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 [11, 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.

E. Kashefi, J. Krivine, F. van Raamsdonk (Eds.)  
DCM 2011

© L. Bioglio, M. Dezani-Ciancaglini, P. Giannini and A. Troina
This work is licensed under the Creative Commons Attribution License.
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 mitochondrion internal 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 $\mathcal{E}$ of symbols ranged over by $a, b, c, \ldots$, a set of element variables $\mathcal{X}$ ranged over by $x, y, z, \ldots$, a set of sequence variables $\mathcal{V}$ ranged over by $x, y, z, \ldots$, and a set of term variables $\mathcal{Y}$ 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, $\mathcal{V} = \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$, and with $\chi$ a generic variable of $\mathcal{V}$. 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:

$$
P ::= \ (SP)^* \ | \ P \ | \ P \ | \ L \ 
SP ::= \ \varepsilon \ | \ a \ | \ x \ | \ SP \cdot SP \ | \ \bar{x} 
L ::= \ X \ | \ SP \ | \ L \ | \ L \ | \ R 
R ::= \ L \rightarrow L \ | \ L \rightarrow SP \rightarrow L \ | \ L \rightarrow SP \rightarrow L \ 
$$

where $a$ is a generic element of $\mathcal{E}$, and $X, \bar{x}$ and $x$ are generic elements of $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$, respectively. Sequence patterns $SP$ defines patterns for sequences of elements, $(SP)^*$ denotes a closed (looping) sequence which may contain other patterns through the $\ |$ operator, $|$ is used to denote the parallel composition (juxtaposition) of patterns, and $R$ denotes the syntax of local rules that may either exit $(L\rightarrow SP)$ or enter $(L \rightarrow SP)$ a closed sequence $SP$. We denote with $\mathcal{P}$ the infinite set of patterns.

A local rule $L_1 \rightarrow L_2$ is well formed if $L_1 \notin \varepsilon$ and $\text{Var}(L_2) \subseteq \text{Var}(L_1)$, where $\text{Var}(P)$ denotes set of variables appearing in $P$.

A local rule $L_1^{SP_1} \rightarrow L_2^{SP_2}$ or $L_1^{SP_1} \rightarrow L_2^{SP_2}$ is well formed if $L_1 \notin \varepsilon$, $\text{Var}(L_2) \subseteq \text{Var}(L_1)$ and $\text{Var}(SP_2) \subseteq \text{Var}(SP_1)$.

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

An instantiation is a partial function $\sigma : (\mathcal{V} \rightarrow \mathcal{T}) \cup (\mathcal{V} \rightarrow \mathcal{S}) \cup (\mathcal{X} \rightarrow \mathcal{E})$. Given $P \in \mathcal{P}$, with $P\sigma$ we denote the term obtained by replacing each occurrence of each variable $\chi \in \mathcal{V}$ 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 $\sigma$. With $\Sigma$ 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.
which produces terms in $T$.  

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 bookkeeping in the definition of rule application. Let $\eta : T \mapsto$ denote the set of terms in which some subterms can be underlined. The erasing mapping $\eta : T \mapsto$ 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 $T$.  

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 $\mathcal{R}$, the rule application $\rightarrow$ is the least relation closed with respect to $\equiv$ defined by:

\[
P_1 \mapsto P_2 \in \mathcal{R} \quad \sigma \in \Sigma \quad P_1 \sigma \neq \varepsilon \quad \frac{E[P_1 \sigma] \rightarrow E[P_2 \sigma]}{(GRT)}
\]

\[
\sigma \in \Sigma \quad L_1 \sigma \neq \varepsilon \quad \frac{E[L_1 \rightarrow L_2 \mid L_1 \sigma \mid T] \rightarrow E[L_1 \rightarrow L_2 \mid L_2 \sigma \mid T]}{(LR)}
\]

\[
\sigma \in \Sigma \quad L_1 \sigma \neq \varepsilon \quad \frac{E[(S_1 \sigma)^\ominus \mid (L_1^{S_1} \rightarrow L_2^{S_2} \mid L_1 \sigma \mid T)] \rightarrow E[L_2 \sigma \mid (S_2 \sigma)^\ominus \mid (L_1^{S_1} \rightarrow L_2^{S_2} \mid T)\]}{(LR-OUT)}
\]

\[
\sigma \in \Sigma \quad L_1 \sigma \neq \varepsilon \quad \frac{E[L_1^{S_1} \rightarrow L_2^{S_2} \mid L_1 \sigma \mid (S_1 \sigma)^\ominus \mid T] \rightarrow E[L_1^{S_1} \rightarrow L_2^{S_2} \mid (S_2 \sigma)^\ominus \mid (T \mid L_2 \sigma)]\} {(LR-IN)}
\]

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 $L^{S} \mapsto R^{S'}$ or $L^{S} \mapsto R^{S'}$.

