Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 238 (2009) 23–43
www.elsevier.com/locate/entcs

Strategy-Based Proof Calculus for Membrane Systems

Oana Andrei[1,3]

INRIA Nancy Grand-Est & LORIA
Nancy, France

Dorel Lucanu[1,2]

Faculty of Computer Science
Alexandru Ioan Cuza University
Iaşi, Romania

Abstract

In a previous work, we showed how rewrite strategies can be used for defining the semantics of membrane systems, in particular for expressing the different control mechanisms in membranes. The insufficient expressivity of the existing concept of rewrite strategies for describing certain control mechanisms for membranes lead us to defining a more generic concept of strategies.
In this paper we introduce strategy-based rewriting logic which uses strategy controllers to reason at the higher level of computation given by the evolution of the membrane systems. We give a detailed presentation of the proof calculus, model theory, and completeness. A main consequence of the approach is that we get an algebraic semantics for membrane systems. Implementation issues are also discussed.

Keywords: Rewriting Logic, Strategy-based Rewriting Logic, Strategy Controller, Membrane System, Membership Equational Logic.

1 Introduction

Membrane systems represent an abstract model of parallel and distributed computing inspired by cell compartments and molecular membranes [22]. A *membrane system* consists of several disjoint or nested membranes, among which a particular one surrounding them all called *skin membrane*, and a finite set of objects O. Each membrane consists of a multiset (also called soup) w of objects from O, a set R of evolution rules over multisets of objects, and a control mechanism describing the

[1] This work is partially supported by the PN II grant ID 393/2007 and CEEX grant 47/2005.
[2] Email: dlucanu@info.uaic.ro
[3] Email: Oana.Andrei@loria.fr

doi:10.1016/j.entcs.2009.05.011

way in which the rules are used to modify the multiset w in an evolution step. The application of an evolution rule $r : u \to v$ on a multiset w consists in replacing the occurrence of u in w with v, if possible; otherwise we say that w is r-irreducible. A multiset is irreducible if it r-irreducible for each evolution rule r. An evolution step of a membrane is given by the application of several evolution rules according to its control mechanism.

There are various control mechanisms in membrane systems inspired by some biological entities. Each membrane may have its own control mechanism and a different task, and all membranes must work simultaneously to accomplish a more general task for the whole system. In this paper we consider the control mechanisms given by *maximal parallel rewriting (mpr)* and *maximal parallel rewriting with priorities (pri)*. Other available control mechanism are defined in the literature for the membrane systems like for instance: maximal parallel rewriting with promoters or inhibitors [22] where promoters and inhibitors are represented as multisets of objects associated to rules, and such a rule can be used only if all the promoting objects are present, and none of the inhibiting object is present in its membrane; the minimal parallel rewriting [8] saying that if at least a rule from a set of rules associated with a membrane can be used, then at least one rule from that membrane must be used, without any other restriction; non-synchronized rewriting [13] where any number of rules is used in any step; or cooperative rewriting [10] where, for a given k, we can use exactly k, at least k, or at most k rules.

Maximal parallel rewriting means that as many as possible evolution rules are applied in parallel. Formally, we say that $w \overset{mpr}{\Longrightarrow}_M w'$ (or $w \overset{mpr}{\Longrightarrow} w'$ if unambiguous) if and only if there is a multiset of rules $r_1 : u_1 \to v_1, \ldots, r_n : u_n \to v_n$ in the membrane M such that $w = u_1 \ldots u_n z$, $w' = v_1 \ldots v_n z$, and z is irreducible (z could be the empty multiset). In a membrane, a priority relation among rules is expressed by a partial order relation on the set of all rules, with the meaning that a rule can be chosen to be applied only if no rule of a higher priority is applicable in the same membrane. Formally, we say that $w \overset{pri}{\Longrightarrow}_M w'$ (or $w \overset{pri}{\Longrightarrow} w'$ if unambiguous) if and only if $w \overset{mpr}{\Longrightarrow} w'$ using the multiset of rules $r_1 : u_1 \to v_1, \ldots, r_n : u_n \to v_n$ and there is no rule $r : u \to v$ in M applicable to w which has a greater priority than any $r_i : u_i \to v_i$.

The following two membrane examples are used throughout the whole paper. Let M_1 denote a membrane over the set of objects $\{a, b, c, d\}$ with the evolution rules $R_1 = \{r_{11} : aa \to ab,\ r_{12} : ab \to cd,\ r_{13} : b \to dd\}$ such that r_{12} has a higher priority than r_{13}, i.e., $r_{12} > r_{13}$. We have $aaaaabb \overset{pri}{\Longrightarrow} ababcdb$ using the multiset r_{11}, r_{11}, r_{12}. The rule r_{13} cannot be applied because it has a lower priority than r_{12} and r_{12} is applicable to $aaaabb$. Another possible evolution step is $aaaaabb \overset{pri}{\Longrightarrow} abcdcda$ obtained by using the multiset of rules r_{11}, r_{12}, r_{12}. The second membrane M_2 is defined over the same set of objects as M_1 and has the evolution rules $R_2 = \{r_{21} : bb \to a,\ r_{22} : a \to ab\}$. Since M_2 does not include priorities over evolution rules, only evolution steps defined by maximal parallel rewriting are possible. Such an evolution step is $aabb \overset{mpr}{\Longrightarrow} ababa$ using r_{22}, r_{22}, r_{21}

(and this is the only evolution step starting from *aabb*).

A *state* of a membrane system is given by the state of its skin membrane, where the states of elementary or composite membranes are formally defined as follows:

- if M is the name of an elementary membrane with w the multiset of objects, then $\langle\, M \mid w \,\rangle$ is the state of the membrane;
- if M is the name of a composite membrane with w the multiset of objects, and $t_1, \ldots, t_n$ are the states of its internal membranes, then $\langle\, M \mid w \,\{\, t_1, \ldots, t_n \,\}\, \rangle$ is the state of the membrane.

We denote by $ctrl_M \in \{mpr, pri, \ldots\}$ the control associated to a membrane named M.

An *evolution step* $t \Rightarrow t'$ for a membrane system is defined over states by structural induction:

- for $t = \langle\, M \mid w \,\rangle$ and $t' = \langle\, M \mid w' \,\rangle$, if $w \overset{ctrl_M}{\Longrightarrow}_M w'$, then $t \Rightarrow t'$;
- for $t = \langle\, M \mid w \,\{\, t_1, \ldots, t_n \,\}\, \rangle$ and $t' = \langle\, M \mid w \,\{\, t'_1, \ldots, t'_n \,\}\, \rangle$,
  if
  either w is irreducible w.r.t. $\overset{ctrl_M}{\Longrightarrow}_M$ and $w' = w$, or $w \overset{ctrl_M}{\Longrightarrow}_M w'$,
  and for $i = 1, \ldots, n$,
  either t_i is irreducible (w.r.t. $\Rightarrow$) and $t'_i = t_i$, or $t_i \Rightarrow t'_i$
  and not all of $w, t_1, \ldots, t_n$ are irreducible,
  then $t \Rightarrow t'$.

For instance, a possible evolution step for the system Π_1 with the structure $\langle M_1 \mid \{M_2\}\rangle$ is $\langle\, M_1 \mid aabb \,\{\, \langle\, M_2 \mid aabb \,\rangle\, \}\, \rangle \Rightarrow \langle\, M_1 \mid abbb \,\{\, \langle\, M_2 \mid ababa \,\rangle\, \}\, \rangle$ because $aabb \overset{pri}{\Longrightarrow}_{M1} abbb$ and $aabb \overset{mpr}{\Longrightarrow}_{M2} ababa$.

Our aim is to find a suitable rewrite-based framework for specifying the dynamics of systems similar to membrane systems, where one step evolutions are given by controlled applications of sets of elementary evolution steps expressed as rewrite rules. In our opinion, *a framework (semantics) is rewrite-based if it satisfies (at least) the following conditions: the evolution rules are viewed as rewrite rules, it preserves the locality of the evolution rules, the communication and structural actions are modelled as rewrite rules, and an event (action) occurrence is given by the application of a rewrite rule.*

The first candidate for membrane systems was (conditional) Rewriting Logic (RL) [20]. We showed in [2] that it is possible to describe the operational semantics of membrane systems in RL. Moreover, it is known that the Maude system [9] can be used to analyse the behaviour of concurrent systems. However, in practice we noticed that it is not easy to identify the one step evolutions of a nested membrane system or of its subsystems because in Maude the description of the whole system is given by a flat rewrite theory, hence the difficulty to separate the rules belonging to a membrane. Moreover, since one step evolution is given by the execution of a sequence of rewrites, it is difficult the use of analysis tools, like the Maude LTL model checker [12] or `search` command. In a recent paper [17], it is shown that the rewrite theories describing membrane systems cannot preserve the maximal

concurrency given by the maximal parallel rewriting.

