

# On the Size of $P$ Systems with Minimal Symport/Antiport\*

György VASZIL

Computer and Automation Research Institute  
Hungarian Academy of Sciences  
Kende utca 13-17, 1111 Budapest, Hungary  
E-mail:vaszil@sztaki.hu

## Abstract

We show that  $P$  systems with symport/antiport rules sending at most one object per direction generate any recursively enumerable set of natural numbers with three membranes. This improves the previously known best bound of four membranes.

## 1 Introduction

Membrane systems, or  $P$  systems were introduced in [8] as computing models inspired by the functioning of the living cell. Their main components are membrane structures consisting of membranes hierarchically embedded in the outermost skin membrane. Each membrane encloses a region containing a multiset of objects and possibly other membranes. Each region has an associated set of operators operating on the objects contained by the region. These operators can be of different types, they can change the objects present in the regions or they can provide the possibility of transferring the objects from one region to another one. The evolution of the objects inside the membrane structure from an initial configuration to a somehow specified end configuration correspond to a computation having a result which is derived from some properties of the specific end configuration. Several variants of the basic notion have been introduced and studied proving the power of the framework, see the monograph [9] for a summary of notions and results of the area.

One of the most interesting variants of the model was introduced in [10] called  $P$  systems with symport/antiport. In these systems the modification of the objects present in the regions are not possible, they may only move through the membranes from one region to another. The movement is described by communication rules called symport/antiport rules associated to the regions. A symport rule specifies a multiset of objects that might travel through a given membrane in a given direction, an antiport rule specifies two multisets of objects which might simultaneously travel through a given membrane in the opposite directions. The result can be read as

---

\*Research supported in part by the Hungarian Scientific Research Fund "OTKA" grant no. F037567, and by the EU commission under the project "MolCoNet" IST-2001-32008.

the number of objects present inside a previously given output membrane after the system reaches a halting configuration, that is, a configuration when no application of any rule in any region is possible.

$P$  systems with symport/antiport were shown to be able to generate any recursively enumerable set of numbers already in [10]. This result was improved from the point of view of the number of necessary membranes and the complexity of communication rules in [5, 6, 7]. The study of minimal symport/antiport, when the multisets in the rules contain at most one object, started in [1] where it was shown that such systems with nine membranes generate any recursively enumerable set of numbers. Then the number of necessary membranes were decreased to six in [3], to five in [2], and then to four in [4].

In the present paper we continue to improve this result by showing that three membranes are sufficient to generate any recursively enumerable set of numbers with minimal symport/antiport.

## 2 Preliminaries and Definitions

We first recall the notions and the notations we use. The reader is assumed to be familiar with the basics of formal language theory, for details see [11]. Let  $V$  be an alphabet, let  $V^*$  be the set of all words over  $V$ , and let  $V^+ = V^* - \{\varepsilon\}$  where  $\varepsilon$  denotes the empty word. The set of natural numbers is denoted by  $\mathbb{N}$ , the class of recursively enumerable sets of natural numbers is denoted by  $NRE$ .

A multiset is a pair  $M = (V, f)$ , where  $V$  is an arbitrary (not necessarily finite) set of objects and  $f : V \rightarrow \mathbb{N}$  is a mapping which assigns to each object its multiplicity. The support of  $M = (V, f)$  is the set  $supp(M) = \{a \in V \mid f(a) \geq 1\}$ . If  $V$  is a finite set, then  $M$  is called a finite multiset. The set of all finite multisets over the set  $V$  is denoted by  $V^\circ$ .

The number of objects in a finite multiset  $M = (V, f)$ , the cardinality of  $M$ , is defined by  $card(M) = \sum_{a \in V} f(a)$ . We say that  $a \in M = (V, f)$  if  $a \in supp(M)$ .  $M_1 = (V_1, f_1) \subseteq M_2 = (V_2, f_2)$  if  $supp(M_1) \subseteq supp(M_2)$  and for all  $a \in V_1$ ,  $f_1(a) \leq f_2(a)$ . The union of two multisets is defined as  $(M_1 \cup M_2) = (V_1 \cup V_2, f')$  where for all  $a \in V_1 \cup V_2$ ,  $f'(a) = f_1(a) + f_2(a)$ . We say that  $M$  is empty, denoted by  $\epsilon$ , if its support is empty,  $supp(M) = \emptyset$ . In the following we enumerate the not necessarily distinct elements  $a_1, \dots, a_n$  of a multiset as  $M = \{\{a_1, \dots, a_n\}\}$ , by using double brackets to distinguish from the usual set notation.