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

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

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

\[
T \Longrightarrow T'
\]

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:

\[
C ::= \square \mid T \mid C \mid C \mid (\square)^\ominus \mid C \mid (S)^\ominus \mid C
\]

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 using the membrane, entering $\varepsilon$ 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
\[
C[T_i^*] \ldots [T_{i_1}] \mid ((T_{i_2})^\circ) \mid [T_{i_3}] \ldots [T_n^*] \rightarrow C[T_i^*] \ldots [T_{i_1}'] \mid ((T_{i_2}')^\circ) \mid [T_{i_3}'] \ldots [T_n']
\]
where \( T_j' \) can be either \( T_j \) (subterm of \( T \)) or \( T_j' \) (subterm of \( T' \)).

Proof. If \( T \rightarrow T' \), then for some \( U_0, \ldots, U_{m+1} \) we get \( T = U_0 \rightarrow \cdots \rightarrow U_{m+1} \) 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 \( U_h = C[T_i^*] \ldots [T_j] \ldots [T_n^*] \) and \( U_{h+1} = C[T_i^*] \ldots [T_j'] \ldots [T_n^*] \)

• or there are \( i_1, i_2, i_3 \) such that \( i \in \{i_1, i_2, i_3\} \) and
\[
U_h = C[T_i^*] \ldots [T_{i_1}] \mid ((T_{i_2})^\circ) \mid [T_{i_3}] \ldots [T_n^*] \) and \( U_{h+1} = C[T_i^*] \ldots [T_{i_1}'] \mid ((T_{i_2}')^\circ) \mid [T_{i_3}'] \ldots [T_n'] \)

where “*” can be either “” or “’” and all terms with ‘ are either underlined or \( \varepsilon \).

If the last applied rule is
\[
\sigma \in \Sigma \quad L_1 \sigma \neq \varepsilon
\]
\[
E[L_1 \mapsto L_2 \mid L_1 \sigma \mid V] \rightarrow E[L_1 \mapsto L_2 \mid L_2 \sigma \mid V]
\]
then \( U_h = E[L_1 \mapsto L_2 \mid L_1 \sigma \mid V] \) and \( U_{h+1} = E[L_1 \mapsto L_2 \mid L_2 \sigma \mid V] \). By induction \( U_h = C[T_i^*] \ldots [T_n^*] \).

Since \( L_1 \sigma \) is a subterm of \( T \) and \( L_2 \sigma \) is a subterm of \( U_{m+1} \) there must be an index \( i \) such that \( T_i = L_1 \sigma \) and \( T_i' = L_2 \sigma \).

If the last applied rule is
\[
\sigma \in \Sigma \quad L \sigma \neq \varepsilon \quad L_1 \sigma \in \mathcal{T} \quad S_1 \sigma \in \mathcal{T}
\]
\[
E[L_1^{S_1} \mapsto L_2^{S_2} \mid L_1 \sigma \mid (S_1 \sigma)^\circ \mid V] \rightarrow E[L_1^{S_1} \mapsto L_2^{S_2} \mid (S_2 \sigma)^\circ \mid V \mid L_2 \sigma]
\]
then \( U_h = E[L_1^{S_1} \mapsto L_2^{S_2} \mid L_1 \sigma \mid (S_1 \sigma)^\circ \mid V] \) and \( U_{h+1} = E[L_1^{S_1} \mapsto L_2^{S_2} \mid (S_2 \sigma)^\circ \mid V \mid L_2 \sigma] \). By induction \( U_h = C[T_i^*] \ldots [T_n^*] \). Notice that \( L_1 \sigma, S_1 \sigma \) are subterms of \( T \) and \( L_2 \sigma, S_2 \sigma \) are subterm of \( U_{m+1} \). Moreover \( L_1 \sigma, S_1 \sigma \) and \( \varepsilon \) in \( T \) are replaced by \( \varepsilon, S_2 \sigma \) and \( L_2 \sigma \) in \( U_{m+1} \). Therefore there must be indexes \( i_1, i_2, i_3 \) such that \( T_{i_1} = L_1 \sigma, T_{i_1}' = \varepsilon, T_{i_2} = S_1 \sigma, T_{i_2}' = S_2 \sigma, T_{i_3} = \varepsilon, T_{i_3}' = L_2 \sigma \).

