Tissue P Systems with Antiport Rules and Small Numbers of Symbols and Cells

Artiom Alhazov[1], Rudolf Freund[2], Marion Oswald[2]

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Str. Academiei 5, Chişinău, MD 2028, Moldova
E-mail: artiom@math.md
and
Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
E-mail: artiome.alhazov@estudiants.urv.es

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

Summary. We consider tissue P systems with antiport rules and investigate their computational power when using only a (very) small number of symbols and cells. Even when using only one symbol, any recursively enumerable set of natural numbers can be generated with at most seven cells. On the other hand, with only one cell we can only generate regular sets when using one channel with the environment, whereas one cell with two channels between the cell and the environment obtains computational completeness with at most five symbols. Between these extreme cases of one symbol and one cell, respectively, there is a trade-off between the number of cells and the number of symbols, e.g., for the case of tissue P systems with two channels between a cell and the environment we show that computational completeness can be obtained with two cells and three symbols as well as with three cells and two symbols, respectively.

1 Introduction

Membrane systems with a hierarchical (tree-like) structure (P systems) were introduced in the original paper of Gheorghe Păun (see [12]) with the rules being applied in a maximally parallel manner; tissue P systems with cells arranged in an arbitrary graph structure (see [9]) allow only one rule to be applied in each channel between two cells or a cell and the environment, but all channels work together in a maximally parallel manner. We here consider tissue P systems using symport/antiport rules (these communication rules first were investigated in [11]) for the communication between two cells or a cell and the environment.

It is well known that equipped with the maximally parallel derivation mode P systems with only one membrane (one cell) already reach computational completeness, even with antiport rules of weight two (e.g., see [4], [6]); the same result also holds true for P systems with one cell and two channels (working in opposite directions) between the cell and the environment, whereas with only one channel between the environment and the single cell only regular sets of natural numbers can be generated as will be proved in Section 5.

Considering the generation of recursively enumerable sets of natural numbers we may also ask the question how many symbols we need for obtaining computational completeness, especially for small numbers of membranes (in P systems) and cells (in tissue P systems), respectively. In [14], the quite surprising result was proved that three symbols in and four membranes are enough in the case of P systems with symport/antiport rules. The specific type of maximally parallel application of at most one rule in each channel between two cells or a cell and the environment, respectively, in tissue P systems allowed for an even more surprising result proved in [5]: The minimal number of one symbol is already sufficient to obtain computational completeness, e.g., it was shown that any recursively enumerable set of natural numbers can be generated by a tissue P system with at most seven cells using symport/antiport rules of only one symbol. On the other hand, for "classical" P systems using symport/antiport rules it was shown in [1] that computational completeness can already be obtained in one membrane by using only five symbols.

We here further investigate the power of tissue P systems with symport/antiport rules as well as small numbers of symbols and cells. After some preliminary definitions, we recall the definition of tissue P systems as they are considered in this paper. We then first consider tissue P systems with only one symbol and recall the completeness result from [5] in Section 3. Afterwards, in Section 4 we elaborate computational completeness results for tissue P systems with two and three symbols, respectively. On the other hand, for tissue P systems with only one cell, computational completeness cannot be achieved when using only one channel with the environment, instead with at least two symbols we obtain only regular sets as is shown in Section 5, whereas one cell with two channels between the cell and the environment gains computational completeness with at most five symbols (a similar result for P systems is established in [1]). In Section 6 we finally give an overview about the results obtained in this paper.

2 Preliminaries

For the basic elements of formal language theory needed in the following, we refer to [2] and [16]. We just list a few notions and notations: $\mathbf{N}$ denotes the set of positive integers. V^* is the free monoid generated by the alphabet V under the operation of concatenation and the empty string, denoted by λ, as unit element; by NRE, $NREG$, and $NFIN$ we denote the family of recursively enumerable sets,

regular sets, and finite sets of natural numbers, respectively; the family of sets of natural numbers representing the length sets of one-letter languages generated by matrix grammars is denoted by $NMAT$.

2.1 Register machines

The proofs of the main results established in this paper are based on the simulation of register machines; we refer to [10] for original definitions, and to [4] for definitions like those we use in this paper:

A *(non-deterministic) register machine* is a construct $M = (n, R, l_0, l_h)$, where n is the number of registers, R is a finite set of instructions injectively labelled with elements from a given set $lab(M)$, l_0 is the initial/start label, and l_h is the final label.

The instructions are of the following forms:

- $l_1 : (A(r), l_2, l_3)$
  Add 1 to the contents of register r and proceed to one of the instructions (labelled with) l_2 and l_3. (We say that we have an ADD instruction.)
- $l_1 : (S(r), l_2, l_3)$
  If register r is not empty, then subtract 1 from its contents and go to instruction l_2, otherwise proceed to instruction l_3. (We say that we have a SUB instruction.)
- $l_h : halt$
  Stop the machine. The final label l_h is only assigned to this instruction.

A register machine M is said to generate a vector $(s_1, \ldots, s_k)$ of natural numbers if, starting with the instruction with label l_0 and all registers containing the number 0, the machine stops (it reaches the instruction $l_h : halt$) with the first k registers containing the numbers $s_1, \ldots, s_k$ (and all other registers being empty).

