

An Intermediate Language for the Simulation of Biological Systems¹

Roberto Barbuti,² Giulio Caravagna,³
Andrea Maggiolo–Schettini⁴ and Paolo Milazzo⁵

*Dipartimento di Informatica
Università di Pisa
Largo Bruno Pontecorvo 3, 56127 - Pisa, Italy*

Abstract

We propose String MultiSet Rewriting (SMSR) as an intermediate language for simulation of biomolecular systems. Higher level formalisms for biological systems description can be translated into SMSR and SMSR descriptions can be simulated by adapting an existing simulator. In this paper we show the translation of one of these formalisms, CLS+, into SMSR, and we prove correctness and completeness of the translation.

Keywords: MultiSet Rewriting, Stochastic Simulation, Calculus of Looping Sequences.

1 Introduction

Stochastic simulation of biomolecular systems traditionally is based on Gillespie's framework [8] which describes a system as a multiset of elements representing molecules. A system transformation due to a chemical reaction among molecules is described as the replacement in the multiset of the elements representing reactants with those representing products of the reaction.

Multisets and their transformations are easily implemented and many tools exist to the purpose. Moreover, multisets and their transformations are formalized as multiset rewriting systems [10].

In the last years the need has arisen to describe biological phenomena at system level, namely by ignoring structural and behavioral details of individual system com-

¹ This research has been partially supported by MiUR PRIN 2006 Project "Biologically Inspired Systems and Calculi and their Applications (BISCA)".

² Email: barbuti@di.unipi.it

³ Email: caravagn@di.unipi.it

⁴ Email: maggiolo@di.unipi.it

⁵ Email: milazzo@di.unipi.it

ponents and by taking into account organization of components in compartments and their interaction capabilities.

Multiset rewriting does not allow descriptions at this high level and, consequently, many formalisms, sometimes adaption of existing ones, have been proposed. We mention, as examples, The κ -calculus [6], the biochemical Stochastic pi-calculus [14], the Brane Calculi [3] and P Systems [12]. For some of the formalisms mentioned specific simulators exist (e.g. SPiM [15] based on the Stochastic pi-calculus, and CytoSim and PSym [13] based on P Systems). However, the development of simulators for high level descriptions may be not easy. Moreover, also the translation from a high level formalism to multiset rewriting which allows the use of existing simulators, may pose some difficulties.

In this paper we propose an extension of multiset rewriting, called *String MultiSet Rewriting (SMSR)*, in which multiset elements are strings and left hand sides of rewrite rules can contain an operator, called maximal matching operator, which allows representing the multiset of all strings having a common given prefix.

SMSR can be used as an intermediate language for simulation. On the one end, it is easy to develop simulators for SMSR, for instance by extending the GBS simulator [9]. On the other hand, the maximal matching operator facilitates the translation of higher level languages, in particular those based on term rewriting. The idea is that a term can be seen as a tree, a tree can be seen as a set of strings representing all paths from root to leaves and the replacement of a subtree becomes the replacement of a set of strings having a common prefix. As an example we show how a formalism based on term rewriting, CLS+ [11,2] can be translated into SMSR, proving translation correctness and completeness.

Other formalisms, such as Brane Calculi [3] and P Systems [12], could be translated into SMSR directly or via their translation into CLS+ along the lines described in [11,1]. In both cases one would have the possibility of using the simulator for SMSR to simulate high level descriptions.

2 String MultiSet Rewriting

In this section we introduce the *String MultiSet Rewriting* formalism. It is based on term rewriting: we will define the syntax of terms and a structural congruence relation on them. Then we will introduce rewrite rules and define an operational semantics describing the evolution of terms by means of rewrite rules application.

We assume an infinite alphabet $\mathcal{E} = \{e_1, e_2, \dots\}$ and an infinite set of variables $\mathcal{V} = \mathcal{V}_{\mathcal{E}} \cup \mathcal{V}_{\mathcal{S}} \cup \mathcal{V}_{\mathcal{M}}$ where $\mathcal{V}_{\mathcal{E}}$ is a infinite set of *element variables*, ranged over by x, y, z, \dots , $\mathcal{V}_{\mathcal{S}}$ is a infinite set of *string variables*, ranged over by $\tilde{x}, \tilde{y}, \tilde{z}, \dots$, and $\mathcal{V}_{\mathcal{M}}$ is a infinite set of *multiset variables*, ranged over X, Y, Z, \dots .

Definition 2.1 (Terms) *Multisets* M and *strings* S over an alphabet \mathcal{E} are defined by the following grammar:

$$\begin{array}{l} M ::= S \mid M \mid M \\ S ::= \epsilon \mid e \mid S \cdot S \end{array}$$

where ϵ is the empty string, e is a generic element of \mathcal{E} , \cdot is string concatenation,

and $|$ is multiset union. We denote the set of all multisets as \mathcal{M} and the set of all strings as \mathcal{S} .

Strings over \mathcal{E} can be constructed by means of the concatenation operator \cdot , with ϵ representing the concatenation of zero elements. Multisets of strings can be constructed by means of the union operator $|$. We use the notation $|$ for multiset union instead of the more usual notation \cup to have a notation similar to that of process calculi.

Now we define a structural congruence relation on terms to express the associativity of \cdot and $|$, the commutativity of the latter, and the neutral role of ϵ .

Definition 2.2 (Structural Congruence) The structural congruence relation \equiv is the least congruence relation on multisets satisfying following axioms:

$$\begin{aligned} S_1 \cdot (S_2 \cdot S_3) &\equiv (S_1 \cdot S_2) \cdot S_3 & S \cdot \epsilon &\equiv S \\ M_1 | M_2 &\equiv M_2 | M_1 & M_1 | (M_2 | M_3) &\equiv (M_1 | M_2) | M_3 & M | \epsilon &\equiv M \end{aligned}$$

By means of the structural congruence we can define the usual containment operator \in on multisets as follows: $S \in M \iff \exists M'. M \equiv S | M'$.

Now we introduce SMSR patterns, that are terms enriched with variables and with a maximal matching operator. This operator is the main novelty of SMSR.

Definition 2.3 (Patterns) *Multiset patterns* MP and *string patterns* SP over an alphabet \mathcal{E} are defined by the following grammar:

$$\begin{aligned} MP &::= SP \mid MP | MP \mid \{\!\{SP\}\!\}_X \mid \{\!\{SP\}\!\}_{SP} \\ SP &::= \epsilon \mid e \mid SP \cdot SP \mid \tilde{x} \mid x \end{aligned}$$

where ϵ is the empty string, e is a generic element of \mathcal{E} , \tilde{x} , x and X are generic string, element and multiset variables, \cdot is string concatenation, $|$ is multiset union and $\{\!\{-\}\!\}_-$ is the maximal matching operator. We denote the set of all multiset patterns as \mathcal{MP} and the set of all string patterns as \mathcal{SP} .

Patterns are used to define rewrite rules of SMSR. A rewrite rule is essentially a pair of patterns in which the first element describes the term that is modified by an application of the rule and the second describes how the term changes after the application. Variables in patterns allow a rewrite rule to be applicable to any term that can be obtained by replacing such variables with proper elements or strings. The maximal matching operator $\{\!\{-\}\!\}_-$ represents a multiset of strings which have as prefix the same instantiation of the string pattern SP .