\[\blacksquare\]

**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:

\[
\text{CELL} = (\text{cell})^\circ \mid (\text{NUCLEUS} \mid \text{MITOCH} \ldots \mid \text{MITOCH})
\]
\[
mRNA \rightarrow \text{protein} \mid \text{protein}^{\text{Tom}} \rightarrow \text{protein}^{\text{Tom}}
\]

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:

\[
\text{NUCLEUS} = (\text{nucleus})^\circ \mid (\text{DNA} \mid \tilde{x} \cdot g \cdot \tilde{y} \rightarrow (\tilde{x} \cdot g \cdot \tilde{y}) \mid \text{mRNA} \mid \text{mRNA}^{\text{nucleus}} \rightarrow \text{mRNA}^{\text{nucleus}})
\]
\[
\ldots \implies^+ (\text{cell})^\wedge \{(\text{nucleus})^\wedge (\text{mRNA} \mid \ldots \mid \text{mRNA} \mid \ldots) \mid \ldots\)
\implies^+ (\text{cell})^\wedge (\text{mRNA} \mid \ldots \mid \text{mRNA} \mid \ldots)
\implies (\text{cell})^\wedge (\text{protein} \mid \ldots \mid \text{protein} \mid \ldots)
\implies (\text{cell})^\wedge ((\text{Tom})^\wedge (\text{protein} \mid \ldots) \mid \ldots \mid (\text{Tom})^\wedge (\text{protein} \mid \ldots) \mid \ldots)
\implies (\text{cell})^\wedge ((\text{Tom})^\wedge ((\text{Tim})^\wedge (\text{Mit}_A \mapsto (\text{Mit}_A \mid \text{ATP}) \mid \ldots) \mid \ldots) \mid \ldots)
\implies (\text{cell})^\wedge ((\text{Tom})^\wedge ((\text{Tim})^\wedge (\text{ATP} \mid \ldots) \mid \ldots) \mid \ldots)
\implies (\text{cell})^\wedge (\text{ATP} \mid \ldots \mid \text{ATP} \mid (\text{Tom})^\wedge (\text{ATP} \mid \ldots) \mid \ldots) \mid \ldots)
\implies (\text{cell})^\wedge (\text{ATP} \mid (\text{Tim})^\wedge (\text{ATP} \mid \ldots) \mid \ldots) \mid \ldots)
\]

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 (\text{INN} \text{MITOCH}) and a couple of rules:

\[
\text{MITOCH} = (\text{Tom})^\wedge ((\text{INN} \text{MITOCH}) \mid \text{protein} \xrightarrow{\text{Tim}} (\text{Mit}_A \mapsto (\text{Mit}_A \mid \text{ATP}) \mid \text{ATP}) \xrightarrow{\text{Tim}})
\]

where we denote with the element Mit\text{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\text{A} \mapsto (Mit\text{A} \mid \text{ATP}) which will lead the production of ATP (step (6) of our example).

Finally, in \text{INN} \text{MITOCH} we point out the Tim complex:

\[
\text{INN} \text{MITOCH} = (\text{Tim})^\wedge ((\text{Mit}_A \mid \text{ATP}) \xrightarrow{\text{Tim}} \text{ATP})
\]

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 $\text{Vars}(L_1) \supset \text{Vars}(L_2)$ (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 $L_1 \uparrow S_1 \rightarrow L_2 \uparrow S_2$ is that it is an out rule (denoted by $o$);
• the feature of rule $L_1 \downarrow S_1 \rightarrow L_2 \downarrow S_2$ 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 $\phi$ of $\{d, r, s, e, o, i\}$ to every element in $E$. This is called a membrane type. We use $\Lambda$ to denote a classification of elements. The type assignment in Figure 4, where a basis $\Delta$ assigns membrane types to element and sequence variables, defines the type of a sequence as the union of the membrane types of its elements.

