酵素を用いた数値膜計算における基本演算およびソートの実現

(Enzymatic Numerical P Systems for basic operations and sorting)

Shohei Maeda and Akihiro Fujiwara
Graduate School of Computer Science and Systems Engineering
Kyushu Institute of Technology
Iizuka, Fukuoka, 820-8502, Japan

***Abstract*—Membrane computing, which is a computational model inspired by the structures and behaviors of living cells, has considerable attention as one of non-silicon based computing. In the present paper, we propose EN P systems for basic operations, and sorting.**

We first propose three EN P systems for computing three logic operations, OR, AND, and EX-OR functions. All of the EN P systems work in a constant number of steps.

Next, we propose two EN P systems that operates as a half adder and a full adder, and then, propose two EN P system for additions of two binary numbers and n binary numbers. We show the EN P systems for two additions work in $O(m)$ steps and $O(nm)$ steps, respectively.

Finally, we propose two EN P systems for sorting. We propose an EN P system for compare-and-exchange operation. Then, using the EN P system as sub-systems, we propose an EN P system for sorting n numbers, and show that the EN P system works in $O(n)$ steps.

I. Introduction

Membrane computing, which is a representative example of natural computing, is a computational model inspired by the structures and behaviors of living cells. In the initial study on membrane computing, a basic feature of the membrane computing was introduced by Păun[1] as a P system. In the P system, activities in living cells are considered as parallel computing. In [2], [3], algorithms of P system for logical function and additions have been proposed as basic operations.

As a derived model of the P system, A *Numerical P system*, which is inspired from structures of living cells and economics, has been introduced in [4] by Păun. Each region of numerical P system contains a number of numerical *variables* that are evolved according to a *program*. Each program consists of a *production function* and a *repartition protocol*. The production function calculates an output value from numerical variables of the same region, and the output value is distributed into the region and neighboring regions, which are outside and inside regions, by a repartition protocol. *Enzymatic Numerical P systems* [5] (EN P systems, for short) is also a model such that a number of variables, which is called *enzyme*, is used to promote evolution programs.

In the present paper, we propose EN P systems for logic operation, additions, and sorting. we first propose three EN P systems for computing OR, AND, and EX-OR functions. All of the proposed EN P system work in a constant number of steps by using O(1) variables, O(1) membranes, and a constant number of programs.

Second, we propose four EN P systems for additions. The first and second EN P systems operates as a half adder and a full adder, respectively. Both EN P systems work in a constant number of steps by using O(1) variables, O(1) membranes, and a constant number of programs. The third and fourth EN P systems are for addition of binary numbers of m bits. The third EN P system computes addition of two binary numbers of m bits, and works in $O(m)$ steps by using $O(m)$ kinds of variables, $O(m)$ membranes, and programs of size $O(m)$. The fourth P system computes addition of n binary numbers of m bits by using the above EN P system as a sub-system. The EN P system for the addition works in $O(nm)$ steps by using $O(nm)$ kinds of variables, $O(nm)$ membranes, and programs of size $O(nm)$.

Finally, we propose two EN P systems for sorting. The first EN P system executes a compare-and-exchange operation for two integers in constant number of steps by using $O(1)$ variables, $O(1)$ membrane, and a constant number of programs. The second EN P system executes sorting for n integers. The EN P system is based on an idea of the odd-even sort [6], and works in $O(n)$ steps by using $O(n)$ types of variables, $O(n)$ size of membrane, and programs of size $O(n)$.

II. Preliminaries

A. Enzymatic numerical P systems

In the P system, a membrane is a computing cell, in which independent computation is executed, and may contain objects and other membranes. In other words, the membranes form nested structures. In the present paper, each membrane is denoted by using a pair of square brackets, and the number on the right-hand side of each right-hand bracket denotes the label of the corresponding membrane.

For example, $[\ [\ ]_2\ [\ ]_3\ ]_1$ and Fig. 1 denote the same membrane structure that consists of three membranes. The membrane labeled 1 contains two membranes labeled 2 and 3.

We now describe details of the numerical P system. The Numerical P system and the sets used in the system are defined as follows.

$$\Pi_{NP} = (m, H, \mu, (V_m, P_m, V_m(0)), \ldots, (V_m, P_m, V_m(0)), V_o))$$

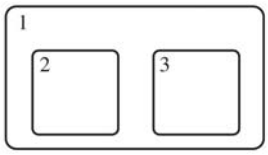

Fig. 1. An example of membrane structure

m: m is the number of membranes in the system.

