University "Alexandru Ioan Cuza" of Iaşi
Faculty of Computer Science

TECHNICAL REPORT

# Strategies and Tactics in Operational Semantics

**O. Andrei, G. Ciobanu, D. Lucanu**

**TR 06-01,** May 2006

**ISSN 1224-9327**

**Universitatea "Alexandru Ioan Cuza" Iaşi**
**Facultatea de Informatică**
Str. Berthelot 16, 700483-Iaşi, Romania
Tel. +40-32-201090, email: bibl@infoiasi.ro

# Strategies and Tactics in Operational Semantics

Oana Andrei [2]

*INRIA-LORIA, 54602 Villers-lès-Nancy Cedex, France*

Gabriel Ciobanu [1,3]

*Institute of Computer Science, Romanian Academy, 700505 Iaşi, Romania*

Dorel Lucanu [1,4]

*Faculty of Computer Science, "A.I.Cuza" University, 700483 Iaşi, Romania*

**Abstract**

We refer to strategies setting the objective of a computation step rather than to evaluation strategies (as eager or lazy evaluations). We use these strategies to define semantics of a system starting from its elementary operational entities, and then combining them. The tactics of a strategy are given by rewriting steps. While a strategy is at a higher level and defines "what is the goal" of a computation, the tactics are at a lower level and tell "how it is possible to reach the goal".

We use rewriting systems as a general model of computing. In this framework we give an operational semantics of the strategic transitions in terms of tactical rewritings. The approach is inspired by a new model of computation given by membrane systems. We define the strategy semantics for membrane systems involving the maximal parallel rewriting and priorities. We show that strategies are not powerful enough to define alone the semantics of membrane systems involving promoters. This is possible when we encode the state of the membrane in a richer structure.

*Key words:* strategic transitions, tactical rewritings, operational semantics, membrane computing

## 1 Introduction

In general terms, a strategy is setting the objective(s) of a computation. On the other hand, tactics indicate how we are supposed to reach the objectives. In semantics of systems involving parallelism, a complex computation steps can be explained as indicating

[1] Research partially supported by CEEX Project 47/2005
[2] Email: Oana.Andrei@loria.fr
[3] Email: gabriel@iit.tuiasi.ro
[4] Email: dlucanu@info.uaic.ro

a strategy for possible sequential steps understandable by a human mind and leading to the same result.

In this paper we discuss about strategic transitions and tactical rewritings in operational semantics. In rewriting systems, we can describe a computation by using a set of strategy operators and combining them. We add new axioms in defining the strategies. The corresponding tactics of a strategy can be expressed as a sequence of rewriting steps. Tactics corresponding to a given strategy could be derived by using a continuation-passing semantics.

We use strategic transitions, and define tactical rewritings of the strategic transitions in order to provide semantics for membrane computing [7]. Several results and examples of the paper explain how the strategies and tactics can be use in describing the semantics of the membrane systems (called also P systems). In this paper we consider the systems composed of a single membrane. Given a set of rules of such a system composed by objects $w$, we investigate whether we have a strategy $s$ exists such that $w \Rrightarrow w'$ iff $w \xrightarrow{s} w'$. We are able to give a strategic semantics for maximal parallel rewriting, as well as for maximal parallel rewriting with priorities between the rules. A more powerful mechanism than strategies is needed to provide semantics for maximal rewriting with promoters. A useful encoding of the rules can solve this problem, and finally we can provide the semantics for maximal rewriting of membrane systems involving promoters. The encoding can be used in a uniform way to provide the strategy semantics for all the maximal rewritings mentioned before, namely simple maximal rewriting, maximal rewriting of membrane systems with priorities, and maximal rewriting of systems with promoters and inhibitors.

## 2 Rewriting Strategies

A *rewriting strategy* is an algorithm for exploring the reduction graph induced by a set of rules. In particular rewriting rules are atomic strategies. In order to define strategies for membrane systems we use elements of the strategy language defined in [10].

A *signature* $\Sigma$ consists of a set of operator names together with a natural number $arity(f)$ for each operator (function) name $f \in \Sigma$. We use $\Sigma_n$ to denote the subsignature of the operator names of arity $n$. An operator name $f \in \Sigma_0$ is called *constant*. If $X$ is a set of variables, then $\Sigma(X)$ denotes the signature $\Sigma \cup X$, where the variables $X$ are seen as constants. The set $T_\Sigma(X)$ of *terms* over the signature $\Sigma$ and the set of variables $X$ is inductively defined as follows:

(i) each constant is a term, i.e., $\Sigma_0 \subseteq T_\Sigma(X)$;

(ii) each variable is a term, i.e., $X \subseteq T_\Sigma(X)$;

(iii) if $f \in \Sigma_n$, $n > 0$, and $t_1, \ldots, t_n \in T_\Sigma(X)$, then $f(t_1, \ldots, t_n) \in T_\Sigma(X)$.

If $t$ is a term, the $var(t)$ denotes the variables occurring in $t$. A term $t$ is a *subterm* of the term $t'$ iff either $t = t'$ or $t' = f(t_1, \ldots, t_n)$ and there is $i$ such that $t$ is a subterm of $t_i$. A *context* is a term $t$ having a single occurrence of a special variable $\bullet$; we write $t[\bullet]$. A *substitution* is a mapping $\vartheta : X \to T_\Sigma(Y)$. A substitution $\vartheta$ is uniquely extended to terms $T_\Sigma(X)$: $\vartheta(f) = f$ for each constant $f$, and $\vartheta(f(t_1, \ldots, t_n)) = f(\vartheta(t_1), \ldots, \vartheta(t_n))$. By $t[x := t']$ we denote the term obtained from $t$ by replacing the occurrences of the variable $x$ by $t'$; $t[x_1 := t_1, \ldots, x_n := t_n]$ denotes the term $t[x_1 := t_1] \cdots [x_n := t_n]$. A

*rewrite rule* is a triple of the form $\ell : u \to v$, where $\ell$ is the label of the rule, $u$ and $v$ are terms such that $var(v) \subseteq var(u)$. If $R$ is a set of rewrite rules, We say that $t_1$ *(directly) rewrites* $t_2$ w.r.t. $R$ iff there is a a rule $\ell : u \to v \in R$, a substitution $\vartheta : var(u) \to T_\Sigma(X)$, and a context $t_0$ such that $t_1 = t_0[\bullet := \vartheta(u)]$ and $t_2 = t_0[\bullet := \vartheta(v)]$. We write $t_1 \to_R t_2$; we omit to write the subscript $R$ whenever it is clear from context.

Consider a set $Z$ of *(strategy) variables* $z, z_1, z_2, \ldots$, $\Sigma$ a signature, and a set $R$ of rewrite rules $\ell : u \to v$, where $\ell$ is the label of the rule, and $u, v$ are $\Sigma(X)$-terms. The set of *strategy terms* over $Z$ and $R$ is inductively defined:

(i) each variable $z$ is a strategy term;

(ii) if $\ell : u \to v$ a rule in $R$, then $\ell$ is a strategy term;

(iii) $\mathsf{id}$ and $\mathsf{fail}$ are strategy terms;

(iv) if $s, s_1, s_2, \ldots$ are strategies terms, $i$ is a natural number, and $f$ is a function name of arity $n$, then $\mathtt{test}\ s$, $\neg s$, $s_1 + s_2$, $s_1; s_2$, $s_1 \nleftarrow s_2$, $\mu z(s)$, $i(s)$, $f(s_1, s_2, \ldots, s_n)$, $\Diamond(s)$, $\Box(s)$, and $\boxdot(s)$ are strategy terms.

An occurrence of $z$ in $s$ is *bound* iff it is in the scope of a $\mu z$; otherwise we say that the occurrence of $z$ is *free*. A *ground strategy term* is a strategy term without free variables.

A *strategy definition* is an expression of the form

$$\varphi(z_1, \ldots, z_n) \stackrel{\text{def}}{=} s$$

where any free variable in $s$ belongs to $\{z_1, \ldots, z_n\}$, and $\varphi$ is a *strategy identifier*.