Then we started to look for a framework where the one step evolution can be specified by other mechanisms. In [1] we showed that the control mechanisms for membranes can be expressed by means of rewrite strategies. While the maximal parallel rewriting can be expressed using a strategy language including recursive definitions, other mechanisms, e.g., the maximal parallel rewriting with priorities, require more elaborated specification languages. For instance, the language must include means to compute the set of rules applicable in the current state.

The solution we propose in this paper is based on the combination between rewrite theories, strategies, and *strategy controllers*. The intuition behind a strategy controller is that it decides which strategy is applied in the current state. We refer to the set of evolution rules of a membrane as a *control-free membrane*. Similarly, by a *control-free membrane system* we refer to the set of evolution rules of a membrane system. A control-free membrane (system) is specified by a rewrite theory in RL. Based on the current state, a *strategy controller* produces a strategy (called *evolution strategy*) describing the possibly next one-step evolutions. For instance, if the strategy controller for M_1 is *pri*, then strategy computed by *pri* for *aabb* is $(r_{11}\, id) + (r_{12}\, r_{12})$ (the next possible one-steps are $aabb \stackrel{pri}{\Longrightarrow} abbb$ and $aabb \stackrel{pri}{\Longrightarrow} ccdd$).

We use Membership Equational Logic (MEL) as a framework where the three above concepts can be uniformly presented. We proceed in a similar way to that of Generalized Rewriting Logic [7]. For a given membrane system Π, we construct a (nontrivial) MEL theory $\mathcal{P}roof(\Pi)$ which supplies a sequents-based proof calculus for membrane systems. $\mathcal{P}roof(\Pi)$ has initial model which can be used to extract the usual operational semantics for membrane systems [2]. This is true even for membrane systems with heterogeneous control mechanisms which are not considered in [2]. Moreover, using the theory $\mathcal{P}roof(\Pi)$, we can define a model-theoretic semantics for membrane systems. In this way, a membrane system Π can be regarded as a specification of a larger class of systems having similar behavior.

Based on the theoretical foundation supplied by the theory $\mathcal{P}roof(\Pi)$, we developed a pattern-based implementation of the membrane systems using the Maude strategy language [18].

The structure of the paper is as follows. Section 2 gives the main notations and concepts related to Rewriting Logic. Section 3 is the main part of the paper. Here we describe why the membrane systems are not rewrite theories and we present the strategy-based rewriting logic for membrane systems. Both the syntax and the semantics are presented as MEL theories. Soundness and completenes results are also included. In Section 4 an implementation based on the Maude strategy language is discussed. The case of communicating and dissolving membranes is discussed in Section 5.

2 Preliminaries

We assume that the reader is familiar with the basic definitions and notations for many-sorted equational logic [14] and term rewriting [3].

Term Algebra. A *many-sorted signature* is a pair (S, Σ), where S is called the *sort set* and Σ is an $S^* \times S$-sorted family $\{\Sigma_{w,s} \mid w \in S^* \text{ and } s \in S\}$. A many-sorted Σ-*algebra* A consists of a many-sorted family of sets $\{A_s\}_{s\in S}$ and an operation $A_f : A_{s_1} \dots A_{s_n} \longrightarrow A_s$ for every operation symbol $f \in \Sigma_{s_1\dots s_n,s}$. For $X = \{X_s\}$ an S-sorted family of disjoint sets of variables, $\mathbb{T}_\Sigma(X)$ is the smallest set of Σ-*terms over* X built with operators from Σ and variables from X. If X is empty, then we write $\mathbb{T}_\Sigma$ for $\mathbb{T}_\Sigma(\emptyset)$. The set of all terms of sort s is denoted by $\mathbb{T}_{\Sigma,s}(X)$. We denote by $Var(t)$ the set of variables occurring in t. A *ground substitution* is a partial mapping from X to terms in $\mathbb{T}_\Sigma$ and it uniquely extends to a Σ-homomorphism from $\mathbb{T}_\Sigma(X)$ to $\mathbb{T}_\Sigma$. We denote by $\bar{t}^n$ a sequence of terms $t_1, \dots, t_n$, for $t_i \in \mathbb{T}_\Sigma(X)$, $i = 1, n$. Considering S as a partially ordered set leads to quite similar definitions for order-sorted signature, terms and substitutions [14].

Term Rewriting. A set R of *rewrite rules* is a set of ordered pairs of terms from $\mathbb{T}_\Sigma(X)$, denoted $u \to v$, such that u and v belong to the same sort, $u \notin X$ and $Var(v) \subseteq Var(u)$. The *rewriting relation* induced by R is denoted by $\to_R$ ($\to$ if there is no ambiguity about R), and defined by $t \to t'$ iff there exists a substitution σ and a position p in t such that $t = t[\sigma u]_p$ for some rule $u \to v$ of R, and $t' = t[\sigma v]_p$.

Membership Equational Logic (MEL). Membership equational logic [21,7] extends many-sorted equational logic with membership assertions $t : s$ stating that a term t belongs to a sort s. A *signature* in membership equation logic is a triple (K, Σ, S) (Σ for short) where K is a set of kinds, $\Sigma = \{\Sigma_{w,k}\}_{(w,k)\in K^*\times K}$ is a many-kinded signature, and $S = \{S_k\}_{k\in K}$ a K-kinded family of disjoint sets of sorts. Note that in this paper we adopt the same approach as in Maude [9] of not explicitly name the kinds. A kind is identified with its equivalence class of sorts and we usually denote it by enclosing the name of a sort in square brackets. A *MEL* Σ-*algebra* A is a many-kinded algebra having a set A_s with $A_s \subseteq A_k$ for every $s \in S_k$. We denote by $\mathbb{T}_\Sigma(X)_k$ the set of k-kinded terms built over Σ with variables from a set X. An *atomic formula* is either a Σ-equation $(\forall X)\, t = t'$ or a membership assertion $(\forall X)\, t : s$, for $t, t' \in \mathbb{T}_\Sigma(X)_k$ and $s \in S_k$. A Σ-sentence is a universally quantified Horn clause on an atomic formula F over $\mathbb{T}_\Sigma(X)$ of the form $(\forall X)\, F$ if $\bigwedge_i u_i = u'_i \wedge \bigwedge_j v_j : s_j$ with $u_i, u'_i, v_j \in \mathbb{T}_\Sigma(X)$. A MEL *theory* is a pair (Σ, E) of a MEL signature Σ and a set E of Σ-sentences.

As a shorthand, we use subsort declarations instead of giving the corresponding membership equations, as well as operation declaration at the sort level instead of the kind level, if not otherwise needed.

Example 2.1 The static description of the membranes is represented by the MEL theory (Σ_m, E_m), where Σ_m includes the sorts *Object* and *Soup* with *Object* $<$ *Soup*, $\epsilon : \longrightarrow$ *Soup*, the concatenation $\_\_ :$ *Soup Soup* $\longrightarrow$ *Soup*, together with

- the sort *RuleLabel* for representing rule labels,

$$\frac{t \in \mathbb{T}_\Sigma(X)_k}{(\forall X)\, t \to t}\ \textbf{Reflexivity} \qquad \frac{(\forall X)\, t_1 \to t_2, \quad (\forall X)\, t_2 \to t_3}{(\forall X)\, t_1 \to t_3}\ \textbf{Transitivity}$$

$$\frac{E \vdash (\forall X)\, t = u, \quad (\forall X)\, u \to u', \quad E \vdash (\forall X)\, u' = t'}{(\forall X)\, t \to t'}\ \textbf{Equality}$$

$$\frac{f \in \Sigma_{k_1 \ldots k_n, k}, t_i, t_i' \in \mathbb{T}_\Sigma(X)_{k_i}, \quad (\forall X)\, t_i \to t_i'}{(\forall X)\, f(t_1, \ldots, t_n) \to f(t_1', \ldots, t_n')}\ \textbf{Congruence}$$

$$\frac{(\forall X)\, r : u \to v \in R, \quad \theta, \theta' : X \longrightarrow \mathbb{T}_\Sigma(Y), \quad \begin{array}{c}\text{for all } x \in X \\ (\forall Y)\, \theta(x) \to \theta'(x)\end{array}}{(\forall Y)\, \theta(u) \to \theta(v)}\ \textbf{Replacement}$$

Fig. 1. Inference rules for (unconditional) MEL-based rewrite theories

- the sort *MembraneName* for representing membrane names,
- the projections $lhs : RuleLabel \longrightarrow Soup$ for the left-hand side of a rule, and $label : Rule \longrightarrow RuleLabel$ for the label of a rule.

