e.g., ODE), and makes it possible to abstractly reason on the behaviour of biological systems and to perform a quantitative *in silico* investigation. This kind of modelling, however, becomes more and more difficult, both in the specification phase and in the analysis processes, when the complexity of the biological systems taken into consideration increases. This has probably been one of the main motivations for the application of Computer Science formalisms to the description of biological systems [19]. Other motivations can also be found in the fact that the use of formal methods from Computer Science permits the application of analysis techniques that are practically unknown to biologists, such as, for example, static analysis and model checking.

Among the formalisms that have either been applied to biology or have been inspired by biological systems there are automata-based models [2, 16], rewrite systems [11, 17], and process calculi [19, 20, 18, 10]. On the one hand, automata inspired models have the advantage of allowing the direct use of many verification tools such as model checkers. On the other hand, rewrite systems usually allow describing biological systems with a notation that can be easily understood by biologists. While automata-based models and rewrite systems present, in general, problems of compositionality (which might allow studying the behaviour of a system componentwise), these are resolved in general by using process calculi, included those used to describe biological systems.

In [6] 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 [7] 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 [11], and extended in [15]. 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 Linked Calculus of Looping Sequences (LCLS), is obtained by labelling elements of sequences. Two elements with the same label are considered to be linked.

The possibility of modelling protein interaction at the domain level allows some combinatorial problems in the description of cellular pathways to be avoided. In particular, cellular pathways often involve many different proteins, which can bind with each other in several different ways, thus forming a huge number of different protein complexes. The usual approach to the modelling of these pathways is by considering each possible protein complex as a different species, hence as a different entity in the model. The result is that models often become too complex to be analysed. The modelling of protein interaction at the domain level allows the set of species considered in a model to consist only of the proteins involved in the described pathway by representing complexes as structures having proteins as building blocks. In the case of LCLS, we can combine the advantages of this way of modelling protein interactions with the advantages of having a rather general formalism.

The use of pairs of labels to represent links in LCLS requires a notion of well-formedness of terms to be defined. Moreover, it is important to ensure that well-formedness is preserved by the application of rewrite rules. This is not true in general, but we propose two approaches to obtain this result: the first approach is based on a syntactic constraint for rewrite rules and the second on a type system. The two approaches are different in terms of expressiveness and efficiency. As an example of application, we show the LCLS description of a biological system, namely the EGF pathway.

This paper derives from the merging of the two papers [4, 3]. Both papers study well-formedness of links between protein sites within the Linked Calculus of Looping Sequences. The present version contains a detailed description of the two approaches, a theoretical comparison, and an explicit application in which the two approaches are put at work.

1.1. Systems Biology and Type Systems

In the last few years there has been a growing interest in the use of type disciplines to enforce biological properties. In [3] a type system has been defined to ensure the well-formedness of links between protein sites within the Linked Calculus of Looping Sequences (see [4]). In [14] three type systems are defined for the Biochemical Abstract Machine, BIOCHAM (see [1]). The first one is used to infer the functions of proteins in a reaction model, the second one to infer activation and inhibition effects of proteins, and the last one to infer the topology of compartments. In [13] we have defined a type system for the Calculus of Looping Sequences (see [6]) to guarantee the soundness of reduction rules with respect to the requirement of certain elements, and the repellency of others. Bioglio, in [8], refines this idea by designing a type system which assures that the cardinalities of elements of some types are in given numerical intervals. We have proposed in [12] a type system for the Stochastic Calculus of Looping sequences (see [5]) that allows for a quantitative analysis and models how the presence of catalysers (or inibitors) can modify the speed of reactions. In [9], we enrich the BioAmbients calculus with a static type system classifying each ambient with group types specifying the kind of compartments in which the ambient can stay. The type system ensures that, in a well-typed process, ambients cannot be nested in a way that violates the type hierarchy.

1.2. Summary

The remainder of this paper is organised as follows. In Section 2 we present the Calculus of Looping Sequences and introduce our running example about the EGF signalling pathway. In Section 3 we provide the formalisation of the Linked Calculus of Looping Sequences and introduce the definition of well-formed terms with links. We also redefine the EGF signalling pathway example by taking links into account. The well-formedness condition on the link structure should be checked at run-time on the term resulting after the application of a rewrite rule. In Section 4 we investigate two statically verifiable conditions that allow to avoid or simplify the run-time checking. Hence, we introduce the condition of *compartment safety*, checking that the structure of our rewrite rules do not alter the well-formed linkage of a term to be rewritten, and the one of *typed safety*, checking that the right hand side of the rule has a type which is *similar* to the one of the left hand side. For this last condition, we develop a type inference technique which simplifies checking applicability of rewrite rules. In Section 5 we discuss, through an

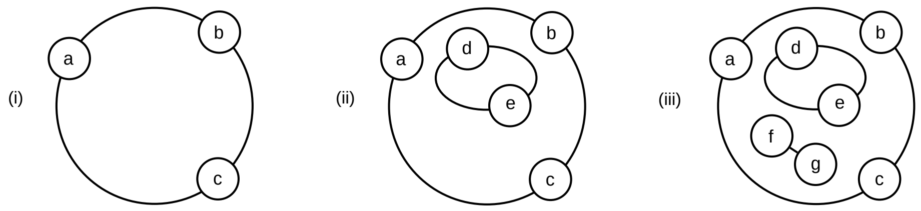

Figure 1. Examples of CLS terms: (i) represents $\left(a \cdot b \cdot c\right)^L$; (ii) represents $\left(a \cdot b \cdot c\right)^L \rfloor \left(d \cdot e\right)^L$; (iii) represents $\left(a \cdot b \cdot c\right)^L \rfloor (\left(d \cdot e\right)^L \mid f \cdot g)$.

example, our techniques to guarantee the well-formedness of terms. Finally, in Section 6, we draw our conclusions.

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 modelled system, and the rewrite rules to represent the events that may cause the system to evolve.

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

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

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

where a is a generic element of $\mathcal{E}$, and ϵ represents the empty sequence. We denote with $\mathbb{T}$ 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 $\left(\_\right)^L$, 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 $\left(\_\right)^L \rfloor \_$ that applies to one sequence and one term.

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

Brackets can be used to indicate the order of application of the operators, and we assume $\left(\_\right)^L \rfloor \_$ 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 introduce a structural congruence relation to identify such terms.

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

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

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 modelled by the rule may occur, and the second term describes how that portion of the system changes when the event occurs. In the terms of a rewrite rule we allow the use of variables. As a consequence, a rule will be applicable to all terms which can be obtained by properly instantiating their 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 2.3. (Patterns)
Patterns P and *sequence patterns* SP of CLS are given by the following grammar:

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

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 $\sigma : \mathcal{V} \to \mathbb{T}$. An instantiation must preserve the type of variables, thus for $X \in TV, \widetilde{x} \in SV$ and $x \in \mathcal{X}$ we have $\sigma(X) \in \mathbb{T}, \sigma(\widetilde{x}) \in \mathcal{S}$ 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 2.4. (Rewrite Rules)
A rewrite rule is a pair of patterns (P_1, P_2), denoted with $P_1 \mapsto P_2$, where $P_1, P_2 \in \mathcal{P}$, $P_1 \not\equiv \epsilon$ and such that $Var(P_2) \subseteq Var(P_1)$. 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 2.5. (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:

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

$$\frac{T_1 \to T_2}{T \mid T_1 \to T \mid T_2} \qquad \frac{T_1 \to T_2}{\left(S\right)^L \rfloor T_1 \to \left(S\right)^L \rfloor T_2}$$

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.

In order to show the usage of CLS as a model of biological systems, we give the CLS model of a well-known example of cellular signal transduction, namely the EGF signalling pathway.

In Biology, signal transduction refers to any process by which a cell converts one kind of signal or stimulus into another. Signals are typically proteins that may be present in the environment of the cell. In order to be able to receive the signal, namely to recognize that the corresponding ligand is available in the environment, a cell exposes some receptors on its external membrane. A receptor is a transmembrane protein that can bind to a signal protein on its extracellular end. When such a binding is established, the intracellular end of the receptor undergoes a conformational change that enables interaction with other proteins inside the cell. This typically causes an ordered sequence of biochemical reactions inside the cell, usually called signalling pathway, that are carried out by enzymes and may produce different effects on the cell behaviour.

A complex signal transduction cascade, that modulates cell proliferation, survival, adhesion, migration and differentiation, is based on a family of receptors called epidermal growth factor receptors (EGFRs). While EGFR signalling is essential for many normal morphogenic processes, the aberrant activity of these receptors has been shown to play a fundamental role in proliferation of tumor cells. Epidermal growth factor receptors are sinthesized from specific genes in the DNA through transcription into RNA (mediated by polymerase enzymes) and translation into protein (mediated by ribosomes), and they are located on the cell surface. Receptors are activated by the binding with a specific ligand (epidermal growth factor, EGF) to form a EGFR (ligand-receptor) complex. Upon activation, EGFR undergoes a

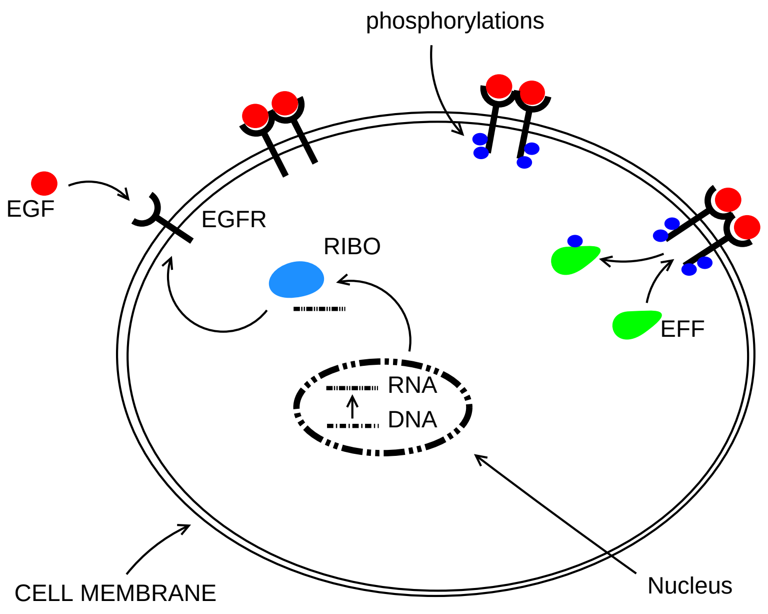

Figure 2. The EGF signaling pathway.

transition from a monomeric form to an active dimeric one. EGFR dimerization stimulates its intracellular phosphorylation which activates signalling proteins. These activated signalling proteins (effector proteins) initiate several signal transduction cascades, leading to DNA synthesis and cell proliferation.

Now we give the CLS model of the first steps of the signalling pathway. We model the cell membrane as a looping sequence $(m)^L$, where the alphabet symbol m generically denotes the whole membrane surface when no receptors are present yet on the membrane. Similarly, we model the membrane of the nucleus as the looping sequence $(n)^L$. Moreover, we model DNA and RNA as the elements DNA and RNA, and the signal and receptor proteins as the elements EGF and $EGFR$, respectively. Polymerases and ribosomes are modelled as elements $POLY$ and $RIBO$, respectively. A signal-receptor complex is denoted as $CMPLX$ and a dimer composed by two of such complexes is denoted either as DIM, before phosphorylation, or as $DIMp$, after phosphorylation. We denote an effector protein with EFF, that becomes $DIMEFF$ when bound to a dimer and $EFFp$ after phosphorylation.

First of all, synthesis of the EGFR receptor from the DNA is described by the following rules:

$$POLY \mid DNA \;\mapsto\; POLY \mid DNA \mid RNA \tag{R1}$$

$$(n)^L \rfloor (RNA \mid X) \;\mapsto\; RNA \mid (n)^L \rfloor X \tag{R2}$$

$$RNA \mid RIBO \;\mapsto\; RNA \mid RIBO \mid EGFR \tag{R3}$$

$$(m \cdot \widetilde{x})^L \rfloor (EGFR \mid X) \;\mapsto\; (m \cdot \widetilde{x} \cdot EGFR)^L \rfloor X \tag{R4}$$

Rule (R1) describes the transcription of DNA into RNA performed by the polymerase enzyme. Rule (R2) describes the coming out of the RNA from the nucleus. Rule (R3) describes the translation of RNA into the EGFR protein performed by the ribosome. Rule (R4) describes the incorporation of the EGFR in the cell membrane.

