A Calculus of Looping Sequences with Local Rules*

Livio Bioglio
Dip. di Informatica, Univ. di Torino, Italy

Mariangiola Dezani-Ciancaglini
Dip. di Informatica, Univ. di Torino, Italy

Paola Giannini
Dip. di Informatica, Univ. del Piemonte Orientale, Italy

Angelo Troina
Dip. di Informatica, Univ. di Torino, Italy

In this paper we present a variant of the Calculus of Looping Sequences (CLS for short) with global and local rewrite rules. While global rules, as in CLS, are applied anywhere in a given term, local rules can only be applied in the compartment on which they are defined. Local rules are dynamic: they can be added, moved and erased. We enrich the new calculus with a parallel semantics where a reduction step is lead by any number of global and local rules that could be performed in parallel. A type system is developed to enforce the property that a compartment must contain only local rules with specific features. As a running example we model some interactions happening in a cell starting from its nucleus and moving towards its mitochondria.

1 Introduction

The Calculus of Looping Sequences (CLS for short) [5, 4, 15, 6], is a formalism for describing biological systems and their evolution. CLS is based on term rewriting with a set of predefined rules modelling the activities one would like to describe. CLS terms are constructed by starting from a set of basic constituent elements which are composed with operators of sequencing, looping, containment and parallel composition. Sequences may represent DNA fragments and proteins, looping sequences may represent membranes, parallel composition may represent juxtaposition of elements and populations of chemical species.

The model has been extended with several features such as bisimulations [4, 7], combining the simplicity of notation of rewrite systems with the advantage of a form of compositionality. In [1, 2] a type system was defined to ensure the well-formedness of links between protein sites within the Linked Calculus of Looping Sequences (see [3]). In [13] we defined a type system to guarantee the soundness of reduction rules with respect to the requirement of certain elements, and the repellency of others.

In this paper we present a variant of CLS with global and local rewrite rules (CLSLR, for short). Global rules are applied anywhere in a given term wherever their patterns match the portion of the system under investigation, local rules can only be applied in the compartment in which they are defined. Terms written in CLSLR are thus syntactically extended to contain explicit local rules within the term, on different compartments. Local rules can be created, moved between different compartments and deleted. We feel that having a calculus in which we can model the dynamic evolution of the rules describing the system results in a more natural and direct way to study emerging properties of complex systems. As it happens in nature, where *data* and *programs* are encoded in the same kind of molecular structures, we insert rewrite rules within the terms modelling the system under investigation.

In CLSLR we also enrich CLS with a parallel semantics in which we define a reduction step lead by any number of global and local rules that could be performed in parallel.

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

To appear in EPTCS.

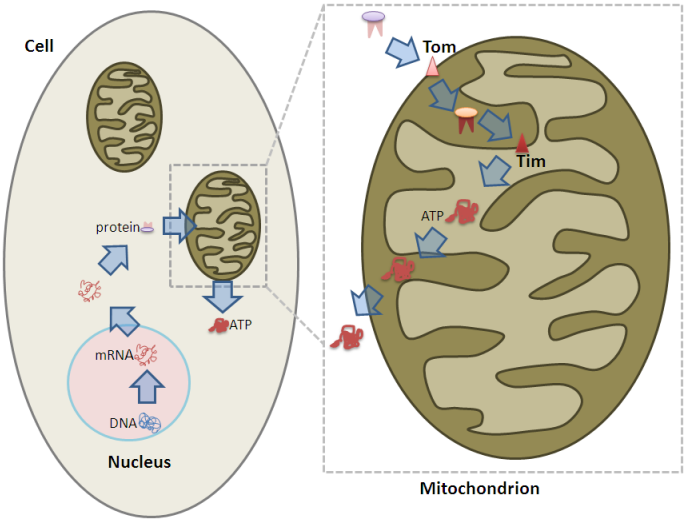

Figure 1: From the cell nucleus to mitochondria.

Since in this framework the focus is put on local rules, we define a set of *features* that can be associated to each local rule. Features may define general properties of rewrite rules or properties which are strictly related to the model under investigation. We define a *membrane type* for the compartments of our model and develop a type systems enforcing the property that a compartment must contain only local rules with specific features.

Thus, the main features of CLSLR are:

- different compartments can evolve according to different *local* rules;
- the set of *global* rules is fixed;
- local rules are *dynamic*: they can be added, moved and erased according to both global and local rules;
- a *parallel* reduction step permits the application of several global and local rules;
- compartments are enforced to contain only rules with specific *features*.

As a running case study, emphasising the peculiarities of the calculus, we consider some mitochondrial activity underlining the form of symbiosis between a cell and its mitochondria (see [14]). Mitochondria are membrane-enclosed organelle found in eukaryotic cells that generate most of the cell's energy supply in the form of adenosine triphosphate (ATP). A mitochondrion is formed by two membranes, the outer and the inner membrane, having different properties and proteins on their surfaces. Both membranes have receptors to mediate the entrance of molecules. In Figure 1 we show the expression of a gene (encoded in the DNA within the nucleus of the cell) destined to be translated into a protein that will be catch by mitochondria and will then catalyse the production of ATP. In particular, we will model the following steps: (1) genes within the nucleus' DNA are transcribed into mRNA, (2) mRNA moves from the nucleus to the cell's cytoplasm, (3) where it is translated into the protein.

The vast majority of proteins destined for the mitochondria are encoded in the nucleus of the cell and synthesised in the cytoplasm. These are tagged by a signal sequence, which is recognised by a receptor protein in the Transporter Outer Membrane complex (TOM). The signal sequence and adjacent portions

of the polypeptide chain are inserted in the intermembranous space through the TOM complex, then in the mitochondrion internal space through a Transporter Inner Membrane complex (TIM). According to this description, we model the following final steps: (4) protein is detected by TOM and brought within the intermembranous space, (5) then, through TIM, in the mitochondrion's inside, (6) where it catalyses the production of ATP, (7) that exits the inner, (8) and the outer mitochondrial membranes towards the cell's cytoplasm.

2 The calculus

In this Section we present the Calculus of Looping Sequences with Local Rules (CLSLR).

2.1 Syntax of CLSLR

We assume a possibly infinite alphabet $\mathscr{E}$ of symbols ranged over by $a,b,c,\ldots$, a set of element variables $\mathscr{X}$ ranged over by $x,y,z,\ldots$, a set of sequence variables $\mathscr{SV}$ ranged over by $\widetilde{x},\widetilde{y},\widetilde{z},\ldots$, and a set of term variables $\mathscr{TV}$ ranged over by $X,Y,Z,\ldots$. All these sets are possibly infinite and pairwise disjoint. We denote by $\mathscr{V}$ the set of all variables, $\mathscr{V}=\mathscr{X}\cup\mathscr{SV}\cup\mathscr{TV}$, and with χ a generic variable of $\mathscr{V}$. Hence, a pattern is a term that may include variables.

Definition 2.1. *[Patterns]* Patterns *P*, sequence patterns *SP and* local rules *R of* CLS *are given by the following grammar:*

$$\begin{array}{lcl} P & ::= & (SP)^{\circlearrowleft} \rfloor P \;\mid\; P \mid P \;\mid\; L \\ SP & ::= & \varepsilon \;\mid\; a \;\mid\; x \;\mid\; SP\cdot SP \;\mid\; \widetilde{x} \\ L & ::= & X \;\mid\; SP \;\mid\; L \mid L \;\mid\; R \\ R & ::= & L\mapsto L \;\mid\; L^{\uparrow SP}\mapsto L^{\uparrow SP} \;\mid\; L^{\downarrow SP}\mapsto L^{\downarrow SP} \end{array}$$

where a is a generic element of $\mathscr{E}$, and $X,\widetilde{x}$ and x are generic elements of $\mathscr{TV},\mathscr{SV}$ and $\mathscr{X}$, respectively. Sequence patterns SP defines patterns for sequences of elements, $(SP)^{\circlearrowleft}$ denotes a closed (looping) sequence which may contain other patterns through the $\rfloor$ operator, $\mid$ is used to denote the parallel composition (juxtaposition) of patterns, and R denotes the syntax of local rules that may either exit ($L^{\uparrow SP}$) or enter ($L^{\downarrow SP}$) a closed sequence SP. We denote with $\mathscr{P}$ the infinite set of patterns.
A local rule $L_1\mapsto L_2$ is well formed *if $L_1\not\equiv\varepsilon$ and $Var(L_2)\subseteq Var(L_1)$, where $Var(P)$ denotes set of variables appearing in P.*
A local rule $L_1^{\uparrow SP_1}\mapsto L_2^{\uparrow SP_2}$ or *$L_1^{\downarrow SP_1}\mapsto L_2^{\downarrow SP_2}$* is well formed *if $L_1\not\equiv\varepsilon$, $Var(L_2)\subseteq Var(L_1)$ and $Var(SP_2)\subseteq Var(SP_1)$.*

Terms are patterns containing variables only inside local rules. Sequences are closed sequence patterns. We denote with $\mathscr{T}$ the infinite set of terms, ranged over by T, and with $\mathscr{S}$ the infinite set of sequences, ranged over by S.