H: H is a set of labels for membranes. In the present paper, we always use consecutive integer labels such that $H = \{1, 2, \cdots, m\}$. (We assume that a membrane labeled 1, which is called the *skin* membrane, is the outermost membrane, i.e., the *skin* membrane contains all of the other membranes.)

μ: μ is membrane structure that consists of m membranes. Each membrane in the structure is labeled with an element in H.

V_i: V_i is a set of numerical variables in the membrane labeled i.

P_i: P_i is a set of programs in the membrane labeled i.

$V_i(0)$: $V_i(0)$ is a set of initial values of variables in the membrane labeled i.

V_o: V_o is a set of output variables.

The elements of V_i are written in the form $x_{j,i}(t)$, where j is an integer running from 1 to $|V_i|$ at step t. In this paper, V_o is included in the outermost region of the system.

We next formally define a k-th program $pr_{k,i} \in P_i$, which is a program applied in membrane labeled i, as follows.

$$\begin{aligned} pr_{k,i} \quad = \quad & \{F_{k,i}(y_{1,i}, \ldots, y_{l_k,i}) \\ & \rightarrow c_{k,1}|v_1 + c_{k,2}|v_2 + \ldots c_{k,n_i}|v_{n_i})\} \end{aligned}$$

In the above expression, $F_{k,i}(y_{1,i}, \ldots, y_{l_k,i})$ is called a *production function* and computed as arguments $y_{1,i}, \ldots, y_{l_k,i} \subset V_i$. $c_{k,1}|v_1 + c_{k,2}|v_2 + \ldots c_{k,n_i}|v_{n_i}$ is called a *repartition protocol*. $\{v_1, \ldots, v_{n_i}\}$ is a set of variables in the region and in neighboring regions, which are outside and inside regions, and a repartition protocol allocates an output of $F_{k,i}(y_{1,i}, \ldots, y_{l_k,i})$ to the variables according to coefficients $\{c_{k,1}, \ldots, c_{,n_i}\} \subset \mathbf{N}$.

The numerical P system is executed repeatedly according to the following procedure.

(1) Select an applicable program in each region.
(2) Calculate the production functions of the selected programs.
(3) Change the variables used in production functions or repartition protocols to zero.
(4) Allocate output values of (2) according to repartition protocols.

We next describe details of the enzymatic numerical P system. The EN P system is a model such that a number of variables, which are called *enzyme*, in the numerical P systems is used to promote evolution programs. The enzyme in the membrane labeled i is denoted by e_i.

TABLE I
OR FUNCTION

Input 1: $x_{1,1}$	Input 2: $x_{2,1}$	Output: x_{out}
-1	-1	-1
-1	-2	-2
-2	-1	-2
-2	-2	-2

We formally define EN P system Π_{ENP} as follows.

$$\Pi_{ENP} = (m, H, \mu, (V_1, P_1, V_1(0)), \ldots, (V_m, P_m, V_m(0)), V_o)$$

The above definition is the same as the numerical P system because an enzyme is just a numerical variable of the numerical P system. The enzymes e_i are written in the form $e_{j,i}(t)$, where j is an integer running from 1 and $|V_i|$ at step t in the same way as $x_{j,i}(t)$.

We next formally define a k-th program $pr_{k,i}$, which is a program applied in membrane labeled i as follows.

$$\begin{aligned} pr_{k,i} \quad = \quad & \{F_{k,i}(y_{1,i}, \ldots, y_{l_k,i})|_{e_{j,i}} \\ & \rightarrow c_{k,1}|v_1 + c_{k,2}|v_2 + \ldots + c_{k,n_i}|v_{n_i}\} \end{aligned}$$

In case that $e_{j,i}(t) > min\{y_{1,i}(t), \ldots, y_{l_k,i}(t)\}$ at step t, the enzyme works as catalyst, and then the program is applicable.

In this paper, the EN P system is executed repeatedly according to the following procedure.

(1) Select applicable programs in which enzymes are used.
(2) Select an applicable program in each region.
(3) Calculate the production functions of the selected programs.
(4) Change the variables that used in the production functions or repartition protocols to zero.
(5) Allocate output values of (3) by repartition protocols.

The above EN P system has two features, which are maximal parallelism and non-determinism. Maximal parallelism means that all applicable programs are applied in parallel. (However, only one program is applied in a membrane.) On the other hand, non-determinism means that applicable programs are non-deterministically chosen in case that there are several possibilities of the applicable programs.

III. LOGIC FUNCTIONS

A. Input and output for OR function

In this section, we first consider simple EN P system that computes two input OR functions. An input of the function is two binary bits $x_{1,1}$, $x_{2,1}$, and we assume that -1 and -2 denotes general binary numbers 0 (FALSE) and 1 (TRUE), due to computation on the EN P system. Then, outputs of the function on the EN P system are defined in Table I.