![Figure 4: Typing Rules for Membranes](image)

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 $\tau$, is a sequence of membrane types, i.e., $\tau \in \{\phi\}^*$. 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 $\phi \cdot \tau$, then $\phi$ is the union of the features of the rules $R_i$ $(1 \leq i \leq n)$, and $\tau$ is the type of the parallel composition of rules in $L'$ for those $L'$ such that $R_i = L_i \uparrow SP \rightarrow L_i \uparrow SP'$ 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:

- $\emptyset \sqcup \tau = \tau \sqcup \emptyset = \tau$.
- $\phi_1 \cdot \tau_1 \sqcup \phi_2 \cdot \tau_2 = (\phi_1 \sqcup \phi_2) \cdot (\tau_1 \sqcup \tau_2)$.

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

- $\emptyset \sqsubseteq \tau$
• $\phi \cdot \tau \sqsubseteq \phi' \cdot \tau'$ if $\phi \sqsubseteq \phi'$ and $\tau \sqsubseteq \tau'$.

The judgment $\Delta \vdash P : \tau$, defined in Figure 5, asserts that the pattern $P$ is well formed and has pattern type $\tau$, assuming the basis $\Delta$, which assigns membrane types to element and sequence variables and pattern types to term variables. The judgement $\Delta \vdash gr P_1 \mapsto P_2 : \text{ok}$ 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 $\Delta \sqsubseteq \Delta'$ then $\Delta \vdash SP : \varphi$ implies $\Delta' \vdash SP : \varphi$, $\Delta \vdash P : \tau$ implies $\Delta' \vdash P : \tau$, and $\Delta \vdash gr P_1 \mapsto P_2 : \text{ok}$ implies $\Delta' \vdash gr P_1 \mapsto P_2 : \text{ok}$.

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 $\text{features}(R)$, 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 $L_1^{S_1} \mapsto L_2^{S_2}$, it will be well formed afterwards (when $S_2$ replaces $S_1$). Moreover, the pattern type of the rule is $\{o\}$, concatenated with the pattern type of $L_2$, since $L_2$ is the pattern sent outside the compartment. Rule (TRLOCIN) checks rule $L_1^{S_1} \mapsto L_2^{S_2}$. 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 $\{i\}$ 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 \( T_{\text{TRGLOB}} \) 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 \( T_{\text{TRLOC}} \) 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 \( \sigma \) agree with a basis \( \Delta \) (notation \( \sigma \in \Sigma_{\Delta} \)) if \( x : \phi \in \Delta \) implies \( \vdash_x \sigma(x) : \phi \), \( \vec{x} : \phi \in \Delta \) implies \( \vdash_\vec{x} \sigma(\vec{x}) : \phi \), and \( X : \tau \in \Delta \) implies \( \vdash_p \sigma(X) : \tau \). This is sound since the judgments \( \vdash_x \), 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 \( \mathcal{R} \), the typed rule application \( \rightarrow_\tau \) is the least relation closed with respect to \( \equiv \) and satisfying the following rules:*

\[
\begin{align*}
\mathcal{R}_\Delta &= \{ P_1 \mapsto P_2 \in \mathcal{R} \mid \Delta \vdash_{gr} P_1 \mapsto P_2 : \text{ok} \} \\
\frac{P_1 \mapsto P_2 \in \mathcal{R}_\Delta \quad \sigma \in \Sigma_{\Delta} \quad P_1 \sigma \neq \varepsilon}{E[P_1 \sigma] \rightarrow_\tau E[P_2 \sigma]} \quad \text{(T-GRT)} \\
\frac{\sigma \in \Sigma_{\Delta} \quad L_1 \sigma \neq \varepsilon}{E[L_1 \mapsto L_2 \mid L_1 \sigma] \rightarrow_\tau E[L_1 \mapsto L_2 \mid L_2 \sigma]} \quad \text{(T-LR)} \\
\frac{\sigma \in \Sigma_{\Delta} \quad L_1 \sigma \neq \varepsilon \quad E[(S_1 \sigma) \circ] \mid (L_1^{S_1} \mapsto L_2^{S_2} \mid L_1 \sigma \mid T)] \rightarrow_\tau E[L_2 \sigma \mid (S_2 \sigma) \circ] \mid (L_1^{S_1} \mapsto L_2^{S_2} \mid T)]}{\text{T-LR-OUT}} \\
\frac{\sigma \in \Sigma_{\Delta} \quad L_1 \sigma \neq \varepsilon \quad E[L_1^{S_1} \mapsto L_2^{S_2} \mid L_1 \sigma \mid (S_1 \sigma) \circ] \mid T] \rightarrow_\tau E[L_1^{S_1} \mapsto L_2^{S_2} \mid (S_2 \sigma) \circ] \mid (T \mid L_2 \sigma)]}{\text{T-LR-IN}}
\end{align*}
\]