An *instantiation* is a partial function $\sigma:(\mathscr{TV}\to\mathscr{T})\cup(\mathscr{SV}\to\mathscr{S})\cup(\mathscr{X}\to\mathscr{E})$. Given $P\in\mathscr{P}$, with $P\sigma$ we denote the term obtained by replacing each occurrence of each variable $\chi\in\mathscr{V}$ appearing in P with the corresponding term $\sigma(\chi)$, but for local rules, which are left unchanged by instantiations, i.e., $R\sigma=R$ for all R and σ. With Σ we denote the set of all the possible instantiations.

Definition 2.2 (Structural Congruence)**.** *The structural congruence relations $\equiv_S$ and $\equiv_R$ and $\equiv_P$ are the least congruence relations on sequence patterns, local rules and on patterns, respectively, satisfying the rules shown in Figure 2.*

$$SP_1 \cdot (SP_2 \cdot SP_3) \equiv_S (SP_1 \cdot SP_2) \cdot SP_3 \qquad SP \cdot \varepsilon \equiv_S \varepsilon \cdot SP \equiv_S SP$$
$$SP_1 \equiv_S SP_2 \text{ implies } SP_1 \equiv_P SP_2 \text{ and } (SP_1)^{\circlearrowleft} \rfloor P \equiv_P (SP_2)^{\circlearrowleft} \rfloor P$$
$$L_1 \equiv_P L'_1 \text{ and } L_2 \equiv_P L'_2 \text{ imply } L_1 \mapsto L_2 \equiv_R L'_1 \mapsto L'_2$$
$$L_1 \equiv_P L'_1 \text{ and } L_2 \equiv_P L'_2 \text{ and } SP_1 \equiv_P SP'_1 \text{ and } SP_2 \equiv_P SP'_2$$
$$\text{imply } L_1^{\uparrow SP_1} \mapsto L_2^{\uparrow SP_2} \equiv_R L'^{\uparrow SP'_1}_1 \mapsto L'^{\uparrow SP'_2}_2 \text{ and } L_1^{\downarrow SP_1} \mapsto L_2^{\downarrow SP_2} \equiv_R L'^{\downarrow SP'_1}_1 \mapsto L'^{\downarrow SP'_2}_2$$
$$R_1 \equiv_R R_2 \text{ implies } R_1 \equiv_P R_2$$
$$P_1 \mid P_2 \equiv_P P_2 \mid P_1 \qquad P_1 \mid (P_2 \mid P_3) \equiv_P (P_1 \mid P_2) \mid P_3 \qquad P \mid \varepsilon \equiv_P P$$
$$(\varepsilon)^{\circlearrowleft} \rfloor \varepsilon \equiv_P \varepsilon \qquad (SP_1 \cdot SP_2)^{\circlearrowleft} \rfloor P \equiv_P (SP_2 \cdot SP_1)^{\circlearrowleft} \rfloor P$$

Figure 2: Structural Congruence

Structural congruence rules the state the associativity of $\cdot$ and $\mid$, the commutativity of the latter and the neutral role of ε. Moreover, axiom $(SP_1 \cdot SP_2)^{\circlearrowleft} \rfloor P \equiv_P (SP_2 \cdot SP_1)^{\circlearrowleft} \rfloor P$ says that looping sequences can rotate. In the following, for simplicity, we will use $\equiv$ in place of $\equiv_P$.

2.2 (Parallel) Operational Semantics

In order to define a reduction step in which (possibly) more than one rule is applied, following [10], we first define the application of a single rule, either global or local. We resort to the standard notion of evaluation contexts.

Definition 2.3 (Contexts)**.** Evaluation Contexts E *are defined as:*

$$E ::= \Box \quad \mid \quad E \mid T \quad \mid \quad T \mid E \quad \mid \quad (S)^{\circlearrowleft} \rfloor E$$

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

By definition, every evaluation context contains a single hole $\Box$. Let us assume $E \in \mathbb{E}$, with $E[T]$ we denote the term obtained by replacing $\Box$ with T in E. The structural equivalence is extended to contexts in the natural way (by considering $\Box$ as a new and unique symbol of the alphabet $\mathscr{E}$). Note that the shape of evaluation contexts does not permit to have holes in sequences. A rewrite rule introducing a parallel composition on the right hand side (as $a \mapsto b \mid c$) applied to an element of a sequence (e.g., $m \cdot a \cdot m$) would result into a syntactically incorrect term (in this case $m \cdot (b \mid c) \cdot m$).

To enforce the fact that local and global rule applications can be done in parallel, we underline those subterms that are produced by the application of the rule involved. Terms matching the left-hand-side of a (local or global) rule must not have any underlined subterm. Underlined terms are only used for bookkeeping in the definition of rule application.

Let $\underline{\mathscr{T}}$ denote the set of terms in which some subterms can be underlined. The erasing mapping $\eta : \underline{\mathscr{T}} \mapsto \mathscr{T}$ erases all underlining obtaining a term generated by the grammar of Definition 2.1.

We first define the application of global or local rules to terms in $\underline{\mathscr{T}}$ which produces terms in $\underline{\mathscr{T}}$.

Global rules are of the shape $P_1 \mapsto P_2$. They can be applied to terms only if they occur in a legal evaluation context.

Local rules are inside compartments, and can be applied only if a term matching the left-hand-side of the rule occurs in the same compartment.

Notice that global rules have patterns P on both the left and the right-hand side of the rules, whereas local rules have the less general L patterns. In particular, L patterns do not contain compartments, and therefore cannot change the nesting structure of the compartments of a term.

Definition 2.4 (Rule Application)**.** *Given a finite set of global rules $\mathscr{R}$,* the rule application $\rightarrow$ *is the least relation closed with respect to $\equiv$ defined by:*

$$\frac{P_1 \mapsto P_2 \in \mathscr{R} \quad \sigma \in \Sigma \quad P_1\sigma \not\equiv \varepsilon}{E[P_1\sigma] \rightarrow E[\underline{P_2\sigma}]}\ (\text{GRT})$$

$$\frac{\sigma \in \Sigma \quad L_1\sigma \not\equiv \varepsilon}{E[L_1 \mapsto L_2 \mid L_1\sigma \mid T] \rightarrow E[L_1 \mapsto L_2 \mid \underline{L_2\sigma} \mid T]}\ (\text{LR})$$

$$\frac{\sigma \in \Sigma \quad L_1\sigma \not\equiv \varepsilon}{E[(S_1\sigma)^{\circlearrowleft} \rfloor (L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid L_1\sigma \mid T)] \rightarrow E[\underline{L_2\sigma} \mid \underline{(S_2\sigma)^{\circlearrowleft}} \rfloor (L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid T)]}\ (\text{LR-Out})$$

$$\frac{\sigma \in \Sigma \quad L_1\sigma \not\equiv \varepsilon}{E[L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid L_1\sigma \mid (S_1\sigma)^{\circlearrowleft} \rfloor T] \rightarrow E[L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid \underline{(S_2\sigma)^{\circlearrowleft}} \rfloor (T \mid \underline{L_2\sigma})]}\ (\text{LR-In})$$

With rule (LR-Out) a term or a rule $L_1\sigma$ exits from the membrane in which it is contained only if the membrane has as loop the sequence $S_1\sigma$: when outside, $L_1\sigma$ is transformed into $L_2\sigma$, and the sequence $S_1\sigma$ into $S_2\sigma$. (LR-In) is similar, but it moves terms or rules into local membranes. Local rules do not permit to move, create or delete membranes: only global rules can do that.

Observe that local rules can be dynamically added and deleted both by global and local rules. A global rule which adds the local rule R is of the shape $P \mapsto R$ and a global rule which erases the same local rule is of the shape $R \mapsto P$. A local rule which adds the local rule R is of the shape $L \mapsto R$ and a local rule which erases the same local rule is of the shape $R \mapsto L$. Moreover local rules can add local rules in compartments separated by just one membrane, since they can be of the shapes $L^{\uparrow S} \mapsto R^{\uparrow S'}$ or $L^{\downarrow S} \mapsto R^{\downarrow S'}$.

A reduction step of the parallel semantics $\Longrightarrow$, starting from a term in $\mathscr{T}$ applies any number of global or local rules that could be performed in parallel, producing a final term in $\mathscr{T}$ (with no underlined subterms).

