P Systems with Antiport Rules for Evolution Rules

Rudolf FREUND, Marion OSWALD

Faculty of Computer Science
Vienna University of Technology
Favoritenstr. 9–11, A–1040 Vienna, Austria
E-mail: {rudi,marion}@emcc.at

Abstract. We investigate a variant of evolution-communication P systems where the computation is performed in two substeps. First, all possible antiport rules are applied in a non-deterministic, maximally parallel way, moving evolution rules across membranes. In the second substep, evolution rules are applied to suitable objects in a maximally parallel way, too. Thus, objects can be the subject of change, but are never moved themselves. As result of a halting computation, we consider the multiset of objects present in a designated output membrane. When using catalytic evolution rules, we already obtain universal computational power with only one catalyst and one membrane. For systems without catalysts we obtain a characterization of the Parikh images of ET0L languages.

1 Introduction

P systems were introduced in [8] as bio-inspired computing devices that work in a parallel and distributed way (see [9] for a comprehensive overview and [11] for actual developments in the area). In [1], a variant called evolution-communication P systems was investigated, combining evolution and communication rules for objects. Following this idea but moving evolution rules instead of objects, P systems with symport/antiport of rules were presented in [2]. We here take a similar approach: starting from the initial configuration of a P system with antiport rules for evolution rules, a transition from one configuration to the next one is, other than in [2], performed in two substeps: in the first substep evolution rules can be interchanged across a membrane by antiport rules (that are assigned to the membranes, see e.g., [6]), whereas in the second substep, objects can evolve by evolution rules in a maximally parallel manner. This means that in this model only the evolution rules but no objects are moved. If the computation halts, i.e., neither antiport nor evolution rules can be applied anymore, the result is given as the multiset of objects present in the output membrane.

In the following section we first give some preliminary definitions and recall some notions for graph-controlled grammars, the computation model we use for proving the results elaborated in this paper; in the third section we introduce P systems with antiport rules for evolution rules. In the fourth section we show that when using purely catalytic evolution rules (i.e., every evolution rule involves a catalyst, as introduced in [7]), the

introduced systems can simulate graph-controlled grammars quite easily, which proves their universal computational power. A characterization of Parikh sets of ET0L languages is obtained when considering systems without catalysts, which is shown in section five where we also give an illustrative example. A short summary and an outlook to future research conclude the paper.

2 Preliminary Definitions

The set of non-negative integers is denoted by $\mathbf{N}$. An *alphabet* V is a finite non-empty set of abstract *symbols*. Given V, the free monoid generated by V under the operation of concatenation is denoted by V^*; the *empty string* is denoted by λ, and $V^* \setminus \{\lambda\}$ is denoted by V^+. A multiset over V is represented as string over V (and any of its permutations); a set over V is represented as a string, too, but with each element of V occurring at most once. By $\mid x \mid$ we denote the length of the word x over V as well as the number of elements in the multiset represented by x.

A *graph-controlled grammar* is a construct

$$G = (N, T, Lab, S, R, \{1\}, \{n\}),$$

where N denotes the set of non-terminals, T is the set of terminal symbols, $Lab = \{1, ..., n\}$ is the set of labels, S is the start symbol, R is a finite set of rules that can be represented as a function from Lab to $P \times 2^{Lab} \times 2^{Lab}$, where P denotes the set of all context-free productions over the set N of non-terminal symbols and the set of terminal symbols T. A rule in R usually is written in the form

$$(i : p(i), \sigma(i), \varphi(i)),$$

where $\sigma(i)$ is called the success field and $\varphi(i)$ is called the failure field of the rule labelled by i; the context-free production $p(i)$ is of the form $A(i) \rightarrow w(i)$, where $A(i) \in N$ and $w(i) \in (N \cup T)^*$. Without loss of generality we not only assume that $N \cap Lab = \emptyset$ and that there is only one initial label (i.e., 1) and only one final label (i.e., n, with $\sigma(n) = \varphi(n) = \emptyset$), but we also may assume that if a computation has reached the final label n, then the obtained sentential form is terminal, i.e., it must not contain any non-terminal symbol.