A  $P$  system is a structure of hierarchically embedded membranes, each having a label and enclosing a region containing a multiset of objects and possibly other membranes. The out-most membrane which is unique and usually labelled with 1, is called the skin membrane. The membrane structure is denoted by a sequence of matching parentheses where the matching pairs have the same label as the membranes they represent. If  $x \in \{[i, ]_i \mid 1 \leq i \leq n\}^*$  is such a string of matching parentheses of length  $2n$ , denoting a structure where membrane  $i$  contains membrane  $j$ , then  $x = x_1 [i x_2 [j x_3 ]_j x_4 ]_i x_5$  for some  $x_k \in \{[l, ]_l \mid 1 \leq l \leq n, l \neq i, j\}^*$ ,  $1 \leq k \leq 5$ . If membrane  $i$  contains membrane  $j$ , and there is no other membrane,  $k$ , such that  $k$  contains  $j$  and  $i$  contains  $k$  ( $x_2$  and  $x_4$  above are strings of matching parentheses

themselves), then we say that membrane  $i$  is the parent membrane of  $j$ . A membrane  $m$  is called elementary, if there is no other membrane inside it, that is, if  $x = x_1 [m]_m x_2$ .

The evolution of the contents of the regions of a  $P$  system is described by rules associated to the regions. Applying the rules synchronously in each region, the system performs a computation by passing from one configuration to another one. In the following we concentrate on communication rules called symport or antiport rules.

A symport rule is of the form  $(x, in)$  or  $(x, out)$ ,  $x \in V^\circ$ . If such a rule is present in a region  $i$ , then the objects of the multiset  $x$  must enter from the parent region or must leave to the parent region. An antiport rule is of the form  $(x, in; y, out)$ ,  $x, y \in V^\circ$ , in this case, objects of  $x$  enter from the parent region and in the same step, objects of  $y$  leave to the parent region.

The rules are applied in the maximal parallel manner, that is, as many rules are applied in each region as possible. The end of the computation is defined by halting: A  $P$  system halts when no more rules can be applied in any of the regions, the result is the number of objects in an elementary membrane labelled as output.

**Definition 1** A  $P$  system with symport/antiport of degree  $n \geq 1$  is a construct

$$\Pi = (V, \mu, E, w_1, \dots, w_n, R_1, \dots, R_n, out)$$

where

- $V$  is an alphabet of objects,
- $\mu$  is a membrane structure of  $n$  membranes,
- $E \subseteq V$  is the set of objects which can be found in the environment in an arbitrary number of copies,
- $w_i \in V^\circ$ ,  $1 \leq i \leq n$ , are the initial contents of the  $n$  regions,
- $R_i$ ,  $1 \leq i \leq n$ , are the sets of symport/antiport rules associated to the regions,
- $out$  is the label of an elementary membrane, the output membrane.

To simplify the notations we denote symport and antiport rules as  $(x, in; y, out)$ ,  $x, y \in V^\circ$  where we also allow at most one of  $x, y$  to be the empty multiset. If  $y = \epsilon$  or  $x = \epsilon$ , then the notation above denotes the symport rule  $(x, in)$  or  $(y, out)$ , respectively.

The  $n + 1$ -tuple of finite multisets of objects present in finite number of copies in the environment and in the  $n$  regions of the  $P$  system  $\Pi$  describes a *configuration* of  $\Pi$ ;  $(\epsilon, w_1, \dots, w_n) \in (V^\circ)^{n+1}$  is the initial configuration.

**Definition 2** For a configuration  $(u_0, \dots, u_n)$ , the  $P$  system may enter the new configuration  $(u'_0, \dots, u'_n)$ , denoted as  $(u_0, \dots, u_n) \Rightarrow (u'_0, \dots, u'_n)$ , if there exist rules as follows.