Definition 2.5 (Parallel Reduction)**.** *The reduction $\Longrightarrow$ between terms in $\mathscr{T}$ is defined by:*

$$\frac{T = T_0 \rightarrow T_1 \rightarrow \cdots \rightarrow T_{n+1} \quad n \geq 0 \quad T' = \eta(T_{n+1})}{T \Longrightarrow T'}$$

To justify the definition of the reduction $\Longrightarrow$ we have to show that the order in which the local or global rules are applied is not important. The notion of multi-hole context, i.e., of term where some disjoint subterms are replaced by holes is handy. More precisely the syntax of *multi-hole contexts* is:

$$C ::= \Box \quad \mid \quad T \quad \mid \quad C \mid C \quad \mid \quad (\Box)^{\circlearrowleft} \rfloor C \quad \mid \quad (S)^{\circlearrowleft} \rfloor C$$

We can show that in a parallel reduction only disjoint subterms can change. Rules (GRT) and (LR) modify just one subterm. Rules (LR-Out) and (LR-In) modify three subterms, i.e., a membrane and a term exiting or entering the membrane, using ε for the missing term.

Theorem 2.6. *If $T \Longrightarrow T'$, then there is a multi-hole context $C[\,]\ldots[\,]$ and terms $T_1,\ldots,T_n$, $T'_1,\ldots,T'_n$ such that $T \equiv C[T_1]\ldots[T_n]$, $T' \equiv C[T'_1]\ldots[T'_n]$ and for all $1 \leq i \leq n$:*

- *either $C[T_1^*]\ldots[T_i]\ldots[T_n^*] \rightarrow C[T_1^*]\ldots[T'_i]\ldots[T_n^*]$*

- *or there are i_1, i_2, i_3 such that $i \in \{i_1, i_2, i_3\}$ and*
  $C[T_1^*]\dots[T_{i_1}] \mid ([T_{i_2}])^{\circlearrowleft} \rfloor [T_{i_3}]\dots[T_n^*] \to C[T_1^*]\dots[T'_{i_1}] \mid \left([T'_{i_2}]\right)^{\circlearrowleft} \rfloor [T'_{i_3}]\dots[T_n^*]$

where T_j^* can be either T_j (subterm of T) or T'_j (subterm of T').

Proof. If $T \Longrightarrow T'$, then for some $U_0, \dots, U_{m+1}$ we get $T = U_0 \to \cdots \to U_{m+1}$ where $m \geq 0$ and $T' = \eta(U_{m+1})$. We show by induction on $h \leq m$ and by cases on the last applied reduction rule that

- *either $U_h = C[T_1^*]\dots[T_i]\dots[T_n^*]$ and $U_{h+1} = C[T_1^*]\dots[T'_i]\dots[T_n^*]$*
- *or there are i_1, i_2, i_3 such that $i \in \{i_1, i_2, i_3\}$ and*
  $U_h = C[T_1^*]\dots[T_{i_1}] \mid ([T_{i_2}])^{\circlearrowleft} \rfloor [T_{i_3}]\dots[T_n^*]$ *and* $U_{h+1} = C[T_1^*]\dots[T'_{i_1}] \mid \left([T'_{i_2}]\right)^{\circlearrowleft} \rfloor [T'_{i_3}]\dots[T_n^*]$

where " * " can be either " " or " $'$ " and all terms with $'$ are either underlined or ε.

If the last applied rule is

$$\frac{\sigma \in \Sigma \quad L_1\sigma \not\equiv \varepsilon}{E[L_1 \mapsto L_2 \mid L_1\sigma \mid V] \to E[L_1 \mapsto L_2 \mid \underline{L_2\sigma} \mid V]}$$

then $U_h = E[L_1 \mapsto L_2 \mid L_1\sigma \mid V]$ and $U_{h+1} = E[L_1 \mapsto L_2 \mid \underline{L_2\sigma} \mid V]$. By induction $U_h = C[T_1^*]\dots[T_n^*]$. Since $L_1\sigma$ is a subterm of T and $\underline{L_2\sigma}$ is a subterm of U_{m+1} there must be an index i such that $T_i = L_1\sigma$ and $T'_i = \underline{L_2\sigma}$.

If the last applied rule is

$$\frac{\sigma \in \Sigma \quad L\sigma \not\equiv \varepsilon \quad L_1\sigma \in \mathscr{T} \quad S_1\sigma \in \mathscr{T}}{E[L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid L_1\sigma \mid (S_1\sigma)^{\circlearrowleft} \rfloor V] \to E[L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid \left(\underline{S_2\sigma}\right)^{\circlearrowleft} \rfloor (V \mid \underline{L_2\sigma})]}$$

then $U_h = E[L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid L_1\sigma \mid (S_1\sigma)^{\circlearrowleft} \rfloor V]$ and $U_{h+1} = E[L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid \left(\underline{S_2\sigma}\right)^{\circlearrowleft} \rfloor (V \mid \underline{L_2\sigma})]$. By induction $U_h = C[T_1^*]\dots[T_n^*]$. Notice that $L_1\sigma, S_1\sigma$ are subterms of T and $\underline{L_2\sigma}, \underline{S_2\sigma}$ are subterm of U_{m+1}. Moreover $L_1\sigma$, $S_1\sigma$ and ε in T are replaced by ε, $\underline{S_2\sigma}$ and $\underline{L_2\sigma}$ in U_{m+1}, respectively. Therefore there must be indexes i_1, i_2, i_3 such that $T_{i_1} = L_1\sigma$, $T'_{i_1} = \varepsilon$, $T_{i_2} = S_1\sigma$, $T'_{i_2} = \underline{S_2\sigma}$, $T_{i_3} = \varepsilon$, $T'_{i_3} = \underline{L_2\sigma}$.

□

Example 2.7. *[Mitochondria Running Example: Syntax and Reductions] A CLSLR term representing the mitochondria evolution inside the cell's activity discussed in the introduction could be:*

$$\begin{array}{rll} \mathtt{CELL} = (cell)^{\circlearrowleft} \rfloor (& \mathtt{NUCLEUS} \mid \mathtt{MITOCH} \mid \dots \mid \mathtt{MITOCH} \mid & \\ & mRNA \mapsto protein \mid & \\ & protein^{\downarrow Tom} \mapsto protein^{\downarrow Tom} &) \end{array}$$

A cell is composed by its membrane (here just represented by the element cell) and its content (in this case, the nucleus, a certain number of mitochondria and a few rules modelling the activity of interest). In particular, the two rules above model the steps (3) and (4), respectively, of the example schematised in the introduction.

Assuming that DNA *is the sequence of genes representing the cell's DNA, and g is the particular gene (contained in* DNA*) codifying the protein, we define the nucleus of the cell with the CLSLR term:*

$$\begin{array}{rll} \mathtt{NUCLEUS} = (nucleus)^{\circlearrowleft} \rfloor (& \mathsf{DNA} \mid & \\ & \widetilde{x} \cdot g \cdot \widetilde{y} \mapsto (\widetilde{x} \cdot g \cdot \widetilde{y} \mid mRNA) \mid & \\ & mRNA^{\uparrow nucleus} \mapsto mRNA^{\uparrow nucleus} &) \end{array}$$

$$
\begin{array}{lll}
\ldots & \Longrightarrow^{+} & (cell)^{\circlearrowleft} \rfloor ((nucleus)^{\circlearrowleft} \rfloor (mRNA \mid \ldots \mid mRNA \mid \ldots) \mid \ldots) \\
& \Longrightarrow^{+} & (cell)^{\circlearrowleft} \rfloor (mRNA \mid \ldots \mid mRNA \mid \ldots) \\
& \Longrightarrow & (cell)^{\circlearrowleft} \rfloor (protein \mid \ldots \mid protein \mid \ldots) \\
& \Longrightarrow & (cell)^{\circlearrowleft} \rfloor ((Tom)^{\circlearrowleft} \rfloor (protein \mid \ldots) \mid \ldots \mid (Tom)^{\circlearrowleft} \rfloor (protein \mid \ldots) \mid \ldots) \\
& \Longrightarrow & (cell)^{\circlearrowleft} \rfloor ((Tom)^{\circlearrowleft} \rfloor ((Tim)^{\circlearrowleft} \rfloor (Mit_A \mapsto (Mit_A \mid ATP) \mid \ldots) \mid \ldots) \mid \\
& & \quad (Tom)^{\circlearrowleft} \rfloor ((Tim)^{\circlearrowleft} \rfloor (Mit_A \mapsto (Mit_A \mid ATP) \mid \ldots) \mid \ldots) \mid \ldots) \\
& \Longrightarrow & (cell)^{\circlearrowleft} \rfloor ((Tom)^{\circlearrowleft} \rfloor ((Tim)^{\circlearrowleft} \rfloor (ATP \mid \ldots) \mid \ldots) \mid \\
& & \quad (Tom)^{\circlearrowleft} \rfloor ((Tim)^{\circlearrowleft} \rfloor (ATP \mid \ldots) \mid \ldots) \mid \ldots) \\
& \Longrightarrow & (cell)^{\circlearrowleft} \rfloor ((Tom)^{\circlearrowleft} \rfloor (ATP \mid (Tim)^{\circlearrowleft} \rfloor (ATP \mid \ldots) \mid \ldots) \mid \\
& & \quad (Tom)^{\circlearrowleft} \rfloor (ATP \mid (Tim)^{\circlearrowleft} \rfloor (ATP \mid \ldots) \mid \ldots) \mid \ldots) \\
& \Longrightarrow & (cell)^{\circlearrowleft} \rfloor (ATP \mid \ldots \mid ATP \mid (Tom)^{\circlearrowleft} \rfloor (ATP \mid (Tim)^{\circlearrowleft} \rfloor (ATP \mid \ldots) \mid \ldots) \mid \\
& & \quad (Tom)^{\circlearrowleft} \rfloor (ATP \mid (Tim)^{\circlearrowleft} \rfloor (ATP \mid \ldots) \mid \ldots) \mid \ldots)
\end{array}
$$

Figure 3: Mitochondria evolution

Note that the first of the two rules above models step (1) of our example (DNA transcription into mRNA), the second one (mRNA exits the nucleus) models step (2).

