Journal of Universal Computer Science, vol. 8, no. 2 (2002), 317-331
submitted: 15/9/09, accepted: 30/1/02, appeared: 28/2/02

On the Power of P Systems with Symport Rules[1]

Carlos Martín-Vide
(Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
E-mail: cmv@astor.urv.es)

Andrei Păun
(Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7
E-mail: apaun@csd.uwo.ca)

Gheorghe Păun
(Institute of Mathematics of the Romanian Academy
PO Box 1-764, 70700 Bucureşti, Romania, and
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
E-mail: gpaun@imar.ro, gp@astor.urv.es)

Abstract: A purely communicative variant of P systems was considered recently, based on the trans-membrane transport of couples of chemicals. When using both symport rules (the chemicals pass together in the same direction) and antiport rules (one chemical enters and the other exits a membrane), one obtains the computational completeness, and the question was formulated what happens when only symport rules are considered. We address here this question. First, we surprisingly find that "generalized" symport rules are sufficient: if more than two chemicals pass together through membranes, then we get again the power of Turing machines. Three results of this type are obtained, with a trade-off between the number of chemicals which move together (at least three in the best case) and the number of membranes used. The same result is obtained for standard symport rules (couples of chemicals), if the passing through membranes is conditioned by some permitting contexts (certain chemicals should be present in the membrane). In this case, four membranes suffice. The study of other variants of P systems with symport rules (for instance, with forbidding contexts) is formulated as an open problem.
Key Words: Molecular Computing, Membrane Computing, Symport, Antiport, Computational Universality
Category: F.1.1

1 Introduction

This paper is a direct continuation of the paper [Păun and Păun, 2002], where P systems with symport and antiport rules were introduced.

P systems are distributed parallel computing models which abstract from the structure and the functioning of the living cells. In short, we have a *membrane structure*, consisting of several membranes embedded in a main membrane

[1] C. S. Calude, K. Salomaa, S. Yu (eds.). *Advances and Trends in Automata and Formal Languages. A Collection of Papers in Honour of the 60th Birthday of Helmut Jürgensen.*

(called the *skin*) and delimiting *regions* (Figure 1 illustrates these notions) where multisets of certain *objects* are placed; the objects evolve according to given *evolution rules*, which are applied nondeterministically (the rules to be used and the objects to evolve are randomly chosen) in a maximally parallel manner (in each step, all objects which can evolve must do it). The objects can also be communicated from a region to another one. In this way, we get *transitions* from a *configuration* of the system to the next one. A sequence of transitions constitutes a *computation*; with each *halting computation* we associate a *result*, the number of objects from a specified *output membrane*.

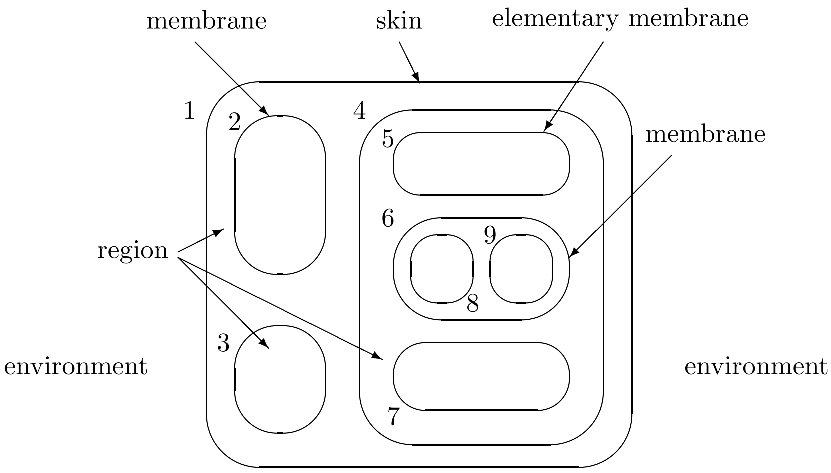

Figure 1: A membrane structure

Since these computing devices have been introduced ([Păun, 2000]) several different classes of P systems have been considered. Many of them were proved to be computationally complete, able to compute all Turing computable sets of natural numbers. When membrane division, or membrane creation (or string-object replication) is allowed, NP-complete problems are shown to be solved in polynomial (often linear) time. Comprehensive details can be found at the web address `http://bioinformatics.bio.disco.unimib.it/psystems`.

Starting from the observation that communication is rather important in this framework, a purely communicative variant of P systems was introduced in [Păun and Păun, 2002] (a previous proposal, of a completely different type, was discussed in [Martin-Vide et al., 2002]), modeling a real life phenomenon, that of trans-membrane transport in pairs of chemicals. When two chemicals can pass together through a membrane in the same direction, the process is called *symport*. When the two chemicals pass only with the help of each other, but in opposite directions, one says that we have *antiport*. We refer to [Alberts et al., 1998] and [Ardelean, 2002] for biological details.

Technically, the rules used in P systems as models of these biological processes are of the forms (x, in) and (x, out) for symport, and $(x, out; y, in)$ for antiport, where x, y are strings of symbols representing multisets of chemicals.

Of course, this is a generalization of what happens in nature, where mainly pairs of chemicals are coupled. Still, even using only such standard rules, it is found in [Păun and Păun, 2002] that we can compute in this "osmotic" manner whatever a Turing machine can compute. The proof uses both symport and antiport rules.

However, the symport look more attractive than the antiport (not only because of the name...): one can imagine that in the case of antiport a sort of "traffic jam" appears in a protein channel through which the chemicals have to pass simultaneously in opposite directions (of course, this discussion has nothing to do with biochemistry, it is just a way to introduce a mathematical restriction). So, what about using only symport rules?

At the first sight, using only symport rules seems not to be too powerful, but we find that if we are allowed to use packages of more than two chemicals, then we characterize again the family of Turing computable sets of numbers. In the results proved below, the decrease in the number of chemicals in a symport rule is obtained at the expense of increasing the number of membranes. The universality is obtained also with standard symport rules, that is, involving at most pairs of molecules which pass together through a membrane conditionally, depending on the content of that membrane. We consider here only the case of promoters, single chemicals which should be present when a symport rule is applied. The case of inhibitors (chemicals which should not be present when a symport rule is applied) remains to be investigated.

2 P Systems with Symport/Antiport

The language theory notions we use here are standard, and can be found, for instance, in [Rozenberg and Salomaa, 1997].

