Cell-Like P Systems With Channel States and Symport/Antiport Rules

Bosheng Song, Linqiang Pan*, *Member, IEEE*, and Mario J. Pérez-Jiménez

***Abstract*— Cell-like P systems with symport/antiport rules are inspired by the structure of a cell and the way of communicating substances through membrane channels between neighboring regions. In this work, channel states are introduced into cell-like P systems with symport/antiport rules, and we call this variant of communication P systems as cell-like P systems with channel states and symport/antiport rules. In such P systems, at most one channel is established between neighboring regions, each channel associates with one state in order to control communication at each step, and rules are used in a sequential manner: on each channel at most one rule can be used at each step. The computational power of such P systems is investigated. Specifically, we show that cell-like P systems with two states and using uniport rules, or with any number of states and using antiport rules of length two, are able to compute only finite sets of non-negative integers. We further prove that cell-like P systems with two membranes are as powerful as Turing machines when channel states and symport/antiport rules are suitably combined. The results show that channel states are a feature that can increase the computational power of cell-like P systems with symport/antiport rules.**

***Index Terms*— Bio-inspired computing, channel state, membrane computing, P system, universality.**

I. Introduction

MEMBRANE COMPUTING is a fast-growing branch of natural computing, which arose as an abstraction of the architecture and functioning of living cells, and the way in which biochemical substances are processed in a single cell or in a net of cells [1]. The computing devices investigated in membrane computing are called *P systems*. Many variants of P systems were introduced with mathematical, computer science or biological motivation [2]–[5], and many of them have been applied to solve real problems [6]–[10]. According to the membrane structure, there are two main families of P systems: *cell-like P systems* which have a hierarchical arrangement of membranes as in a cell [1] and *tissue-like P systems* or *neural-like P systems* which have several one-membrane cells as in a tissue or a neural net [11], [12]. A comprehensive introduction of membrane computing can be found in the monograph [13], [14], and for the most up-to-date source of information on both general and technical levels, one can go to the P systems web page `http://ppage.psystems.eu`.

A cell-like P system consists of a hierarchical arrangement of membranes, each membrane delimiting a *compartment* (also called *region*), where multisets of objects and rules to make these objects evolve are placed. A membrane with no compartments inside is called *elementary*, otherwise it is called *non-elementary*. The outmost membrane is called a *skin* membrane, the space outside the skin membrane is called the *environment*. Evolution rules have the form of either *rewriting* or *communication (symport/antiport)* rules. The present work focuses on symport/antiport rules, which were proposed in [15], [16]. A symport rule is written in the forms (u, in) or (u, out), meaning that the objects specified by multiset u can enter or exit the membrane with which the rule is associated. Uniport rules are those by which only one object is moved. An antiport rule is written in the form $(u, out; v, in)$, meaning that the objects of u exit and simultaneously, those of v enter the membrane with which the rule is associated. The length of a communication rule is the total number of objects involved in that rule.

The computational power of cell-like P systems with symport/antiport rules has been investigated widely, and under several strategies concerning the use of rules such P systems are proved to be computationally complete [15], [17]–[24]. An interesting strategy concerning the use of symport/antiport rules was presented in [25], where channel states were introduced into tissue P systems to control the symport/antiport rules. Hence a variant of tissue P systems, called *tissue P systems with channel states*, was proposed, where there is at most one channel between two cells or between a cell and the environment, a state is associated with each channel to control the communication at each step, and rules are used in a sequential manner: on each channel at most one rule can be used at a computation step.

It is known that cell-like P systems with two membranes and using symport rules of length 2 or using symport rules of rule 1 and antiport rules of length 2 are Turing universal (see [14, Theorem 5.10, Ch. 5]). However, in these systems, rules are used in a maximally parallel manner, that is, at each step, a maximal multiset of rules between two cells or between a cell and the environment are applied, no further rule can be added being applicable.

The work of B. Song and L. Pan was supported by National Natural Science Foundation of China (61033003, 91130034, 61602192, and 61320106005), Ph.D. Programs Foundation of Ministry of Education of China (20120142130008), the Innovation Scientists and Technicians Troop Construction Projects of Henan Province (154200510012). The work of M. J. Pérez-Jiménez was supported by "Ministerio de Economía y Competitividad" of Spanish government (TIN2012-37434), cofunded by FEDER funds. *Asterisk indicates corresponding author.*

B. Song is with the Key Laboratory of Image Information Processing and Intelligent Control, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China.

L. Pan is with the Key Laboratory of Image Information Processing and Intelligent Control, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China. (e-mail: lqpan@mail.hust.edu.cn).

M. J. Pérez-Jiménez is with the Department of Computer Science and Artificial Intelligence, University of Sevilla, Sevilla Avda. Reina Mercedes s/n, 41012, Spain.

In this work, channel states are introduced into cell-like P systems with symport/antiport rules, and we call this new variant of communication P systems as *cell-like P systems with channel states and symport/antiport rules*. In such P systems, at most one channel is established between neighboring regions, a state is associated with each channel to control the communication at each step, and rules are used in a sequential manner at the level of each channel (on each channel associated with two neighboring regions, at most one rule can be used at one step) and in a parallel manner at the level of the system (all channels which can use a rule must do it).

The computational power of cell-like P systems with channel states and symport/antiport rules is investigated. Specifically, we prove that such P systems with two states and only using uniport rules, or with any number of states and using antiport rules of length two, are able to compute only finite sets of non-negative integers. We further show that cell-like P systems with two membranes are Turing universal when having any number of states and only using uniport rules, or four states and only using symport rules of length two, or two states and only using symport rules of length three, or one state and only using antiport rules of length three, or three states and using uniport rules and antiport rules of length two.

It is known that cell-like P systems with two membranes using only symport rules of length at most 1 and working in a sequential manner are not Turing universal. It will be proved that such kind of P systems with channel states are Turing universal (see Theorem 4.4) (note that by the definition, cell-like P systems with channel states work in a sequential manner). This comparison shows that the feature of channel states can increase the computational power of cell-like P systems with symport/antiport rules.

II. Preliminaries

In this section, we recall some basic notions and notations from formal language theory and membrane computing that we use in this work. For further details and information the reader is referred to [14], [26].

An *alphabet* Γ is a non-empty set and its elements are called *symbols*. A *string* u over Γ is a finite sequence of symbols from Γ. The number of occurrences in u of symbols from Γ is called the *length* of the string u, denoted by $|u|$. The empty string (with length 0) is denoted by λ. The set of all strings over an alphabet Γ is denoted by Γ^*, and by $\Gamma^+ = \Gamma^* \backslash \{\lambda\}$ we denote the set of all non-empty strings. A *language* over Γ is a set of strings over Γ.

A *multiset* over an alphabet Γ is a function $m : \Gamma \to \mathbb{N}$, which gives a nonnegative *multiplicity* $m(x)$ for each $x \in \Gamma$. The *support* of a multiset m is defined as $supp(m) = \{x \in \Gamma \mid f(x) > 0\}$. A multiset is finite if its support is a finite set. We denote by $\emptyset$ the empty multiset. The size of a finite multiset $m = (\Gamma, f)$ is $\sum_{x \in \Gamma} f(x)$, and it is denoted by $|m|$. Let m_1, m_2 be multisets over Γ. The union of m_1 and m_2, denoted by $m_1 + m_2$, is the multiset over Γ defined as $(m_1 + m_2)(x) = m_1(x) + m_2(x)$ for each $x \in \Gamma$. We say that m_1 is contained in m_2 and we denote it by $m_1 \subseteq m_2$, if $m_1(x) \leq m_2(x)$ for each $x \in \Gamma$. The relative complement of m_2 in m_1, denoted by $m_1 \backslash m_2$, is the multiset defined as $(m_1 \backslash m_2)(x) = m_1(x) - m_2(x)$ if $m_1(x) \geq m_2(x)$, and $(m_1 \backslash m_2)(x) = 0$ otherwise.

We denote by $NFIN$ the family of all finite sets of positive integers. By $NMAT$ we denote the family of sets of positive integers generated by matrix grammars without appearance checking. By NRE we denote the family of recursively enumerable sets of natural numbers. It is known that $NMAT \subset NRE$ [27].

A *register machine* is a tuple $M = (m, H, l_0, l_h, I)$, where

- m is the number of registers;
- H is a set of labels;
- $l_0, l_h \in H$ are distinguished labels, where l_0 is the label of the initial instruction, and l_h is the label of the halting instruction;
- I is a set of labeled program instructions of the following forms (each label from H labels only one instruction from I, thus precisely identifying it):
  - 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, non-deterministically chosen);
  - 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 : HALT (the halt instruction).

A register machine M generates a set $N(M)$ of numbers in the following way: initially, all registers of the machine are empty (i.e., storing the number zero); the machine starts to apply the instruction with label l_0 and continues to apply instructions as indicated by the labels (and made possible by the contents of registers); if the machine M reaches the halt instruction l_h, then the number n presented in specified register 1 at that time is said to be generated by M. If the computation does not halt, then no number is generated. It is known that register machines generate all sets of numbers which are Turing computable, hence they characterize NRE [28], [29].

III. Cell-Like P Systems With Channel States and Symport/Antiport Rules

In this section, we give the definition of cell-like P systems with channel states and symport/antiport rules.

Definition 3.1: A cell-like P system with channel states and symport/antiport rules, of degree $q \geq 1$, is a tuple

$$\Pi = (\Gamma, K, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, s_1, \ldots, s_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{out}),$$

where

- Γ is a finite alphabet of objects and $\mathcal{E} \subseteq \Gamma$;
- K is an alphabet of states (not necessarily disjoint of Γ);
- μ is a rooted tree with q nodes labelled by $1, \ldots, q$;
- $\mathcal{M}_i$, $1 \leq i \leq q$, are finite multisets over Γ;
- s_i, $1 \leq i \leq q$, are channel states;
- $\mathcal{R}_i$, $1 \leq i \leq q$, are finite sets of rules of the following forms (associated with the channels):
  - Symport rules: $(s, (u, in), s')$ or $(s, (u, out), s')$, for $s, s' \in K$, $u \in \Gamma^+$;
  - Antiport rules: $(s, (u, out; v, in), s')$, for $s, s' \in K$, $u, v \in \Gamma^+$;
- $i_{out} \in \{0, 1, \ldots, q\}$.