B. An overview of EN P system

We now describe an overview of an EN P system for the OR function. The EN P system consists of inner and outer membranes, i.e. membrane structure of the EN P system is $[\ [\ ]_2\ ]_1$.

The computation in the EN P system mainly consists of the following 3 steps.

Step 1: Compute a sum s of two input value. (s is -2 in case of the output is 0, otherwise, the sum is -3 or -4.)

Step 2: Compute two values from the sum of Step 1. The first value is $-s+3$ such that the value is 0 if and only if the output is FALSE. The second value is $s+2$ such that the value is negative if and only if the output is TRUE.

Step 3: Send out an output value depending on the result of Step 2. In case that the result of the first value in Step 2 is a negative value, a value -1 is sent out as FALSE. On the other hand, a value -2 is sent of as as TRUE in case that the result of the second value in Step 2 is negative.

C. Details of EN P system

We now show details of each step of the EN P system for the OR function.

Step 1 is executed by applying the following program $pr_{1,1}$ to two input values $x_{1,1}$ and $x_{2,1}$.
(A programs for the outer membrane)

$$pr_{1,1} \quad = \quad \{2(x_{1,1}+x_{2,1})|_{e_{1,1}} \rightarrow 1|x_{3,1}+1|x_{1,2}\}$$

In this step, a sum s of two input value is computed from two input values, and the sum is copied to two variable $x_{3,1}$ and $x_{1,2}$. ($x_{3,1}$ is a variable in the outer membrane, and $x_{1,2}$ is a variable in the inner membrane.)

Step 2 is executed by applying the following two programs $pr_{2,1}$ and $pr_{1,2}$ in the outer and inner membranes, respectively.
(A program for the outer membrane)

$$pr_{2,1} \quad = \quad \{-x_{3,1}-3|_{e_{1,1}} \rightarrow 1|x_{4,1}\}$$

(A program for the inner membrane)

$$pr_{1,2} \quad = \quad \{x_{1,2}+2|_{e_{1,2}} \rightarrow 1|x_{5,1}\}$$

In this step, two values are computed in the outer and inner membranes in parallel. The first value $-s-3$ is computed according to $pr_{2,1}$ in the outer membrane, and the value is copied to $x_{4,1}$ in the same membrane. The second value $s+2$ is computed according to $pr_{1,2}$ in the inner membrane, and the value is copied to $x_{5,1}$ in the outer membrane.

Step 3 is executed by applying one of the following two programs $pr_{3,1}$ and $pr_{4,1}$ according to two values $x_{4,1}$ and $x_{5,1}$.
(A program for the outer membrane)

$$\begin{aligned} pr_{3,1} \quad &= \quad \{x_{4,1}|_{e_{1,1}} \rightarrow 1|x_{out}\} \\ pr_{4,1} \quad &= \quad \{-2+0x_{5,1}|_{e_{1,1}} \rightarrow 1|x_{out}\} \end{aligned}$$

In this step, a value x_{out} is determined by two values $x_{4,1}$ and $x_{5,1}$. In case that $x_{4,1}$ is negative, an output is FALSE, and a variable x_{out} is set to -1. Otherwise, $x_{5,1}$ is negative, and a variable x_{out} is set to -2 as TRUE.

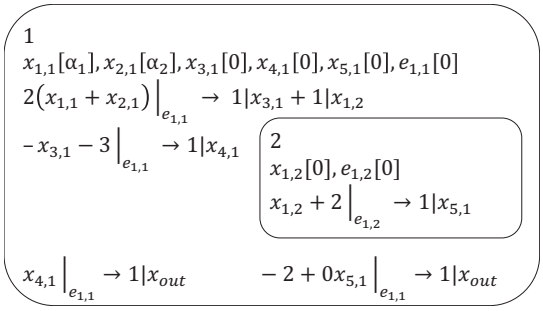

Fig. 2. EN P system for the OR function

TABLE II
AN EXECUTION OF Π_{OR}

	Initial value	Step 1	Step 2	Step 3
Input 1: $x_{1,1}$	-1	0	0	0
Input 2: $x_{2,1}$	-2	0	0	0
Variable $x_{3,1}$	0	-3	0	0
Variable $x_{4,1}$	0	0	0	0
Variable $x_{5,1}$	0	0	-1	0
Enzyme $e_{1,1}$	0	0	0	0
Variable $x_{1,2}$	0	-3	0	0
Enzyme $e_{1,2}$	0	0	0	0
Output x_{out}	-	-	-	-2