A membrane structure is pictorially represented by a Venn diagram (like the one in Figure 1); it can be mathematically represented by a tree or by a string of matching parentheses associated in a standard manner with a tree. A multiset over a set X is a mapping $M : X \longrightarrow \mathbf{N} \cup \{\infty\}$ (we allow infinite multiplicities). Here we always use multisets over finite sets X (that is, X will be an alphabet). A multiset with a finite support and with finite multiplicities of elements can be represented by a string over X; the number of occurrences of a symbol $a \in X$ in a string $x \in X^*$ represents the multiplicity of a in the multiset represented by x. Clearly, all permutations of a string represent the same multiset, and the empty multiset is represented by the empty string, λ.

We introduce first the P systems with rules of arbitrary length. Such a device is a construct

$$\Pi = (V, \mu, w_1, \ldots, w_m, M_e, R_1, \ldots, R_m, i_o),$$

where:

1. V is the alphabet of chemicals (we call them *objects*);
2. μ is a membrane structure with m membranes (injectively labeled by positive integers $1, 2, \ldots, m$; the skin membrane is labeled by 1); $m \geq 1$ is called the *degree* of the system;
3. $w_1, \ldots, w_m$ are strings over V representing the multisets of objects initially present in the regions of the system, and M_e is the multiset of objects present outside the system, in the *environment*; the "internal multisets" have finite

multiplicities of objects, while for each $a \in V$ we have either $M_e(a) = 0$ or $M_e(a) = \infty$ (outside the system, an object is either absent, or present in arbitrarily many copies); that is why M_e is identified by its support (the set of objects appearing at least once – hence in infinitely many copies – in the environment);

4. $R_1, \ldots, R_m$ are finite sets of *rules* of the form $(x, out; y, in)$, for $x, y \in V^*$ with $xy \neq \lambda$; if one of x, y is empty, then we have a symport rule (written in the form $(x, out), (y, in)$), when both x and y are non-empty we have an antiport rule;
5. $i_o \in \{1, \ldots, m\}$ is an elementary membrane of μ (the output membrane).

The meaning of a rule $(x, out; y, in)$ from $R_i, 1 \leq i \leq m$, is the following: the objects present in x exit the region of membrane i, and, simultaneously, the objects present in y enter the region of membrane i. Of course, x and y represent multisets, hence the multiplicities of objects matters.

Note that the rules do not change the nature of objects, they just move objects from a region to another one, possibly sending objects out of the system or bringing objects in the system, from the environment. Thus, a P system with symport/antiport rules observes the conservation law (which is not necessarily the case with other classes of P systems).

The multisets of objects present in the m regions of Π constitute the *configuration* of the system; $(w_1, \ldots, w_m)$ is the initial configuration. We pass from a configuration to another configuration by using the rules from $R_1, \ldots, R_m$, as customary in P systems: the rules are applied in the non-deterministic maximally parallel manner, in the sense that we apply the rules in parallel, to all objects which can be processed, non-deterministically choosing the rules and the objects. Thus, a transition means a redistribution of objects among regions (and environment), which is maximal for the chosen set of rules. A sequence of transitions between configurations of the system constitutes a *computation*; a computation is successful if it *halts*, i.e., it reaches a configuration where no rule can be applied to any of the objects.

The *result* of a successful computation is the number of objects present within the membrane with the label i_o in the halting configuration. A computation which never halts yields no result. The set of all the numbers computed by Π is denoted by $N(\Pi)$.

The family of all sets $N(\Pi)$, computed as above by systems Π of degree at most $m \geq 1$, using symport rules $(x, in), (x, out)$ with $|x| \leq p$ (we say that $|x|$ is the *weight* of the symport rule $(x, in), (x, out)$), and antiport rules $(x, out; y, in)$ with $|x|, |y| \leq q$, is denoted by $NPP_m(sym_p, anti_q)$, for $m \geq 1$ and $p, q \geq 0$. (The case from biochemistry corresponds to $p \leq 2$ and $q = 1$.) When the number of membranes is not bounded we replace the subscript m by $*$.

Also, we use NRE to denote the family of recursively enumerable sets of natural numbers; this is precisely the family of the length sets of recursively enumerable languages, and we will make below an essential use of this observation.

The following results are proved in [Păun and Păun, 2002]:

1. $NPP_5(sym_2, anti_1) = NRE$,
2. $NPP_2(sym_2, anti_2) = NRE$.

The P systems with symport and antiport rules were also considered in [Păun et al., 2001], where, among others, a complete mathematical formalization is given, both of the structure and the functioning of such a machinery.

3 The Symport Suffices

At the end of [Păun and Păun, 2002] it is proposed to investigate the power of P systems using symport rules only, and it is suggested that in order to obtain a significantly powerful class of computing devices it is necessary to supplement the model with a control of using the rules. We will consider this suggestion in the subsequent sections, but first we show that, surprisingly enough, generalized (with respect to the biochemical case) symport rules are already universal.

In the proof of this result, as well as in the subsequent sections, we will use the notion of a *matrix grammar with appearance checking*.

Such a grammar is a construct $G = (N, T, S, M, F)$, where N, T are disjoint alphabets, $S \in N$, M is a finite set of sequences of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \geq 1$, of context-free rules over $N \cup T$ (with $A_i \in N, x_i \in (N \cup T)^*$), and F is a set of occurrences of rules in M (N is the nonterminal alphabet, T is the terminal alphabet, S is the axiom, while the elements of M are called matrices).

For $w, z \in (N \cup T)^*$ we write $w \Longrightarrow z$ if there is a matrix $(A_1 \to x_1, \ldots, A_n \to x_n)$ in M and the strings $w_i \in (N \cup T)^*, 1 \leq i \leq n+1$, such that $w = w_1, z = w_{n+1}$, and, for all $1 \leq i \leq n$, either $w_i = w_i' A_i w_i'', w_{i+1} = w_i' x_i w_i''$, for some $w_i', w_i'' \in (N \cup T)^*$, or $w_i = w_{i+1}$, A_i does not appear in w_i, and the rule $A_i \to x_i$ appears in F. (The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied – therefore we say that these rules are applied in the *appearance checking* mode.)

The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$. The family of languages of this form is denoted by MAT_{ac}.

It is known that matrix grammars with appearance checking generate the family RE of recursively enumerable languages.

A matrix grammar $G = (N, T, S, M, F)$ is said to be in the *binary normal form* if $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets mutually disjoint, and the matrices in M are in one of the following forms:

1. $(S \to XA)$, with $X \in N_1, A \in N_2$,
2. $(X \to Y, A \to x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2$,
3. $(X \to Y, A \to \#)$, with $X, Y \in N_1, A \in N_2$,
4. $(X \to \lambda, A \to x)$, with $X \in N_1, A \in N_2$, and $x \in T^*, |x| \leq 2$.