For all  $i, 1 \leq i \leq n$ , there is a multiset of rules  $P_i = \{\{r_{i,1}, \dots, r_{i,m_i}\}\}$ , where  $r_{i,j} = (x_{i,j}, in; y_{i,j}, out) \in R_i$  satisfying the conditions below where  $x_i, y_i$  denote the multisets  $\bigcup_{1 \leq j \leq m_i} x_{i,j}$  and  $\bigcup_{1 \leq j \leq m_i} y_{i,j}$ , respectively. Furthermore, there is no  $r \in R_j$ , for any  $j, 1 \leq j \leq n$ , such that the rule multisets  $P'_i$  with  $P'_i = P_i$  for  $i \neq j$  and  $P'_j = \{\{r\}\} \cup P_j$ , also satisfy the conditions which are given as

$$x_1 = x'_1 \cup x''_1 \text{ with } \text{supp}(x'_1) \subseteq E, x''_1 \subseteq u_0, \text{ and}$$

$$\bigcup_{\text{parent}(j)=i} x_j \cup y_i \subseteq u_i, \text{ for } 1 \leq i \leq n.$$

Then the new configuration is obtained by

$$u'_0 = u_0 - x''_1 \cup y_1, \text{ and}$$

$$u'_i = u_i \cup x_i - y_i \cup \bigcup_{\text{parent}(j)=i} y_j - \bigcup_{\text{parent}(j)=i} x_j, \text{ for } 1 \leq i \leq n.$$

The  $P$  system generates numbers as follows.

**Definition 3** The set of natural numbers generated by a symport/antiport  $P$  system as above,  $N(\Pi) \in \mathbb{N}RE$ , is the following set.

$$N(\Pi) = \{x = \text{card}(u_{out}) \mid (\epsilon, w_1, \dots, w_n) \Rightarrow^* (u_0, \dots, u_n), \\ \text{where } (u_0, \dots, u_n) \text{ is a halting configuration}\}$$

and  $\Rightarrow^*$  denotes the reflexive and transitive closure of  $\Rightarrow$ .

Let  $NO P_n(\text{sym}_r, \text{anti}_s)$  denote the class of sets of numbers generated by symport/antiport  $P$  systems of degree  $n$  where for all  $(x, in), (y, out), (v, in; z, out) \in R_i, 1 \leq i \leq n, \text{card}(x) \leq r, \text{card}(y) \leq r,$  and  $\text{card}(v) \leq s, \text{card}(z) \leq s$ .

Before we proceed, we need the notion of a counter automaton which will be used in the proof of our result. We present the definition in the form given in [4], for more details see [4] and its references.

**Definition 4** A counter automaton is a construct

$$M = (Q, C, R, q_0, f)$$

where  $Q$  is a set of states,  $C = \{c_0, c_1, \dots, c_n\}$  is a set of counters,  $c_0$  being the output counter,  $R$  is a set of transitions of the form  $(r \rightarrow s, X)$ , for two states  $r, s \in Q$ , and an instruction  $X \in \{i+, i-, e, i = 0\}$ ,  $q_0 \in Q$  is the initial state, and  $f \in Q$  is the final state.

If a transition  $(r \rightarrow s, X)$  is an element of  $R$ , then the machine can pass from state  $r$  to state  $s$  executing  $X$ . If  $X$  is  $i+$  or  $i-$ , then it instructs the machine to increase or decrease the value of counter  $c_i$  by one, if it is  $e$ , then it instructs the machine to leave the counter values unchanged, if it is  $i = 0$ , then the transition from  $r$  to  $s$  is only possible by having zero as the contents of counter  $c_i$ .

**Definition 5** The configuration of a counter automaton  $M$  is given by the  $(n+2)$ -tuple  $(q, c_0, c_1, \dots, c_n)$  where  $q \in Q$  is a state, and  $c_i \in \mathbb{N}$ ,  $0 \leq i \leq n$ , are the values stored in the counters,  $c_0$  being the value of the output counter. The initial configuration is  $(q_0, 0, 0, \dots, 0)$ , a final configuration is of the form  $(f, c_0, c_1, \dots, c_n)$  where  $f$  is the final state of  $M$ .