The first steps of the EGF signalling pathway are described by the following rules:

$$
\begin{aligned}
EGF \mid (EGFR \cdot \widetilde{x})^L \rfloor X &\mapsto (CMPLX \cdot \widetilde{x})^L \rfloor X && \text{(R5)}\\
(CMPLX \cdot \widetilde{x} \cdot CMPLX \cdot \widetilde{y})^L \rfloor X &\mapsto (DIM \cdot \widetilde{x} \cdot \widetilde{y})^L \rfloor X && \text{(R6)}\\
(DIM \cdot \widetilde{x})^L \rfloor X &\mapsto (DIMp \cdot \widetilde{x})^L \rfloor X && \text{(R7)}\\
(DIMp \cdot \widetilde{x})^L \rfloor (X \mid EFF) &\mapsto (DIMEFF \cdot \widetilde{x})^L \rfloor X && \text{(R8)}\\
(DIMEFF \cdot \widetilde{x})^L \rfloor X &\mapsto (DIMp \cdot \widetilde{x})^L \rfloor (X \mid EFFp) && \text{(R9)}
\end{aligned}
$$

Rule (R5) describes the binding of the EGF signal protein with a receptor EGFR on the cell membrane. The result of the binding is a signal-receptor complex whose dimerization is described by rule (R6). Rule (R7) describes the phosphorylation (activation) of a dimer, which enables the propagation of the signal inside the cell by means of phosphorylation of effector proteins (rules (R8) and (R9)).

3. The Linked Calculus of 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. For example, the CLS term $a \cdot b \cdot c \mid d \cdot e \cdot f$ could model two proteins each having three domains. However, CLS does not provide any suitable method to model the binding of one of the domains of the first protein to one of the domains of the second protein. 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. For example, we will denote with $a \cdot b^1 \cdot c \mid d \cdot e^1 \cdot f$ two proteins in which domain b of the first protein is bound to domain e of the second protein. 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 may 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.

Now we formally define the Linked Calculus of Looping Sequences (LCLS), namely the extension of CLS with labels on basic symbols. In the definition we often abuse of notations already introduced for CLS. However, this will not cause ambiguities because in the following we shall always use these notations with reference to LCLS.

The syntax of LCLS terms is defined as follows. We use as labels natural numbers.

Definition 3.1. (Terms)
Terms T and *Sequences* S of LCLS are given by the following grammar:

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

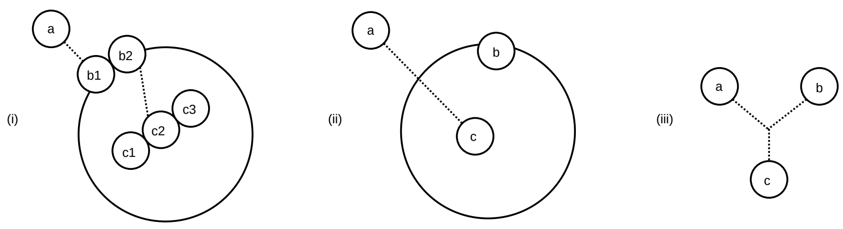

Figure 3. Examples of well-formed and non well-formed terms: (i) represents $a^1 \mid \big(b1^1 \cdot b2^2\big)^L \rfloor\, c1 \cdot c2^2 \cdot c3$; (ii) represents $a^1 \mid \big(b\big)^L \rfloor\, c^1$; (iii) represents $a^1 \mid b^1 \mid c^1$.

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

We will denote by $\mathcal{O}(T)$, $\mathcal{T}(T)$ the set of labels which occur once or twice in T, respectively, and by $\mathcal{L}(T)$ the set $\mathcal{O}(T) \cup \mathcal{T}(T)$.

In what follows, we will use the notion of *top-level compartment* of a term. The top-level compartment is the portion of the term that is not inside any looping sequence. For instance, the top-level compartment of the following term

$$T = a \mid \big(b\big)^L \rfloor\, c \mid \big(d\big)^L \rfloor\, (e \cdot f \mid \big(g\big)^L \rfloor\, h)$$

is

$$a \mid \big(b\big)^L \rfloor\, \epsilon \mid \big(d\big)^L \rfloor\, \epsilon.$$

Formally, we define a function $tlc(T)$ that gives the top-level compartment of term T as follows:

$$tlc(S) = S \qquad tlc\big(\big(S\big)^L \rfloor\, T\big) = \big(S\big)^L \rfloor\, \epsilon \qquad tlc(T_1 \mid T_2) = tlc(T_1) \mid tlc(T_2).$$

As explained before, it is intended that in an LCLS term each label should appear either once or twice. Moreover, for a term to be *well formed* it is also important that labels in $\mathcal{T}(T)$ either appear in the same compartment, or one in a looping sequence and the other in the compartment immediately inside such a looping sequence. Moreover, labels in $\mathcal{O}(T)$ must appear in the top-level compartment of T in order to allow them to correctly pair with another label provided by a possible context of T. In Figure 3 we show some examples of well-formed and non well-formed terms: term (i) is well formed since labels appear twice and in proper positions; term (ii) is not well formed since the two occurrences of the label are associated with elements that are in completely different compartments; term (iii) is not well formed since there is a label occurring three times.

We now define a unary relation wf on terms such that $wf(T)$ holds if and only if term T is well formed.

Definition 3.2. (Well-formedness)
The unary relation wf on LCLS terms is the least relation satisfying the following rules:

$$wf(\epsilon) \qquad wf(a) \qquad wf(a^n)$$

$$\frac{wf(S_1) \quad wf(S_2) \quad \mathcal{L}(S_1) \cap \mathcal{T}(S_2) = \mathcal{T}(S_1) \cap \mathcal{L}(S_2) = \varnothing}{wf(S_1 \cdot S_2)}$$

$$\frac{wf(T_1) \quad wf(T_2) \quad \mathcal{L}(T_1) \cap \mathcal{T}(T_2) = \mathcal{T}(T_1) \cap \mathcal{L}(T_2) = \varnothing}{wf(T_1 \mid T_2)}$$

$$\frac{wf(S) \quad wf(T) \quad \mathcal{L}(S) \cap \mathcal{T}(T) = \mathcal{T}(S) \cap \mathcal{L}(T) = \varnothing \quad \mathcal{O}(T) \subseteq \mathcal{O}(S)}{wf(\left(S\right)^L \rfloor T)}$$

Rules of well-formedness simply check that labels do not occur more than twice in a term, and that symbols occurring only once in a term contained in a looping sequence occur once also in the looping sequence itself.

We can show that, as expected, labels occurring only once in a term are placed in the top-level compartment of the term.

Proposition 3.1. Given $T \in \mathcal{T}$, if $wf(T)$ holds, then $\mathcal{O}(T) = \mathcal{O}(tlc(T))$.

Proof:
Easy by structural induction on T. $\square$

The structural congruence for LCLS extends the corresponding relation for CLS with a notion of α-renaming that allows labels in $\mathcal{T}(T)$ to be replaced in T by other unused labels (i.e. labels not belonging to $\mathcal{L}(T)$). The α-renaming is assumed not to change labels in $\mathcal{O}(T)$ since this could break a binding between an element in T and another in the context of T, leading to a non well-formed term.

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

Definition 3.3. (Patterns)
Patterns P and *sequence patterns* SP of LCLS are given by the following grammar:

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

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.

Structural congruence, the notions of top-level compartment, and the well-formedness relation wf trivially extend to patterns.

Rewrite rules for LCLS are defined exactly as for CLS, namely as pairs of patterns $P_1 \mapsto P_2$ with the conditions $P_1 \neq \epsilon$ and $Var(P_2) \subseteq Var(P_1)$. Obviously, P_1 and P_2 are now LCLS patterns.

The definition of the semantics of LCLS is similar to that of CLS, but with the requirement that states are well-formed terms. This means that the initial term of an LCLS model has to be well formed and that application of rules leading to non well-formed terms are forbidden.

Definition 3.4. (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 (according to the definition of wf) that is closed under $\equiv$, and satisfies the following inference rules:

$$\frac{P_1 \mapsto P_2 \in \mathcal{R} \quad P_1\sigma \not\equiv \epsilon \quad \sigma \in \Sigma}{P_1\sigma \rightarrow P_2\sigma}$$

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

Now, let us reformulate the CLS model of signal transduction given in Section 2 as an LCLS model. 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} describe two extra-cellular domains, whereas R_{I1} and R_{I2} describe two intra-cellular domains. In particular, R_{E1} models the EGF signal binding site, R_{E2} the dimerization site, R_{I1} the phosphorilation site, and R_{I2} the effector binding site. The use of links will allow us to avoid introducing different elements to model complexes, such as the $CMPLX$, DIM and $DIMEFF$ elements we used in the CLS model in Section 2.

The rewrite rules modeling the pathway are the following. Each rule in this model has a corresponding (unprimed) rule in the CLS model in Section 2. In the description of the rules we will focus on the main differences with respect to the corresponding ones in the CLS model.

First of all, synthesis of the EGFR receptor from the DNA is described by the following rules:

$$POLY \mid DNA \;\mapsto\; POLY \mid DNA \mid RNA \tag{R1'}$$

$$\big(n\big)^L \rfloor (RNA \mid X) \;\mapsto\; RNA \mid \big(n\big)^L \rfloor X \tag{R2'}$$

$$RNA \mid RIBO \;\mapsto\; RNA \mid RIBO \mid R_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \tag{R3'}$$

$$\big(m \cdot \widetilde{x}\big)^L \rfloor (R_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \mid X) \;\mapsto\; \big(m \cdot \widetilde{x} \cdot R_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2}\big)^L \rfloor X \tag{R4'}$$

Rule (R3') represents the ribosome translation of the RNA into the EGFR sequence $R_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2}$. Rule (R4') defines the embedding of the sequence within the external cell membrane.

The first steps of the EGF signalling pathway are described by the following rules:

$$EGF \mid \big(R_{E1} \cdot \widetilde{x}\big)^L \rfloor X \;\mapsto\; EGF^1 \mid \big(R^1_{E1} \cdot \widetilde{x}\big)^L \rfloor X \tag{R5'}$$

$$\big(R^1_{E1} \cdot R_{E2} \cdot \widetilde{x} \cdot R^2_{E1} \cdot R_{E2} \cdot \widetilde{y}\big)^L \rfloor X \;\mapsto\; \big(R^1_{E1} \cdot R^3_{E2} \cdot \widetilde{x} \cdot R^2_{E1} \cdot R^3_{E2} \cdot \widetilde{y}\big)^L \rfloor X \tag{R6'}$$

$$\big(R^1_{E2} \cdot R_{I1} \cdot \widetilde{x}\big)^L \rfloor X \;\mapsto\; \big(R^1_{E2} \cdot Rp_{I1} \cdot \widetilde{x}\big)^L \rfloor X \tag{R7'}$$

$$\big(R^1_{E2} \cdot Rp_{I1} \cdot R_{I2} \cdot \widetilde{x} \cdot R^1_{E2} \cdot Rp_{I1} \cdot \widetilde{y}\big)^L \rfloor (X \mid EFF) \;\mapsto\; \big(R^1_{E2} \cdot Rp_{I1} \cdot R^2_{I2} \cdot \widetilde{x} \cdot R^1_{E2} \cdot Rp_{I1} \cdot \widetilde{y}\big)^L \rfloor (X \mid EFF^2) \tag{R8'}$$

$$\big(R^1_{I2} \cdot \widetilde{x}\big)^L \rfloor (X \mid EFF^1) \;\mapsto\; \big(R_{I2} \cdot \widetilde{x}\big)^L \rfloor (X \mid EFFp) \tag{R9'}$$

Rule (R5') describes the binding of the EGF signal protein with the R_{E1} domain by creating a link with label 1. When two of these bindings are constructed on the membrane, the signal-receptor complexes might dimerise originating a new link between the two R_{E2} domains of the two receptors: see the link labelled with 3 in rule (R6'). Then, the R_{I1} domain of a receptor of a dimer formed on the membrane can be activated (phosphorylated) moving to the form Rp_{I1}. This process is represented in rule (R7'). The activated dimer enables the propagation of the signal inside the cell by promoting the binding of the effector protein EFF inside the cell with one of its R_{I2} domains (link labelled with 2 in rule (R8')). The effector protein bound to the dimer gets phosphorylated and released within the cell: in Rule (R9') the link between EFF and the R_{I2} domain is removed.