We summarize the EN P system for the OR function Π_{OR} and the sets used in the system as follows.

$$\Pi_{OR} = (2, H, \mu, (V_1, P_1, V_1(0)), (V_2, P_2, V_2(0)), V_o)$$

- $H = \{1, 2\}$
- $\mu = [\ [\ ]_2]_1$
- $V_1 = \{x_{1,1},\ x_{2,1},\ x_{3,1},\ x_{4,1},\ x_{5,1},\ e_{1,1}\}$
- $P_1 = pr_{1,1} \cup pr_{2,1} \cup pr_{3,1} \cup pr_{4,1}$
- $V_1(0) = (\alpha_1,\ \alpha_2,\ 0,\ 0,\ 0,\ 0)$
- $V_2 = \{x_{1,2},\ e_{1,2}\}$
- $P_2 = pr_{1,2}$
- $V_2(0) = (0,\ 0)$
- $V_o = \{x_{out}\}$

The above system Π_{OR} is also illustrated in Fig. 2.

In addition, Table II shows an example of variables in execution on Π_{OR} in case of $x_{1,1} = -1$ and $x_{2,1} = -2$.

D. Complexity and other logic operations

Since the number of types of variable in the EN P system Π_{OR} is $O(1)$, and $O(1)$ kinds of programs are used, we obtain the following theorem for Π_{OR}.

Theorem 1: The EN P system Π_{OR}, which computes OR function, works in $O(1)$ steps by using $O(1)$ types of variables, a constant number of membranes, and programs of size $O(1)$. □

We can also propose two EN P systems Π_{AND} and $\Pi_{\text{EX-OR}}$ for other two logic operations, AND and EX-OR, using a similar idea, and obtain the following theorems for Π_{AND} and $\Pi_{\text{EX-OR}}$. (We omit details of the EN P systems due to space limitation.)

Theorem 2: The EN P systems Π_{AND}, which computes AND function, works in $O(1)$ steps by using $O(1)$ types of

TABLE III
HALF ADDER

x_{in_1}	x_{in_2}	$x_{1,1}$	Carry: x_{carry}	Sum: x_{sum}
-1	-1	-2	-1	-1
-1	-2	-3	-1	-2
-2	-1	-3	-1	-2
-2	-2	-4	-2	-1

variables, a constant number of membranes, and programs of size $O(1)$. □

Theorem 3: The EN P systems $\Pi_{EX\text{-}OR}$, which computes EX-OR function, works in $O(1)$ steps by using $O(1)$ types of variables, a constant number of membranes, and programs of size $O(1)$. □

IV. ADDITION

A. Input and output for a half adder

In this section, we first consider an EN P system that executes computation of a half adder of two inputs. We assume that -1 and -2 denotes general binary numbers 0 (FALSE) and 1 (TRUE), due to computation on the EN P system. In addition, we assume that an input of the function is $x_{1,1}$, which is a sum of two input binary bits x_{in_1}, x_{in_2} because the sum is easily computed on the EN P system. Outputs of the function on the EN P system are a carry bit and a sum, which are defined in Table III.

B. An overview of EN P system

We now describe an overview of an EN P system for the half adder. The EN P system consists of a main membrane, and the main membrane includes two EN P systems Π_{AND} and $\Pi_{EX\text{-}OR}$, which are described in the previous section, as sub-system. We define that an output of $\Pi_{EX\text{-}OR}$ is $x_{2,1}$ and an output of Π_{AND} is $x_{3,1}$.

The computation on the EN P system mainly consists of the following 2 steps.

Step 1: Move input values in the main membrane into Π_{AND} and $\Pi_{EX\text{-}OR}$.

Step 2: Compute a carry bit and a sum in Π_{AND} and $\Pi_{EX\text{-}OR}$, respectively, and send out the two values to the main membrane.

Step 3: Send out an output value of Π_{AND} as a carry bit of the half adder, and send out an output value of $\Pi_{EX\text{-}OR}$ as a sum of the half adder.

C. Details of EN P system

We now show details of each step of the EN P system for the half adder.

Step 1 is executed by applying the following program $pr_{1,1}$ to input values $x_{1,1}$.
(A programs for membrane 1)

$$pr_{1,1} \quad = \quad \{2x_{1,1}|_{e_{1,1}} \rightarrow 1|x_{1,EX\text{-}OR} + 1|x_{1,AND}\}$$

In this step, an input variable $x_{1,1}$ is copied to a variable $x_{1,EX\text{-}OR}$ in $\Pi_{EX\text{-}OR}$ and a variable $x_{1,AND}$ in Π_{AND}.