Finally, again without loss of generality we may also assume that in the case of a string language, the terminal symbols are generated by G exactly in the correct sequence as they form a terminal word. All these features of a normal form for graph-controlled grammars, for example, follow from the constructions and results proved in [5], Theorem 6.

An *ET0L system* is a construct $G_L = (V, T, t_1, ..., t_n, w)$ where V is the *alphabet*, $T \subseteq V$ is the *terminal alphabet*, and we define $N := V \setminus T$; $t_i, 1 \leq i \leq n$, are lists of context-free rules that are called *tables* and $w \in V^*$ is the *axiom*. All tables have to fulfill the condition that for every $a \in V$ there is at least one production in $t_i, 1 \leq i \leq n$.

A derivation step in the ET0L system consists of the parallel application of the rules in one of the tables t_i. The language generated by G_L consists of all terminal words being derivable from the axiom w in an arbitrary number of derivation steps using the tables t_i. The family of all languages of Parikh vectors generated by ET0L systems is denoted by $PsET0L$.

A *programmed ET0L system* is a pair (G_L, σ), where $G_L = (V, T, t_1, ..., t_n, w)$ is an ET0L *system* and σ is a function which to each table $t_i, 1 \leq i \leq n$, assigns a finite subset of the set of tables.

The language $L(G_L)$ of a programmed ET0L system G_L consists of all words x which have a derivation

$$w \underset{t_{i_1}}{\Longrightarrow} w_1 \underset{t_{i_2}}{\Longrightarrow} w_2 \underset{t_{i_3}}{\Longrightarrow} ... \underset{t_{i_{n-1}}}{\Longrightarrow} w_{n-1} \underset{t_{i_n}}{\Longrightarrow} w_n = x$$

such that $t_{i_{j+1}} \in \sigma\left(t_{i_j}\right)$ for $1 \leq j < n$.

The family of all languages of Parikh vectors generated by programmed ET0L systems is denoted by $PsEpT0L$.

According to Theorem 8.3 in [3], $PsET0L = PsEpT0L$.

For more details on formal language theory we refer to [10] and [3].

3 P Systems with Antiport Rules for Evolution Rules

A *P system with antiport rules for evolution rules* is a construct Π defined as follows:

$$\Pi = (O, C, R, l, \mu, w_1, ..., w_m, R_1, ..., R_m, i_0),$$

where:

- O is an alphabet of *objects*;
- C is a set of catalysts and $O \cap C = \emptyset$;
- R is a finite set of evolution rules;
- $l : L \to R$ is an injective labelling of the rules in R; L is called the set of labels with $L \cap (O \cup C) = \emptyset$;
- μ is a *membrane structure* with m membranes bijectively labelled by $1, ..., m$;
- $w_i, 1 \leq i \leq m$, are finite multisets over O together with finite sets over L (altogether represented as strings over $O \cup L$) associated with the regions $1, ..., m$ of μ;
- $R_i \subseteq R, 1 \leq i \leq m$, are finite sets of antiport rules that are associated with the membranes $1, ..., m$ of μ of the form $(j, in; k, out)$, where j, k are sets of labels of evoution rules from R.
- i_0 is the label of the output membrane.

The rules in R_i can easily be represented in a more depictive way: for $(i, in; j, out) \in R_k$ we simply write ${}_i \underset{k}{\leftrightarrows}^{j}$. If we list the antiport rules of Π in this depictive way, instead of the sequence $R_1, ..., R_m$ we only include the set R_A containing the list of antiport rules. The weight of an antiport rule $(i, in; j, out)$ is defined as $\max\{|i|, |j|\}$.

In the environment we assume every label of an evolution rule to occur. The initial configuration of such a system consists of μ as well as of $w_1, ..., w_m$. A transition from one configuration to the next one is performed in two substeps:

1. The antiport rules from $R_1, ..., R_m$ (respectively R_A) are applied in a non-deterministic maximally parallel way across all membranes;

2. The evolution rules are applied to suitable objects in a non-deterministic maximally parallel way within all membranes (using the catalysts in a multiset manner).

As long as either antiport rules or evolution rules can be applied, the system proceeds performing these two substeps. Only if neither antiport rules nor evolution rules can be applied anymore, we say that the systems halts and consider the multiset of objects present in the designated output membrane in the final configuration to be the result (a non-halting computation does not produce a result).