As it is shown in the grammar of multiset patterns, we have two different forms for the maximal matching operator. In the first case we have a multiset variable as subscript of the operator, namely we have $\{\!\{SP\}\!\}_X$. In the second case the subscript is a sequence pattern, namely we have $\{\!\{SP_1\}\!\}_{SP_2}$.

We define a *pattern expansion function* $\langle _ \rangle_-$. In the case of the first form of the maximal matching operator the function transforms each maximal matching operator into the union of sequence patterns, all with the same prefix S followed by different sequence variables. We call *n-expansion* of $\{\!\{S\}\!\}_X$ the replacement of

a maximal matching operator $\{\!\{S\}\!\}_X$ by a union of n sequence patterns $S \cdot \tilde{x}_1 \mid \dots \mid S \cdot \tilde{x}_n$. The value of n for the expansion of each maximal matching operator will be given by an auxiliary function $\rho : \mathcal{V}_S \rightarrow \mathbb{N}$, which is a parameter of the pattern expansion function. In the case of the second form of the maximal matching operator, $\{\!\{SP_1\}\!\}_{SP_2}$, the expansion function gives $SP_1 \cdot SP_2$.

Definition 2.4 (Pattern Expansion Function) The pattern expansion function $\langle _ \rangle_\rho : \mathcal{MP} \times (\mathcal{V}_M \rightarrow \mathbb{N}) \rightarrow \mathcal{M}$ is recursively defined as follows:

$$\begin{aligned} \langle SP \rangle_\rho &= SP \\ \langle \{\!\{SP\}\!\}_X \rangle_\rho &= SP \cdot \tilde{x}_1 \mid \dots \mid SP \cdot \tilde{x}_{\rho(X)} \quad \text{where } \tilde{x} \in \mathcal{V}_S \\ \langle \{\!\{SP_1\}\!\}_{SP_2} \rangle_\rho &= SP_1 \cdot SP_2 \quad \text{where } SP_2 \in \mathcal{SP} \\ \langle MP_1 \mid MP_2 \rangle_\rho &= \langle MP_1 \rangle_\rho \mid \langle MP_2 \rangle_\rho \end{aligned}$$

Let us define the following functions:

$Var : \mathcal{MP} \mapsto \wp(\mathcal{V})$

$Var(M)$ denotes the set of variables that appear in multiset M , notice that by the expansion of the maximal matching operator we have that $Var(\{\!\{SP\}\!\}_X) = Var(SP) \cup \{\tilde{x}_i \mid i \in \mathbb{N}\}$ and that $Var(\{\!\{SP_1\}\!\}_{SP_2}) = Var(SP_1) \cup Var(SP_2)$; for instance $Var(a \cdot \tilde{x} \mid a \cdot x \mid \{\!\{d\}\!\}_Y) = \{\tilde{x}, x\} \cup \{\tilde{y}_i \mid i \in \mathbb{N}\}$;

$Symbols : \mathcal{MP} \mapsto \wp(\mathcal{E})$

$Symbols(M)$ denotes the set of elements of \mathcal{E} that appear in multiset M ; for instance $Symbols(a \cdot \tilde{x} \mid a \cdot x \mid \{\!\{d\}\!\}_e) = \{a, d, e\}$.

We assume $Symbols$ to be trivially extended to sets of SMSR terms and we define the set of *fresh names* for a multiset M as the infinite set $\mathcal{E} \setminus Symbols(M)$.

An *instantiation* is a function $\sigma : \mathcal{V}_E \cup \mathcal{V}_S \rightarrow \mathcal{M}$; we denote by Σ be the set of all possible instantiations. Given $M \in \mathcal{MP}$, with $M\sigma$ we denote the multiset obtained by replacing each occurrence of variable v appearing in M with the corresponding instantiation $\sigma(v)$. We denote with $\sigma(V)$ the set $\{T \mid \sigma(v) = T, v \in V\}$.

A rewrite rule is a pair of patterns.

Definition 2.5 (Rewrite Rule) A rewrite rule R is a pair (M, M') , denoted by $M \mapsto M'$, where $M, M' \in \mathcal{MP}$ and $M \not\equiv \epsilon$. With \mathfrak{R} we denote the set of all possible rewrite rules.

A multiset M is *ground* if and only if $Var(M) = \emptyset$; a rule $M \mapsto M'$ is ground if and only if both M and M' are ground. We write $Var(R)$ for $Var(M) \cup Var(M')$ and $Symbols(R)$ for $Symbols(M) \cup Symbols(M')$.

Notice that, in a rewrite rule $M \mapsto M'$ we have no constraints on $Var(M)$ and $Var(M')$. In particular, some variables may exist that appear only in M' and not in M . We call these variables *free* and the others, namely those appearing in M , *bound*. Formally, $FV : \mathcal{R} \mapsto \wp(\mathcal{V})$ and $BV : \mathcal{R} \mapsto \wp(\mathcal{V})$ are the functions

$$\begin{aligned} FV(M \mapsto M') &= \{v \mid v \in Var(M') \wedge v \notin Var(M)\} \\ BV(M \mapsto M') &= Var(R) \setminus FV((M, M')) = Var(M). \end{aligned}$$

Free variables are related to the generation of fresh names, in particular their

instantiation in the process of application of a rule R to a multiset M will be such that $\sigma(FV(R)) \cap (Symbols(M) \cup Symbols(R)) = \emptyset$. The semantics of SMSR, that follows, provides for the choice of the correct instantiation function.

Definition 2.6 (Semantics) Given a finite set of rewrite rules $\mathcal{R} \subset \mathfrak{R}$ the semantics of SMSR is the least labeled transition relation $\xrightarrow{\xi, \zeta}$ closed with respect to \equiv and satisfying the following inference rules:

$$(1) \quad \frac{R : M_1 \mapsto M_2 \in \mathcal{R} \quad \sigma \in \Sigma \quad \exists \rho. (\langle M_1 \rangle_\rho)\sigma \equiv M \wedge (\langle M_2 \rangle_\rho)\sigma \equiv M' \quad (Symbols(\sigma(BV(R))) \cup Symbols(R)) \cap Symbols(\sigma(FV(R))) = \emptyset}{M \xrightarrow{\{SP\sigma \mid \{SP\}_- \in M'\}, Symbols(\sigma(FV(R)))} M'}$$

$$(2) \quad \frac{M \xrightarrow{\xi, \zeta} M' \quad \forall S \in \xi. \nexists S' \in \mathcal{S}. (S \cdot S') \in M'' \quad \zeta \cap Symbols(M'') = \emptyset}{M'' \mid M \xrightarrow{\xi, \zeta} M'' \mid M'}$$

The semantics rules use the concepts of patterns expansion and instantiation.

Semantic rule (1) expresses that each occurrence of maximal matching operator, $\{SP\}_-$, in a rewrite rule must be expanded and the instantiation of SP by σ , $SP\sigma$, must be recorded as the first label of the transition in the conclusion of the semantic rule. Semantic rule 2 uses, in the parallel composition of terms, the mentioned label to ensure that the operator is instantiated into the multiset of all strings, in the term to be rewritten, prefixed by $SP\sigma$.

The second label on the transition is used to ensure that for any rule that contains free variables, the result of applying the rule to a multiset generates always fresh names. This mechanism is similar to the one used for existential quantification in first-order multiset rewriting [4].