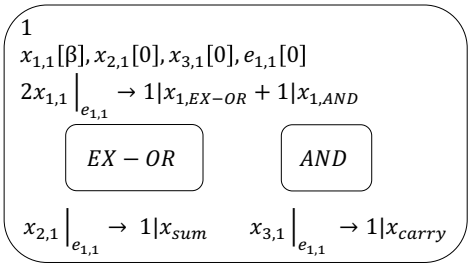

Fig. 3. EN P system for the half adder

In Step 2, two values are computed from EN P systems $\Pi_{EX\text{-}OR}$ and Π_{AND}, and the two output value are sent out from the two sub-systems to $x_{2,1}$ and $x_{3,1}$, respectively.

Step 3 is executed by applying one of the two programs $pr_{2,1}$ or $pr_{3,1}$.
(A program for membrane 1)

$$\begin{aligned} pr_{2,1} &= \{x_{2,1}|_{e_{1,1}} \rightarrow 1|x_{sum}\} \\ pr_{3,1} &= \{x_{3,h}|_{e_{1,h}} \rightarrow 1|x_{carry}\} \end{aligned}$$

After application of the above programs, the output values x_{carry} and x_{sum} are sent out as a carry and a sum of half adder, respectively.

We summarize the EN P system for the half adder Π_{HADD} and the sets used in the system as follows.

$$\Pi_{HADD} = (6, H, \mu, (V_1, P_1, V_1(0)), VP_{AND}, VP_{EX\text{-}OR}, V_{out})$$

- $H = \{h, H_{EX\text{-}OR}, H_{AND}\}$
  - $H_{EX\text{-}OR}$: a label of $\Pi_{EX\text{-}OR}$
  - H_{AND}: a label of Π_{AND}
- $\mu = [\ \mu_{EX\text{-}OR}\ \mu_{AND}\ ]_h$
  - $\mu_{EX\text{-}OR}$: $\mu_{EX\text{-}OR}$ is the membrane structure of Π_{EXOR}
  - μ_{AND}: μ_{AND} is the membrane structure of Π_{AND}
- $Var_1 = \{x_{1,1}, x_{2,1}, x_{3,1}, e_{1,1}\}$
- $Par_1 = pr_{1,1} \cup pr_{2,1} \cup pr_{3,1}$
- $Var_1(0) = (\beta, 0, 0, 0)$
- VP_{AND}: VP_{AND} is the set of $(V_i, P_i, V_i(0))$ in Π_{AND}.
- $VP_{EX\text{-}OR}$: $VP_{EX\text{-}OR}$ is the set of $(V_i, P_i, V_i(0))$ in $\Pi_{EX\text{-}OR}$.
- $V_{out} = \{x_{sum}, x_{carry}\}$

The above system Π_{HADD} is also illustrated in Fig. 3.

In addition, Table IV shows an example of variables in execution on Π_{HADD} in case of $x_{1,1} = -3$. In the example, we assume that program $pr_{2,1}$ is applied before application of program $pr_{3,1}$.

D. Complexity and other operations

Since the number of types of variable in the EN P system Π_{HADD} is $O(1)$, and $O(1)$ kinds of programs are used, we obtain the following theorem for Π_{HADD}.

Theorem 4: The EN P system Π_{HADD}, which executes computation of a half adder of two inputs, works in $O(1)$

TABLE IV
AN EXECUTION OF Π_{HADD}

	Initial value	Step 1	$\cdots$	Step 4	Step 5	Step 6
Input: $x_{1,1}$	-3	0		0	0	0
Variable $x_{2,1}$	0	0		-2	0	0
Variable $x_{3,1}$	0	0		-1	-1	0
Enzyme $e_{1,1}$	0	0		0	0	0
Variable x_{sum}	-	-		-	-2	-2
Output x_{carry}	-	-		-	-	-1

steps by using $O(1)$ types of variables, a constant number of membranes, and programs of size $O(1)$. □

We can also propose three EN P systems Π_{FADD}, Π_{2ADD} and Π_{nADD} for other three operations, which are a full adder, addition of two binary numbers of m bits, and addition of n binary numbers of m bits, using a similar idea. We obtain the following theorems for Π_{FADD}, Π_{2ADD}, and Π_{nADD}. (We omit details of the EN P systems due to space limitation.)