According to [7] (also see [4]), the P system with antiport rules for evolution rules Π is called *catalytic,* if every evolution rule involves a catalyst.

The family of recursively enumerable Parikh sets generated by P system with antiport rules for evolution rules with m membranes and k catalysts (counted in the multiset sense over the whole system) and with the weight of the antiport rules being at most l is denoted by $PsOPA_m\left(cat_k, anti_l\right)$; if only catalytic evolution rules are used, we write $PsOPA_m\left(purecat_k, anti_l\right)$; if $k = 0$,we simply write $PsOPA_m\left(anti_l\right)$; if one of the parameters k, l or m is not bounded, it is replaced by $*$.

4 Catalytic P Systems with Antiport Rules for Evolution Rules

Theorem 1 *Let $L \subseteq \mathbf{N}^k$, $k \geq 1$, be a recursively enumerable set of (vectors of) non-negative integers. Then L can be generated by a P system with antiport rules for purely catalytic evolution rules in one membrane using one catalyst, i.e.,*

$$PsOPA_1\left(purecat_1\right) = PsRE.$$

Proof. Let $G = (N, T, Lab, S, R_G, \{1\}, \{n\})$ be a graph-controlled grammar. Then we construct the P system with antiport rules for evolution rules Π simulating G as follows. We start with the initial configuration containing a catalyst c, the start symbol S and an additional object Z as well as the labels for three evolution rules r_0, r, r', where r introduces the trap symbol # that causes an infinite loop by r', whereas r_0 simply starts the simulation of a derivation in G. Moreover, we define an injective mapping $l : L \to R$ so that every production in R_G has a different label in l.

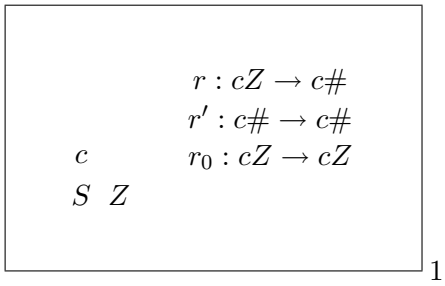

Figure 1: Initial configuration of Π.

The simulation of a derivation in G works as follows. We start from the initial configuration that is illustrated in figure 1. To simulate a rule $(i : p(i), \sigma(i), \varphi(i))$, where $p(i)$ is of the form $A(i) \to w(i)$, $A(i) \in N$ and $w(i) \in (N \cup T)^*$, we take the labelled evolution rules $\widetilde{i} : cA(i) \to cw$ as well as $\widehat{i} : cA(i) \to c\#$ and proceed as follows.

- To start a simulation we take ${}_{\widetilde{1}} \underset{1}{\leftrightarrows}^{r_0}$ as well as ${}_{\widehat{1}} \underset{1}{\leftrightarrows}^{r_0}$.
- To be able to communicate the correct sequence of evolution rules, we need the following antiport rules in R_A :
  1. ${}_{\widetilde{s}} \underset{1}{\leftrightarrows}^{\widetilde{t}}$ as well as ${}_{\widehat{s}} \underset{1}{\leftrightarrows}^{r\widetilde{t}}$ for $s \in \sigma(t) \setminus \{n\}$.
  2. ${}_{r\widetilde{s}} \underset{1}{\leftrightarrows}^{\widehat{t}}$ as well as ${}_{\widehat{s}} \underset{1}{\leftrightarrows}^{\widehat{t}}$ for $s \in \varphi(t) \setminus \{n\}$.
  3. ${}_{f} \underset{1}{\leftrightarrows}^{r\widetilde{t}}$ as well as ${}_{f} \underset{1}{\leftrightarrows}^{\widehat{t}}$ for the case that the final label n is either in $\sigma(t)$ or in $\varphi(t)$, where $f : cZ \to c$.