As example, given multiset $M \equiv a \cdot b \cdot c \mid a \cdot b \cdot d \mid b$ and rules $R_1 : \{a \cdot x\}_X \mapsto c \cdot y$, $R_2 : \{a \cdot x\}_c \mapsto c$ we have that $FV(R_1) = \{y\}$, $BV(R_1) = \{x, X\}$, $Symbols(R_1) = Symbols(R_2) = \{a, c\}$, $FV(R_2) = \emptyset$ and $BV(R_2) = \{x\}$. Given a ρ function such that $\rho(X) = 2$ the expansion of patterns with respect to ρ is $\langle \{a \cdot x\}_X \rangle_\rho = a \cdot x \cdot \widetilde{x}_1 \mid a \cdot x \cdot \widetilde{x}_2$ and $\langle \{a \cdot x\}_c \rangle_\rho = a \cdot x \cdot c$. Furthermore, given an instantiation function $\sigma \in \Sigma$ such that $\sigma = \{(x, b), (\widetilde{x}_1, c), (\widetilde{x}_2, d), (y, e)\}$ we may have, when applying R_1 to M , the following transition of the semantics $M \xrightarrow{\{a \cdot b\}, \{e\}} b \mid c \cdot e$. Differently, as regards the application of R_2 to M , we have that no possible transitions of the semantics can be done; note that if $a \cdot b \cdot d$ would not be in M then also R_2 would be applicable with $b \mid c$ as a result.

3 Encoding CLS+ into SMSR

In this section we recall CLS+ and show its encoding into SMSR. At the end of this section we prove *correctness* and *completeness* of such an encoding.

3.1 The Calculus of Looping Sequences

In this section we recall the Calculus of Looping Sequences (CLS) in one of its variants, CLS+[11,2]. CLS+ is essentially based on term rewriting, hence a CLS+ model consists of a term and a set of rewrite rules. The term is intended to represent the structure of the modeled system, and the rewrite rules the events that may cause the system to evolve.

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

Definition 3.1 (Terms) *Terms* T , *Branes* B , and *Sequences* S of CLS+ are given by the following grammar:

$$\begin{aligned} T & ::= S \mid (B)^L \mid T \mid T \mid T \\ B & ::= S \mid B \mid B \\ S & ::= \epsilon \mid a \mid S \cdot S \end{aligned}$$

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

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

In CLS+ we may have syntactically different terms representing the same structure. The structural congruence relation of CLS+ expresses associativity of both \cdot and \mid , commutativity of the latter and the neutral role of ϵ with respect to all the operators.

Definition 3.2 (Structural Congruence) The structural congruence relations \equiv_S , \equiv_B and \equiv_T are the least congruence relations on sequences, on branes and on terms, respectively, satisfying the following rules:

$$\begin{aligned} S_1 \cdot (S_2 \cdot S_3) &\equiv_S (S_1 \cdot S_2) \cdot S_3 & S \cdot \epsilon &\equiv_S \epsilon \cdot S \equiv_S S \\ S_1 &\equiv_S S_2 \text{ implies } S_1 &\equiv_B S_2 \\ B_1 \mid B_2 &\equiv_B B_2 \mid B_1 & B_1 \mid (B_2 \mid B_3) &\equiv_B (B_1 \mid B_2) \mid B_3 & B \mid \epsilon &\equiv_B B \\ S_1 &\equiv_S S_2 \text{ implies } S_1 &\equiv_T S_2 & B_1 &\equiv_B B_2 \text{ implies } (B_1)^L \mid T &\equiv_T (B_2)^L \mid T \\ T_1 \mid T_2 &\equiv_T T_2 \mid T_1 & T_1 \mid (T_2 \mid T_3) &\equiv_T (T_1 \mid T_2) \mid T_3 & T \mid \epsilon &\equiv_T T & (\epsilon)^L \mid \epsilon &\equiv \epsilon \end{aligned}$$

Rewrite rules will be defined essentially as pairs of terms, in which the first term describes the portion of the system in which the event modeled by the rule

may occur, and the second term describes how that portion of the system changes when the event occurs. In the terms of a rewrite rule we allow the use of variables. As a consequence, a rule will be applicable to all terms which can be obtained by properly instantiating variables. Variables can be of four kinds associated with terms, branes, sequences, and alphabet elements, respectively. We assume a set of term variables TV ranged over by X, Y, Z, \dots , a set of brane variables BV ranged over by $\bar{x}, \bar{y}, \bar{z}, \dots$, a set of sequence variables SV ranged over by $\tilde{x}, \tilde{y}, \tilde{z}, \dots$, and a set of element variables \mathcal{X} ranged over by x, y, z, \dots . All these sets are possibly infinite and pairwise disjoint. We denote by \mathcal{V} the set of all variables, $\mathcal{V} = TV \cup BV \cup SV \cup \mathcal{X}$, and with ρ a generic variable of \mathcal{V} . Hence, a pattern is a term which may include variables and a rewrite rule is a pair of patterns.

Definition 3.3 (Patterns) Patterns P , brane patterns BP and sequence patterns SP of CLS+ are given by the following grammar:

$$\begin{aligned} P & ::= SP \mid (BP)^L \mid P \mid P \mid X \\ BP & ::= SP \mid BP \mid BP \mid \bar{x} \\ SP & ::= \epsilon \mid a \mid SP \cdot SP \mid \tilde{x} \mid x \end{aligned}$$

where a is a generic element of \mathcal{E} , and X, \bar{x}, \tilde{x} and x are generic elements of TV, BV, SV and \mathcal{X} , respectively. We denote with \mathcal{P} the infinite set of patterns.

Definition 3.4 (Rewrite Rules) A rewrite rule is a pair of patterns (P_1, P_2) , written as $P_1 \mapsto P_2$, where $P_1, P_2 \in \mathcal{P}$, $P_1 \neq \epsilon$ and such that $Var(P_2) \subseteq Var(P_1)$. We denote with \mathfrak{R} the infinite set of all the possible rewrite rules.

In CLS+ we have some rules that are applicable everywhere in a term, while others cannot be applied to branes. For instance, a rule such as $a \mid b \mapsto (a)^L \mid b$ cannot be applied to the elements of a looping (as in $(a \mid b)^L \mid c$) because the result of the application would not be a syntactically correct CLS+ term $((a)^L \mid b)^L \mid c$. The rules that can be applied to elements of a looping sequence are those having the form (B_1, B_2) with $B_1, B_2 \in \mathcal{B}$. We call these rules *brane rules* and we denote as $\mathfrak{R}_{\mathcal{B}} \subseteq \mathfrak{R}$ the infinite set containing all of them. Now, in the semantics of CLS+ we have to take into account brane rules and allow them to be applied also to elements of looping sequences. Hence, we define the semantics as follows.

