Further Remarks on Trace Languages in P Systems with Symport/Antiport

Guangwu Liu[1,2], Mihai Ionescu[1]

[1] Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tarraco 1, 43005 Tarragona, Spain
`guangwu.liu@urv.net, armandmihai.ionescu@urv.net`

[2] Department of Control Science and Engineering
Huazhong University of Science and Technology
Wuhan 430074, Hubei, P.R.China

Summary. P systems are parallel molecular computing models which process multisets of objects in cell-like membrane structures. In this paper we consider the trace languages of a special symbol, *the traveler*, in symport/antiport P systems where, instead of *multisets* of objects, *sets* of objects were considered. Two different ways to define the trace language are proposed. One of the families of languages obtained in this way is proved to be equal to the family of regular languages and the other one to be strictly smaller. Some ideas for further research are also considered.

1 Introduction

This paper is a contribution to the study of P systems where the trace of an object (traveler), as introduced in [7] and further investigated in [6, 5], is considered. The model belongs to the area of P systems with purely communicative functioning, as introduced in [8], inspired from the biological processes of symport and antiport – see [1], [4] for biochemical details about these kinds of trans-membrane transfer of chemicals (in short, symport is the process in which two molecules pass together, in the same direction through a membrane, while antiport refers to the process when two molecules pass simultaneously through a membrane, but in opposite directions). As explained in [7], we are not interested now in the *number* of certain objects present in a certain membrane, or in the environment of the system (as in the classical definition of P systems), but in the *itineraries* of a unique object, *the traveler*, through the membranes of the system, and in the result (a string of labels of the visited membranes) produced by these itineraries. In this paper, we investigate the power of the family of languages generated by such systems considering *sets* of objects instead of *multisets* of objects (as in the classical variants of P systems).

We recall the reader that 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 enclosed within the *skin* membrane, and delimiting *regions* (see Figure 1) where multisets of *objects* (which evolve according to given *evolution rules*) are placed. The rules are applied nondeterministically, in a maximally parallel manner.

In this way we obtain *transitions* from a *configuration* of the system to the next one. A sequence of transitions constitutes a *computation*; to each *halting computation* we associate a *result*, the number of objects from a specified *output membrane*.

For more details on various variants of P systems we refer to [10] and to the papers available at the web address `http://psystems.disco.unimib.it/`.

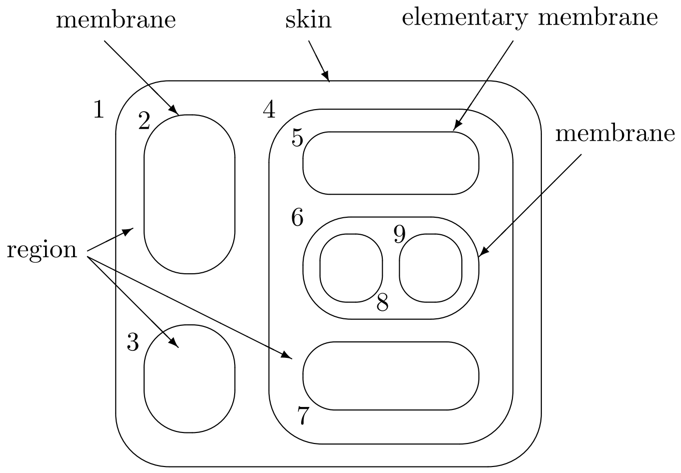


Figure 1: A membrane structure

The P systems models which we consider in this paper are inspired by the real life phenomenon 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*.

Technically, the formalization in P systems of these biological processes is made through the following rules: (x, in) and (x, out) model the symport, and $(x, out; y, in)$ models the antiport (x, y are strings of symbols representing multisets of chemicals). Of course, this is a generalization of what happens in biology, where mainly pairs of chemicals are coupled. Several classes of P systems of this type were considered. Because such systems work with multisets of objects, it is natural that the result of a computation is a number (or a vector of numbers), describing the multiset of objects present at the end of a computation in a specified output membrane.