4. Syntactic Constraints and Types for LCLS

The semantics of LCLS ensures that by starting from a well-formed term it is not possible to reach non well-formed terms. In practice, this means that before applying a rewrite rule one has to check that the whole term obtained as the result of the application is well formed. Checking the well-formedness of a term at run-time, before applying every rule, could be a very costly operation. Hence, in this section we investigate two statically verifiable conditions that allow run-time checking to be either avoided or reduced.

The first condition we investigate is whether the two patterns of each rule are similar enough to avoid non well-formed terms to be obtained as result of their application to a well-formed term. If this condition, that we shall call *compartment safety*, is satisfied by the rules of an LCLS model, then it is possible to avoid any run-time well-formedness checking.

The second condition we investigate is whether the two patterns of each rule have similar types, according to a type system we shall define. If this condition, that we shall call *typed safety*, is satisfied by the rules of an LCLS model, then the run-time checking can be limited to a control on the instantiation of variables. Note that the constraints of typed safety are weaker than those of compartment safety, hence the corresponding semantics is more general. Moreover, we shall develop a type inference technique for typed safety based on the machinery of *principal typing*.

4.1. Compartment safety in LCLS

Compartment safety is defined as a condition on pairs of (well-formed) patterns, intended to be LCLS rewrite rules. Roughly speaking, the condition forbids rewrite rules (i) to introduce single occurrences of labels, (ii) to create copies or delete sequence and term variables, and (iii) to move sequence and term variables from one compartment to another. In fact, all of these three actions may lead to non well-formed terms. For instance, the introduction of a single occurrence of a label may lead to a non well-formed term if in the same compartment two occurrences of such a label are already present. Moreover, duplication or movement of a sequence (or term) variable from one compartment to another may cause either too many occurrences of the same labels to be created in the same compartment or single occurrences to remain in some inner compartment.

The compartment safety relation is defined as follows.

Definition 4.1. (Compartment Safety)
The *compartment safety relation* cs is the least congruence on well-formed patterns satisfying the following rules:

$$cs(\epsilon, \nu) \qquad cs(\nu^n, \mu^n) \qquad cs(\epsilon, \nu^n \mid \mu^n) \qquad \text{(cs1,cs2,cs3)}$$

$$cs(P_1 \mid P_2, P_2 \mid P_1) \qquad cs(P, \epsilon \mid P) \qquad \text{(cs4,cs5)}$$

$$cs(SP_1 \mid SP_2, SP_1 \cdot SP_2) \qquad cs(SP, \big(SP\big)^L \rfloor \epsilon) \qquad \text{(cs6,cs7)}$$

$$cs(\big(SP_1 \cdot SP_2\big)^L \rfloor P, \big(SP_2\big)^L \rfloor (SP_1 \mid P)) \quad \text{with } SP_1 \in \mathcal{SP}^* \qquad \text{(cs8)}$$

$$cs(\big(SP_1 \cdot SP_2\big)^L \rfloor P, SP_1 \mid \big(SP_2\big)^L \rfloor P) \quad \text{with } SP_1 \in \mathcal{SP}^* \text{ or } tlc(P) \in \mathcal{P}^* \qquad \text{(cs9)}$$

where $\nu, \mu \in \mathcal{E} \cup \mathcal{X}$, $n \in \mathbb{N}$, P_1, P_2, P_3, P_4 are any pattern, SP_1, SP_2, SP_3 are any sequence pattern. Moreover, $\mathcal{P}^*$ and $\mathcal{SP}^*$ denote the set of all patterns and sequence patterns, respectively, in which only element varialble are allowed (sequence and term variables are not allowed).

We remark that compartment safety is defined on well-formed patterns and that it is a congruence, hence the condition can be verified on two patterns in a compositional and transitive way.

The definition of the relation cs is such that one can prove $cs(P_1, P_2)$ to hold by proving that $cs(P_1, P_1')$, $cs(P_1', P_2')$ and $cs(P_2', P_2)$ hold, where P_1' and P_2' are obtained by moving all elements and element variables in P_1 and P_2, respectively, in a parallel composition in the top-level compartment. For example, from $P_1 = \big(a \cdot b^1 \cdot \widetilde{x}\big)^L \rfloor (c^1 \mid X)$ we obtain $P_1' = a \mid b^1 \mid c^1 \mid \big(\widetilde{x}\big)^L \rfloor X$, and from $P_2 = \big(a \cdot b^1 \cdot \widetilde{x}\big)^L \rfloor (c^1 \cdot c^2 \mid b^2 \mid X)$ we obtain $P_2' = a \mid b^1 \mid c^1 \mid c^2 \mid b^2 \mid \big(\widetilde{x}\big)^L \rfloor X$.

In order to prove $cs(P_1, P_2)$, rules (cs6-9) should be used to prove $cs(P_1, P_1')$ and $cs(P_2, P_2')$, whereas rules (cs1-5) to prove $cs(P_1', P_2')$. The essence of the relation is actually represented by rules (cs1-3) which state that two compartment safe patterns may differ only in the presence of elements and element variables without labels, in the element or element variable carrying a specific label or in the presence of additional pairs of elements or element variables with the same label. The fact that P_1 and P_2 are required to be well formed ensures that two labels of each pair possibly introduced by a compartment safe rewrite rule will be placed in the same compartment.

Definition 4.2. (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.

Compartment safety is a rather strong syntactical requirement on rewrite rules. For example, it forbids rules such as $a \mapsto a^1$, $\widetilde{x} \cdot \widetilde{y} \mapsto \widetilde{x}$, and $a \cdot \widetilde{x} \mid \big(b\big)^L \rfloor X \mapsto \big(b\big)^L (a \cdot \widetilde{x} \mid X)$ to be used. However, compartment safe rewrite rules are expressive enough to describe most of the biochemical interactions typical of cellular processes. For example, $\widetilde{x} \cdot a \cdot \widetilde{y} \mid \widetilde{w} \cdot b \cdot \widetilde{z} \mapsto \widetilde{x} \cdot a^1 \cdot \widetilde{y} \mid \widetilde{w} \cdot b^1 \cdot \widetilde{z}$ could describe the creation of protein/protein or protein/dna bindings, $\widetilde{x} \cdot a \cdot \widetilde{y} \mid \big(\widetilde{w} \cdot b \cdot \widetilde{z}\big)^L \rfloor X \mapsto \widetilde{x} \cdot a^1 \cdot \widetilde{y} \mid \big(\widetilde{w} \cdot b^1 \cdot \widetilde{z}\big)^L \rfloor X$ the creation of signal-receptor complexes on cell membranes, and $\big(\widetilde{x}\big)^L \rfloor (X \mid a \cdot x \cdot y) \mapsto a \cdot x \cdot y \mid \big(\widetilde{x}\big)^L \rfloor X$ the release by a cell of a signal protein (represented by the sequence $a \cdot x \cdot y$ not containing any sequence variable). In all these cases sequence and term variables actually play the role of the context in which the interaction occurs, hence they are not moved from one compartment to another. What cannot be described by compartment safe rules are more complex events which involve, for example, changes in compartments structure. Moreover, it does not allow elements (such as signal proteins) that are usually moved from one compartment to another to be described in an abstract way by means of sequence variables.

The application of a rule satisfying compartment safety to a well-formed term preserves the well-formedness of the term.

Lemma 4.1. Given $\sigma \in \Sigma$ and a rewrite rule $P_1 \mapsto P_2 \in \Re^{CS}$, it holds that $wf(P_1\sigma)$ implies $wf(P_2\sigma)$.

Proof:
By induction on the derivation of $cs(P_1, P_2)$. The only non-trivial (base) cases are those of rules (cs8) and (cs9).

- Let (cs8) be the last applied rule, namely $P_1 = \big(SP_1 \cdot SP_2\big)^L \rfloor P$ and $P_2 = \big(SP_2\big)^L \rfloor (SP_1 \mid P)$. By definition of wf and by distributivity of σ we have that $wf\big(((SP_1 \cdot SP_2)^L \rfloor P)\sigma\big)$ implies $wf(SP_1\sigma)$, $wf(SP_2\sigma)$ and $wf\big(P\sigma\big)$. By definition of wf we also have that $\mathcal{L}((SP_1 \cdot SP_2)\sigma) \cap \mathcal{T}(P\sigma) = \mathcal{T}((SP_1 \cdot SP_2)\sigma) \cap \mathcal{L}(P\sigma) = \varnothing$ and $\mathcal{O}(P\sigma) \subseteq \mathcal{O}((SP_1 \cdot SP_2)\sigma)$. It follows immediately that $wf((SP_1 \mid P)\sigma))$ holds.

  In order to prove that $wf\big(((SP_2)^L \rfloor (SP_1 \mid P))\sigma\big)$ holds it remains to prove that $\mathcal{L}((SP_2)\sigma) \cap \mathcal{T}((SP_1 \mid P)\sigma) = \mathcal{T}((SP_2)\sigma) \cap \mathcal{L}((SP_1 \mid P)\sigma) = \varnothing$ and $\mathcal{O}((SP_1 \mid P)\sigma) \subseteq \mathcal{O}(SP_2\sigma)$. Since cs is defined on well-formed patterns, we have that $\mathcal{L}(SP_2) \cap \mathcal{T}(SP_1 \mid P) = \mathcal{T}(SP_2) \cap \mathcal{L}(SP_1 \mid P) = \varnothing$ and $\mathcal{O}(SP_1 \mid P) \subseteq \mathcal{O}(SP_2)$ hold. In other words, the only possible cause of violation of well-formedness is instantiation of sequence and term variables. However, condition $SP_1 \in \mathcal{SP}^*$ in (cs8) implies that $\mathcal{O}(SP_1\sigma) = \mathcal{O}(SP_1)$, $\mathcal{T}(SP_1\sigma) = \mathcal{T}(SP_1)$ and $\mathcal{L}(SP_1\sigma) = \mathcal{L}(SP_1)$. Hence, from the facts that $\mathcal{L}((SP_1 \cdot SP_2)\sigma) \cap \mathcal{T}(P\sigma) = \varnothing$ implies $\mathcal{L}(SP_2\sigma) \cap \mathcal{T}(P\sigma) = \varnothing$, that $\mathcal{L}(SP_2) \subseteq \mathcal{L}(SP_2\sigma)$, and that $\mathcal{L}(SP_2) \cap \mathcal{T}(SP_1) = \varnothing$, we can conclude that $\mathcal{L}(SP_2\sigma) \cap \mathcal{T}((SP_1 \mid P)\sigma) = \mathcal{L}(SP_2\sigma) \cap (\mathcal{T}(SP_1) \cup \mathcal{T}(P\sigma)) = \varnothing$. Similarly we can also prove that $\mathcal{T}((SP_2)\sigma) \cap \mathcal{L}((SP_1 \mid P)\sigma) = \varnothing$. As regards $\mathcal{O}((SP_1 \mid P)\sigma) \subseteq \mathcal{O}(SP_2\sigma)$, namely $(\mathcal{O}(SP_1\sigma) \cup \mathcal{O}(P\sigma)) \subseteq \mathcal{O}(SP_2\sigma)$, from condition $SP_1 \in \mathcal{SP}^*$ in (cs8) we have that $\mathcal{O}(SP_1\sigma) = \mathcal{O}(SP_1)$. Moreover, from $\mathcal{O}(SP_1 \mid P) \subseteq \mathcal{O}(SP_2)$ we have that $\mathcal{O}(SP_1) \subseteq \mathcal{O}(SP_2)$, which implies $\mathcal{O}(SP_1\sigma) \subseteq \mathcal{O}(SP_2\sigma)$. Hence, it remains to prove that $\mathcal{O}(P\sigma) \subseteq \mathcal{O}(SP_2\sigma)$, but this follows immediately from $\mathcal{O}(P\sigma) \subseteq \mathcal{O}((SP_1 \cdot SP_2)\sigma)$.

- The case in which (cs9) is the last applied rule can be proved similarly to the case of (cs8) if condition $SP_1 \in \mathcal{SP}^*$ of (cs9) is satisfied. The case in which condition $tlc(P) \in \mathcal{P}^*$ is satisfied can be proved by following a similar approach and by exploiting Proposition 3.1.

$\square$

Now, we can define the semantics of LCLS.