Definition 3.5 (Semantics) Given a set of rewrite rules $\mathcal{R} \subseteq \mathfrak{R}$, and a set of brane rules $\mathcal{R}_{\mathcal{B}} \subseteq \mathcal{R}$, such that $(\mathcal{R} \setminus \mathcal{R}_{\mathcal{B}}) \cap \mathfrak{R}_{\mathcal{B}} = \emptyset$, the *semantics* of CLS is the least transition relation \rightarrow on terms closed under \equiv , and satisfying the following inference rules:

$$\begin{aligned} & \frac{(P_1, P_2) \in \mathcal{R} \quad P_1\sigma \neq \epsilon \quad \sigma \in \Sigma}{P_1\sigma \rightarrow P_2\sigma} & \frac{T_1 \rightarrow T_2}{T \mid T_1 \rightarrow T \mid T_2} & \frac{T_1 \rightarrow T_2}{(B)^L \mid T_1 \rightarrow (B)^L \mid T_2} \\ & \frac{(BP_1, BP_2) \in \mathcal{R}_{\mathcal{B}} \quad BP_1\sigma \neq \epsilon \quad \sigma \in \Sigma}{BP_1\sigma \rightarrow_{\mathcal{B}} BP_2\sigma} & \frac{B_1 \rightarrow_{\mathcal{B}} B_2}{B \mid B_1 \rightarrow_{\mathcal{B}} B \mid B_2} \\ & \frac{B_1 \rightarrow_{\mathcal{B}} B_2}{(B_1)^L \mid T \rightarrow (B_2)^L \mid T} \end{aligned}$$

where $\rightarrow_{\mathcal{B}}$ is a transition relation on branes, and where the symmetric rules for the parallel composition of terms and of branes are omitted.

The relation $\rightarrow_{\mathcal{B}}$ is used to describe the application of a brane rule to elements of a looping sequence. As usual, a CLS+ model is composed by a term, representing the initial state of the modeled system, and a set of rewrite rules.

3.2 Encoding of CLS+

Firstly we notice that, for the sake of simplicity, we assume a slight restriction on the syntax of CLS+ patterns. In particular, we require that term and brane variables appear always inside the operands of some $(-)^L \rfloor -$ operator. In other words, we avoid rewrite rules having the following forms: $X \rfloor \dots \mapsto \dots$ and $\bar{x} \rfloor \dots \mapsto \dots$

The reason for this restriction is that, as we shall see, we will use the maximal matching operator of SMSR to encode term and brane variables of CLS+. This means that the translation of a term or of a brane variable will be always instantiated in a maximal way. This is correct with respect to the semantics of CLS+ for variables that appear in the operand of a looping and containment operator (for instance, the only possible instantiation for X in $(a)^L \rfloor (X \mid a) \mapsto d$ when the rule is applied to $(a)^L \rfloor (a \mid b \mid c)$ is $b \mid c$), but not otherwise (for instance, X in $X \mid a \mapsto d$ when the rule is applied to $a \mid b \mid c$ can be instantiated either to ϵ , or b , or c , or $b \mid c$, so to obtain as results of the application either $d \mid b \mid c$, or $d \mid c$, or $d \mid b$, or d , respectively). This restriction could be avoided by defining also a non-maximal matching operator in SMSR. The definition of the semantics of such an operator would be quite straightforward.

We will give two encoding functions that map CLS+ terms and patterns into SMSR terms and patterns, respectively. These functions will be used to translate both the rewrite rules and the initial term of a CLS+ model. The encoding functions are defined recursively on the structure of the CLS+ term or pattern, hence they perform a complete visit of the abstract syntax tree of such a term or pattern from root to leaves (we consider CLS+ sequences and sequence patterns as the leaves). While performing this visit, the encoding functions construct one SMSR string for each path from the root to one leaf and, eventually, they concatenate a string corresponding to the leaf to the string corresponding to its path in the tree. The result of the translation of a CLS+ pattern is hence a multiset of strings. We have a multiset of strings instead of a set because, as we shall see, the encoding will consider only the nodes of the abstract syntax tree corresponding to applications of the looping operator.

We assume that each element $e \in \mathcal{E}_{\text{CLS+}}$, where $\mathcal{E}_{\text{CLS+}}$ is the alphabet of CLS+, is also contained in the alphabet of SMSR. Moreover, we assume that for each element variable x and sequence variable \tilde{x} of CLS+, the same x and \tilde{x} exist in $\mathcal{V}_{\mathcal{E}}$ and $\mathcal{V}_{\mathcal{S}}$, respectively. We assume also that $\mathcal{V}_{\mathcal{S}}$ contains a special variable Δ that will be used in the encoding of rewrite rules. Finally, for each brane variable \bar{x} and term variable X of CLS+ we assume the existence of \bar{x} and X in $\mathcal{V}_{\mathcal{M}}$ and of infinitely many variables \bar{x}_i and X_i , for all $i \in \mathbb{N}$, in $\mathcal{V}_{\mathcal{S}}$.

In order to encode paths in the abstract syntax tree of CLS+ terms and patterns we assume that the SMSR alphabet \mathcal{E} contains two symbols $\bar{\lambda}$ and λ and all the

natural numbers \mathbb{N} . The two symbols $\bar{\lambda}$ and λ are used to distinguish between the branes and the contents in an application of the looping operator, and the natural numbers are used to distinguish two different applications of the looping operator. The two symbols $\bar{\lambda}$ and λ will be always followed by either a natural number or an element variable. To simplify the notation we will write λ_i and λ_x for $\lambda \cdot i$ and $\lambda \cdot x$, respectively, and the same with $\bar{\lambda}$ instead of λ .

We now define the encoding of CLS+ terms into SMSR multisets and of CLS+ patterns into SMSR multiset patterns. In the definitions we will use an auxiliary injection function $\triangleright : \mathcal{SP} \times \mathcal{MP} \rightarrow \mathcal{MP}$ that inserts a sequence pattern SP as a prefix of all the elements of a multiset pattern MP . The injection function is represented with infix notation and is recursively defined as follows:

$$\begin{aligned} SP_1 \triangleright SP_2 &= SP_1 \cdot SP_2 \\ SP \triangleright (MP_1 \mid MP_2) &= (SP \triangleright MP_1) \mid (SP \triangleright MP_2) \\ SP_1 \triangleright \{ \! \{ SP_2 \} \! \}_X &= \{ \! \{ SP_1 \cdot SP_2 \} \! \}_X \\ SP_1 \triangleright \{ \! \{ SP_2 \} \! \}_{SP_3} &= \{ \! \{ SP_1 \cdot SP_2 \} \! \}_{SP_3} \end{aligned}$$

We define the encoding of CLS+ terms. The encoding technique is the same as the one used in [5,7] to define enhanced semantics for the study of causality properties. Here, the idea is to represent a path in the abstract syntax tree of CLS+ term as a sequence of $\bar{\lambda}_i$ and λ_i symbols representing applications of the looping operator. We do not use any symbol to represent applications of the parallel composition operator \mid of CLS+ as it is directly translated into multiset union of SMSR. The same holds for the sequencing operator \cdot of CLS+ that is directly translated into SMSR string concatenation.