Without loss of generality, in the succeeding proofs we will assume that in each ADD instruction $l_1 : (A(r), l_2, l_3)$ and in each SUB instruction $l_1 : (S(r), l_2, l_3)$ the labels l_1, l_2, l_3 are mutually distinct (for a short proof see [7]).

The register machines are known to be computationally complete, equal in power to (non-deterministic) Turing machines; especially we know that three registers are enough to generate any recursively enumerable set of natural numbers, where, moreover, the only instructions on the first register are ADD instructions (see [10], [4]).

2.2 Tissue P systems with symport/antiport rules

The reader is supposed to be familiar with basic elements of membrane computing, e.g., from [13]; comprehensive information can be found on the P systems web page `http://psystems.disco.unimib.it`.

Tissue P systems were introduced in [9], and tissue-like P systems with channel states were investigated in [7]. Here we deal with the following type of systems omitting the channel states:

A tissue P system (of degree $m \geq 1$) with symport/antiport rules is a construct

$$\Pi = \left(m, O, T, w_1, \ldots, w_m, ch, (R(i,j))_{(i,j)\in ch}\right),$$

where m is the number of cells, O is the alphabet of *objects*, $T \subseteq O$ is the alphabet of *terminal* objects, $w_1, \ldots, w_m$ are strings over O representing the *initial* multiset of *objects* present in the cells of the system (it is assumed that the m cells are labelled with $1, 2, \ldots, m$, and, moreover, we assume that all objects from O appear in an unbounded number in the environment), $ch \subseteq \{(i,j) \mid i,j \in \{0,1,2,\ldots,m\}, (i,j) \neq (0,0)\}$ is the set of links (*channels*) between cells (these were called *synapses* in [7]; 0 indicates the environment), $R(i,j)$ is a finite set of antiport rules of the form x/y, for some $x, y \in O^*$, associated with the channel $(i,j) \in ch$.

An *antiport rule* of the form $x/y \in R(i,j)$ for the ordered pair (i,j) of cells means moving the objects specified by x from cell i (from the environment, if $i = 0$) to cell j, at the same time moving the objects specified by y in the opposite direction. The rules with one of x, y being empty are, in fact, *symport rules*, but we do not always explicitly consider this distinction here, as it is not relevant for what follows. For short, we shall also speak of a *tissue P system* only when dealing with a *tissue P system with symport/antiport rules* as defined above.

The computation starts with the multisets specified by $w_1, \ldots, w_m$ in the m cells; in each time unit, a rule is used on each channel for which a rule can be used (if no rule is applicable for a channel, then no object passes over it). Therefore, the use of rules is sequential at the level of each channel, but it is parallel at the level of the system: all channels which can use a rule must do it (the system is synchronously evolving). The computation is successful if and only if it halts.

The result of a halting computation is the vector which describes the multiplicity of objects from T present in cell 1 in the halting configuration (the objects from $O - T$ are ignored when considering the result). The set of all k-dimensional vectors computed in this way by the system Π is denoted by $N(\Pi, k)$. The family of sets $N(\Pi, k)$ of vectors computed as above by systems with at most n symbols and m cells is denoted by $NO_ntP'_m(k)$. When any of the parameters k, m, n is not bounded, it is replaced by $*$.

In [7], only channels (i,j) with $i \neq j$ are allowed, and, moreover, for any i, j only one channel out of $\{(i,j), (j,i)\}$ is allowed, i.e., between two cells (or one cell and the environment) only one channel is allowed (as we shall see in Section 5, this technical detail may influence considerably the computational power). The family of sets $N(\Pi, k)$ of vectors computed as above by such tissue P systems with at most m cells is denoted by $NO_ntP_m(k)$.

In the following, we will only consider sets of natural numbers, i.e., the case $k = 1$. Hence, we will omit the parameter k, i.e., we will write $N(\Pi)$ as well as $NO_ntP'_m$ and NO_ntP_m only. Moreover, in the following we will not distinguish between

a language $L \subseteq \{a\}^*$ and the corresponding set of natural numbers $Ps(L) = \{k \mid a^k \in L\}$, the Parikh set of L.

3 Computational Completeness with One Symbol

In [5] it was shown that one symbol is enough for obtaining computational completeness when using at least seven cells:

Theorem 1. *$NRE = NO_1tP_n$ for all $n \geq 7$.*

Omitting the condition that for any i, j only one channel out of $\{(i,j),(j,i)\}$ is allowed, at least one cell can be saved (i.e., the one used as the trap, see [5]):

Theorem 2. *$NRE = NO_1tP'_n$ for all $n \geq 6$.*

4 Computational Completeness with Two and Three Symbols

As we are going to prove in this section, there is a trade-off between the number of cells and the number of symbols: as our main result, we show that in the case of allowing two channels between a cell and the environment (we can restrict ourselves to only one channel between cells) computational completeness can be obtained with two cells and three symbols as well as with three cells and two symbols, respectively. We first show that when allowing only two symbols we need at most three cells for obtaining computational completeness:

Theorem 3. *$NRE = NO_ntP'_m$ for all $n \geq 2$ and $m \geq 3$.*

Proof. We only prove $NRE \subseteq NO_2tP'_3$.