A *strategy* is a ground strategy term or an application $\varphi(s_1, \ldots, s_n)$.

### *2.1 Strategy Operational Semantics*

A *strategic transition* is an expression of the form $t \xrightarrow{s} t'$, where $t$ is a term, $t'$ is either a term or $\uparrow$, and $s$ is a strategy. In what follows we recall from [10] the semantics of the strategies. We write $(\Sigma, R) \vdash t \xrightarrow{s} t'$ iff $t \xrightarrow{s} t'$ can be obtained by applying deduction rules $S_0 - S_{25}$ (see Fig. 1 and Fig. 2) of finitely times and using only operators from $\Sigma$ and rules from $R$. The operational semantics of an unconditional rewrite rule $\ell : u \to v \in R$ is the set of strategic transitions $t \xrightarrow{\ell} t'$ where $t'$ is obtained from $t$ by applying at top the rewrite rule labelled by $\ell$.

Some basic strategy operators which can be combined to provide more complex strategies are the following:

- the *identity* strategy $\mathsf{id}$ always succeeds;
- the *fail* strategy $\mathsf{fail}$ is the dual of identity and it always fails;
- the strategy $\mathsf{test}\ s$ is used to test whether the strategy $s$ is successful or not without having the transforming effect of $s$;
- the *negation* strategy is similar to $\mathsf{test}$, but tests for the failure of $s$;
- the *sequential composition* $s_1; s_2$ applies $s_1$ and, if that succeeds, it applies $s_2$;
- the *non-deterministic choice* $s_1 + s_2$ chooses between the strategies $s_1$ and $s_2$ such that the chosen strategy succeeds;
- the *deterministic* or *left choice* $s_1 \nleftarrow s_2$ chooses the left argument first; the second strategy is considered if the first does not succeed.

$$(S_0)\ \frac{}{(\Sigma,R)\vdash \vartheta(u)\xrightarrow{\ell}\vartheta(v)}\quad \text{if } \ell: u\to v\in R, \vartheta: var(u)\to T_\Sigma(X)$$

$$(S_1)\ \frac{}{(\Sigma,R)\vdash t\xrightarrow{\mathsf{id}} t} \qquad (S_2)\ \frac{}{(\Sigma,R)\vdash t\xrightarrow{\mathsf{fail}}\uparrow}$$

$$(S_3)\ \frac{(\Sigma,R)\vdash t\xrightarrow{s} t'}{(\Sigma,R)\vdash t\xrightarrow{\mathsf{test}\ s} t} \qquad (S_4)\ \frac{(\Sigma,R)\vdash t\xrightarrow{s}\uparrow}{(\Sigma,R)\vdash t\xrightarrow{\mathsf{test}\ s}\uparrow}$$

$$(S_5)\ \frac{(\Sigma,R)\vdash t\xrightarrow{s}\uparrow}{(\Sigma,R)\vdash t\xrightarrow{\neg s} t} \qquad (S_6)\ \frac{(\Sigma,R)\vdash t\xrightarrow{s} t'}{(\Sigma,R)\vdash t\xrightarrow{\neg s}\uparrow}$$

$$(S_7)\ \frac{(\Sigma,R)\vdash t\xrightarrow{s_1} t'\quad (\Sigma,R)\vdash t'\xrightarrow{s_2} t''}{(\Sigma,R)\vdash t\xrightarrow{s_1;s_2} t''} \qquad (S_8)\ \frac{(\Sigma,R)\vdash t\xrightarrow{s_1}\uparrow}{(\Sigma,R)\vdash t\xrightarrow{s_1;s_2}\uparrow}$$

$$(S_9)\ \frac{(\Sigma,R)\vdash t\xrightarrow{s_1} t'\ (\Sigma,R)\vdash t'\xrightarrow{s_2}\uparrow}{(\Sigma,R)\vdash t\xrightarrow{s_1;s_2}\uparrow}$$

$$(S_{10})\ \frac{(\Sigma,R)\vdash t\xrightarrow{s_1} t'}{(\Sigma,R)\vdash t\xrightarrow{s_1+s_2} t'}$$

$$(S_{11})\ \frac{(\Sigma,R)\vdash t\xrightarrow{s_2} t'}{(\Sigma,R)\vdash t\xrightarrow{s_1+s_2} t'} \qquad (S_{12})\ \frac{(\Sigma,R)\vdash t\xrightarrow{s_1}\uparrow \quad (\Sigma,R)\vdash t\xrightarrow{s_2}\uparrow}{(\Sigma,R)\vdash t\xrightarrow{s_1+s_2}\uparrow}$$

$$(S_{13})\ \frac{(\Sigma,R)\vdash t\xrightarrow{s_1} t'}{(\Sigma,R)\vdash t\xrightarrow{s_1 \leftarrow\!\!+ s_2} t'}$$

$$(S_{14})\ \frac{(\Sigma,R)\vdash t\xrightarrow{s_1}\uparrow\ (\Sigma,R)\vdash t\xrightarrow{s_2} t'}{(\Sigma,R)\vdash t\xrightarrow{s_1 \leftarrow\!\!+ s_2} t'} \qquad (S_{15})\ \frac{(\Sigma,R)\vdash t\xrightarrow{s_1}\uparrow\ (\Sigma,R)\vdash t\xrightarrow{s_2}\uparrow}{(\Sigma,R)\vdash t\xrightarrow{s_1 \leftarrow\!\!+ s_2}\uparrow}$$

$$(S_{16})\ \frac{(\Sigma,R)\vdash t\xrightarrow{s[z:=\mu z(s)]} t'}{(\Sigma,R)\vdash t\xrightarrow{\mu z(s)} t'} \qquad (S_{17})\ \frac{(\Sigma,R)\vdash t\xrightarrow{s[z:=\mu z(s)]}\uparrow}{(\Sigma,R)\vdash t\xrightarrow{\mu z(s)}\uparrow}$$

$$(S_{18})\ \frac{(\Sigma,R)\vdash t\xrightarrow{s[z_1:=s_1,\ldots,z_n:=s_n]} t'}{(\Sigma,R)\vdash t\xrightarrow{\varphi(s_1,\ldots,s_n)} t'}\quad \text{if } \varphi(z_1,\ldots,z_n)\stackrel{\text{def}}{=} s$$

$$(S_{19})\ \frac{(\Sigma,R)\vdash t\xrightarrow{s[z_1:=s_1,\ldots,z_n:=s_n]}\uparrow}{(\Sigma,R)\vdash t\xrightarrow{\varphi(s_1,\ldots,s_n)}\uparrow}\quad \text{if } \varphi(z_1,\ldots,z_n)\stackrel{\text{def}}{=} s$$

Fig. 1. Operational semantics for the basic strategy operators

The recursion operator $\mu x(s)$ allows defining strategies that repeatedly apply the strategy $s$. For example the strategy *repeat* defined as

$$\mathsf{repeat}(s) = \mu z((s; z) \leftarrow\!\!+ \mathsf{id})$$

applies $s$ as many times as possible and it never fails.

