Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. VII (2012), No. 1 (March), pp. 173-183

Small Universal Tissue P Systems with Symport/Antiport Rules

X. Zhang, B. Luo, L. Pan

Xingyi Zhang, Bin Luo
Key Lab of Intelligent Computing and
Signal Processing of Ministry of Education
School of Computer Science and Technology
Anhui University, Hefei 230039, China
E-mail: xyzhanghust@gmail.com,luobinahu@yahoo.com.cn

Linqiang Pan
Key Laboratory of Image Processing and Intelligent Control
Department of Control Science and Engineering
Huazhong University of Science and Technology
Wuhan 430074, China
E-mail: lqpan@mail.hust.edu.cn (corresponding author)

Abstract:
In this note, we consider the problem of looking for small universal one-symbol tissue P systems with symport/antiport rules. It is proved that six cells suffice to generate any recursively enumerable set of natural numbers by such a one-symbol tissue P system with symport/antiport rules, under the restriction that only one channel is allowed between two cells or between a cell and the environment. As for the case of allowing two channels between a cell and the environment, it is shown that the computational completeness can be obtained by one-symbol tissue P systems with symport/antiport rules having at most five cells. These results partially answer an open problem formulated by Artiom Alhazov, Rudolf Freund and Marion Oswald.
Keywords: membrane computing, tissue P system, symport/antiport rule, universality.

1 Introduction

Membrane computing is one of the recent branches of natural computing, which was initiated by Gh. Păun in 1998 [7]. The aim of membrane computing is to abstract novel computing ideas or models from the structure and the functioning of a living cell, as well as from the organization of cells in tissues, organs, and other higher order structures. The obtained models, called *P systems*, provide distributed parallel and non-deterministic computing models. The field of membrane computing has rapidly developed (already in 2003, ISI considered membrane computing as a "fast emerging research area in computer science", see http://esi-topics.com). Please refer to the handbook of membrane computing [9] for general information in this area, and to the membrane computing web site [11] for the up-to-date information.

Tissue P systems form a class of P systems, which were introduced in [6]. Tissue P systems were inspired by intercellular communication and cooperation between cells. Briefly, a tissue P system consists of a set of membranes (abstracted from cells) placed in the nodes of a graph. The net of membranes deals with symbols and communicates symbols along channels specified in advance. The communication among cells is based on symport/antiport rules [8]. Symport rules move objects across a membrane in one direction, whereas antiport rules move objects across a membrane in opposite directions. Between two cells or between a cell and the environment, it is

possible that there exists only one channel or more than one channel [2]. A tissue P system works in a synchronized mode (a global clock is assumed, marking the time for the whole system). In each time unit, if there are rules that can be applied in each channel, then one of the rules must be applied. At most one rule is applied in each channel, non-deterministically chosen among the rules that can be applied in the channel. So, the use of rules is sequential at the level of each channel, but it is parallel at the level of the system. In this note, we shall consider the following two restrictive versions of tissue P systems: (1) only one channel is allowed between two cells or between a cell and the environment, (2) two channels are allowed between a cell and the environment.

Table 1: The computational completeness results for tissue P systems with only one channel between two cells, or between a cell and the environment; where ? indicates an open problem, † indicates that the result is obtained in this note, the other results are from [1].

symbols/cells	1	2	3	4	5	6	7
1	NREG	?	?	?	?	NRE†	NRE
2	NREG	?	?	NRE	NRE	NRE	NRE
3	NREG	?	NRE	NRE	NRE	NRE	NRE
4	NREG	NRE	NRE	NRE	NRE	NRE	NRE

One of the central problems about tissue P systems is to investigate their computational power. This topic has been widely investigated for tissue P systems of various forms (e.g., see [4, 6, 10]). For the special case when tissue P systems use only a (very) small number of symbols and cells, please refer to [1, 3]. There seems to be a trade-off between the number of cells and the number of symbols needed for the computational power of tissue P systems. If only one channel is allowed between two cells or between a cell and the environment, it was shown that any recursively enumerable set of natural numbers can be generated by a tissue P system with at most seven cells and only one symbol [3]. The number of cells can decrease when two symbols are used; specifically, a tissue P system with two symbols and at most four cells can generate any recursively enumerable set of natural numbers [3]. The known results about the computational power of tissue P systems with only one channel between two cells or between a cell and the environment are listed in Table 1. If two channels are allowed between a cell and the environment, then one cell is enough to obtain computational completeness for tissue P systems with at most five symbols [1]. The number of symbols for computational completeness can decrease when the number of cells increases: for the case of tissue P systems with two channels between a cell and the environment, the computational completeness can be obtained by tissue P systems with two cells and three symbols, or three cells and two symbols. The known results about the computational power of tissue P systems with two channels between a cell and the environment are listed in Table 2.