In the present paper we use another idea: we take into consideration the string of labels of membranes visited by a specified object during a computation, this time not working with multisets of objects but with sets of objects. In this way, a P system will generate a language. In Section 4 we prove that the family of languages generated by P systems with traces, with the traveler marking only once the visited membrane (when entering it) and symport/antiport rules equals REG, the family of regular languages. The variant of P systems (with symport/antiport rules) in which the traveler marks twice the visited membrane (once for entering the membrane and once for getting out of it) generates a family of languages strictly included in REG.

2 P Systems with Symport/Antiport

The language theory notions we use here are standard, and can be found, for instance, in [11], [12]. We only mention that we denote by V^* the free monoid generated by an alphabet V; λ is the empty string, $|x|$ is the length of $x \in V^*$, and $|x|_a$ is the number of occurrences of the symbol $a \in V$ in the string $x \in V^*$. For $x \in V^*$ we denote $alph(x) = \{a \in V \mid |x|_a \geq 1\}$ (the set of symbols appearing in x), and for a language $L \subseteq V^*$ we write $alph(L) = \bigcup_{x \in L} alph(x)$. By REG we denote the family of regular languages.

A membrane structure is pictorially represented by an Euler-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 $M : X \longrightarrow \mathbf{N}$. Here we always use multisets over finite sets X (that is, X will be an alphabet). A multiset with a finite support 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 first introduce the standard P systems with symport/antiport rules, with multisets of objects, computing sets of numbers. Such a device is a construct

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

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 $1, 2, \ldots, m$); $m \geq 1$ 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 E is the set of objects which are supposed to be continuously present outside the system, in the *environment*, in an arbitrary number of copies;

4. $R_1, \ldots, R_m$ are finite sets of *rules* of the forms $(x, in), (x, out)$, and $(x, out; y, in)$, for $x, y \in V^*$;
5. $i_o \in \{1, \ldots, m\}$ is an elementary membrane of μ (the output membrane).

The rules from a set R_i are used with respect to membrane i as suggested above. In the case of (x, in), the multiset of objects x enters the region defined by the membrane, from the immediately upper region; this is the environment when the rule is associated with the skin membrane. In the case of (x, out), the objects specified by x are sent out of membrane i, into the region immediately outside; this is the environment in the case of the skin membrane. The use of a rule $(x, out; y, in)$ means expelling from membrane i the objects specified by x at the same time with bringing in membrane i the objects specified by y. The objects from E are supposed to appear in arbitrarily many copies in the environment (because we only move objects from a membrane to another membrane, hence we do not create new objects in the system, we need a supply of objects in order to compute with arbitrarily large multisets). The rules are used in the nondeterministic maximally parallel manner specific to P systems with symbol-objects. In this way, we obtain transitions between the configurations of the system. A configuration is described by the m-tuple of the multisets of objects present in the m regions of the system, as well as the multiset of objects which were sent out of the system during the computation, others than the objects appearing in the set E; it is important to keep track of such objects because they appear in a finite number of copies in the initial configuration and can enter again the system. We do not need to take care of the objects from E which leave the system because they appear in arbitrarily many copies in the environment (the environment is supposed inexhaustible, irrespective how many copies of an object from E are introduced into the system, still arbitrarily many remain in the environment). The initial configuration is $(w_1, \ldots, w_m, \lambda)$. Therefore, 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 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) or (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 $|x|, |y| \leq q$ (we say that $\max(|x|, |y|)$ is the *weight* of the antiport rule $(x, out; y, in)$), is denoted by $NOP_m(sym_p, anti_q)$, for $m \geq 1$ and $p, q \geq 0$. When the number of membranes is not bounded we replace the subscript m by $*$, and when the weight of rules is not bounded we replace the subscripts p and q by $*$.

We use NRE to denote the family of recursively enumerable sets of natural numbers (that is, the family of the length sets of recursively enumerable languages).