Moreover, there is only one matrix of type 1 and F consists exactly of all rules $A \to \#$ appearing in matrices of type 3; $\#$ is a trap-symbol, because once introduced, it is never removed. A matrix of type 4 is used only once, in the last step of a derivation.

According to Lemma 1.3.7 in [Dassow and Păun, 1989], for each matrix grammar there is an equivalent matrix grammar in the binary normal form.

We are now ready to give our first result: as announced above, the (generalized) symport rules suffice.

Theorem 1. *$NPP_2(sym_5, anti_0) = NRE$.*

Proof. We only prove the inclusion $NRE \subseteq NPP_2(sym_5, anti_0)$, the opposite one being straightforward.

Because $RE = MAT_{ac}$, it follows that each set from NRE is the length set of a language from MAT_{ac}; moreover, we can consider a language in MAT_{ac} over the one-letter alphabet. Thus, let us start from a matrix grammar with appearance checking $G = (N, \{a\}, S, M, F)$ in the binary normal form, hence with $N = N_1 \cup N_2 \cup \{S, \#\}$, and with the matrices of the four types mentioned above. Assume that we have n matrices in total, k of types 2 and 4, hence of the form $m_i : (X \to \alpha_1, A \to \alpha_2\alpha_3)$, for some $X \in N_1$, $\alpha_1 \in N_1 \cup \{\lambda\}$, $A \in N_2$, and $\alpha_2, \alpha_3 \in N_2 \cup \{a, \lambda\}$, and the others of type 3, hence of the form $m_i : (X \to Y, A \to \#)$, for some $X, Y \in N_1$, and $A \in N_2$. Let us denote by $(S \to X_0A_0)$ the initial matrix of G.

We construct the following P system (of degree 2, using only symport rules)

$$
\begin{aligned}
\Pi &= (V, [_1[_2\ ]_2]_1, w_1, \lambda, M_e, R_1, R_2, 2),\\
V &= N_1 \cup N_2 \cup \{d_i \mid 1 \le i \le n\} \cup \{g_i \mid k+1 \le i \le n\}\\
&\quad \cup \{a, b, b', c, e, f, g, h\},\\
w_1 &= X_0A_0bced_1d_2\dots d_n,\\
M_e &= N_1 \cup N_2 \cup \{a, b', f, g, h\} \cup \{g_i \mid k+1 \le i \le n\},\\
R_1 &= \{(cd_iXA, out),\ (cd_i\alpha_1\alpha_2\alpha_3, in) \mid\\
&\qquad \text{for } m_i : (X \to \alpha_1, A \to \alpha_2\alpha_3), 1 \le i \le k,\\
&\qquad \text{with } X \in N_1, \alpha_1 \in N_1, A \in N_2, \alpha_2, \alpha_3 \in N_2 \cup \{a, \lambda\}\}\\
&\quad \cup \{(bc, out),\ (bb', in)\}\\
&\quad \cup \{(cd_iXA, out),\ (d_i\alpha_2\alpha_3f, in) \mid \text{for } m_i : (X \to \lambda, A \to \alpha_2\alpha_3),\\
&\qquad \text{with } X \in N_1, A \in N_2, \alpha_2, \alpha_3 \in \{a, \lambda\}\}\\
&\quad \cup \{(cd_iX, out),\ (d_iYg_ih, in),\ (d_ig_iAb, out) \mid\\
&\qquad \text{for } m_i : (X \to Y, A \to \#), k+1 \le i \le n, \text{ with } X, Y \in N_1, A \in N_2\}\\
&\quad \cup \{(gg_i, out) \mid k+1 \le i \le n\}\\
&\quad \cup \{(eh, out),\ (ceg, in)\},\\
R_2 &= \{(a, in),\ (b', in),\ (b', out)\}\\
&\quad \cup \{(fA, in),\ (fA, out) \mid A \in N_2\}.
\end{aligned}
$$

Assume that in (the region of) membrane 1 we have a multiset corresponding to a sentential form Xw of G (initially, we have X_0A_0, for the matrix $(S \to X_0A_0)$ of G), as well as copies of the symbols b, c, e and $d_1, \dots, d_n$. If c does not exit together with a symbol $d_i, 1 \le i \le n$, by a rule $(cd_iXA, out) \in R_1$ or $(cd_iX, out) \in R_1$, then it will be used by the rule $(bc, out) \in R_1$ and will introduce into the system a copy of the trap-object b', which will pass forever back and forth through membrane 2, preventing the halting of the computation. Thus, as long as c is present in the system, we have to continue the computation.

Assume that we use a rule $(cd_iXA, out) \in R_1$, for some $1 \le i \le k$. If the first rule of the matrix $m_i : (X \to \alpha_1, A \to \alpha_2\alpha_3)$ has $\alpha_1 \in N_1$, then we use the rule $(cd_i\alpha_1\alpha_2\alpha_3, in) \in R_1$, and the matrix is correctly simulated. Because we return to a configuration with the symbols c, d_i again present in membrane 1, we can continue.

If the matrix was a terminal one, to be used at the end of a derivation in G, then we use the rule $(d_i\alpha_1\alpha_2 f, in) \in R_1$, which simulates the matrix, but does not bring back the symbol c, hence no other matrix can be simulated. Instead, we introduce into the system a copy of the symbol f, which checks whether or not any symbol A from N_2 is still present in the system. In the affirmative case, the computation will never end, because of the rules $(fA, in), (fA, out)$, from R_2. Thus, in the moment when the symbol f is introduced into the system the derivation in G should be terminal.

Assume now that we start by using a rule $(cd_iX, out) \in R_1$ for some $k + 1 \leq i \leq n$, hence associated with a matrix $m_i : (X \to Y, A \to \#)$ with the second rule to be used in the appearance checking mode. The symbol d_i returns to the system by the rule $(d_iYg_ih, in) \in R_1$, which simulates the use of the first rule of the matrix, and, at the same time, the auxiliary symbols g_i, h are introduced in the system. Because c is not present, we cannot start the simulation of another matrix. In the presence of g_i, if any copy of A is present, then we can send it out, together with the symbol b, which brings back the trap-symbol b', hence the computation never stops (note that there is only one symbol of the form g_j present in the system, with $j = i$ for i identifying the matrix m_i which is simulated). If no copy of A is present in membrane 1, then the rule $(d_ig_iAb, out) \in R_1$ cannot be used, and g_i waits in the system. In parallel, the rule $(eh, out) \in R_1$ is used, followed by $(ceg, in) \in R_1$, which brings back the symbols c, e, as well as one copy of g. The multiset contains again the auxiliary symbols b, c, e and $d_1, \dots, d_n$, hence the simulation of matrices from M can be iterated. (The symbols g and g_i present in the system exit immediately, by the rule $(gg_i, out) \in R_1$.)