Theorem 5: The EN P system Π_{FADD}, which executes computation of a full adder, works in $O(1)$ steps by using $O(1)$ types of variables, a constant number of membranes, and programs of size $O(1)$. □

Theorem 6: The EN P system Π_{2ADD}, which computes addition of two binary numbers of m bits, works in $O(m)$ steps by using $O(m)$ types of variables, $O(m)$ kinds of membranes, and programs of size $O(m)$. □

Theorem 7: The EN P system Π_{nADD}, which computes addition of n binary numbers of m bits, works in $O(nm)$ steps by using $O(nm)$ types of variables, $O(nm)$ kinds of membranes, and programs of size $O(nm)$. □

V. COMPARE-AND-EXCHANGE AND SORTING

A. Input and output for compare-and-exchange

In this section, we first consider an EN P system that executes compare-and-exchange operation for two input values. Inputs are two negative integers, $x_{1,1}$, $x_{2,1}$, and we also assume that $x_{1,1}$ and $x_{2,1}$ are values between -1 to $n_{\min -1}+1$, where $n_{\min}$ is the minimum values on the EN P system. (The assumption is considered due to computation on the EN P system.)

The result of the computation is outputted to x_{large} and x_{small}. x_{large} and x_{small} are larger and smaller values of the inputs, respectively.

B. An overview of EN P system

We now describe an overview of an EN P system for the compare-and-exchange. The membrane structure of the EN P system is [[[ ]$_3$]$_2$ [ ]$_4$]$_1$.

The computation on the EN P system mainly consists of the following 5 steps.

Step 1: Compute $x_{1,1} - 0.1$.

Step 2: Copy the value of Step 1 to variable v_1 and enzyme e_1 and copy the inputs $x_{2,1}$ to variable v_2 and enzyme e_2.

Step 3: Compare v_1 and e_2, and compare v_2 and e_1.

Step 4: Send out a smaller value according to a result of Step 3. In case that the result of the first comparison in Step 3 is $v_1 < e_2$, send out v_1 as a smaller value, otherwise, send out v_2 as a small value.

Step 5: Send out a larger value according to a result of Step 3. In case that the result of the first comparison in Step 3 is $v_1 < e_2$, send out e_2 as a larger value, otherwise send out e_1 as a larger value.

C. Details of EN P system

We now show details of each step of the EN P system for the compare-and-exchange.

Step 1 is executed by applying a program $pr_{1,1}$ to an input value $x_{1,1}$.
(A programs for membrane 1)

$$pr_{1,1} = \{4(x_{1,1} - 0.1)|_{e_{1,1}} \rightarrow 1|x_{3,1} + 1|x_{5,1} + 1|x_{1,2} + 1|x_{3,2}\}$$

In this step, $x_{1,1} - 0.1$ is computed, and the result is copied to four variable $x_{3,1}$, $x_{5,1}$, $x_{1,2}$, and $x_{3,2}$. ($x_{3,1}$ and $x_{5,1}$ are variables in membrane 1, and $x_{1,2}$ and $x_{3,2}$ are variables in membrane 2.)

Step 2 is executed by applying the following two programs $pr_{2,1}$ and $pr_{1,2}$ in membrane 1 and membrane 2, respectively.
(A program for membrane 1)

$$pr_{2,1} = \{4x_{2,1}|_{e_{1,1}} \rightarrow 1|x_{4,1} + 1|x_{6,1} + 1|x_{2,2} + 1|x_{4,2}\}$$

(A program for membrane 2)

$$pr_{1,2} = \{x_{5,2} - 0.1 + 0x_{1,2}|_{e_{1,2}} \rightarrow 1|x_{5,2}\}$$

In this step, input $x_{2,1}$ is copied to four variable $x_{4,1}$, $x_{6,1}$, $x_{2,2}$, and $x_{4,2}$ by $pr_{2,1}$. ($x_{4,1}$ and $x_{6,1}$ are variables in membrane 1, and $x_{2,2}$ and $x_{4,2}$ are variables in membrane 2.) A formula $x_{5,2} - 0.1$ is computed by program $pr_{1,2}$, and the result is copied to $x_{5,2}$.

Step 3 is executed by applying the following program $pr_{2,2}$.
(A program for membrane 2)

$$pr_{2,2} = \{x_{5,2} - 0.1 + 0x_{2,2}|_{e_{1,2}} \rightarrow 1|x_{5,2}\}$$

In this step, $x_{5,2} - 0.1$ is computed by program $pr_{2,2}$, and the result is copied to $x_{5,2}$. Then, $x_{5,2} = -0.2$, and the value denotes a finish of comparison of the two values $x_{1,1}$ and $x_{2,1}$.