The basic strategy operators can be applied only at the root of terms. In order to apply strategies at arbitrary depth in a term, one can use (combination of) the basic following traversal operators:

- $i(s)$ applies the strategy $s$ to the $i$-th child of the root;
- *congruence* operators specify application of strategies to the children of terms constructed with a specific constructor;
- $\Diamond(s)$ applies $s$ non-deterministically to one child for which it succeeds; it fails if there is no child for which it succeeds;
- $\Box(s)$ applies $s$ to each child of the root and succeeds if $s$ succeeds for each child;

• $\overset{\cdot}{\square}(s)$ applies $s$ to some children of the root; it fails if it fails for all children.

$$(S_{20})\ \frac{(\Sigma,R)\vdash t_i \xrightarrow{s} t_i'}{(\Sigma,R)\vdash f(t_1,\ldots,t_i,\ldots,t_n)\xrightarrow{i(s)} f(t_1,\ldots,t_i',\ldots,t_n)}$$

$$(S_{21})\ \frac{(\Sigma,R)\vdash t_1 \xrightarrow{s_1} t_1' \quad \ldots (\Sigma,R)\vdash t_n \xrightarrow{s_n} t_n'}{(\Sigma,R)\vdash f(t_1,\ldots,t_n)\xrightarrow{f(s_1,\ldots,s_n)} f(t_1',\ldots,t_n')}$$

$$(S_{22})\ \frac{(\Sigma,R)\vdash t_i \xrightarrow{s} t_i'}{(\Sigma,R)\vdash f(t_1,\ldots,t_i,\ldots,t_n)\xrightarrow{\Diamond(s)} f(t_1,\ldots,t_i',\ldots,t_n)}$$

$$(S_{23})\ \frac{(\Sigma,R)\vdash t_1 \xrightarrow{s} t_1' \quad \ldots (\Sigma,R)\vdash t_n \xrightarrow{s} t_n'}{(\Sigma,R)\vdash f(t_1,\ldots,t_n)\xrightarrow{\Box(s)} f(t_1',\ldots,t_n')}$$

$$(S_{24})\ \frac{}{(\Sigma,R)\vdash f(t_1,\ldots,t_n)\xrightarrow{\overset{\cdot}{\square}(s)} f(P(t_1),\ldots,P(t_n))}$$

$$\text{if } \exists j\forall i : i,j\in\{1,\ldots,n\}\wedge P(t_i)\begin{cases} t_i' & \text{if } (\Sigma,R)\vdash t_i \xrightarrow{s} t_i' \\ t_i & \text{if } (\Sigma,R)\vdash t_i \xrightarrow{s}\uparrow \wedge i\neq j\end{cases}$$

$$(S_{25})\ \frac{(\Sigma,R)\vdash t_1 \xrightarrow{s}\uparrow \ \ldots\ (\Sigma,R)\vdash t_n \xrightarrow{s}\uparrow}{(\Sigma,R)\vdash f(t_1,\ldots,t_n)\xrightarrow{\overset{\cdot}{\square}(s)}\uparrow}$$

Fig. 2. Operational semantics for term traversal operators

Various traversal strategies can be defined by means of these operators. Some traversal strategies we use in the following are:

$$\mathsf{topdown}(s) \stackrel{\text{def}}{=} \mu z(s;\Box(z)) \tag{1}$$

$$\mathsf{bottomup}(s) \stackrel{\text{def}}{=} \mu z(\Box(z);s) \tag{2}$$

$$\mathsf{once}(s) \stackrel{\text{def}}{=} \mu z(\Diamond(z)+s) \tag{3}$$

$$\mathsf{reduce}(s) \stackrel{\text{def}}{=} \mathsf{repeat}(\mu z(\Diamond(z)+s)) = \mathsf{repeat}(\mathsf{once}(s)) \tag{4}$$

**Example 2.1** We suppose that $f\in\Sigma_2$ and $a,b\in\Sigma_0$, and $R$ consists of the rules $\ell_1 : f(b,f(a,x))\to f(a,f(b,x))$ and $\ell_2 : f(a,a)\to a$. Then
$(\Sigma,R)\vdash f(b,f(a,f(a,a)))) \xrightarrow{\mathsf{topdown}(s)} f(a,f(a,f(b,a)))$ and
$(\Sigma,R)\vdash f(b,f(a,f(a,a)))) \xrightarrow{\mathsf{bottomup}(s)} f(b,a)$, where $s=(\ell_1+\ell_2)\nleftarrow \mathrm{id}$.

### *2.2 Tactical Rewritings of Strategic Transitions*

We often denote a rewriting $t=t_0\to\cdots\to t_n=t'$ by the shorter notation $t\rightsquigarrow t'$. If $n=0$, then we write $t\rightsquigarrow_0 t'$. The short notation is extended as follows:

- if $t_i\rightsquigarrow t_i'$ is $t_i=t_{i_0}\to\cdots\to t_{i_n}=t'$, then $f(t_1,\ldots,t_i,\ldots,t_n)\rightsquigarrow f(t_1,\ldots,t_i',\ldots,t_n)$ denotes $f(t_1,\ldots,t_{i_0},\ldots,t_n)\to\cdots\to f(t_1,\ldots,t_{i_n},\ldots,t_n)$;
- $f(t_1,\ldots,t_n)\rightsquigarrow f(t_1',\ldots,t_n')$ denotes $f(t_1,\ldots,t_n)\rightsquigarrow f(t_1',t_2,\ldots,t_n)\rightsquigarrow\cdots\rightsquigarrow f(t_1',\ldots,t_n')$.

Intuitively, a *tactical rewriting* for a strategic transition $t\xrightarrow{s}t'$ is a rewriting $t\rightsquigarrow t'$ whose rewriting steps are given according to $s$; we write $t\rightsquigarrow t'\models t\xrightarrow{s}t'$. We denote by $[\![t\xrightarrow{s}t']\!]$ the set of tactical rewritings corresponding to $t\xrightarrow{s}t'$.

Formally, we have:

(i) $t \to t' \models t \xrightarrow{\ell} t'$ if $t \xrightarrow{\ell} t'$ is defined as in $S_0$;

(ii) $t \rightsquigarrow_0 t \models t \xrightarrow{\mathrm{id}} t$;

(iii) $t \rightsquigarrow t \models t \xrightarrow{\texttt{test}\ s} t'$ if $[\![t \xrightarrow{s} t']\!] \neq \emptyset$;

(iv) $t \rightsquigarrow t \models t \xrightarrow{\neg s} t$ if $(\forall t')[\![t \xrightarrow{s} t']\!] = \emptyset$;

(v) $t \rightsquigarrow t' \rightsquigarrow t'' \models t \xrightarrow{s;s'} t''$ if $t \rightsquigarrow t' \models t \xrightarrow{s} t'$ and $t' \rightsquigarrow t'' \models t' \xrightarrow{s'} t''$;

(vi) $t \rightsquigarrow t' \models t \xrightarrow{s+s'} t'$ if $t \rightsquigarrow t' \models t \xrightarrow{s} t'$ or $t \rightsquigarrow t' \models t \xrightarrow{s'} t'$;

(vii) $t \rightsquigarrow t' \models t \xrightarrow{s+\!\!\!\!-\,s'} t'$ if $[\![t \xrightarrow{s} t']\!] = \emptyset$ and $t \rightsquigarrow t' \models t \xrightarrow{s'} t'$;

(viii) $t \rightsquigarrow t' \models t \xrightarrow{\mu z(s)} t'$ if $t \rightsquigarrow t' \models t \xrightarrow{s[z:=\mu z(s)]} t'$;

(ix) $t \rightsquigarrow t' \models t \xrightarrow{\varphi(s_1,\ldots,s_n)} t'$ if $\varphi(z_1,\ldots,z_n) \stackrel{\text{def}}{=} s$ and $t \rightsquigarrow t' \models t \xrightarrow{s[z_1:=s_1,\ldots,z_n:=s_n]} t'$.