Table 2: The computational completeness results for tissue P systems with two channels between a cell and the environment; where ? indicates an open problem, † indicates that the result is obtained in this note, the other results are from [1].

symbols/cells	1	2	3	4	5	6
1	NFIN	?	?	?	NRE†	NRE
2	?	?	NRE	NRE	NRE	NRE
3	?	NRE	NRE	NRE	NRE	NRE
4	?	NRE	NRE	NRE	NRE	NRE
5	NRE	NRE	NRE	NRE	NRE	NRE

In this note, we continue to investigate the computational power of tissue P systems with small numbers of symbols and cells. Specifically, we look for universal one-symbol tissue P systems with symport/antiport rules having a small number of cells. It is proved that six cells suffice to generate any recursively enumerable set of natural numbers for such a one-symbol tissue P system, under the restriction that only one channel is allowed between two cells or between a cell and the environment. The idea used in the proof of the result can also be extended to the case of allowing two channels between a cell and the environment, then one cell can be saved. The results obtained in this note partially answer open problems formulated in [1] (see Tables 1 and 2).

This note is organized as follows. In Section 2, the formal language theory preliminaries are recalled, including the formal definition of register machine. The formal definition of tissue P systems is introduced in Section 3. Two universal one-symbol tissue P systems are presented in Section 4, with an overview of the computation. Conclusions and comments are presented in Section 5.

2 Formal language theory preliminaries

For an alphabet V, V^* denotes the set of all finite strings over V, with the empty string denoted by λ. The set of all nonempty strings over V is denoted by V^+.

A *register machine* is a construct $M = (m, H, l_0, l_h, I)$, where m is the number of registers (each holds a natural number), H is the set of instruction labels, l_0 is the start label (labeling an ADD instruction), l_h is the halt label (assigned to instruction HALT), and I is the set of instructions. Each label from H labels only one instruction from I, thus precisely identifying it. The instructions are of the following forms:

- $l_i : (\texttt{ADD}(r), l_j, l_k)$ (add 1 to register r and then go to one of the instructions with labels l_j, l_k),
- $l_i : (\texttt{SUB}(r), l_j, l_k)$ (if register r is non-zero, then subtract 1 from it, and go to the instruction with label l_j; otherwise, go to the instruction with label l_k),
- $l_h : \texttt{HALT}$ (the halt instruction).

A register machine M computes (generates) a number n in the following way. The register machine starts with all registers empty (i.e., storing the number zero). It applies the instruction with label l_0 and proceeds to apply instructions as indicated by labels (and, in the case of SUB instructions, by the content of registers). If the register machine reaches the halt instruction, then the number n stored at that time in the first register is said to be computed by M. It is known that register machines compute all sets of numbers which are Turing computable, hence they characterize NRE [5] (NRE is the family of length sets of recursively enumerable languages; that is, those recognized by Turing machines). Especially, it is known that three registers are enough to generate any recursively enumerable set of natural numbers.

Without loss of generality, it can be assumed that l_0 labels an ADD instruction and that in the halting configuration all registers different from the first one are empty, and that the output register is never decremented during the computation (its content is only added to).

Without loss of generality, it can be assumed that in each ADD instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$ and in each SUB instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$ the labels l_i, l_j, l_k are mutually distinct (for a short proof, see [2]).

3 Tissue P systems with symport/antiport rules

Tissue P systems were introduced in [6], and tissue-like P systems with channel states were introduced in [2]. In this note, the following type of systems is considered, omitting the channel states.

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

$$\Pi = (O, T, E, w_1, \ldots, w_m, ch, (R(i,j))_{(i,j)\in ch}), \text{ where:}$$

- O is the alphabet of *objects*;
- $T \subseteq O$ is the alphabet of *terminal* objects;
- $E \subseteq O$ is the set of objects present in the environment in arbitrarily copies each;
- $w_1, \ldots, w_m$ are strings over O, representing the multisets of objects placed in the cells of the system at the beginning of the computation (it is assumed that the system contains m cells, labelled with $1, 2, \ldots, m$);
- $ch \subseteq \{(i,j) \mid i,j \in \{0,1,2,\ldots,m\}, (i,j) \neq (0,0)\}$ is the set of links (*channels*) between cells (they were also called *synapses* in [2]; 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, and at the same time moving the objects specified by y from cell j to cell i. 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.

Note that the objects from E are never exhausted, irrespective of how many copies of each of them are brought into the system, arbitrarily many copies remain available in the environment.