E_m includes axioms expressing the associativity and commutativity of $\_\ \_$ with ϵ the identity element.
The static description of membrane systems is represented by the MEL theory (Σ_p, E_p) consisting of (Σ_m, E_m) together with:

- the sort *Membrane* for states of both simple and composite membranes,
- the sort *MembraneBag* for multisets of membranes, together with its constructors: the subsort relation $Membrane < MembraneBag$, the constant *NULL* denoting the empty multiset, and the union of multisets $\_,\_ : MembraneBag\ MembraneBag \longrightarrow MembraneBag$,
- the constructors for *Membrane*: $\langle \_ | \_ \rangle : MembraneName\ Soup \longrightarrow Membrane$ and $\langle \_ | \_ \{ \_ \} \rangle : MembraneName\ Soup\ MembraneBag \longrightarrow Membrane$,
- the axioms expressing the associativity and commutativity of $\_,\_$ with *NULL* the identity element.

MEL-based Rewrite Theories. A *(unconditional) MEL-based rewrite theory* [7] is a triple $\mathcal{R} = (\Sigma, E, R)$, where (Σ, E) is a MEL theory and R is a set of (universally quantified) labelled (unconditional) rewrite rules having the form $(\forall X)\, r : u \to v$, with $u, v \in \mathbb{T}_\Sigma(X)_s$ for some sort s and $Var(v) \subseteq Var(u)$. The rewriting logic (RL) of a MEL-based rewrite theory R is a sequent calculus, where a *sequent* of $\mathcal{R}$ is a pair of (universally quantified) terms written as $(\forall X)\, t \to t'$, with $t, t' \in \mathbb{T}_\Sigma(X)$. We say that $\mathcal{R}$ *entails* the sequent $(\forall X)\, t \to t'$, and write $\mathcal{R} \vdash (\forall X)\, t \to t'$, if $(\forall X)\, t \to t'$ can be derived using the inference rules in Figure 1.

The rewrite theories defined in [7] also includes frozen operators. These operators are not needed for our purpose. Therefore by a MEL-based rewrite theory we mean a rewrite theory defined as in [7] but without frozen operators.

Example 2.2 The complete description of a control-free membrane M is represented by the MEL rewrite theory $\mathcal{M} = (\Sigma_m \cup O, E_m, R)$, where (Σ_m, E_m) is the MEL theory given in Example 2.1, O the set of object constants, and R includes

the rewrite rules corresponding to the evolution rules. For the example of M_1, we have $O = \{a, b, c, d\}$ and $R = R_1$. We may use the inference rules in Figure 1 to deduce a concurrent rewriting step describing an evolution step:

$$\dfrac{\dfrac{\dfrac{r_{11} : aa \to ab}{aa \to ab}\ \textbf{Replacement} \qquad \dfrac{r_{12} : ab \to cd}{ab \to cd}\ \textbf{Replacement}}{aaaaabb \to ababcdb}\ (\textbf{Congruence\&Equality})^*}{\langle\, M_1 \mid aaaaabb \,\rangle \to \langle\, M_1 \mid ababcdb \,\rangle}$$

where by $(\textbf{Congruence\&Equality})^*$ we mean that the rules **Congruence** and **Equality** are applied of several times. Note that the evolution rules have no variables, so that we omitted to specify the set of universal quantified variables for the above rules and sequents.

3 Strategy-based Rewriting Logic for Membrane Systems

3.1 Why Membrane Systems are not Rewrite Theories

A control-free membrane system is described by the rewrite theory $\mathcal{R}_\Pi = (\Sigma_p \cup O, E_p, R)$, where (Σ_p, E_p) is the MEL theory given in Example 2.1, and R includes the rewrite rules coming from all the component membranes. We include further in this theory the following items:

- for each rule $r : u \to v$, add a constant $r : \mathit{RuleLabel}$ and an equation $\mathit{lhs}(r) = u$;
- for each membrane name M, add a constant $M : \mathit{MembraneName}$.

From now on we assume that the object constants O are included in Σ_p.

If we apply the inference rules in Figure 1 for $\mathcal{R}_\Pi$, then we might deduce transitions which are not valid for membrane systems. For instance, the concurrent rewriting obtained by the following inference

$$\dfrac{\dfrac{\dfrac{r_{11} : aa \to ab}{aa \to ab}\ \textbf{Replacement} \qquad \dfrac{r_{21} : bb \to a}{bb \to a}\ \textbf{Replacement}}{aabbb \to abab}\ (\textbf{Congruence\&Equality})^*}{\langle\, M_1 \mid aabbb\ \{\, \langle\, M_2 \mid bcc \,\rangle\, \}\, \rangle \to \langle\, M_1 \mid abab\ \{\, \langle\, M_2 \mid bcc \,\rangle\, \}\, \rangle}$$

does not describe a valid evolution step for Π_1 because it uses a rule of M_2 in order to modify the content of M_1. The evolution rules are local to a region (membrane). A rewrite theory correctly describing a rewrite theory should use the reflection property of RL in order to include meta-level information as that regarding the locality of the evolution rules. But in that case, an evolution step of a compound membrane cannot be described by a concurrent rewriting step (see [17] for more details).

3.2 Syntax of Strategy-based Rewriting Logic for Membrane Systems

In this section we present a new framework, called *strategy-based rewriting logic*, which avoids the above drawback. The main idea is to use strategies instead of the proof-terms [7] describing the inference trees corresponding to the concurrent rewriting steps. In practice, in order to apply a concurrent rewrite step in a rewrite theory $\mathcal{R}$, we need an algorithm which for a given term t, it returns a set of triples (p_i, θ_i, r_i), where $\{p_i\}$ is a set of disjoint positions, θ_i is a substitution, and r_i is a rewrite rule $r_i : u_i \to v_i$ such that $t|_{p_i} = \theta_i(u_i)$. Then a concurrent rewrite step is $t \to t'$, where t' is $t\{\theta(v_i)/p_i\}$. In strategy-based rewriting logic we need an algorithm that for a given term t, returns a strategy expression s. Then a step is $t \to t'$, where t' is obtained from t according to the the strategy s (the evolution strategy). In Subsection 3.2.1 we introduce a MEL theory STRAT(Π) defining the strategy language needed for membrane systems, and in Subsection 3.2.2 we introduce a MEL theory STRAT-CTRL(Π) defining the *strategy controllers* for membrane systems as a means to specify algorithms that compute strategies for state terms. We refer the triple $(\mathcal{R}_\Pi, \mathsf{STRAT}(\Pi), \mathsf{STRAT\text{-}CTRL}(\Pi))$ as *strategy-based rewrite theory*.

3.2.1 Strategies for Membrane Systems

We can think of a rewriting strategy as an algorithm for defining a computation step induced by a set of rules. In particular, the rewriting rules are elementary strategies. In general, a rewriting strategy language consists of expressions built using rewriting rules and strategy operators. Various approaches have been followed, yielding slightly different strategy languages such as ELAN [16,6], Stratego [23], TOM [4], or Maude [18]. All these languages provide flexible and expressive strategy languages where high-level strategies are defined by combining low level primitives, and they all share the concern to provide abstract ways to express control of rule applications, by using reflexivity and the metalevel for Maude, or the notion of rewriting strategies for ELAN or Stratego. Strategies such as bottom-up, top-down or leftmost-innermost are higher-order features that describe how rewrite rules should be applied.

In this section we introduce the MEL theory defining the syntax for the strategies defined over the rewrite theory (Σ_p, E_p, R) specifying a control-free membrane system Π. The syntax for these strategies can be split in two: a first part which is independent of the particular choice of Π (in fact of R), and a second part which describes the rules in Π as elementary strategies. We start by defining the independent part.