A cell-like P system with channel states and symport/antiport rules of degree $q \geq 1$ can be viewed as a set of q membranes labelled by $1, \ldots, q$, arranged in a hierarchical structure of a rooted tree (the root labelled by 1 is the skin membrane), such that: a) $\mathcal{M}_1, \ldots, \mathcal{M}_q$ represent the finite multisets of objects initially placed in the q membranes of the system; b) $\mathcal{E}$ is the set of objects initially located in the environment of the system, all of them available in an arbitrary number of copies; c) $s_1, \ldots, s_q$ are initial channel states between neighboring regions; specifically s_i is the initial state associated with the channel between membrane i and its parent; d) $\mathcal{R}_1, \ldots, \mathcal{R}_q$ are finite sets of rules ($\mathcal{R}_i$ corresponds to membrane i of μ); e) i_{out} is a distinguished region which will encode the output of the system. The term region i ($0 \leq i \leq q$) refers to membrane i in case $1 \leq i \leq q$ and to the environment in case $i = 0$. For each membrane $i \in \{2, \ldots, q\}$, we denote by $p(i)$ the parent of membrane i in the rooted tree μ, the "parent" of the skin membrane is the environment, denoted by $p(1) = 0$. The length of a symport rule $(s, (u, in), s')$ or $(s, (u, out), s')$ (an antiport rule $(s, (u, out; v, in), s')$, respectively) is defined as $|u|$ ($|u| + |v|$, respectively). It is worth pointing out that any rule in $\mathcal{R}_i$ involves membrane i and its parent region $p(i)$.

We note the important restriction that there is at most one channel between neighboring regions, where at each step a state from K is associated. This does not restrict the communication between the neighboring regions, because the movement of objects in the two directions of a channel is allowed.

A symport rule $(s, (u, out), s') \in \mathcal{R}_i$ is applicable to a configuration at a moment if the channel between membrane i and its parent region $p(i)$ has the state s and membrane i contains multiset u at that moment. When such a rule is applied, multiset u is sent to region $p(i)$ and the channel state between region i and region $p(i)$ is changed from s to s'.

A symport rule $(s, (u, in), s') \in \mathcal{R}_i$ is applicable to a configuration at a moment if the channel between membrane i and its parent region $p(i)$ has the state s and the region $p(i)$ contains multiset u at that moment. When such a rule is applied, multiset u enters membrane i from region $p(i)$ and the channel state between region i and region $p(i)$ is changed from s to s'.

An antiport rule $(s, (u, out; v, in), s') \in \mathcal{R}_i$ is applicable to a configuration at a moment if the channel between membrane i and its parent region $p(i)$ has the state s, and membrane i contains multiset u as well as its parent region contains multiset v at that moment. When such a rule is applied, multiset u from membrane i is sent to region $p(i)$, at the same time multiset v enters region i from region $p(i)$, and the channel state between region i and region $p(i)$ is changed from s to s'.

The rules of a cell-like P system with channel states and symport/antiport rules are applied in a sequential manner at the level of each channel (on each channel associated with two neighboring regions, at most one rule can be used at one step) and in a parallel manner at the level of the system (all channels which can use a rule must do it).

A *configuration* of a cell-like P system with channel states and symport/antiport rules at any instant is described by all multisets of objects over Γ associated with the regions, all states associated with each channel and the multiset of objects over $\Gamma \setminus \mathcal{E}$ associated with the environment at that moment. Note that the objects from $\mathcal{E}$ have infinite copies, so they are not properly changed along the computation. The *initial configuration* is $(\mathcal{M}_1, \ldots, \mathcal{M}_q, s_1, \ldots, s_q; \emptyset)$.

Starting from the initial configuration and applying rules as described above, one obtains a sequence of consecutive configurations. Each passage from a configuration to a successor configuration is called a *transition*. A configuration is a *halting configuration* if no rule of the system is applicable to it. A sequence of transitions starting from the initial configuration is a *computation*. Only a halting computation gives a result, encoded by the number of copies of objects present in the output region i_{out}.

The set of natural numbers computed in the way mentioned above by a cell-like P system with channel states and symport/antiport rules Π is denoted by $N(\Pi)$. The family of all sets of numbers computed by such P systems with at most m membranes, k states, and using symport rules of length at most t_1, antiport rules of length at most t_2 is denoted by $NOP_m(state_k, sym_{t_1}, anti_{t_2})$. Note that if only symport rules (resp., only antiport rules) are used in a system, then it is simply denoted by $NOP_m(state_k, sym_{t_1})$ (resp., $NOP_m(state_k, anti_{t_2})$). If one of the parameters m, k, t_1, t_2 is not bounded, then it is replaced with $*$.

IV. Computational Power of Cell-Like P Systems With Channel States and Symport/Antiport Rules

In this section, we investigate the computational power of cell-like P systems with channel states and symport/antiport rules.

In the proofs of Theorems 4.4, 4.5, 4.6, 4.7 and 4.8, we consider a register machine $M = (m, H, l_0, l_h, I)$ as described in Section II.

Theorem 4.1: $NOP_*(state_*, anti_2) \subseteq NFIN$.

Proof: The proof follows from the fact that the number of objects in a membrane cannot be changed by using only antiport rules of length 2, therefore the number of objects in any cell-like P system with antiport rules of length 2 will not change during any sequence of transitions starting from the initial configuration and ending with a halting configuration, and the number of channel states is finite during this process. Hence, only finite sets of natural numbers can be generated.

Theorem 4.2: $NOP_*(state_2, sym_1) \subseteq NFIN$.

Proof: It is enough to observe that the number of objects in the system can increase only if a rule $(s, (a, in), s')$ with $a \in \mathcal{E}$, is used in the skin membrane (the channel state between the skin membrane and the environment must be changed, otherwise the system never halts); on the other hand, if there is a rule $(s, (a, in), s')$ with $a \in \mathcal{E}$ and a rule $(s', (b, in), s)$ for $b \in \mathcal{E}, a \neq b$ in the skin membrane then the computation never halts. Hence it is easy to see that for any halting computation, at most $n + 1$ objects may be generated, with n being the number of objects initially present in the system. Therefore, the generated set of numbers is included in $NFIN$.

Tissue P systems with one cell can be viewed as cell-like P systems with one membrane and symport/antiport rules, hence we have the following results [25].

Theorem 4.3: $NOP_1(state_1, anti_4) = NOP_1(state_*, sym_1, anti_2) = NMAT$.

In what follows, several Turing universal results are obtained by cell-like P systems when channel states and symport/antiport rules are suitably combined.

Theorem 4.4: $NOP_2(state_*, sym_1) = NRE$.

Proof: We only prove the inclusion $NRE \subseteq NOP_2(state_*, sym_1)$. The reverse inclusion follows from the Church-Turing thesis.

Let us consider a register machine $M = (m, H, l_0, l_h, I)$. We construct the cell-like P system with channel states and symport/antiport rules Π to simulate the register machine M.

$$\Pi = (\Gamma, K, \mathcal{E}, \mu, \mathcal{M}_1, \mathcal{M}_2, l_0, s, \mathcal{R}_1, \mathcal{R}_2, 1),$$

where

- $\Gamma = \{a_i \mid 1 \le i \le m\} \cup \{b, e\}$,
- $K = \{s, s', s'', s'''\} \cup \{l, l', l'', l''', l^{iv} \mid l \in H\}$,
- $\mathcal{E} = \{a_i \mid 1 \le i \le m\} \cup \{e\}$,
- $\mu = [\ [\ ]_2\ ]_1$,
- $\mathcal{M}_1 = \emptyset$, $\mathcal{M}_2 = \{b\}$, and the sets $\mathcal{R}_1$, $\mathcal{R}_2$ of rules are as follows:
- For each ADD instruction $l_i : (\mathtt{ADD}(r), l_j, l_k)$ of M, we consider the following rules in $\mathcal{R}_1$:
  $r_1 : (l_i, (a_r, in), l_j)$,
  $r_2 : (l_i, (a_r, in), l_k)$

An ADD instruction l_i is simulated obviously. Under the control of channel state l_i between membrane 1 and the environment, one of rules r_1 and r_2 is used non-deterministically. By applying rule r_1 (resp., r_2), one copy of object a_r is introduced into membrane 1 (simulating that the number stored in register r is increased by one), and the channel state between membrane 1 and the environment is changed to l_j (resp., l_k). Hence, the system starts to simulate an instruction with label l_j or l_k. Clearly, the instruction l_i of M is correctly simulated by Π.

- For each SUB instruction $l_i : (\mathtt{SUB}(r), l_j, l_k)$ of M,
  - we consider the following rules in $\mathcal{R}_1$:
    $r_3 : (l_i, (e, in), l_i')$,
    $r_4 : (l_i', (a_r, out), l_i'')$,
    $r_5 : (l_i'', (b, out), l_i'')$,
    $r_6 : (l_i', (b, out), l_i')$,
    $r_7 : (l_i'', (b, in), l_i''')$,
    $r_8 : (l_i', (b, in), l_i^{iv})$,
    $r_9 : (l_i''', (e, out), l_j)$,
    $r_{10} : (l_i^{iv}, (e, out), l_k)$;
  - and we consider the following rules in $\mathcal{R}_2$:
    $r_{11} : (s, (e, in), s')$,
    $r_{12} : (s', (b, out), s'')$,
    $r_{13} : (s'', (e, out), s''')$,
    $r_{14} : (s''', (b, in), s)$.