A *configuration* of a tissue P system is described by the multisets of objects over O associated with the cells of the system. The tuple $(w_1, w_2, \ldots, w_m)$ is the *initial* configuration. A *halting* configuration is a configuration such that there is no rule that can be applied. The *computation* starts from the initial configuration; 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 (reaching a configuration where no rule can be applied). The result of a halting computation is encoded by the multiset of objects over T appearing in a cell specified in advance in the halting configuration.

In this note, we deal with a restricted version of the systems introduced above. Only channels (i,j) with $i \neq j$ are allowed. If only one channel is allowed between two cells or between a cell and the environment, then for any i, j only one of (i,j) and (j,i) is allowed. If two channels are allowed between two cells or between a cell and the environment, then both (i,j) and (j,i) are allowed. Furthermore, only one-symbol tissue P systems are considered, hence we assume $O = T = E = \{a\}$. Then, for simplicity, a one-symbol tissue P system is written as $\Pi = (w_1, \ldots, w_m, ch, (R(i,j))_{(i,j)\in ch})$; the output cell is that with label 1. We also write x/y for an antiport rule a^x/a^y and x for multiset a^x.

4 Universality results

In [3], it was shown that one-symbol tissue P systems with symport/antiport rules having seven cells are Turing universal when only one channel is allowed between two cells or between a cell and the environment. This result is improved by showing that six cells are enough for Turing universality.

Theorem 1. *For any recursively enumerable set L of natural numbers, a one-symbol tissue P system with symport/antiport rules having at most 6 cells can be constructed to generate L, under the restriction that only one channel is allowed between two cells or between a cell and the environment.*

Proof: Let us consider a register machine $M = (m, H, l_0, l_h, I)$. As stated in Section 2, a register machine with three registers can generate any recursively enumerable set L of natural numbers, where the instructions that act on the first register are ADD instructions. This means that $m = 3$. In what follows, we shall construct a tissue P system Π with only one symbol a to simulate the register machine M.

The system Π consists of six cells labeled by $1, \ldots, 6$. The cell with label 1 represents register 1 (the output register); the cells with labels 2 and 3 represent registers 2 and 3, respectively; the cells with labels 4 and 5 are program cells (the cell with label 4 controls the simulation of the instructions that act on registers 1 and 2, the cell with label 5 controls the simulation of the instructions that act on register 3); the cell with label 6 is used as a trap that, whenever started, leads to a non-halting computation by using the antiport rule $2/2$ in channel $(6, 0)$. The number stored in each register r ($r = 1, 2, 3$) is represented by the number of copies of symbol a in the cell with label r in the following way. If register 1 contains number n ($n \geq 0$), then the cell with label 1 has $n + 5$ copies of symbol a; if register 2 (resp. 3) contains number n ($n \geq 0$), then the cell with label 2 (resp. 3) has $2n + 2$ copies of symbol a.

Without loss of generality, we assume that $l_i = 2i + 1$, $0 \leq i \leq t$, $t \geq 0$, are the labels of instructions of register machine M, and l_t is the label of halting instruction HALT. We define the function c:

$$c(0) = 12, \quad c(i+1) = \sum_{j=0}^{i} c(j) + c(0), \text{ for } i \geq 0.$$

It is easy to check that the function c has the following properties.

- For any $i < k, j < k$, $i \neq j$, we have $c(k) > c(i) + c(j) + 12$.
- For any i, $0 \leq i \leq t$, we have $c(l_i) > 12$.
- For any $i \neq j$, $0 \leq i, j \leq t$, we have $c(l_i) \neq c(l_j)$.
- For any $i < j$, $0 \leq i, j \leq t$, the value $c(l_i + 1)$ is between $c(l_i)$ and $c(l_j)$.

In the initial configuration of Π, the cell with label 1 contains five copies of symbol a, and the cells with labels 2 and 3 contain two copies of symbol a, which represents the fact that registers 1, 2 and 3 initially have number 0; the cell with label 4 contains $c(l_0)$ copies of symbol a; the cells with labels 5 and 6 contain 0 copy of symbol a. In general, when the cell with label 4 contains $c(l_i)$ copies of symbol a, system Π starts to simulate the instruction l_i.