Definition 4.3. (Semantics)
Given a finite set of rewrite rules $\mathcal{R} \subset \Re^{CS}$, the *semantics* of LCLS is the least relation on terms closed under $\equiv$, and satisfying the following inference rules:

$$\text{(appCS)} \quad \frac{P_1 \mapsto P_2 \in \mathcal{R} \quad P_1\sigma \not\equiv \epsilon \quad \sigma \in \Sigma}{P_1\sigma \xrightarrow{CS} P_2\sigma}$$

$$\text{(par)} \quad \frac{T_1 \xrightarrow{CS} T_1' \qquad \mathcal{T}(T_1') \cap \mathcal{L}(T_2) = \{n_1, \ldots, n_M\} \qquad n_1', \ldots, n_M' \text{ fresh}}{T_1 \mid T_2 \xrightarrow{CS} T_1'\{^{n_1', \ldots, n_M'}/_{n_1, \ldots, n_M}\} \mid T_2}$$

$$\text{(cont)} \quad \frac{T \xrightarrow{CS} T' \qquad \mathcal{T}(T') \cap \mathcal{L}(S) = \{n_1, \ldots, n_M\} \qquad n_1', \ldots, n_M' \text{ fresh}}{\big(S\big)^L \rfloor T \xrightarrow{CS} \big(S\big)^L \rfloor T'\{^{n_1', \ldots, n_M'}/_{n_1, \ldots, n_M}\}}$$

Rule (appCS) describes the application of compartment safe rewrite rules. 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 prove a lemma stating that transitions performed by the semantics preserve the set of labels occurring only once in the term. This lemma will be used to prove the main theorem stating that the application of well-formed rewrite rules to well-formed terms produces new well-formed terms.

Lemma 4.2. Given a finite set of rewrite rules $\mathcal{R} \subset \Re^{CS}$ and $T, T' \in \mathcal{T}$, it holds that $wf(T)$ and $T \xrightarrow{CS} T'$ imply $\mathcal{O}(T) = \mathcal{O}(T')$.

Proof:
By induction on $\xrightarrow{CS}$.

- If the last applied rule is (appCS), then $P_1 \mapsto P_2 \in \mathcal{R}$, $T \equiv P_1\sigma$ and $T' \equiv P_2\sigma$. It can be easily proved by induction on the derivation of $cs(P_1, P_2)$ that $\mathcal{O}(P_1) = \mathcal{O}(P_2)$. Hence, a difference between $\mathcal{O}(T)$ and $\mathcal{O}(T')$ can only be caused by a difference in the occurrences of sequence and term variables in P_1 and in P_2. By Proposition 3.1 we have that actually only variables in the top-level compartments of P_1 and P_2 have to be considered. Again, it can be easily proved by induction on the derivation of $cs(P_1, P_2)$ that every occurrence of a sequence or a term variable in $tlc(P_1)$ is also in $tlc(P_2)$ and viceversa.

- If the last applied rule is either (par) or (cont), the proof is a trivial application of the induction hypothesis.

$\square$

Theorem 4.1. Given a finite set of rewrite rules $\mathcal{R} \subset \Re^{CS}$ and $T, T' \in \mathcal{T}$, it holds that $wf(T)$ and $T \xrightarrow{CS} T'$ imply $wf(T')$.

Proof:
By induction on $\xrightarrow{CS}$.

- If the last applied rule is (appCS), then $P_1 \mapsto P_2 \in \mathcal{R}$. The fact that $wf(T')$ holds follows from $wf(T)$ and Lemma 4.1.

- If the last applied rule is (par), then $T \equiv T_1 \mid T_2$ and $T_1 \xrightarrow{CS} T_1'$. By definition of wf we have that $wf(T_1 \mid T_2)$ implies $wf(T_1)$ and $wf(T_2)$. Moreover, $wf(T_1')$ holds by induction hypothesis. By definition of wf, from $wf(T_1 \mid T_2)$ we get $\mathcal{L}(T_1) \cap \mathcal{T}(T_2) = \mathcal{T}(T_1) \cap \mathcal{L}(T_2) = \varnothing$, and in order to prove $wf(T_1' \mid T_2')$ we have to prove that $\mathcal{L}(T_1') \cap \mathcal{T}(T_2) = \mathcal{T}(T_1') \cap \mathcal{L}(T_2) = \varnothing$. By Lemma 4.2 we have $\mathcal{O}(T_1) = \mathcal{O}(T_1')$. This means that what we only have to prove is $\mathcal{T}(T_1') \cap \mathcal{L}(T_2) = \varnothing$, but this is ensured by the substitution of conflicting labels with fresh ones done in rule (par) of the semantics.

- For rule (cont) the proof is similar to that of rule (par).

$\square$

From Theorem 4.1 it follows immediately that the semantics obtained by considering only compartment safe rules is contained in the general semantics of LCLS, namely $T_1 \xrightarrow{CS} T_2$ implies $T_1 \rightarrow T_2$. The viceversa does not hold as $a \mapsto a^1 \in \Re^{CU}$ could be used to derive $a \mid b \rightarrow a^1 \mid b$, whereas this rule cannot be used in the semantics based on compartment safety.

4.2. Typed safety in LCLS

In this section we propose a generalisation of compartment safety in two respects:

- we classify elements with basic types and we assure that only elements of the same basic types are linked;
- we relax the restrictions on the shapes of the rewrite rules and of the instantiations.

To this aim we require atomic elements to be of some basic type. We assume a possibly infinite set $\mathcal{B}$ of basic types and we use $\mathtt{t}$ to range over basic types. Intuitively, given a molecule represented by an element in $\mathcal{E}$, we associate with it a type in $\mathcal{B}$ which specifies the kind of the molecule and the kind of bindings the molecule can create. We assume a fixed typing Γ_0 for the elements in $\mathcal{E}$, i.e. Γ_0 is a mapping from $\mathcal{E}$ to $\mathcal{B}$.

We use $\eta \in SV \cup TV$ to denote either sequence or term variables. With $\mathtt{T}$ we denote a finite set of natural numbers and with $\mathtt{O}$ we denote a finite set of typed natural numbers, i.e. numbers associated with basic types.

Intuitively, given a pattern P, we can associate with P a set of numbers $\mathtt{T}$ which contains all the numbers used to create closed point to point bindings. A pattern may however also contain some numbers which do not bind another molecule in P but may bind somewhere else in the environment. Thus, we may associate with a pattern P a set $\mathtt{O}$ of typed numbers. We do not need to keep track of types for closed links, but, for open links, we should guarantee that a molecule of some type is bound, somewhere else in the environment, with a molecule of the same type. To sum up we associate with a pattern a *pair type* of the shape $(\mathtt{T}, \mathtt{O})$. We associate pair types also with sequence patterns.

We define the domain of a set of typed numbers $\mathtt{O}$ as

$$dom(\mathtt{O}) = \{n \mid n : \mathtt{t} \in \mathtt{O}\}.$$

We say that two sets of typed numbers $\mathtt{O}$ and $\mathtt{O}'$ are *compatible* (written $\mathtt{O} \bowtie \mathtt{O}'$) if and only if whenever $n : \mathtt{t} \in \mathtt{O}$ and $n : \mathtt{t}' \in \mathtt{O}'$, then it holds $\mathtt{t} = \mathtt{t}'$. The *linked union* of two compatible sets $\mathtt{O}$ and $\mathtt{O}'$ (notation $\mathtt{O} \uplus \mathtt{O}'$) is defined as

$$\mathtt{O} \uplus \mathtt{O}' = \{n : \mathtt{t} \in \mathtt{O} \wedge n \notin dom(\mathtt{O}')\} \cup \{n : \mathtt{t}' \in \mathtt{O}' \wedge n \notin dom(\mathtt{O})\}.$$

With the following grammar we define *bases* Γ, which map element variables to basic types, and map sequence and term variables to pair types:

$$\Gamma \ ::= \ \emptyset \quad \Big| \quad \Gamma, x : \mathtt{t} \quad \Big| \quad \Gamma, \eta : (\mathtt{T}, \mathtt{O}).$$

The type discipline to check safe bindings, namely, to avoid non well-formed bindings, is defined by the typing rules in Figure 4.

Rules (ϵ)-(a)-(x): any basis types with $\mathtt{T}$ and $\mathtt{O}$ empty sets, the term ϵ and any elementary object without binding labels. Rules (a^n)-(x^n): an elementary object with a binding label n gets typed with $\mathtt{T}$ empty (there are no labels defining a closed link) and with $\mathtt{O} = \{n : \mathtt{t}\}$ (there is an open link represented by label n of type $\mathtt{t}$). Rule (η): complex sequences and terms may contain both closed links and open links.

$$\Gamma \vdash \epsilon : (\emptyset, \emptyset)\ (\epsilon) \qquad \frac{a : \mathtt{t} \in \Gamma_0}{\Gamma \vdash a : (\emptyset, \emptyset)}\ (a) \qquad \frac{a : \mathtt{t} \in \Gamma_0}{\Gamma \vdash a^n : (\emptyset, \{n : \mathtt{t}\})}\ (a^n)$$

$$\Gamma, x : \mathtt{t} \vdash x : (\emptyset, \emptyset)\ (x) \qquad \Gamma, x : \mathtt{t} \vdash x^n : (\emptyset, \{n : \mathtt{t}\})\ (x^n) \qquad \Gamma, \eta : (\mathtt{T}, \mathtt{O}) \vdash \eta : (\mathtt{T}, \mathtt{O})\ (\eta)$$

$$\frac{\Gamma \vdash SP : (\mathtt{T}, \mathtt{O}) \quad \Gamma \vdash SP' : (\mathtt{T}', \mathtt{O}') \quad \mathtt{T} \cap \mathtt{T}' = \mathtt{T} \cap dom(\mathtt{O}') = \mathtt{T}' \cap dom(\mathtt{O}) = \emptyset \quad \mathtt{O} \bowtie \mathtt{O}'}{\Gamma \vdash SP \cdot SP' : (\mathtt{T} \cup \mathtt{T}' \cup (dom(\mathtt{O}) \cap dom(\mathtt{O}')), \mathtt{O} \uplus \mathtt{O}')}\ (seq)$$

$$\frac{\Gamma \vdash P : (\mathtt{T}, \mathtt{O}) \quad \Gamma \vdash P' : (\mathtt{T}', \mathtt{O}') \quad \mathtt{T} \cap \mathtt{T}' = \mathtt{T} \cap dom(\mathtt{O}') = \mathtt{T}' \cap dom(\mathtt{O}) = \emptyset \quad \mathtt{O} \bowtie \mathtt{O}'}{\Gamma \vdash P \mid P' : (\mathtt{T} \cup \mathtt{T}' \cup (dom(\mathtt{O}) \cap dom(\mathtt{O}')), \mathtt{O} \uplus \mathtt{O}')}\ (par)$$

$$\frac{\Gamma \vdash SP : (\mathtt{T}, \mathtt{O}) \quad \Gamma \vdash P : (\mathtt{T}', \mathtt{O}') \quad \mathtt{T} \cap \mathtt{T}' = \mathtt{T} \cap dom(\mathtt{O}') = \mathtt{T}' \cap dom(\mathtt{O}) = \emptyset \quad \mathtt{O}' \subseteq \mathtt{O}}{\Gamma \vdash \left(SP\right)^L \rfloor P : (\mathtt{T} \cup dom(\mathtt{O}'), \mathtt{O} \setminus \mathtt{O}')}\ (loop)$$

Figure 4. Typing rules for safe bindings

Rule (seq): when putting two sequences together, the numbers representing the closed links should not appear in any other binding. Open links in the two sequences to be join can form a closed link if they have the same label (since we require compatibility between $\mathtt{O}$ and $\mathtt{O}'$, the labels closing each open link are of the same type). In the resulting sequences, the labels which got closed are removed form $\mathtt{O}$ and $\mathtt{O}'$ and added to the final set of labels representing closed links. Rule (par): similarly to what happens for Rule (seq), putting two patterns in parallel may allow to close some of the links which are open in the two patterns in isolation.

Rule $(loop)$: we can put a pattern P inside a looping sequence SP only when all the open links of P are closed. This is because if P gets inside a compartment (represented by the looping sequence), it cannot interact any more with the environment. Thus, if P has some open link, it should be bound with equal open links present on the looping sequence SP, which now represents the only environment surrounding P. For this to be done, we require that the set of open links of P is a subset of the set of open links of SP (all the open links in P can be closed by SP).

The following lemma clarifies the meaning of pair types and can be easily shown by induction on LCLS terms.