A SUB instruction l_i is simulated by system Π in the following way: At step 1, under the influence of channel state l_i between membrane 1 and the environment, one copy of object e is sent into membrane 1 by using rule r_3, and the channel state is changed to l_i'. In what follows, we have two cases:

TABLE I

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION $l_i : (\mathtt{SUB}(r), l_j, l_k)$ WITH REGISTER r NOT EMPTY, WHERE z, z' ARE MULTISETS OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$, $z = z'\, a_r$

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s_1	s_2	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	l_i	s	z	$\{b\}$
1	r_3	$-$	l_i'	s	$z \cup \{e\}$	$\{b\}$
2	r_4	r_{11}	l_i''	s'	z'	$\{b, e\}$
3	$-$	r_{12}	l_i''	s''	$z' \cup \{b\}$	$\{e\}$
4	r_5	r_{13}	l_i''	s'''	$z' \cup \{e\}$	$-$
5	r_7	$-$	l_i'''	s'''	$z' \cup \{b, e\}$	$-$
6	r_9	r_{14}	l_j	s	z'	$\{b\}$

- There is at least one copy of object a_r in membrane 1 (corresponding to the fact that the number stored in register r is greater than 0). In this case, at step 2, rule r_4 is enabled and applied, one copy of object a_r is sent to the environment (one copy of object a_r is consumed), and channel state between membrane 1 and the environment is changed from l_i' to l_i''; simultaneously, by using rule r_{11}, object e is sent into membrane 2, the channel state between membrane 2 and membrane 1 is changed from s to s'. At step 3, rule r_{12} is used, object b is sent out of membrane 2 (changing the channel state between membrane 2 and membrane 1 to s''), which will be sent to the environment at the next step by applying rule r_5; by using rule r_{13}, object e is sent out of membrane 2, the channel state between membrane 2 and membrane 1 is changed to s'''. At step 5, rule r_7 is enabled and used, object b is sent into membrane 1, which will be sent back to membrane 2 at the next step by using rule r_{14} (the channel state between membrane 2 and membrane 1 is changed back to s again); simultaneously, the channel state between membrane 1 and the environment is changed to l_i''', which will be changed to l_j at step 6 by using rule r_9. Hence in this case, one copy of object a_r is consumed in membrane 1 (simulating that the number stored in register r is decreased by one), and the system starts to simulate the instruction l_j (see Table I).
- There is no object a_r in membrane 1 (corresponding to the fact that the number stored in register r is 0). In this case, at step 2, only rule r_{11} is enabled and applied, object e is sent into membrane 2 and object b will be sent out of this membrane at the next step. At step 4, by using rule r_6, object b is sent to the environment, which will be sent into membrane 1 at the next step by using rule r_8, the channel state between membrane 1 and the environment is changed from l_i' to l_i^{iv}; by applying rule r_{13}, the channel state between membrane 2 and membrane 1 is changed to s'''. At step 6, rule r_{14} is applied, object b is sent back to membrane 2, the channel state between membrane 2 and membrane 1 is changed to s again; simultaneously, by using rule r_{10}, the channel state between membrane 1 and the environment is changed to l_k. Hence, the system starts to simulate the instruction l_k (see Table II).

TABLE II

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION $l_i : (\mathtt{SUB}(r), l_j, l_k)$ WITH REGISTER r EMPTY, WHERE z IS A MULTISET OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s_1	s_2	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	l_i	s	z	$\{b\}$
1	r_3	$-$	l'_i	s	$z \cup \{e\}$	$\{b\}$
2	$-$	r_{11}	l'_i	s'	z	$\{b, e\}$
3	$-$	r_{12}	l'_i	s''	$z \cup \{b\}$	$\{e\}$
4	r_6	r_{13}	l'_i	s'''	$z \cup \{e\}$	$-$
5	r_8	$-$	l^{iv}_i	s'''	$z \cup \{b, e\}$	$-$
6	r_{10}	r_{14}	l_k	s	z	$\{b\}$

Hence, the SUB instruction of M is correctly simulated by system Π.

When the channel state between membrane 1 and the environment is l_h, no rule can be used in the system, and the computation halts. The number of the copies of object a_1 in membrane 1 corresponds to the result of the computation, hence $N(M) = N(\Pi)$.

Theorem 4.5: $NOP_2(state_4, sym_2) = NRE$.

Proof: We only prove the inclusion $NRE \subseteq NOP_2(state_4, sym_2)$. The reverse inclusion follows from the Church-Turing thesis.

Let us consider a register machine $M = (m, H, l_0, l_h, I)$. We construct the cell-like P system with channel states and symport/antiport rules Π to simulate the machine M.

$$\Pi = (\Gamma, K, \mathcal{E}, \mu, \mathcal{M}_1, \mathcal{M}_2, s, s, \mathcal{R}_1, \mathcal{R}_2, 1),$$

where

- $\Gamma = \{a_i \mid 1 \leq i \leq m\} \cup \{b, b', b'', l, l', l'' \mid b, l \in H\} \cup \{e\}$,
- $K = \{s, s', s'', s'''\}$,
- $\mathcal{E} = \{a_i \mid 1 \leq i \leq m\} \cup \{l, l', l'' \mid l \in H\} \cup \{e\}$,
- $\mu = [[]_2]_1$,
- $\mathcal{M}_1 = \{l_0\}$, $\mathcal{M}_2 = \{b, b', b'' \mid b \in H\}$, and the sets $\mathcal{R}_1$, $\mathcal{R}_2$ of rules are constructed as follows:
- For each ADD instruction $l_i : (\mathtt{ADD}(r), l_j, l_k)$ of M, we consider the following rules in $\mathcal{R}_1$:
  $r_1 : (s, (l_i b_i, out), s)$,
  $r_2 : (s, (b_i a_r, in), s')$,
  $r_3 : (s', (b_i, out), s')$,
  $r_4 : (s', (b_i l_j, in), s'')$,
  $r_5 : (s', (b_i l_k, in), s'')$,
  $r_6 : (s'', (b_i, out), s'')$,
  $r_7 : (s'', (b_i e, in), s)$;
- and we consider the following rules in $\mathcal{R}_2$:
  $r_8 : (s, (l_i, in), s)$,
  $r_9 : (s, (l_i b_i, out), s')$,
  $r_{10} : (s', (b_i e, in), s)$

An ADD instruction l_i is simulated by system Π in the following way: At step 1, under the control of channel state s between membrane 2 and membrane 1, object l_i is sent into membrane 2 by using rule r_8, this object together with object b_i will be sent out of this membrane at the next step (by using rule r_9), and the channel state between membrane 2 and membrane 1 is changed from s to s'. With the appearance of object b_i in membrane 1, rule r_1 is enabled at step 3; by using this rule, object b_i is sent to the environment, and one copy of object a_r will be sent into membrane 1 at the next step (by applying rule r_2), the channel state between membrane 1 and the environment is changed to s'. At step 5, under the influence of channel state s' between membrane 1 and the environment, rule r_3 can be used, object b_i is sent to the environment again. At the next step, one of rules r_4 and r_5 is used non-deterministically. By applying rule r_4 (resp., r_5), objects b_i, l_j (resp., b_i, l_k) are sent into membrane 1, changing the channel state between membrane 1 and the environment to s''. Note that the instruction l_j or l_k starts to be simulated only if the channel state between membrane 2 and membrane 1 is s, where this state appears only at the last step (as we will see below, the channel state between membrane 2 and membrane 1 is changed to s at step 9, which is the last step for simulating an ADD instruction). At the next two steps, by using rules r_6 and r_7 one by one, object b_i is sent to the environment and this object together with object e will be sent into membrane 1, changing the channel state between membrane 1 and the environment to s again. At step 9, rule r_{10} is enabled and applied, object b_i is sent back to membrane 2, and the channel state between membrane 2 and membrane 1 is changed from s' to s. Hence, one copy of object a_r is introduced in membrane 1 (simulating that the number stored in register r is increased by one), the system starts to simulate the instruction with label l_j or l_k. So the instruction l_i of M is correctly simulated by Π (see Table III).

- For each SUB instruction $l_i : (\mathtt{SUB}(r), l_j, l_k)$ of M,
- we consider the following rules in $\mathcal{R}_1$:
  - $r_{11} : (s, (b'_i l_i, out), s)$,
    $r_{12} : (s, (b'_i l'_i, in), s')$,
    $r_{13} : (s', (b'_i, out), s')$,
    $r_{14} : (s', (b'_i l''_i, in), s'')$,
    $r_{15} : (s'', (l'_i a_r, out), s''')$,
    $r_{16} : (s''', (b''_i, out), s''')$,
    $r_{17} : (s'', (b''_i l'_i, out), s'')$,
    $r_{18} : (s''', (b''_i l_j, in), s)$,
    $r_{19} : (s'', (b''_i l_k, in), s)$,
    $r_{20} : (s, (b''_i l''_i, out), s)$,
    $r_{21} : (s, (b''_i e, in), s)$;
- and we consider the following rules in $\mathcal{R}_2$:
  $r_{22} : (s, (l_i, in), s)$,
  $r_{23} : (s, (b'_i l_i, out), s')$,
  $r_{24} : (s', (b'_i l''_i, in), s')$,
  $r_{25} : (s', (b''_i l''_i, out), s')$,
  $r_{26} : (s', (b''_i e, in), s)$

A SUB instruction l_i is simulated by system Π in the following way: At step 1, rule r_{22} is used, object l_i is sent into membrane 2, this object together with object b'_i will be sent out of this membrane at the next step by using rule r_{23}, and the channel state between membrane 2 and membrane 1 is changed to s'. At step 3, rule r_{11} is used, object b'_i is sent to the environment, and object l'_i will be introduced into membrane 1 at the next step by using rule r_{12}, the channel state between membrane 1 and the environment is changed from s to s'. At the next

TABLE III

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF AN ADD INSTRUCTION l_i : (ADD(r), l_j, l_k), WHERE z IS A MULTISET OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m, e\}$, AND $u(H), v(H), w(H)$ ARE MULTISETS WHICH CONTAIN EACH $b \in H, b' \in H, b'' \in H$ EXACTLY ONCE, RESPECTIVELY