STRAT-BASIC is the MEL theory consisting of:

- the sort *Rule* for representing rules, the sort *Strategy* for strategies,
- the constructor for the sort *Rule*, $(\_ : \_ \to \_) : \mathit{RuleLabel}\ \mathit{State}\ \mathit{State} \longrightarrow \mathit{Rule}$,
- the subsort relation $\mathit{RuleLabel} < \mathit{Strategy}$,
- the strategy constructors for identity, failure, non-deterministic choice, and se-

quential composition respectively:

$$\begin{aligned} id\ \ fail : &\longrightarrow Strategy \\ \_ + \_ : Strategy\ Strategy &\longrightarrow Strategy\ [\texttt{assoc comm}] \\ \_;\_ : Strategy\ Strategy &\longrightarrow Strategy\ [\texttt{assoc id :}\ id] \end{aligned}$$

- a congruence strategy operator for each of the constructors of *Soup*, *Membrane* and *MembraneBag*:

$$\begin{aligned} \_\ \_ : Strategy\ Strategy &\longrightarrow Strategy\ [\texttt{assoc comm}] \\ \langle \_ | \_ \rangle : MembraneName\ Strategy &\rightarrow Strategy \\ \langle \_ | \_ \{ \_ \} \rangle : MembraneName\ Strategy\ Strategy &\longrightarrow Strategy \\ \_,\_ : Strategy\ Strategy &\longrightarrow Strategy\ [\texttt{assoc comm}] \end{aligned}$$

- the equations

$$\begin{array}{ll} s + s = s & id\ id = id \\ s; fail = fail; s = fail & s + fail = s \\ \langle M \mid s + s' \rangle = \langle M \mid s \rangle + \langle M \mid s' \rangle & s\ fail = fail\ s = fail \\ \langle M \mid s_1 + s_1'\{s_2\} \rangle = \langle M \mid s_1\{s_2\} \rangle + \langle M \mid s_1'\{s_2\} \rangle & s_1\,(s_2 + s_2') = s_1\,s_2 + s_1\,s_2' \\ \langle M \mid s_1\{s_2 + s_2'\} \rangle = \langle M \mid s_1\{s_2\} \rangle + \langle M \mid s_1\{s_2'\} \rangle & s_1, (s_2 + s_2') = s_1, s_2 + s_1, s_2' \end{array}$$

The strategy language defined by the above theory corresponds to the membrane systems we consider in this paper. It includes the identity strategy (id), the failure strategy ($fail$), non-deterministic choice ($\_+\_$), sequential composition ($\_;\_$), and the strategy congruence operators corresponding to the constructors for *Soup*, *Membrane*, and *MembraneBag*. This language can be enriched with new constructs needed for defining other control mechanisms [1].

The strategy language corresponding to a rewrite theory representing a control-free membrane system is obtained by adding to the basic strategies the elementary strategies defined by the rules: STRAT(Π) contains the MEL theory STRAT-BASIC together with a membership axiom $r : RuleLabel$ and an equation axiom $lhs(r) = u$, for each rule $r : u \rightarrow v$ in R.

3.2.2 Strategy Controllers for Membrane Systems

Strategy controllers generalize strategies in the sense that they supply a very general solution for describing algorithms that computes evolution strategies. Such a strategy is computed in each state term t and it should describe all evolution steps possible from t. It should be equal to *fail* if and only if no evolution step is possible from t.

In order to be able to represents the control mechanisms of membranes, we might need to enrich the algebraic structure (Σ_m, E_m) for membranes with special items. For instance, if the control is the maximal parallel rewriting with priorities, then a binary operator $\_>\_$ is introduced for the priority relation, and, for each pair in the priority relation $r > r'$, an equation $r > r' = true$ is added.

The MEL theory STRAT-CTRL describes the strategy controllers for the membrane systems and it consists of:

- the MEL theory STRAT-BASIC given in Sect. 3.2.1;
- a sort *StrategyController* for strategy controllers together a constant for each membrane control mechanism like *mpr*, *pri* and so on, together with a constant *evrl* corresponding to the application of the evolution rules;
- an operation $getCtrl : MembraneName \longrightarrow StrategyController$ which returns the strategy controller of a given membrane.

3.3 Semantics of Strategy-based Rewriting Logic for Membrane Systems

The semantics is given by a MEL theory $Proof(\Pi)$ which includes the semantics of the strategies and the semantics of the strategy controllers.

3.3.1 Semantics of Strategies

The definition of the semantics for strategies can be given in various ways, see, e.g., [24,23,6,5,18,11]. We proceed here on a slightly different approach and give the semantics of the strategies by means of a MEL theory. This helps us later when we define the algebraic semantics of the membrane systems.

The MEL theory STRAT-BASIC-SEM which includes STRAT-BASIC, a sort *StratSequent* for *strategic sequents*, the constructor for strategic sequents:

$$(\_ \triangleright \_ \rightarrow \_) : Strategy\ State\ State \longrightarrow [StratSequent],$$

a sort *State*, the subsort relations $Soup\ Membrane\ MembraneBag < State$, the following membership axioms:

$$(r \triangleright u \rightarrow v) : StratSequent \ \texttt{if}\ r : RuleLabel \wedge (r : u \rightarrow v) : Rule \tag{1}$$

$$id \triangleright t \rightarrow t' : StratSequent \ \texttt{if}\ E_p \vdash t = t' \tag{2}$$

$$s_1 + s_2 \triangleright t \rightarrow t' : StratSequent \tag{3}$$
$$\texttt{if}\ s_1 \triangleright t \rightarrow t' : StratSequent \vee s_2 \triangleright t \rightarrow t' : StratSequent$$

$$s_1; s_2 \triangleright t \rightarrow t' : StratSequent \tag{4}$$
$$\texttt{if}\ s_1 \triangleright t \rightarrow t'' : StratSequent \wedge s_2 \triangleright t'' \rightarrow t' : StratSequent$$

$$s_1\, s_2 \triangleright t_1\, t_2 \rightarrow t_1'\, t_2' : StratSequent \tag{5}$$
$$\texttt{if}\ s_1 \triangleright t_1 \rightarrow t_1' : StratSequent \wedge s_1 \triangleright t_1 \rightarrow t_1' : StratSequent$$

$$\langle M \mid s \rangle \triangleright \langle M \mid t \rangle \rightarrow \langle M \mid t' \rangle : StratSequent \ \texttt{if}\ s \triangleright t \rightarrow t' : StratSequent \tag{6}$$

$$\langle\, M \mid s_1\, \{\, s_2 \,\}\, \rangle \triangleright \langle\, M \mid t_1\, \{\, t_2 \,\}\, \rangle \rightarrow \langle\, M \mid t_1'\, \{\, t_2' \,\}\, \rangle : StratSequent \tag{7}$$
$$\texttt{if}\ s_1 \triangleright t_1 \rightarrow t_1' : StratSequent \wedge s_2 \triangleright t_2 \rightarrow t_2' : StratSequent$$

$$s_1, s_2 \triangleright t_1, t_2 \rightarrow t_1', t_2' : StratSequent \tag{8}$$
$$\texttt{if}\ s_1 \triangleright t_1 \rightarrow t_1' : StratSequent \wedge s_1 \triangleright t_1 \rightarrow t_1' : StratSequent$$

If Π is a membrane system, then STRAT-SEM(Π) is the MEL theory STRAT-BASIC-SEM together with a membership axiom $(r : u \rightarrow v) : Rule$, for each rule $r : u \rightarrow v$ in R.

The axiom 1 defines the semantics of a rule as the only sequent it defines, i.e., the unique one-step rewriting obtained by applying the rule at the top. Recall that the evolution rules have no variables. The axioms 2-4 define the semantics for identity, non-deterministic choice, and sequential composition, respectively, according to their intuitive definitions. The semantics of the congruence strategy operator given by axiom 5 consists of the parallel application of the rewrites given by the arguments. Recall that the operator _ _ is associative and commutative and hence we can have an arbitrary number of parallel rewrites. Axioms 6-8 give semantics for the congruence strategy operators corresponding to the constructors of membranes (including systems).

Remark 3.1 The theory STRAT-BASIC-SEM has only a theoretical value. In practice, systems like Maude are not able to handle the axiom [4]. See, e.g., [5,18,19] for an executable semantics of these strategies. In terms of [11], STRAT-BASIC-SEM can be defined as a parameterized strategy module and STRAT-SEM(Π) as an instance of it.

The next result shows that the equations included in STRAT-BASIC are sound w.r.t. the above semantics.