Lemma 4.3. The following implications hold:

1. $\vdash T : (\mathtt{T}, \mathtt{O})$ if and only if $wf(T)$.
2. If $\vdash T : (\mathtt{T}, \mathtt{O})$, then $\mathtt{T} = \mathcal{T}(T)$ and $dom(\mathtt{O}) = \mathcal{O}(T)$.
3. If $\vdash T : (\mathtt{T} \cup \{n\}, \mathtt{O})$ and n' is fresh, then $\vdash T\{^{n'}/_{n}\} : (\mathtt{T} \cup \{n'\}, \mathtt{O})$.

Rewrite rules may modify the status of the bindings by creating new closed links or destroying some of them. We require, however, that rewrite rules do not change the status of open bindings, which are assumed to be closed by the environment and represent only a partial state of the system.

Definition 4.4. (Γ-Safe Rules)
A rewrite rule $P \mapsto P'$ is Γ-*safe* (notation $P \mapsto P' \in \mathcal{R}_\Gamma$) if $\Gamma \vdash P : (\mathtt{T}, \mathtt{O})$ and $\Gamma \vdash P' : (\mathtt{T}', \mathtt{O})$ for some $\mathtt{T}, \mathtt{T}', \mathtt{O}$.

An instantiation σ *agrees* with a basis Γ (notation $\sigma \in \Sigma_\Gamma$) if $x : \mathtt{t} \in \Gamma$ implies $\sigma(x) : \mathtt{t} \in \Gamma$ and $\eta : (\mathtt{T}, \mathtt{O}) \in \Gamma$ implies $\Gamma \vdash \sigma(\eta) : (\mathtt{T}, \mathtt{O})$.

Lemma 4.4. If $\sigma \in \Sigma_\Gamma$, then $\vdash P\sigma : (\mathtt{T}, \mathtt{O})$ if and only if $\Gamma \vdash P : (\mathtt{T}, \mathtt{O})$.

Proof:

($\Leftarrow$) By induction on $\Gamma \vdash P : (\mathtt{T}, \mathtt{O})$. Consider the last applied rule.

- For rules (ϵ), (a), (a^n) we have $P\sigma = P$ and, moreover, P is typable from the empty environment. If the rule is (x), (x^n), or (η), the proof follows from $\sigma \in \Sigma_\Gamma$.
- Rule (seq). In this case $P = SP_1 \cdot SP_2$, $\mathtt{T} = \mathtt{T}_1 \cup \mathtt{T}_2 \cup (dom(\mathtt{O}_1) \cap dom(\mathtt{O}_2))$, $\mathtt{O} = \mathtt{O}_1 \uplus \mathtt{O}_2$, $\Gamma \vdash SP_1 : (\mathtt{T}_1, \mathtt{O}_1)$, $\Gamma \vdash SP_2 : (\mathtt{T}_2, \mathtt{O}_2)$, $\mathtt{T}_1 \cap \mathtt{T}_2 = \mathtt{T}_1 \cap dom(\mathtt{O}_2) = \mathtt{T}_2 \cap dom(\mathtt{O}_1) = \emptyset$ and $\mathtt{O}_1 \bowtie \mathtt{O}_2$. By induction hypotheses, $\vdash SP_1\sigma : (\mathtt{T}_1, \mathtt{O}_1)$ and $\vdash SP_2\sigma : (\mathtt{T}_2, \mathtt{O}_2)$. Therefore, since $SP_1\sigma \cdot SP_2\sigma = (SP_1 \cdot SP_2)\,\sigma$, applying rule (seq) we conclude $\vdash (SP_1 \cdot SP_2)\,\sigma : (\mathtt{T}, \mathtt{O})$.
- For rules (par), $(loop)$ the proof is similar.

($\Rightarrow$) By induction on P.

- If P is a variable the proof follows from $\sigma \in \Sigma_\Gamma$. If $P = \epsilon$, or $P = a$, or $P = a^n$, the lemma holds by weakening.
- Let P be $SP_1 \cdot SP_2$. Since $(SP_1 \cdot SP_2)\,\sigma = SP_1\sigma \cdot SP_2\sigma$, the fact that $\vdash (SP_1 \cdot SP_2)\,\sigma : (\mathtt{T}, \mathtt{O})$ implies that the last applied rule must be (seq). Therefore, $\mathtt{T} = \mathtt{T}_1 \cup \mathtt{T}_2 \cup (dom(\mathtt{O}_1) \cap dom(\mathtt{O}_2))$, $\mathtt{O} = \mathtt{O}_1 \uplus \mathtt{O}_2$, $\Gamma \vdash SP_1 : (\mathtt{T}_1, \mathtt{O}_1)$, $\Gamma \vdash SP_2 : (\mathtt{T}_2, \mathtt{O}_2)$, $\mathtt{T}_1 \cap \mathtt{T}_2 = \mathtt{T}_1 \cap dom(\mathtt{O}_2) = \mathtt{T}_2 \cap dom(\mathtt{O}_1) = \emptyset$ and $\mathtt{O}_1 \bowtie \mathtt{O}_2$. By induction hypothesis on SP_1 and SP_2 we get $\Gamma \vdash SP_1 : (\mathtt{T}_1, \mathtt{O}_1)$ and $\Gamma \vdash SP_2 : (\mathtt{T}_2, \mathtt{O}_2)$. Applying rule (seq) we conclude $\Gamma \vdash SP_1 \cdot SP_2 : (\mathtt{T}, \mathtt{O})$.
- If $P = P' \mid P''$ or $P = \big(SP\big)^L \rfloor P'$ the proof is similar.

$\square$

We can safely apply a Γ-safe rule to a term only if the involved instantiation agrees with Γ. In this case we denote by $\xrightarrow{TS}$ the so obtained reduction. More formally:

Definition 4.5. (Typed Semantics)
Given a finite set of rewrite rules $\mathcal{R}$, the *typed semantics* of LCLS is the least relation on terms closed

with respect to $\equiv$ and satisfying the following inference rules:

$$\text{(appTS)} \quad \frac{P_1 \mapsto P_2 \in \mathcal{R}_\Gamma \qquad P_1\sigma \not\equiv \epsilon \qquad \sigma \in \Sigma_\Gamma}{P_1\sigma \xrightarrow{TS} P_2\sigma}$$

$$\text{(par)} \quad \frac{T_1 \xrightarrow{TS} T_1' \qquad \mathcal{T}(T_1') \cap \mathcal{L}(T_2) = \{n_1, \dots, n_M\} \qquad n_1', \dots, n_M' \text{ fresh}}{T_1 \mid T_2 \xrightarrow{TS} T_1'\{^{n_1', \dots, n_M'}/n_1, \dots, n_M\} \mid T_2}$$

$$\text{(cont)} \quad \frac{T \xrightarrow{TS} T' \qquad \mathcal{T}(T') \cap \mathcal{L}(S) = \{n_1, \dots, n_M\} \qquad n_1', \dots, n_M' \text{ fresh}}{(S)^L \rfloor T \xrightarrow{TS} (S)^L \rfloor T'\{^{n_1', \dots, n_M'}/n_1, \dots, n_M\}}$$

As expected, $\xrightarrow{TS}$ reduction preserves typing of LCLS terms.

Theorem 4.2. If $\vdash T : (\mathtt{T}, \mathtt{O})$ and $T \xrightarrow{TS} T'$, then $\vdash T' : (\mathtt{T}', \mathtt{O})$ for some $\mathtt{T}'$.

Proof:
By induction on $\xrightarrow{TS}$.

- If the last applied rule is (appTS), then $P_1 \mapsto P_2 \in \mathcal{R}_\Gamma$, $\sigma \in \Sigma_\Gamma$ and $\vdash P_1\sigma : (\mathtt{T}, \mathtt{O})$. By Lemma 4.4 we get $\Gamma \vdash P_1 : (\mathtt{T}, \mathtt{O})$, which implies $\Gamma \vdash P_2 : (\mathtt{T}', \mathtt{O})$ for some $\mathtt{T}'$ by definition of Γ-safe rule. Again by Lemma 4.4 we conclude $\vdash P_2\sigma : (\mathtt{T}', \mathtt{O})$.

- If the last applied rule is (par), then $T \equiv T_1 \mid T_2$ and $\vdash T : (\mathtt{T}, \mathtt{O})$ is derived by using rule (par). Therefore $\vdash T_1 : (\mathtt{T}_1, \mathtt{O}_1)$, $\vdash T_2 : (\mathtt{T}_2, \mathtt{O}_2)$, $\mathtt{T}_1 \cap \mathtt{T}_2 = \mathtt{T}_1 \cap dom(\mathtt{O}_2) = \mathtt{T}_2 \cap dom(\mathtt{O}_1) = \emptyset$, $\mathtt{O}_1 \bowtie \mathtt{O}_2$, $\mathtt{T} = \mathtt{T}_1 \cup \mathtt{T}_2 \cup (dom(\mathtt{O}_1) \cap dom(\mathtt{O}_2))$, $\mathtt{O} = \mathtt{O}_1 \uplus \mathtt{O}_2$. By induction hypothesis $\vdash T_1' : (\mathtt{T}_1', \mathtt{O}_1)$ for some $\mathtt{T}_1'$. By Lemma 4.3(2) $\mathtt{T}_1' = \{n_1, \dots, n_M\} \cup \mathtt{T}_1''$ for some $\mathtt{T}_1''$, and then by (3) of the same lemma $\vdash T_1'\{^{n_1', \dots, n_M'}/n_1, \dots, n_M\} : (\{n_1', \dots, n_M'\} \cup \mathtt{T}_1'', \mathtt{O}_1)$. We can then apply rule (par) to $\vdash T_1'\{^{n_1', \dots, n_M'}/n_1, \dots, n_M\} : (\{n_1', \dots, n_M'\} \cup \mathtt{T}_1'', \mathtt{O}_1)$ and $\vdash T_2 : (\mathtt{T}_2, \mathtt{O}_2)$, since by construction $\{n_1', \dots, n_M'\} \cup \mathtt{T}_1''$, $\mathtt{O}_1$, $\mathtt{T}_2$, $\mathtt{O}_2$ satisfy the required conditions, and conclude $\vdash T' : (\mathtt{T}', \mathtt{O})$, where $\mathtt{T}' = \{n_1', \dots, n_M'\} \cup \mathtt{T}_1'' \cup \mathtt{T}_2 \cup (dom(\mathtt{O}_1) \cap dom(\mathtt{O}_2))$.

- For rule (cont) the proof is similar to that of rule (par).

□

In order to decide which rewriting rules are Γ-safe the inference of principal basis schemes, type schemes and typing conditions for patterns is handy.

We convene that for each variable $x \in \mathcal{X}$ there is an *e-type variable* φ_x ranging over basic types, and for each variable $\eta \in SV \cup TV$ there are two variables ϕ_η, ψ_η (called *t-type variable* and *o-type variable*) ranging over sets of untyped and typed numbers, respectively.