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

\[
T = T_0 \rightarrow_\tau T_1 \rightarrow_\tau \cdots \rightarrow_\tau T_{n+1} \quad n \geq 0 \quad T' = \eta(T_{n+1})
\]

\[
T \Longrightarrow_\tau T'
\]

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 \( T \Longrightarrow_\tau T' \), then \( \vdash_p T' : \tau' \) for some \( \tau' \subseteq \tau \).*

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

- \( R_g = \vec{x} \cdot g \cdot \vec{y} \rightarrow (\vec{x} \cdot g \cdot \vec{y} \mid \text{mRNA}) \),
- \( R_m = \text{mRNA} \rightarrow \text{protein} \),
- \( R_{\text{a}T} = \text{protein}^{\text{Tom}} \rightarrow \text{protein}^{\text{Tom}} \),
- \( R_{\text{m}T} = \text{mRNA}^{\text{nucleus}} \rightarrow \text{mRNA}^{\text{nucleus}} \),
- \( R_{\text{i}T} = \text{protein}^{\text{Tim}} \rightarrow R_{\text{a}T}^{\text{Tim}} \),
- \( R_{\text{a}} = \text{Mit}_{\text{A}} \rightarrow (\text{Mit}_{\text{A}} \mid \text{ATP}) \)
\[ R_{at} = ATP^{i-x} \rightarrow ATP^{i-x}. \]

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

\[
\{ \text{cell} : \varphi_{\text{cell}}, \text{nucleus} : \varphi_{\text{nucleus}}, \text{Tom} : \varphi_{\text{Tom}}, \text{Tim} : \varphi_{\text{Tim}} \} \subseteq \Delta
\]

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

We can type the MITOC with the following derivations:

\[
\begin{align*}
\Delta \vdash p \text{Mit}_A \mid ATP : \emptyset & \quad (\text{TRLoc}) & \Delta \vdash s \text{Tim} : \varphi_{\text{Tim}} & \quad \varphi_a \subseteq \varphi_{\text{Tim}} & \Delta \vdash p \text{Mit}_A : \{ o \} & \quad (\text{TPar}) \\
\end{align*}
\]

\[
\begin{align*}
\Delta \vdash p \text{R}_{ij} : \{ i \} & \quad (\text{TRLocIn}) & \Delta \vdash s \text{R}_{at} : \{ i, o \} & \quad (\text{TPar}) \\
\end{align*}
\]

\[
\begin{align*}
\Delta \vdash s \text{Tom} : \varphi_{\text{Tom}} & \quad \Delta \vdash p \text{INN-MITOCH : } \emptyset & \quad \Delta \vdash p \text{INN-MITOCH} \mid R_{ij} \mid R_{at} : \{ i, o \} & \quad (\text{TPar}) & \{ i, o \} \subseteq \varphi_{\text{Tom}} & \quad (\text{TComp}) \\
\end{align*}
\]

where we can type INN-MITOCH with:

\[
\begin{align*}
\Delta \vdash p \text{Mit}_A : \emptyset & \quad \Delta \vdash s \text{Tim : } \varphi_{\text{Tim}} & \quad \varphi_{\text{Tom}} \subseteq \varphi_{\text{Tim}} & \quad (\text{TRLocOut}) \\
\Delta \vdash p \text{Mit}_A : \emptyset & \quad \Delta \vdash p \text{R}_{at} : \{ o \} & \quad (\text{TPar}) \\
\end{align*}
\]

\[
\begin{align*}
\Delta \vdash s \text{Tim} : \varphi_{\text{Tim}} & \quad \varphi_{\text{Tom}} \subseteq \varphi_{\text{Tim}} & \quad (\text{TComp}) \\
\Delta \vdash p \text{Mit}_A : \emptyset & \quad \Delta \vdash p \text{R}_{at} : \{ o \} & \quad (\text{TPar}) \\
\end{align*}
\]

\[
\begin{align*}
\Delta \vdash p \text{Tim}^\circ \mid (\text{Mit}_A \mid R_{at} : \emptyset & \quad (\text{TComp}) \\
\end{align*}
\]

\[
\]
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_5

[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
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 $a : \varphi \in \Delta$.

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

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

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

7. If $\Delta \vdash_p L_1 \rightarrow L_2 : \tau$, then $\tau = \text{features}(L_1 \rightarrow L_2) \cup \tau'$ and $\Delta \vdash_p L_2 : \tau'$.