In sum, we obtain the following P system with antiport rules for evolution rules Π :

$$
\begin{array}{lcl}
\Pi & = & (V, \{c\}, R, l, [_1\ ]_1, cZSrr'r_0, R_A, 1), \\
V & = & N \cup T \cup \{Z, \#\}, \\
R & = & \{cZ \to c\#, c\# \to c\#, cZ \to cZ, cZ \to c\} \\
 & \cup & \{cA(i) \to cw(i) \mid (i : A(i) \to w(i), \sigma(i), \varphi(i)) \in R_G\}, \\
l & = & \{r : cZ \to c\#, r' : c\# \to c\#, r_0 : cZ \to cZ, f : cZ \to c\} \\
 & \cup & \left\{\widetilde{i} : cA(i) \to cw(i), \widehat{i} : cA(i) \to c\# \mid (i : A(i) \to w(i), \sigma(i), \varphi(i)) \in R_G\right\}, \\
R_A & = & \left\{{}_{\widetilde{1}} \underset{1}{\leftrightarrows}^{r_0}, {}_{\widehat{1}} \underset{1}{\leftrightarrows}^{r_0}\right\} \\
 & \cup & \left\{{}_{\widetilde{s}} \underset{1}{\leftrightarrows}^{\widetilde{t}}, {}_{\widehat{s}} \underset{1}{\leftrightarrows}^{r\widetilde{t}} \mid s \in \sigma(t) \setminus \{n\}, t \in Lab\right\} \\
 & \cup & \left\{{}_{r\widetilde{s}} \underset{1}{\leftrightarrows}^{\widehat{t}}, {}_{\widehat{s}} \underset{1}{\leftrightarrows}^{\widehat{t}} \mid s \in \varphi(t) \setminus \{n\}, t \in Lab\right\} \\
 & \cup & \left\{{}_{f} \underset{1}{\leftrightarrows}^{\widehat{t}}, {}_{f} \underset{1}{\leftrightarrows}^{r\widetilde{t}} \mid n \in \sigma(t) \cup \varphi(t)\right\}.
\end{array}
$$

The simulation of the labelled productions of G is controlled by the antiport rules assigned to membrane 1. If we guess that the rule labelled by i is applicable, we use $\widetilde{i}$, in the opposite case we take $\widehat{i}$. If we have guessed that the rule labelled by i is applicable, then in order not to enter an infinite loop by using $r : cZ \to c\#$, $\widetilde{i}$ has to be applied. If $\widehat{i}$ is taken in, then the trap symbol # can only be avoided if no symbol $A(i)$ occurs in membrane 1, but to guarantee the enforced application of $\widehat{i}$ (in contrast to the case $\widetilde{i}$) the rule r must not be present.

If in any moment the catalyst c is not used for $\widetilde{i}$ or is used for $\widehat{i}$, then we immediately introduce the trap symbol # which leads to an infinite loop (using $r' : c\# \to c\#$), hence no result is produced. On the other hand, whenever the final label n of G occurs in either the success or the failure field of a production having just been applied, then we can assume that the obtained sentential form is already terminal (see section 2). Hence it is correct to halt after the elimination of the symbol Z by applying the evolution rule $cZ \to c$ with the result being the objects contained in membrane 1. □

In the original variant of catalytic P systems without antiport rules for evolution rules two catalysts in one membrane are needed if we also allow non-catalytic rules; for purely catalytic systems only allowing for catalytic rules, three catalysts are needed as shown in [4].

Considering strings as objects being affected by the evolution rules, we even need no catalysts: The same proof as above with just omitting the catalyst c shows that any recursively enumerable set of strings can be generated by a P system with antiport rules for evolution rules working on strings in only one membrane (at each substep where the evolution rules have to be applied, to each string only one evolution rule is applied, yet only if no rule is applicable the string remains unchanged).

5 P systems with Antiport Rules for Evolution Rules without Catalysts

According to the original definition in [2] we consider the evolution rules occurring in the regions to be applied in a maximally parallel manner. Using (purely) catalytic rules, this parallelism is reduced, but as Theorem 1 shows this reduction of the parallelism yields maximal computational power. The importance of parallelism becomes visible when we deal with P systems with antiport rules for evolution rules without catalysts.