The following results are the best known in this moment (see [5], [3]):

$$NRE = NOP_3(sym_2) = NOP_3(sym_1, anti_1) = NOP_1(anti_2).$$

3 Considering the Trace of Certain Objects

In [7], P systems of the following form were considered:

$$\Pi = (V, t, T, h, \mu, w_1, \dots, w_m, E, R_1, \dots, R_m),$$

where all components $V, \mu, w_1, \dots, w_m, E, R_1, \dots, R_m$ are as above, $t \in V$ (a distinguished object, "the traveler"), T is an alphabet, and $h : \{1, 2, \dots, m\} \longrightarrow T \cup \{\lambda\}$ is a weak coding. The traveler is present in exactly one copy in the system, that is, $|w_1 \dots w_m|_t = 1$ and $t \notin E$.

Let $\sigma = C_1C_2 \dots C_k, k \geq 1$, be a halting computation with respect to Π, with $C_1 = (w_1, \dots, w_m, \lambda)$ the initial configuration, and $C_i = (z_1^{(i)}, \dots, z_m^{(i)}, z_e^{(i)})$ the configuration at step $i, 1 \leq i \leq k$. If $|z_j^{(i)}|_t = 1$ for some $1 \leq j \leq m$, then we write $C_i(t) = j$ (therefore, $C_i(t)$ is the label of the membrane where t is placed). If $|z_j^{(i)}|_t = 0$ for all $j = 1, 2, \dots, m$, then we put $C_i(t) = \lambda$. Then, *the trace* of t in the computation σ is

$$trace(t, \sigma) = C_1(t)C_2(t) \dots C_k(t).$$

The computation σ is said to generate the string $h(trace(t, \sigma))$, hence the language generated by Π is $L(\Pi) = \{h(trace(t, \sigma)) \mid \sigma \text{ is a halting computation in } \Pi\}$.

We denote by $LTP_m(sym_p, anti_q)$ the family of trace languages $L(\Pi)$ generated by P systems with at most m membranes and using symport rules of weight at most p and antiport rules of weight at most q. When one of the parameters m, p, q is not bounded, we replace the respective subscript with $*$.

The power of these systems was investigated in [7, 6] and the currently best result (stated in [5]) is the following:

$$RE = LTP_*(sym_0, anti_2).$$

In [5] the following results are also proved:

$$\begin{aligned} lRE &= lLTP_{l+1}(sym_0, anti_2), \\ lRE &= lLTP_{l+1}(sym_3, anti_0), \\ lRE &= lLTP_{l+2}(sym_2, anti_0), \end{aligned}$$

where the parameter l in front of language families indicates that only languages L with $card(alph(L)) \leq l$ are considered.

4 Trace Languages in Symport/Antiport P Systems with Sets of Objects

In this section, we consider P systems whose regions contain finite *sets* of objects, not *multisets* as in the classical variant of P systems (such systems were investigated in [2] as number generating devices); moreover, we assume that the environment is empty.

Such a system is a construction

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

where:

1. V is the alphabet of chemicals (objects);
2. $t \in V$ is the *traveler.* There is exactly one traveler t in the system, that is, $|w_1 \cdots w_m|_t = 1$;
3. μ is a membrane structure with m membranes (injectively labeled by positive integers $1, 2, \ldots, m$);
4. w_i are strings representing the sets of objects present in the regions of μ, $1 \leq i \leq m$;
5. R_i is the set of symport and antiport rules associated with the membrane i; they have the forms $(x, in), (x, out)$ and $(x, out; y, in)$, for $x, y \in V^*$, $1 \leq i \leq m$.

The trace of the traveler across membranes is encoded as a string over the alphabet $\{a_1, a_2, \ldots, a_m\}$, by recording, in order, every membrane visited by t.

We consider two different cases of marking the trace. In the first one, we count only the event of the traveler entering a membrane, and in this case we denote by $LTP_m^{set}(sym_p, anti_q, in)$ the family of languages generated by P systems with traces and symport/antiport rules over finite set of objects, using at most m membranes, symport rules of weight at most p and antiport rules of weight at most q. In the second case we take into account both the fact that the traveler enters a membrane and that it exits it (so we collect twice the label of the membrane t visits). We denote by $LTP_m^{set}(sym_p, anti_q, in/out)$ the family of languages generated in this case, with the usual meaning of the parameters m, p, q.

In what follows we investigate the place of these families with respect to Chomsky hierarchy.