The mitochondria of our model are composed of a membrane, on which we point out the Tom complex, containing an inner membrane (`INN_MITOCH`) and a couple of rules:

$$
\begin{array}{lll}
\texttt{MITOCH} = (Tom)^{\circlearrowleft} \rfloor (& \texttt{INN\_MITOCH} \mid & \\
& protein^{\downarrow Tim} \mapsto (Mit_A \mapsto (Mit_A \mid ATP))^{\downarrow Tim} \mid & \\
& ATP^{\uparrow \widetilde{x}} \mapsto ATP^{\uparrow \widetilde{x}} \quad) &
\end{array}
$$

where we denote with the element Mit_A a mitochondrial factor inside the inner membrane (activated by our protein), necessary to produce ATP. In particular, the protein, when in the intermembranous space, is moved through Tim inside the inner mitochondrial space (step (5) of our example) and then transformed into the newly generated rule $Mit_A \mapsto (Mit_A \mid ATP)$ which will lead the production of ATP (step (6) of our example).

Finally, in `INN_MITOCH` we point out the Tim complex:

$$
\begin{array}{lll}
\texttt{INN\_MITOCH} = (Tim)^{\circlearrowleft} \rfloor (& Mit_A \mid & \\
& ATP^{\uparrow \widetilde{x}} \mapsto ATP^{\uparrow \widetilde{x}} \quad) &
\end{array}
$$

Both in `MITOCH` and `INN_MITOCH` we have the rules needed to transport the ATP towards the cell's cytoplasm (steps (7) and (8) of the example).

A possible (parallel) reduction of this term, when g initially produces a certain number of mRNA is (by focusing only on the more interesting changes) shown in Figure 3. The ATP produced in the last but two reductions in `INN_MITOCH` moves to `MITOCH` in the last but one reduction and new ATP is produced in `INN_MITOCH`. In the last reduction, the firstly generated ATP moves to the cell, the secondly generated ATP moves to `MITOCH` and new ATP is produced in `INN_MITOCH`.

3 Types

In this section we introduce a type system that enforces the fact that compartments must contain rules having specific features. E.g., in [16] the following *rule features* for $L_1 \mapsto L_2$ are suggested:

- the rule is *deleting* if $Vars(L_1) \supset Vars(L_2)$ (denoted by d);
- the rule is *replicating* if some variable in L_2 occurs twice (denoted by r);
- the rule is *splitting* if L_1 has a subterm containing two different variables (denoted by s);
- the rule is *equating* if some variable in L_1 occurs twice (denoted by e).

This kind of features reflects a structure of rewrite features which could be common for rewrite systems in general. Other, model-dependent, features could be defined to reflect peculiarities and properties of the particular model under investigation. The features of the rules allowed in a compartment are controlled by the wrapping sequence of the compartment. Our *typing system* and the consequent *typed parallel reduction* ensure that, in spite of the facts that reducing a term may move rules in and out of compartments, compartments always contain rules permitted by their wrapping sequence. In addition to the previous features of rules we say that:

- the feature of rule $L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2}$ is that it is an *out rule* (denoted by o);
- the feature of rule $L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2}$ is that it is an *in rule* (denoted by i).

To express the control of the wrapping sequence over the content of the compartment, we associate a subset φ of $\{\mathsf{d},\mathsf{r},\mathsf{s},\mathsf{e},\mathsf{o},\mathsf{i}\}$ to every element in $\mathscr{E}$. This is called a *membrane type*. We use Λ to denote a classification of elements. The type assignment in Figure 4, where a basis Δ assigns membrane types to element and sequence variables, defines the type of a sequence as the union of the membrane types of its elements.

$$\Delta \vdash_s \varepsilon : \emptyset \quad (\text{TSEPS}) \qquad \Delta, \chi : \varphi \vdash_s \chi : \varphi \quad (\text{TSVAR}) \qquad \frac{a : \varphi \in \Lambda}{\Delta \vdash_s a : \varphi}\ (\text{TSELM})$$

$$\frac{\Delta \vdash_s SP : \varphi \quad \Delta \vdash_s SP' : \varphi'}{\Delta \vdash_s SP \cdot SP' : \varphi \cup \varphi'}\ (\text{TSSEQ})$$

Figure 4: Typing Rules for Membranes

To define the type of patterns, that may contain parallel (composition) of rules, we consider:

1. the features of the rules, contained in the pattern, and
2. in case there are output rules the type of the rules that are emitted by these output rules.

Therefore a *pattern type*, denoted by τ, is a sequence of membrane types, i.e., $\tau \in \{\varphi\}^*$. With $\emptyset$ we denote the empty sequence. If the parallel composition of local rules $R_1 \mid \cdots \mid R_n$, $n \geq 0$, has type $\varphi \cdot \tau$, then φ is the union of the features of the rules R_i ($1 \leq i \leq n$), and τ is the type of the parallel composition of rules in L' for those L' such that $R_i = L^{\uparrow SP} \mapsto L'^{\uparrow SP'}$ for some i, $1 \leq i \leq n$ (the type of the parallel composition of the rules that are emitted). If no rule is emitted, then $\tau = \emptyset$.

Union of pattern types, $\sqcup$, is defined inductively by:

- $\emptyset \sqcup \tau = \tau \sqcup \emptyset = \tau$, and
- $\varphi_1 \cdot \tau_1 \sqcup \varphi_2 \cdot \tau_2 = (\varphi_1 \cup \varphi_2) \cdot (\tau_1 \sqcup \tau_2)$.

and *containment*, $\sqsubseteq$, is defined by:

- $\emptyset \sqsubseteq \tau$

- $\varphi \cdot \tau \sqsubseteq \varphi' \cdot \tau'$ if $\varphi \subseteq \varphi'$ and $\tau \sqsubseteq \tau'$.

The judgment $\Delta \vdash_p P : \tau$, defined in Figure 5, asserts that the pattern P is *well formed* and has pattern type τ, assuming the basis Δ, which assigns membrane types to element and sequence variables and pattern types to term variables. The judgement $\Delta \vdash_{gr} P_1 \mapsto P_2 : \mathtt{ok}$ in last rule defines well-formedness of global rules. It is easy to verify that the typing rules in Figures 4 and 5 enjoy weakening, i.e., if $\Delta \subseteq \Delta'$

$$\Delta, X : \tau \vdash_p X : \tau \quad \text{(TVAR)} \qquad \Delta \vdash_p SP : \emptyset \quad \text{(TSEQ)}$$

$$\frac{\Delta \vdash_p L_2 : \tau}{\Delta \vdash_p L_1 \mapsto L_2 : \mathsf{features}(L_1 \mapsto L_2) \sqcup \tau} \text{(TRLOC)}$$

$$\frac{\Delta \vdash_p L_2 : \tau \quad \Delta \vdash_s S_1 : \varphi_1 \quad \Delta \vdash_s S_2 : \varphi_2 \quad \varphi_1 \subseteq \varphi_2}{\Delta \vdash_p L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} : \{\mathtt{o}\} \cdot \tau} \text{(TRLOCOUT)}$$

$$\frac{\Delta \vdash_p L_2 : \varphi \cdot \tau \quad \Delta \vdash_s S_1 : \varphi_1 \quad \Delta \vdash_s S_2 : \varphi_2 \quad \varphi \cup \varphi_1 \subseteq \varphi_2}{\Delta \vdash_p L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} : \{\mathtt{i}\} \sqcup \tau} \text{(TRLOCIN)}$$

$$\frac{\Delta \vdash_p P : \tau \quad \Delta \vdash_p P' : \tau'}{\Delta \vdash_p P \mid P' : \tau \sqcup \tau'} \text{(TPAR)}$$

$$\frac{\Delta \vdash_s SP : \varphi \quad \Delta \vdash_p P : \varphi' \cdot \tau' \quad \varphi' \subseteq \varphi}{\Delta \vdash_p (SP)^{\circlearrowleft} \rfloor P : \tau'} \text{(TCOMP)}$$

$$\frac{\Delta \vdash_p P_1 : \tau_1 \quad \Delta \vdash_p P_2 : \tau_2 \quad \tau_2 \sqsubseteq \tau_1}{\Delta \vdash_{gr} P_1 \mapsto P_2 : \mathtt{ok}} \text{(TRGLOB)}$$

Figure 5: Typing Rules for Patterns and Global Rules

then $\Delta \vdash_s SP : \varphi$ implies $\Delta' \vdash_s SP : \varphi$, $\Delta \vdash_p P : \tau$ implies $\Delta' \vdash_p P : \tau$, and $\Delta \vdash_{gr} P_1 \mapsto P_2 : \mathtt{ok}$ implies $\Delta' \vdash_{gr} P_1 \mapsto P_2 : \mathtt{ok}$.