In order to show the influence of parallelism we give a simple example of a P system with antiport rules for evolution rules without catalysts:

Example 2 *Consider the following P system* Π :

$$\begin{aligned}
\Pi &= (V, R, l, [_1]_1, S1, R_A, 1)\,, \\
V &= \{S, a\}\,, \\
R &= \{S \to SS, S \to a\}\,, \\
l &= \{(1, S \to SS)\,, (2, S \to a)\}\,, \\
R_A &= \left\{ {}_1 \underset{1}{\leftrightarrows}^1, {}_2 \underset{1}{\leftrightarrows}^1 \right\}.
\end{aligned}$$

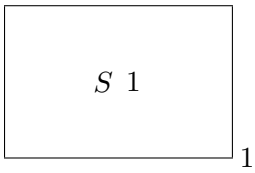

Figure 2: Initial configuration of Π.

From the initial configuration, which is also illustrated in figure 2, we start by applying an antiport rule from R_A. *Taking* ${}_1 \underset{1}{\leftrightarrows}^1$ *does not change the actual configuration of the system, so that in the next substep the evolution rule* $S \to SS$ *can be applied. The system can go on by repeatedly carrying out these two steps, thereby each time doubling the occurrences of* S. *If in any moment the antiport rule* ${}_2 \underset{1}{\leftrightarrows}^1$ *is used instead of* ${}_1 \underset{1}{\leftrightarrows}^1$, *then the rule* $S \to a$ *is brought into the system. In the next step, all objects* S *are changed into* a, *and no further rule can be applied anymore, hence the system halts.*

Observe that a rule occurring in a region has to be applied in the maximally parallel manner, hence, we obtain $2^n, n \geq 0$, objects a in the final configuration.

For P systems with antiport rules for evolution rules without catalysts we obtain a characterization of $PsET0L$, i.e.,

$$PsOPA_* (anti_*) = PsET0L.$$

This result will be proved in the following two lemmas:

Lemma 3 $PsET0L \subseteq PsOPA_1 (anti_*)$.

Proof. Let $G_L = (V, T, t_1, ..., t_n, w)$ be an ET0L system. Then we can construct a P system with antiport rules for evolution rules Π simulating G_L in the following way:

First we define two additional tables:

$$\begin{aligned} t_0 &= \{X \to X \mid X \in V\}, \\ t_{n+1} &= \{X \to X \mid X \in V \setminus T\}. \end{aligned}$$

We then define $R = \bigcup_{i=0}^{n+1} t_i$, which is the set of all context-free productions occurring in the tables t_i, $0 \leq i \leq n+1$.

Now let m be the cardinality of R and $L = \{0, 1, ..., m, m+1\}$. Moreover we define a bijective mapping $l : L \to R$; hence, we can represent each table t_i, $0 \leq i \leq n+1$, by the corresponding word r_i over L (observe that r_i contains any element of L at most once and that $|r_i| \geq card(V)$ for $0 \leq i \leq n$).

We now define the P system with antiport rules for evolution rules Π simulating the ET0L system G_L

$$\Pi = (V, R, l, [_1\ ]_1, wr_0, R_A, 1),$$

with

$$\begin{aligned} R_A &= \left\{ r_i \underset{1}{\leftrightarrows}^{r_0} \mid 1 \leq i \leq n \right\} \\ &\cup \left\{ r_j \underset{1}{\leftrightarrows}^{r_i} \mid 1 \leq i, j \leq n \right\} \\ &\cup \left\{ r_{n+1} \underset{1}{\leftrightarrows}^{r_i} \mid 0 \leq i \leq n \right\}. \end{aligned}$$

The label r_0 is just a dummy to allow us to start with the simulation of any of the tables from the ET0L system. The antiport rules in $\left\{ r_j \underset{1}{\leftrightarrows}^{r_i} \mid 1 \leq i, j \leq n \right\}$ allow us to continue with any arbitrary table t_j after the application of the table t_i. Finally, the table r_{n+1} coming into action after the application of any of the antiport rules in $\left\{ r_{n+1} \underset{1}{\leftrightarrows}^{r_i} \mid 0 \leq i \leq n \right\}$ allows us to check for the appearance of non-terminal symbols from N. If the underlying sentential form in G_L is terminal, then the corresponding configuration in Π halts. Hence, we have proved that the terminal derivations in G_L exactly correspond to the halting computations in Π, which observation concludes the proof. □

Due to our definition of enforced substeps, we only need one membrane instead of two membranes needed for the corresponding result proved in [2].