Theorem 3.2 *If*

$$\mathsf{STRAT}(\Pi) \models s = s' \text{ and } \mathsf{STRAT}-\mathsf{SEM}(\Pi) \models s \triangleright t_1 \to t_2 : \mathit{StratSequent},$$

then there are t_1', t_2' *such that*

$$\mathsf{STRAT}-\mathsf{SEM}(\Pi) \models s' \triangleright t_1' \to t_2' : \mathit{StratSequent},$$
$$(\Sigma_p, E_p) \models t_1 = t_1' \text{ and } (\Sigma_p, E_p) \models t_2 = t_2'.$$

Proof (Sketch) We proceed by structural induction on $s = s'$ by showing only on a few cases, since the others are handled similarly.

Case $s + s = s$**.** By definition, $s + s \triangleright t_1 \to t_2 : \mathit{StratSequent}$ implies
$s \triangleright t_1 \to t_2 : \mathit{StratSequent} \vee s \triangleright t_1 \to t_2 : \mathit{StratSequent}$,
which is equivalent to $s \triangleright t_1 \to t_2 : \mathit{StratSequent}$. Obviously, $(\Sigma_p, E_p) \models t_1 = t_1$ and $(\Sigma_p, E_p) \models t_2 = t_2$.

Case $\langle M \mid s + s' \rangle = \langle M \mid s \rangle + \langle M \mid s' \rangle$. $\langle M \mid s + s' \rangle \triangleright t_1 \to t_2 : \mathit{StratSequent}$ implies by axiom 6 the existence of t_1', t_2' such that $(\Sigma_p, E_p) \models \langle M \mid t_1' \rangle = t_1$ and $(\Sigma_p, E_p) \models \langle M \mid t_2' \rangle = t_2$, and $s + s' \triangleright t_1' \to t_2' : \mathit{StratSequent}$. By axiom 3 we have ($s \triangleright t_1' \to t_2' : \mathit{StratSequent} \vee s' \triangleright t_1' \to t_2' : \mathit{StratSequent}$); then, by axiom 6, $\langle M \mid s \rangle \triangleright \langle M \mid t_1' \rangle \to \langle M \mid t_2' \rangle : \mathit{StratSequent} \vee \langle M \mid s' \rangle \triangleright \langle M \mid t_1' \rangle \to \langle M \mid t_2' \rangle : \mathit{StratSequent}$, and by axiom 3, we obtain $\langle M \mid s \rangle + \langle M \mid s' \rangle \triangleright \langle M \mid t_1' \rangle \to \langle M \mid t_2' \rangle : \mathit{StratSequent}$.

□

3.3.2 Semantics of Strategy Controllers

The semantics of the strategy controllers for membrane systems is given by a MEL theory STRAT-CTRL-SEM which includes STRAT-CTRL, $\mathcal{R}(\Pi)$, together with:

- a sort *RuleSet* for representing sets of rules;

- an operation $getRules : MembraneName \rightarrow RuleSet$, which returns the set of rules for a given membrane name;
- a subsort relation $Rule < RuleSet$ and the usual set-like constructors for the sort $RuleSet$, $none$ for the empty set and $\_,\_$ for the union;
- an overloaded operation
  $getStrat : StrategyController\ State \longrightarrow Strategy$
  $getStrat : MembraneName\ StrategyController\ State \longrightarrow Strategy$
  which returns the strategy instance of (a given strategy controller for) a given term representing a (membrane) state;
- a set of operations and equations giving semantics of the strategy controllers like mpr, pri (see below);
- a set of equations giving the semantics of the strategy controller $evrl$:
  $getStrat(evrl, \langle\ M \mid w\ \rangle) = \langle\ M \mid getStrat(M, getCtrl(M), w)\ \rangle$
  $getStrat(evrl, \langle\ M \mid w\ \{\ t_1, \ldots, t_n\ \}\ \rangle) =$
  $\langle\ M \mid getStrat(M, getCtrl(M), w)\ \{\ getStrat(evrl, t_1), \ldots, getStrat(evrl, t_n)\ \}\ \rangle$

In what follows we present the equational definitions for mpr and pri controls. The definition of the membrane control mpr is given as follows:

- an overloaded strategy controller defined by $mpr : \longrightarrow StrategyController$ and $mpr : RuleSet \longrightarrow StrategyController$, where the unary operation takes as parameter the set of rules a strategy controller must be built from.
- the equational definition of the binary operation $getStrat$ for mpr is given using an auxiliary operation
  $mpr : RuleSet\ Soup\ RuleSet\ Strategy \longrightarrow StrategyController$
  together with the following equations:
  $getStrat(M, mpr, w) = getStrat(mpr(getRules(M)), w)$
  $getStrat(mpr(RS), w) = getStrat(mpr(RS, w, none, fail), w)$
  $getStrat(M, mpr((RS, r), w, RS', S), w) =$
  $(r\ getStrat(M, mpr((RS, r), w', none, id))) + getStrat(mpr((RS, r), w, (RS', r), fail), w)$
  $\texttt{if}\ lhs(r)w' := w \wedge notIn(r, RS')$
  $M, getStrat(mpr(RS, w, RS', S), w) = S[\texttt{owise}]$

The third argument of the quaternary operation mpr represents the set of rules already used for the construction of a strategy applicable to w; such iterative application allows to equally consider each of the rules given in the first argument for computing a strategy. As soon as a rule is tested as applicable and is used in a concatenation for building a strategy, then the fourth argument of mpr becomes id to mark the success. When the first two equations can no longer be applied, the third one is applied: if last argument of mpr is $fail$, this means no rule was applicable, and the strategy computed is $fail$; otherwise, the strategy id is concatenated to the multiset of rules already found as applicable.

In the definition of an operator using conditional equations, the cases when none of the equations can be applied are accumulated by an unconditional equation with

the attribute "`owise`" (otherwise).

The definition of the membrane control *pri* uses the definition of *mpr* and the following items:

- an operator for defining a (partial) order on rule labels corresponding to a priority relation on rule applications, $\_ > \_ : \mathit{RuleLabel}\ \mathit{RuleLabel} \longrightarrow \mathit{Bool}$;
- an operation $\mathit{filter} : \mathit{RuleSet}\ \mathit{Soup} \longrightarrow \mathit{RuleSet}$ that discards from a set of rules RS all rules either not applicable on a given soup or of a lower priority than an applicable rule r:

$$\begin{aligned} \mathit{filter}((RS, r, r'), w) &= \mathit{filter}((RS, r), w) \\ &\qquad \texttt{if}\ \ \mathit{lhs}(r)w' := w \wedge \mathit{label}(r) > \mathit{label}(r') \\ \mathit{filter}((RS, r), w) &= \mathit{filter}(RS, w)\ \ \texttt{if}\ \ \mathit{notIn}(\mathit{lhs}(r), w) \\ \mathit{filter}(RS, w) &= RS\ [\texttt{owise}] \end{aligned}$$

  where $\mathit{notIn}(u, w)$ returns *true* if and only if u is not a sub-multiset of w;
- an equation defining *getStrat* for *pri* that proceeds as for *mpr* but with a filtered set of rules:

$$\mathit{getStrat}(M, \mathit{pri}, w) = \mathit{getStrat}(\mathit{mpr}(\mathit{filter}(\mathit{getRules}(M), w)), w)$$

Remark 3.3 The mechanism given by the *filter* function can be also used for other mechanisms as promoters and inhibitors [22], minimal parallel rewriting [8] and so on.

Theorem 3.4 STRAT-CTRL-SEM *is terminating and confluent.*

Proof (Sketch) We can define a reduction order based on the following:

- the operation *filter* reduces the number of rules in the first argument at each recursive call;
- for the equations defining *getStrat*, either:
  - the operator is propagated in the term (hence the size as number of operations of the second argument decreases in the right-hand side), or,
  - in the conditional equation, either:
    - the size of the soup of objects decreases or
    - the size of the difference between the rules taken as first argument by *mpr* and the rules already used taken as the third argument decreases.

STRAT-CTRL-SEM has no critical pairs, and since it is terminating, it follows that it is confluent. □

The above theorem shows that for each state t and appropriate strategy controller *ctrl*, $\mathit{getStrat}(\mathit{ctrl}, t)$ returns a unique strategy. If from t more than one evolution step are possible, then these are given by a sum of strategies. If no evolution step is possible from t, then $\mathit{getStrat}(\mathit{ctrl}(RS), t)$ returns *fail*. Since this strategy describes the possible evolutions from the current state, it is called *evolution strategy*.