Theorem 1. $LTP_*^{set}(sym_*, anti_*, in) = REG.$

Proof. Given a P system $\Pi = (V, t, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m)$ we can construct a regular grammar $G = (C, T, C_0, R)$, where C is the set of all configurations which can be reached by Π starting from the initial configuration (this set is finite, because the system only handles a finite number of objects), C_0 is the initial configuration, and $T = \{a_1, a_2, \ldots, a_m\}$. The rules of the grammar are constructed as follows.

1. $C_i \to C_j$ if $C_i \to C_j$ is a transition and the traveler does not enter any membrane;
2. $C_i \to a_k C_j$ if $C_i \to C_j$ is a transition and the traveler enters membrane k $(1 \leq k \leq m)$;
3. $C_i \to \lambda$ if C_i is a halting configuration.

It is obvious that $trace(\Pi) = L(G)$ which implies $LTP_*^{set}(sym_*, anti_*, in) \subseteq REG$.

Conversely, given a regular grammar $G = (N, T, S, R)$ (N is the set of nonterminals, $T = \{a_1, a_2, \ldots, a_m\}$, $S \in N$, and R is the set of productions of the form $A \to a_i B$, and $A \to a_i$, with $A, B \in N, a_i \in T$), we can construct a P system as follows. The initial configuration (in the bracketed form) is $[\,cSt[\,dNN'\,]_0[\,f_1\,]_1[\,f_2\,]_2 \ldots [\,f_m\,]_m\,]_s$, $\{s, 0, 1, 2, \ldots, m\}$ is the set of labels for membranes, s is the label of the skin membrane, f_i $(1 \leq i \leq m)$ is the symbol within membrane i, dNN' is the (strings which describes the) set of objects in membrane 0 (N is the nonterminal alphabet of grammar G, N' is the set of primed versions of elements of N, and $d \in V$), cSt is the set of objects in the skin membrane, where t is the traveler, S is the initial symbol of grammar G, and $c \in V$.

In order to simulate a rule $A \to a_i B \in R$ we will use the following rules:

step	R_0	R_i
1		$(f_i, out; cAt, in)$
2	$(d, out; f_i, in)$	(At, out)
3	$(f_i B', out; A, in)$	$(c, out; d, in)$
4		$(d, out; f_i, in)$
5	$(B, out; dB', in)$	

where R_0 and R_i are the sets of rules associated with membranes labeled 0 and i, respectively. For the other membranes no rules are specified.

The process of simulating a rule $A \to a_i B$ is detailed below.

In the first step, the antiport rule $(f_i, out; cAt, in)$ makes the traveler t (altogether with objects c and A) enter membrane labeled i, thus introducing the symbol a_i in the trace. In the same time, symbol f_i, initially present in membrane i, is expelled within the skin membrane.

What is left to simulate is the process of transforming A to B and to bring the system back to its starting configuration, in order to be ready for a new simulation of a rule.

In the second step of computation, rules $(d, out; f_i, in)$, and (At, out) indicate that our traveler goes out membrane i (altogether with object A), while f_i enters membrane labeled 0, with the help of object d (its counterpart in the antiport rule).

In the next step, the position of A and B' in the system is interchanged, in the same time with expelling from membrane labeled 0 the symbol f_i. The position of objects c, and d is also modified by the antiport rule that applies for the membrane labeled i. In this moment, our system has the following configuration:

$$[f_i cB't[N\{N'-B'\}]_0[f_1]_1[f_2]_2\ldots[d]_i\ldots[f_m]_m]_s.$$

We have now succeeded in rewriting A to B' (which is a copy of B). The following thing to accomplish is to make the system gain its initial configuration, this time with B in the place of A so the derivation could continue.

In step 4, we use rule $(d, out; f_i, in)$ to move back f_i in its initial place. In the last step of the computation, rule $(B, out; dB', in)$ sends d and B' in membrane labeled 0 while B is expelled from membrane labeled 0 to the skin membrane, and the system can now continue to simulate the derivation process.

We can continue the above process to simulate nonterminal rules of G.

A rule $D \to a_i$ is simulated using the rule

$$(f_i, out; cDt, in) \in R_i.$$

The work of this rule is obvious.

The derivation grammar G ends by using such a rule, hence also our system will halt.