(x) $t \rightsquigarrow t' \models t \xrightarrow{i(s)} t'$ if $t = f(t_1,\ldots,t_i,\ldots,t_n)$, $t' = f(t_1,\ldots,t'_i,\ldots,t_n)$, and $t_i \rightsquigarrow t'_i \models t_i \xrightarrow{s} t'_i$;

(xi) $t \rightsquigarrow t' \models t \xrightarrow{f(s_1,\ldots,s_n)} t'$ if $t = f(t_1,\ldots,t_n)$, $t' = f(t'_1,\ldots,t'_n)$, and $t_i \rightsquigarrow t'_i \models t_i \xrightarrow{s_i} t'_i$ for $i = 1,\ldots,n$;

(xii) $t \rightsquigarrow t' \models t \xrightarrow{\Diamond(s)} t'$ if $t = f(t_1,\ldots,t_i,\ldots,t_n)$, $t' = f(t_1,\ldots,t'_i,\ldots,t_n)$, and $t_i \rightsquigarrow t'_i \models t_i \xrightarrow{s} t'_i$ for certain $i \in \{1,\ldots,n\}$;

(xiii) $t \rightsquigarrow t' \models t \xrightarrow{\Box(s)} t'$ if $t = f(t_1,\ldots,t_n)$, $t' = f(t'_1,\ldots,t'_n)$, and $t_i \rightsquigarrow t'_i \models t_i \xrightarrow{s} t'_i$ for $i = 1,\ldots,n$;

(xiv) $t \rightsquigarrow t' \models t \xrightarrow{\boxtimes(s)} t'$ if $t = f(t_1,\ldots,t_n)$, $t' = f(t'_1,\ldots,t'_n)$, there is $j$ such that $t_j \rightsquigarrow t'_j \models t_j \xrightarrow{s} t'_j$, and $t_i \rightsquigarrow t'_i \models t_i \xrightarrow{s+\!\!\!\!-\,\mathrm{id}} t'_i$ for $i \in \{1,\ldots,n\} \wedge i \neq j$;

The proof of the following result follows from definitions:

**Theorem 2.2** *If $t, t' \in T_\Sigma(X)$ and $(\Sigma, R) \vdash t \xrightarrow{s} t'$, then $[\![t \xrightarrow{s} t']\!]$ is not empty.*

**Theorem 2.3** *If $(\Sigma, R) \vdash t \xrightarrow{s}\uparrow$, then $[\![t \xrightarrow{s} t']\!] = \emptyset$ for each term $t'$.*

**Proof.** We assume that $t \rightsquigarrow t' \models t \xrightarrow{s} t'$ and we show by structural induction on $s$ that this assumption implies a contradiction. For instance, we assume $s$ is of the form $\neg s'$. By definition of $\models$, it follows that $t' = t$ and $[\![t \xrightarrow{s'} t'']\!] = \emptyset$, for each $t''$. On the other hand, we have $(\Sigma, R) \vdash t \xrightarrow{s'} t''$ for certain $t''$ by definition of $\vdash$. We apply Theorem 2.2 and we get $[\![t \xrightarrow{s'} t'']\!] \neq \emptyset$. Contradiction. □

**Theorem 2.4** *If $[\![t \xrightarrow{s} t']\!]$ is not empty, then $(\Sigma, R) \vdash t \xrightarrow{s} t'$.*

**Corollary 2.5** *$(\Sigma, R) \vdash t \xrightarrow{s} t'$ iff $[\![t \xrightarrow{s} t']\!]$ is not empty.*

Theorems 2.2 and 2.3 show that $\vdash$ is sound w.r.t.$[\![\_]\!]$, and Theorem 2.4 shows that $\vdash$ is complete w.r.t. $[\![\_]\!]$. We can also prove some nice properties of the strategy operators. Here are two examples:

**Proposition 2.6**

(i) $[\![id; s]\!] = [\![s]\!] = [\![s; id]\!]$.

(ii) $[\![(s; s'); s'']\!] = [\![s; (s'; s'')]\!]$.

We conclude that Section 2.1 defines a proof system providing a big-step semantics for strategic transitions, and Section 2.2 shows how rewriting tactics achieve the strategic transitions, and so providing a small-step semantics.

## 3 Strategic Semantics of Membrane Computing

Membrane systems represent a new abstract model inspired by cell compartments and molecular membranes. Such a system is composed of various compartments, each compartment with a different task, and all of them working simultaneously to accomplish a more general task of the whole system. A detailed description of the membrane systems (also called P systems) can be found in [7]. Informally, a membrane system consists of a hierarchy of membranes which do not intersect, with a distinguishable membrane, called the *skin membrane*, surrounding all the other membranes. The membranes produce a delimitation between *regions*. For each membrane there is a unique associated region. Regions contain multisets of *objects*, *evolution rules* and possibly other membranes. Only rules in a region delimited by a membrane act on the objects in that region. The multisets of objects from a region correspond to the "chemicals swimming in the solution in the cell compartment", while the rules correspond to the "chemical reactions possible in the same compartment". The rules must contain target indications, specifying the membrane where the new objects obtained after applying the rule are sent. The new objects either remain in the same region when they have a *here* target, or they pass through membranes, in two directions: they can be sent *out* of the membrane delimiting a region from outside, or can be sent *in* one of the membranes delimiting a region from inside, precisely identified by its label. In a step, the objects can pass only through one membrane. We refer mainly to the so-called *transition membrane systems*. Other variants and classes of P systems are introduced [7]. In this paper we work with P systems having a single membrane (the skin membrane). In this way we focus on the maximal parallel application of the rules.

We consider $\Sigma = \Sigma_0 \cup \Sigma_2$, where $\Sigma_0$ consists of the empty multiset (string) $\varepsilon$ together the objects, and $\Sigma_2$ consists of concatenation operator $\_\_$ (inspired from the infix notation). We denote by $ACU$ the axioms expressing associativity and commutativity of the concatenation and the unit law of $\varepsilon$. Moreover, working with terms modulo $ACU$, the operator $\_\_$ becomes variadic. For instance, $aabbb$ can also be written as $\_\_(aa, bbb)$, $\_\_(a, a, b, b, b)$, $\_\_(a, ab, bb)$ and so on. All these expressions represents the same multiset.

In this section we answer the following question: Given a membrane with the set of evolution rules $R$, is there a strategy $s$ such that any evolution step $w \Rrightarrow w'$ can be implemented by $s$, i.e., $(\Sigma, R) \vdash w \xrightarrow{s} w'$? For maximal parallel rewriting and maximal parallel rewriting with priorities the answer is positive, whereas for maximal parallel rewriting with promoters the answer is negative. However, for the last case we show that there is an encoding $(\widehat{\Sigma}, \widehat{R})$ of $(\Sigma, R)$ such that $(\widehat{\Sigma}, \widehat{R}) \vdash w \xrightarrow{s} w'$.

### 3.1 Strategic Semantics of Maximal Parallel Rewriting

Let $R$ be a set of evolution rules and $w$ a multiset of objects. If $\ell : u \to v$ an evolution rule in $R$, then $w$ is *$\ell$-irreducible* if $(\Sigma, R) \vdash w \xrightarrow{\mathsf{once}(\ell)} \uparrow$. Moreover, $w$ is *$R$-irreducible* if $w$ is $\ell$-irreducible for all $\ell : u \to v \in R$. In other words, $w$ is $\ell$-irreducible iff there is no $w'$ such that $w \to w'$ applying the rule labelled by $\ell$. We say that $w$ *maximal parallel rewrites* in $w'$, write $w \Rrightarrow_R w'$, iff $w =_{ACU} u_1 \cdots u_n z$, $w' =_{ACU} v_1 \cdots v_n z$, $\ell_i : u_i \to v_i$ is a rule in $R$ for $i = 1, \ldots, n$, $n > 0$, and $z$ is $R$-irreducible.

**Proposition 3.1** *If $w \Rrightarrow_R w'$, then there are $w_1$, $w_1'$, and a strategy $s$ such that $w =_{ACU} w_1$, $w_1' =_{ACU} w'$, and $w_1 \xrightarrow{s} w_1'$.*

We note that the strategy $s$ is applied modulo associativity, commutativity, and identity of concatenation. Moreover, for two different maximal parallel rewrites we have two distinct strategies.

For instance, let $R$ consist of the rules $\ell_1 : ab \to c$ and $\ell_2 : bb \to d$. We have

$$aabbb =_{ACU} ababb \wedge ababb \xrightarrow{s_1} ccb$$

where $s_1 = \_\_(\ell_1, \ell_1, \mathrm{id})$, and

$$bb =_{ACU} abbba \wedge abbba \xrightarrow{s_2} cda$$

where $s_2 = \_\_(\ell_1, \ell_2, \mathrm{id})$. If $s = s_1 + s_2$, then $aabbb =_{ACU} w \xrightarrow{s} w'$, where $(w, w') \in \{(ababb, ccb), (abbba, cda)\}$. The challenging question we wish to answer is: does it exist a strategy $s$ such that $aabbb \xrightarrow{s} w'$? We recall that a strategy can be nondeterministic and hence we can obtain more than one $w'$ from $aabbb$ with the same strategy.