Formally, the tissue P system Π is a construct of the form

$$\Pi = (5, 2, 2, c(l_0), 0, 0, ch, (R(i,j))_{(i,j)\in ch}), \text{ where:}$$

- $ch = \{(4, i) \mid i \in \{0, 1, 2, 5, 6\}\} \cup \{(5, 0), (5, 3), (5, 6), (6, 0)\}$;

- the sets of rules $R(i,j)$, $(i,j) \in ch$, are as follows:

$$\begin{aligned}
R(4,0) &= \{(c(l_i)-6)/(c(l_j)-5), (c(l_i)-6)/(c(l_k)-5) \mid l_i : (\texttt{ADD}(1), l_j, l_k) \in R\} \\
&\cup \{(c(l_i)-3)/(c(l_j)-1), (c(l_i)-3)/(c(l_k)-1) \mid l_i : (\texttt{ADD}(2), l_j, l_k) \in R\} \\
&\cup \{(c(l_i)-1)/(c(l_i+1)-3), (c(l_i+1))/(c(l_j)), \\
&\quad (c(l_i+1)-2)/c(l_k) \mid l_i : (\texttt{SUB}(2), l_j, l_k) \in R\}, \\
R(4,1) &= \{6/5\}, R(4,2) = \{3/1, 1/3\}, R(4,6) = \{2/0\}, R(5,3) = \{3/1, 1/3\}, \\
R(4,5) &= \{c(l_i)/0, 0/c(l_j), 0/c(l_k) \mid l_i : (\texttt{ADD}(3), l_j, l_k)\} \\
&\cup \{c(l_i)/0, 0/c(l_j), 0/c(l_k) \mid l_i : (\texttt{SUB}(3), l_j, l_k)\}, \\
R(5,0) &= \{(c(l_i)-3)/(c(l_j)-1), (c(l_i)-3)/(c(l_k)-1) \mid l_i : (\texttt{ADD}(3), l_j, l_k) \in R\} \\
&\cup \{(c(l_i)-1)/(c(l_i+1)-3), (c(l_i+1))/(c(l_j)), \\
&\quad (c(l_i+1)-2)/c(l_k) \mid l_i : (\texttt{SUB}(3), l_j, l_k)\}, \\
R(5,6) &= \{2/0\}, \quad R(6,0) = \{2/2\}.
\end{aligned}$$

In order to show that tissue P system Π can correctly simulate register machine M, we only need to show how the following six kinds of rules of register machine M are simulated by tissue P system Π.

(1) Simulating ADD instructions that act on register 1.

Let $l_i : (\texttt{ADD}(1), l_j, l_k)$ be an ADD instruction that acts on register 1, and the cell with label 4 contains $c(l_i)$ copies of symbol a (in the initial configuration, the cell with label 4 contains $c(l_0)$ copies of symbol a). The simulation uses the rules:

$$6/5 \in R(4,1), \quad (c(l_i)-6)/(c(l_k)-5) \in R(4,0), \quad (c(l_i)-6)/(c(l_j)-5) \in R(4,0).$$

The simulation takes one step. At this step, six copies of symbol a from the cell with label 4 are sent to the cell with label 1 by the rule $6/5 \in R(4,1)$ (the number n stored in register 1 is encoded by $n+5$ copies of symbol a in the cell with label 1, so the cell with label 1 contains at least five copies of symbol a, and thus the rule $6/5 \in R(4,1)$ can be applied), exchanging with five copies of symbol a. The number of copies of symbol a in the cell with label 1 increases by one, which simulates that the number stored in register 1 is increased by one. At the same time, the other copies of symbol a from the cell with label 4 can be used by the rule $(c(l_i)-6)/(c(l_j)-5) \in R(4,0)$ or $(c(l_i)-6)/(c(l_k)-5) \in R(4,0)$, non-deterministically chosen. Note that the cell with label 4 also gets five copies of symbol a at this step by the rule $6/5 \in R(4,1)$. The cell with label 4 accumulates $c(l_j)$ or $c(l_k)$ copies of symbol a in total after this step. In this way, system Π starts to simulate instruction l_j or l_k.

All objects in cell 4 are simultaneously used by communications along the channels (4,0) and (4,1). If the system does not use at the same time the rule $6/5 \in R(4,1)$ and one of the rules $(c(l_i)-6)/(c(l_j)-5) \in R(4,0)$ or $(c(l_i)-6)/(c(l_k)-5) \in R(4,0)$, then the system will enter an infinite loop and then the computation gives no result.

For instance, the rule $(c(l_i)-6)/(c(l_j)-5) \in R(4,0)$ or $(c(l_i)-6)/(c(l_k)-5) \in R(4,0)$ consumes $c(l_i)-6$ copies of symbol a in the cell with label 4, and there are 6 copies of symbol a that are subject to other possible rules. Because for $0 \leq i\prime, j\prime, k\prime \leq t$, the encoding $c(l_{i\prime})$ of each instruction $l_{i\prime}$ of register machine M is larger than 6, these 6 copies of symbol a cannot be used by the rules $(c(l_{i\prime})-6)/(c(l_{j\prime})-5)$, $(c(l_{i\prime})-6)/(c(l_{k\prime})-5)$, $(c(l_{i\prime})-3)/(c(l_{j\prime})-1)$, $(c(l_{i\prime})-3)/(c(l_{k\prime})-1)$, $(c(l_{i\prime})-1)/(c(l_{i\prime}+1)-3)$, $c(l_{i\prime}+1)/c(l_{j\prime})$, $(c(l_{i\prime}+1)-2)/c(l_{k\prime})$, $c(l_t)/0$ from $R(4,0)$ and the rule $c(l_{i\prime})/0$ from $R(4,5)$. These 6 copies of symbol a can be used by the rule $3/1$ or $1/3$ from $R(4,2)$, but not both, since at most one rule is allowed to be used in each channel at one step. So there are at least 3 copies of symbol a that are subject to other possible