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

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

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

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

12. If $\Delta \vdash_{gr} P_1 \rightarrow P_2 : \text{ok}$, then $\Delta \vdash_p P_1 : \tau_1$, $\Delta \vdash_p P_2 : \tau_2$ and $\tau_2 \subseteq \tau_1$.

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 $\tau_0$ and
2. If $P'$ is such that $\Delta \vdash_p P' : \tau'$ with $\tau' \subseteq \tau_0$, then $\Delta \vdash_p E[P'] : \tau''$ with $\tau'' \subseteq \tau$.

**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 $\tau' \subseteq \tau$ 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 $\Delta \vdash_p E'[P'] : \tau'_1$ with $\tau'_1 \subseteq \tau_1$. Applying rule (Tpar) we conclude $\Delta \vdash_p E'[P'] : \tau''$ with $\tau'' = \tau'_1 \sqcup \tau_2$, and then $\tau'' \subseteq \tau$.

- If $E = (S)^\ominus \mid E'$, then $E[P] = (S)^\ominus \mid E'[P]$. From Lemma A.1[11] we derive $\Delta \vdash_p S : \phi_0$, and $\Delta \vdash_p E'[P] : \phi \cdot \tau$, with $\phi \subseteq \phi_0$. By induction hypothesis on $E'[P]$ we get $\Delta \vdash_p E'[P'] : \phi' \cdot \tau'$ with $\phi' \cdot \tau' \subseteq \phi \cdot \tau$. Applying rule (Tcomp) we conclude $\Delta \vdash_p E'[P'] : \tau'$, with $\tau' \subseteq \tau$.

\[\square\]

**Lemma A.3.** If $\sigma \in \Sigma_\Delta$, then $\vdash_\tau SP\sigma : \phi$ if and only if $\Delta \vdash_\tau SP : \phi$.

**Proof.** ($\Rightarrow$) By induction on $\vdash_\tau SP : \phi$. Consider the last applied rule.

- If the rule is (Tsva), 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 (Tseq). In this case $SP = SP_1 \cdot SP_2$, and from Lemma A.1[4] we derive $\vdash_\tau SP_1 : \phi_1$, $\vdash_\tau SP_2 : \phi_2$, and $\phi = \phi_1 \cup \phi_2$. By induction hypotheses on $SP_1$ and $SP_2$ we get $\vdash_\tau SP_1 \sigma : \phi_1$ and $\vdash_\tau SP_2 \sigma : \phi_2$. Therefore, since $SP_1 \sigma \cdot SP_2 \sigma = (SP_1 \cdot SP_2)\sigma$, applying the rule (Tseq) we conclude $\vdash_\tau (SP_1 \cdot SP_2)\sigma : \phi$.

($\Rightarrow$) 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 $\phi = \phi_1 \cup \phi_2$, $\vdash_\tau SP_1 \sigma : \phi_1$, and $\vdash_\tau SP_2 \sigma : \phi_2$. By induction hypotheses we get $\vdash_\tau SP_1 : \phi_1$, and $\vdash_\tau SP_2 : \phi_2$. Applying rule (Tseq) we conclude $\vdash_\tau SP_1 \cdot SP_2 : \phi$.

\[\square\]

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

**Proof.** ($\Rightarrow$) By induction on $\Delta \vdash_\tau P : \tau$. 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 $\Delta \vdash_\tau P_1 : \tau_1$, $\Delta \vdash_\tau P_2 : \tau_2$, and $\tau = \tau_1 \sqcup \tau_2$. By induction hypotheses on $P_1$ and $P_2$ we get $\vdash_\tau P_1 \sigma : \tau_1$, and $\vdash_\tau P_2 \sigma : \tau_2$. Therefore, since $P_1 \sigma \mid P_2 \sigma = (P_1 \mid P_2)\sigma$, applying the rule (Tpar) we conclude $\vdash_\tau (P_1 \mid P_2)\sigma : \tau$.

- 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 \( P = P_1 \cup P_2 \). Since \( (P_1 \cup P_2)\sigma = P_1\sigma \cdot P_2\sigma \), and the fact that \( \vdash_p (P_1 \cup P_2)\sigma : \tau \), from Lemma A.1(10) we derive \( \vdash P_1\sigma : \tau_1 \), \( \vdash P_2\sigma : \tau_2 \), 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 \( \Delta \vdash_p (P_1 \cup P_2) : \tau \).