Lemma 3.5 *Given a membrane M with the strategy controller $ctrl \in \{mpr, pri\}$ and a soup w, then $getStrat(M, ctrl, w)$ reduces either to fail or to a strategy of the form $s_1 + \ldots + s_n$, $n \geq 1$, where each s_i is a concatenation of rules from R.*

Proof (Sketch) Only the concatenation congruence operator and the sum are used in the definition of $getStrat(M, ctrl, w)$. The conclusion follows applying the axiom $s\,(s_1 + s_2) = s\,s_1 + s\,s_2$. □

Theorem 3.6 *Given a state term t, then $getStrat(t)$ reduces either to fail or to a strategy of the form $s_1 + \ldots + s_n$, $n \geq 1$, where s_i corresponds to an evolution step.*

Proof (Sketch) By previous lemma and using the other axioms included in STRAT-BASIC. □

Remark 3.7 It is worth noting that the sequential composition operator ; is not used. This is due to the fact we did not consider yet the case of communication and structural actions for membrane systems (see Section 5).

Remark 3.8 For s a strategy, we write $s_i <: s$ whenever s can be written as a non-deterministic choice between the strategies $s_1, \ldots, s_n$, $n \geq 1$,with $s_j \neq id$ and $s_j \neq fail$, for all $j = 1, \ldots, n$. We note that we can express $s <: s'$, for any two strategies s, s', as a MEL formula. In particular, in the context of Lemma 3.5, we have $s_i <: getStrat(M, ctrl, w)$ for all $i = 1, \ldots, n$.

In an evolution step for a membrane system Π, the strategy controller for a membrane that does not evolve is reduced to the strategy *fail*. If for all membranes in the system the results of computing their strategy controllers for a given state are equal to *fail*, then the strategy controller of the entire system reduces to *fail*. Otherwise, since the membranes that can evolve must evolve despite others that cannot, all *fail* strategies are replaced by *id*.

Example 3.9 Let us consider the membrane system Π_1 introduced in Section 1 with the structure $\langle M_1 \mid \{M_2\}\rangle$. By definition, $getStrat(M_1, pri, aabb)$ reduces to $r_{11}\,id + r_{12}\,r_{12}$ which will rewrite $aabb$ into $abbb$ or $ccdd$, and $getStrat(M_2, mpr, aabb)$ reduces to $r_{22}\,r_{22}\,r_{21}$ which rewrites $aabb$ into $ababa = aaabb$. Then

$$getStrat(evrl, \langle\, M_1 \mid aabb\, \{\, \langle\, M_2 \mid aabb \,\rangle\, \}\,\rangle)$$

reduces to

$$\langle\, M_1 \mid getStrat(M_1, pri, aabb)\, \{\, getStrat(M_2, mpr, aabb)\, \}\,\rangle$$

which reduces to

$$\langle\, M_1 \mid r_1\, id + r_2\, r_2\, \{\, \langle\, M_2 \mid r_{22}\, r_{22}\, r_{21} \,\rangle\, \}\,\rangle$$

which is equal to

$$\langle\, M_1 \mid r_1\, id\, \{\, \langle\, M_2 \mid r_{22}\, r_{22}\, r_{21} \,\rangle\, \}\,\rangle + \langle\, M_1 \mid r_2\, r_2\, \{\, \langle\, M_2 \mid r_{22}\, r_{22}\, r_{21} \,\rangle\, \}\,\rangle.$$

Hence, the states which can be obtained from $\langle\, M_1 \mid aabb\, \{\, \langle\, M_2 \mid aabb \,\rangle\, \}\,\rangle$ in one

evolution step are:

$$\langle\, M_1 \mid abbb\, \{\, \langle\, M_2 \mid aaabb \,\rangle\, \}\, \rangle$$
$$\langle\, M_1 \mid ccdd\, \{\, \langle\, M_2 \mid aaabb \,\rangle\, \}\, \rangle$$

3.3.3 The theory $\mathcal{P}roof(\Pi)$

Let Π be a membrane system. The MEL theory $\mathcal{P}roof(\Pi)$ includes STRAT-SEM(Π), STRAT-CTRL-SEM, equations defining $getRules(M)$ for each membrane M in Π, a sort $EvStepSequent$, an operation $\_ \rightarrow_{evStep} \_ : State\ State \longrightarrow [EvStepSequent]$ for evolution steps sequents, and the membership axiom

$$(t \rightarrow_{evStep} t') : EvStepSequent$$
$$\textbf{if}\ s := getStrat(evrl, t)\ \wedge\ s' <: s\ \wedge\ s' \triangleright t \rightarrow t' : StratSequent$$

The matching equation $s := getStrat(evrl, t)$ assigns to variable s the result of the evaluation of $getStrat(\texttt{ctrl}(\Pi), t)$. The membership predicate $s' \triangleright t \rightarrow t' : StratSequent$ is equivalent with checking if there is a computation among all computed by $getStrat$ which rewrites t into t'. Note that $s' \triangleright t \rightarrow t' : StratSequent$ implies $s' : Strategy$, i.e., s' is indeed a strategy, and, moreover, a non-failing strategy since a failure does not correspond to an evolution step of the system.

In Figure 2 we give the diagram describing the construction of the theory $\mathcal{P}roof(\Pi)$.

Since the proof calculus is described as a MEL theory, we get for free the algebraic semantics for the strategic-based rewrite theories.

Definition 3.10 An *algebraic model* for Π is a $\mathcal{P}roof(\Pi)$-algebra.

$\mathcal{P}roof(\Pi)$ is a MEL theory, hence it admits initial model [21], denoted by $\mathbb{T}_{\mathcal{P}roof(\Pi)}$.

Theorem 3.11 *Let Π be a membrane system.*

$$\mathcal{P}roof(\Pi) \vdash (t \rightarrow_{evStep} t') : EvStepSequent \iff$$
$$\mathcal{P}roof(\Pi) \models (t \rightarrow_{evStep} t') : EvStepSequent \iff$$
$$[(t \rightarrow_{evStep} t')] \in \mathbb{T}_{\mathcal{P}roof(\Pi), EvStepSequent}.$$

Theorem 3.12 *Let Π be a membrane system and let t, t' be two states. Then $t \Rightarrow t'$ if and only if $\mathcal{P}roof(\Pi) \models (t \rightarrow_{evStep} t') : EvStepSequent$.*

Proof (Sketch) We give the proof for a simple membrane with R the sets of rules and mpr the strategy controller. Then the theorem reduces to:
$w \overset{mpr}{\Longrightarrow} w'$ iff $s := getStrat(M, mpr, w) \wedge s' <: s \wedge s' \triangleright w \rightarrow w' : StratSequent$.
The case of pri as strategy controller is similar, apart from an initial step when the set of applicable rules w.r.t. the priority relation is computed using the operation $filter$.