rules. By the maximal parallel manner of application of rules, the rule $2/0 \in R(4,6)$ must be applied at this step, hence two copies of symbol a are sent to the cell with label 6.

Conversely, the rule $6/5 \in R(4,1)$ consumes 6 copies of symbol a in the cell with label 4, and there are $c(l_i) - 6$ copies of symbol a that are subject to other possible rules. For $0 \leq i, i\prime, i\prime\prime, j\prime, k\prime \leq t$, $i\prime \neq i\prime\prime$, $i\prime < i$ and $i\prime\prime < i$, these $c(l_i) - 6$ copies of symbol a can be used by the rule $c(l_{i\prime\prime})/0$ from $R(4,5)$, and one of the rules $(c(l_{i\prime}) - 6)/(c(l_{j\prime}) - 5)$, $(c(l_{i\prime}) - 6)/(c(l_{k\prime}) - 5)$, $(c(l_{i\prime}) - 3)/(c(l_{j\prime}) - 1)$, $(c(l_{i\prime}) - 3)/(c(l_{k\prime}) - 1)$, $(c(l_{i\prime}) - 1)/(c(l_{i\prime} + 1) - 3)$, $c(l_{i\prime} + 1)/c(l_{j\prime})$, $(c(l_{i\prime} + 1) - 2)/c(l_{k\prime})$ from $R(4,0)$. By the first property of the function c, $c(i) > c(i\prime) + c(i\prime\prime) + 12$, there are at least 6 copies of symbol a that are subject to other rules at this moment. These 6 copies of symbol a can be used by the rule $3/1$ or $1/3$ from $R(4,2)$, but not both. Therefore, there are at least 3 copies of symbol a that are subject to the rule $2/0 \in R(4,6)$; the rule $2/0 \in R(4,6)$ must be applied, and two copies of symbol a are sent to the cell with label 6.

In each of these two cases, the cell with label 6 receives two copies of symbol a, and the rule $(2,2) \in R(6,0)$ would be applied forever. Consequently, the rule $6/5 \in R(4,1)$ and one of the rules $(c(l_i) - 6)/(c(l_j) - 5) \in R(4,0)$ or $(c(l_i) - 6)/(c(l_k) - 5) \in R(4,0)$ must be used, hence the instruction $l_i : (\texttt{ADD}(1), l_j, l_k)$ is correctly simulated by system Π (otherwise, the computation cannot halt).

(2) Simulating ADD instructions that act on register 2.

Let $l_i : (\texttt{ADD}(2), l_j, l_k)$ be an ADD instruction that acts on register 2, and the cell with label 4 contains $c(l_i)$ copies of symbol a. The following rules are used to simulate the instruction $l_i : (\texttt{ADD}(2), l_j, l_k)$:

$$3/1 \in R(4,2), \quad (c(l_i) - 3)/(c(l_j) - 1) \in R(4,0), \quad (c(l_i) - 3)/(c(l_k) - 1) \in R(4,0).$$

The simulation takes one step. The content of the cell with label 4 is split into two parts, with one part containing $c(l_i) - 3$ copies of symbol a and the other part containing 3 copies of symbol a. The $c(l_i) - 3$ copies of symbol a in the cell with label 4 are exchanged with $c(l_j) - 1$ copies of symbol a from the environment by the rule $(c(l_i) - 3)/(c(l_j) - 1) \in R(4,0)$, or exchanged with $c(l_k) - 1$ copies of symbol a by the rule $(c(l_i) - 3)/(c(l_k) - 1) \in R(4,0)$, non-deterministically chosen. The 3 copies of symbol a in the cell with label 4 are exchanged with one copy of symbol a from the cell with label 2 by the rule $3/1 \in R(4,2)$. In this way, the number of copies of symbol a in the cell with label 2 increases by two, which simulates that the number stored in register 2 is increased by one. The number of copies of symbol a in the cell with label 4 becomes $c(l_j)$ or $c(l_k)$, which means that system Π starts to simulate instruction l_j or l_k.