If \( P = (SP)^\circ \cup P' \) the proof is similar using Lemmas A.1(11) and A.2 for the first premise.

\[ \square \]

**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 \( \Delta \vdash_{gr} P_1 \to P_2 : \text{ok} \). Therefore, Lemma A.2(11) implies \( \vdash_p P_1\sigma : \tau_1 \) for some \( \tau_1 \), and from Lemma A.4 we derive \( \Delta \vdash_p P_1 : \tau_1 \). From \( \Delta \vdash_{gr} P_1 \to P_2 : \text{ok} \), Lemma A.1(12) implies \( \Delta \vdash_p P_2 : \tau_2 \) with \( \tau_2 \subseteq \tau_1 \). We can apply Lemma A.4 obtaining \( \vdash P_2\sigma : \tau_2 \). From Lemma A.2(12) we conclude \( \Delta \vdash_p E[P_2\sigma] : \tau' \) for some \( \tau' \subseteq \tau \).

**Rule (TLR)**

From Definition 3.1 \( T = E[L_1 \mapsto L_2 | L_1\sigma] \), \( T' = E[L_1 \mapsto L_2 | L_2\sigma] \), and \( \sigma \in \Sigma_\Delta \). By hypothesis \( \vdash_p T : \tau \). Lemma A.2(11) implies \( \vdash_p L_1\sigma : \tau_0 \) for some \( \tau_0 \). Since \( \sigma \in \Sigma_\Delta \), Lemma A.4 implies \( \Delta \vdash_p L_1 \mapsto L_2 | L_1\sigma : \tau_0 \). From Lemma A.1(10) we derive \( \Delta \vdash_{\tau_0} L_1 \mapsto L_2 : \tau_0 \), and \( \Delta \vdash_p L_1 : \tau_1 \) with \( \tau_0 \sqcup \tau_1 = \tau_0 \). By Lemma A.1(11) we derive \( \Delta \vdash_p L_2 : \tau_2 \) with \( \tau_2 \subseteq \tau_1 \). Lemma A.4 implies \( \vdash_p L_2\sigma : \tau_2 \), then from (TPar) we derive \( \vdash_p L_1 \mapsto L_2 | L_2\sigma : \tau_3 \) with \( \tau_3 = \tau_0 \sqcup \tau_2 \). Since \( \tau_3 \subseteq \tau_0 \) we can apply Lemma A.2(12) obtaining \( \vdash_p E[L_1 \mapsto L_2 | L_2\sigma] : \tau' \) with \( \tau' \subseteq \tau \).

**Rule (LR-out)**

From Definition 3.1 \( T = E[(S_1\sigma)^\circ \cup (L_1^{S_1} \mapsto L_2^{S_1} | L_1\sigma \sqcup T_0)^\circ] \), \( T' = E[L_2\sigma | (S_2\sigma)^\circ \cup (L_1^{S_1} \mapsto L_2^{S_1} | L_2\sigma \sqcup T_0)^\circ] \), and \( \sigma \in \Sigma_\Delta \). By hypothesis \( \vdash_p T : \tau \). Lemma A.2(11) implies \( \vdash_p (S_1\sigma)^\circ \cup (L_1^{S_1} \mapsto L_2^{S_1} | L_1\sigma \sqcup T_0)^\circ : \tau_0 \). By Lemma A.4(11) we get \( \Delta \vdash_{\tau_0} S_1 : \varphi_1 \), and \( \Delta \vdash_p L_1^{S_1} \mapsto L_2^{S_1} | L_1 \mapsto \varphi_0 \cdot \tau_0 \) where \( \varphi_0 \sqsubseteq \varphi_1 \). By Lemma A.1(10) we have \( \Delta \vdash_p T_0 : \varphi \cdot \tau_1 \), and \( \Delta \vdash_p L_1^{S_1} \mapsto L_2^{S_1} : \varphi' \cdot \tau_2 \) for some \( \varphi \cdot \tau_1 \sqcup \varphi' \cdot \tau_2 \sqsubseteq \varphi_0 \cdot \tau_0 \). Lemma A.1(11) implies \( \varphi' = \{o\}, \Delta \vdash_p L_2 : \tau_2 \) and \( \Delta \vdash_{\tau_0} S_2 : \varphi_2 \) where \( \varphi_1 \sqsubseteq \varphi_2 \). Since \( \sigma \in \Sigma_\Delta \), Lemma A.4 implies \( \vdash_p L_2\sigma : \tau_2 \), then \( \vdash_p S_2\sigma : \varphi_2 \), and \( \Delta \vdash_p L_1^{S_1} \mapsto L_2^{S_1} : \{o\} \cdot \tau_2 \), and \( \vdash_p T_0 : \varphi \cdot \tau_1 \). Using these premises, we apply rule (TPar) deriving \( \vdash_p L_1^{S_1} \mapsto L_2^{S_1} : \{o\} \cdot \tau_2 \sqcup \varphi \cdot \tau_1 \), and then \( \vdash_p (S_2\sigma)^\circ \cup (L_1^{S_1} \mapsto L_2^{S_1} | T_0)^\circ : \tau_1 \sqcup \tau_2 \) by rule (TComp). Finally \( \vdash_p L_2\sigma | (S_2\sigma)^\circ \cup (L_1^{S_1} \mapsto L_2^{S_1} | T_0)^\circ : \tau_1 \sqcup \tau_2 \) by rule (TPar): since \( \tau_1 \sqcup \tau_2 \sqsubseteq \tau_0 \), we can apply the Lemma A.2(12) obtaining \( \vdash_p E[L_2\sigma | (S_2\sigma)^\circ \cup (L_1^{S_1} \mapsto L_2^{S_1} | T_0)^\circ] : \tau' \) with \( \tau' \subseteq \tau \).