In addition, a result of the comparison is set to an enzyme by applying the following program $pr_{3,2}$ and $pr_{4,2}$.
(A program for membrane 2)

$$pr_{3,2} = \{x_{5,2}|_{e_{2,2}} \rightarrow 1|e_{3,2}\}$$
$$pr_{4,2} = \{2x_{3,2}|_{e_{3,2}} \rightarrow 1|e_{2,1} + 1|x_{1,3}\}$$

In this step, $x_{5.2}$ is copied to $e_{3,2}$ by $pr_{3,2}$, and then, $x_{3.2}$ is copied to $e_{2,1}$ and $x_{1,3}$ by $pr_{4,2}$.

Step 4 is executed by applying the following programs $pr_{3,1}$, $pr_{5,1}$, $pr_{5,2}$, and $pr_{1,3}$. (At first, programs $pr_{3,1}$ and $pr_{5,2}$ are applied, and then program $pr_{1,3}$ is applied.)
(A program for membrane 1)

$$pr_{3,1} = \{2(x_{3,1} + 0.1 + 0x_{4,1})|_{e_{2,1}} \rightarrow 1|x_{large} + 1|x_{1,4}\}$$

(A program for membrane 2)

$$pr_{5,2} = \{x_{4,2}|_{e_{3,2}} \rightarrow 1|e_{3,1}\}$$

(A program for membrane 3)

$$pr_{1,3} = \{n_{min-1} + 0x_{1,3}|_{e_{1,3}} \rightarrow 1|e_{3,2}\}$$

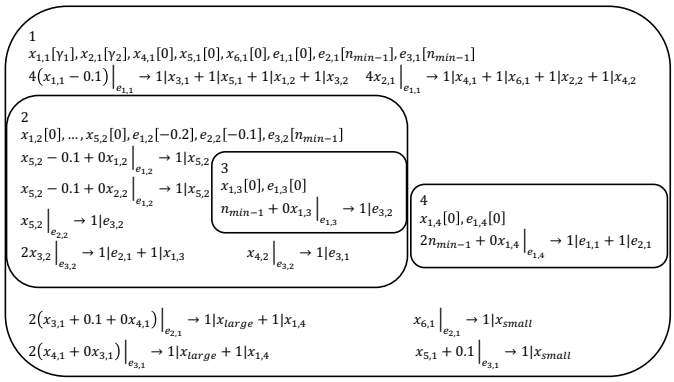

Fig. 4. EN P system for the compare and exchange

In this step, a larger value $x_{1,1}$ is sent out to x_{large} by $pr_{3,1}$, and the value is copied to $x_{1,4}$. A value $x_{4,2}$ is also copied to $e_{3,1}$ by $pr_{5,2}$. In addition, $e_{3,2}$ is reset to n_{min-1} by $pr_{1,3}$.

Finally, Step 5 is executed by applying the following two programs, $pr_{5,1}$ and $pr_{1,4}$.

(A program for membrane 1)

$$pr_{5,1} \quad = \quad \{x_{5,1}+0.1|_{e_{3,1}} \to 1|x_{small}\}$$

(A program for membrane 4)

$$pr_{1,4} \quad = \quad \{2n_{min-1}+0x_{1,4}|_{e_{1,4}} \to 1|e_{1,1}+1|e_{2,1}\}$$

In this step, a smaller value $x_{2,1}$ is sent out to x_{small} by $pr_{5,1}$, and $e_{1,1}$ and $e_{2,1}$ are reset to n_{min-1} by $pr_{5,2}$.

We summarize the EN P system for the compare-and-exchange function Π_{CMPEX} and the sets used in the system as follows.

$$\Pi_{CMPEX} \quad = \quad (4, H, \mu, (V_1, P_1, V_1(0)), (V_2, P_2, V_2(0)), (V_3, P_3, V_3(0)), (V_4, P_4, V_4(0)), V_{out})$$

- $H = \{1, 2, 3, 4\}$
- $\mu = [\ [\ [\ ]_3\ ]_2\ [\ ]_4\ ]_1$
- $V_1 = \{x_{1,1},\ x_{2,1},\ x_{3,1},\ x_{4,1},\ x_{5,1},\ x_{6,1},\ e_{1,1},\ e_{2,1},\ e_{3,1}\}$
- $P_1 = pr_{1,1} \cup pr_{2,1}$
- $V_1(0) = (\gamma_1,\ \gamma_2,\ 0,\ 0,\ 0,\ 0,\ 0,\ n_{min-1},\ n_{min-1})$
- $V_2 = \{x_{1,2},\ x_{2,2},\ x_{3,2},\ x_{4,2},\ x_{5,2},\ e_{1,2},\ e_{2,2},\ e_{3,2}\}$
- $P_2 = pr_{1,2} \cup pr_{2,2} \cup pr_{3,2} \cup pr_{4,2} \cup pr_{5,2}$
- $V_2(0) = (0,\ 0,\ 0,\ 0,\ 0,\ -0.2,\ -0.1,\ n_{min-1})$
- $V_3 = \{x_{1,3},\ e_{1,3}\}$
- $P_3 = pr_{1,3}$
- $V_3(0) = (0,\ 0)$
- $V_4 = \{x_{1,4},\ e_{1,4}\}$
- $P_4 = pr_{1,4}$
- $V_4(0) = (0,\ 0)$
- $V_{out} = \{x_{large},\ x_{small}\}$