Consequently, a computation in Π stops if and only if it simulates a terminal derivation in G. All copies of the terminal symbol a are introduced in the output membrane, hence $N(\Pi) = \{m \mid a^m \in L(G)\}$, which completes the proof.

We do not know whether or not the previous result can be improved in what concerns the weight of the symport rules (whether $NRE = NPP_2(sym_p, anti_0)$ holds or doesn't hold for some $p \leq 4$), but we can achieve this goal by paying in the number of membranes used.

Theorem 2. $NPP_3(sym_4, anti_0) = NRE$.

Proof. We start again from a matrix grammar $G = (N, \{a\}, S, M, F)$ in the binary normal form. Using the same notations as in the proof of the previous theorem, we construct the P system (of degree 3)

$$\begin{aligned}
\Pi &= (V, [_1[_2\ ]_2[_3\ ]_3]_1, w_1, \lambda, w_3, M_e, R_1, R_2, R_3, 2),\\
V &= N_1 \cup N_2 \cup \{d_i \mid 1 \leq i \leq n\} \cup \{g_i \mid k+1 \leq i \leq n\}\\
&\quad \cup \{a, b, b', c, c', c'', e, f, g, h, q\},\\
w_1 &= A_0bed_1d_2\dots d_n,\\
w_3 &= X_0qcc'c'',\\
M_e &= N_1 \cup N_2 \cup \{a, b', f, g, h\} \cup \{g_i \mid k+1 \leq i \leq n\},\\
R_1 &= \{(cd_iXA, out),\ (d_iY\alpha_1\alpha_2, in) \mid\\
&\qquad \text{for } m_i : (X \to Y, A \to \alpha_1\alpha_2), 1 \leq i \leq k,
\end{aligned}$$

$$\begin{aligned}
&\quad \text{with } X, Y \in N_1, A \in N_2, \alpha_1, \alpha_2 \in N_2 \cup \{a, \lambda\}\} \\
&\cup \{(c, in),\ (bc, out),\ (bb', in)\} \\
&\cup \{(c''d_iXA, out),\ (d_i\alpha_1\alpha_2 f, in) \mid \text{for } m_i : (X \to \lambda, A \to \alpha_1\alpha_2), \\
&\quad 1 \le i \le k,\ \text{with } X \in N_1, A \in N_2, \alpha_1, \alpha_2 \in \{a, \lambda\}\} \\
&\cup \{(c'd_iX, out),\ (d_iYg_ih, in),\ (d_ig_iAb, out) \mid \\
&\quad \text{for } m_i : (X \to Y, A \to \#), k+1 \le i \le n,\ \text{with } X, Y \in N_1, A \in N_2\} \\
&\cup \{(gg_i, out) \mid k+1 \le i \le n\} \\
&\cup \{(eh, out),\ (c'eg, in),\ (q, out)\}, \\
R_2 &= \{(a, in),\ (b', in),\ (b', out)\} \\
&\cup \{(fA, in),\ (fA, out) \mid A \in N_2\}, \\
R_3 &= \{(qc, out),\ (qc, in),\ (qc', out),\ (qc', in)\ (qc'', out)\} \\
&\cup \{(X, in),\ (X, out) \mid X \in N_1\}.
\end{aligned}$$

The difference from the system constructed in the previous proof is that the simulation of nonterminal matrices $m_i, 1 \le i \le k$, is done in the presence of the symbol c, the simulation of terminal matrices is done in the presence of the symbol c'', while the simulation of matrices $m_i, k+1 \le i \le n$, is done in the presence of the symbol c'. These symbols are released from membrane 3 (at the same time with the unique symbol from N_1) by means of the symbol q, which nondeterministically chooses one of them. When c is in the skin membrane, several matrices without rules to be used in the appearance checking manner can be simulated, and then the symbol c returns to membrane 3. As long as any symbol $X \in N_2$ is present in the system, the computation should continue, as we can use the rules $(X, in), (X, out)$ from R_3. When a terminal matrix is simulated, no symbol from N_1 is reintroduced and, also, c'' is not reintroduced into the system, hence the computation can stop (if the derivation in G was terminal) by sending out the symbol q. After simulating a matrix having a rule used in the appearance checking mode, the symbol c' has to return to membrane 3 (we can simulate two matrices $m_i, k+1 \le i \le n$, in a row only if they have the first rule of the form $X \to X$). We conclude that $N(\Pi) = \{m \mid a^m \in L(G)\}$, which ends the proof.

The weight of symport rules can be further decreased, but this time we needed two more membranes (and a different construction):

Theorem 3. $NPP_5(sym_3, anti_0) = NRE$.

Proof. Let $G = (N, \{a\}, S, M, F)$ be a matrix grammar with appearance checking in the binary normal form. Assume that M contains k_1 matrices of the form $m_i : (X \to Y,\ A \to \alpha_1\alpha_2), 1 \le i \le k_1$, with $X, Y \in N_1, A \in N_2, \alpha_1, \alpha_2 \in N_2 \cup \{a, \lambda\}$, k_2 matrices of the form $m_i : (X \to \lambda, A \to \alpha_1\alpha_2), k_1+1 \le i \le k_1+k_2$, with $X \in N_1, A \in N_2, \alpha_1, \alpha_2 \in \{a, \lambda\}$ (terminal matrices), and k_3 matrices of the form $m_i : (X \to Y, A \to \#), k_1+k_2+1 \le i \le n$, with $X, Y \in N_1, A \in N_2$.

We construct the P system (of degree 5)

$$\begin{aligned}
\Pi &= (V, \mu, w_1, \ldots, w_5, M_e, R_1, \ldots, R_5, 5), \\
V &= N_1 \cup N_2 \cup \{c_i, c_i', c_i'', d_i, d_i', d_i'' \mid 1 \le i \le n\}
\end{aligned}$$