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s_1	s_2	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	s	s	$z \cup \{l_i\}$	$u(H) \cup v(H) \cup w(H) \cup \{e^*\}$
1	$-$	r_8	s	s	z	$u(H) \cup v(H) \cup w(H) \cup \{l_i, e^*\}$
2	$-$	r_9	s	s'	$z \cup \{l_i, b_i\}$	$(u(H) \setminus \{b_i\}) \cup v(H) \cup w(H) \cup \{e^*\}$
3	r_1	$-$	s	s'	z	$(u(H) \setminus \{b_i\}) \cup v(H) \cup w(H) \cup \{e^*\}$
4	r_2	$-$	s'	s'	$z \cup \{b_i, a_r\}$	$(u(H) \setminus \{b_i\}) \cup v(H) \cup w(H) \cup \{e^*\}$
5	r_3	$-$	s'	s'	$z \cup \{a_r\}$	$(u(H) \setminus \{b_i\}) \cup v(H) \cup w(H) \cup \{e^*\}$
6	r_4 or r_5	$-$	s''	s'	$z \cup \{a_r, b_i, l_j\}$ or $z \cup \{a_r, b_i, l_k\}$	$(u(H) \setminus \{b_i\}) \cup v(H) \cup w(H) \cup \{e^*\}$
7	r_6	$-$	s''	s'	$z \cup \{a_r, l_j\}$ or $z \cup \{a_r, l_k\}$	$(u(H) \setminus \{b_i\}) \cup v(H) \cup w(H) \cup \{e^*\}$
8	r_7	$-$	s	s'	$z \cup \{a_r, b_i, e, l_j\}$ or $z \cup \{a_r, b_i, e, l_k\}$	$(u(H) \setminus \{b_i\}) \cup v(H) \cup w(H) \cup \{e^*\}$
9	$-$	r_{10}	s	s	$z \cup \{a_r, l_j\}$ or $z \cup \{a_r, l_k\}$	$u(H) \cup v(H) \cup w(H) \cup \{e^*\}$

TABLE IV

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION l_i : (SUB(r), l_j, l_k) WITH REGISTER r NOT EMPTY, WHERE z, z' ARE MULTISETS OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$, $z = z'a_r$, AND $u(H), v(H), w(H)$ ARE MULTISETS WHICH CONTAIN EACH $b \in H, b' \in H, b'' \in H$ EXACTLY ONCE, RESPECTIVELY

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s_1	s_2	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	s	s	$z \cup \{l_i\}$	$u(H) \cup v(H) \cup w(H) \cup \{e^*\}$
1	$-$	r_{22}	s	s	z	$u(H) \cup v(H) \cup w(H) \cup \{l_i, e^*\}$
2	$-$	r_{23}	s	s'	$z \cup \{b'_i, l_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\}) \cup w(H) \cup \{e^*\}$
3	r_{11}	$-$	s	s'	z	$u(H) \cup (v(H) \setminus \{b'_i\}) \cup w(H) \cup \{e^*\}$
4	r_{12}	$-$	s'	s'	$z \cup \{b'_i, l'_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\}) \cup w(H) \cup \{e^*\}$
5	r_{13}	$-$	s'	s'	$z \cup \{l'_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\}) \cup w(H) \cup \{e^*\}$
6	r_{14}	$-$	s''	s'	$z \cup \{b'_i, l'_i, l''_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\}) \cup w(H) \cup \{e^*\}$
7	r_{15}	r_{24}	s'''	s'	z'	$u(H) \cup v(H) \cup w(H) \cup \{l''_i, e^*\}$
8	$-$	r_{25}	s'''	s'	$z' \cup \{b''_i, l''_i\}$	$u(H) \cup v(H) \cup (w(H) \setminus \{b''_i\}) \cup \{e^*\}$
9	r_{16}	$-$	s'''	s'	$z' \cup \{l''_i\}$	$u(H) \cup v(H) \cup (w(H) \setminus \{b''_i\}) \cup \{e^*\}$
10	r_{18}	$-$	s	s'	$z' \cup \{b''_i, l''_i, l_j\}$	$u(H) \cup v(H) \cup (w(H) \setminus \{b''_i\}) \cup \{e^*\}$
11	r_{20}	$-$	s	s'	$z' \cup \{l_j\}$	$u(H) \cup v(H) \cup (w(H) \setminus \{b''_i\}) \cup \{e^*\}$
12	r_{21}	$-$	s	s'	$z' \cup \{b''_i, l_j, e\}$	$u(H) \cup v(H) \cup (w(H) \setminus \{b''_i\}) \cup \{e^*\}$
13	$-$	r_{26}	s	s	$z' \cup \{l_j\}$	$u(H) \cup v(H) \cup w(H) \cup \{e^*\}$

two steps, rules r_{13} and r_{14} are used one by one, one copy of object l''_i is sent into membrane 1, and the channel state between membrane 1 and the environment is changed from s' to s''. In what follows, we have two cases:

- There is at least one copy of object a_r in membrane 1 (corresponding to the fact that the number stored in register r is greater than 0). In this case, at step 7, rule r_{15} is enabled and applied, objects l'_i, a_r are sent to the environment (one copy of object a_r is consumed, simulating that the number stored in register r is decreased by one), changing the channel state between membrane 1 and the environment from s'' to s'''; simultaneously, objects b'_i, l''_i are sent into membrane 2 by using rule r_{24}. With the appearance of object l''_i in membrane 2, rule r_{25} is used, object b''_i is sent out of membrane 2, which will be sent to the environment at the next step by using rule r_{16}. Under the influence of channel state s''' between membrane 1 and the environment, instruction object l_j is sent into membrane 1 by using rule r_{18}, changing the channel state between membrane 1 and the environment to s. At step 11, by applying rule r_{20}, object b''_i is sent to the environment, and this object together with object e will be sent into membrane 1 at the next step (by using rule r_{21}). At step 13, rule r_{26} is used, object b''_i is sent back to membrane 2, and the channel state between membrane 2 and membrane 1 is changed from s' to s. In this case, one copy of object a_r is consumed in membrane 1, and the system starts to simulate the instruction l_j. Note that only when the channel state between membrane 2 and membrane 1 is changed to s, the follow-up instruction l_j can be simulated (see Table IV).
- There is no object a_r in membrane 1 (corresponding to the fact that the number stored in register r is 0). In this case, at step 7, only rule r_{24} can be used, objects b'_i, l''_i are sent into membrane 2 and objects b''_i, l''_i will be sent out of this membrane at the next step by using rule r_{25}. Rule r_{17} can be used when object b''_i appears in membrane 1; this object together with object l'_i will be sent to the environment at step 9. Under the control of channel state s'' between membrane 1 and the environment,

TABLE V

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION l_i : (SUB$(r), l_j, l_k$) WITH REGISTER r EMPTY, WHERE z IS A MULTISET OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$, AND $u(H), v(H), w(H)$ ARE MULTISETS WHICH CONTAIN EACH $b \in H, b' \in H, b'' \in H$ EXACTLY ONCE, RESPECTIVELY

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s_1	s_2	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	s	s	$z \cup \{l_i\}$	$u(H) \cup v(H) \cup w(H) \cup \{e^*\}$
1	$-$	r_{22}	s	s	z	$u(H) \cup v(H) \cup w(H) \cup \{l_i, e^*\}$
2	$-$	r_{23}	s	s'	$z \cup \{b_i', l_i\}$	$u(H) \cup (v(H) \setminus \{b_i'\}) \cup w(H) \cup \{e^*\}$
3	r_{11}	$-$	s	s'	z	$u(H) \cup (v(H) \setminus \{b_i'\}) \cup w(H) \cup \{e^*\}$
4	r_{12}	$-$	s'	s'	$z \cup \{b_i', l_i'\}$	$u(H) \cup (v(H) \setminus \{b_i'\}) \cup w(H) \cup \{e^*\}$
5	r_{13}	$-$	s'	s'	$z \cup \{l_i''\}$	$u(H) \cup (v(H) \setminus \{b_i'\}) \cup w(H) \cup \{e^*\}$
6	r_{14}	$-$	s''	s'	$z \cup \{b_i', l_i', l_i''\}$	$u(H) \cup (v(H) \setminus \{b_i'\}) \cup w(H) \cup \{e^*\}$
7	$-$	r_{24}	s''	s'	$z \cup \{l_i'\}$	$u(H) \cup v(H) \cup w(H) \cup \{l_i'', e^*\}$
8	$-$	r_{25}	s''	s'	$z \cup \{b_i'', l_i', l_i''\}$	$u(H) \cup v(H) \cup (w(H) \setminus \{b_i''\}) \cup \{e^*\}$
9	r_{17}	$-$	s''	s'	$z \cup \{l_i''\}$	$u(H) \cup v(H) \cup (w(H) \setminus \{b_i''\}) \cup \{e^*\}$
10	r_{19}	$-$	s	s'	$z \cup \{b_i'', l_i'', l_k\}$	$u(H) \cup v(H) \cup (w(H) \setminus \{b_i''\}) \cup \{e^*\}$
11	r_{20}	$-$	s	s'	$z \cup \{l_k\}$	$u(H) \cup v(H) \cup (w(H) \setminus \{b_i''\}) \cup \{e^*\}$
12	r_{21}	$-$	s	s'	$z \cup \{b_i'', l_k, e\}$	$u(H) \cup v(H) \cup (w(H) \setminus \{b_i''\}) \cup \{e^*\}$
13	$-$	r_{26}	s	s	$z \cup \{l_k\}$	$u(H) \cup v(H) \cup w(H) \cup \{e^*\}$

instruction object l_k is sent into membrane 1 by using rule r_{19}, changing the channel state between membrane 1 and the environment to s. Rule r_{20} can be used at step 11, thus objects b_i'', l_i'' are sent to the environment. At the next two steps, objects b_i'', e are sent into membrane 1 by using rule r_{21} and then these objects will be sent into membrane 2 by applying rule r_{26}, changing the channel state between membrane 2 and membrane 1 from s' to s. Hence, the system starts to simulate the instruction l_k. Note that only when the channel state between membrane 2 and membrane 1 is changed to s, the follow-up instruction l_k can be simulated (see Table V).

Hence, the SUB instruction of M is correctly simulated by system Π.