Rule (TVAR) asserts that a term variable is well typed when its pattern type is found in the basis. Rule (TSEQ) asserts that, since a sequence does not contain rules, its pattern type is empty. Rule (TRLOC) asserts that the type of a local rule $R = L_1 \mapsto L_2$ is the union of the set of features of the rule R, denoted by $\mathsf{features}(R)$, and the pattern type of its right-hand-side L_2. This is because once the rule is applied an instance of the pattern L_2 will substitute the instance of its left-hand-side L_1. Rule (TRLOCOUT) checks that the features of rules permitted by the membrane S_2 include the one permitted by S_1, so that if the compartment was well formed before applying $L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2}$, it will be well formed afterwards (when S_2 replaces S_1). Moreover, the pattern type of the rule is $\{\mathtt{o}\}$, concatenated with the pattern type of L_2, since L_2 is the pattern sent outside the compartment. Rule (TRLOCIN) checks rule $L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2}$. Since the pattern L_2 will get into a compartment with membrane S_1, the membrane S_2, that replaces S_1, must permit all the features of rules that were permitted before, and moreover, it permits the features of the rules in L_2. The type is $\{\mathtt{i}\}$ union the type of the patterns that are emitted by the out rules contained in L_2. Rule (TPAR) enforces the fact that the patterns in parallel are both well formed and the final pattern type is the union of the two pattern types. Rule (TCOMP) checks that a compartment contain only rules whose features are

permitted by its wrapping sequence. The pattern type of the compartment is the type of the rules that are emitted. Finally, rule (TRGLOB) says that the global rule $P_1 \mapsto P_2$ is well formed in case the pattern P_2 that will replace P_1 has less features, so that it is permitted by all the compartments in which P_1 is permitted.

As we can see from rule (TRLOC) the type system is independent from the specific set of features considered. Any syntactic characterisation of rules could be considered for a feature.

Based on the previous typing system we define a *typed semantics*, that preserves well-formedness of terms. Let an instantiation σ *agree* with a basis Δ (notation $\sigma \in \Sigma_\Delta$) if $x : \varphi \in \Delta$ implies $\vdash_s \sigma(x) : \varphi$, $\widetilde{x} : \varphi \in \Delta$ implies $\vdash_s \sigma(\widetilde{x}) : \varphi$, and $X : \tau \in \Delta$ implies $\vdash_p \sigma(X) : \tau$. This is sound since the judgments $\vdash_s$ use assumptions on element and sequence variables, while the the judgments $\vdash_p$ use assumptions on term variables.

Definition 3.1 (Typed Rule Application)**.** *Given a finite set of global rules $\mathscr{R}$, the* typed rule application $\rightarrow_\mathsf{T}$ *is the least relation closed with respect to $\equiv$ and satisfying the following rules:*

$$\frac{\mathscr{R}_\Delta = \{P_1 \mapsto P_2 \in \mathscr{R} \mid \Delta \vdash_{gr} P_1 \mapsto P_2 : \mathtt{ok}\} \quad P_1 \mapsto P_2 \in \mathscr{R}_\Delta \quad \sigma \in \Sigma_\Delta \quad P_1\sigma \not\equiv \varepsilon}{E[P_1\sigma] \rightarrow_\mathsf{T} E[\underline{P_2\sigma}]} \text{ (T-GRT)}$$

$$\frac{\sigma \in \Sigma_\Delta \quad L_1\sigma \not\equiv \varepsilon}{E[L_1 \mapsto L_2 \mid L_1\sigma] \rightarrow_\mathsf{T} E[L_1 \mapsto L_2 \mid \underline{L_2\sigma}]} \text{ (T-LR)}$$

$$\frac{\sigma \in \Sigma_\Delta \quad L_1\sigma \not\equiv \varepsilon}{E[(S_1\sigma)^\circlearrowleft \rfloor (L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid L_1\sigma \mid T)] \rightarrow_\mathsf{T} E[\underline{L_2\sigma} \mid \underline{(S_2\sigma)}^\circlearrowleft \rfloor (L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid T)]} \text{ (T-LR-OUT)}$$

$$\frac{\sigma \in \Sigma_\Delta \quad L_1\sigma \not\equiv \varepsilon}{E[L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid L_1\sigma \mid (S_1\sigma)^\circlearrowleft \rfloor T] \rightarrow_\mathsf{T} E[L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid \underline{(S_2\sigma)}^\circlearrowleft \rfloor (T \mid \underline{L_2\sigma})]} \text{ (T-LR-IN)}$$

Definition 3.2 (Typed Parallel Reduction)**.** *The reduction $\Longrightarrow_\mathsf{T}$ between term in $\mathscr{T}$ is defined by:*

$$\frac{T = T_0 \rightarrow_\mathsf{T} T_1 \rightarrow_\mathsf{T} \cdots \rightarrow_\mathsf{T} T_{n+1} \quad n \geq 0 \quad T' = \eta(T_{n+1})}{T \Longrightarrow_\mathsf{T} T'}$$

The property enforced by the type system is that well-typed terms reduce to well-typed terms: the proof is the content of the Appendix.

Theorem 3.3 (Subject Reduction)**.** *If $\vdash_p T : \tau$ and $T \Longrightarrow_\mathsf{T} T'$, then $\vdash_p T' : \tau'$ for some $\tau' \sqsubseteq \tau$.*

Example 3.4 (Mitochondria Running Example: Typing)**.** *Let the rules used in Example 2.7 be labelled by:*

- $R_g = \widetilde{x} \cdot g \cdot \widetilde{y} \mapsto (\widetilde{x} \cdot g \cdot \widetilde{y} \mid mRNA)$,
- $R_m = mRNA \mapsto protein$,
- $R_{o\downarrow} = protein^{\downarrow Tom} \mapsto protein^{\downarrow Tom}$,
- $R_{m\uparrow} = mRNA^{\uparrow nucleus} \mapsto mRNA^{\uparrow nucleus}$,
- $R_{i\downarrow} = protein^{\downarrow Tim} \mapsto R_a{}^{\downarrow Tim}$,
- $R_a = Mit_A \mapsto (Mit_A \mid ATP)$

- $R_{a\uparrow} = ATP^{\uparrow\widetilde{x}} \mapsto ATP^{\uparrow\widetilde{x}}$.

Let $\varphi_g = \mathsf{features}(R_g)$, $\varphi_m = \mathsf{features}(R_m)$, $\varphi_a = \mathsf{features}(R_a)$. The term representing our model can be typed if Λ contains appropriate membrane types for the elements which occur in the membranes, i.e.:

$$\{cell : \varphi_{cell}, nucleus : \varphi_{nucleus}, Tom : \varphi_{Tom}, Tim : \varphi_{Tim}\} \subseteq \Lambda$$

where $\{\mathsf{i}\} \cup \varphi_m \subseteq \varphi_{cell}$, $\{\mathsf{o}\} \cup \varphi_g \subseteq \varphi_{nucleus}$, $\{\mathsf{o},\mathsf{i}\} \subseteq \varphi_{Tom}$, and $\{\mathsf{o}\} \cup \varphi_a \in \varphi_{Tim}$. In this case the given parallel reduction is also a typed parallel reduction for this term.

We can type the `MITOCH` *with the following derivations:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\Delta \vdash_p Mit_A \mid ATP : \emptyset}{\Delta \vdash_p R_a : \varphi_a}\,(\textsc{TrLoc}) \qquad \Delta \vdash_s Tim : \varphi_{Tim} \quad \varphi_a \subseteq \varphi_{Tim}}{\Delta \vdash_p R_{i\downarrow} : \{\mathsf{i}\}}\,(\textsc{TrLocIn}) \qquad \Delta \vdash_p R_{a\uparrow} : \{\mathsf{o}\}}{\Delta \vdash_p R_{i\downarrow} \mid R_{a\uparrow} : \{\mathsf{i},\mathsf{o}\}}\,(\textsc{Tpar})}$$

$$\dfrac{\Delta \vdash_s Tom : \varphi_{Tom} \qquad \dfrac{\Delta \vdash_p \mathtt{INN\_MITOCH} : \emptyset \quad \Delta \vdash_p R_{i\downarrow} \mid R_{a\uparrow} : \{\mathsf{i},\mathsf{o}\}}{\Delta \vdash_p \mathtt{INN\_MITOCH} \mid R_{i\downarrow} \mid R_{a\uparrow} : \{\mathsf{i},\mathsf{o}\}}\,(\textsc{Tpar}) \qquad \{\mathsf{i},\mathsf{o}\} \subseteq \varphi_{Tom}}{\Delta \vdash_p (Tom)^{\circlearrowleft} \rfloor (\mathtt{INN\_MITOCH} \mid R_{i\downarrow} \mid R_{a\uparrow}) : \emptyset}\,(\textsc{Tcomp})$$

where we can type `INN_MITOCH` *with:*

$$\dfrac{\dfrac{\Delta \vdash_p Mit_A : \emptyset \qquad \dfrac{\Delta \vdash_p ATP : \emptyset \quad \Delta \vdash_s \widetilde{x} : \varphi_{Tom} \quad \varphi_{Tom} \subseteq \varphi_{Tom}}{\Delta \vdash_p R_{a\uparrow} : \{\mathsf{o}\} \cdot \emptyset}\,(\textsc{TrLocOut})}{\Delta \vdash_p Mit_A \mid R_{a\uparrow} : \{\mathsf{o}\}}\,(\textsc{Tpar}) \qquad \Delta \vdash_s Tim : \varphi_{Tim} \qquad \{\mathsf{o}\} \subseteq \varphi_{Tim}}{\Delta \vdash_p (Tim)^{\circlearrowleft} \rfloor (Mit_A \mid R_{a\uparrow}) : \emptyset}\,(\textsc{Tcomp})$$