A *basis scheme* Θ is a map from atomic variables to their e-type variables, and from sequence and term variables to pairs of their t-type variables and o-type variables:

$$\Theta \ ::= \ \emptyset \quad \mid \quad \Theta, x : \varphi_x \quad \mid \quad \Theta, \eta : (\phi_\eta, \psi_\eta).$$

$$\vdash \epsilon : \emptyset; (\emptyset, \emptyset); \emptyset \; (I\epsilon) \qquad \frac{a : \mathtt{t} \in \Gamma_0}{\vdash a : \emptyset; (\emptyset, \emptyset); \emptyset} \; (Ia) \qquad \frac{a : \mathtt{t} \in \Gamma_0}{\vdash a^n : \emptyset; (\emptyset, \{n : \mathtt{t}\}); \emptyset} \; (Ia^n)$$

$$\vdash x : \{x : \varphi_x\}; (\emptyset, \emptyset); \emptyset \; (Ix) \qquad \vdash x^n : \{x : \varphi_x\}; (\emptyset, \{n : \varphi_x\}); \emptyset \; (Ix^n)$$

$$\vdash \eta : \{\eta : (\phi_\eta, \psi_\eta)\}; (\phi_\eta, \psi_\eta); \emptyset \; (I\eta)$$

$$\frac{\vdash SP : \Theta; (\Phi, \Psi); \Xi \qquad \vdash SP' : \Theta'; (\Phi', \Psi'); \Xi'}{\vdash SP \cdot SP' : \Theta \cup \Theta'; (\Phi \cup \Phi' \cup (dom(\Psi) \cap dom(\Psi')), \Psi \uplus \Psi'); \Xi''} \; (Iseq)$$
where $\Xi'' = \Xi \cup \Xi' \cup \{\Phi \cap \Phi' = \Phi \cap dom(\Psi') = \Phi' \cap dom(\Psi) = \emptyset\} \cup \{\Psi \bowtie \Psi'\}$

$$\frac{\vdash P : \Theta; (\Phi, \Psi); \Xi \qquad \vdash P' : \Theta'; (\Phi', \Psi'); \Xi'}{\vdash P \mid P' : \Theta \cup \Theta'; (\Phi \cup \Phi' \cup (dom(\Psi) \cap dom(\Psi')), \Psi \uplus \Psi'); \Xi''} \; (Ipar)$$
where $\Xi'' = \Xi \cup \Xi' \cup \{\Phi \cap \Phi' = \Phi \cap dom(\Psi') = \Phi' \cap dom(\Psi) = \emptyset\} \cup \{\Psi \bowtie \Psi'\}$

$$\frac{\vdash SP : \Theta; (\Phi, \Psi); \Xi \qquad \vdash P : \Theta'; (\Phi', \Psi'); \Xi'}{\vdash \left(SP\right)^L \rfloor P : \Theta \cup \Theta'; (\Phi \cup dom(\Psi'), \Psi \setminus \Psi'); \Xi''} \; (Iloop)$$
where $\Xi'' = \Xi \cup \Xi' \cup \{\Phi \cap \Phi' = \Phi \cap dom(\Psi') = \Phi' \cap dom(\Psi) = \emptyset\} \cup \{\Psi' \subseteq \Psi\}$

Figure 5. Inference rules

A *type scheme* is a pair (Φ, Ψ), where Φ ranges over unions of sets of untyped numbers and t-type variables, and Ψ ranges over unions of sets of typed numbers and o-type variables.

A *typing condition* is either a set theoretic or a compatibility condition involving unions of sets of untyped numbers and t-type variables, and unions of sets of typed numbers and o-type variables.

The inference rules use judgements of the shape:

$$\vdash P : \Theta; (\Phi, \Psi); \Xi$$

where Θ is the *principal basis scheme* in which P is well formed, (Φ, Ψ) is the *principal type scheme* of P, and Ξ is the *principal set of typing conditions* which should be satisfied when building up P.

Figure 5 gives these inference rules, derived from the typing rules in Figure 4.

Rules ($I\epsilon$), (Ia) and (Ia^n) directly derive from rules (ϵ), (a) and (a^n). The rules for typing variables (rules (Ix), (Ix^n) and ($I\eta$)) put the variable with its type in the basis. In rules ($Iseq$), ($Ipar$) and ($Iloop$), the principal type is derived as in (seq), (par) and ($loop$) rules, respectively. The set of constraints is the union between the constraints in the premise of the rule itself and the constraints in the premise of (seq), (par) and ($loop$) rules, respectively. The principal basis is the union of the principal bases of the composing patterns, without renaming, because each variable x, x^n or η is associated to a unique e-type variable or to a unique pair of t-type and o-type variables, respectively.
The key difference between inference rules, in Figure 5, and typing rules, in Figure 4, is that the conditions are not premises, but conclusions. In this way, at the end of inference all these conditions create a set of constraints, that must be checked to decide the applicability of the rules.

Since the inference rules follow the structure of patterns it is easy to verify that the complexity of inference is linear in the number of symbols occurring in patterns.

Soundness and completeness of our inference rules can be stated as usual. A *type mapping* maps e-type variables to basic types, t-type variables to sets of numbers and o-type variables to sets of typed numbers. A type mapping m *satisfies* a set of constraints Ξ if all constraints in $\mathsf{m}(\Xi)$ hold true.

Theorem 4.3. (Soundness of Type Inference)
If $\vdash P : \Theta; (\Phi, \Psi); \Xi$ and m is a type mapping which satisfies Ξ, then $\mathsf{m}(\Theta) \vdash P : (\mathsf{m}(\Phi), \mathsf{m}(\Psi))$.

Proof:
By induction on derivations, and by cases on the last applied rule.

- For rules $(I\epsilon)$, (Ia), (Ia^n), (Ix), (Ix^n), and $(I\eta)$ the result is trivial.
- Rule $(Iseq)$. In this case the conclusion of the rule is $\vdash SP_1 \cdot SP_2 : \Theta; (\Phi, \Psi); \Xi$ where $\Theta = \Theta_1 \cup \Theta_2$, $\Phi = \Phi_1 \cup \Phi_2 \cup (dom(\Psi_1) \cap dom(\Psi_2))$, $\Psi = \Psi_1 \uplus \Psi_2$, $\Xi = \Xi_1 \cup \Xi_2 \cup \{\Phi_1 \cap \Phi_2 = \Phi_1 \cap dom(\Psi_2) = \Phi_2 \cap dom(\Psi_1) = \emptyset\} \cup \{\Psi_1 \bowtie \Psi_2\}$ and the assumptions are $\vdash SP_1 : \Theta_1; (\Phi_1, \Psi_1); \Xi_1$ and $\vdash SP_2 : \Theta_2; (\Phi_2, \Psi_2); \Xi_2$. Since m satisfies Ξ_1 and Ξ_2, by induction hypothesis, and weakening, we derive that $\mathsf{m}(\Theta_1 \cup \Theta_2) \vdash SP_1 : (\mathsf{m}(\Phi_1), \mathsf{m}(\Psi_1))$ and $\mathsf{m}(\Theta_1 \cup \Theta_2) \vdash SP_2 : (\mathsf{m}(\Phi_2), \mathsf{m}(\Psi_2))$. Moreover, since m satisfies Ξ, we have that $\mathsf{m}(\Phi_1) \cap \mathsf{m}(\Phi_2) = \mathsf{m}(\Phi_1) \cap dom(\mathsf{m}(\Psi_2)) = \mathsf{m}(\Phi_2) \cap dom(\mathsf{m}(\Psi_1)) = \emptyset$ and $\mathsf{m}(\Psi_1) \bowtie \mathsf{m}(\Psi_2)$. So rule (seq) can be applied, and $\mathsf{m}(\Theta_1 \cup \Theta_2) \vdash SP_1 \cdot SP_2 : (\mathsf{m}(\Phi), \mathsf{m}(\Psi))$.
- For rules $(Ipar)$, and $(Iloop)$ the result can be proved like for rule $(Iseq)$.

□

Theorem 4.4. (Completeness of Type Inference)
If $\Gamma \vdash P : (\mathtt{T}, \mathtt{O})$, then $\vdash P : \Theta; (\Phi, \Psi); \Xi$ for some Θ, Φ, Ψ, Ξ and there is a type mapping m that satisfies Ξ and such that $\Gamma \supseteq \mathsf{m}(\Theta)$, $\mathtt{T} = \mathsf{m}(\Phi)$, $\mathtt{O} = \mathsf{m}(\Psi)$.

Proof:
By induction on the derivation of $\Gamma \vdash P : (\mathtt{T}, \mathtt{O})$.

- If the last rule of the derivation is (ϵ), (a), (a^n), (x), (x^n), or (η) the result is obvious.
- Rule (seq). In this case $P = SP_1 \cdot SP_2$, $\mathtt{T} = \mathtt{T}_1 \cup \mathtt{T}_2 \cup (dom(\mathtt{O}_1) \cap dom(\mathtt{O}_2))$, $\mathtt{O} = \mathtt{O}_1 \uplus \mathtt{O}_2$, $\Gamma \vdash SP_1 : (\mathtt{T}_1, \mathtt{O}_1)$, $\Gamma \vdash SP_2 : (\mathtt{T}_2, \mathtt{O}_2)$, $\mathtt{T}_1 \cap \mathtt{T}_2 = \mathtt{T}_1 \cap dom(\mathtt{O}_2) = \mathtt{T}_2 \cap dom(\mathtt{O}_1) = \emptyset$ and $\mathtt{O} \bowtie_1 \mathtt{O}_2$. By induction hypothesis, there are Θ_1, Φ_1, Ψ_1, Ξ_1, Θ_2, Φ_2, Ψ_2, Ξ_2 such that $\vdash SP_1 : \Theta_1; (\Phi_1, \Psi_1); \Xi_1$ and $\vdash SP_2 : \Theta_2; (\Phi_2, \Psi_2); \Xi_2$. These are the assumptions of rule $(Iseq)$, whose conclusion is $\vdash SP_1 \cdot SP_2 : \Theta_1 \cup \Theta_2; (\Phi_1 \cup \Phi_2 \cup (dom(\Psi_1) \cap dom(\Psi_2)), \Psi_1 \uplus \Psi_2); \Xi$, where $\Xi = \Xi_1 \cup \Xi_2 \cup \{\Phi_1 \cap \Phi_2 = \Phi_1 \cap dom(\Psi_2) = \Phi_2 \cap dom(\Psi_1) = \emptyset\} \cup \{\Psi_1 \bowtie \Psi_2\}$. Moreover, by induction there is a type mapping m_1 satisfying Ξ_1 such that $\Gamma \supseteq \mathsf{m}_1(\Theta_1)$, $\mathtt{T}_1 = \mathsf{m}_1(\Phi_1)$ and $\mathtt{O}_1 = \mathsf{m}_1(\Psi_1)$, and there is a type mapping m_2 satisfying Ξ_2 such that $\Gamma \supseteq \mathsf{m}_2(\Theta_2)$, $\mathtt{T}_2 = \mathsf{m}_2(\Phi_2)$ and $\mathtt{O}_2 = \mathsf{m}_2(\Psi_2)$. Therefore, we derive $\Gamma \supseteq \mathsf{m}_1(\Theta_1) \cup \mathsf{m}_2(\Theta_2)$, $\mathtt{T} = \mathsf{m}_1(\Phi_1) \cup \mathsf{m}_2(\Phi_2) \cup (dom(\mathsf{m}_1(\Psi_1)) \cap dom(\mathsf{m}_2(\Psi_2)))$ and $\mathtt{O} = \mathsf{m}_1(\Psi_1) \uplus \mathsf{m}_2(\Psi_2)$. Since the basis $\mathsf{m}_1(\Theta_1)$ and $\mathsf{m}_2(\Theta_2)$ are both subsets of the same basis Γ, then for all the (e-type, t-type or o-type) variables ζ such that $\zeta \in dom(\mathsf{m}_1) \cap dom(\mathsf{m}_2)$ we get $\mathsf{m}_1(\zeta) = \mathsf{m}_2(\zeta)$. Therefore the

mapping m

$$\mathsf{m}(\zeta) = \begin{cases} \mathsf{m}_1(\zeta) & \text{if } \zeta \in dom(\mathsf{m}_1) \\ \mathsf{m}_2(\zeta) & \text{if } \zeta \in dom(\mathsf{m}_2) \end{cases}$$

is well defined.
Moreover, since m satisfies Ξ_1, Ξ_2, $\Phi_1 \cap \Phi_2 = \Phi_1 \cap dom(\Psi_2) = \Phi_2 \cap dom(\Psi_1) = \emptyset$, and $\Psi_1 \bowtie \Psi_2$, then m satisfies also all the constraints of the conclusion of the rule $(Iseq)$.

- If the last rule is (par) or $(loop)$ the proof is similar.

□

Now, we put our inference rules at work in order to decide the applicability of Γ-safe rules.

Lemma 4.5. (Characterization of Γ-safe rules)
A rule $P_1 \mapsto P_2$ is a Γ-safe rule if and only if the type mapping m defined by

1. $\mathsf{m}(\varphi_x) = \mathtt{t}$ if $\Gamma(x) = \mathtt{t}$
2. $\mathsf{m}(\phi_\eta) = \mathtt{T}'$ if $\Gamma(\eta) = (\mathtt{T}', \mathtt{O}')$
3. $\mathsf{m}(\psi_\eta) = \mathtt{O}'$ if $\Gamma(\eta) = (\mathtt{T}', \mathtt{O}')$

satisfies the set of constraints $\Xi_1 \cup \Xi_2 \cup \{\Psi_1 = \Psi_2\}$, where $\vdash P_1 : \Theta_1; (\Phi_1, \Psi_1); \Xi_1$ and $\vdash P_2 : \Theta_2; (\Phi_2, \Psi_2); \Xi_2$.