Example 4 *Taking again the previous example, the corresponding ET0L system (with the constraint that every table has to contain at least one rule for each symbol) would be*

$$(\{S,a\},\{a\},\{S\to SS, a\to a\},\{S\to a, a\to a\}).$$

According to the construction of the proof elaborated above, we would get the following P system with antiport rules for evolution rules:

$$\begin{array}{lcl}
\Pi & = & (O,R,l,[_1\ ]_1,S24,R_A,1),\\
O & = & \{S,a\},\\
R & = & \{S\to SS, a\to a, S\to a, S\to S\},\\
L & = & \{1,2,3,4\},\\
l & = & \{(1,S\to SS),(2,a\to a),(3,S\to a),(4,S\to S)\},\\
R_A & = & \left\{ {}_{12}\underset{1}{\leftrightarrows}{}^{24},{}_{23}\underset{1}{\leftrightarrows}{}^{24} \right\}\\
 & \cup & \left\{ {}_{12}\underset{1}{\leftrightarrows}{}^{12},{}_{23}\underset{1}{\leftrightarrows}{}^{12},{}_{23}\underset{1}{\leftrightarrows}{}^{23},{}_{12}\underset{1}{\leftrightarrows}{}^{23} \right\}\\
 & \cup & \left\{ {}_{4}\underset{1}{\leftrightarrows}{}^{24},{}_{4}\underset{1}{\leftrightarrows}{}^{12},{}_{4}\underset{1}{\leftrightarrows}{}^{23} \right\}.
\end{array}$$

Lemma 5 $PsOPA_*(anti_*) \subseteq PsEpT0L$.

Proof (Sketch). Let

$$\Pi = (O,R,l,\mu,w_1...w_m,R_A,i_0)$$

be an arbitrary P system with antiport rules for evolution rules without catalysts. We now construct a programmed ET0L system

$$G_L = (V,T,t_0,t_1,...,t_n,w,\sigma)$$

with $V = O\times H\cup O\cup\{\#\}$, where $H=\{1,...,m\}$ is the set of membrane labels, $T=O$, taking into accout the following observations.

- The contents of each membrane region in any configuration over Π can be represented as a string over the alphabet $O\times H$. The notation (a,i) for $a\in O, i\in H$, indicates that the object a occurs in membrane region i. Moreover, for each $i\in H$ we define the homomorphism $h_i: O\to O\times\{i\}$ such that $h_i(a)=(a,i)$ for all $a\in O$.
- There is only a finite number of possible distributions of the evolution rules over the whole membrane structure.
- It is easily decidable which of these distributions of evolution rules is reachable from the initial configuration.
- For each reachable distribution of evolution rules, it is decidable which distributions are possible after applying the given antiport rules from Π in a maximally parallel manner.
- For each reachable distribution d of evolution rules we now construct a table of context-free productions t_d for G_L in the following way:

  $t_d = t'_d\cup t''_d\cup\widetilde{t}$;

t'_d contains the context-free production $h_i(a) \to h_i(w)$ if and only if $a \to w$ is a rule contained in membrane i in the current distribution of evolution rules d;

t''_d contains the rule $(a,i) \to (a,i)$ if and only if there is no production in t'_d with (a,i) on its left-hand side (this guarantees the completeness condition for the tables in the ET0L system G_L);

$\widetilde{t} = \{a \to a \mid a \in O \cup \{\#\}\}$.

- For every reachable distribution of evolution rules d, $\sigma'(t_d)$ exactly contains all tables t_e such that e is reachable in one step from d in Π.
- To start the simulation of Π in G_L we encode the initial contents of all membrane regions in the initial configuration and use an additional table t_0 in G_L with