$$\begin{aligned}
&\cup \{a, b, b', f, e\},\\
\mu &= [_1[_2[_3[_4\ ]_4]_3]_2[_5\ ]_5]_1,\\
w_1 &= XAb, \text{ for } (S \to XA) \text{ being the initial matrix of } G,\\
w_2 &= c''_1 \dots c''_n,\\
w_3 &= c'_1 \dots c'_n,\\
w_4 &= c_1 \dots c_n f,\\
w_5 &= \lambda,\\
M_e &= N_1 \cup N_2 \cup \{a, b', e\} \cup \{d_i, d'_i, d''_i \mid 1 \le i \le n\},\\
R_1 &= \{(c_iX, out),\ (c'_iA, out),\ (c''_i, out),\\
&\quad (c_id_iY, in),\ (c_id'_i\alpha_1, in),\ (c''_id''_i\alpha_2, in),\\
&\quad (c_ib, out),\ (c'_ib, out),\ (d_ib, out),\ (d'_ib, out),\ (d''_ib, out) \mid\\
&\quad \text{for all } m_i : (X \to Y, A \to \alpha_1\alpha_2), 1 \le i \le k_1,\\
&\quad \text{with } X, Y \in N_1, A \in N_2, \alpha_1, \alpha_2 \in N_2 \cup \{a, \lambda\}\}\\
&\cup \{(bb', in)\}\\
&\cup \{(c_iX, out),\ (c'_iA, out),\ (c''_i, out),\\
&\quad (c_id_ie, in),\ (c_id'_i\alpha_1, in),\ (c''_id''_i\alpha_2, in),\\
&\quad (c_ib, out),\ (c'_ib, out),\ (d_ib, out),\ (d'_ib, out),\ (d''_ib, out) \mid\\
&\quad \text{for all } m_i : (X \to \lambda, A \to \alpha_1\alpha_2), k_1 + 1 \le i \le k_1 + k_2,\\
&\quad \text{with } X \in N_1, A \in N_2, \alpha_1, \alpha_2 \in \{a, \lambda\}\}\\
&\cup \{(c_iX, out),\ (c_ib, out),\ (c'_iA, out),\ (c''_i, out),\\
&\quad (c_id_iY, in),\ (c'_ib', in),\ (c''_id''_id'_i, in) \mid\\
&\quad \text{for all } m_i : (X \to Y, A \to \#), k_1 + k_2 + 1 \le i \le n,\\
&\quad \text{with } X, Y \in N_1, A \in N_2\},\\
R_2 &= \{(c'_i, out),\ (c_ic''_i, out),\ (c_id_i, in),\ (c'_id'_i, in),\ (c''_id''_i, in) \mid\\
&\quad \text{for all } 1 \le i \le n\}\\
&\cup \{(b', in),\ (b', out)\},\\
R_3 &= \{(c_ic'_i, out),\ (c_id_i, in),\ (c'_id'_i, in) \mid \text{ for all } 1 \le i \le n\},\\
R_4 &= \{(fc_i, out),\ (fc_id_i, in) \mid \text{ for all } i \in \{1, 2, \dots, k_1\} \cup \{k_1 + k_2 + 1, \dots, n\}\}\\
&\cup \{(c_id_i, in) \mid \text{ for all } k_1 + 1 \le i \le k_1 + k_2\},\\
R_5 &= \{(a, in)\}\\
&\cup \{(eA, in),\ (eA, out) \mid A \in N_2\}.
\end{aligned}$$

This system works as follows. Nondeterministically, one c_i exits membrane 4, together with f; this latter symbol will wait in membrane 3 until the matrix m_i was simulated. Then, the symbol c_i exits membrane 3 together with c'_i, and after that both c_i and c'_i exit membrane 2, also bringing out of this membrane the symbol c''_i.

Assume that $1 \le i \le k_1$, corresponding to a matrix $m_i : (X \to Y, A \to \alpha_1\alpha_2)$, $1 \le i \le k_1$, with $X, Y \in N_1, A \in N_2, \alpha_1, \alpha_2 \in N_2 \cup \{a, \lambda\}$. By using the rules (c_iX, out), (c'_iA, out), (c''_i, out), (c_id_iY, in), $(c_id'_i\alpha_1, in)$, $(c''_id''_i\alpha_2, in)$ from R_1 we simulate this matrix. The simulation is correct, as none of c_i, c'_i may remain unused: if any of the rules $(c_ib, out), (c'_ib, out)$ is used, then b exits the system and then it brings into the system the trap-symbol b': by using the rules $(b', in), (b', out)$ from R_2 we can continue the computation forever.

We cannot continue by simulating again this matrix, because the symbols d_i, d'_i, d''_i should not use any of the rules $(d_i b, out), (d'_i b, out), (d''_i b, out) \in R_1$. Thus, all pairs $c_i d_i, c'_i d'_i, c''_i d''_i$ will enter membrane 2, then $c_i d_i, c'_i d'_i$ will enter membrane 3, and after that $c_i d_i$ enter membrane 4, at the same time with f. In this way, we have again f in membrane 4, and the simulation of matrices from M can be continued. (The symbols d_i, d'_i, d''_i will remain in the corresponding membranes, unused.) However, when the symbols d_i, d'_i, d''_i are not in membrane 1, hence the rules $(d_i b, out), (d'_i b, out), (d''_i b, out) \in R_1$ cannot be used, the symbols c_i, c'_i, c''_i can exit again membranes 4, 3, 2, and the simulation of the same matrix m_i is entailed, which is correct with respect to G.

If we have started with $k_1 + k_2 + 1 \leq i \leq n$, corresponding to a matrix $m_i : (X \rightarrow Y,\, A \rightarrow \#)$, with $X, Y \in N_1, A \in N_2$, then, after bringing c_i, c'_i, c''_i in the skin membrane, we continue as follows. The rule $(c_i X, out) \in R_1$ must be used, otherwise we use $(c_i b, out) \in R_1$ and the computation will never stop. At the same time c''_i exits. If any copy of A is present, then the rule $(c'_i A, out) \in R_1$ must be used, and at the next step $(c'_i b', in) \in R_1$ brings the trap-symbol in the system. If no copy of A is present, then c'_i waits in the skin membrane. At the next step, c_i brings d_i and Y into the system, while c''_i brings d'_i and d''_i. In this way, we have again the pairs $c_i d_i, c'_i d'_i, c''_i d''_i$, which will bring the symbols c_i, c'_i, c''_i in the starting membranes. The matrix m_i was correctly simulated. (If $X = Y$, then the simulation can be done once again, after removing the symbols d_i, d'_i, d''_i from the skin membrane, but this changes nothing.)

The process can be iterated. When we simulate a terminal matrix $m_i, k_1+1 \leq i \leq k_1 + k_2$, we introduce the symbol e which will check whether or not any nonterminal from N_2 is still present (in the positive case the computation will continue forever), while the symbols $c_i, d_i, c'_i, d'_i, c''_i, d''_i$ enter membranes 2 and 3 as above, but only c_i, d_i enter membrane 4, not also the symbol f. In this way, the computation stops, because f is no longer available in membrane 4.

Thus, $N(\Pi) = \{m \mid a^m \in L(G)\}$ and the proof is complete.