4 Related Works and Conclusions

κ-*calculus* is a formalism proposed by Danos and Laneve [11] that idealises protein-protein interactions using graphs and graph-rewriting operations. A protein is a node with a fixed number of sites, that may be bound or free. Proteins may be assembled into complexes by connecting two-by-two bound sites of proteins, thus building connected graphs. Collections of proteins and complexes evolve by means of reactions, which may create or remove proteins and bounds between proteins: κ-calculus essentially deals with complexations and decomplexations, where complexation is a combination of substances into a new substance called complex, and the decomplexation is the reaction inverse to complexation, when a complex is dissociated into smaller parts. These rules contain variables and are pattern-based, therefore may be applied in different contexts. Even if κ-calculus does not deal with membranes, we have in common the use of variables and contexts for rule application. Moreover, both approaches emphasise the key rule of the surface components, in proteins (κ-calculus) or membranes (CLSLR), for biological modelling.

P-Systems [17] are a biologically inspired computational model. A P-System is formed by a membrane structure: each membrane may contain molecules, represented by symbols of an alphabet, other

membranes and rules. The rules contained into a membrane can be applied only to the symbols contained in the same membrane: these symbols can be modified or moved across membranes. The key feature of P-Systems is the maximal parallelism, i.e., in a single evolution step all symbols in all membranes evolve in parallel, and every applicable rule is applied as many times as possible. Locality and intrinsic parallelism of rules are also present in our approach, but in CLSLR the level of parallelism is not necessarily maximal, and moreover not only molecules but also rules can be created, deleted or moved across membranes. In both approaches the local rules cannot describe some possible biological behaviours such as fusion, deletion or creation of membranes. P-Systems are not so flexible in the description of new activities observed on membranes without extending the formalism to model such activities. In CLSLR this limit is overcome by global rules, that contain generic patterns.

In rewrite system models, the term (describing the systems under consideration) and the list of rules (describing the system's evolution) could be considered as separate (written on two different sheets of paper). In this work, we have presented a calculus with global (separate from the system) and local (dynamic and system intrinsic) rewrite rules. While global rules can, as usual, be applied anywhere in a given term, local rules can only be applied in the compartment on which they are defined. Local rules are equipped with dynamic features: they can be created, moved and erased.

As it happens for P-Systems, local rules are intrinsically parallel. Indeed, expressing rules that are local to well delimited compartments, and with the possibility to define systems with multiple, parallel, compartments, naturally leads to the definition of a parallel semantics.

As a future work, in the lines of [9, 12, 8], we plan to investigate how to adapt this model with a quantitative semantics, also studying the limits and constraints imposed by a parallel semantics.

References

[1] Bogdan Aman, Mariangiola Dezani-Ciancaglini & Angelo Troina (2009): *Type Disciplines for Analysing Biologically Relevant Properties*. In: *Membrane Computing and Biologically Inspired Process Calculi (MeCBIC'08). ENTCS* 227, Elsevier, pp. 97–111, doi:10.1016/j.entcs.2008.12.106.

[2] Roberto Barbuti, Mariangiola Dezani-Ciancaglini, Andrea Maggiolo-Schettini, Paolo Milazzo & Angelo Troina (2010): *A Formalism for the Description of Protein Interaction*. *Fundamenta Informaticae* 104(1-4), pp. 1–29, doi:10.3233/FI-2010-316.

[3] Roberto Barbuti, Andrea Maggiolo-Schettini & Paolo Milazzo (2006): *Extending the Calculus of Looping Sequences to Model Protein Interaction at the Domain Level*. In: *International Symposium on Bioinformatics Research and Applications (ISBRA'07). LNBI* 4463, Springer, pp. 638–649, doi:10.1007/978-3-540-72031-7_58.

[4] Roberto Barbuti, Andrea Maggiolo-Schettini, Paolo Milazzo & Angelo Troina (2006): *Bisimulation Congruences in the Calculus of Looping Sequences*. In: *International Colloquium on Theoretical Aspects of Computing (ICTAC'06). LNCS* 4281, Springer, pp. 93–107, doi:10.1007/11921240_7.

[5] Roberto Barbuti, Andrea Maggiolo-Schettini, Paolo Milazzo & Angelo Troina (2006): *A Calculus of Looping Sequences for Modelling Microbiological Systems*. *Fundamenta Informaticæ* 72(1–3), pp. 21–35, doi:10.3233/FI-2006-79.

[6] Roberto Barbuti, Andrea Maggiolo-Schettini, Paolo Milazzo & Angelo Troina (2007): *The Calculus of Looping Sequences for Modeling Biological Membranes*. In: *Workshop on Membrane Computing. LNCS* 4860, Springer, pp. 54–76, doi:10.1007/978-3-540-77312-2_4.

[7] Roberto Barbuti, Andrea Maggiolo-Schettini, Paolo Milazzo & Angelo Troina (2008): *Bisimulations in Calculi Modelling Membranes*. *Formal Aspects of Computing* 20(4-5), pp. 351–377, doi:10.1007/s00165-008-0071-x.

[8] Livio Bioglio (2011): *Enumerated type semantics for the calculus of looping sequences. RAIRO - Theoretical Informatics and Applications* 45(01), pp. 35–58, doi:10.1051/ita/2011010.

[9] Livio Bioglio, Mariangiola Dezani-Ciancaglini, Paola Giannini & Angelo Troina (2012): *Typed stochastic semantics for the calculus of looping sequences*. *Theoretical Computer Science* 431, pp. 165–180, doi:10.1016/j.tcs.2011.12.062.

[10] Nadia Busi (2007): *Using Well-structured Transition Systems to Decide Divergence for Catalytic P Systems*. *Theoretical Computer Science* 372, pp. 125–135, doi:10.1016/j.tcs.2006.11.021.

[11] Vincent Danos & Cosimo Laneve (2004): *Formal Molecular Biology*. *Theoretical Computer Science* 325, pp. 69–110, doi:10.1016/j.tcs.2004.03.065.

[12] Mariangiola Dezani-Ciancaglini, Paola Giannini & Angelo Troina (2009): *A Type System for a Stochastic CLS*. In: *Proc. of 4th Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC), Bologna, Italy*. 11, EPTCS, pp. 91–105, doi:10.4204/EPTCS.11.6.

[13] Mariangiola Dezani-Ciancaglini, Paola Giannini & Angelo Troina (2009): *A Type System for Required/Excluded Elements in CLS*. In: *Developments in Computational Models (DCM'09). EPTCS* 9, pp. 38–48, doi:10.4204/EPTCS.9.5.

[14] Pavel Dolezal, Vladimir Likic, Jan Tachezy & Trevor Lithgow (2006): *Evolution of the Molecular Machines for Protein Import into Mitochondria*. *Science* 313(5785), pp. 314–318, doi:10.1126/science.1127895.

[15] Paolo Milazzo (2007): *Qualitative and Quantitative Formal Modeling of Biological Systems*. Ph.D. thesis, University of Pisa.

[16] Nicolas Oury & Gordon Plotkin (2012): *Multi-Level Modelling via Stochastic Multi-Level Multiset Rewriting*. *Mathematical Structures in Computer Science*. To appear.

[17] Gheorghe Păun (2002): *Membrane Computing. An Introduction*. Springer.

A APPENDIX

Lemma A.1 (Inversion Lemma)**.** *1. If $\Delta \vdash_s \varepsilon : \varphi$, then $\varphi = \emptyset$.*

2. If $\Delta \vdash_s \chi : \varphi$, then $\chi : \varphi \in \Delta$.

3. If $\Delta \vdash_s a : \varphi$, then $a : \varphi \in \Lambda$.

4. If $\Delta \vdash_s SP \cdot SP' : \varphi$, then $\Delta \vdash_s SP : \varphi_1$, $\Delta \vdash_s SP' : \varphi_2$ and $\varphi = \varphi_1 \sqcup \varphi_2$.