If we restrict the strategies at those defined in Section 2, then the answer is no. The reason is given by the fact that these strategies are given over terms, and in membrane systems we use multisets (terms which are equal modulo associativity, commutativity, and identity). Therefore, a positive answer can be obtained if we extend the strategies over terms modulo a set of axioms. We add a new operator over strategies defined as $f[attr](s_1, s_2, \ldots, s_n)$, where *attr* is a set of attributes like associativity (`assoc`), commutativity (`comm`), identity (`id:` *neutral element*). The attributes *attr* gives rise to a set of axioms $ax(attr)$. For instance, if *attr* is "`assoc comm id:` $\varepsilon$", then $ax(attr) = ACU$ and it consists of the following axioms:

$$\{(\forall x, y, z) f(x, f(y, z)) = f(f(x, y), z), (\forall x, y) f(x, y) = f(y, x), (\forall x) f(x, \varepsilon) = x\}.$$

The semantics of this new operator is:

$$(S_{26}) \; \frac{\begin{array}{c} f(t_1, \ldots, t_m) =_{ax(attr)} f(t_1', \ldots, t_n') \text{ and} \\ (\Sigma, R) \vdash f(t_1', \ldots, t_n') \xrightarrow{f(s_1, \ldots, s_n)} f(t_1'', \ldots, t_n'') \end{array}}{(\Sigma, R) \vdash f(t_1, \ldots, t_m) \xrightarrow{f[attr](s_1, \ldots, s_n)} f(t_1'', \ldots, t_n'')}$$

$$(S_{27}) \; \frac{\begin{array}{c} (\forall f(t_1', \ldots, t_n')) \; f(t_1, \ldots, t_m) =_{ax(attr)} f(t_1', \ldots, t_n') \text{ implies} \\ (\Sigma, R) \vdash f(t_1', \ldots, t_n') \xrightarrow{f(s_1, \ldots, s_n)} \uparrow \end{array}}{(\Sigma, R) \vdash f(t_1, \ldots, t_m) \xrightarrow{f[attr](s_1, \ldots, s_n)} \uparrow}$$

Given a set of evolution rules $R = \{\ell_i : u_i \to v_i \mid 1 \le i \le n\}$, we define the strategy

$$mpr \stackrel{\text{def}}{=} \mu x(s_1 + \cdots + s_n)$$
$$\text{where } s_i = \_\_[\texttt{assoc comm id}:\ \varepsilon](\ell_i, x \leftarrow \text{id}),\ \text{for } i = 1, \ldots, n.$$

Since the definition of the strategy $mpr$ is depending on $R$, we prefer to write it in an equivalent form

$$mpr(R) \stackrel{\text{def}}{=} +_{i=1}^{n} (\_\_[\texttt{assoc comm id}:\ \varepsilon](\ell_i, mpr(R) \leftarrow \text{id}))$$

If $R = \emptyset$, then $mpr(\emptyset) = \mathsf{fail}$.

We extend the definition of $\vdash$ to $(\Sigma, Ax, R) \vdash t \xrightarrow{s} t'$, where $Ax$ is the axioms corresponding to the attributes of the operators in $\Sigma$. For membranes, $\Sigma$ consists of object symbols (viewed as constants) and the concatenation, and $Ax$ consists of the axioms expressing the associativity, commutativity, and the neutrality of $\varepsilon$ for the concatenation.

**Theorem 3.2** *Given a set $R$ of evolution rules, then*

$$w \Rrightarrow_R w' \text{ iff } (\Sigma, Ax, R) \vdash w \xrightarrow{mpr(R)} w'.$$

**Proof.** We first assume that $w \Rrightarrow_R w'$. It follows that $w =_{ACU} u_{i_1} \cdots u_{i_k} z$, $w' =_{ACU} v_{i_1} \cdots v_{i_k} z$, $\ell_{i_j} : u_{i_j} \to v_{i_j}$ is a rule in $R = \{\ell_i : u_i \to v_i \mid i = 1, \ldots, n\}$ for $j = 1, \ldots, k$, $k > 0$, and $z$ is $R$-irreducible. We show by induction on $k$ that $w \xrightarrow{mpr(R)} w'$.
If $k = 1$, then the proof is:

$$\dfrac{\dfrac{\dfrac{\dfrac{z \xrightarrow{mpr(R)}\uparrow \qquad S_0\ z \xrightarrow{\text{id}} z}{S_{14}}}{z \xrightarrow{mpr(R)+\text{id}} z} \qquad u_i \xrightarrow{\ell_i} v_i}{S_{26}}}{u_i z \xrightarrow{s_i} v_i z}\, S_{10}}{u_i z \xrightarrow{+_i s_i} v_i z}\, S_{16}}{u_i z \xrightarrow{mpr(R)} v_i z}$$

where $i = i_1$.
If $k > 1$, then $u_{i_k} z \xrightarrow{mpr(R)} v_{i_k} z$ as above, and $u_1 \ldots u_{i_{k-1}} \xrightarrow{mpr} v_1 \ldots v_{i_{k-1}}$ by inductive hypothesis. We get $w \xrightarrow{mpr(R)} w'$ by the definition of the fixpoint operator, and by the fact that the concatenation in the left-hand side does not produce new reducible configurations.
Conversely, if $w \xrightarrow{mpr(R)} w'$ then we show that $w \Rrightarrow_R w'$ by induction on the depth of the proof tree. There are $\ell_i : u_i \to v_i$ in $R$, $w_i$ and $w_i'$ such that $w = u_i w_i$, $w' = v_i w_i'$, and $w_i \xrightarrow{mpr(R)} w_i'$, by definition of $mpr(R)$. We have $w_i \Rrightarrow_R w_i'$ by inductive hypothesis and we get $w = u_i w_i \Rrightarrow_R v_i w_i' = w'$ by the definition of $\Rrightarrow$. □

For instance, if $R$ is that given in the above example, then the proof of $aabbb \xrightarrow{mpr} cda$ is:

$$
S_{14}\ \dfrac{\dfrac{S_0\ a \xrightarrow{\mathrm{id}} a \qquad a \xrightarrow{mpr(R)} \uparrow}{S_{26}\ a \xrightarrow{mpr(R) \leftarrow\!\!+ \mathrm{id}} a \qquad\qquad bb \xrightarrow{\ell_2} d}}{S_{11}\ \dfrac{abb \xrightarrow{s_2} da}{S_{16}\ \dfrac{abb \xrightarrow{s_1 + s_2} da}{S_{13}\ \dfrac{abb \xrightarrow{mpr(R)} da}{S_{26}\ \dfrac{abb \xrightarrow{mpr(R) \leftarrow\!\!+ \mathrm{id}} da \qquad\qquad ab \xrightarrow{\ell_1} c}{S_{10}\ \dfrac{aabbb \xrightarrow{s_1} cda}{S_{16}\ \dfrac{aabbb \xrightarrow{s_1 + s_2} cda}{aabbb \xrightarrow{mpr(R)} cda}}}}}}}
$$

### *3.2 Strategic Semantics of Maximal Parallel Rewriting with Priorities*

Let $R$ be a set of evolution rules together with a partial order $\succ$. If $\ell \succ \ell'$, then we say that $\ell$ has a greater priority than $\ell'$. An evolution rule is applied in an evolution step only if no rule of a higher priority can be applied or has been already applied during that step.