4 What About Antiport Rules?

At the first sight, the antiport rules are a generalization of symport rules, because each rule (u, out) can be transformed into $(u, out; d, in)$, where d is a dummy object, and the same for rules (u, in). Actually, this is not true, no system Π using only antiport rules can compute the number 0, because if $0 \in N(\Pi)$, then $N(\Pi) = \{0\}$: if the output membrane is initially empty, then it will remain forever empty, if it is not initially empty, then it will never become empty.

However, if we consider two sets of numbers equal if they differ at most in the element 0, then all recursively enumerable sets of numbers can be computed by P systems using only antiport rules. The proof of this assertion is not trivial, because we have to start with finite multisets of objects present in the system and we have to ensure that arbitrarily large multisets can be produced. That is why we give a proof of the next result, although not dealing directly with symport rules, the topic of the present paper.

Let us denote $NRE' = \{M \in \mathbf{N} - \{0\} \mid M \in NRE\}$.

Theorem 4. $NPP_3(sym_0, anti_2) = NRE'$.

Proof. The proof is based on the construction from the proof of Theorem 2 from [Păun and Păun, 2002], with an additional care paid to the fact that each symport rule should be replaced by an antiport rule.

Let $G = (N, \{a\}, S, M, F)$ be a matrix grammar with appearance checking in the binary normal form, with $N = N_1 \cup N_2 \cup \{S, \#\}$, and with M containing matrices $m_i : (X \to \alpha, A \to x)$, $X \in N_1, \alpha \in N_1 \cup \{\lambda\}, A \in N_2, x \in (N_2 \cup \{a\})^*$, for $i = 1, \ldots, k$, and $m_i : (X \to Y, A \to \#), X, Y \in N_1, A \in N_2$, for $i = k+1, \ldots, n$, for some $k \geq 1, n \geq k$.

We construct the P system with antiport rules

$$
\begin{aligned}
\Pi &= (V, [_1 [_2\ ]_2 [_3\ ]_3]_1, w_1, h', g, M_e, R_1, R_2, R_3, 2),\\
V &= N_1 \cup N_2 \cup \{a, c, f, g, h, h', h'', F, H, Z\}\\
&\quad \cup \{c_i, c_i', d_i \mid 1 \leq i \leq n\},\\
w_1 &= XAch, \text{ for } (S \to XA) \text{ being the initial matrix of } G,\\
M_e &= N_1 \cup N_2 \cup \{a, f, h, h', h'', F, H, Z\} \cup \{c_i, c_i', d_i \mid 1 \leq i \leq n\},\\
R_1 &= \{(cX, out; c_iY, in),\ (c_iA, out; cc_i', in),\ (c_i', out; u, in) \mid\\
&\qquad m_i : (X \to Y, A \to u), 1 \leq i \leq k\}\\
&\quad \cup \{(cX, out; c_if, in),\ (c_iA, out; c_i', in),\ (c_i', out; u, in) \mid\\
&\qquad m_i : (X \to \lambda, A \to u), 1 \leq i \leq k\}\\
&\quad \cup \{(cX, out; c_id_i, in),\ (d_iA, out; YH, in),\ (c_iA, out; Z, in),\\
&\qquad (H, out; cF, in),\ (c_iF, out; h'', in) \mid m_i : (X \to Y, A \to \#), k+1 \leq i \leq n\}\\
&\quad \cup \{(h, out; hh, in),\ (h, out; h', in)\},\\
R_2 &= \{(h', out; h'h', in),\ (h', out; a, in),\ (h', out; h', in)\},\\
R_3 &= \{(g, out; \alpha, in),\ (\alpha, out; g, in) \mid \alpha \in \{Z\} \cup \{c_i, c_i' \mid 1 \leq i \leq k\}\}\\
&\quad \cup \{(g, out; fD, in),\ (fD, out; g, in) \mid D \in N_2\}.
\end{aligned}
$$

For the sake of completeness, we discuss some details about the work of this system, although they are very similar to those from the proof of Theorem 2 from [Păun and Păun, 2002].

The symbols c, c_i, c_i' control the simulation of matrices $m_i, 1 \leq i \leq k$, in the following way. After using the rule $(cX, out; c_iY, in) \in R_1$, we have c_i in the system. If the rule $(c_iA, out; c_i'c, in) \in R_1$ cannot be used, then the symbol c_i will go forever back and forth through membrane 3, hence the computation will never finish. If the rule $(c_iA, out; c_i'c, in) \in R_1$ is used, then at the next step c_i' will exit, bringing into the system the string u, which completes the simulation of the matrix $m_i : (X \to Y, A \to u)$.

In the case of terminal matrices, that is, with the first rule of the form $X \to \lambda$, we bring the symbols c_i, f in the system, c_i simulates the second rule of the matrix, and f checks whether or not the derivation was a terminal one. In the negative case, the rules $(g, out; fD, in), (fD, out; g, in)$ of R_3, for $D \in N_2$, are used forever (the symbol c is no longer present, hence no other rule can be used).

If we start with a rule of the form $(cX, out; c_id_i, in) \in R_1$, for some $k+1 \leq i \leq n$, for $m_i : (X \to Y, A \to \#)$, then at the next step we have to use the rule $(d_i, out; YH, in) \in R_1$. If in the skin membrane there is any copy of A, then the rule $(c_iA, out; Z, in) \in R_1$ has to be used, and the computation will never finish, because of the rules $(g, out; Z, in), (Z, out; g, in)$ from R_3. If no copy of A is present, then c_i waits in the skin membrane. At the next step, H exits and brings cF inside. Now, together with F, the symbol c_i can leave the system. In this way, the application of matrix m_i was simulated.

Note that in all cases the symbol c is again available, hence the process can be iterated.

Let us now see how arbitrarily many copies of a can be introduced in membrane 2. Initially, we have one copy of h in membrane 1 and one copy of h' in membrane 2. By using the rules $(h, out; hh, in), (h, out; h', in) \in R_1$ any number of copies of h' can be introduced in membrane 1. From here, any number of copies of h' can be introduced into membrane 2. If all these symbols h' are exchanged for copies of a, then no rule $(h', out; x, in)$ can be used for membrane 2. Thus, if we introduce exactly m copies of h' when we want to generate the number m, then the computation will correctly stop, hence we have $N(\Pi) = \{m \mid a^m \in L(G)\}$, and this completes the proof.

5 Controls on the Use of Symport Rules

Let us return to the "realistic" symport process, that is, referring to at most two chemicals/objects. We do not know whether $NPP_*(sym_2, anti_0)$ is equal to NRE. However, we can supplement the power of symport rules by using them in a conditional manner, depending on the content of the membrane where they are used, and then such an equality holds true.