When the object l_h appears in membrane 1, no rule can be used in the system, and the computation halts. The number of copies of the object a_1 in membrane 1 corresponds to the result of the computation, hence $N(M)= N(\Pi)$.

Theorem 4.6: $NOP_2(state_2, sym_3) = NRE$.

Proof: We only prove the inclusion $NRE \subseteq NOP_2(state_2, sym_3)$. The reverse inclusion follows from the Church-Turing thesis.

Let us consider a register machine $M = (m, H, l_0, l_h, I)$. We construct the cell-like P system with channel states and symport/antiport rules Π to simulate the machine M.

$$\Pi = (\Gamma, K, \mathcal{E}, \mu, \mathcal{M}_1, \mathcal{M}_2, s, s, \mathcal{R}_1, \mathcal{R}_2, 1),$$

where

- $\Gamma = \{a_i \mid 1 \leq i \leq m\} \cup \{l, l', l'', l''', l^{iv}, l^v \mid l \in H\} \cup \{e\}$,
- $K = \{s, s'\}$,
- $\mathcal{E} = \{l^{iv}, l^v \mid l \in H\} \cup \{e\}$,
- $\mu = [\ [\ ]_2\ ]_1$,
- $\mathcal{M}_1 = \{l_0\} \cup \{l', l'' \mid l \in H\}$, $\mathcal{M}_2 = \{l''' \mid l \in H\}$, and the sets $\mathcal{R}_1$, $\mathcal{R}_2$ of rules are as follows:
  - For each ADD instruction l_i : (ADD$(r), l_j, l_k$) of M, we consider the following rules in $\mathcal{R}_1$:
    $r_1 : (s, (l_i l_i', out), s)$,
    $r_2 : (s, (l_i' l_j a_r, in), s)$,
    $r_3 : (s, (l_i' l_k a_r, in), s)$.

An ADD instruction l_i is simulated in two steps. At step 1, objects l_i, l_i' are sent to the environment by using rule r_1. At the next step, one of rules r_2 and r_3 is used non-deterministically. By applying rule r_2 (resp., r_3), objects l_i', l_j, a_r (resp., l_i', l_k, a_r) are sent into membrane 1 from the environment. Hence, one copy of object a_r is introduced into membrane 1 (simulating that the number stored in register r is increased by one), and the system starts to simulate the instruction with label l_j or l_k. So the instruction l_i of M is correctly simulated by Π.

- For each SUB instruction l_i : (SUB$(r), l_j, l_k$) of M,
- we consider the following rules in $\mathcal{R}_1$:
  - $r_4 : (s, (l_i l_i'', out), s)$,
    $r_5 : (s, (l_i'' l_i^{iv} l_i^v, in), s)$,
    $r_6 : (s, (l_i^{iv} a_r, out), s')$,
    $r_7 : (s', (l_i''' l_i^v, out), s')$,
    $r_8 : (s, (l_i''' l_i^{iv} l_i^v, out), s)$,
    $r_9 : (s', (l_i''' l_j e, in), s)$,
    $r_{10} : (s, (l_i''' l_k e, in), s)$;
    and we consider the following rules in $\mathcal{R}_2$:
  - $r_{11} : (s, (l_i^v, in), s')$,
    $r_{12} : (s', (l_i''' l_i^v, out), s')$,
    $r_{13} : (s', (l_i''' e, in), s)$

A SUB instruction l_i is simulated by system Π in the following way: At step 1, rule r_4 is applied, object l_i'' is sent to the environment, and this object together with objects l_i^{iv}, l_i^v will be sent into membrane 1 at the next step by using rule r_5. In what follows, we have two cases:

- There is at least one copy of object a_r in membrane 1 (corresponding to the fact that the number stored in register r is greater than 0). In this case, at step 3, rules r_6 and r_{11} are enabled; by applying rule r_6, objects l_i^{iv}, a_r are sent to the environment (one copy of object a_r is consumed, simulating that the number stored in register r is decreased by one), changing the channel state between membrane 1 and the environment to s'; by using rule r_{11}, object l_i^v

TABLE VI

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION $l_i : (\text{SUB}(r), l_j, l_k)$ WITH REGISTER r NOT EMPTY, WHERE z, z' ARE MULTISETS OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$, $z = z'a_r$, AND $u(H), v(H)$ ARE MULTISETS WHICH CONTAIN EACH $l', l'' \in H$ EXACTLY ONCE IN MEMBRANE 1, $w(H)$ IS MULTISET WHICH CONTAINS EACH $l''' \in H$ EXACTLY ONCE IN MEMBRANE 2

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s_1	s_2	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	s	s	$z \cup \{l_i\} \cup u(H) \cup v(H)$	$w(H) \cup \{e^*\}$
1	r_4	$-$	s	s	$z \cup u(H) \cup (v(H) \setminus \{l_i''\})$	$w(H) \cup \{e^*\}$
2	r_5	$-$	s	s	$z \cup u(H) \cup v(H) \cup \{l_i^{iv}, l_i^v\}$	$w(H) \cup \{e^*\}$
3	r_6	r_{11}	s'	s'	$z' \cup u(H) \cup v(H)$	$w(H) \cup \{l_i^v, e^*\}$
4	$-$	r_{12}	s'	s'	$z' \cup u(H) \cup v(H) \cup \{l_i''', l_i^v\}$	$(w(H) \setminus \{l_i'''\}) \cup \{e^*\}$
5	r_7	$-$	s'	s'	$z' \cup u(H) \cup v(H)$	$(w(H) \setminus \{l_i'''\}) \cup \{e^*\}$
6	r_9	$-$	s	s'	$z' \cup u(H) \cup v(H) \cup \{l_i''', l_j, e\}$	$(w(H) \setminus \{l_i'''\}) \cup \{e^*\}$
7	r_1 or r_4	r_{13}	s	s	$z' \cup (u(H) \setminus \{l_j'\}) \cup v(H)$ or $z' \cup u(H) \cup (v(H) \setminus \{l_j''\})$	$w(H) \cup \{e^*\}$

is sent into membrane 2 (the channel state between membrane 1 and membrane 2 is changed from s to s'), and this object together with object l_i''' will be sent out of this membrane at the next step (by using rule r_{12}). When object l_i''' appears in membrane 1, by using rule r_7, object l_i''' is sent to the environment, and instruction object l_j will be sent into membrane 1 at the next step by applying rule r_9; the channel state between membrane 1 and the environment is changed to s from s'. At step 7, by using rule r_{13}, object l_i''' and object e are sent back to membrane 2, and the channel state between membrane 1 and membrane 2 is changed to s. Note that during the simulation of a SUB instruction l_i, the next (ADD or SUB) instruction starts to be simulated at step 7, which does not affect the correctness of the simulation of the next instruction. In this case, one copy of object a_r is consumed in membrane 1, and the system starts to simulate the instruction l_j (see Table VI).

- There is no object a_r in membrane 1 (corresponding to the fact that the number stored in register r is 0). In this case, at step 3, only rule r_{11} can be used, object l_i^v is sent into membrane 2 (the channel state between membrane 1 and membrane 2 is changed from s to s'), and this object together with object l_i''' will be sent out of this membrane at the next step (by using rule r_{12}). At step 5, rule r_8 is enabled; by using this rule, object l_i''' is sent to the environment, and object l_k will be sent into membrane 1 at the next step (by using rule r_{10}). At step 7, by using rule r_{13}, object l_i''' and object e are sent back to membrane 2, and the channel state between membrane 1 and membrane 2 is changed to s. Note that during the simulation of a SUB instruction l_i, the next (ADD or SUB) instruction starts to be simulated at step 7, which does not affect the correctness of the simulation of the next instruction. Hence, the system starts to simulate the instruction l_k (see Table VII).

Hence, the SUB instruction of M is correctly simulated by system Π.

When the object l_h appears in membrane 1, no rule can be used in the system, and the computation halts. The number of copies of object a_1 in membrane 1 corresponds to the result of the computation, hence $N(M) = N(\Pi)$.

Theorem 4.7: $NOP_2(state_1, anti_3) = NRE$.

Proof: We only prove the inclusion $NRE \subseteq NOP_2(state_1, anti_3)$. The reverse inclusion follows from the Church-Turing thesis.

Let us consider a register machine $M = (m, H, l_0, l_h, I)$. We construct the cell-like P system with channel states and symport/antiport rules Π to simulate the machine M.

$$\Pi = (\Gamma, K, \mathcal{E}, \mu, \mathcal{M}_1, \mathcal{M}_2, s, s, \mathcal{R}_1, \mathcal{R}_2, 1),$$

where

- $\Gamma = \{a_i \mid 1 \leq i \leq m\} \cup \{l, l', l'', l''', l^{iv}, l^v, l^{vi} \mid l \in H\} \cup \{b\}$,
- $K = \{s\}$,
- $\mathcal{E} = \Gamma \setminus \{b\}$,
- $\mu = [\ [\ ]_2\ ]_1$,
- $\mathcal{M}_1 = \{l_0\}$, $\mathcal{M}_2 = \{b\}$, and the sets $\mathcal{R}_1$, $\mathcal{R}_2$ of rules are as follows:
  - For each ADD instruction $l_i : (\text{ADD}(r), l_j, l_k)$ of M, we consider the following rules in $\mathcal{R}_1$:$r_1 : (s, (l_i, out; l_j a_r, in), s)$,
    $r_2 : (s, (l_i, out; l_k a_r, in), s)$.

An ADD instruction l_i is simulated obviously. One of rules r_1 and r_2 is used non-deterministically. By applying rule r_1 (resp., r_2), object l_i in membrane 1 is exchanged with objects l_j, a_r (resp., l_k, a_r) from the environment. Hence, the system starts to simulate an instruction with label l_j or l_k. Clearly, the instruction l_i of M is correctly simulated by Π.