**Definition 3.3** Let $R$ be a set of evolution rules together with a priority relation $\succ$. Then we say that $w$ *maximally parallel rewrites* in $w'$ w.r.t. $R$, write $w \Rrightarrow_R w'$, iff $w =_{ACU} u_1 \cdots u_n z$, $w' =_{ACU} v_1 \cdots v_n z$, $\ell_i : u_i \to v_i$ is a rule in $R$ for $i = 1, \ldots, n$, $n > 0$, $z$ is $R$-irreducible, and for any $\ell : u \to v$ in $R$ and for any $i$, if $\ell \succ \ell_i$, $w$ is $\ell$-irreducible.

**Definition 3.4** Let $R$ be a set of evolution rules together with a priority relation $\succ$ and $\{\ell_i : u_i \to v_i \mid i = 1, \ldots, n\}$ the subset of the rules with maximal priority. Then the strategy $pri(R)$ is defined as follows:

$$
\begin{aligned}
&pri(R) \stackrel{\text{def}}{=} s_1 + \cdots + s_n,\\
&s_i = \_\_[\texttt{assoc comm id}:\ \varepsilon](\ell_i, pri(filter(R, \ell_i)) \leftarrow\!\!+ \mathrm{id}) \leftarrow\!\!+ pri(R \setminus \{l_i\})\\
&\text{for } i = 1, \ldots, n,
\end{aligned}
$$

where $filter(R, \ell_i)$ is obtained from $R$ by removing all the rules having priority less than $\ell_i$.
If $R = \emptyset$, then $pri(\emptyset) = \texttt{fail}$.

For instance, let us suppose that $R$ consists of $\ell_1 : a \to c \succ \ell_2 : b \to d$.
We have:

$$
\begin{aligned}
&pri(R) \stackrel{\text{def}}{=} s_1 \quad (\ell_1 \text{ is the only maximal element in } R)\\
&s_1 = \_\_[\texttt{assoc comm id}:\ \varepsilon](\ell_1, pri(\ell_1) \leftarrow\!\!+ \mathrm{id}) \leftarrow\!\!+ pri(\ell_2)\\
&pri(\ell_1) \stackrel{\text{def}}{=} \_\_[\texttt{assoc comm id}:\ \varepsilon](\ell_1, pri(\ell_1) \leftarrow\!\!+ \mathrm{id})\\
&pri(\ell_2) \stackrel{\text{def}}{=} \_\_[\texttt{assoc comm id}:\ \varepsilon](\ell_2, pri(\ell_2) \leftarrow\!\!+ \mathrm{id})
\end{aligned}
$$

The proof of $aab \Rrightarrow ccb$ is:

$$
\dfrac{
\dfrac{
\dfrac{S_{14}\ \dfrac{S_0\ b \xrightarrow{\mathrm{id}} b \qquad b \xrightarrow{pri(\ell_1)} \uparrow}{b \xrightarrow{pri(\ell_1)+\mathrm{id}} b} \qquad a \xrightarrow{\ell_1} c}{ab \xrightarrow{\_\_[\texttt{assoc comm id:}\ \varepsilon](\ell_1, pri(\ell_1)+\mathrm{id})} cb}\ S_{26}}{ab \xrightarrow{pri(\ell_1)} cb}\ S_{18}
\qquad
\dfrac{
\dfrac{
\dfrac{S_{13}\ ab \xrightarrow{pri(\ell_1)} cb}{ab \xrightarrow{pri(\ell_1)+\mathrm{id}} cb} \qquad a \xrightarrow{\ell_1} c}{aab \xrightarrow{\_\_[\texttt{assoc comm id:}\ \varepsilon](\ell_1, pri(\ell_1)+\mathrm{id})} ccb}\ S_{26}}{aab \xrightarrow{\_\_[\texttt{assoc comm id:}\ \varepsilon](\ell_1, pri(filter(R,\ell_1))+\mathrm{id})} ccb}\ S_{18}
\qquad (=)\ aab \xrightarrow{pri(R)} ccb
$$

**Theorem 3.5** *Given a set $R$ of evolution rules together with a priority relation $\succ$, then $w \Rrightarrow_R w'$ iff $(\Sigma, Ax, R) \vdash w \xrightarrow{pri(R)} w'$.*

**Proof.** The main idea is similar to that of Theorem 3.2. The correct handling of the priorities is assured by the following facts:

- only rules with maximal priority are applied,
- once a rule is applied, all rules having smaller priorities are removed from the current set of rules by the operator $filter$ , and
- if a rule with a maximal priority cannot be applied then it is removed.

□

### *3.3 Strategic Semantics of Maximal Parallel Rewriting with Promoters*

In a *P system with promoters* ([3]), the set of evolution rules $R$ for a region is given by $(R_1, p_1), \ldots, (R_m, p_m)$, $m \geq 1$, where $R_1, \ldots, R_m$ are finite and disjoint sets of rules and $p_1, \ldots, p_m$ are multisets of objects called *promoters*. The rules from the set $R_i$ can be used only if the multiset represented by $p_i$ is present in the region, for all $1 \leq i \leq m$. If several promoting multisets $p_{i_1}, \ldots, p_{i_k}$ are present in the region, then only the rules from $R_{i_1}, \ldots, R_{i_k}$ are enabled.

An *evolution rule with promoter* is a rewrite rule of the form $\ell : u \to v|_p$, where the promoter $p$ does not necessarily occur in $u$. Such a rule can be applied in an evolution step $w \Rrightarrow$ only if the promoter is present in $w$. The problem we try to solve in this section is if the rules with promoters can be implemented with strategies. Starting from this intuitive definition, we may be tempted to implement a rule with promoter by the following strategy:

$$\_\_[\texttt{assoc comm id}:\varepsilon](\mathrm{id}(p), \ell, \mathrm{id})$$

where $\mathrm{id}(p)$ is the strategy corresponding to the rule $\mathrm{id}(p) : p \to p$; the role of this rule is to test the presence of the promoter $p$. Note that a single occurrence of the promoter can be used by more than one rule or even the promoter can be consumed by some other rule, and the presence of the promoters makes it possible to use a rule from the associated set as many times as possible, without any restriction. For instance, let $R$ consist of the following rules with promoters: $\ell_1 : aq \to c|_p$ and $\ell_2 : bp \to d|_q$. Consider $s$ a strategy applying the rules $R$ over $abpq$. If $s$ applies first $\ell_1$, then the information that we initially had the promoter $p$ is lost and it does not know that $\ell_2$ can be applied. If $s$ applies first $\ell_2$, then the information that we initially had the promoter $q$ is lost and it does not know that

$\ell_1$ can be applied. Therefore we claim that there is no a strategy expressed in terms of rules $R$ and strategy operators which implements the maximal rewriting with promoters. A more powerful mechanism is needed. In this section we show that an encoding together with the strategies over the corresponding encoded rules are enough for implementing the maximal rewriting with promoters.

Given a set $R$ of evolution rules with or without promoters, we construct a set $\widehat{R}$ of rewrite rules and a strategy $prom(\widehat{R})$ such that $w \Rrightarrow_R w'$ iff $w \xrightarrow{prom(\widehat{R})} w'$. Let $pset(R, w)$ denote the set of promoters occurring in $w$ w.r.t. $R$. The set $\widehat{R}$ consists of the following rules:

- $\texttt{compute} : w \to (w, pset(R, w))$,
- $\texttt{forget} : u(w, s) \to uw$, together with
- a rule $\hat{\ell} : (wu, s) \to v(w, s)$ for each rule without promoter $\ell : u \to v$ in $R$, and
- a rule $\hat{\ell} : (wu, ps) \to v(w, ps)$ for each rule with promoter $\ell : u \to v|_p$ in $R$.

The rule `compute` stores the set of promoters occurring in $w$ as the second component of the pair and it remains unchanged during the application of the evolution rules. This information is used by the rules with promoters: such a rule is applied only if its promoter is present in the second component. Note that the processed part lies in the front of the pair and it is not affected by the next evolution rules applied in the current step; the evolution rules consumes only from the first component of the pair.