  $t_0 = \{(a,i) \to (a,i) \mid a \in O, i \in H\}$ and $\sigma(t_0) = \{t_{d_0}\}$

  where d_0 is the distribution of the evolution rules in the initial configuration.
- For all d with $\sigma'(t_d) \neq \emptyset$ we take $\sigma(t_d) = \sigma'(t_d)$. In that way the derivation steps in Π can be simulated in G_L.
- If $\sigma'(t_d) = \emptyset$, then this means that the distribution of the evolution rules in Π will not be changed anymore and only evolution rules may still be working. Hence, in that case we take $\sigma(t_d) = \left\{t_d, t_d^f\right\}$. At some point of the simulation we assume that no rule from t'_d can be applied anymore, i.e., the computation in Π is halting. This halting condition for Π is checked by applying the table t_d^f in G_L where $t_d^f = t_d^{f,1} \cup t_d^{f,2} \cup t_d^{f,3} \cup \widetilde{t}$ and

  $t_d^{f,1} = \{(a,i_0) \to a \mid a \in O,$ there is no rule for (a,i_0) in $t'_d\}$,

  $t_d^{f,2} = \{(X,j) \to \lambda \mid X \in O,$ there is no rule for (X,j) in $t'_d,\ j \in H \setminus \{i_0\}\}$,

  $t_d^{f,3} = \{(X,j) \to \# \mid X \in O,$ there is a rule for (X,j) in $t'_d,\ j \in H\}$.

  The tables of G_L as well as σ can easily be extracted from the constructions elaborated above, hence, the proof is complete. □

Combining the results proved in the preceding two lemmas with Theorem 8.3 from [3] (providing the equality $PsEpT0L = PsET0L$) we immediately obtain the following theorem, i.e., a characterization of $PsET0L$ by P systems with antiport rules for evolution rules without catalysts, but also a normal form for this kind of P systems telling us that one membrane is already sufficient:

Theorem 6 $PsOPA_1(anti_*) = PsOPA_*(anti_*) = PsEpT0L = PsET0L$.

6 Conclusion

We have investigated P systems with antiport rules for evolution rules, where only evolution rules but no objects are moved across the membranes during a computation. We have shown that for such systems we obtain universal computational power when using purely catalytic evolution rules even with only one catalyst in one membrane. When using such systems without catalysts, a characterization of the family of languages generated

by ET0L systems is obtained, and again one membrane is already sufficient. Hence, the results elaborated in this paper are already optimal with respect to the number of membranes. The influence of the weight of the antiport rules remains as a challenging open problem for future research.

Acknowledgements. The paper was initiated during the Brainstorming Week on Membrane Computing taking place in Sevilla during the first week of February, 2004. The authors appreciate interesting discussions with Matteo Cavaliere un on some topics elaborated in this paper.

References

[1] M. Cavaliere: Evolution-Communication P Systems. In: Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, Eds., *Membrane Computing. International Workshop, WMC-CdeA 2002, Curtea de Argeş, Romania, August 2002*, Springer-Verlag, LNCS 2597, Berlin, 2003, 134–145.

[2] M. Cavaliere, D. Genova: P Systems with Symport/Antiport of Rules, in the present volume.

[3] J. Dassow, Gh. Păun: *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.

[4] R. Freund, L. Kari, M. Oswald, P. Sosík: Computationally Universal P Systems without Priorities: Two Catalysts Are Sufficient. *Theoretical Computer Science*, to appear.

[5] R. Freund, C. Martín-Vide, Gh. Păun: From Regulated Rewriting to Computing with Membranes: Collapsing Hierarchies. *Theoretical Computer Science*, **312** (2004), 143–188.

[6] R. Freund, M. Oswald: P Systems with Conditional Communication Rules Assigned to Membranes. *JALC*, to appear.

[7] O.H. Ibarra: The Number of Membranes Matters. In A. Alhazov, C. Martín-Vide, Gh. Păun, Eds.: *Preproceedings of Workshop on Membrane Computing, WMC-2003*, Tarragona, July 17-22, 2003, Rovira i Virgili Univ., Tech. Rep. No. 28, Tarragona, 2003, 273–285.

[8] Gh. Păun: Computing with Membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143, and TUCS Research Report 208 (1998) (`http://www.tucs.fi`)

[9] Gh. Păun: *Membrane Computing: An Introduction*. Springer-Verlag, Berlin, 2002.

[10] Rozenberg, G., Salomaa, A., Eds.: *Handbook of Formal Languages.* Springer-Verlag, Berlin, Heidelberg, 1997.

[11] The P Systems Web Page: `http://psystems.disco.unimib.it`