Given a configuration  $(q, c_0, c_1, \dots, c_n)$ , the machine can pass to configuration  $(q', c'_0, c'_1, \dots, c'_n)$ , denoted as  $(q, c_0, c_1, \dots, c_n) \Rightarrow (q', c'_0, c'_1, \dots, c'_n)$ , if  $(q \rightarrow q', X) \in R$ , and

- if  $X = j+$ , then  $c'_j = c_j + 1$  and  $c'_i = c_i$ ,  $0 \leq i \leq n$ ,  $i \neq j$ ,
- if  $X = j-$ , then  $c'_j = c_j - 1$  and  $c'_i = c_i$ ,  $0 \leq i \leq n$ ,  $i \neq j$ ,
- if  $X = e$ , then  $c'_i = c_i$ ,  $0 \leq i \leq n$ ,
- if  $X = (j = 0)$ , then  $c'_i = c_i$ ,  $0 \leq i \leq n$ , and  $c_j = 0$ .

Let  $\Rightarrow^*$  denote the reflexive and transitive closure of  $\Rightarrow$ .

A computation is a sequence of such transitions leading from the initial configuration to a final configuration, its result can be read from the output counter.

**Definition 6** The set of natural numbers generated by a counter automaton  $M$  as above is the following.

$$N(M) = \{c_0 \in \mathbb{N} \mid (q_0, 0, 0, \dots, 0) \Rightarrow^* (f, c_0, c_1, \dots, c_n), \text{ where } q_0 \text{ and } f \text{ are the initial and the final states, respectively}\}.$$

It is known that counter automata are able to generate any recursively enumerable set of numbers if they have two or more counters beside the output counter.

### 3 The Number of Membranes

In this section we show that  $P$  systems with minimal symport/antiport generate any recursively enumerable set of numbers with three membranes. To do this, we will show how these systems simulate the computations of counter automata.

**Theorem 1**  $NOP_3(sym_1, anti_1) = \mathbb{N}RE$ .

*Proof.* Consider the counter automaton  $M = (Q, C, R, q_0, f)$  with counters  $C = \{c_0, c_1, \dots, c_n\}$  as above,  $c_0$  being the output counter. We construct a  $P$  system  $\Pi$  with minimal symport/antiport generating the language  $L(\Pi) = \{x+4 \mid x \in L(M)\}$  as follows. Let

$$\Pi = (V, \mu, E, w_1, w_2, w_3, R_1, R_2, R_3, 3)$$

where  $\mu = [1 [2 [3 ]_3 ]_2 ]_1$ , and

$$\begin{aligned} V = & \{I_1, I_2, I_3, I'_3, I_4, I_5, \infty_1, \infty_2, \infty_3, \infty_4, C, f_1, f'_1, f_2, f'_2, \bar{f}_2, f_3, f'_3, f_4\} \cup \\ & \{qr, qr' \mid (q \rightarrow r, X) \in R \text{ for some } X \in \{i+, i-, i = 0, e\}\} \cup \\ & \{c_i, 0_i \mid 0 \leq i \leq n\}, \end{aligned}$$

$$\begin{aligned} E = & \{qr, qr' \mid (q \rightarrow r, X) \in R \text{ for some } X \in \{i+, i-, i = 0, e\}\} \cup \\ & \{I_4, f_1, f_2, \bar{f}_2, f_3, f_4\} \cup \{c_i \mid 0 \leq i \leq n\}. \end{aligned}$$

The initial region contents are

$$\begin{aligned} w_1 &= \{I_1, \bar{I}_1, I_2, I_3, \infty_1, \infty_2, \infty_4, \infty_4, C\}, \\ w_2 &= \{\infty_1, \infty_2, \infty_3, \infty_3\} \cup \{0_i \mid 0 \leq i \leq n\}, \text{ and} \\ w_3 &= \{I_5, f'_1, f'_2, f'_3\}. \end{aligned}$$

The work of the  $P$  system can be divided into three phases:

- Initialization,
- simulation of the counter automaton, and
- termination.

In the *initialization phase* an arbitrary number of counter symbols  $c_i, 0 \leq i \leq n$ , are moved into region 1 and an arbitrary number of transition symbols  $qr'$  for some  $q, r \in Q$  are moved into region 3.