Since the components of a pair are multisets (defined as commutative strings), we have to consider a new strategy for rules $\hat{\ell}$. We denote by $\hat{\ell}(AX)$ the strategy of applying the rule $\hat{\ell}$ using an matching algorithm modulo the axioms $AX$ for left hand-side. In our case, $AX$ is the set of axioms given by associativity, commutativity, and the identity of the concatenation.

**Definition 3.6** We suppose that $R = \{\ell_i : u_i \to v_i \mid i = 1, \ldots, n\}$ and let $\widehat{R}$ be computed as above. Then the strategy $prom(\widehat{R})$ is

$$prom(\widehat{R}) \stackrel{\text{def}}{=} \texttt{compute}; \texttt{reduce}(s_1 + \cdots + s_n); \texttt{forget}$$
$$s_i = \hat{\ell}_i(ax(\_\_[\texttt{assoc comm id}: \ \varepsilon])) \text{ for } i = 1, \ldots, n.$$

For the example given above, $\widehat{R}$ consists of `compute`, `forget`, together with $\hat{\ell}_1 : (aqw, ps) \to c(w, ps)$ and $\hat{\ell}_2 : (bpw, qs) \to d(w, ps)$. The proof of $abpq \xrightarrow{prom(\widehat{R})} cd$ is given by the following facts:

Fact 1:

$$\dfrac{\dfrac{\dfrac{(bp,pq) \xrightarrow{s_2} d(\varepsilon,pq)}{c(bp,pq) \xrightarrow{\Diamond(s_1+s_2)} cd(\varepsilon,pq)}\,S_{11};S_{21}}{c(bp,pq) \xrightarrow{\mu x(\Diamond(x)+(s_1+s_2))} cd(\varepsilon,pq)}\,S_{16} \qquad (\varepsilon,pq) \xrightarrow{\mu x(\Diamond(x)+(s_1+s_2))} \uparrow}{c(bp,pq) \xrightarrow{\texttt{repeat}(\mu x(\Diamond(x)+(s_1+s_2)))} cd(\varepsilon,pq)}\,\textsf{repeat}$$

Fact 2:

$$\dfrac{\dfrac{(abpq,pq) \xrightarrow{s_2} c(bp,pq)}{(abpq,pq) \xrightarrow{\Diamond(s_1+s_2)} c(bp,pq)}\,S_{11};S_{21}}{(abpq,pq) \xrightarrow{\mu x(\Diamond(x)+(s_1+s_2))} c(bp,pq)}\,S_{16}$$

Fact 3:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{c(bp,pq) \xrightarrow{\texttt{repeat}(\mu x(\Diamond(x)+(s_1+s_2)))} cd(\varepsilon,pq) \qquad (abpq,pq) \xrightarrow{\mu x(\Diamond(x)+(s_1+s_2))} c(bp,pq)}{(abpq,pq) \xrightarrow{\mu x(\Diamond(x)+(s_1+s_2));\texttt{repeat}(\mu x(\Diamond(x)+(s_1+s_2)))} cd(\varepsilon,pq)}\,\textsf{repeat}}{(abpq,pq) \xrightarrow{\texttt{repeat}(\mu x(\Diamond(x)+(s_1+s_2)))} cd(\varepsilon,pq)}\,S_{19}}{(abpq,pq) \xrightarrow{\texttt{reduce}(s_1+s_2)} cd(\varepsilon,pq) \quad abpq \xrightarrow{\texttt{compute}} (abpq,pq)}\,S_7}{abpq \xrightarrow{prom(R')} cd(\varepsilon,pq) \quad cd(\varepsilon,pq) \xrightarrow{\texttt{forget}} cd}\,S_7}{abpq \xrightarrow{prom(R')} cd}$$

We denote by $(\widehat{\Sigma}, \widehat{Ax})$ the specification over which $\widehat{R}$ is defined.

**Theorem 3.7** *Given a set $R$ of evolution rules with promoters, then*

$$w \Rrightarrow_R w' \text{ iff } (\widehat{\Sigma}, \widehat{Ax}, \widehat{R}) \vdash w \xrightarrow{prom(\widehat{R})} w'.$$

**Proof.** We have $w \Rrightarrow_R w'$ iff $(w, pset(R,w)) \xrightarrow{\texttt{reduce}(s_1+\cdots+s_n)} (w', pset(R,w))$. If $w \Rrightarrow_R w'$, then it follows that $w =_{ACU} u'_1 \cdots u'_k z$, $w' =_{ACU} v'_1 \cdots v'_k z$, either $\ell'_j : u'_j \to v'_j$ or $\ell'_j : u'_j \to v'_j|_p$ is a rule in $R = \{\ell_i : u_i \to v_i \mid i = 1, \ldots, n\}$ for $j = 1, \ldots, k$, $k > 0$, and $z$ is $R$-irreducible. If $\ell'_j : u'_j \to v'_j|_p$ is a rule with the promoter $p$, then $p$ occurs in $w$. We prove that $w \xrightarrow{prom(\widehat{R})} w'$ by induction on $k$. Conversely, if $(w, pset(w)) \xrightarrow{\texttt{reduce}(s_1+\cdots+s_n)} (w', pset(R,w))$ then we prove that $w \Rrightarrow_R w'$ by induction on the depth of the proof tree. □

### *3.4 Strategic Semantics of Maximal Parallel Rewriting with Inhibitors*

An *evolution rule with inhibitor* is a rewrite rule of the form $\ell : u \to v|_{\neg p}$. Such a rule can be applied in an evolution step $w \Rrightarrow w'$ only if the inhibitor is not present in $w$.

We proceed in a similar way to that of promoters but taking in the rule `compute` the complementary set of inhibitors occurring in $w$ w.r.t. the whole set of objects $A$, denoted by $iset(A,w)$ instead of $pset(R,w)$, and considering rules with inhibitors instead of rules with promoters:

- `compute` : $w \to (w, iset(A,w))$,
- `forget` : $w'(w,s) \to w'w$, together with
- a rule $\hat{\ell} : w'(wu,s) \to w'v(w,s)$ for each rule $\ell : u \to v$ without inhibitor, and

- a rule $\hat{\ell} : w'(wu, ps) \to w'v(w, ps)$ for each rule $\ell : u \to v|_{\neg p}$ with inhibitor,

where $w', w$ range over configurations, and $s$ range over sets of inhibitors.

**Definition 3.8** We suppose that $R = \{\ell_i : u_i \to v_i \mid i = 1, \ldots, n\}$, and let $\widehat{R}$ be computed as above. Then the strategy $inhib(\widehat{R})$ is

$$inhib(\widehat{R}) \stackrel{\text{def}}{=} \texttt{compute}; \texttt{repeat}(\hat{\ell}_1 + \cdots + \hat{\ell}_n); \texttt{forget}$$

**Theorem 3.9** *Given a set $R$ of evolution rules with inhibitors, then*

$$w \Rrightarrow_R w' \text{ iff } w \xrightarrow{inhib(\widehat{R})} w'.$$

### *3.5 Strategic Semantics of Maximal Parallel Rewriting with Promoters and Inhibitors*

When we have rules involving both promoters and inhibitors, we encode a configuration by a triple $(w, pset(R, w), iset(A, w))$ in order to have information about both promoters and inhibitors. The set $\widehat{R}$ includes the following rules:

- $\texttt{compute} : w \to (w, pset(R, w), iset(A, w))$,
- $\texttt{forget} : w'(w, s, s') \to w'w$, together with
- a rule $\hat{\ell} : w'(wu, s, s') \to w'v(w, s, s')$ for each rule $\ell : u \to v$ in $R$ without promoter or inhibitor,
- a rule $\hat{\ell} : w'(wu, ps, s') \to w'v(w, ps, s')$ for each rule $\ell : u \to v|_p$ in $R$ with promoter, and
- a rule $\hat{\ell} : w'(wu, s, ps') \to w'v(w, s, ps')$ for each rule $\ell : u \to v|_{\neg p}$ in $R$ with inhibitor.