Specifically, a symport rule with a *promoter* is of the form $(x, in)_b$ or $(x, out)_b$, where x is a string (representing a multiset of objects) and b is an object. The meaning is that the multiset represented by x enters or exits, respectively, a membrane only if b is present in that membrane. We say that we have a *permitting context* use of the rule.

Dually, the object b can be used as an inhibitor, and the rule is *not* used in its presence. We say that the rules are used in the *forbidding context* mode, but we do not investigate this case here.

We denote by $NPP_m(psym_p, anti_0)$ the family of sets of numbers generated by P systems with symport rules $(x, in)_b, (x, out)_b$, with $|x| \leq p$, used in the permitting context mode, with at most m membranes. A rule of the form $(x, in), (x, out)$, hence without any symbol associated with it, is by default considered as having a permitting condition, and it is applied freely, as in a usual P system with symport rules.

The use of the symport rules in the conditional manner is rather useful, as we can decrease the weight of symport rules to two.

Theorem 5. $NPP_4(psym_2, anti_0) = NRE$.

Proof. Let us consider again a matrix grammar $G = (N, \{a\}, S, M, F)$ with appearance checking in the binary normal form, hence with $N = N_1 \cup N_2 \cup \{S, \#\}$, and with matrices of the four known types. We assume the matrices from M labeled in a one-to-one manner. Assume that we have n matrices in total, k of types 2 and 4, hence of the form $m_i : (X \to \alpha_1,\ A \to \alpha_1\alpha_2)$, for some $X \in N_1$, $\alpha_1 \in N_1 \cup \{\lambda\}$, $A \in N_2$, and $\alpha_1, \alpha_2 \in N_2 \cup \{a, \lambda\}$, and the others of type 3, hence of the form $m_i : (X \to Y, A \to \#)$, for some $X, Y \in N_1$, and $A \in N_2$.

We construct the P system (of degree 4) with symport rules

$$
\begin{aligned}
\Pi &= (V, \mu, w_1, w_2, w_3, \lambda, M_e, R_1, R_2, R_3, R_4, 4),\\
V &= N_1 \cup N_2 \cup \{c_i, d_i, d_i', d_i'', d_i''' \mid 1 \leq i \leq n\}\\
&\quad \cup \{d_i^{iv} \mid 1 \leq i \leq k\} \cup \{a, Z, Z', f, g, h, q, r\},\\
\mu &= [_1[_2[_3\ ]_3]_2[_4\ ]_4]_1,\\
w_1 &= XAZr, \text{ for } (S \to XA) \text{ being the initial matrix of } G,\\
w_2 &= d_1d_1'd_1''d_1'''d_1^{iv} \ldots d_kd_k'd_k''d_k'''d_k^{iv}d_{k+1}d_{k+1}'d_{k+1}''d_{k+1}''' \ldots d_nd_n'd_n''d_n''',\\
w_3 &= c_1c_2 \ldots c_nfg,\\
M_e &= N_1 \cup N_2 \cup \{a, Z', h, q\},\\
R_1 &= \{(d_iX, out)_f,\ (d_i'A, out)_f,\ (d_i'', out)_f,\ (d_i''', out)_f,\ (d_i^{iv}f, out),\\
&\qquad (d_iZ, out)_f,\ (d_i'Z, out)_f,\ (d_iq, in),\ (d_i'h, in),\ (d_i''\alpha_1, in),\\
&\qquad (d_i'''\alpha_2, in),\ (d_i^{iv}\alpha_3, in) \mid \text{ for } m_i : (X \to \alpha_1, A \to \alpha_2\alpha_3), 1 \leq i \leq k,\\
&\qquad \text{with } X, \alpha_1 \in N_1, A \in N_2, \alpha_2, \alpha_3 \in N_2 \cup \{a, \lambda\}\}\\
&\quad \cup \{(ZZ', in),\ (r, out)_f,\ (r, in)_h,\ (f, in)_r\}\\
&\quad \cup \{(d_iX, out)_f,\ (d_i'A, out)_f,\ (d_i'', out)_f,\\
&\qquad (d_i''', out)_f,\ (d_i^{iv}f, out),\ (d_iZ, out),\ (d_i'Z, out),\\
&\qquad (d_i''t, in),\ (d_i'''\alpha_2, in),\ (d_i^{iv}\alpha_3, in) \mid \text{ for } m_i : (X \to \lambda, A \to \alpha_2\alpha_3),\\
&\qquad 1 \leq i \leq k, \text{ with } X \in N_1, A \in N_2, \alpha_2, \alpha_3 \in \{a, \lambda\}\}\\
&\quad \cup \{(d_iX, out)_f,\ (d_iZ, out),\ (d_i'', out)_f,\ (d_i'''f, out),\\
&\qquad (d_iq, in),\ (d_i''Y, in),\ (d_i'''h, in) \mid \text{ for } m_i : (X \to Y, A \to \#),\\
&\qquad \text{with } k+1 \leq i \leq n, X, Y \in N_1, A \in N_2\},\\
R_2 &= \{(d_i, out)_{c_i},\ (d_i', out)_{c_i},\ (d_i'', out)_{c_i},\\
&\qquad (d_i'''f, out)_{c_i},\ (d_i^{iv}, out)_{c_i},\ (d_i, in)_q,\ (d_i', in)_q,\ (d_i'', in)_q,\\
&\qquad (d_i''', in)_q,\ (d_i^{iv}, in)_q \mid \text{ for } m_i : (X \to \alpha_1, A \to \alpha_2\alpha_3), 1 \leq i \leq k,\\
&\qquad \text{with } X, \alpha_1 \in N_1, A \in N_2, \alpha_2, \alpha_3 \in N_2 \cup \{a, \lambda\}\}\\
&\quad \cup \{(q, in),\ (f, in),\ (h, in)\}\\
&\quad \cup \{(d_i, out)_{c_i},\ (d_i', out)_{c_i},\ (d_i'', out)_{c_i},\ (d_i'''f, out)_{c_i},\ (d_i, in)_q,\\
&\qquad (d_i', in)_q,\ (d_i'', in)_q,\ (d_i''', in)_q,\ (d_i'A, in) \mid \text{ for } m_i : (X \to Y, A \to \#),\\
&\qquad k+1 \leq i \leq n, \text{ with } X, Y \in N_1, A \in N_2\},\\
&\quad \cup \{(tA, in) \mid A \in N_2\},\\
R_3 &= \{(fc_i, out),\ (c_i, in)_g \mid 1 \leq i \leq n\}\\
&\quad \cup \{(h, in),\ (q, in),\ (g, out)_f,\ (f, in)_g,\ (g, in)\}\\
&\quad \cup \{(tA, in),\ (tA, out) \mid A \in N_2\}\\
&\quad \cup \{(d_i'A, in),\ (d_i'A, out) \mid \text{ for } m_i : (X \to Y, A \to \#),\\
&\qquad k+1 \leq i \leq n, \text{ with } X, Y \in N_1, A \in N_2\},\\
R_4 &= \{(a, in),\ (Z', in),\ (Z', out)\}.
\end{aligned}
$$

Let us examine the work of this P system, when starting from a multiset of the form $XwZr$ present in membrane 1, for some $w \in (N_2 \cup \{a\})^*$ (at the beginning we have $w = A$, for $(S \to XA)$ the initial matrix of G), and with the multisets w_2, w_3 in membranes 2 and 3, respectively. All copies of a enter immediately membrane 4, hence from now on we will ignore this symbol.