Definition 3.6 (Encoding of terms) The encoding of CLS+ terms into multisets of strings is given by the function $\lfloor _ \rfloor : \mathcal{T} \rightarrow \mathcal{M} \times \wp(\mathbb{N})$ recursively defined as follows:

$$\begin{aligned} \lfloor S \rfloor &= (S, \emptyset) \\ \lfloor T_1 \mid T_2 \rfloor &= (M_1 \mid M_2, I_1 \cup I_2) \quad \text{where } \lfloor T_i \rfloor = (M_i, I_i) \text{ and } I_1 \cap I_2 = \emptyset \\ \lfloor (B)^L \rfloor T \rfloor &= (\bar{\lambda}_i \triangleright M_1 \mid \lambda_i \triangleright M_2, I_1 \cup I_2) \\ &\quad \text{where } \lfloor B \rfloor = (M_1, I_1), \lfloor T \rfloor = (M_2, I_2) \text{ and } i \in \mathbb{N} \setminus (I_1 \cup I_2) \end{aligned}$$

The encoding of terms translates a CLS+ term T into a pair (M, I) where M is the actual result of the translation, namely the SMSR multiset corresponding to T , and I is the set of natural numbers that occur in M . Such a set of numbers is used in the definition of the encoding to ensure that different applications of the looping operator in T will be translated into occurrences of $\bar{\lambda}_i$ and λ_i having different indexes. In what follows, we will ignore this set of natural numbers and we will use $\lfloor T \rfloor$ to denote only the SMSR multiset M .

Now we define the encoding of CLS+ patterns into SMSR multiset patterns. This encoding will be used to translate CLS+ rewrite rules into rewrite rules of SMSR.

Definition 3.7 (Encoding of patterns) The encoding of CLS+ patterns into multiset patterns is given by the function $\llbracket _ \rrbracket : \mathcal{P} \rightarrow \mathcal{MP} \times \mathcal{V}_{\mathcal{E}} \times \mathcal{V}_{\mathcal{E}}$ recursively defined as follows:

$$\begin{aligned} \llbracket SP \rrbracket &= (SP, \emptyset, Var(SP)) \\ \llbracket v \rrbracket &= (\{\epsilon\}_v, \emptyset, \{v_i \mid i \in \mathbb{N}\}) \quad \forall v \in TV \cup BV \\ \llbracket P_1 \mid P_2 \rrbracket &= (MP_1 \mid MP_2, \Sigma_1 \cup \Sigma_2, \Sigma'_1 \cup \Sigma'_2) \\ &\quad \text{where } \llbracket P_i \rrbracket = (MP_i, \Sigma_i, \Sigma'_i) \text{ and } \Sigma_1 \cap \Sigma_2 = \emptyset \\ \llbracket (BP)^L \rrbracket &= (\overline{\lambda}_x \triangleright MP_1 \mid \lambda_x \triangleright MP_2, \{x\} \cup \Sigma_2, \Sigma'_1 \cup \Sigma'_2) \\ &\quad \text{where } \llbracket BP \rrbracket = (MP_1, \emptyset, \Sigma'_1), \llbracket P \rrbracket = (MP_2, \Sigma_2, \Sigma'_2), \\ &\quad \text{and } x \in \mathcal{V}_{\mathcal{E}} \setminus (\Sigma_1 \cup \Sigma'_1 \cup \Sigma_2 \cup \Sigma'_2) \end{aligned}$$

where the auxiliary encoding function $\langle _ \rangle : \mathcal{P} \rightarrow \mathcal{MP} \times \mathcal{V}_{\mathcal{E}} \times \mathcal{V}_{\mathcal{E}}$ is recursively defined as follows:

$$\begin{aligned} \langle SP \rangle &= (\{\epsilon\}_{SP}, \emptyset, Var(SP)) \\ \langle v \rangle &= (\{\epsilon\}_v, \emptyset, \{v_i \mid i \in \mathbb{N}\}) \quad \forall v \in TV \cup BV \\ \langle P_1 \mid P_2 \rangle &= (MP_1 \mid MP_2, \Sigma_1 \cup \Sigma_2, \Sigma'_1 \cup \Sigma'_2) \\ &\quad \text{where } \langle P_i \rangle = (MP_i, \Sigma_i, \Sigma'_i) \text{ and } \Sigma_1 \cap \Sigma_2 = \emptyset \\ \langle (BP)^L \rrbracket &= (\overline{\lambda}_x \triangleright MP_1 \mid \lambda_x \triangleright MP_2, \{x\} \cup \Sigma_2, \Sigma'_1 \cup \Sigma'_2) \\ &\quad \text{where } \langle BP \rangle = (MP_1, \emptyset, \Sigma'_1), \langle P \rangle = (MP_2, \Sigma_2, \Sigma'_2), \\ &\quad \text{and } x \in \mathcal{V}_{\mathcal{E}} \setminus (\Sigma_1 \cup \Sigma'_1 \cup \Sigma_2 \cup \Sigma'_2) \end{aligned}$$

The encoding of patterns translates a CLS+ pattern P into a triple (MP, Σ, Σ') , where MP is the actual result of the translation, namely the SMSR multiset pattern corresponding to P , the set Σ contains all the element variables that are used in MP as subscripts of $\overline{\lambda}$ and λ symbols, and the set Σ' contains all the other variables that may appear in MP . The set Σ is used to ensure that different applications of the looping operator in P will be translated into occurrences of symbols $\overline{\lambda}$ and λ having different subscripts. The set Σ' , instead, will be used in the following to translate CLS+ rewrite rules. In what follows, when we do not represent explicitly the triple (MP, Σ, Σ') obtained from the encoding, we will use $\llbracket T \rrbracket$ and $\langle T \rangle$ to denote only the SMSR multiset pattern MP .

Now, a CLS+ model consisting in an initial term T and a set of rewrite rules $\mathcal{R} = \{P_1 \mapsto P'_1, \dots, P_n \mapsto P'_n\}$ can be translated into a SMSR model consisting in the initial term $\llbracket T \rrbracket$ and in a set of rewrite rules derived from \mathcal{R} . The translation of CLS+ rewrite rules uses the encoding $\llbracket _ \rrbracket$ and is different in the two cases of brane rules and non-brane rules.

The translation of a CLS+ brane rule $BP_1 \mapsto BP_2 \in \mathfrak{R}_{\mathcal{B}}$ is simple as brane rules can be applied everywhere in a CLS+ term. Consequently, the SMSR rule

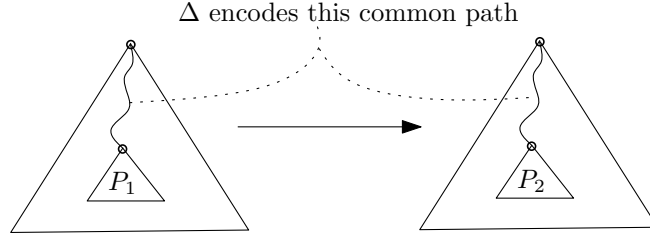


Fig. 1. Visual representation of the variable Δ in the application of rewrite rules.

corresponding to $BP_1 \mapsto BP_2$ is

$$\Delta \triangleright MP_1 \mapsto \Delta \triangleright MP_2 \quad (1)$$

where $\llbracket BP_1 \rrbracket = (MP_1, \Sigma_1, \Sigma'_1)$, $\llbracket BP_2 \rrbracket = (MP_2, \Sigma_2, \Sigma'_2)$, $\Sigma_1 \cap \Sigma_2 = \emptyset$ and $\Delta \notin \Sigma_1 \cup \Sigma'_1 \cup \Sigma_2 \cup \Sigma'_2$.

Note that, as shown in Figure 1, the instantiation of the special variable Δ added in MP_1 and in MP_2 will represent, during the application of R , the path from the root of the abstract syntax tree to the point in the tree in which the rule will be applied.