This encoding is general, and it can be used even if one or both sets of promoters and inhibitors are empty.

**Definition 3.10** We suppose that $R = \{\ell_i : u_i \to v_i \mid i = 1, \ldots, n\}$, and let $\widehat{R}$ be computed as above. Then the strategy $prominhib(\widehat{R})$ is

$$prominhib(\widehat{R}) \stackrel{\text{def}}{=} \texttt{compute}; \texttt{repeat}(\hat{\ell}_1 + \cdots + \hat{\ell}_n); \texttt{forget}$$

**Theorem 3.11** *Given a set $R$ of rules involving eventually promoters and inhibitors, then*

$$w \Rrightarrow_R w' \text{ iff } w \xrightarrow{prominhib(\widehat{R})} w'.$$

## 4 Deriving Tactical Rewritings by Continuation Semantics

In this section we deal with the following problem: given a term $t$ and a strategy $s$, find a tactical rewriting $t \rightsquigarrow t'$ that accommodate with $s$, i.e., $t \rightsquigarrow t' \models t \xrightarrow{s} t'$, if any. This tactical rewriting is given by a mechanism based on *continuation passing style (CPS)*. It consists in a transition relation $\Rightarrow$ over pairs (*term*, *strategy*) with the meaning *strategy* describes what steps follows to be executed starting from *term*.

The transition relation is (partially) defined as follows:

(i) $(t, \ell) \Rightarrow (t', \mathrm{id})$, where $t'$ is defined as in $S_0$;

(ii) $(t, s_1; s_2) \Rightarrow (t', s_1'; s_2)$ if $(t, s_1) \Rightarrow (t', s_1')$;

(iii) $(t, s_1 + s_2) \Rightarrow (t', s_1')$ if $(t, s_1) \Rrightarrow (t', s_1')$;

(iv) $(t, s_1 + s_2) \Rightarrow (t', s_2')$ if $(t, s_2) \Rightarrow (t', s_2')$;

(v) $(t, s_1 \nleftarrow s_2) \Rightarrow (t', s_1')$ if $(t, s_1) \Rrightarrow (t', s_1')$;

(vi) $(t, s_1 \nleftarrow s_2) \Rightarrow (t', s_2')$ if $(t, s_2) \Rightarrow (t', s_2')$ and $(t, s_1) \Rightarrow (t', \texttt{fail})$;

(vii) $(t, \varphi(s_1, \ldots, s_n)) \Rightarrow (t, s(z_1 := s_1, \ldots, z_n := s_n))$ if $\varphi(z_1, \ldots, z_n) \stackrel{\text{def}}{=} s$;

(viii) $(f(t_1, \ldots, t_i, \ldots, t_n), i(s)) \Rightarrow f(t_1, \ldots, (t_i, s), \ldots, t_n)$

(ix) $(f(t_1, \ldots, t_n), f(s_1, \ldots, s_n)) \Rightarrow f((t_{i_1}, s_1), \ldots, (t_{i_n}, s_n))$
if $f(t_1, \ldots, t_n) =_{ax(attr)} f(t_{i_1}, \ldots, t_{i_n})$

(x) $(f(t_1, \ldots, t_n), \Diamond(s)) \Rightarrow f(t_1, \ldots, (t_i, s), \ldots, t_n)$

(xi) $(f(t_1, \ldots, t_n), \boxdot(s)) \Rightarrow f((t_1, s \nleftarrow \mathrm{id}), \ldots, (t_i, s), \ldots, (t_n, s \nleftarrow \mathrm{id}))$

If $R$ is that given by the example in Section 3.1, then we can derive the following sequence of transitions starting from the pair $(aabbb, mpr(R))$:

$$
\begin{array}{l}
(aabbb, mpr(R)) \Rightarrow (aabbb, s_1 + s_2) \Rightarrow \\
(aabbb, \_\_[ACU](\ell_1, mpr(R) \nleftarrow \mathrm{id})) \Rightarrow (ab, \ell_1)(abb, mpr(R) \nleftarrow \mathrm{id}) \Rightarrow \\
(c, \mathrm{id})(abb, (s_1 + s_2) \nleftarrow \mathrm{id}) \Rightarrow (c, \mathrm{id})(abb, \_\_[ACU](\ell_2, mpr(R) \nleftarrow \mathrm{id})) \Rightarrow \\
(c, \mathrm{id})(bb, \ell_2)(a, mpr(R) \nleftarrow \mathrm{id}) \Rightarrow (c, \mathrm{id})(d, \mathrm{id})(a, mpr(R) \nleftarrow \mathrm{id}) \Rightarrow \\
(c, \mathrm{id})(d, \mathrm{id})(a, id)
\end{array}
$$

From this transition sequence it is easy to extract the tactical rewriting $aabbb \rightarrow cabb \rightarrow cda$ for $aabbb \xrightarrow{mpr(R)} cda$.

## 5 Conclusion

We think that the main contribution of the paper is given by the novel use of strategies and tactics in defining the operational semantics of some systems where many parallel rules are applied in a single step, and with complex relationships between elementary operational entities (rules) and resources. Our strategy semantics approach is related but different to that presented by Visser [9]. We do not know a similar approach. Systems as Tom [6] and Stratego [8] use the operational semantics given by strategies à la Visser. ELAN [5] uses special strategies in order to express and manipulate non-determinism of type "choice", "don't care", or "don't know". The strategies can be implemented in Maude by reflection [4]; moreover reflection allows to implement encodings of rules and therefore Maude is a suitable candidate for implementing the strategies described in this paper.

A specific contribution is given in membrane computing by defining a formal semantics for maximal parallel rewriting by using strategies and tactics. Several results and examples of the paper explain how the strategies and tactics can be use in describing the strategy

semantics of the membrane systems. Given a set of rules of such a system, we investigate whether we have a strategy able to express faithfully the maximal parallel rewriting. Moreover, we show how tactics corresponding to a given strategy could be derived by using a continuation-passing semantics.

The limits of the strategies and tactics in operational semantics are detected when we describe the maximal rewriting of membrane systems involving promoters. Other membrane systems can lead to new interesting results in strategy semantics.

## References

[1] Andrei, O., G. Ciobanu and D. Lucanu, *Executable Specifications of the P Systems*, Lecture Notes in Computer Science **3365**, 127–146, Springer, 2005.

[2] Andrei, O., G. Ciobanu and D. Lucanu, *A Structural Operational Semantics of the P Systems*, Lecture Notes in Computer Science **3850**, 32–49, Springer, 2006.

[3] P. Bottoni, C. Martín-Vide, Gh. Păun, G. Rozenberg. Membrane systems with promoters/inhibitors. In *Acta Informatica*, vol. 38(10):695–720, 2002.

[4] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, J.F. Quesada. Maude: Specification and Programming in Rewriting Logic. *Theoretical Computer Science* 285:187-243, 2002.

[5] H. Kirchner, P-E. Moreau. Promoting rewriting to a programming language: a compiler for non-deterministic rewrite programs in associative-commutative theories. *Journal of Functional Programming*, 11:207-251, 2001.

[6] P-E. Moreau, C. Ringeissen, M. Vittek. A Pattern Matching Compiler for Multiple Target Languages. In *12th Conference on Compiler Construction*, Lecture Notes in Computer Science **2622**, 61–76, Springer, 2003.

[7] Gh. Păun. *Membrane Computing. An Introduction*. Springer, 2002.

[8] E. Visser. Stratego: A language for program transformation based on rewriting strategies. In *Rewriting Techniques and Applications (RTA'01)*, Lecture Notes in Computer Science **2051**, 357–361, Springer, 2001.

[9] E. Visser. A Survey of Strategies in Rule-Based Program Transformation Systems. *Journal of Symbolic Computation* 40:831-873, 2005.

[10] E. Visser, Z.-e.-A. Benaissa, A. Tolmach. Building program optimizers with rewriting strategies. *ACM SIGPLAN Notices*, 34:13–26, 1999.