Let us consider a register machine $M = (3, R, l_0, l_h)$ with three registers generating $L \in NRE$. We now construct the tissue P system (of degree 3)

$$\begin{aligned} \Pi &= \left(3, \{a,b\}, \{a\}, w_1, \lambda, \lambda, ch, (R(i,j))_{(i,j)\in ch}\right), \\ ch &= \{(0,1),(0,2),(0,3),(1,0),(1,2),(2,0),(2,3), \\ &\quad (3,0),(3,1)\}, \end{aligned}$$

which simulates the actions of M in such a way that Π halts if and only if M halts, thereby representing the final contents of register 1 of M by the corresponding multisets of symbols in the first cell (and no other symbols contained there). Throughout the computation, cell i of Π represents the contents of register i by the corresponding number of symbols a, whereas specific numbers of the symbols b represent the instructions to be simulated; moreover, b also has the function of a trap symbol, i.e., in case of the wrong choice for a rule to be applied we take

in so many symbols b that we can never again get rid of them and therefore get "trapped" in an infinite loop.

An important part of the proof is to define a suitable encoding c for the instructions of the register machine. Without loss of generality we assume the labels of M to be positive integers such that the labels assigned to ADD and SUB instructions have the values $di+1$ for $0 \leq i < t$, as well as $l_0 = 0$ and $l_h = t-1$, for some $t \geq 1$ and some constant $d > 1$ which allows us to have d consecutive codes for each instruction. As we shall see, in this proof it suffices to take $d = 7$.

We now define the encoding c on non-negative integers in such a way that $c : \mathbf{N} \rightarrow \mathbf{N}$ is a linear function that has to obey to the following additional conditions:

- For any i, j with $1 \leq i, j < dt$, $c(i) + c(j) > c(dt)$, i.e., the sum of the codes of two instruction labels has to be larger than the largest code $c(dt)$ we will ever use for the given M.
- The distance g between any two codes $c(i)$ and $c(i+1)$ has to be large enough to allow one copy of the symbol b to be used for appearance checking as well as to allow specific numbers between 1 and g of copies of b to detect an incorrect application of rules.

  As we shall see in the construction of the rules below, we may take

  $$g = 2.$$

A function c fulfilling all the conditions stated above then, for example, is

$$c(x) = gx + gdt = 2x + 14t, \text{ for } x \geq 0.$$

With $l_0 = 0$ we therefore obtain

$$c(l_0) = 14t \text{ and } w_1 = b^{c(l_0)} = b^{14t}.$$

Finally, we have to find a number f which is so large that after the introduction of f symbols we inevitably enter an infinite loop with the rule b^{2f}/b^2; as we shall see below, we can take

$$f = 2c(dt).$$

Equipped with this coding function c and the constants defined above we are now able to define the following sets of symport/antiport rules for simulating the actions of the given register machine M:

$$\begin{aligned} R_{(0,1)} &= \left\{ b^{2f}/b^2 \right\}, \\ R_{(0,2)} &= \left\{ b^{2f}/b^2 \right\}, \\ R_{(0,3)} &= \left\{ b^{2f}/b^2 \right\}, \end{aligned}$$

$$
\begin{aligned}
R_{(1,0)} &= \left\{ b^{c(l_1)}/b^{c(l_2)}a, b^{c(l_1)}/b^{c(l_3)}a \mid l_1 : (A(1), l_2, l_3) \in R \right\} \\
&\cup \left\{ b^{c(l_1)}/b^{c(l_1+1)}a, b^{c(l_1+2)}/b^{c(l_2)}, b^{c(l_1+2)}/b^{c(l_3)} \mid \right. \\
&\quad \left. l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\} \right\} \\
&\cup \left\{ b^{c(l_1)}/b^{c(l_1+1)+1}, b^{c(l_1+2)}/b^{c(l_1+3)}, b^{c(l_1+4)}/b^{c(l_3)}, \right. \\
&\quad b^{c(l_1)}/b^{c(l_1+5)}, b^{c(l_1+6)}/b^{c(l_2)} \mid \\
&\quad \left. l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\} \right\} \\
&\cup \left\{ b^{c(l_h)}/\lambda \right\}, \\
R_{(1,2)} &= \left\{ b^{c(l_1+1)}a/\lambda \mid l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\} \right\} \\
&\cup \left\{ b^{c(l_1+1)+1}/\lambda, b^{c(l_1+3)}/\lambda, b^{c(l_1+5)}/\lambda \mid \right. \\
&\quad \left. l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\} \right\}, \\
R_{(2,0)} &= \left\{ ba/b^{2f} \right\}, \\
R_{(2,3)} &= \left\{ b^{c(l_1+1)}/\lambda \mid l_1 : (A(2), l_2, l_3) \in R \right\} \\
&\quad \left\{ b^{c(l_1+1)}a/\lambda \mid l_1 : (A(3), l_2, l_3) \in R \right\} \\
&\cup \left\{ b^{c(l_1+1)}/\lambda, b^{c(l_1+3)+1}/\lambda, b^{c(l_1+5)}a/\lambda \mid \right. \\
&\quad \left. l_1 : (S(2), l_2, l_3) \in R \right\} \\
&\cup \left\{ b^{c(l_1+1)+1}/\lambda, b^{c(l_1+3)}/\lambda, b^{c(l_1+5)}/\lambda \mid \right. \\
&\quad \left. l_1 : (S(3), l_2, l_3) \in R \right\}, \\
R_{(3,0)} &= \left\{ b^{c(l_1+1)}/b^{c(l_1+2)} \mid l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\} \right\} \\
&\cup \left\{ b^{c(l_1+1)}/b^{c(l_1+2)}, b^{c(l_1+3)+1}/b^{c(l_1+4)}, \right. \\
&\quad \left. b^{c(l_1+5)}a/b^{c(l_1+6)} \mid l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\} \right\} \\
&\cup \left\{ ba/b^{2f} \right\}, \\
R_{(3,1)} &= \left\{ b^{c(l_1+2)}/\lambda \mid l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\} \right\} \\
&\cup \left\{ b^{c(l_1+2)}/\lambda, b^{c(l_1+4)}/\lambda, b^{c(l_1+6)}/\lambda \mid \right. \\
&\quad \left. l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\} \right\}.
\end{aligned}
$$