In the *simulation phase*  $\Pi$  simulates  $M$  by modifying the number of counter symbols present in region 2 according to the counter contents of  $M$  as follows. First  $\Pi$  imports a transition symbol  $qr$  for a possible transition  $(q \rightarrow r, X)$  of  $M$  into region 1. Then this symbol travels to region 2 and then to region 3 where it remains until the termination phase, and from where the primed version,  $qr'$ , is released to region 2 moving to region 1 where it is sent out to the environment and at the same time an other transition symbol  $rs$  is imported for an other valid transition  $(r \rightarrow s, Y)$  of  $M$ . While these transition symbols travel through the system to region 3 and back, the modifications on the number of counter symbols in region 2 are realized. If  $X = i-$ , then a copy of  $c_i$  is removed from region 2 when  $qr$  enters this region from region 1. If  $X = i+$ , then a copy of  $c_i$  is imported from region 1 to region 2 when  $qr'$  moves from region 2 to region 1. For  $X = (i = 0)$ , a simple mechanism is used which, through the aid of maximal parallel rule application, allows the above described movement of the transition symbols only in the case when region 2 does not contain any symbol  $c_i$ .

In the *termination phase*, the counter symbols corresponding to the output counter are moved to region three, then the possibly still present transition symbols of the form  $qr$  or  $qr'$  for some  $q, r \in Q$  are moved from region 3 to region 2. In case of an unsuccessful simulation,  $\Pi$  may never stop which is ensured by an infinite loop: A pair of  $\infty_1$  symbols are present in region 1 and 2, together with the rule

$(\infty_1, in; \infty_1, out)$  in region 2. This loop is “destroyed” only in the termination phase after a successful simulation allowing the computation to stop.

For the sake of easier readability we present the rules of  $\Pi$  in groups corresponding to these phases  $R_i = R_i^{ini} \cup R_i^{sim} \cup R_i^{ter}$ ,  $1 \leq i \leq 3$ . For  $j$ ,  $0 \leq j \leq n$ ,

$$\begin{aligned} (qr', in; I_1, out), (I_1, in), (I_4, in; I_1, out), (c_j, in; \bar{I}_1, out), (\bar{I}_1, in) &\in R_1^{ini} \\ (I_2, in), (qr', in; I_2, out), (I_3, in; \infty_2, out) &\in R_2^{ini} \\ (I_3, in), (qr', in; I_3, out), (\infty_3, in; I_3, out) &\in R_3^{ini} \end{aligned}$$

With the help of the initialization symbols  $I_1, \bar{I}_1, I_2, I_3 \in w_1$ , these rules import an arbitrary number of transition symbols  $qr'$  with  $q, r \in Q$  into region 3 and counter symbols  $c_i$ ,  $0 \leq i \leq n$ , into region 1. In the first step,  $I_1$  and  $\bar{I}_1$  are moved out of the system,  $I_2$  and  $I_3$  are moved to region 2. Since there will be other rules in region 1 for  $I_3$ , it is necessary to make sure that it is moved to region 2 by sending out the symbol  $\infty_2$ . If  $\infty_2$  is not sent out, an infinite loop is formed which can not be later destroyed. By applying these rules in succession, the imported transition symbols are moved to region 3, the counter symbols remain in region 1. If for some reason, because of the application of some other rule, a transition symbol cannot be moved to region 3 from region 2, then  $\infty_3$  is moved into region 3 instead, creating an infinite loop. Another infinite loop involving the two  $\infty_1$  objects keeps the system working until a correct simulation of a successful computation is finished, then it is removed, otherwise if the simulation does not follow the right track, the system will produce no result. These infinite loops need rules

$$\begin{aligned} (\infty_1, in; \infty_1, out), (\infty_2, in; \infty_2, out) &\in R_2^{ini} \\ (\infty_3, in; \infty_3, out) &\in R_3^{ini} \end{aligned}$$

If once in region 1, instead of a transition symbol,  $I_4$  is imported, then the initialization phase is finished using the following rules.

$$\begin{aligned} (I_3, out) &\in R_1^{ini} \\ (I_4, in; I_3, out), (I_1, in; I_4, out), (\infty_4, in; I_4, out), (\bar{I}_1, in; I_5, out) &\in R_2^{ini} \\ (I_1, in; I_5, out), (I_2, in; I_1, out) &\in R_3^{ini} \end{aligned}$$