Similar to the case of simulating ADD instructions that act on register 1, if system Π does not use the above rules, then the rule $2/2 \in R(6,0)$ can be applied and would be applied forever. The computation is not a successful one and gives no result. Therefore, the ADD instruction $l_i : (\texttt{ADD}(2), l_j, l_k)$ is correctly simulated by system Π.

(3) Simulating SUB instructions that act on register 2.

Let $l_i : (\texttt{SUB}(2), l_j, l_k)$ be an SUB instruction that acts on register 2, and the cell with label 4 contains $c(l_i)$ copies of symbol a. The following rules are used to simulate $l_i : (\texttt{SUB}(2), l_j, l_k)$:

$$(c(l_i) - 1)/(c(l_i + 1) - 3) \in R(4,0), \qquad 1/3 \in R(4,2),$$
$$c(l_i + 1)/c(l_j) \in R(4,0), \qquad (c(l_i + 1) - 2)/c(l_k) \in R(4,0).$$

The simulation takes two steps. Note that one of the main issues in the process of simulating $l_i : (\texttt{SUB}(2), l_j, l_k)$ is to check whether the number stored in register 2 is non-zero. The number n in register 2 is represented by $2n+2$ copies of symbol a in the cell with label 2, so the system should check whether the cell with label 2 contains more than two copies of symbol a. The checking is done as follows. At the first step, the $c(l_i)$ copies of symbol a are split into two parts, with one

part containing $c(l_i) - 1$ copies of symbol a and the other part containing one copy of symbol a. The $c(l_i) - 1$ copies of symbol a can be used by the rule $(c(l_i) - 1)/(c(l_i + 1) - 3) \in R(4, 0)$, exchanging with $c(l_i + 1) - 3$ copies of symbol a. For the remaining copy of symbol a, there are two cases. If the number stored in register 2 is non-zero and thus the cell with label 2 contains at least 4 copies of symbol a, then the only rule which can be used is $1/3 \in R(4, 2)$. If the number stored in register 2 is zero and thus the cell with label 2 contains only 2 copies of symbol a, then no rule can be used and the symbol stays in the cell with label 4. In the case of the number stored in register 2 being non-zero, the number of copies of symbol a in the cell with label 2 decreases by two (this simulates that the number stored in register 2 is subtracted by one), and the number of copies of symbol a in the cell with label 4 becomes $c(l_i + 1)$. For the case of the number stored in register 2 being zero, the number of copies of symbol a in the cell with label 2 keeps unchanged (this simulates that the number stored in register 2 is still zero), and the cell with label 4 accumulates $c(l_i + 1) - 2$ copies of symbol a. At the second step, by using the rules $c(l_i + 1)/c(l_j) \in R(4, 0)$ or $c(l_i + 1) - 2/c(l_k) \in R(4, 0)$, the cell with label 4 eventually obtains $c(l_j)$ or $c(l_k)$ copies of symbol a, respectively. Note that system Π should simulate $l_i : (\mathtt{SUB}(2), l_j, l_k)$ by using the above rules; otherwise, the rule $2/0 \in R(4, 6)$ must be used, which actives a trap in the sense of the rule $2/2 \in R(6, 0)$ being used forever.

(4) Simulating ADD instructions that act on register 3.

Let $l_i : (\mathtt{ADD}(3), l_j, l_k)$ be an ADD instruction that acts on register 3, and the cell with label 4 contains $c(l_i)$ copies of symbol a. The simulation uses the rules:

$$c(l_i)/0 \in R(4, 5), \quad (c(l_i) - 3)/(c(l_j) - 1) \in R(5, 0), \quad (c(l_i) - 3)/(c(l_k) - 1) \in R(5, 0),$$
$$3/1 \in (5, 3), \quad 0/c(l_j) \in R(4, 5), \quad 0/c(l_k) \in R(4, 5).$$

The simulations of instructions that act on register 3 are controlled by the cell with label 5. The simulation of the instruction $l_i : (\mathtt{ADD}(3), l_j, l_k)$ takes three steps. At the first step, the $c(l_i)$ copies of symbol a in the cell with label 4 are sent to cell with label 5 by the rule $c(l_i)/0 \in R(4, 5)$. In the cell with label 5, the $c(l_i)$ copies of symbol a are split into two parts at the second step, with one part containing $c(l_i) - 3$ copies of symbol a and the other part containing 3 copies of symbol a. The $c(l_i) - 3$ copies of symbol a are exchanged with $c(l_j) - 1$ copies of symbol a from the environment by the rule $(c(l_i) - 3)/(c(l_j) - 1) \in R(5, 0)$ or exchanged with $c(l_k) - 1$ copies of symbol a from the environment by the rule $(c(l_i) - 3)/(c(l_k) - 1) \in R(5, 0)$, non-deterministically chosen. The 3 copies of symbol a in cell with label 5 are exchanged with one copy of symbol a from the cell with label 3. In this way, the number of copies of symbol a in the cell with label 3 increase by two, which simulates that the number stored in register 3 is increased by one. The cell with label 5 accumulates $c(l_j)$ or $c(l_k)$ copies of symbol a, these copies of symbol a are sent to the cell with label 4 by the rules $0/c(l_j) \in R(4, 5)$ or $0/c(l_k) \in R(4, 5)$ at the third step. In this way, the system can continue to simulate the next instruction l_j or l_k. Note that system Π should simulate $l_i : (\mathtt{ADD}(3), l_j, l_k)$ by using the above rules; otherwise, the rule $2/0 \in R(5, 6)$ must be used, which actives a trap in the sense of the rule $2/2 \in R(6, 0)$ being used forever. Therefore, the ADD instruction $l_i : (\mathtt{ADD}(3), l_j, l_k)$ is correctly simulated by system Π.