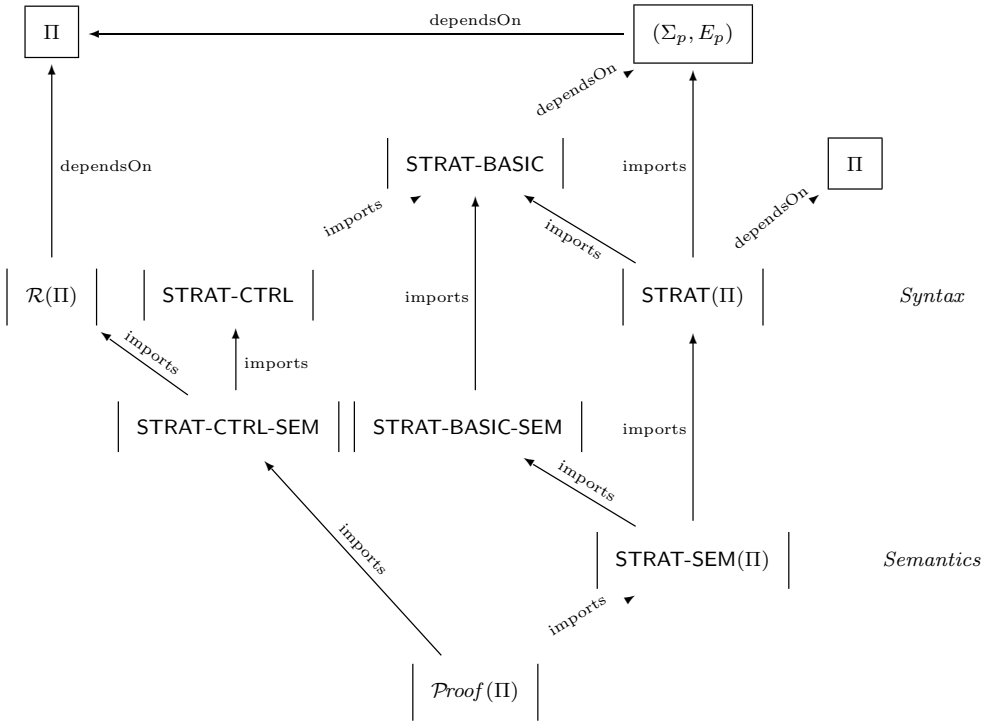

Fig. 2. The construction of the theory $\mathcal{P}roof(\Pi)$

"⇒". Since $w \xrightarrow{mpr} w'$, then, by definition, there is a multiset of rules from $R = getRules(M)$, $r_i : u_i \to v_i$, for $i = 1, \ldots, n$, such that $w = u_1 \ldots u_n z$ and $w' = v_1 \ldots v_n z$ with z irreducible. Then, by Lemma 3.5, $getStrat(M, mpr, w)$ reduces to a strategy equivalent to $r_1 \ldots r_n \, \mathtt{id} + s'$ or $r_1 \ldots r_n \, \mathtt{id}$. By definition, $r_i \triangleright u_i \to v_i : StratSequent$, for $i = 1, \ldots, n$, hence $r_1 \ldots r_n \, \mathtt{id} \triangleright w \to w' : StratSequent$.

"⇐". If $s := getStrat(M, mpr, w)$ with $s' <: s$ and $s' \triangleright w \to w' : StratSequent$, then from the definition of *getStrat* it follows that s' is equivalent to a concatenation of rules $r_1 \ldots r_n \, \mathtt{id}$, with $r_i : u_i \to v_i$, for all $i = 1, \ldots, n$, such that there is a multiset z of objects irreducible by any r_i and $(\Sigma_p, E_p) \models w = u_1 \ldots u_n z$, $(\Sigma_p, E_p) \models w' = v_1 \ldots v_n z$. In other words, $w \xrightarrow{mpr} w'$.

The theorem is then easily proved by using structural induction on the membranes. □

The properties proved here are valid for membrane systems having *mpr* and/or *pri* strategy controllers. Adding new strategy controllers may require reviewing the previous properties and their proofs.

4 Implementation

One of the best candidate for implementing membrane systems, using the approach presented in this paper, is Maude system [9]. This is mainly motivated by the following two reasons: Maude already has implemented a strategy language and Maude language is reflective allowing us to work with operators at metalevel, as it is the case of *getRules*. A prototype of this implementation is already working. The development of this application uses Maude patterns described in [15].

A membrane is specified as a module having a syntax as follows:

```
membrane M1 is
  ev r1 : a a -> a b .
  ev r2 : a b -> c d .
  ev r3 :   b -> d d .
  pr r2 > r3 .
end
```

This module is compiled into a Maude system module according to Section 2. Since `M1` includes a partial order over rules, we can deduce that its control mechanism is *pri*. In general, we assume that the description of a membrane includes enough information such that *getCtrl*() can compute the right controller by analyzing it. Having introduced a set of membranes, these can be combined for constructing systems with different structures. Here is a dialog with the application where all the possible next one step evolutions for the current state $\langle M_1 \,|\, aabb \{ \langle M_2 \,|\, aabb \rangle \} \rangle$ are computed:

```
Maude> (transAll < M1 | a a b b { < M2 | a a b b > } > .)
Solution 1 :    < M1 | a b b b{< M2 | a a a b b >}>
Solution 2 :    < M1 | c c d d{< M2 | a a a b b >}>
```

The execution of this command is performed in three stages:

(i) First the module Π is computed on-the-fly from the configuration given as parameter of the command.

(ii) Then the term $getStrat(\langle\, M_1 \mid aabb \,\{\, \langle\, M_2 \mid aabb \,\rangle\, \}\, \rangle$ is reduced in order to compute the instance strategy corresponding to the state parameter. The result is similar to the strategy expression given in Example 3.9.

(iii) Finally, the operation *procSrewAll* from the API implementing Maude Strategy Language [19] is used to compute all t such that

$$\mathcal{P}roof\,(\Pi) \models \langle\, M_1 \mid aabb \,\{\, \langle\, M_2 \mid aabb \,\rangle\, \}\, \rangle \rightarrow_{evStep} t : EvStepSequent.$$

Since the syntax of Maude Strategy Language is different from that used in this paper, we have to encode the congruence strategy operators in this language. We also note that the execution of the *getStrat* operation produces a sequence of the corresponding API operations calls instead of the strategy expression. Here is a fragment of the sequence corresponding to the above example:

```
matchrew('<_|_'{_'}>['M1.MembraneName, 'X:Soup,
                     '<_|_>['M2.MembraneName, 'Y:Soup]],
         'true.Bool = 'true.Bool,
        ('X:Soup using or(matchrew('__[LHS,'W1:Soup],
                                   'true.Bool = 'true.Bool,
                                   (LHS using 'r1[none]) ...
```

where LHS := lhs(r1). Note that using direct the left-hand side of the rule instead of a generic variable 'W0:Soup, the execution time of the engine applying the strategies is substantially reduced. Moreover, using the axioms from STRAT-BASIC, written now as

```
eq matchrew('__[T1, T2], EC, ((T1 using S1),
                              (T2 using matchrew(T3, EC, TSL))))
   =
   matchrew('__[T1, T3], EC, ((T1 using S1), TSL)) .
```

the evolution rules can be applied in parallel.

The current version of the prototype[4] is not a complete system for handling and analyzing membrane systems. Only the control mechanisms *mpr* and *pri* are implemented. The main goal was to demonstrate that the strategy controllers are appropriate for describing the behavior of the membrane systems in terms of the strategy-based rewrite theory. We claim that the other features of the membrane systems can be easily added to the actual prototype.

5 Communication and Dissolving

The communicating membranes uses cooperative evolution rules, which have the form $r : u \to v$, with u a non-empty multiset over O, v a multiset over $O \cup Tar$, where $Tar = \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$. The elements of Tar are called *target indications* and have the following meaning: an object having associated the indication *here* remains in the same region, one having associated the indication in_M goes immediately into the directly lower membrane M, and *out* indicates that the object has to exit the membrane, thus becoming an element of the region surrounding it. The description of these membrane can be done very easily in strategy-based rewriting logic. We consider a new sort *HotSoup* with $Soup < HotSoup$ and $(w, tar) : HotSoup$ if $w : Soup$ and $tar \in Tar$. The communication between membranes is realized by the means of the following two rules:

$$\begin{aligned} in(M, M') : \langle\, M \mid w_1(w_2, in_{M'})\, \{\, \langle\, M' \mid w'\, \{\, X \,\}\, \rangle, Y \,\}\, \rangle \to \\ \langle\, M \mid w_1\, \{\, \langle\, M' \mid w' w_2\, \{\, X \,\}\, \rangle, Y \,\}\, \rangle \\ out(M', M) : \langle\, M \mid w\, \{\, \langle\, M' \mid w'_1(w'_2, out)\, \{\, X \,\}\, \rangle, Y \,\}\, \rangle \to \\ \langle\, M \mid w w'_2\, \{\, \langle\, M' \mid w'_1\, \{\, X \,\}\, \rangle, Y \,\}\, \rangle \end{aligned}$$

A rule of the form in-out, which simultaneously exchange messages between two membranes, can also be considered. Then we consider a strategy controller *comm* (constant of sort *StrategyController*) and we have to add a set of equations defining $getStrat(comm, t)$, which should return a strategy applying the rewrite rules *in* and *out* in a maximal parallel way. Even if the algorithms returning such a strategy is challenging, it is out the goal of this paper. The actual version of the prototype defines $getStrat(comm, t)$ as $(in + out)!$, where $s!$ means "repeat s as much as

[4] The prototype can be downloaded from http://thor.info.uaic.ro/~rewps/.

possible" [18]. An evolution step is described by the strategy

$$getStrat(evrl, t); getStrat(comm, t')$$

where t' is a state satisfying

$$\mathsf{STRAT\text{-}SEM}(\Pi) \vdash (getStrat(evrl, t) \triangleright t \to t') : StratSequent.$$