The correct work of the rules in Π can be described as follows:

1. Throughout the whole computation in Π, the application of rules is directed by the code $b^{c(l)}$ for some $l \leq l_h$; in order to guarantee the correct sequence of encoded rules, superfluous symbols b in case of a wrong choice guarantee an infinite loop with the symbols b by the "trap rule"
   $$b^{2f}/b^2$$
   in the rule sets $R_{(0,i)}$, $i \in \{1,2,3\}$.
   The number $2f$ is so large that even in cell 1 which allows for the elimination of $c(l_h)$ symbols b enough symbols b remain to repeat the "trap rule" b^{2f}/b^2.
2. Each ADD instruction $l_1 : (A(1), l_2, l_3)$ of M is directly simulated by the rules
   $$b^{c(l_1)}/b^{c(l_2)}a,$$
   $$b^{c(l_1)}/b^{c(l_3)}a$$
   in $R_{(1,0)}$ in one step. The ADD instructions $l_1 : (A(r), l_2, l_3)$ of M, $r \in \{2,3\}$, are simulated in six steps in such a way that the new symbol a is transported to the corresponding cell r and, moreover, in cell 3 the code $c(l_1)$ is exchanged with the code $c(l_1+1)$ in order to guarantee that when returning to cell 1 a different code arrives which does not allow for misusing a symbol a representing the contents of register 1 in cell 1 to start a new cycle with the original

code. For example, the simulation of an ADD instruction $l_1 : (A(2), l_2, l_3)$ is accomplished by applying the following sequence of rules:

$b^{c(l_1)}/b^{c(l_1+1)}a$ from $R_{(1,0)}$,
$b^{c(l_1+1)}a/\lambda$ from $R_{(1,2)}$,
$b^{c(l_1+1)}/\lambda$ from $R_{(2,3)}$,
$b^{c(l_1+1)}/b^{c(l_1+2)}$ from $R_{(3,0)}$,
$b^{c(l_1+2)}/\lambda$ from $R_{(3,1)}$,
$b^{c(l_1+2)}/b^{c(l_2)}$, $b^{c(l_1+2)}/b^{c(l_3)}$ from $R_{(1,0)}$.

If we do not choose one of the correct rules, then an infinite loop will be entered by the rule b^{2f}/b^2 from the rule sets $R_{(0,i)}$, $i \in \{1, 2, 3\}$, as the coding function has been chosen in such a way that instead of the correct rule for a label l only rules for labels $l' < l$ could be chosen, whereas on the other hand, the number of symbols b is not large enough for allowing the remaining rest being interpreted as the code of another instruction label.

3. For simulating the decrementing step of a SUB instruction $l_1 : (S(r), l_2, l_3)$ from R we send the code $b^{c(l_1+5)}$ to the corresponding cell r, where a correct continuation is only possible if this cell contains at least one symbol a. For example, decrementing register 2 is accomplished by applying the following sequence of six rules:

$b^{c(l_1)}/b^{c(l_1+5)}$ from $R_{(1,0)}$,
$b^{c(l_1+5)}/\lambda$ from $R_{(1,2)}$,
$b^{c(l_1+5)}a/\lambda$ from $R_{(2,3)}$,
$b^{c(l_1+5)}a/b^{c(l_1+6)}$ from $R_{(3,0)}$,
$b^{c(l_1+6)}/\lambda$ from $R_{(3,1)}$,
$b^{c(l_1+6)}/b^{c(l_2)}$ from $R_{(1,0)}$.

Again we notice that if at some moment we do not choose the correct rule, then the application of the rule b^{2f}/b^2 will cause an infinite loop.