(5) Simulating SUB instructions that act on register 3.

Let $l_i : (\mathtt{SUB}(3), l_j, l_k)$ be an SUB instruction that acts on register 3, and the cell with label 4 contains $c(l_i)$ copies of symbol a. The following rules are used to simulate the instruction $l_i : (\mathtt{SUB}(3), l_j, l_k)$:

$$c(l_i)/0 \in R(4, 5), \quad (c(l_i) - 1)/(c(l_i + 1) - 3) \in R(5, 0), \quad c(l_i + 1)/c(l_j) \in R(5, 0),$$
$$1/3 \in R(5, 3), \quad (c(l_i + 1) - 2)/c(l_k) \in R(5, 0), \quad 0/c(l_j) \in R(4, 5), \quad 0/c(l_k) \in R(4, 5).$$

The simulation takes four steps. Similar to the simulation of ADD instructions that act on register 3, the $c(l_i)$ copies of symbol a in the cell with label 4 are first sent to the cell with label

5 and then the cell with label 5 takes care of the simulation of SUB instructions that act on register 3. At the second step, system Π starts to check whether the number stored in register 3 is non-zero. Note that the number n in register 3 is represented by $2n+2$ copies of symbol a, thus the system should check whether the cell with label 3 contains more than two copies of symbol a. The checking is done as follows. The $c(l_i)$ copies of symbol a in the cell with label 5 are split into two parts, with one part containing one copy of symbol a and the other one containing $c(l_i)-1$ copies of symbol a. The $c(l_i)-1$ copies of symbol a are exchanged with $c(l_i+1)-3$ copies of symbol a from the environment by the rule $(c(l_i)-1)/(c(l_i+1)-3) \in R(5,0)$. For the one copy of symbol a, there are two cases. If the cell with label 3 contains at least 3 copies of symbol a (this corresponds to the fact that the number stored in register 3 is non-zero), then the only rule which can be applied is $1/3 \in R(5,3)$ and thus the number stored in register 3 is decreased by one. If the number of copies of symbol a in the cell with label 3 is less than 3 (this corresponds to the fact that the number stored in register 3 is zero), then no rule can be applied and thus the number in register 3 keeps unchanged; furthermore, the one copy of symbol a stays in the cell with label 5. In this way, the cell with label 5 accumulates either $c(l_i+1)$ copies of symbol a or $c(l_i+1)-2$ copies of symbol a. By using the rules $c(l_i+1)/c(l_j) \in R(5,0)$ or $(c(l_i+1)-2)/c(l_k) \in R(5,0)$, the $c(l_i+1)$ copies of symbol a are exchanged with $c(l_j)$ copies of symbol a or the $c(l_i+1)-2$ copies of symbol a are exchanged with $c(l_k)$ copies of symbol a at the third step. At the fourth step, the $c(l_j)$ or $c(l_k)$ copies of symbol a in the cell with label 5 are sent to the cell with label 4 by the rules $0/c(l_j) \in R(4,5)$ or $0/(l_k) \in R(4,5)$, and system Π starts to simulate the next instruction l_j or l_k. Note that the rules $2/0 \in R(5,6)$ and $2/2 \in R(6,0)$ guarantee that system Π should use the above rules to simulate $l_i : (\mathtt{SUB}(3), l_j, l_k)$; otherwise, system Π would enter into a computation that cannot halt.

(6) Simulating the halting instruction.

The halting instruction l_h is simulated by the rule $c(l_t)/0$ from $R(4,0)$.

When the register machine M reaches the halting instruction l_h, the cell with label 4 accumulates $c(l_t)$ copies of symbol a. At that moment, the rule $c(l_t)/0 \in R(4,0)$ can be used, by which all the copies of symbol a in the cell with label 4 are moved into the environment, and thus the computation halts.