The case of the translation of a non-brane CLS+ rule, namely of a rule $P_1 \mapsto P_2 \notin \mathfrak{R}_{\mathcal{B}}$, is a bit more complicated as such a rule in CLS+ can be applied only either to the components of a parallel composition at the top level of the term, or to the components of a parallel composition inside some looping sequence. In order to obtain the corresponding result after translation into SMSR, we translate the rule $P_1 \mapsto P_2$ into two SMSR rewrite rules, namely:

$$MP_1 \mapsto MP_2 \quad \text{and} \quad \Delta \cdot \lambda_x \triangleright MP_1 \mapsto \Delta \cdot \lambda_x \triangleright MP_2$$

where $\llbracket P_1 \rrbracket = (MP_1, \Sigma_1, \Sigma'_1)$, $\llbracket P_2 \rrbracket = (MP_2, \Sigma_2, \Sigma'_2)$, $\Sigma_1 \cap \Sigma_2 = \emptyset$ and $\{\Delta, x\} \cap (\Sigma_1 \cup \Sigma'_1 \cup \Sigma_2 \cup \Sigma'_2) = \emptyset$.

The first of the two SMSR rules will be applicable only to strings representing components of a parallel composition at the top level of the CLS+ term, and the second SMSR rule only to strings representing components of a parallel composition inside some looping sequence in the CLS+ term.

3.3 Example

As example we show the encoding and some steps of evolution of a simple CLS+ model where T represents the initial state of the computation and $\{R_1, R_2\}$ is the set of CLS+ rules.

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

$$(R_1) \quad a \mid b \mapsto c$$

$$(R_2) \quad ((a)^L \rfloor (X)) \mid ((b)^L \rfloor (Y)) \mapsto (a|b)^L \rfloor (X|Y)$$

The term T represents two cells: one with membrane a and containing the sequence a , the other with membrane b and containing the sequence b . Brane rule R_1 states that a parallel composition of sequences a and b is rewritten into a sequence c . Non-brane rule R_2 describes the fusion of a cell with membrane a with a neighbor cell with membrane b ; the resulting cell will have both a and b on the membrane and its content will be the merge of the content of both cells.

A possible CLS+ evolution for T is

$$\begin{aligned} T &\equiv ((a)^L \rfloor (a)) \mid ((b)^L \rfloor (b)) \\ &\rightarrow (a|b)^L \rfloor (a|b) \quad \text{applying rule } (R_2) \\ &\rightarrow (a|b)^L \rfloor (c) \quad \text{applying rule } (R_1) \\ &\rightarrow (c)^L \rfloor (c) \quad \text{applying rule } (R_1) \end{aligned}$$

As previously defined, we have that T is encoded into the multiset

$$M \equiv \bar{\lambda}_1 \cdot a \mid \lambda_1 \cdot a \mid \bar{\lambda}_2 \cdot b \mid \lambda_2 \cdot b$$

and rules are encoded into the following rules

- (1) $\Delta \cdot a \mid \Delta \cdot b \mapsto \Delta \cdot c$
- (2) $\{\Delta \cdot \lambda_w \cdot \bar{\lambda}_x\}_a \mid \{\Delta \cdot \lambda_w \cdot \lambda_x\}_X \mid \{\Delta \cdot \lambda_w \cdot \bar{\lambda}_y\}_b \mid \{\Delta \cdot \lambda_w \cdot \lambda_y\}_Y$
 $\mapsto \{\Delta \cdot \lambda_w \cdot \bar{\lambda}_z\}_a \mid \{\Delta \cdot \lambda_w \cdot \bar{\lambda}_z\}_b \mid \{\Delta \cdot \lambda_w \cdot \lambda_z\}_X \mid \{\Delta \cdot \lambda_w \cdot \lambda_z\}_Y$
- (3) $\{\bar{\lambda}_x\}_a \mid \{\lambda_x\}_X \mid \{\bar{\lambda}_y\}_b \mid \{\lambda_y\}_Y \mapsto \{\bar{\lambda}_z\}_a \mid \{\bar{\lambda}_z\}_b \mid \{\lambda_z\}_X \mid \{\lambda_z\}_Y$

where (1) is the result of encoding the brane rule R_1 and rules (2) and (3) are the result of encoding the non-brane rule R_2 . We note that using the maximal matching operator in the encoding of looping operators leads to the fact that rule (2) will not be applicable to the encoding of a term like $(a|b)^L \rfloor \dots$

A SMSR computation corresponding to the shown CLS+ computation is

$$\begin{aligned} T &\equiv \bar{\lambda}_1 \cdot a \mid \lambda_1 \cdot a \mid \bar{\lambda}_2 \cdot b \mid \lambda_2 \cdot b \\ &\rightarrow \bar{\lambda}_3 \cdot a \mid \bar{\lambda}_3 \cdot b \mid \lambda_3 \cdot a \mid \lambda_3 \cdot b \quad \text{applying rule (3)} \\ &\rightarrow \bar{\lambda}_3 \cdot a \mid \bar{\lambda}_3 \cdot b \mid \lambda_3 \cdot c \quad \text{applying rule (1)} \\ &\rightarrow \bar{\lambda}_3 \cdot c \mid \lambda_3 \cdot c \quad \text{applying rule (1)} \end{aligned}$$

3.4 Correctness and Completeness of the Translation

We start with showing how SMSR instantiation functions $\sigma_{[\cdot]}$ can be obtained from a CLS instantiation function σ through the encoding function $[\cdot]$. This can be done as follows:

$$\begin{aligned} \forall v \in SV \cup EV &\Rightarrow \sigma_{[\cdot]}(v) = [\sigma(v)] \\ \forall v \in TV \cup BV. [\sigma(v)] = S_1 \mid \dots \mid S_n &\Rightarrow \forall i = 1, \dots, n. \sigma_{[\cdot]}(v_i) = S_i \end{aligned}$$

Note that for a function σ there exist infinite $\sigma_{[\cdot]}$ that satisfy the given construction.

We then prove the following lemma that will be used in the proof of equivalence.

Lemma 3.8 *For any CLS+ pattern P and any instantiation function $\sigma \in \Sigma$ a function ρ exists such that $\lfloor P\sigma \rfloor \equiv \langle \llbracket P \rrbracket \rangle_{\rho\sigma_{[\cdot]}}$.*

Proof. The proof is made by structural induction on CLS+ patterns.