4. For simulating the zero test, i.e., the case where the contents of register r is zero, of a SUB instruction $l_1 : (S(r), l_2, l_3)$ from R we send the code $b^{c(l_1+1)}$ together with one additional copy of the symbol b to cell r, where this additional symbol b may cause the application of the rule ba/b^{2f} which the will lead to an infinite computation. In cell 3, the code $b^{c(l_1+1)}$ is exchanged with the code $b^{c(l_1+2)}$, which is exchanged with $b^{c(l_1+3)}$ in cell 1. This code $b^{c(l_1+3)}$ then captures the additional symbol b left back in cell r, and in cell 3 this additional symbol b goes out together with code $b^{c(l_1+3)}$, instead, code $b^{c(l_1+4)}$ continues and in cell 1 allows for replacing it with $b^{c(l_2)}$. For example, for testing register 3 for zero we take the following rules:

$b^{c(l_1)}/b^{c(l_1+1)+1}$ from $R_{(1,0)}$,
$b^{c(l_1+1)+1}/\lambda$ from $R_{(1,2)}$,
$b^{c(l_1+1)+1}/\lambda$ from $R_{(2,3)}$,
$b^{c(l_1+1)}/b^{c(l_1+2)}$ from $R_{(3,0)}$,
$b^{c(l_1+2)}/\lambda$ from $R_{(3,1)}$,

$b^{c(l_1+2)}/b^{c(l_1+3)}$ from $R_{(1,0)}$,
$b^{c(l_1+3)}/\lambda$ from $R_{(1,2)}$,
$b^{c(l_1+3)}/\lambda$ from $R_{(2,3)}$,
$b^{c(l_1+3)+1}/b^{c(l_1+4)}$ from $R_{(3,0)}$,
$b^{c(l_1+4)}/\lambda$ from $R_{(3,1)}$,
$b^{c(l_1+4)}/b^{c(l_3)}$ from $R_{(1,0)}$.

Once again we notice that if at some moment we do not choose the correct rule, then the application of the rule b^{2f}/b^2 will cause an infinite loop.

5. Finally, for the halt label l_h we take the rule
$b^{c(l_h)}/\lambda$,
hence, the work of Π will stop exactly when the work of M stops (provided the system has not become overflowed by symbols b due to a wrong non-deterministic choice during the computation).

From the explanations given above we conclude that Π halts if and only if M halts, and moreover, the final configuration of Π represents the final contents of the registers in M. These observations conclude the proof. □

Instead of the second channels $(0,i)$, $i \in \{1,2,3\}$, between the cells and the environment we could also use a fourth cell which then acts as a trap:

Corollary 1. *$NRE = NO_n tP_m$ for all $n \geq 2$ and $m \geq 4$.*

Proof. We only prove $NRE \subseteq NO_2 tP_4$.

Let us consider the tissue P system (of degree 4)

$$\Pi' = \left(4, \{a,b\}, \{a\}, w_1, \lambda, \lambda, \lambda, ch, (R(i,j))_{(i,j)\in ch}\right),$$
$$ch = \{(1,0),(1,2),(2,0),(2,3),(3,0),(3,1),$$
$$(4,0),(4,1),(4,2),(4,3)\},$$

where the rule sets $R(i,j)$ for

$$(i,j) \in \{(1,0),(1,2),(2,0),(2,3),(3,0),(3,1)\}$$

are exactly the same as in the proof of Theorem 3; the additional sets $R(4,j)$ for $j \in \{1,2,3\}$ contain the "trap rule" λ/b^2 which starts the trap represented by the rule b/b in $R(4,0)$. Except for the way of trapping the system, Π' works in the same way as the system Π constructed in the proof of Theorem 3, and this completes the proof. □

On the other hand, when the number of objects is increased to three, we need one cell less:

Theorem 4. *$NRE = NO_n tP'_m$ for all $n \geq 3$ and $m \geq 2$.*

Proof. We only prove $NRE \subseteq NO_3 tP'_2$.

As in the previous proof, we consider a register machine $M = (3, R, l_0, l_h)$ with three registers generating $L \in NRE$; we now construct the tissue P system (of degree 2)

$$\Pi = \left(2, \{a,b,c\}, \{a\}, w_1, \lambda, ch, (R(i,j))_{(i,j)\in ch}\right),$$
$$ch = \{(0,1),(0,2),(1,0),(1,2),(2,0)\},$$

which simulates the actions of M in such a way that throughout the computation, specific numbers of the symbols b represent the instructions to be simulated, the number of symbols a in cell 1 of Π represents the contents of register 1 by the corresponding number of symbols a and the new symbol c represents the contents of registers 2 and 3 by the corresponding number of symbols c in cells 1 and 2, respectively.

We shall use the same encoding c as in the previous proofs. The zero test now uses the rule bc/b^{2f} in $R_{(1,0)}$ or $R_{(2,0)}$, respectively.

In sum, we define the following sets of symport/antiport rules for simulating the actions of the given register machine M:

$$\begin{aligned}
R_{(0,1)} &= \left\{b^{2f}/b^2\right\},\\
R_{(0,2)} &= \left\{b^{2f}/b^2\right\},\\
R_{(1,0)} &= \left\{b^{c(l_1)}/b^{c(l_2)}a, b^{c(l_1)}/b^{c(l_3)}a \mid l_1 : (A(1), l_2, l_3) \in R\right\}\\
&\cup \left\{b^{c(l_1)}/b^{c(l_2)}c, b^{c(l_1)}/b^{c(l_3)}c \mid l_1 : (A(2), l_2, l_3) \in R\right\}\\
&\cup \left\{b^{c(l_1)}/b^{c(l_1+1)}c, b^{c(l_1+2)}/b^{c(l_2)}, b^{c(l_1+2)}/b^{c(l_3)} \mid\right.\\
&\quad \left. l_1 : (A(3), l_2, l_3) \in R\right\}\\
&\cup \left\{b^{c(l_1)}/b^{c(l_1+1)+1}, b^{c(l_1+2)+1}/b^{c(l_3)}, b^{c(l_1)}c/b^{c(l_2)} \mid\right.\\
&\quad \left. l_1 : (S(2), l_2, l_3) \in R\right\}\\
&\cup \left\{b^{c(l_1)}/b^{c(l_1+1)+1}, b^{c(l_1+2)}/b^{c(l_1+3)}, b^{c(l_1+4)}/b^{c(l_3)},\right.\\
&\quad b^{c(l_1)}/b^{c(l_1+5)}, b^{c(l_1+6)}/b^{c(l_2)} \mid\\
&\quad \left. l_1 : (S(3), l_2, l_3) \in R\right\}\\
&\cup \left\{b^{c(l_h)}/\lambda, bc/b^{2f}\right\},\\
R_{(1,2)} &= \left\{b^{c(l_1+1)}c/\lambda, \lambda/b^{c(l_1+2)} \mid l_1 : (A(3), l_2, l_3) \in R\right\}\\
&\cup \left\{b^{c(l_1+1)}/\lambda, \lambda/b^{c(l_1+2)} \mid l_1 : (S(2), l_2, l_3) \in R\right\}\\
&\cup \left\{b^{c(l_1+1)+1}/\lambda, \lambda/b^{c(l_1+2)}/\lambda, b^{c(l_1+3)}/\lambda,\right.\\
&\quad \lambda/b^{c(l_1+4)}, b^{c(l_1+5)}/\lambda, \lambda/b^{c(l_1+6)} \mid\\
&\quad \left. l_1 : (S(3), l_2, l_3) \in R\right\}\\
&\cup \left\{b^4/\lambda, \lambda/b^2\right\},\\
R_{(2,0)} &= \left\{b^{c(l_1+1)}/b^{c(l_1+2)} \mid l_1 : (A(3), l_2, l_3) \in R\right\}\\
&\cup \left\{b^{c(l_1+1)}/b^{c(l_1+2)} \mid l_1 : (S(2), l_2, l_3) \in R\right\}\\
&\cup \left\{b^{c(l_1+1)}/b^{c(l_1+2)}, b^{c(l_1+3)+1}/b^{c(l_1+4)},\right.\\
&\quad \left. b^{c(l_1+5)}c/b^{c(l_1+6)} \mid l_1 : (S(3), l_2, l_3) \in R\right\}\\
&\cup \left\{bc/b^{2f}\right\}.
\end{aligned}$$

As one can easily see, Π halts if and only if M halts, and moreover, in the final configuration of Π cell 1 represents the final contents of register 1 in M. If at some moment we do not use the correct rule, then an infinite loop will be entered by applying the rule b^{2f}/b^2 from the rule sets $R_{(0,i)}$, $i \in \{1,2\}$. These observations conclude the proof. □

Corollary 2. *$NRE = NO_nP_m$ for all $n \geq 3$ and $m \geq 3$.*

Proof. We only prove $NRE \subseteq NO_3tP_3$. The result follows from Theorem 4 as the proof of Corollary 1 followed from the proof of Theorem 3:

Let us consider the tissue P system (of degree 3)

$$\Pi' = \left(3, \{a, b, c\}, \{a\}, w_1, \lambda, \lambda, ch, (R(i,j))_{(i,j)\in ch}\right),$$
$$ch = \{(1,0), (1,2), (2,0), (3,0), (3,1), (3,2)\},$$

where the rule sets $R(i,j)$ for $(i,j) \in \{(1,0), (1,2), (2,0)\}$ are exactly the same as in the proof of Theorem 4; the additional sets $R(3,j)$ for $j \in \{1,2\}$ contain the "trap rule" λ/b^2 which starts the trap represented by the rule b/b in $R(3,0)$. Except for the way of trapping the system, Π' works in the same way as the system Π constructed in the proof of Theorem 4, and this completes the proof. □

5 Tissue P Systems with One Cell

In this section we investigate the remaining case of using only one cell, in which case it turns out that the definition of the tissue P system is essential, i.e., computational completeness can only be obtained with two channels between the cell and the environment, whereas we can only generate regular sets when using only one channel between the cell and the environment.

The proof of the following completeness result can be obtained following the construction given in [1] for P systems and therefore is omitted:

Theorem 5. *$NRE = NO_ntP_1'$ for all $n \geq 5$.*

We now consider the case of tissue P systems with only one channel between the cell and the environment. In the simplest case of only one symbol, we only get finite sets:

Example 1. To each finite one-letter language L we can construct the tissue P system

$$\Pi = (1, \{a\}, \{a\}, w_1, \{(1,0)\}, R(1,0)), \text{ where}$$
$$w_1 = a^m, \text{ with } m = \max\{i \mid a^i \in L\}, \text{ and}$$
$$R(1,0) = \{a^m/a^j \mid a^j \in L, j < m\}.$$

Obviously, $Ps(L) = N(\Pi)$.