5. If $\Delta \vdash_p X : \tau$, then $X : \tau \in \Delta$.

6. If $\Delta \vdash_p SP : \tau$, then $\tau = \emptyset$.

7. If $\Delta \vdash_p L_1 \mapsto L_2 : \tau$, then $\tau = \mathsf{features}(L_1 \mapsto L_2) \sqcup \tau'$ and $\Delta \vdash_p L_2 : \tau'$.

8. If $\Delta \vdash_p L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} : \tau$, then $\tau = \{\mathsf{o}\} \cdot \tau'$, $\Delta \vdash_p L_2 : \tau'$, $\Delta \vdash_s S_1 : \varphi_1$, $\Delta \vdash_s S_2 : \varphi_2$ and $\varphi_1 \subseteq \varphi_2$.

9. If $\Delta \vdash_p L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} : \tau$, then $\tau = \{\mathsf{i}\} \sqcup \tau'$, $\Delta \vdash_p L_2 : \varphi \cdot \tau'$, $\Delta \vdash_s S_1 : \varphi_1$, $\Delta \vdash_s S_2 : \varphi_2$ and $\varphi \cup \varphi_1 \subseteq \varphi_2$.

10. If $\Delta \vdash_p P \mid P' : \tau$, then $\tau = \tau_1 \sqcup \tau_2$, $\Delta \vdash_p P : \tau_1$ and $\Delta \vdash_p P' : \tau_2$.

11. If $\Delta \vdash_p (SP)^{\circlearrowleft} \rfloor P : \tau$, then $\Delta \vdash_s SP : \varphi$, $\Delta \vdash_p P : \varphi' \cdot \tau$ and $\varphi' \subseteq \varphi$.

12. If $\Delta \vdash_{gr} P_1 \mapsto P_2 : \mathtt{ok}$, then $\Delta \vdash_p P_1 : \tau_1$, $\Delta \vdash_p P_2 : \tau_2$ and $\tau_2 \sqsubseteq \tau_1$.

Proof. Immediate from the typing rules in Figures 4 and 5. □

Lemma A.2. *If $\Delta \vdash_p E[P] : \tau$ then*

1. $\Delta \vdash_p P : \tau_0$ for some τ_0, and

2. *if* P' *is such that* $\Delta \vdash_p P' : \tau'$ *with* $\tau' \sqsubseteq \tau_0$, *then* $\Delta \vdash_p E[P'] : \tau''$ *with* $\tau'' \sqsubseteq \tau$.

Proof. By induction on the definition of contexts.

- If $E = \Box$, then $E[P] = P$, and so $\Delta \vdash_p P : \tau$. Since in this case $E[P'] = P'$, and $\Delta \vdash_p P' : \tau'$ with $\tau' \sqsubseteq \tau$ by hypothesis, then $\Delta \vdash_p E[P'] : \tau'$.
- If $E = E' \mid T$, then $E[P] = E'[P] \mid T$. From Lemma A.1(10) we derive $\Delta \vdash_p E'[P] : \tau_1$ and $\Delta \vdash_p T : \tau_2$, with $\tau_1 \sqcup \tau_2 = \tau$. By induction hypothesis on $E'[P]$ we get $\Delta \vdash_p P : \tau_0$ and $\Delta \vdash_p E'[P'] : \tau_1'$ with $\tau_1' \sqsubseteq \tau_1$. Applying rule (TPAR) we conclude $\Delta \vdash_p E[P'] : \tau''$ with $\tau'' = \tau_1' \sqcup \tau_2$, and then $\tau'' \sqsubseteq \tau$.
- If $E = (S)^{\circlearrowleft} \rfloor E'$, then $E[P] = (S)^{\circlearrowleft} \rfloor E'[P]$. From Lemma A.1(11) we derive $\Delta \vdash_p S : \varphi_0$, and $\Delta \vdash_p E'[P] : \varphi \cdot \tau$, with $\varphi \subseteq \varphi_0$. By induction hypothesis on $E'[P]$ we get $\Delta \vdash_p P : \tau_0$, and $\Delta \vdash_p E'[P'] : \varphi' \cdot \tau'$ with $\varphi' \cdot \tau' \sqsubseteq \varphi \cdot \tau$. Applying rule (TCOMP) we conclude $\Delta \vdash_p E[P'] : \tau'$, with $\tau' \sqsubseteq \tau$.

□

Lemma A.3. *If* $\sigma \in \Sigma_\Delta$, *then* $\vdash_s SP\sigma : \varphi$ *if and only if* $\Delta \vdash_s SP : \varphi$.

Proof. ($\Leftarrow$) By induction on $\Delta \vdash_s SP : \varphi$. Consider the last applied rule.

- If the rule is (TSVAR), the proof follows from $\sigma \in \Sigma_\Delta$. For rules (TSEPS) and (TSELM), the fact that SP is a term implies that $SP\sigma = SP$, and, moreover, it is typable from the empty environment.
- Rule (TSSEQ). In this case $SP = SP_1 \cdot SP_2$, and from Lemma A.1(4) we derive $\Delta \vdash_s SP_1 : \varphi_1$, $\Delta \vdash_s SP_2 : \varphi_2$, and $\varphi = \varphi_1 \cup \varphi_2$. By induction hypotheses on SP_1 and SP_2 we get $\vdash_s SP_1\sigma : \varphi_1$ and $\vdash_s SP_2\sigma : \varphi_2$. Therefore, since $SP_1\sigma \cdot SP_2\sigma = (SP_1 \cdot SP_2)\sigma$, applying the rule (TSSEQ) we conclude $\vdash_s (SP_1 \cdot SP_2)\sigma : \varphi$.

($\Rightarrow$) By induction on SP.

- If $SP = \chi$, the proof follows from $\sigma \in \Sigma_\Delta$. If $SP = \varepsilon$ or $SP = a$ we use weakening.
- Let SP be $SP_1 \cdot SP_2$. Since $(SP_1 \cdot SP_2)\sigma = SP_1\sigma \cdot SP_2\sigma$, from Lemma A.1(4) we derive $\varphi = \varphi_1 \cup \varphi_2$, $\vdash_s SP_1\sigma : \varphi_1$, and $\vdash_s SP_2\sigma : \varphi_2$. By induction hypotheses we get $\Delta \vdash_s SP_1 : \varphi_1$, and $\Delta \vdash_s SP_2 : \varphi_2$. Applying rule (*TSseq*) we conclude $\Delta \vdash_s SP_1 \cdot SP_2 : \varphi$.

□

Lemma A.4. *If* $\sigma \in \Sigma_\Delta$, *then* $\vdash_p P\sigma : \tau$ *if and only if* $\Delta \vdash_p P : \tau$.

Proof. ($\Leftarrow$) By induction on $\Delta \vdash_p P : \tau$. Consider the last applied rule.

- If the rule is (TVAR), the proof follows from $\sigma \in \Sigma_\Delta$. For rules (TRLOC), (TRLOCOUT), (TRLOCIN) the fact that P is a rule implies that $P\sigma = P$ and, moreover, it is typable from the empty environment. For rule (TSEQ) if P is a sequence pattern then also $P\sigma$ is a sequence pattern, and then we can apply rule (TSEQ) with the empty environment.
- If the rule is (TPAR), then $P = P_1 \mid P_2$, and from Lemma A.1(10) we derive $\Delta \vdash_p P_1 : \tau_1$, $\Delta \vdash_p P_2 : \tau_2$, and $\tau = \tau_1 \sqcup \tau_2$. By induction hypotheses on P_1 and P_2 we get $\vdash_p P_1\sigma : \tau_1$, and $\vdash_p P_2\sigma : \tau_2$. Therefore, since $P_1\sigma \mid P_2\sigma = (P_1 \mid P_2)\sigma$, applying the rule (TPAR) we conclude $\vdash_p (P_1 \mid P_2)\sigma : \tau$.
- If the rule is (TCOMP), then the proof is similar using Lemmas A.1(11) and A.3 for the first premise.

($\Rightarrow$) By induction on P.

- If $P = X$, the proof follows from $\sigma \in \Sigma_\Delta$. If P is a sequence pattern, then also $P\sigma$ is a sequence pattern, and we can apply the rule (TSEQ). If P is a rule, then $P = P\sigma$.

- Let P be $P = P_1 \mid P_2$. Since $(P_1 \mid P_2)\sigma = P_1\sigma \cdot P_2\sigma$, and the fact that $\vdash_p (P_1 \mid P_2)\sigma : \tau$, from Lemma A.1(10) we derive $\vdash_p P_1\sigma : \tau_1$, $\vdash_p P_2\sigma : \tau_2$, and $\tau = \tau_1 \sqcup \tau_2$. By induction hypotheses on P_1 and P_2 we get $\Delta \vdash_p P_1 : \tau_1$ and $\Delta \vdash_p P_2 : \tau_2$. Applying rule (TPAR) we conclude $\Delta \vdash_p (P_1 \mid P_2) : \tau$.
- If $P = (SP)^{\circlearrowleft} \rfloor P'$ the proof is similar using Lemmas A.1(11) and A.3 for the first premise.