Clearly, $trace(\Pi) = L(G)$, hence we also have the inclusion $REG \subseteq LTP_*^{set}(sym_*, anti_*, in)$. □

If we consider the degree of the P system and the weight of the rules, we can write the previous result in the form

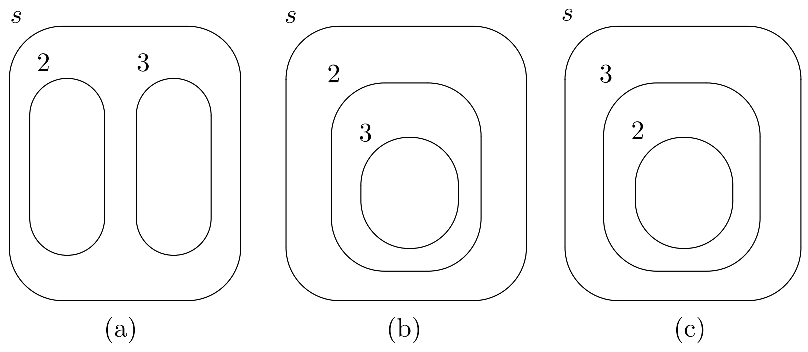
$$mREG = LTP_{m+2}^{set}(sym_2, anti_3, in).$$

We consider now the in-out case.

Theorem 2. *$LTP_*^{set}(sym_*, anti_*, in/out) \subsetneq REG$.*

Proof. We can use the same idea as in the previous theorem to prove the inclusion $LTP_*^{set}(sym_*, anti_*, in/out) \subseteq REG$. The properness of the inclusion is proved by the following counterexample.

Consider the language $L = \{a_2a_3, a_2a_2a_3\}$ and assume that there is a P system Π such that $trace(\Pi) = L$. Besides the skin membrane, this system must contain at least two membranes, with labels 2 and 3. These membranes can have one of the three relationships indicated in the next figure.



In case (a), in order to generate the string $a_2a_2a_3$ we must have the traveler outside membrane 2, and then a_2a_3 cannot be generated.

In case (b), in order to generate the string $a_2a_2a_3$ we must have the traveler in the region between membrane 2 and membrane 3, and then again a_2a_3 cannot be generated.

Similarly, in case (c), in order to generate the string $a_2a_2a_3$ we must have the traveler in the region between membrane 3 and membrane 2, and then a_2a_3 cannot be generated.

Thus, the equality $trace(\Pi) = L$ is not possible, this language is not in $LTP_*^{set}(sym_*, anti_*, in/out)$. $\square$

5 Decreasing the Number of Membranes

By the definition, in order to obtain a language over an alphabet with m symbols, we have to use a system with at least m membranes, hence the hierarchy on the number of membranes is trivially infinite in the case of trace languages. This direct dependence of the number of membranes on the number of symbols raises the question whether it is possible to keep bounded the number of membranes even when generating languages over arbitrary alphabets, or at least to use a number of membranes smaller than the number of symbols.

We present here three possible ways to address this question.

The first proposal is to consider several travelers. For instance, let us suppose that we have $T = \{t_1, t_2, \ldots, t_k\}$ a set of k travelers and that the membrane structure of our P system contains m membranes. In this case, the alphabet of trace symbols, contains $k \cdot m$ symbols $a_{i,j}$, where $1 \leq i \leq k$, and $1 \leq j \leq m$.

Let $\sigma = C_1C_2 \ldots C_k,\ k \geq 1$, be a halting computation and let $C_i(T) = \{a_{ij} \mid t_i$ is in membrane $j\}$. We consider $trace(T, \sigma) = \{w_1w_2 \ldots w_k \mid w_i \in V^*, \Psi_V(w_i) = \Psi_V(C_i(T))\}$, i.e., we concatenate permutations of strings representing each $C_i(T)$. Then, the trace language defined by a P system with several travelers is $LT(\Pi) = \{h(trace(T, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$. Thus, $LT(\Pi)$ is a language over an alphabet with $k \cdot m$ symbols, although we only use m membranes.