Proof:

($\Leftarrow$) Since $\vdash P_1 : \Theta_1; (\Phi_1, \Psi_1); \Xi_1$, $\vdash P_2 : \Theta_2; (\Phi_2, \Psi_2); \Xi_2$ and m satisfies Ξ_1 and Ξ_2, by applying Theorem 4.3 we derive $\mathsf{m}(\Theta_1) \vdash P_1 : (m(\Phi_1), m(\Psi_1))$, and $\mathsf{m}(\Theta_2) \vdash P_2 : (m(\Phi_2), m(\Psi_2))$. From the definition of m, we have that $\mathsf{m}(\Theta_2) \subseteq \Gamma$ and $\mathsf{m}(\Theta_2) \subseteq \Gamma$, and by weakening we derive that $\Gamma \vdash P_1 : (m(\Phi_1), m(\Psi_1))$ and $\Gamma \vdash P_2 : (m(\Phi_2), m(\Psi_2))$. Moreover, from the fact that m satisfies $\Psi_1 = \Psi_2$, we have that $\mathsf{m}(\Psi_1) = \mathsf{m}(\Psi_2)$. Therefore, $P_1 \mapsto P_2$ is a Γ-safe rule.

($\Rightarrow$) Since $P_1 \mapsto P_2$ is a Γ-safe rule, we have that $\Gamma \vdash P_1 : (\mathtt{T}, \mathtt{O})$ and $\Gamma \vdash P_2 : (\mathtt{T}, \mathtt{O})$. From Theorem 4.4, applied to $\Gamma \vdash P_1 : (\mathtt{T}, \mathtt{O})$, we derive that $\vdash P_1 : \Theta_1; (\Phi_1, \Psi_1); \Xi_1$ and there is a type mapping m_1 satisfying Ξ_1 such that $\Gamma \supseteq \mathsf{m}_1(\Theta_1)$, $\mathtt{T} = \mathsf{m}_1(\Phi_1)$, $\mathtt{O} = \mathsf{m}_1(\Psi_1)$. Applying Theorem 4.4 to $\Gamma \vdash P_2 : (\mathtt{T}, \mathtt{O})$ we derive that $\vdash P_2 : \Theta_2; (\Phi_2, \Psi_2); \Xi_2$ and there is a type mapping m_2 satisfying Ξ_2 such that $\Gamma \supseteq \mathsf{m}_2(\Theta_2)$, $\mathtt{T}' = \mathsf{m}_2(\Phi_2)$, $\mathtt{O} = \mathsf{m}_2(\Psi_2)$. Since the basis $\mathsf{m}_1(\Theta_1)$ and $\mathsf{m}_2(\Theta_2)$ are both subsets of Γ, then, (let ζ is an e-type, t-type, or o-type variable) the mapping m defined by

$$\mathsf{m}(\zeta) = \begin{cases} \mathsf{m}'(\zeta) & \text{if } \zeta \in dom(\mathsf{m}_1) \\ \mathsf{m}''(\zeta) & \text{if } \zeta \in dom(\mathsf{m}_2) \end{cases}$$

is well defined. Moreover, m satisfies $\Xi_1 \cup \Xi_2$, and since $\mathsf{m}_1(\Psi_1) = \mathtt{O} = \mathsf{m}_2(\Psi_2)$, then m also satisfies $\Psi_1 = \Psi_2$.

□

Theorem 4.5. (Applicability of rewrite rules)
Let

$$\vdash P_1 : \Theta_1; (\Phi_1, \Psi_1); \Xi_1\ , \qquad \vdash P_2 : \Theta_2; (\Phi_2, \Psi_2); \Xi_2$$

and $P_1\sigma \not\equiv \epsilon$. Then the rule $P_1 \mapsto P_2$ can be applied to the term $P_1\sigma$ (i.e. $P_1\sigma \xrightarrow{TS} P_2\sigma$) if and only if the type mapping m defined by

1. $\mathsf{m}(\varphi_x) = \mathtt{t}$ if $\sigma(x) : \mathtt{t} \in \Gamma_0$,
2. $\mathsf{m}(\phi_\eta) = \mathtt{T}'$ if $\vdash \sigma(\eta) : (\mathtt{T}', \mathtt{O}')$,
3. $\mathsf{m}(\psi_\eta) = \mathtt{O}'$ if $\vdash \sigma(\eta) : (\mathtt{T}', \mathtt{O}')$,

satisfies the set of constraints $\Xi_1 \cup \Xi_2 \cup \{\Psi_1 = \Psi_2\}$.

Proof:
We define the basis Γ as follows:

- $x : \mathtt{t} \in \Gamma$ if $\sigma(x) : \mathtt{t} \in \Gamma_0$, and
- $\eta : (\mathtt{T}', \mathtt{O}') \in \Gamma$ if $\vdash \sigma(\eta) : (\mathtt{T}', \mathtt{O}')$.

In this way we get that $\sigma \in \Sigma_\Gamma$ and the type mapping m is such that:

1. $\mathsf{m}(\varphi_x) = \mathtt{t}$ iff $x : \mathtt{t} \in \Gamma$
2. $\mathsf{m}(\phi_\eta) = \mathtt{T}'$ iff $\eta : (\mathtt{T}', \mathtt{O}') \in \Gamma$
3. $\mathsf{m}(\psi_\eta) = \mathtt{O}'$ iff $\eta : (\mathtt{T}', \mathtt{O}') \in \Gamma$.

($\Leftarrow$) If the mapping m satisfies the the set of constraints $\Xi_1 \cup \Xi_2 \cup \{\Psi_1 = \Psi_2\}$, then by Lemma 4.5 the rule $P_1 \mapsto P_2$ is Γ-safe and we get $P_1\sigma \xrightarrow{TS} P_2\sigma$ by applying rule (appTS).

($\Rightarrow$) If $P_1\sigma \xrightarrow{TS} P_2\sigma$ by applying rule (appTS), then the rule $P_1 \mapsto P_2$ is Γ-safe and then the mapping m satisfies the the set of constraints $\Xi_1 \cup \Xi_2 \cup \{\Psi_1 = \Psi_2\}$ by Lemma 4.5.

□

Notably the inference for a fixed set of rewrite rules can be done once for all. As already said, the complexity is linear in the number of symbols occurring in the rules. The set of typing conditions so generated is also linear in the number of symbols occurring in the rules. Therefore the applicability of a rewrite rule can be decided in linear time with respect to the number of symbols occurring in the rule.

4.3. Compartment safety versus typed safety

Since the type system of Section 4.2 only allows links between elements of the same type, this comparison is sensible only by assuming that Γ_0 associates the same basic type with all atomic elements and Γ associates the same basic type to all element variables. Under this assumption we can show:

Theorem 4.6. Each compartment safe rule is Γ-safe too, i.e. if $wf(P_1)$ and $cs(P_1, P_2)$, then $\Gamma \vdash P_1 : (\mathtt{T}, \mathtt{O})$ and $\Gamma \vdash P_2 : (\mathtt{T}', \mathtt{O})$ for some $\Gamma, \mathtt{T}, \mathtt{T}', \mathtt{O}$.

Proof:
Easy by induction on the definition of cs. □

The opposite implication is not true, since for example the rule $a^1 \mid \widetilde{x} \mapsto a^1 \mid a^1$ is not compartment safe, but it is Γ-safe for $\Gamma = \{\widetilde{x} : (\emptyset, \{1 : \mathtt{t}\})\}$ when $\Gamma_0 = \{a : \mathtt{t}\}$.

As a consequence of Theorem 4.6 we have that $T \xrightarrow{CS} T'$ implies $T \xrightarrow{TS} T'$, while the opposite implication fails again by the above counter-example.

5. An Application

In this section we compare the notions of compartment safety and typed safety proposed in the previous section under the viewpoint of their applicability to the description of biological systems. In Section 4.3 we have shown that every compartment safe rule is also Γ-safe, and that the vice versa does not hold. Hence, the class of biological systems that can be modelled by means of Γ-safe rules is for sure richer than the class of systems that can be modelled by using compartment safe rules only. On the other hand, the semantics of compartment safety can be computed more efficiently than the one of the typed safety. In fact, the latter asks the checking that the mapping m defined in Theorem 4.5 satisfies the set of constraints required by the same theorem.

The existence of a trade-off between biological expressiveness and efficiency in the computation (and analysis) of the semantics makes the characterization of the two considered classes of biological systems very important. In fact, the two proposed notions of safety are both of some interest only if (i) the class of systems that can be modelled with compartment safe rules includes a relevant part of the biological systems of interest, and (ii) there is a relevant part of the biological systems of interest that can be modelled only if Γ-safe rules are used.

As regards (i), we have already described in Section 4.1 the class of biological systems that can be modelled with compartment safe rules. It includes essentially all of the most common biochemical forms of interaction such as protein/protein and protein/dna bindings (and unbindings), dna transcription and translation, complexation/decomplexation of molecules, enzymatic activities, and so on. An example of biological system in this class is the EGF signallig pathway we described in Section 2. The rules of the LCLS model of such a pathway we have given in Section 3, namely rules (R1')-(R9'), are indeed all compartment safe.

As regards (ii), we consider another biological process involving EGFR proteins in cells, namely the internalization and degradation of such proteins. This is a very important process that is also a target of some oncogenic viruses, such as vCBL and the human papilloma virus, which interfere the process by causing EGFR proteins recycling, with the result of increasing their presence on the cellular membrane and consequently stimulating cell proliferation as an effect of the EGF singalling pathway [21]. We shall see that in order to suitably model the internalization and degradation process in LCLS, compartment safe rules are not enough, whereas Γ-safe are.

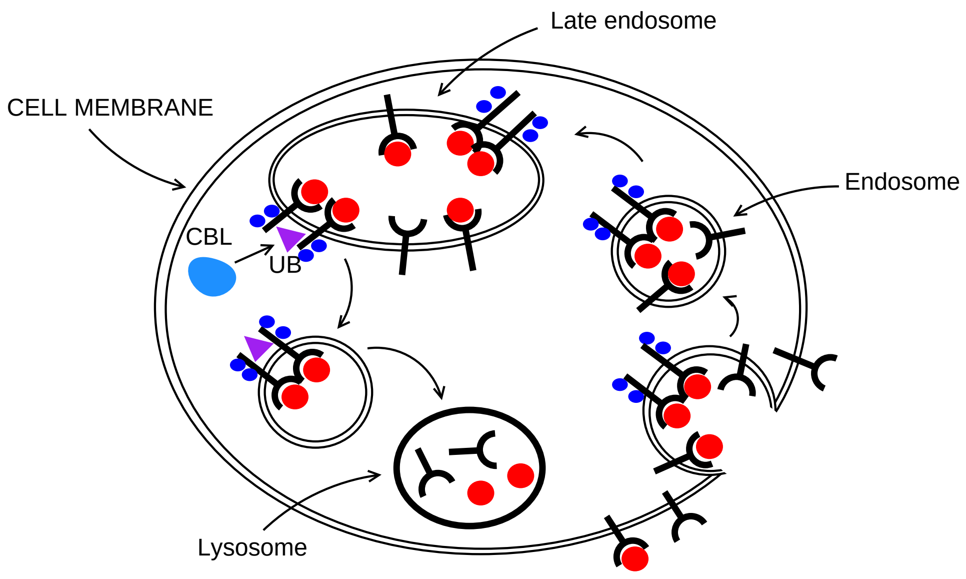

Figure 6. The EGF signaling pathway.

The internalization and degradation process is as follows. After the activation of some effector proteins, ligand-receptor dimers are internalized in endosomes. An endosome consists of a portion of the cellular membrane which forms a vesicle bringing a number of ligand-receptor dimers (but also some individual receptors) inside the cell. Several vesicles join together and form a so called late endosome. Then, a ubiquitin ligase, known as Cbl, binds an ubiquitin protein to a dimer or to a receptor in the late endosome. The ubiquitin protein targets dimers and receptors for degradation in the lysosome. Another vesicle is formed to transport the ubiquinated proteins from the late endosome to the lysosome (see Figure 6).