Theorem 6. *$NFIN = NO_1tP_1 = NO_1tP_1'$.*

Proof. The inclusion $NFIN \subseteq NO_1tP_1$ directly follows from Example 1. The inclusion $NO_1tP_1 \subseteq NO_1tP_1'$ directly follows from the definitions.

The inclusion $NFIN \supseteq NO_1tP_1'$ can easily be argued as follows: Consider a tissue P system

$$\Pi = (1, \{a\}, \{a\}, w_1, \{(1,0), (0,1)\}, R(1,0), R(0,1));$$

then $N(\Pi) \subseteq \{a^j \mid j < m\}$ (hence, $N(\Pi) \in NFIN$), where $m = \min\{i \mid a^i/a^k \in R(1,0) \text{ or } a^k/a^i \in R(0,1)\}$, as to any multiset a^j with $j \geq m$ in the single cell still a rule from $R(1,0) \cup R(0,1)$ can be applied. □

For describing the sets generated by tissue P systems with one cell and more than one symbol, we need the notion of a minimal deterministic finite automaton:

A *deterministic finite automaton* (*DFA* for short) is a quintuple $M = (Q, T, \delta, q_0, F)$, where Q is the finite set of *states,* T is the *input alphabet*, $\delta : Q \times T \to Q$ is the *state transition function*, $q_0 \in Q$ is the *initial state*, and $F \subseteq Q$ is the set of *final states*. The transition function δ can be extended in a natural way to a function $\delta : Q \times T^* \to Q$. The language accepted by the DFA M is the set of all strings $w \in T^*$ that are accepted by M in such a way that $\delta(q_0, w) \in F$. As is well known, for each regular language L there exists a minimal DFA accepting L. For regular languages over a one-letter alphabet, the minimal DFA accepting an infinite language is of the form $(\{q_i \mid 0 \leq i \leq n\}, \{a\}, \delta, q_0, F)$ with δ being of the form $\delta(q_i, a) = q_{i+1}$ for $0 \leq i < n$ and $\delta(q_n, a) = q_j$ for some j with $0 \leq j \leq n$. For any finite language L over a one-letter alphabet, the minimal DFA accepting L obeys to the condition $\delta(q_i, a) = q_{i+1}$ for $0 \leq i < n$, $\delta(q_n, a) = q_n$, and $q_n \notin F$.

If we allow at least two symbols, then all regular sets can be obtained:

Example 2. To each infinite regular language $L \in REG$ accepted by the minimal DFA $(\{q_i \mid 0 \leq i \leq n\}, \{a\}, \delta, q_0, F)$ with $\delta(q_i, a) = q_{i+1}$ for $0 \leq i < n$ and $\delta(q_n, a) = q_j$ for some j with $0 \leq j \leq n$ we can construct the tissue P system

$$\Pi = (1, \{a, b\}, \{a\}, bb, \{(1,0)\}, R(1,0)), \text{ where}$$
$$R(1,0) = \{bb/ba, ba/ba^{n-j+2}\} \cup \{bb/a^i \mid q_i \in F, i < j\}$$
$$\cup \{ba/a^i \mid q_i \in F, i \geq j\}.$$

In the "state" bb we can generate all elements a^i for $q_i \in F$ with $i < j$ or else we directly change to the "state" ba, which allows us to repeat the loop between q_j and q_n arbitrarily often before we finish with introducing a^i for $q_i \in F$ with $i \geq j$ at the same time also eliminating the single symbol b which definitely terminates the derivation in Π. Obviously, $N(\Pi) = Ps(L)$.

The following result shows that with only one cell and one channel between the single cell and the environment only regular sets can be generated:

Theorem 7. *$NREG = NO_ntP_1$ for all $n \geq 2$.*

Proof. As shown by the constructions given in Examples 1 and 2, $NREG \subseteq NO_ntP_1$ for all $n \geq 2$. Hence, it remains to show the opposite inclusion $NREG \supseteq NO_ntP_1$.

As is well known from [8], $NREG = NMAT$, hence, it remains to show that $NO_ntP_1 \subseteq NMAT$ which can be done in a similar way as in the proof of Lemma 2 in [3]:

Consider the tissue P system

$$\Pi = (1, O, \{a\}, w_1, \{(1,0)\}, R(1,0)).$$

A set M representing the reachable configurations, i.e., the possible multisets contained in the single cell during any computation in the tissue P system can easily be generated by a matrix grammar without appearance checking. Moreover, the set K representing the multisets to which a rule from $R(1,0)$ can be applied is regular, hence, the complement K^c of K is regular, too. Therefore, the Parikh set of the projection of $M \cap K^c$ onto $\{a\}^*$ is in $NMAT$ due to the closure properties of the family of languages generated by matrix grammars without appearance checking. We leave the details of the proof to the reader. □

6 Conclusion

In sum, for tissue P systems with only one channel between two cells and between a cell and the environment we could show the results listed in Table 1 (we have omitted the proof of the completeness result $NRE = NO_n tP_m$ for all $n \geq 4$ and $m \geq 2$, which needs a different proof technique following the construction given in [1] for P systems):

symbols							
4	$NREG$	NRE	NRE	NRE	NRE	NRE	NRE
3	$NREG$	?	NRE	NRE	NRE	NRE	NRE
2	$NREG$	?	?	NRE	NRE	NRE	NRE
1	$NFIN$	?	?	?	?	?	NRE
	1	2	3	4	5	6	7 cells

Table 1. Families $NO_m tP_n$

The main open question concerns a characterization of the sets of natural numbers in NO_2tP_2, NO_2tP_3, and NO_3tP_2. Further, it would be interesting to find the minimal number l such that NO_1tP_l contains all recursively enumerable sets of natural numbers, whereas the families NO_1tP_j with $j < l$ do not fulfill this condition. Finally, it remains to find characterizations of the sets of natural numbers in those families NO_1tP_j that do not contain all recursively enumerable sets of natural numbers.