□

Proof of Theorem 3.3 (Subject Reduction)
By cases on the reduction rules.

Rule (TGR)

From Definition 3.1, $T = E[P_1\sigma]$, $T' = E[P_2\sigma]$, and $\sigma \in \Sigma_\Delta$. By hypothesis $\vdash_p T : \tau$ and $\Delta \vdash_{gr} P_1 \mapsto P_2 : \mathtt{ok}$. Therefore, Lemma A.2(1) implies $\vdash_p P_1\sigma : \tau_1$ for some τ_1, and from Lemma A.4 we derive $\Delta \vdash_p P_1 : \tau_1$. From $\Delta \vdash_{gr} P_1 \mapsto P_2 : \mathtt{ok}$, Lemma A.1(12) implies $\Delta \vdash_p P_2 : \tau_2$ with $\tau_2 \sqsubseteq \tau_1$. We can apply Lemma A.4 obtaining $\vdash_p P_2\sigma : \tau_2$. From Lemma A.2(2) we conclude $\Delta \vdash_p E[P_2\sigma] : \tau'$ for some $\tau' \sqsubseteq \tau$.

Rule (TLR)

From Definition 3.1, $T = E[L_1 \mapsto L_2 \mid L_1\sigma]$, $T' = E[L_1 \mapsto L_2 \mid L_2\sigma]$, and $\sigma \in \Sigma_\Delta$. By hypothesis $\vdash_p T : \tau$. Therefore, Lemma A.2(1) implies $\vdash_p L_1 \mapsto L_2 \mid L_1\sigma : \tau_0$ for some τ_0. Since $\sigma \in \Sigma_\Delta$, Lemma A.4 implies $\Delta \vdash_p L_1 \mapsto L_2 \mid L_1 : \tau_0$. From Lemma A.1(10) we derive $\Delta \vdash_p L_1 \mapsto L_2 : \tau_0'$, and $\Delta \vdash_p L_1 : \tau_1$ with $\tau_0' \sqcup \tau_1 = \tau_0$. By Lemma A.1(7) we derive $\Delta \vdash_p L_2 : \tau_2$ with $\tau_2 \sqsubseteq \tau_1$. Lemma A.4 implies $\vdash_p L_2\sigma : \tau_2$, then from (TPAR) we derive $\vdash_p L_1 \mapsto L_2 \mid L_2\sigma : \tau_3$ with $\tau_3 = \tau_0' \sqcup \tau_2$. Since $\tau_3 \sqsubseteq \tau_0$ we can apply Lemma A.2(2) obtaining $\vdash_p E[L_1 \mapsto L_2 \mid L_2\sigma] : \tau'$ with $\tau' \sqsubseteq \tau$.

Rule (LR-OUT)

From Definition 3.1, $T = E[(S_1\sigma)^{\circlearrowleft} \rfloor (L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid L_1\sigma \mid T_0)]$, $T' = E[L_2\sigma \mid (S_2\sigma)^{\circlearrowleft} \rfloor (L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid T_0)]$, and $\sigma \in \Sigma_\Delta$. By hypothesis $\vdash_p T : \tau$. Lemma A.2(1) implies $\vdash_p (S_1\sigma)^{\circlearrowleft} \rfloor (L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid L_1\sigma \mid T_0) : \tau_0$, and, since $\sigma \in \Sigma_\Delta$, Lemma A.4 implies $\Delta \vdash_p (S_1)^{\circlearrowleft} \rfloor (L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid L_1 \mid T_0) : \tau_0$. By Lemma A.1(11) we get $\Delta \vdash_s S_1 : \varphi_1$, and $\Delta \vdash_p L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid L_1 \mid T_0 : \varphi_0 \cdot \tau_0$ where $\varphi_0 \subseteq \varphi_1$. By Lemma A.1(10) we have $\Delta \vdash_p T_0 : \varphi \cdot \tau_1$, and $\Delta \vdash_p L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} : \varphi' \cdot \tau_2$ for some $\varphi \cdot \tau_1 \sqcup \varphi' \cdot \tau_2 \sqsubseteq \varphi_0 \cdot \tau_0$. Lemma A.1(8) implies $\varphi' = \{\mathtt{o}\}$, $\Delta \vdash_p L_2 : \tau_2$ and $\Delta \vdash_s S_2 : \varphi_2$ where $\varphi_1 \subseteq \varphi_2$. Since $\sigma \in \Sigma_\Delta$, Lemma A.4 implies $\vdash_p L_2\sigma : \tau_2$, $\vdash_s S_2\sigma : \varphi_2$, $\Delta \vdash_p L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} : \{\mathtt{o}\} \cdot \tau_2$, and $\vdash_p T_0 : \varphi \cdot \tau_1$. Using these premises, we apply rule (TPAR) deriving $\vdash_p L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid T_0 : \{\mathtt{o}\} \cdot \tau_2 \sqcup \varphi \cdot \tau_1$, and then $\vdash_p (S_2\sigma)^{\circlearrowleft} \rfloor (L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid T_0) : \tau_1 \sqcup \tau_2$ by rule (TCOMP). Finally $\vdash_p L_2\sigma \mid (S_2\sigma)^{\circlearrowleft} \rfloor (L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid T_0) : \tau_1 \sqcup \tau_2$ by rule (TPAR): since $\tau_1 \sqcup \tau_2 \sqsubseteq \tau_0$, we can apply the Lemma A.2(2), obtaining $\vdash_p E[L_2\sigma \mid (S_2\sigma)^{\circlearrowleft} \rfloor (L_1^{\uparrow S_1} \mapsto L_2^{\uparrow S_2} \mid T_0)] : \tau'$ with $\tau' \sqsubseteq \tau$.

Rule (LR-IN)

From Definition 3.1, $T = E[L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid L_1\sigma \mid (S_1\sigma)^{\circlearrowleft} \rfloor T_0]$, $T' = E[L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid (S_2\sigma)^{\circlearrowleft} \rfloor (T_0 \mid L_2\sigma)]$, and $\sigma \in \Sigma_\Delta$. By hypothesis $\vdash_p T : \tau$. Lemma A.2(1) implies $\vdash_p L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid L_1\sigma \mid (S_1\sigma)^{\circlearrowleft} \rfloor T_0 : \tau_0$, and, since $\sigma \in \Sigma_\Delta$, Lemma A.4 implies $\Delta \vdash_p L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid L_1 \mid (S_1)^{\circlearrowleft} \rfloor T_0 : \tau_0$. By Lemma A.1(10) we have $\Delta \vdash_p L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} : \tau_1$ and $\Delta \vdash_p (S_1)^{\circlearrowleft} \rfloor T_0 : \tau_2$ for some $\tau_1 \sqcup \tau_2 \sqsubseteq \tau_0$. Lemma A.1(9) implies $\tau_1 = \{\mathtt{i}\} \sqcup \tau_3$, $\Delta \vdash_s S_1 : \varphi_1$, $\Delta \vdash_s S_2 : \varphi_2$, and $\Delta \vdash_p L_2 : \varphi \cdot \tau_3$, where $\varphi \cup \varphi_1 \subseteq \varphi_2$. By Lemma A.1(11) $\Delta \vdash_p T_0 : \varphi' \cdot \tau_2$ for some $\varphi' \subseteq \varphi_1$. Since $\sigma \in \Sigma_\Delta$, Lemma A.4 implies $\vdash_s S_1\sigma : \varphi_1$, $\vdash_s S_2\sigma : \varphi_2$, $\vdash_p L_2\sigma : \varphi \cdot \tau_3$, $\vdash_p L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} : \{\mathtt{i}\} \sqcup \tau_3$, and $\vdash_p T_0 : \varphi' \cdot \tau_2$. Using these premises, we apply rule (TPAR) deriving $\vdash_p L_2\sigma \mid T_0 : \varphi \cdot \tau_3 \sqcup \varphi' \cdot \tau_2$, and then $\vdash_p (S_2\sigma)^{\circlearrowleft} \rfloor L_2\sigma \mid T_0 : \tau_3 \sqcup \tau_2$ by rule (TCOMP).

Finally $\vdash_p L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid (S_2\sigma)^{\circlearrowleft} \rfloor L_2\sigma \mid T_0 : \tau_1 \sqcup \tau_2$, because $\tau_1 = \{\mathrm{i}\} \sqcup \tau_3$, by rule (TPAR): since $\tau_1 \sqcup \tau_2 \sqsubseteq \tau_0$, we can apply the Lemma A.2(2) obtaining $\vdash_p E[L_1^{\downarrow S_1} \mapsto L_2^{\downarrow S_2} \mid (S_2\sigma)^{\circlearrowleft} \rfloor L_2\sigma \mid T_0] : \tau'$ for some $\tau' \sqsubseteq \tau$.