The following rules describe internalization and degradation of signal-receptor complexes:

$$\begin{aligned}
X \mid \big(m \cdot \widetilde{x} \cdot R_{E1} \cdot \widetilde{y} \cdot R_{I2} \cdot \widetilde{z}\big)^{L} \rfloor Y &\mapsto \big(m \cdot \widetilde{x} \cdot \widetilde{z}\big)^{L} \rfloor (Y \mid \big(endo \cdot R_{E1} \cdot \widetilde{y} \cdot R_{I2}\big)^{L} \rfloor X) && \text{(R10')}\\
\big(endo \cdot \widetilde{x}\big)^{L} \rfloor X \mid \big(late \cdot \widetilde{y}\big)^{L} \rfloor Y &\mapsto \big(late \cdot \widetilde{x} \cdot \widetilde{y}\big)^{L} \rfloor (X \mid Y) && \text{(R11')}\\
EFF^{1} \mid \big(R^{1}_{I2} \cdot \widetilde{x}\big)^{L} \rfloor X &\mapsto EFFp \mid \big(R_{I2} \cdot \widetilde{x}\big)^{L} \rfloor X && \text{(R12')}\\
CBL \mid \big(late \cdot \widetilde{x} \cdot R_{I2} \cdot \widetilde{y}\big)^{L} \rfloor X &\mapsto CBL \mid \big(late \cdot \widetilde{x} \cdot R_{I2ub} \cdot \widetilde{y}\big)^{L} \rfloor X && \text{(R13')}\\
\big(late \cdot \widetilde{x} \cdot \widetilde{y} \cdot R_{I2ub} \cdot \widetilde{w} \cdot \widetilde{z}\big)^{L} \rfloor (X \mid Y) &\mapsto \big(late \cdot \widetilde{x} \cdot \widetilde{z}\big)^{L} \rfloor X \mid \big(endoub \cdot \widetilde{y} \cdot R_{I2ub} \cdot \widetilde{w}\big)^{L} \rfloor Y && \text{(R14')}\\
\big(endoub \cdot \widetilde{x}\big)^{L} \rfloor X \mid \big(lyso\big)^{L} \rfloor Y &\mapsto \big(lyso\big)^{L} \rfloor (\widetilde{x} \mid X \mid Y) && \text{(R15')}\\
\big(lyso\big)^{L} \rfloor (X \mid Y) &\mapsto \big(lyso\big)^{L} \rfloor X && \text{(R16')}
\end{aligned}$$

Rule (R10') models a portion of the cellular membrane forming a vesicle and bringing a number of ligand-receptor dimers inside the cell in the form of an internalized endosome. Rule (R11') describes an endosome vesicle joining a late endosome. By rule (R12’), the effector protein bound to the dimer gets phosphorylated and released. Rule (R13’) describes the ubiquitination of the late endosome performed by the Cbl enzyme which transforms the R_{I2} domain into the R_{I2ub}. A late ubiquanted endosome may form some new vesicle in the form of a ubiquinated endosome: by rule (R14’), parts of the late endosome membrane and content are used to generate an ubiquinated endosome. Ubiquinated endosomes can be encapsulated within lysosomes by rule (R15’). Rule (R16’) models how lysosomes may digest parts of their content.

If we consider the limited EGF pathway model seen in Section 3, starting from a well-formed term, we can statically check that all the terms reached by applying rules (R1’)-(R9’) are still well formed. In fact, as already mentioned, rules (R1’)-(R9’) satisfy the relation of compartment safety.

However, the new rules (R10’)-(R16’) are not compartment safe; except for rules (R12’) and (R13’). Hence, we cannot statically guarantee the well-formedness of the terms reached via the reductions driven by these new rules. In the cases of rules (R11’) and (R14’)-(R16’) we can resort to the notion of typed safety. Just as an example, starting from the initial well-formed term:

$$T = EGF \mid EGF \mid \big(m\big)^L (RIBO \mid EFF \mid CBL \mid \big(n\big)^L \rfloor (POLY \mid DNA))$$

by applying the compartment safe rules (R1’)-(R9’) we can get the well-formed term:

$$\begin{aligned} T' = \;& EGF^1 \mid EGF^2 \mid \big(m \cdot R^1_{E1} \cdot R^3_{E2} \cdot R_p I1 \cdot R_{I2} \cdot R^2_{E1} \cdot R^3_{E2} \cdot R_p I1 \cdot R_{I2}\big)^L \rfloor \\ & (RIBO \mid EFF_p \mid CBL \mid \big(n\big)^L \rfloor (POLY \mid DNA)) \end{aligned}$$

Now, in order to apply rule (R10’) and engulf the endosome inside the cell, we need to resort to the typed semantics. We leave it to the reader to find the correct instantiation σ allowing to reduce T' via (R10’) in a Γ-safe way.

6. Related Work and Conclusions

In this paper we have presented an extension of the Calculus of Looping Sequences (CLS) suitable to describe protein interaction at the domain level. The extended calculus, called Linked Calculus of Looping Sequences (LCLS) is obtained by allowing elements of sequences to be connected by links (denoted as pairs of labels) which can represent bindings between protein domains.

In order to correctly denote links, labels appearing in an LCLS term should occur exactly twice and in a single compartment, with the exception of labels in the top-level compartment of the term which are also allowed to occur only once (thus denoting a link with a component from the environment). We have formalised these requirements on the occurrences of labels in an LCLS term as a well-formedness relation, and we have proposed two approaches (compartment safety and typed safety) to ensure that well-formedness is preserved by rewrite rule applications. We have also shown by means of examples of biological applications, that both the approaches are meaningful.

It is very important to compare our work with the κ-calculus [11, 15]. Actually, the idea of using labels to denote links representing bindings between protein domains is taken from such a calculus. However, LCLS is aimed at describing a much more general class of biological phenomena than that describable by the κ-calculus, with the consequence that the handling of links in LCLS is more complicated.

Let us shortly recall the definition of the κ-calculus as given in [11]. As LCLS, the κ-calculus is a formalism based on term rewriting. Terms denote *graphs-with-sites*, namely graphs in which each node (representing a protein) is enriched with a finite number of sites. The number of sites could be different for different nodes. Edges connect two sites of different nodes and represent protein bindings.

Given a countable set of *protein names* $\mathcal{P}$ ranged over by $A, B, C, \ldots$, and a countable set of *edge names* $\mathcal{E}$ ranged over by $x, y, z, \ldots$, a node of a graph-with-sites (or *protein*) is represented in the κ-calculus as a protein name associated with a partial mapping ρ from $\mathbb{N}$ to $\mathcal{E} \cup \{h, v\}$. The mapping ρ is defined only for values from 1 to the number of sites of the node. The meaning of $\rho(i) = x$ is that site i is connected to a site of another protein by edge x, otherwise we have that site i is not connected and it is either *hidden*, namely not available for connection, if $\rho(i) = h$, or *visible*, namely available for connection, if $\rho(i) = v$.

A whole graph-with-sites (or *solution*) is represented by a multiset of proteins, denoted as a comma separated list. Edges are hence denoted by pairs of edge names occurring in the solution. Binders can be used to define the scope of a edge name, and reuse the same name to denote several edges. In order for a solution to be well formed (or *graph-like*) edge names not in the scope of any binder must occur at most twice, and binders must bind either zero or two occurrences of a edge name.

For the sake of simplicity, let us forget about binders. An example of κ-calculus term is the following solution

$$S \quad = \quad A(1^x + 2 + \overline{3}), B(1 + 2^x), C(1^y + 2)$$

where we have an edge named x between the first site of protein A and the second site of protein B, and (a part of) an edge connected to the first site of protein C. If a line is present over a number it means that the corresponding site is hidden. The site is visible otherwise.

The dynamics of the κ-calculus is driven by the application of rewrite rules. A rewrite rule is a pair L, R, written $L \to R$, of *pre-solutions*, namely solutions in which only a part of the protein sites are mentioned. This allows rules to address only the parts of the proteins that are changed or checked during the modelled reaction. An example of rewrite rule is

$$Rule \quad = \quad A(2 + \overline{3}), C(2) \to A(2^z + 3), C(2^z)$$

that, when applied to S, creates an edge between the second site of A and the second site of C and makes the third site of A visible. In other words, it transforms S into

$$S' \quad = \quad A(1^x + 2^z + 3), B(1 + 2^x), C(1^y + 2^z)\,.$$

Two main classes of rewrite rules are considered in [11], namely *monotonic* and *anti-monotonic* rules. Monotonic and anti-monotonic rules, apart from changing the hidden/visible state of sites, either only add edges or only remove edges, respectively. Both kinds of rules, when applied to a graph-like term, are proved to preserve graph-likeness.

Now, an encoding enc of the κ-calculus into LCLS can be given which translates, for example, solution S into the following LCLS term:

$$enc(S) \quad = \quad A \cdot 1^1 \cdot 2 \cdot \overline{3} \mid B \cdot 1 \cdot 2^1 \mid C \cdot 1^2 \cdot 2$$

and the previously given κ-calculus rewrite rule into the following LCLS rewrite rule:

$$enc(Rule) \quad = \quad A \cdot \widetilde{x} \cdot 2 \cdot \overline{3} \mid C \cdot \widetilde{y} \cdot 2 \mapsto A \cdot \widetilde{x} \cdot 2^1 \cdot 3 \mid C \cdot \widetilde{y} \cdot 2^1\,.$$

The rather simple form of κ-calculus rules, in particular of monotonic and anti-monotonic ones, can be exploited to prove that the translation of such rules gives compartment safe LCLS rules. This implies that graph-likeness in the κ-calculus is a special case of well-formedness of LCLS terms.

Summing up, LCLS combines the advantages of a formalism capable of describing a rather general class of biological phenomena with the advantages of the modelling of protein interaction at the domain level. Further extensions could be defined to describe also quantitative and stochastic aspects of biological systems by following approaches already available for CLS [5]. Possible further work aimed at facing the size and complexity of real biological systems may consist in the development of static analysis techniques for LCLS.

Acknowledgement

We thank the anonymous reviewers for feedback and good suggestions.

References

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

[2] R. Alur, C. Belta, and F. Ivancic. Hybrid modeling and simulation of biomolecular networks. In *HSCC*, volume 2034 of *LNCS*, pages 19–32. Springer, 2001.

[3] B. Aman, M. Dezani-Ciancaglini, and A. Troina. Type Disciplines for Analysing Biologically Relevant Properties. In *MeCBIC'08*, volume 227 of *ENTCS*, pages 97–111. Elsevier, 2009.

[4] R. Barbuti, A. Maggiolo-Schettini, and P. Milazzo. Extending the calculus of looping sequences to model protein interaction at. In *ISBRA'07*, volume 4463 of *LNBI*, pages 638 – 649. Springer, 2006.

[5] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, P. Tiberi, and A. Troina. Stochastic calculus of looping sequences for the modelling and simulation of cellular pathways. *Transactions on Computational Systems Biology*, IX:86 – 113, 2008.

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

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

[8] L. Bioglio. Enumerated type semantics for the calculus of looping sequences. *ITA*, 2010. Submitted.

[9] S. Capecchi and A. Troina. Types for BioAmbients. In *FBTC'10*, volume 19 of *EPTCS*, pages 103 – 115, 2010.

[10] L. Cardelli. Brane calculi. In *CMSB'05*, volume 3082 of *LNCS*, pages 257–278. Springer, 2005.

[11] V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[12] M. Dezani-Ciancaglini, P. Giannini, and A. Troina. A type system for a stochastic CLS. In *MeCBIC'09*, volume 11 of *EPTCS*, pages 91 – 106, 2009.

[13] M. Dezani-Ciancaglini, P. Giannini, and A. Troina. A type system for required/excluded elements in CLS. In *DCM'09*, volume 9 of *EPTCS*, pages 38 – 48, 2009.

[14] F. Fages and S. Soliman. Abstract interpretation and types for systems biology. *Theoretical Computer Science*, 403(1):52–70, 2008.

[15] C. Laneve and F. Tarissan. A simple calculus for proteins and cells. In *MeCBIC'06*, volume 171 of *ENTCS*, pages 139–154. Elsevier, 2007.

[16] H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano. Hybrid Petri net representation of gene regulatory network. In *Pacific Symposium on Biocomputing*, pages 341–352. World Scientific Press, 2000.

[17] G. Păun. *Membrane computing. An introduction*. Springer, 2002.

[18] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. Bioambients: An abstraction for biological compartments. *Theoretical Computer Science*, 325:141–167, 2004.

[19] A. Regev and E. Shapiro. Cells as computation. *Nature*, 419(6905):343, 2002.

[20] A. Regev and E. Shapiro. The π-calculus as an abstraction for biomolecular systems. *Modelling in Molecular Biology*, pages 219–266, 2004.

[21] S. Straight, P. Hinkle, R. Jewers, and D. McCance. The E5 oncoprotein of human papillomavirus type 16 transforms fibroblasts and effects the downregulation of the epidermial growth factor receptor in keratinocytes. *Journal of Virology*, 67:4521–4532, 1993.