The dissolving of a membrane is obtained by the means of a special object δ, called *dissolving action*, and rules of the form $r : u \to v\delta$. If at least one of the rules introduces the dissolving action δ, then the membrane is dissolved, and its content becomes part of the immediately upper membrane, provided that this membrane was not dissolved at the same time, a case where we stop in the first upper membrane which was not dissolved (at least the skin remains intact). The rules of the dissolved membranes are lost. We may describe the dissolving action by the rewriting rule

$$diss(M', M) : \langle\, M \mid w\, \{\, \langle\, M' \mid w'\delta\, \{\, X \,\}\, \rangle, Y \,\}\, \rangle \to \langle\, M \mid ww'\, \{\, X, Y \,\}\, \rangle$$

Besides of this rule, a *StrategyController* constant *diss* and a set of equations defining $getStrat(diss, t)$ it is all we have to add to our strategy-based rewrite theory. In the actual version of the prototype, $getStrat(diss, t)$ returns $diss!$. An evolution step is completely described now by the strategy

$$getStrat(evrl, t); getStrat(comm, t'); getStrat(diss, t'')$$

where t', t'' are two terms satisfying

$$\mathsf{STRAT\text{-}SEM}(\Pi) \vdash (getStrat(evrl, t) \triangleright t \to t') : StratSequent$$

and

$$\mathsf{STRAT\text{-}SEM}(\Pi) \vdash (getStrat(comm, t') \triangleright t' \to t'') : StratSequent.$$

6 Related Work and Perspectives

The construction method of sequents-based theory for membrane systems is similar to the one used for the theory of proof terms $\mathcal{P}roof(\mathcal{R})$ given in [7], and to the one used for defining rewrite theories for strategies given in [5]. In the later work, the strategy language used is the one defined for ELAN, and it is larger than the one we used and needed in this paper. However, the framework we introduced here permits to extend easily the strategy language without changing the basic results.

In [6] the authors present a method of iteratively constructing a tower of strategies starting on the first level with primal strategies (rewrite rules, identity, concatenation and congruence operations), then the second level corresponds to the elementary strategies (selection operations), and the third level corresponds to user-defined strategies, such that each level is based on the previous one, except for the first level. Our work on constructing a hierarchy of MEL theories is somehow based

on the same principle as in [6]. However, strategy controls are more than complex strategies since they cannot be defined using only elementary strategies. They are generic in the sense of being parameterized on the term to rewrite.

A related notion is that of parameterized strategy defined in [11], referring to generic strategies like map and backtracking. We adopted a different meaning for strategy controls, in the sense that they are instantiated on a state term producing an effective strategy.

The construction of the theory $\mathcal{P}roof(\Pi)$ for a membrane system Π gives the theoretical foundations for a rewrite semantics for Π. Based on this foundations, we developed an implementation of the membrane systems using the Maude strategy language [18] and the patterns for Maude Metalanguage Applications described in [15]. In the future we intend to extend the approach to a larger class of systems whose behavior is similar to that of membrane systems.

In this paper we do not concern with defining new control mechanisms for membrane systems or to investigate the role of such mechanism in reasoning about biological systems. However, reasoning at the level of strategies of computing, rather than at the rule level, is an incentive direction in formally studying and analyzing biological systems and we consider it an open question for future work. Our goal was to study ways of defining a framework based on Rewriting Logic suitable for describing and analyzing membrane systems. The concept of strategy controllers proves to be a powerful tool, yet not very difficult to understand and handle, for defining various control mechanism, and we illustrate it in this paper on two such mechanisms.

References

[1] Andrei, O., G. Ciobanu and D. Lucanu, *Expressing Control Mechanisms in P systems by Rewriting Strategies*, in: H. J. Hoogeboom, G. Paun, G. Rozenberg and A. Salomaa, editors, *Workshop on Membrane Computing*, Lecture Notes in Computer Science **4361** (2006), pp. 154–169.

[2] Andrei, O., G. Ciobanu and D. Lucanu, *A rewriting logic framework for operational semantics of membrane systems*, Theoretical Computer Science **373** (2007), pp. 163 – 181.

[3] Baader, F. and T. Nipkow, "Term Rewriting and All That." Cambridge University Press, 1998.

[4] Balland, E., P. Brauner, R. Kopetz, P.-E. Moreau and A. Reilles, *Tom: Piggybacking Rewriting on Java*, in: F. Baader, editor, *RTA*, Lecture Notes in Computer Science **4533** (2007), pp. 36–47.

[5] Borovanský, P., C. Kirchner, H. Kirchner and P.-E. Moreau, *ELAN from a Rewriting Logic Point of View*, Theoretical Computer Science **285** (2002), pp. 155–185.

[6] Borovanský, P., C. Kirchner, H. Kirchner and C. Ringeissen, *Rewriting with Strategies in ELAN: A Functional Semantics*, Int. J. Found. Comput. Sci. **12** (2001), pp. 69–95.

[7] Bruni, R. and J. Meseguer, *Semantic foundations for generalized rewrite theories*, Theoretical Computer Science **360** (2006), pp. 386–414.

[8] Ciobanu, G., L. Pan, G. Paun and M. J. Pérez-Jiménez, *P systems with minimal parallelism*, Theoretical Computer Science **378** (2007), pp. 117–130.

[9] Clavel, M., F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer and C. L. Talcott, "All About Maude - A High-Performance Logical Framework: How to Specify, Program, and Verify Systems in Rewriting Logic," Lecture Notes in Computer Science **4350**, Springer, 2007.

[10] Dang, Z. and O. Ibarra, *On P systems operating in sequential mode*, in: *Pre-proceedings of DCFS Workshop, London, Ontario*, 2004, pp. 164–177.

[11] Eker, S., N. Martí-Oliet, J. Meseguer and A. Verdejo, *Deduction, Strategies, and Rewriting*, Electronic Notes in Theoretical Computer Science **174** (2007), pp. 3–25.

[12] Eker, S., J. Meseguer and A. Sridharanarayanan, *The Maude LTL Model Checker and Its Implementation*, in: T. Ball and S. K. Rajamani, editors, *SPIN*, Lecture Notes in Computer Science **2648** (2003), pp. 230–234.

[13] Freund, R., *Asynchronous P Systems and P Systems Working in the Sequential Mode*, in: G. Mauri, G. Paun, M. J. Pérez-Jiménez, G. Rozenberg and A. Salomaa, editors, *Workshop on Membrane Computing 2004*, Lecture Notes in Computer Science **3365** (2005), pp. 36–62.

[14] Goguen, J. A. and J. Meseguer, *Order-Sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations*, Theoretical Computer Science **105** (1992), pp. 217–273.

[15] Goriac, E., G. Caltais, D. Lucanu, O. Andrei and G. Grigoraş, *Patterns for Maude Metalanguage Applications*, in: *Proceedings of WRLA*, 2008.

[16] Kirchner, C., H. Kirchner and M. Vittek, *Designing Constraint Logic Programming Languages using Computational Systems*, in: P. Van Hentenryck and V. Saraswat, editors, *Principles and Practice of Constraint Programming. The Newport Papers.*, MIT Press, 1995 pp. 131–158.

[17] Lucanu, D., *Rewriting Logic-based Semantics of Membrane Systems and the Maximal Concurrency*, in: *Proceedings of Prague International Workshop on Membrane Computing* (2008), pp. 23–34.

[18] Martí-Oliet, N., J. Meseguer and A. Verdejo, *Towards a Strategy Language for Maude*, Electronic Notes in Theoretical Computer Science **117** (2005), pp. 417–441.

[19] Martí-Oliet, N., J. Meseguer and A. Verdejo, *A Rewriting Semantics for Maude Strategies*, in: *Proceedings of WRLA*, 2008, pp. 207–226, to appear in ENTCS.

[20] Meseguer, J., *Conditional Rewriting Logic as a Unified Model of Concurrency*, Theoretical Computer Science **96** (1992), pp. 73–155.

[21] Meseguer, J., *Membership algebra as a logical framework for equational specification*, in: Francesco Parisi-Presicce, editor, *WADT*, Lecture Notes in Computer Science **1376** (1997), pp. 18–61.

[22] Paun, G., "Membrane Computing. An Introduction," Springer, 2002.

[23] Visser, E., *Stratego: A Language for Program Transformation based on Rewriting Strategies. System Description of Stratego 0.5*, in: A. Middeldorp, editor, *RTA*, Lecture Notes in Computer Science **2051** (2001), pp. 357–361.

[24] Visser, E., Z.-E.-A. Benaissa and A. P. Tolmach, *Building Program Optimizers with Rewriting Strategies*, in: *ICFP*, 1998, pp. 13–26.