First, to avoid the creation of an infinite loop involving two  $\infty_3$  objects in region 2 and 3, the symbol  $I_3$  is moved out from region 2 to region 1 and at the same time  $I_4$  is moved from region 1 to region 2. Then  $I_3$  leaves the system, and  $I_4$  is sent back to region 1 while moving  $I_1$  to region 2. Since there are other rules for  $I_1$  in region 1, an infinite loop is created if in this step it is not moved to region 2. Now  $I_1$  is transferred to region 3, where it brings also  $I_2$  to region 3, and releases  $I_5$  which moves to region 1 while bringing  $\bar{I}_1$  to region 2. The infinite loop needs the rule

$$(\infty_4, in; \infty_4, out) \in R_2^{ini}$$

Thus, at the end of the initialization phase, the system ends up in a configuration where  $u_1, u_2, u_3$  are the multisets contained by the three regions as

$$u_1 = \{\{c_{i_1}, \dots, c_{i_k} \mid i_j \in \{0, \dots, n\}, 1 \leq j \leq k\}\} \cup$$

$$\{\{I_4, I_5, \infty_2, \infty_2, \infty_4, \infty_4, \infty_1, C\}\},$$

$$u_2 = \{\{I_1, \bar{I}_1, \infty_1, 0_0, 0_1, \dots, 0_n, \infty_3, \infty_3\}\}, \text{ and}$$

$$u_3 = \{\{q_1 r'_1, \dots, q_m r'_m \mid q_j, r_j \in Q, 1 \leq j \leq m\}\} \cup \{\{I_2, f'_1, f'_2, f'_3\}\}.$$

The simulation of the counter automaton is realized with the following rules.

$$R_1^{sim} = \{(q_0 q, in; I_5, out), (rs, in; qr', out) \mid q_0, q, r, s \in Q \text{ and} \\ (q_0 \rightarrow q, X), (q \rightarrow r, Y), (r \rightarrow s, Z) \in R \text{ for some } X, Y, Z\},$$

$$R_2^{sim} = \{(rs, in), (c_i, in; rs', out) \mid (r \rightarrow s, i+) \in R\} \cup \\ \{(rs, in; c_i, out), (rs', out) \mid (r \rightarrow s, i-) \in R\} \cup \\ \{(rs, in), (rs', out) \mid (r \rightarrow s, e) \in R\} \cup \\ \{(rs, in; 0_i, out), (0_i, in; c_i, out), (0_i, in; rs', out) \mid (r \rightarrow s, i = 0) \in R\},$$

$$R_3^{sim} = \{(rs, in; rs', out) \mid (r \rightarrow s, X) \in R \text{ for some } X\}.$$

First  $I_5$  is sent out of the system and one transition symbol, denoting a transition from the initial state  $q_0$  is imported, then the transition corresponding to the symbol is simulated. This is done by moving the transition symbol to the third region, exchanging it to its primed version, and moving the primed version back to region 1. While the transition symbol travels through the regions, it adds or subtracts a counter symbol to or from region 2 when necessary. If the instruction corresponding to the simulated transition is  $i = 0$ , then the above described movement of the transition symbol is only possible if there are no  $c_i$  counter symbols present in region 2.

If the system simulates a transition to the final state, it may enter the terminating phase. In this phase the following rules are used.

$$R_1^{ter} = \{(f_1, in; qf', out), (\bar{f}_2, in; f'_1, out), (\bar{f}_2, in; \bar{f}_2, out), (f_2, in; \bar{f}_2, out), \\ (f_3, in; f'_2, out), (f_4, in; f'_3, out)\},$$

$$R_2^{ter} = \{(f_1, in), (C, in; f'_1, out), (f_2, in; 0_0, out), (0_0, in; c_0, out), \\ (0_0, in; f'_2, out), (f_3, in; C, out), (f'_3, out), (f_4, in; \infty_1, out)\},$$

$$R_3^{ter} = \{(f_1, in; f'_1, out), (C, in), (C, out), (c_0, in; C, out), (f_2, in; f'_2, out) \\ (f_3, in; f'_3, out), (f_4, in; rs'out), (f_4, in; rs, out), (f_4, out)\}.$$