- For each SUB instruction $l_i : (\text{SUB}(r), l_j, l_k)$ of M,
- we consider the following rules in $\mathcal{R}_1$:

  $r_3 : (s, (l_i, out; l_i' l_i'', in), s)$,
  $r_4 : (s, (l_i' a_r, out; l_i''', in), s)$,
  $r_5 : (s, (b l_i''', out; l_i^{iv}, in), s)$,
  $r_6 : (s, (b l_i', out; l_i^{vi}, in), s)$,
  $r_7 : (s, (l_i'', out; b l_i^v, in), s)$,
  $r_8 : (s, (l_i^{iv} l_i^v, out; l_j, in), s)$,
  $r_9 : (s, (l_i^v l_i^{vi}, out; l_k, in), s)$;
  and we consider the following rules in $\mathcal{R}_2$:
  $r_{10} : (s, (b, out; l_i'', in), s)$,
  $r_{11} : (s, (l_i'', out; l_i^{iv}, in), s)$,
  $r_{12} : (s, (l_i'', out; l_i^{vi}, in), s)$,
  $r_{13} : (s, (l_i^{iv}, out; b, in), s)$,
  $r_{14} : (s, (l_i^{vi}, out; b, in), s)$

TABLE VII

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION $l_i : (\text{SUB}(r), l_j, l_k)$ WITH REGISTER r EMPTY, WHERE z IS A MULTISET OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$, AND $u(H), v(H)$ ARE MULTISETS WHICH CONTAIN EACH $l', l'' \in H$ EXACTLY ONCE IN MEMBRANE 1, $w(H)$ IS MULTISET WHICH CONTAINS EACH $l''' \in H$ EXACTLY ONCE IN MEMBRANE 2

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s_1	s_2	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	s	s	$z \cup \{l_i\} \cup u(H) \cup v(H)$	$w(H) \cup \{e^*\}$
1	r_4	$-$	s	s	$z \cup u(H) \cup (v(H) \setminus \{l_i''\})$	$w(H) \cup \{e^*\}$
2	r_5	$-$	s	s	$z \cup u(H) \cup v(H) \cup \{l_i^{iv}, l_i^v\}$	$w(H) \cup \{e^*\}$
3	$-$	r_{11}	s	s'	$z \cup u(H) \cup v(H) \cup \{l_i^{iv}\}$	$w(H) \cup \{l_i^v, e^*\}$
4	$-$	r_{12}	s	s'	$z \cup u(H) \cup v(H) \cup \{l_i''', l_i^{iv}, l_i^v\}$	$(w(H) \setminus \{l_i'''\}) \cup \{e^*\}$
5	r_8	$-$	s	s'	$z \cup u(H) \cup v(H)$	$(w(H) \setminus \{l_i'''\}) \cup \{e^*\}$
6	r_{10}	$-$	s	s'	$z \cup u(H) \cup v(H) \cup \{l_i''', l_k, e\}$	$(w(H) \setminus \{l_i'''\}) \cup \{e^*\}$
7	r_1 or r_4	r_{13}	s	s	$z \cup (u(H) \setminus \{l_k'\}) \cup v(H)$ or $z \cup u(H) \cup (v(H) \setminus \{l_k''\})$	$w(H) \cup \{e^*\}$

A SUB instruction l_i is simulated by system Π in the following way: At step 1, by using rule r_3, object l_i in membrane 1 is exchanged with objects l_i', l_i'' from the environment. In what follows, we have two cases:

- There is at least one copy of object a_r in membrane 1. In this case, at step 2, rules r_4 and r_{10} are enabled. By using r_4, objects l_i', a_r in membrane 1 are sent to the environment (one copy of object a_r is consumed in membrane 1), and object l_i''' is sent into membrane 1; by using rule r_{10}, object b in membrane 2 is exchanged with object l_i'' in membrane 1. At the next step, rule r_5 is enabled and used, objects b, l_i''' are sent to the environment, and object l_i^{iv} is sent into membrane 1, which will be exchanged with object l_i'' in membrane 2 at step 4 (by using rule r_{11}). At step 5, rule r_7 is used, object l_i'' is sent to the environment, and objects b, l_i^v are sent into membrane 1. At the next step, by using rule r_{13}, object b is sent back into membrane 2, and object l_i^{iv} is sent to membrane 1, this object together with object l_i^v will be sent to the environment, and object l_j is sent into membrane 1 by using rule r_8 at the next step. In this case, one copy of object a_r is consumed in membrane 1, and the system starts to simulate the instruction l_j (see Table VIII).
- There is no object a_r in membrane 1. In this case, at step 2, only rule r_{10} can be used, object b is sent into membrane 1, this object together with object l_i' will be sent to the environment at the next step; simultaneously, object l_i^{vi} is sent into membrane 1. When object l_i^{vi} appears in membrane 1, rule r_{12} is enabled and used, object l_i'' is sent into membrane 1, this object can be exchanged with objects b, l_i^v from the environment at the next step by applying rule r_{10}. At step 6, rule r_{14} is used, object l_i^{vi} is sent out of membrane 2, object b is sent back to membrane 2. At step 7, objects l_i^v, l_i^{vi} is sent to the environment, and object l_k is sent into membrane 1 from the environment by using rule r_9. Hence, the system starts to simulate the instruction l_k (see Table IX).

Hence, the SUB instruction of M is correctly simulated by system Π.

TABLE VIII

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION $l_i : (\text{SUB}(r), l_j, l_k)$ WITH REGISTER r NOT EMPTY, WHERE z, z' ARE MULTISETS OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$, $z = z' a_r$

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s_1	s_2	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	s	s	$z \cup \{l_i\}$	$\{b\}$
1	r_3	$-$	s	s	$z \cup \{l_i', l_i''\}$	$\{b\}$
2	r_4	r_{10}	s	s	$z' \cup \{l_i''', b\}$	$\{l_i''\}$
3	r_5	$-$	s	s	$z' \cup \{l_i^{iv}\}$	$\{l_i''\}$
4		r_{11}	s	s	$z' \cup \{l_i''\}$	$\{l_i^{iv}\}$
5	r_7	$-$	s	s	$z' \cup \{b, l_i^v\}$	$\{l_i^{iv}\}$
6	$-$	r_{13}	s	s	$z' \cup \{l_i^{iv}, l_i^v\}$	$\{b\}$
7	r_8	$-$	s	s	$z' \cup \{l_j\}$	$\{b\}$

TABLE IX

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION $l_i : (\text{SUB}(r), l_j, l_k)$ WITH REGISTER r EMPTY, WHERE z IS A MULTISET OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s	s	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	s	s	$z \cup \{l_i\}$	$\{b\}$
1	r_3	$-$	s	s	$z \cup \{l_i', l_i''\}$	$\{b\}$
2	$-$	r_{10}	s	s	$z \cup \{b, l_i'\}$	$\{l_i''\}$
3	r_6	$-$	s	s	$z \cup \{l_i^{vi}\}$	$\{l_i''\}$
4	$-$	r_{12}	s	s	$z \cup \{l_i''\}$	$\{l_i^{vi}\}$
5	r_7	$-$	s	s	$z \cup \{b, l_i^v\}$	$\{l_i^{vi}\}$
6	$-$	r_{14}	s	s	$z \cup \{l_i^v, l_i^{vi}\}$	$\{b\}$
7	r_9	$-$	s	s	$z \cup \{l_k\}$	$\{b\}$

When the object l_h appears in membrane 1, no rule can be used in the system, and the computation halts. The number of copies of object a_1 in membrane 1 corresponds to the result of the computation, hence $N(M) = N(\Pi)$.

Theorem 4.8: $NOP_2(state_3, sym_1, anti_2) = NRE$.

Proof: We only prove the inclusion $NRE \subseteq NOP_2(state_3, sym_1, anti_2)$. The reverse inclusion follows from the Church-Turing thesis.

Let us consider a register machine $M = (m, H, l_0, l_h, I)$. We construct the cell-like P system with channel states and symport/antiport rules Π to simulate the machine M.

$$\Pi = (\Gamma, K, \mathcal{E}, \mu, \mathcal{M}_1, \mathcal{M}_2, s, s, \mathcal{R}_1, \mathcal{R}_2, 1),$$

TABLE X

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF AN ADD INSTRUCTION $l_i : (\texttt{ADD}(r), l_j, l_k)$, WHERE z IS A MULTISET OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$, AND $u(H), v(H)$ ARE MULTISETS WHICH CONTAIN EACH $b \in H, b' \in H$ EXACTLY ONCE, RESPECTIVELY

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s_1	s_2	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	s	s	$z \cup \{l_i\}$	$u(H) \cup v(H)$
1	$-$	r_6	s	s	$z \cup \{b_i\}$	$\{l_i\} \cup (u(H) \setminus \{b_i\}) \cup v(H)$
2	r_1	r_7	s'	s'	$z \cup \{a_r, l_i\}$	$(u(H) \setminus \{b_i\}) \cup v(H)$
3	r_2	$-$	s	s'	$z \cup \{a_r, l_i'\}$	$(u(H) \setminus \{b_i\}) \cup v(H)$
4	r_3	$-$	s'	s'	$z \cup \{b_i, a_r, l_i'\}$	$(u(H) \setminus \{b_i\}) \cup v(H)$
5	r_4 or r_5	r_8	s	s	$z \cup \{a_r, l_j\}$ or $z \cup \{a_r, l_k\}$	$u(H) \cup v(H)$

where

- $\Gamma = \{a_i \mid 1 \le i \le m\} \cup \{b, b', l, l', l'' \mid b, l \in H\}$,
- $K = \{s, s', s''\}$,
- $\mathcal{E} = \{a_i \mid 1 \le i \le m\} \cup \{l, l', l'' \mid l \in H\}$,
- $\mu = [\,[\,\,]_2\,]_1$,
- $\mathcal{M}_1 = \{l_0\}$, $\mathcal{M}_2 = \{b, b' \mid b \in H\}$,

and the sets $\mathcal{R}_1$, $\mathcal{R}_2$ of rules are as follows:

- For each ADD instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$ of M,
- we consider the following rules in $\mathcal{R}_1$:
  $r_1 : (s, (b_i, out; a_r, in), s')$,
  $r_2 : (s', (l_i, out; l_i', in), s)$,
  $r_3 : (s, (b_i, in), s')$,
  $r_4 : (s', (l_i', out; l_j, in), s)$,
  $r_5 : (s', (l_i', out; l_k, in), s)$;
- and we consider the following rules in $\mathcal{R}_2$:
  $r_6 : (s, (b_i, out; l_i, in), s)$,
  $r_7 : (s, (l_i, out), s')$,
  $r_8 : (s', (b_i, in), s)$

An ADD instruction l_i is simulated by system Π in the following way: At step 1, rule r_6 is used, object b_i is sent out of membrane 2. Subsequently, it is sent to the environment and one copy of object a_r is sent into membrane 1 by using rule r_1 (changing the channel state between membrane 1 and the environment to s'); simultaneously, object l_i is sent into membrane 2, which will be sent out of this membrane at step 2 by applying rule r_7 (changing the channel state between membrane 2 and membrane 1 to s'). Under the control of channel state s' between membrane 1 and the environment, rule r_2 is used, object l_i' is sent into membrane 1, changing the channel state to s, so rule r_3 is enabled and applied, object b_i is sent into membrane 1, changing the channel state between membrane 1 and the environment from s to s'. At step 5, one of sets of rules $\{r_4, r_8\}$ and $\{r_5, r_8\}$ is used non-deterministically. By applying rule r_4 (resp., r_5), instruction object l_j (resp., l_k) is sent into membrane 1, changing the channel state between membrane 1 and the environment to s; by using rule r_8, object b_i is sent back to membrane 2, the channel state between membrane 2 and membrane 1 is changed from s' to s. Hence, the system starts to simulate the instruction with label l_j or l_k. So the instruction l_i of M is correctly simulated by Π (see Table X).

- For each SUB instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$ of M,
  - we consider the following rules in $\mathcal{R}_1$:
    $r_9 : (s, (b_i', out; l_i'', in), s)$,
    $r_{10} : (s, (a_r, out; b_i', in), s')$,
    $r_{11} : (s', (l_i, out), s')$,
    $r_{12} : (s, (l_i, out; b_i', in), s'')$,
    $r_{13} : (s', (l_i'', out; l_j, in), s)$,
    $r_{14} : (s'', (l_i'', out; l_k, in), s)$;
  - and we consider the following rules in $\mathcal{R}_2$:
    $r_{15} : (s, (b_i', out; l_i, in), s)$,
    $r_{16} : (s, (l_i, out; l_i'', in), s)$,
    $r_{17} : (s, (l_i'', out), s')$,
    $r_{18} : (s', (b_i', in), s)$

A SUB instruction l_i is simulated by system Π in the following way: At step 1, rule r_{15} is used, object l_i is sent into membrane 2, and object b_i' is sent to membrane 1, which will be exchanged with object l_i'' from the environment at the next step (by using rule r_9). In what follows, we have two cases:

- There is at least one copy of object a_r in membrane 1. In this case, at step 3, rule r_{10} is enabled and applied, one copy of object a_r is sent to the environment (one copy of object a_r in membrane 1 is consumed), and object b_i' is sent into membrane 1, changing the channel state between membrane 1 and the environment from s to s'; simultaneously, by applying rule r_{16}, object l_i is sent out of membrane 2, which will be sent to the environment at the next step; and object l_i'' is sent into membrane 2, which will be sent out of this membrane at step 4 by applying rule r_{17} (which changes the channel state between membrane 2 and membrane 1 to s'). Rule r_{13} can be used at step 5, instruction object l_j is sent into membrane 1, changing the channel state between membrane 1 and the environment to s; by applying rule r_{18}, object b_i' is sent back to membrane 2, and the channel state between membrane 2 and membrane 1 is changed to s. In this case, one copy of object a_r is consumed in membrane 1, and the system starts to simulate the instruction l_j (see Table XI).
- There is no object a_r in membrane 1. In this case, at step 3, only rule r_{16} can be used, object l_i is sent to membrane 1, which will be exchanged with object b_i' from the environment at the next step by using rule r_{12} (the channel state between membrane 1 and the environment is changed to s''); and object l_i'' is sent into membrane 2, which will be sent out of this membrane at step 4 by using rule r_{17} (which changes the channel state between membrane 2 and membrane 1 to s'). Note that at the first step for simulating a SUB instruction, there is no object b_i' in the environment, hence rule r_{12}

TABLE XI

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION l_i : (SUB(r), l_j, l_k) WITH REGISTER r NOT EMPTY, WHERE z, z' ARE MULTISETS OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$, $z = z'a_r$, AND $u(H), v(H)$ ARE MULTISETS WHICH CONTAIN EACH $b \in H, b' \in H$ EXACTLY ONCE, RESPECTIVELY

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s_1	s_2	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	s	s	$z \cup \{l_i\}$	$u(H) \cup v(H)$
1	$-$	r_{15}	s	s	$z \cup \{b'_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\}) \cup \{l_i\}$
2	r_9	$-$	s	s	$z \cup \{l''_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\}) \cup \{l_i\}$
3	r_{10}	r_{16}	s'	s	$z' \cup \{b'_i, l_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\}) \cup \{l''_i\}$
4	r_{11}	r_{17}	s'	s'	$z' \cup \{b'_i, l''_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\})$
5	r_{13}	r_{18}	s	s	$z' \cup \{l_j\}$	$u(H) \cup v(H)$

TABLE XII

THE APPLICATION OF RULES IN $\mathcal{R}_1$ AND $\mathcal{R}_2$, THE EVOLUTION OF CHANNEL STATES s_1 AND s_2, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN MEMBRANES 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION l_i : (SUB(r), l_j, l_k) WITH REGISTER r EMPTY, WHERE z IS A MULTISET OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$

Step	$\mathcal{R}_1$	$\mathcal{R}_2$	s_1	s_2	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	s	s	$z \cup \{l_i\}$	$u(H) \cup v(H)$
1	$-$	r_{15}	s	s	$z \cup \{b'_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\}) \cup \{l_i\}$
2	r_9	$-$	s	s	$z \cup \{l''_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\}) \cup \{l_i\}$
3	$-$	r_{16}	s	s	$z \cup \{l_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\}) \cup \{l''_i\}$
4	r_{12}	r_{17}	s''	s'	$z \cup \{b'_i, l''_i\}$	$u(H) \cup (v(H) \setminus \{b'_i\})$
5	r_{14}	r_{18}	s	s	$z \cup \{l_k\}$	$u(H) \cup v(H)$

cannot be used. At step 5, rule r_{14} is enabled and used, instruction object l_k is sent into membrane 1, changing the state between membrane 1 and the environment to s; by using rule r_{18}, object b'_i is sent back to membrane 2, changing the state between membrane 2 and membrane 1 to s. Hence, the system starts to simulate the instruction l_k (see Table XII).

Hence, the SUB instruction of M is correctly simulated by system Π.

When the object l_h appears in membrane 1, no rule can be used in the system, and the computation halts. The number of copies of object a_1 in membrane 1 corresponds to the result of the computation, hence $N(M) = N(\Pi)$.

V. CONCLUSIONS AND DISCUSSIONS

In this work, states associated with the communication channels are considered in the framework of cell-like P systems with symport/antiport rules, and the computational power of such P systems has been investigated. Specifically, we have proved that cell-like P systems with two states and only using uniport rules, or with any number of states and only using antiport rules of length two, are able to compute only finite sets of non-negative integers. We further proved that cell-like P systems with two membranes are Turing universal when having any number of states and only uniport rules, or four states and only symport rules of length two, or two states and only symport rules of length three, or one state and only antiport rules of length three, or three states and uniport rules and antiport rules of length two.

The number of cells, the number of states and the length of communication rules were taken for the descriptional complexity of cell-like P systems with channel states and symport/antiport rules. In Theorem 4.3, it was proved that cell-like P systems with one membrane are able to characterize the family of sets of positive integers generated by matrix grammars without appearance checking when having one state, and using antiport rules of length 4 or any number of states and using symport rules of length 1 and antiport rules of length 2. It is interesting to optimize the number of states or the length of communication rules used in Theorem 4.3.

The rules of the P systems constructed in Section IV are used in a sequential way. It remains open whether cell-like P systems with channel states and symport/antiport rules have the same computational power that use rules in other strategies, for instance, asynchronous (an enabled rule can evolve or not) [30]–[32], time-free (a P system that generates the same family of natural numbers, independently from the value assigned to the execution time of each rule) [33]–[35].

The descriptional complexity of P systems with symport/antiport rules with respect to the number of objects was considered in [18], [36], [37]. It is of interest to construct universal cell-like P systems with channel states and symport/antiport rules with a small number of objects.

The computational efficiency of P systems with symport/antiport rules and membrane division has been considered in [38]–[40], where membrane division provides a way to obtain exponential membranes in linear time. It would be interesting to introduce a variant of cell-like P systems with symport/antiport rules that can have exponential channel states in linear time, thus solving **NP**-complete problems by using the exponential channel states instead of the exponential membranes generated by membrane division.

The length of symport/antiport rules is one of the essential parameters for the computational power, which has already been investigated in tissue-like P systems with cell separation and cell-like P systems with symport/antiport rules and membrane separation [41]–[44]. It remains open whether cell-like P systems with channel states and membrane separation can solve **NP**-complete problems if rules are used in a sequential way. If the answer is negative, it is interesting to give a characterization of tractability as in [45], [46].

Recently, time-free solutions to **NP**-complete problems by various P systems have been investigated [47]–[50]. It is of interest to investigate whether we can construct cell-like P systems with channel states and membrane division (or membrane separation) to solve **NP**-complete problems in the context of time-freeness.

REFERENCES

[1] G. Păun, "Computing with membranes," *J. Comput. Syst. Sci.*, vol. 61, no. 1, pp. 108–143, Aug. 2000.

[2] A. Alhazov and R. Freund, "Variants of small universal P systems with catalysts," *Fundam. Informat.*, vol. 138, nos. 1–2, pp. 227–250, 2015.

[3] T. Song, J. Xu, and L. Pan, "On the universality and non-universality of spiking neural P systems with rules on synapses," *IEEE Trans. NanoBiosci.*, vol. 14, no. 8, pp. 960–966, Dec. 2015.