- if $P \equiv SP$, $P \equiv \tilde{x}$ or $P \equiv x$ for any ρ the proof is trivial because encoding and expansion are the identity on P .
- If $P \equiv v \in TV \cup BV$ we have to prove that $\lfloor \sigma(v) \rfloor \equiv \langle \llbracket v \rrbracket \rangle_{\rho\sigma_{[\cdot]}}$. Let $\lfloor \sigma(v) \rfloor = S_1 | \dots | S_n$ then given a ρ such that $\rho(v) = n$, we have that $\langle \llbracket v \rrbracket \rangle_{\rho\sigma_{[\cdot]}} \equiv \langle \{\epsilon\}_v \rangle_{\rho\sigma_{[\cdot]}} \equiv \sigma_{[\cdot]}(v_1) | \dots | \sigma_{[\cdot]}(v_n)$. The proof follows trivially from the definition of $\sigma_{[\cdot]}$.
- If $P \equiv P_1 | P_2$ for P_1 and P_2 patterns, we prove that $\lfloor (P_1 | P_2)\sigma \rfloor \equiv \langle \llbracket P_1 | P_2 \rrbracket \rangle_{\rho\sigma_{[\cdot]}}$. As instantiation σ , encoding function $\lfloor _ \rfloor$ and patterns expansion function distribute over $|$ we have that $\lfloor (P_1 | P_2)\sigma \rfloor \equiv \lfloor P_1\sigma | P_2\sigma \rfloor \equiv \lfloor P_1\sigma \rfloor | \lfloor P_2\sigma \rfloor$ and that $\langle \llbracket P_1 | P_2 \rrbracket \rangle_{\rho\sigma_{[\cdot]}} \equiv \langle \llbracket P_1 \rrbracket \rangle_{\rho\sigma_{[\cdot]}} | \langle \llbracket P_2 \rrbracket \rangle_{\rho\sigma_{[\cdot]}}$; assuming inductive hypothesis on P_1 and P_2 yields the proof.
- If $P \equiv (BP)^L \rfloor P$ we prove that $\lfloor ((BP)^L \rfloor P)\sigma \rfloor \equiv \langle \llbracket (BP)^L \rfloor P \rrbracket \rangle_{\rho\sigma_{[\cdot]}}$. We have that $\lfloor ((BP)^L \rfloor P)\sigma \rfloor \equiv \lfloor (BP\sigma)^L \rfloor P\sigma \rfloor \equiv \bar{\lambda}_i \triangleright \lfloor BP\sigma \rfloor | \lambda_i \triangleright \lfloor P\sigma \rfloor$ and that $\langle \llbracket (BP)^L \rfloor P \rrbracket \rangle_{\rho\sigma_{[\cdot]}} \equiv \langle \bar{\lambda}_i \triangleright \langle \llbracket BP \rrbracket \rangle_{\rho\sigma_{[\cdot]}} | \lambda_i \triangleright \langle \llbracket P \rrbracket \rangle_{\rho\sigma_{[\cdot]}} \rangle_{\rho\sigma_{[\cdot]}}$. The definition of $\langle _ \triangleright _ \rangle$ is similar to the one of $\llbracket _ \rrbracket$, hence it could be easily proved that the lemma holds also when $\llbracket P \rrbracket$ is replaced by $\langle \llbracket P \rrbracket \rangle$. By the definition of $_ \triangleright _$ and $\langle _ \triangleright _ \rangle_{\rho}$ we can write $\langle \bar{\lambda}_i \triangleright \langle \llbracket BP \rrbracket \rangle_{\rho\sigma_{[\cdot]}} | \lambda_i \triangleright \langle \llbracket P \rrbracket \rangle_{\rho\sigma_{[\cdot]}} \rangle_{\rho\sigma_{[\cdot]}} \equiv \bar{\lambda}_i \triangleright \langle \llbracket BP \rrbracket \rangle_{\rho\sigma_{[\cdot]}} | \lambda_i \triangleright \langle \llbracket P \rrbracket \rangle_{\rho\sigma_{[\cdot]}}$ and the proof follows from inductive hypothesis on BP and P . □

Finally, we prove the correspondence between the semantics of CLS+ and SMSR.

Theorem 3.9 *For any CLS+ terms T and T'*

$$T \rightarrow T' \Leftrightarrow \exists \xi, \zeta. \lfloor T \rfloor \xrightarrow{\xi, \zeta} \lfloor T' \rfloor$$

Proof. The proof of *correctness* (\Rightarrow) is made by induction on the rules of the semantics of CLS+.

- We prove that $P_1\sigma \rightarrow P_2\sigma \Rightarrow \lfloor P_1\sigma \rfloor \xrightarrow{\xi, \zeta} \lfloor P_2\sigma \rfloor$ using rule (1) of SMSR. We assume $P_1\sigma \rightarrow P_2\sigma$; as (P_1, P_2) is a CLS+ rule, then $(\Delta \triangleright \llbracket P_1 \rrbracket, \Delta \triangleright \llbracket P_2 \rrbracket)$ is its encoding in SMSR. Let $\llbracket P_1 \rrbracket \equiv M_1$, $\llbracket P_2 \rrbracket \equiv M_2$, $\lfloor P_1\sigma \rfloor \equiv M$ and $\lfloor P_2\sigma \rfloor \equiv M'$. The SMSR instantiation function $\sigma_{[\cdot]}$ that we need is one out of the infinite that can be built with respect to the σ of CLS+ and is such that $\sigma_{[\cdot]}(\Delta) = \epsilon$ and it also has to satisfy the constraints imposed by SMSR rule (1). Notice that infiniteness of SMSR alphabet guarantees the existence of a $\sigma_{[\cdot]}$ satisfying such constraints. As regards the function ρ we need, its existence is proved by Lemma 3.8.
- We prove that $T|T_1 \rightarrow T|T_2 \Rightarrow \lfloor T|T_1 \rfloor \xrightarrow{\xi, \zeta} \lfloor T|T_2 \rfloor$. Since $\lfloor _ \rfloor$ distributes over $|$ rule (2) of SMSR can be applied where $M'' \equiv \lfloor T \rfloor$, $M \equiv \lfloor T_1 \rfloor$ and $M' \equiv \lfloor T_2 \rfloor$. By inductive hypothesis $T_1 \rightarrow T_2 \Rightarrow \lfloor T_1 \rfloor \xrightarrow{\xi, \zeta} \lfloor T_2 \rfloor$ constraints in (2) are satisfied

because either T_1 does not contain term, brain variables or looping operators hence ξ is empty, or, by the restrictions on the term and brane variables and by the encoding of patterns containing looping operators, we have that all the prefixes in ξ are different from any other prefix in M'' . Finally, as regards constraints on ζ we have that, as in the previous case of the proof, there exist a $\sigma_{[\cdot]}$ such that it generates fresh names.

- We prove that $(B)^L \mid T_1 \rightarrow (B)^L \mid T_2 \Rightarrow \llbracket (B)^L \mid T_1 \rrbracket \xrightarrow{\xi, \zeta} \llbracket (B)^L \mid T_2 \rrbracket$. Since $\llbracket _ \rrbracket$ distributes over $(_)^L \mid _$ rule (2) of SMSR can be applied where $M'' \equiv \llbracket B \rrbracket$, $M \equiv \llbracket T_1 \rrbracket$ and $M' \equiv \llbracket T_2 \rrbracket$. By inductive hypothesis $T_1 \rightarrow T_2 \Rightarrow \llbracket T_1 \rrbracket \xrightarrow{\xi, \zeta} \llbracket T_2 \rrbracket$ we can notice that constraints on ξ are satisfied. This because, by the definition of the encoding function, every string in M'' has a prefix $\bar{\lambda}_i$ that is different from any prefix λ_i in the encoding of T_1 . As before, constraints on ζ are satisfied by a proper choice of a $\sigma_{[\cdot]}$.
- We prove that $(B_1)^L \mid T \rightarrow (B_2)^L \mid T \Rightarrow \llbracket (B_1)^L \mid T \rrbracket \xrightarrow{\xi, \zeta} \llbracket (B_2)^L \mid T \rrbracket$. The proof of $B_1 \rightarrow_B B_2 \Rightarrow \llbracket B_1 \rrbracket \xrightarrow{\xi, \zeta} \llbracket B_2 \rrbracket$ is given by the first two cases of the proof of correctness.