The most interesting open problems for the families $NO_n tP'_m$ are to find the minimal number k as well as the minimal number l such that $NO_k tP'_1$ and $NO_1 tP'_l$, respectively, contain all recursively enumerable sets of natural numbers, whereas the families $NO_i tP'_1$ and $NO_1 tP'_j$ with $i < k$ and $j < l$, respectively, do not fulfill this condition. Moreover, it remains to find characterizations of the sets of natural numbers in those families $NO_n tP'_m$ that do not contain all recursively enumerable sets of natural numbers.

Related open problems concern the families $NO_m P_n$ of sets of natural numbers generated by P systems with symport/antiport rules as well as n symbols and m

symbols						
5	NRE	NRE	NRE	NRE	NRE	NRE
4	?	NRE	NRE	NRE	NRE	NRE
3	?	NRE	NRE	NRE	NRE	NRE
2	?	?	NRE	NRE	NRE	NRE
1	$NFIN$	?	?	?	?	NRE
	1	2	3	4	5	6 cells

Table 2. Families $NO_mtP'_n$

membranes. The first result proving computational completeness for P systems with three symbols and four membranes was obtained in [14] and continued in [1], where P systems with five symbols in only one membrane were shown to be computationally complete. The main open problem in the case of P systems is the question whether one symbol is sufficient to obtain computational completeness as was shown for the case of tissue P systems in [5].

Acknowledgement

The work of Marion Oswald was supported by FWF-project T225-N04.

References

1. A. Alhazov, R. Freund: P systems with one membrane and symport/antiport rules of five symbols are computationally complete. In *Proceedings of the Third Brainstorming Week on Membrane Computing* (M.A. Gutiérrez, A. Riscos, F.J. Romero, D. Sburlan, eds.), Report RGNC 01/05, University of Seville, 2005, in press.
2. J. Dassow, Gh. Păun: *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.
3. R. Freund, A. Leporati, M. Oswald, C. Zandron: Sequential P systems with unit rules and energy assigned to membranes. In *Machines, Computations and Universality, MCU'2004* (M. Margenstern, ed.), LNCS 3354 (2005), 200-210.
4. R. Freund, M. Oswald: P Systems with activated/prohibited membrane channels. In *Membrane Computing. International Workshop WMC 2002, Curtea de Argeş, Romania, Revised Papers* (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.), LNCS 2597 (2003), 261–268.
5. R. Freund, M. Oswald: Tissue P systems with symport/antiport rules of one symbol are computationally complete. Downloadable from [17].
6. R. Freund, A. Păun: Membrane systems wih symport/antiport rules: universality results. In *Membrane Computing. International Workshop WMC 2002, Curtea de Argeş, Romania, Revised Papers* (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.), LNCS 2597 (2003), 270–287.
7. R. Freund, Gh. Păun, M.J. Pérez-Jiménez: Tissue-like P systems with channel states. In *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos, A. Romero, F. Sancho, eds.), Report RGNC 01/04, University of Seville, 2004, 206–223, and *Theoretical Computer Science*, 330 (2005), 101–116.

8. D. Hauschildt, M. Jantzen: Petri net algorithms in the theory of matrix grammars. *Acta Informatica*, 31 (1994), 719–728.
9. C. Martín-Vide, J. Pazos, Gh. Păun, A. Rodriguez-Paton: Tissue P systems. *Theoretical Computer Science*, 296, 2 (2003), 295–326.
10. M.L. Minsky: *Computation: Finite and Infinite Machines*, Prentice Hall, Englewood Cliffs, New Jersey, 1967.
11. A. Păun, Gh. Păun: The power of communication: P systems with symport/antiport. *New Generation Computing*, 20, 3 (2002), 295–306.
12. Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000) 108–143, and TUCS Research Report, 208 (1998) (`http://www.tucs.fi`)
13. Gh. Păun: *Computing with Membranes: An Introduction.* Springer-Verlag, Berlin, 2002.
14. Gh. Păun, J. Pazos, M.J. Pérez-Jiménez, A. Rodríguez-Patón: Symport/antiport P systems with three objects are universal. *Fundamenta Informaticae*, 64, 1-4 (2005), 345–358.
15. Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron (eds.): *Membrane Computing. International Workshop WMC 2002, Curtea de Argeş, Romania, Revised Papers*, LNCS 2597, Springer-Verlag, Berlin, 2003.
16. G. Rozenberg, A. Salomaa (eds.): *Handbook of Formal Languages* (3 volumes), Springer-Verlag, Berlin, 1997.
17. The P Systems Web Page: `http://psystems.disco.unimib.it`