The only applicable rules are $(g, out)_f$ and one of (fc_i, out) from R_3, for some $i \in \{1, 2, \ldots, n\}$. This i will determine the simulation of the matrix m_i from M.

At the next step, g returns to membrane 3 (and it will remain there, because f is no longer present in order to allow the use of the rule $(g, out)_f$), and c_i makes possible the exit of all symbols $d_i, d_i', d_i'', d_i''', d_i^{iv}$, for $1 \leq i \leq k$, or d_i, d_i', d_i'', d_i''', for $k+1 \leq i \leq n$, from membrane 2. Together with d_i''' also f exits membrane 2.

Now, all symbols $d_i, d_i', d_i'', d_i''', d_i^{iv}$ exit membrane 1 (d_i together with X, d_i' together with A, and d_i^{iv} together with f; in the same step, also r exits the system), and then these symbols return to this membrane together with the symbols $q, h, \alpha_1, \alpha_2, \alpha_3$, thus simulating the matrix $m_i : (X \to \alpha_1, A \to \alpha_2\alpha_3)$, with $X, \alpha_1 \in N_1$ and $A \in N_2, \alpha_2, \alpha_3 \in N_2 \cup \{a, \lambda\}$.

Note that if one of the symbols d_i, d_i' is not involved in this simulation, then the corresponding rule $(d_iZ, out), (d_i'Z, out) \in R_1$ should be used, and in this way the symbol Z' is brought into the system; the computation will continue forever by using the rules $(Z', in), (Z', out)$ from R_4. In this way, we ensure the fact that both rules of the matrix are used.

When returned to membrane 1, the symbols $d_i, d_i', d_i'', d_i''', d_i^{iv}$ cannot exit again, because the promoter f is not present, but they have to wait until either f will be here, or q will be present in membrane 2. Actually, this latter event will happen: q is immediately sent to membrane 2, in the step when h also enters membrane 2, and r enters membrane 1 (promoted by h). Because q is present, the symbols $d_i, d_i', d_i'', d_i''', d_i^{iv}$ enter membrane 2, and at the same time f enters membrane 1, promoted by r. At the next step, f enters membrane 2 and then membrane 3. In this way, we return to a configuration similar with the one we have started with, with all control symbols in their initial places, hence we can start simulating another matrix.

If the matrix was a terminal one, hence with $\alpha_1 = \lambda$, then q and h are not introduced into the system, hence also r and f remain outside. Instead, the symbol t is introduced, which will check whether or not any nonterminal of G is still present: by $(tA, in) \in R_3$, a nonterminal is introduced in membrane 2, then it will pass forever through membrane 3 by means of rules $(tA, in), (tA, out)$ from R_3.

When the symbol c_i sent out of membrane 3 was associated with a matrix $m_i : (X \to Y,\ A \to \#), k+1 \leq i \leq n$, with the second rule to be used in the appearance checking manner, then only the symbols d_i, d_i', d_i'', d_i''' are sent out of membrane 2, together with f. At the next step, the symbols d_i, d_i'', d_i''' exit membrane 1, together with X and f, and then return to the system together with the symbols Y, q, h. At the same time, d_i', remained in membrane 1, either enters to membrane 2 in the presence of A, if this symbol is present in membrane 1, or it waits. In the former case, the computation will continue forever, because of the rules $(d_i'A, in), (d_i'A, out)$ from R_3. The symbols q, h will help the symbols d_i, d_i', d_i'', d_i''' to enter membrane 3 and the symbol f to return to the system, in the same manner as explained above, when discussing matrices without appearance checking rules.

In this way, the use of the matrix m_i is correctly simulated, and we return to a configuration of the same form as that we have started with. The simulation of derivations in G can be iterated.

Therefore, we have $N(\Pi) = \{m \mid a^m \in L(G)\}$, which completes the proof.

The size of the families $NPP_m(psym_2, anti_0)$, for $m \leq 3$, remains to be investigated (we do not know whether or not the result in Theorem 5 is optimal in the number of membranes).

6 Final Remarks

The manner of "computing by osmosis", as done in P systems with symport and antiport rules, is interesting from many points of view (it has a good biological motivation, observes the conservation law, is computationally powerful and mathematically elegant, etc), hence it deserves further research efforts. A possible continuation of this paper is to consider other controls on the use of (symport) rules, starting with forbidding contexts, as already mentioned above, and proceeding with other ideas already investigated in the P system area: controlling the permeability of membranes, using a priority relation, and so on. We will return to this topic in a forthcoming paper.

References

[Alberts et al., 1998] Alberts, B., et al., "Essential Cell Biology. An Introduction to the Molecular Biology of the Cell", Garland Publ. Inc., New York/London (1998).

[Ardelean, 2002] Ardelean, I.I., "On the relevance of cell membranes for P systems; General aspects", Fundamenta Informaticae, 49 (2002), in press.

[Dassow and Păun, 1989] Dassow, J., Păun, Gh., "Regulated Rewriting in Formal Language Theory", Springer, Berlin (1989).

[Martin-Vide et al., 2002] Martin-Vide, C., Păun, Gh., Rozenberg, G., "Membrane systems with carriers", Theoretical Computer Sci., to appear.

[Păun, 2000] Păun, Gh., "Computing with membranes", Journal of Computer and System Sciences", 61, 1 (2000), 108–143.

[Păun and Păun, 2002] Păun, A., Păun, Gh., "The power of communication: P systems with symport/antiport", New Generation Computers, to appear.

[Păun et al., 2001] Păun, Gh., Perez-Jimenez, M., Sancho-Caparrini, F., "On the reachability problem for P systems with symport and antiport", submitted, 2001.

[Rozenberg and Salomaa, 1997] Rozenberg, G., Salomaa, A., eds., "Handbook of Formal Languages", 3 volumes, Springer, Berlin (1997).