In order to prove the *completeness* (\Leftarrow) we define the inverse of the encoding function $\llbracket _ \rrbracket$.

$$\llbracket \langle \Delta \cdot C_1, S_1 \rangle \mid \dots \mid \langle \Delta \cdot C_n, S_n \rangle \rrbracket^{-1} = \llbracket \langle C_1, S_1 \rangle \mid \dots \mid \langle C_n, S_n \rangle \rrbracket^{-1}$$

$$\begin{aligned} \llbracket \langle \epsilon, S_1 \rangle \mid \dots \mid \langle \epsilon, S_{n_1} \rangle \mid \langle \bar{\lambda}_1, S'_1 \rangle \mid \dots \mid \langle \bar{\lambda}_{n_2}, S'_{n_2} \rangle \mid \langle \lambda_1 \cdot C_1, S''_1 \rangle \mid \dots \mid \langle \lambda_1 \cdot C_{n_3}, S''_{n_3} \rangle \rrbracket^{-1} = \\ S_1 \mid \dots \mid S_{n_1} \mid (S'_1 \mid \dots \mid S'_{n_2})^L \mid (\llbracket \langle C_1, S'_1 \rangle \mid \dots \mid \langle C_{n_3}, S''_{n_3} \rangle \rrbracket^{-1}) \end{aligned}$$

where $S_i \notin \mathcal{C}_S$ and $n_1, n_2, n_3 \geq 0$.

We assume also the inversion of the encoding functions $\llbracket _ \rrbracket$. We can now prove completeness by induction on the rules of the semantics of SMSR.

- We prove that $\exists \xi, \zeta. M \xrightarrow{\xi, \zeta} M' \Rightarrow \llbracket M \rrbracket^{-1} \rightarrow \llbracket M' \rrbracket^{-1}$. We assume that (M_1, M_2) is a SMSR rule and we have that its corresponding CLS+ rule (P_1, P_2) is the one such that $\sigma_{[\cdot]}(\Delta) \triangleright \llbracket P_1 \rrbracket \equiv M$ and $\sigma_{[\cdot]}(\Delta) \triangleright \llbracket P_2 \rrbracket \equiv M'$. Notice that the value $\sigma_{[\cdot]}(\Delta)$ represents the encoding of the CLS+ context in which the rule is applied; in other words this corresponds to a series of applications of the rules of the CLS+ semantics that chooses the point of application of the rule (P_1, P_2) inside the term representing the state of the computation. As regards the instantiation function for CLS+ we can notice that, since we have the one for SMSR, we can build an ad-hoc function σ as follows: $\forall v \in EV \cup SV. \sigma(v) = \sigma_{[\cdot]}(v)$ and $\forall V \in TV \cup BV. \sigma_{[\cdot]}(V_1) = S_1, \dots, \sigma_{[\cdot]}(V_n) = S_n \Rightarrow \sigma(V) = \llbracket S_1 \mid \dots \mid S_n \rrbracket^{-1}$.
- We prove that $\exists \xi, \zeta. M'' \mid M \xrightarrow{\xi, \zeta} M'' \mid M' \Rightarrow \llbracket M'' \mid M \rrbracket^{-1} \rightarrow \llbracket M'' \mid M' \rrbracket^{-1}$. By inductive hypothesis $M \xrightarrow{\xi, \zeta} M' \Rightarrow \llbracket M \rrbracket^{-1} \rightarrow \llbracket M' \rrbracket^{-1}$; let $\llbracket M \rrbracket^{-1} \equiv T_1$ and $\llbracket M' \rrbracket^{-1} \equiv T_2$. The decoding distributes over \mid , and hence $\llbracket M'' \mid M \rrbracket^{-1} \equiv \llbracket M'' \rrbracket^{-1} \mid \llbracket M \rrbracket^{-1}$ and $\llbracket M'' \mid M' \rrbracket^{-1} \equiv \llbracket M'' \rrbracket^{-1} \mid \llbracket M' \rrbracket^{-1}$. We have the proof with $\llbracket M'' \rrbracket^{-1} \equiv T$. \square

4 Conclusions

We have proposed String MultiSet Rewriting (SMSR) as an intermediate language for the simulation of biomolecular systems. SMSR is an extension of multiset rewriting with strings as multiset elements and with a maximal matching operator. Higher level formalisms for biological systems descriptions can be translated into SMSR and SMSR descriptions can be simulated by adapting an existing simulator. We have shown the translation of one of these formalisms, CLS+, into SMSR, and we have proven correctness and completeness of the translation.

References

- [1] Barbuti, R., A. Maggiolo-Schettini, P. Milazzo and A. Troina. *A Calculus of Looping Sequences for Modelling Microbiological Systems*. *Fundamenta Informaticae* **72** (2006), 21–35.
- [2] Barbuti, R., A. Maggiolo-Schettini, P. Milazzo, and A. Troina. The Calculus of Looping Sequences for Modeling Biological Membranes. In *8th Workshop on Membrane Computing (WMC8)*, LNCS, Springer, to appear.
- [3] Cardelli L. Brane Calculi. Interactions of Biological Membranes. In *Computational Methods in Systems Biology (CMSB'04)*, LNCS 3082, pages 257–280, Springer, 2005.
- [4] Cervesato, I. *The Logical Meeting Point of Multiset Rewriting and Process Algebra*. Technical Memo CHACS-5540-153, Center for High Assurance Computer Systems, Naval Research Laboratory, Washington, DC, 2004.
- [5] Curti, M., P. Degano, C. Priami and C.T. Baldari. *Modelling Biochemical Pathways through Enhanced Pi-calculus*. *Theoretical Computer Science* **325** (2004), 111–140.
- [6] Danos, V., and C. Laneve. *Formal Molecular Biology*. *Theoretical Computer Science* **325** (2004), 69–110.
- [7] Degano, P. and C. Priami. *Enhanced Operational Semantics: A Tool for Describing and Analyzing Concurrent Systems*. *ACM Computing Surveys* **33** (2001), 135–176.
- [8] Gillespie, D. *Exact Stochastic Simulation of Coupled Chemical Reactions*. *Journal of Physical Chemistry* **81** (1977), 2340–2361.
- [9] Generic Biological Simulator (GBS) web site <http://www.di.unipi.it/~milazzo/biosims/>.
- [10] Martinelli F., Bistarelli S., Cervesato I., Lenzini G., Marangoni R. Representing Biological Systems Through Multiset Rewriting. In *Computer Aided Systems Theory (EUROCAST'03)*, LNCS 2809, pages 415–426, Springer, 2004.
- [11] Milazzo, P. “Qualitative and Quantitative Formal Modeling of Biological Systems”. PhD Thesis, University of Pisa, 2007.
- [12] Păun, G. “Membrane Computing. An Introduction”. Springer, 2002.
- [13] The P Systems web page: <http://psystems.disco.unimib.it/>.
- [14] Priami, C., A. Regev, W. Silvermann, and E. Shapiro. *Application of a Stochastic Name-Passing Calculus to Representation and Simulation of Molecular Processes*. *Information Processing Letters* **80** (2001), 25–31.
- [15] The SPiM web page: <http://research.microsoft.com/~aphillip/spim/>.