**Rule (LR-in)**

From Definition 3.1 \( T = E[L_1^{S_1} \mapsto L_2^{S_1} | L_1\sigma \sqcup (S_1\sigma)^\circ | T_0], T' = E[L_1^{S_1} \mapsto L_2^{S_1} | (S_2\sigma)^\circ | T_0 \sqcup L_2\sigma] \), and \( \sigma \in \Sigma_\Delta \). By hypothesis \( \vdash_p T : \tau \). Lemma A.2(11) implies \( \vdash_p L_1^{S_1} \mapsto L_2^{S_1} | L_1\sigma | (S_1\sigma)^\circ : \tau_0 \). And, since \( \sigma \in \Sigma_\Delta \), Lemma A.4 implies \( \Delta \vdash_p L_1^{S_1} \mapsto L_2^{S_1} | L_1 | (S_1\sigma)^\circ : \tau_0 \). By Lemma A.1(10) we have \( \Delta \vdash_p L_1^{S_1} \mapsto L_2^{S_1} : \tau_1 \) and \( \Delta \vdash_p (S_1\sigma)^\circ : \tau_0 \) for some \( \tau_1 \sqcup \tau_2 \subseteq \tau_0 \). Lemma A.1(9) implies \( \tau_1 = \{i\} \sqcup \tau_3 \), \( \Delta \vdash_p S_1 : \varphi_1 \), \( \Delta \vdash_{\tau_3} S_2 : \varphi_2 \), and \( \Delta \vdash_p L_2 : \varphi \cdot \tau_3 \), where \( \varphi \sqsubseteq \varphi_1 \). 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 \( \vdash_{\tau_0} S_1\sigma : \varphi_1 \), \( \vdash_{\tau_3} S_2\sigma : \varphi_2 \), \( \vdash_p L_2\sigma : \varphi \cdot \tau_3 \) \( \vdash_p L_1^{S_1} \mapsto L_2^{S_1} : \{i\} \sqcup \tau_3 \), and \( \vdash_p T_0 : \varphi \cdot \tau_2 \). Using these premises, we apply rule (TPar) deriving \( \vdash_p L_2\sigma \sqcup T_0 : \varphi \cdot \tau_3 \sqcup \varphi' \cdot \tau_2 \), and then \( \vdash_p (S_2\sigma)^\circ \cup L_2\sigma | T_0 : \tau_3 \sqcup \tau_2 \) by rule (TComp).
Finally $\vdash_{\rho} L_1^{S_1} \rightarrow L_2^{S_2} | (S_2 \sigma)^\Diamond | L_2 \sigma | T_0 : \tau_1 \sqcup \tau_2$, because $\tau_1 = \{i\} \sqcup \tau_3$, by rule (T\textsc{par}): since $\tau_1 \sqcup \tau_2 \subseteq \tau_0$, we can apply the Lemma A22 obtaining $\vdash_{\rho} E[L_1^{S_1} \rightarrow L_2^{S_2} | (S_2 \sigma)^\Diamond | L_2 \sigma | T_0] : \tau'$ for some $\tau' \subseteq \tau$. 