Let us consider an **example** – returning again to the case of P systems with multisets of objects. We take the P system

$$\Pi = (V, T, \mu, w_1, w_2, w_3, \{d\}, R_1, R_2, R_3),$$

with:

1. $V = \{d, t_1, t_2\}$ the set of all objects in the system; object d is present in arbitrarily many copies in the environment;
2. $T = \{t_1, t_2\}$ is the set of travelers;
3. $\mu = [\ [\ ]_2 [\ ]_3\ ]_1$ is a membrane structure with 3 membranes;
4. $w_1 = t_1t_2$, $w_2 = w_3 = \emptyset$ are the sets of objects in the initial configuration;
5. R_i is the set of symport rules associated with the membrane i, as follows:

- $R_1 = \{(t_1, out), (t_1 d, in), (t_2, out), (t_2 d, in)\}$,
- $R_2 = \{(t_1 d, in), (t_1, out), (t_2, in)\}$,
- $R_3 = \{(t_2 d, in), (t_2, out), (t_1, in)\}$.

The computation begins with travelers t_1 and t_2 in membrane labeled 1, and with the inner membranes (labeled 2, and 3) without any object. The initial configuration of the system is: $[\,t_1 t_2 [\ ]_2 [\ ]_3]_1$. According to the rules mentioned above, we can distinguish many cases in the evolution of the system. We discuss here only four:

- t_1 enters membrane labeled 3 and in the same time t_2 enters membrane labeled 2. In this case the computation halts, because membranes 3 and 2 act as trap membranes for travelers t_1 and t_2, respectively. The result is:
$$trace_1(\{t_1, t_2\}, \sigma) = \{a_{1,3}a_{2,2},\ a_{2,2}a_{1,3}\}.$$
- t_2 enters membrane 2 and is trapped, while traveler t_1 makes several (let us say n) journeys to the environment, and then to membrane 2 (bringing in objects d) until it enters membrane labeled 3, and the computation halts. We remind that we work with multisets of objects. For that, it is obvious that the number of trips traveler t_1 is paying to the environment and back to membrane 1 is as least as high as the number of trips it is paying to membrane 2. Now, we obtain:
$$trace_2(\{t_1, t_2\}, \sigma) = a_{2,2}a_{1,1}^{n}a_{1,2}^{m}a_{1,3},\ \text{for some } m \geq n.$$
- t_1 enters membrane 3 and is trapped while traveler t_2 behaves as traveler t_1 in the case above. The result of the computation is:
$$trace_3(\{t_1, t_2\}, \sigma) = a_{1,3}a_{2,3}^{s}a_{2,1}^{t}a_{2,2},\ \text{for some } t \leq s.$$
- Both t_1 and t_2 make different trips to membranes 2 and 3, respectively, provided that they have copies of d to do so. An easy-to-follow trace of the travelers, when both of them are moving, is the case when they get out membrane 1 and enter back (with a d), in an arbitrary number of steps, and then they go directly to their trap membranes. Thus,
$$trace_4(\{t_1, t_2\}, \sigma) = a_{1,1}^{p}a_{2,1}^{p}a_{1,3}a_{2,2}.$$

  The other cases (when, for example both t_1 and t_2 go out and back membrane 1 for an arbitrary number of steps and then, at a point traveler t_1 enters and exits membrane 2 for an arbitrary number of steps, and then decides to go again out of membrane 1, and so on) are already very difficult to follow, and no precise relationship between the trips of the two travelers can be given.

The second idea we propose in order to diminish the number of membranes is to consider an inverse morphism (for a morphism $h : V^* \to U^*$, the inverse morphism $h^{-1} : U^* \to 2^{V^*}$ is defined by $h^{-1}(y) = \{x \in V^* \mid h(x) = y\}$, $y \in U^*$). The idea is explained on the following example: take two alphabets, $V = \{a_1, a_2, \ldots, a_m\}$ and

$U = \{0, 1\}$ and the morphism h defined by $h(a_i) = 0^i1$, $1 \leq i \leq m$. It is obvious that this mapping is injective, hence $card(h^{-1}(y)) = 1$ for each $y \in h(V^*)$. Thus, for any language $L \subseteq V^*$ we have $L = h^{-1}(h(L))$. It it obvious (from the way we defined h) that choosing L from a family mFL we have $h(L) \in 2FL$.