[4] T. Song and L. Pan, "Spiking neural P systems with request rules," *Neurocomputing*, vol. 193, pp. 193–200, Jun. 2016.

[5] X. Zhang, L. Pan, and A. Păun, "On the universality of axon P systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2816–2829, Nov. 2015.

[6] H. Peng, J. Wang, and M. J. Pérez-Jiménez, and A. Riscos-Núñez, "An unsupervised learning algorithm for membrane computing," *Inf. Sci.*, vol. 304, pp. 80–91, May 2015.

[7] H. Peng, J. Wang, and M. J. Pérez-Jiménez, H. Wang, J. Shao, and T. Wang, "Fuzzy reasoning spiking neural P system for fault diagnosis," *Inf. Sci.*, vol. 235, pp. 106–116, Jun. 2013.

[8] G. Zhang, M. Gheorghe, L. Pan, and M. J. Pérez-Jiménez, "Evolutionary membrane computing: A comprehensive survey and new results," *Inf. Sci.*, vol. 279, pp. 528–551, Sep. 2014.

[9] G. Zhang, H. Rong, F. Neri, and M. J. Pérez-Jiménez, "An optimization spiking neural P system for approximately solving combinatorial optimization problems," *Int. J. Neural Syst.*, vol. 24, no. 5, pp. 1–16, 2014.

[10] A. M. Colomer, A. Margalida, L. Valencia, and A. Palau, "Application of a computational model for complex fluvial ecosystems: The population dynamics of zebra mussel Dreissena polymorpha as a case study," *Ecol. Complex.*, vol. 20, pp. 116–126, Dec. 2014.

[11] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fund. Informat.*, vol. 71, nos. 2–3, pp. 279–308, Aug. 2006.

[12] C. Martín-Vide, J. Pazos, G. Păun, and A. Rodríguez-Patón, "Tissue P systems," *Theor. Comput. Sci.*, vol. 296, no. 2, pp. 295–326, Mar. 2003.

[13] G. Păun, *Membrane Computing: An Introduction*. Berlin, Germany: Springer, 2002.

[14] G. Păun, G. Rozenberg, and A. Salomaa, Eds., *The Oxford Handbook of Membrane Computing*. New York: Oxford Univ. Press, 2010.

[15] A. Păun and G. Păun, "The power of communication: P systems with symport/antiport," *New Generat. Comput.*, vol. 20, no. 3, pp. 295–305, Sep. 2002.

[16] A. Păun, G. Păun, and G. Rozenberg, "Computing by communication in networks of membranes," *Int. J. Found. Comput. Sci.*, vol. 13, no. 6, pp. 779–798, Dec. 2002.

[17] G. Ciobanu, L. Pan, Gh. Păun, and M. J. Pérez-Jiménez, "P systems with minimal parallelism," *Theor. Comput. Sci.*, vol. 378, no. 1, pp. 117–129, Jun. 2007.

[18] A. Alhazov and R. Freund, "P systems with one membrane and symport/antiport rules of five symbols are computationally complete," in *Proc. 3rd Brainstorming Week Membrane Comput.*, 2005, pp. 19–28.

[19] A. Alhazov and Y. Rogozhin, "Towards a characterization of P systems with minimal symport/antiport and two membranes," in *Membrane Computing*. Berlin, Germany: Springer, 2006, pp. 135–153.

[20] F. Bernardini and M. Gheorghe, "On the power of minimal symport/antiport," in *Proc. 3rd Workshop Membrane Comput.*, 2003, pp. 72–83.

[21] E. Csuhaj-Varjú, M. Margenstern, G. Vaszil, and S. Verlan, "On small universal antiport P systems," *Theor. Comput. Sci.*, vol. 372, nos. 2–3, pp. 152–164, Mar. 2007.

[22] R. Freund and M. Oswald, "P systems with activated/prohibited membrane channels," in *Membrane Computing*. Berlin, Germany: Springer, 2003, pp. 261–269.

[23] R. Freund and A. Păun, "Membrane systems with symport/antiport rules: Universality results," in *Membrane Computing*. Berlin, Germany: Springer, 2003, pp. 270–287.

[24] P. Frisco and H. J. Hoogeboom, "P systems with symport/antiport simulating counter automata," *Acta Informat.*, vol. 41, no. 2, pp. 145–170, Dec. 2004.

[25] R. Freund, G. Păun, and M. J. Pérez-Jiménez, "Tissue P systems with channel states," *Theor. Comput. Sci.*, vol. 330, no. 1, pp. 101–116, Jan. 2005.

[26] G. Rozenberg and A. Salomaa, Eds., *Handbook of Formal Languages*, vol. 3, Berlin, Germany: Springer, 1997.

[27] R. Freund, O. H. Ibarra, G. Păun, and H.-C. Yen, "Matrix languages, register machines, vector addition systems," in *Proc. 3rd Brainstorming Week Membrane Comput.*, 2005, pp. 155–168.

[28] I. Korec, "Small universal register machines," *Theor. Comput. Sci.*, vol. 168, no. 2, pp. 267–301, Nov. 1996.

[29] M. L. Minsky, *Computation: Finite and Infinite Machines*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1967.

[30] P. Frisco, G. Govan, and A. Leporati, "Asynchronous P systems with active membranes," *Theor. Comput. Sci.*, vol. 429, pp. 74–86, Apr. 2012.

[31] L. Pan, J. Wang, and H. J. Hoogeboom, "Limited asynchronous spiking neural P systems," *Fundam. Informat.*, vol. 110, nos. 1–4, pp. 271–293, 2011.

[32] T. Song, L. Pan, and G. Păun, "Asynchronous spiking neural P systems with local synchronization," *Inf. Sci.*, vol. 219, pp. 197–207, Jan. 2013.

[33] M. Cavaliere and D. Sburlan, "Time–independent P systems," in *Membrane Computing*. Berlin, Germany: Springer, 2005, pp. 239–258.

[34] M. Cavaliere and V. Deufemia, "Further results on time-free P systems," *Int. J. Found. Comput. Sci.*, vol. 17, no. 1, pp. 69–89, Feb. 2006.

[35] A. Alhazov, "Number of protons/bi-stable catalysts and membranes in P systems. Time-freeness," in *Membrane Computing*. Berlin, Germany: Springer, 2006, pp. 79–95.

[36] A. Alhazov, R. Freund, and M. Oswald, "Symbol/membrane complexity of P systems with symport/antiport rules," in *Membrane Computing*. Berlin, Germany: Springer, 2006, pp. 96–113.

[37] G. Păun, J. Pazos, M. J. Pérez-Jiménez, and A. Rodríguez-Patón, "Symport/antiport P systems with three objects are universal," *Fundam. Informat.*, vol. 64, nos. 1–4, pp. 353–367, Jul. 2004.

[38] L. F. Macías-Ramos, L. Valencia-Cabrera, B. Song, T. Song, L. Pan, and M. J. Pérez-Jiménez, "A P_Lingua based simulator for P systems with symport/antiport rules," *Fundam. Informat.*, vol. 139, no. 2, pp. 211–227, 2015.

[39] B. Song, M. J. Pérez-Jiménez, and L. Pan, "Efficient solutions to hard computational problems by P systems with symport/antiport rules and membrane division," *BioSystems*, vol. 130, pp. 51–58, Apr. 2015.

[40] L. V. Cabrera, B. Song, and L. F. Macías-Ramos, L. Pan, A. R. Núñez, and M. J. P. Jiménez, "Opened access minimal cooperation in P systems with symport/antiport: A complexity approach," in *Proc. 13th Brainstorming Week Membrane Comput.*, 2015, pp. 301–324.

[41] L. F. Macías-Ramos, B. Song, L. Valencia-Cabrera, L. Pan, and M. J. Pérez-Jiménez, "Membrane fission: A computational complexity perspective," *Complexity*, vol. 21, no. 6, pp. 321–334, Jul./Aug. 2016.

[42] L. Pan and M. J. Pérez-Jiménez, "Computational complexity of tissue-like P systems," *J. Complex.*, vol. 26, no. 3, pp. 296–315, Jun. 2010.

[43] L. F. Macías-Ramos, M. J. Pérez-Jiménez, A. Riscos-Núñez, and L. Valencia-Cabrera, "Membrane fission versus cell division: When membrane proliferation is not enough," *Theor. Comput. Sci.*, vol. 608, pp. 57–65, Dec. 2015.

[44] M. J. Pérez-Jiménez and P. Sosík, "An optimal frontier of the efficiency of tissue P systems with cell separation," *Fundam. Informat.*, vol. 138, nos. 1–2, pp. 45–60, 2015.

[45] R. Gutiérrez-Escudero, M. J. Pérez-Jiménez, and M. Rius-Font, "Characterizing tractability by tissue-like P systems," in *Membrane Computing*. Berlin, Germany: Springer, 2009, pp. 289–300.

[46] G. Mauri, M. J. Pérez-Jiménez, and C. Zandron, "On a Păun's conjecture in membrane systems," in *Bio-Inspired Modeling of Cognitive Tasks*. Berlin, Germany: Springer, 2007, pp. 180–192.

[47] T. Song, L. F. Macías-Ramos, L. Pan, and M. J. Pérez-Jiménez, "Time-free solution to SAT problem using P systems with active membranes," *Theor. Comput. Sci.*, vol. 529, pp. 61–68, Apr. 2014.

[48] B. Song and L. Pan, "Computational efficiency and universality of timed P systems with active membranes," *Theor. Comput. Sci.*, vol. 567, pp. 74–86, Feb. 2015.

[49] B. Song, M. J. Pérez-Jiménez, and L. Pan, "Computational efficiency and universality of timed P systems with membrane creation," *Soft Comput.*, vol. 19, no. 11, pp. 3043–3053, Nov. 2015.

[50] X. Liu, J. Suo, S. C. H. Leung, J. Liu, and X. Zeng, "The power of time-free tissue P systems: Attacking NP-complete problems," *Neurocomputing*, vol. 159, pp. 151–156, Jul. 2015.