From the above explanation, we can find that the register machine M can be correctly simulated by tissue P system Π. From the formal definition of tissue P system Π, we see that system Π has at most one channel between two cells or between a cell and the environment. Therefore, Theorem 1 holds. □

Corollary 2. *For any recursively enumerable set L of natural numbers, a one-symbol tissue P system with symport/antiport rules having at most 5 cells can be constructed to generate L, when two channels are allowed between a cell and the environment.*

Proof: Let us first recall that the cell with label 6 in system Π constructed in the proof of Theorem 1 is used as a trap. If two channels are allowed between a cell and the environment, then the "trap" cell can be saved by introducing two antiport rules $2/2c(l_t) \in R(4,0)$ and $2/2c(l_t) \in R(5,0)$, where the function c is defined as in the proof of Theorem 1. These two rules are used as a "trap" in the sense that, once one of the rules is used, then the rule will be used forever. Note that the number $2c(l_t)$ is so large that even in the case of all possible rules which can be used being used, there remain enough copies of symbol a to repeat the "trap" rule $2/2c(l_t) \in R(4,0)$ or $2/2c(l_t) \in R(5,0)$. In this way, the computation enters an infinite loop. The changes in the construction from the proof of Theorem 1 are left to the reader. □

5 Conclusions and Comments

In this note, a one-symbol universal tissue P system with symport/antiport rules having six cells is obtained, under the restriction that one channel is allowed between two cells or between a cell and the environment. As a corollary of the above result, a one-symbol universal tissue P system with symport/antiport rules having five cells is also constructed, under the restriction that two channel are allowed between a cell and the environment. These results partially answer open problems formulated in [1] (see Tables 1 and 2).

Some improvement of the number of cells used in the universal tissue P systems given in this work may be still possible (thus answering more open problems in Tables 1 and 2). A natural idea is to consider removing the trap cell, just as done in the case of allowing two channels between the cell and the environment.

In the universal tissue P systems given in this work, two cells have been used to control the simulation of the SUB instructions of register machine: the cell with label 4 is responsible for the simulation of the SUB instructions that act on register 2; the cell with label 5 takes care of the simulation of the SUB instructions that act on register 3. We conjecture that one cell is enough to take care of the simulation of the SUB instructions that act on registers 2 and 3.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61033003, 30870826, 61003131 and 61003038), the Fundamental Research Funds for the Central Universities (2010ZD001), Ph.D. Programs Foundation of Ministry of Education of China (20100142110072), the Opening Foundation of Key Laboratory of University of Science and Technology of China for High-Performance Computing and Applications (Grant No. NHPCC-KF-1102), and Scientific Research Foundation for Doctor of Anhui University (Grant No. 02203104).

Bibliography

[1] A. Alhazov, R. Freund, M. Oswald, Tissue P Systems with Symport/Antiport Rules and Small Numbers of Symbols and Cells, *Lecture Notes in Computer Science*, 3572:100–111, 2005.

[2] R. Freund, Gh. Păun, M.J. Pérez-Jiménez, Tissue-Like P Systems with Channel States, *Theoretical Computer Science*, 296:295–326, 2003.

[3] R. Freund, M. Oswald, Tissue P Systems with Symport/Antiport Rules of One Symbol Are Computational Complete, in: M.A. Gutiérrez-Naranjo, Gh. Păun, M.J. Pérez-Jiménez (eds.), *Proceedings of the European Science Foundation PESC Exploratory Workshop Cellular Computing (Complexity Aspects)*, Sevilla, pp.178–187, 2005.

[4] S.N. Krishna, K. Lakshmanan, R. Rama, Tissue P Systems with Contextual and Rewriting Rules, *Lecture Notes in Computer Science*, 2597:339–351, 2003.

[5] M. Minsky (eds.), *Computation: Finite and Infinite Machines*, Prentice Hall, 1967.

[6] C. Martín Vide, J. Pazos, Gh. Păun, A. Rodríguez Patón, Tissue P Systems, *Theoretical Computer Science*, 296:295–326, 2003.

[7] Gh. Păun, Computing with Membranes, *Journal of Computer and System Sciences*, 61(1):108–143, 2000.

[8] A. Păun, Gh. Păun, The Power of Communication: P Systems with Symport/Antiport, *New Generation Computing*, 20(3):295–305, 2002.

[9] Gh. Păun, G. Rozenberg, A. Salomaa (eds.), *Handbook of Membrane Computing*, Oxford University Press, 2010.

[10] Y. Rogozhin, S. Verlan, On the Rule Complexity of Universal Tissue P Systems, *Lecture Notes in Computer Science*, 3850:356–362, 2006.

[11] The P systems web page: http://ppage.psystems.eu