The above system Π_{CMPEX} is also illustrated in Fig. 4.

In addition, Table V shows an example of variables in execution on Π_{CMPEX} in case of $x_{1,1} = -3$ and $x_{2,1} = -8$.

D. Complexity and sorting

Since the number of types of variable in the EN P system Π_{CMPEX} is $O(1)$, and $O(1)$ kinds of programs are used, we obtain the following theorem for Π_{CMPEX}.

Theorem 8: The EN P system Π_{CMPEX}, which computes the compare-exchange operation, works in $O(1)$ steps by using $O(1)$ types of variables, a constant number of membranes, and programs of size $O(1)$. □

We can also propose an EN P system $\Pi_{OddEven}$ for soring. The EN P system is based on the odd-even sort [6], which is a well-known parallel sorting algorithm. For $\Pi_{OddEven}$, we obtain the following theorem. (We omit details of the EN P systems due to space limitation.)

Theorem 9: The EN P system $\Pi_{OddEven}$, which performs the odd-even sort for n inputs, works in $O(n)$ steps by using $O(n)$ types of variables, $O(n)$ kinds of membranes, and programs of size $O(n)$. □

TABLE V
AN EXECUTION OF Π_{CMPEX}

	Initial value	Step 1	Step 2	Step 3	Step 4	Step 5
$x_{1,1}$	-3	0	0	0	0	0
$x_{2,1}$	-8	-8	0	0	0	0
$x_{3,1}$	0	-3.1	-3.1	-3.1	0	0
$x_{4,1}$	0	0	-8	-8	0	0
$x_{5,1}$	0	-3.1	-3.1	-3.1	-3.1	0
$x_{5,1}$	0	0	-8	-8	-8	0
$e_{0,1}$	0	0	0	0	0	0
$e_{1,1}$	-10	-10	-10	-3.1	-3.1	-10
$e_{2,1}$	-10	-10	-10	-10	-8	-10
$x_{1,2}$	0	-3.1	0	0	0	0
$x_{2,2}$	0	0	-8	0	0	0
$x_{3,2}$	0	-3.1	-3.1	0	0	0
$x_{4,2}$	0	0	-8	0	0	0
$x_{5,2}$	0	0	-0.1	0	0	0
$e_{0,2}$	0	0	0	0	0	0
$e_{1,2}$	-10	-10	-10	-3.1	-3.1	-10
$e_{2,2}$	0	0	0	0	-3.1	0
$x_{1,3}$	0	0	0	0	0	0
$e_{1,4}$	0	0	0	0	0	0
$x_{1,4}$	0	0	0	0	-3	0
x_{large}	-	-	-	-	-3	-3
x_{small}	-	-	-	-	-	-8

VI. CONCLUSIONS

In the present paper, we proposed EN P systems for logic operations, additions and a sorting. As future work, we are considering an EN P system using the fewer number of membranes and programs.

REFERENCES

[1] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[2] A. Leporati and C. Zandron, "P systems with input in binary form," *International Journal of Foundations of Computer Science*, vol. 17, pp. 127–146, 2006.

[3] A. Fujiwara and T. Tateishi, "Logic and arithmetic operations with a constant number of steps in membrane computing," *International Journal of Foundations of Computer Science*, vol. 22, no. 3, pp. 547–564, 2011.

[4] G. Păun and R. Păun, "Membrane computing and economics: Numerical p systems," *Fundamenta Informaticae*, vol. 73, pp. 213–227, 2006.

[5] A. Pavel, O. Arsene, and C. Buiu, "Enzymatic numerical p systems - a new class of membrane computing systems," *IEEE Fifth International Conference on BioInspired Computing: Theories and Applications (BIC-TA)*, pp. 1331–1336, 2010.

[6] N. Haberman, "Parallel neighbor-sort(or the glory of the induction principle)," *CMU Computer Science Report*, 1972.