During the terminating phase, symbols  $f_1, f_2, f_3$ , and  $f_4$  travel through the system, each performing a specific task. First, after a transition symbol of the form  $qf'$ ,  $f_1$  is imported into region 1. It moves to region 3 where  $f'_1$  is released which moves to

region 1 bringing  $C$  to region 2. The task of  $C$  is to move all  $c_0$  counter symbols corresponding to the output counter to region 3. This may take several steps, so  $f'_1$  is exchanged with  $\bar{f}_2$  in region 1, and  $\bar{f}_2$  can move in and out of the system for an arbitrary amount of time. When  $\bar{f}_2$  is exchanged with  $f_2$ , the termination process continues. The travel of  $f_2$  is only possible if there is no  $c_0$  present in region 2. In this case, after  $f'_2$  leaves the system,  $f_3$  is imported. While  $f_3$  moves through the system it removes  $C$  from region 2, so no further movement of possibly newly appearing  $c_0$  will be allowed to region 3, then, when  $f'_3$  leaves the system,  $f_4$  is introduced. When  $f_4$  moves to region 2, it removes  $\infty_1$ , thus removes the infinite loop, and then it also removes the remaining transition symbols from region 3. When all of these symbols are out of region 3, the system stops working, having only the counter symbols plus four other symbols,  $f_1, f_2, f_3$ , and  $I_2$  in region 3, the output region, thus producing a result  $x \in \mathbb{N}$  for some  $(x - 4) \in L(M)$ .  $\square$

## 4 Conclusion

We have shown how to simulate counter automata using  $P$  systems with minimal symport/antiport and three membranes, thus we have improved the previously known best result stating that four membranes are sufficient to reach this power. The optimality of our result is still to be demonstrated, but we conjecture that it cannot be further improved.

## References

- [1] F. Bernardini, M. Gheorghe. On the power of minimal symport/antiport. In: *Workshop on Membrane Computing, WMC-2003, Tarragona, July 17-22, 2003*. Edited by A. Alhazov, C. Martín-Vide, Gh. Păun. Technical Report 28/03 of the Research Group on Mathematical Linguistics, Rovira i Virgili University, Tarragona, Spain, 2003, 72-83.
- [2] F. Bernardini, A. Păun. Universality of minimal symport/antiport: Five membranes suffice. In: *Membrane Computing. International Workshop WMC-CdeA, Curtea de Arges, Romania, August 19-23, 2002. Revised Papers*. Volume 2597 of *Lecture Notes in Computer Science*, edited by Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron. Springer-Verlag, 2003, 43-54.
- [3] L. Kari, C. Martín-Vide, A. Păun. On the universality of  $P$  systems with minimal symport/antiport rules. In: *Aspects of Molecular Computing, Essays Dedicated to Tom Head on the Occasion of His 70th Birthday*. Volume 2950 of *Lecture Notes in Computer Science*, edited by N. Jonoska, Gh. Păun, G. Rozenberg. Springer-Verlag, 2004, 254-265.
- [4] P. Frisco. About  $P$  systems with symport/antiport. In: *Second Brainstorming Week in Membrane Computing. Sevilla, February 2-7, 2004*. Edited by Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini. Technical Report 01/2004 of the Research Group in Natural Computing, University of Sevilla, Spain, 2004, 224-236.

- [5] P. Frisco, H. J. Hoogeboom. P systems with symport/antiport simulating counter automata. Submitted.
- [6] C. Martín-Vide, A. Păun, Gh. Păun. On the power of P systems with symport rules. *Journal of Universal Computer Science*, 8:317-331, 2002.
- [7] C. Martín-Vide, A. Păun, Gh. Păun, G. Rozenberg. Membrane systems with coupled transport. *Fundamenta Informaticae*, 49:1-15, 2002.
- [8] Gh. Păun. Computing with Membranes. *Journal of Computer and System Sciences*, 61(1):108-143, 2000.
- [9] Gh. Păun, *Computing with Membranes: An Introduction*. Springer-Verlag, Berlin, 2002.
- [10] A. Păun, Gh. Păun. The power of communication: P systems with symport/antiport. *New Generation Computing*, 20(3):295-306, 2002.
- [11] G. Rozenberg, A. Salomaa (eds.) *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.