Therefore, from the relation $mREG = LTP^{set}_{m+2}(sym_2, anti_3, in)$ mentioned in the previous section we obtain the following result – note that the use of an inverse morphism does not depend on the type of systems we deal with (with sets or with multisets of objects).

Proposition 1. *Every $L \in mREG$ can be written in the form $L = h^{-1}(L')$, for $L' \in 2LTP^{set}_4(sym_2, anti_3, in)$.*

The last proposal for obtaining more symbols in the trace languages than the number of membranes is to consider the possibility of changing the labels of membranes, as used in the area of P systems with active membranes. Changing the labels is not considered in standard symport/antiport rules, but we can introduce this feature in a simple way: symport rules of the form (x, in), (x, out) can be written as $x[\ ]_i \to [x]_i$, and $[x]_i \to [\ ]_i x$, respectively, while an antiport rule $(x, out; y, in)$ can be written as $y[x]_i \to [y]_i x$. Generalizing, we can consider that whenever an object enters or gets out of a membrane it can change its label. Thus, our rules will become: $x[\ ]_i \to [x]_j$, $[xri \to [\ ]_j x$, and $y[x]_i \to [y]_j x$, respectively.

In this way, we can have as many different labels as we want – with an important aspect to take care: not to have conflicts among the used rules, i.e., not to apply at the same time two rules which intend to change the label of the membrane in different ways. There are several possibilities for avoiding such conflicts: using the rules sequentially (only one in each membrane), using in parallel only a set of rules which change the label in the same way, allowing to only certain rules to change the labels and to use rules from this set in a restrictive way (e.g., sequentially), etc.

Although this last idea seems to be the most promising, we leave its examination to the reader.

6 Conclusions

We considered P systems with traces and sets of objects (instead of multisets) and we showed that the computational power of these systems with respect to Chomsky hierarchy does not go beyond the family of regular languages.

We also proposed three ideas to brake the infinite hierarchy provoked by the direct relationship between the number of membranes and the cardinality of the alphabet of languages.

Acknowledgments

The work of the first author was supported by National Natural Science Foundation of China (Grant No. 60373089).

The work of the second author was made possible thanks to the fellowship "Programa Nacional para la Formación del Profesorado Universitario" from the Spanish Ministry of Education, Culture and Sport.

References

1. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter: *Molecular Biology of the Cell.* 4th ed. Garland Science, New York, 2002.
2. A. Alhazov: P systems without multiplicities of symbol-objects. *Information Processing Letters*, accepted 2005.
3. A. Alhazov, R. Freund, Y. Rogozhin: Computational power of symport/antiport: history, advances and open problems. *Proceedings of Workshop on Membrane Computing, WMC6*, Vienna, Austria, July 2005, 44–78.
4. I.I. Ardelean: The relevance of cell membranes for P systems. General aspects. *Fundamenta Informaticae*, 49, 1-3 (2002), 35–43.
5. P. Frisco, H.J. Hoogeboom: Simulating counter automata by P systems with symport/antiport. In *Membrane Computing*, LNCS 2597, Springer, 2003, 288–301.
6. M. Ionescu, C. Martín-Vide, A. Păun, Gh. Păun: Unexpected universality results for three classes of P systems with symport/antiport. *Natural Computing*, 2 (2003), 337–348.
7. M. Ionescu, C. Martín-Vide, Gh. Păun: P systems with symport/antiport rules: The traces of objects. *Grammars*, 5, 2 (2002), 65–79.
8. A. Păun, Gh. Păun: The power of communication. P systems with symport/antiport. *New Generation Computers*, 20, 3 (2002), 295–306.
9. Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143, and *Turku Center for Computer Science-TUCS Report* No 208, 1998 (www.tucs.fi).
10. Gh. Păun: *Membrane Computing. An Introduction.* Springer-Verlag, Berlin, 2002.
11. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages*, 3 volumes. Springer-Verlag, Berlin, 1997.
12. A. Salomaa: *Formal Languages.* Academic Press, New York, 1973.