生化学反応計算における基本演算およびソートの実現

(Reaction systems for logical operations and sorting)

Akifumi Nakanishi   Akihiro Fujiwara
Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology
Iizuka, Fukuoka, 820-8502, Japan
Email: o676122a@mail.kyutech.jp,   fujiwara@cse.kyutech.ac.jp

***Abstract*—In the present paper, we consider the reaction system, which is a computational model based on biochemical reactions in living cells, and propose reaction systems for logical operations and sorting. We first propose a reaction system that executes a two-input logical operations, such as AND, OR, and XOR, and show that the reaction system works in $O(1)$ steps. We next propose a reaction system for a compare-and-swap operation of two binary numbers of m bits. We show that the reaction system works in $O(m)$ parallel steps using $O(m)$ types of objects and reaction rules. We finally propose a reaction system for sorting of n binary numbers of m bits. The reaction system is based on an idea of the odd-even sort, and we show that the reaction system works in $O(mn)$ parallel steps and using $O(mn)$ types of objects and reaction rules.**

I. Introduction

A number of next-generation computing paradigms have been considered due to limitation of silicon based computation. As an example of the computing paradigms, natural computing, which works using natural materials for computation, has considerable attention. A membrane computing [1], which is a computational model inspired by the structures and behaviors of living cells, is a representative of the natural computing. A computational model of the membrane computing is called P system, and a number of P systems [2], [3], [4], [5], [6], [7] have been proposed for solving NP problems. In addition, a number of P systems [2], [8] are also proposed for basic operations such as logical and arithmetic operations.

On the other hand, a reaction system [9], [10], [11], which is called R system, has been proposed as another computational model of natural computing. The R system is based on biochemical reactions in living cells, and the fundamental idea of the R system is based on interaction between biochemical reactions, which are the mechanisms of facilitation and inhibition. For the reaction system, a number of primitive operations are considered in [9], [10], [11], and no R system that executes basic operations, such as logic or arithmetic operations, has been proposed. However, the reaction system for the basic operation is needed to apply the reaction system on a wide range of problems.

In the present paper, we propose R systems for logical operations and sorting of binary numbers. We first propose a reaction system that executes a two-input logical operation, such as AND, OR, and XOR. We show that the R system works in $O(1)$ parallel steps and using $O(1)$ types of objects and reaction rules.

We next propose an R system for a compare-and-swap operation of two binary numbers of m bits. The R system first computes the most significant bit between the two input values, and then, swap operation is executed according to the result of the most significant bit. We show that the R system works in $O(m)$ parallel steps and using $O(m)$ types of objects and reaction rules.

We finally propose an R system for sorting of n binary numbers of m bits. The R system is based on an idea of the odd-even sort, and the R system employs an object that works as a counter, and executes the sorting for odd and even steps using the counter. We show that the R system works in $O(mn)$ parallel steps and using $O(mn)$ types of objects and reaction rules.

II. Preliminaries

A. Reaction system

A reaction system [9], [10], [11] is a computational model based on biochemical reactions in living cells. In this paper, we first explain definition of a reaction on the reaction system, which is based on [11].

A reaction a is defined by the following equation.

$$a = (R_a, I_a, P_a)$$

R_a, I_a and P_a are sets of reactant, inhibitor and product, respectively, and all of the three sets are finite nonempty sets such that $R_a \cap I_a = \emptyset$, $M = R_a \cup I_a$, and $|M| \geq 2$.

The reaction a is applied if $R_a \subseteq T$ and $I_a \cap T = \emptyset$ for a finite set T. The result of a on T is denoted by $Res_a(T)$, and $Res_a(T) = P_a$ in case that reaction a is applied, and otherwise, $Res_a(T) = \emptyset$.

I now show an example of the reaction. Let $a = (\{3\}, \{1, 2\}, \{1, 2, 4\})$ and $T_1 = \{3, 4\}, T_2 = \{2, 3, 4\}$. In this case, $Res_a(T_1)$ and $Res_a(T_2)$ are sets given below.

$$\begin{aligned} Res_a(T_1) &= P_a = \{1, 2, 4\} \\ Res_a(T_2) &= \emptyset \end{aligned}$$

As shown in the above example, all non-reacted objects, which are not included in P_a, are disappeared after application

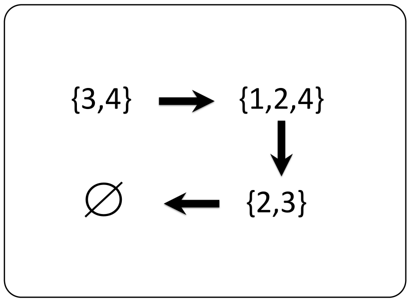

Fig. 1. Operation of R system Λ

of reaction a. The property is called non-persistency of the object.

Next, we explain definition of the R system. R system Λ is defined by the following equation.

$$\Lambda = (S, A)$$

In the above equation, S is a set of all objects, and A is a set of reactions. In addition, the result of a set of reactions A for T is an union of obtained results of all reactions in A. In other words, $Res_A(T)$, which is a set of result of A for a finite set T, is defined as follows.

$$Res_A(T) = \bigcup_{a \in A} res_a(T)$$

In the R system, an application of all reaction is called a transition. In this paper, we assume that all applicable reactions are applied simultaneously in a transition, and also assume that each transition is called one parallel step. The complexity of R system is defined the number of parallel steps executed in the computation.

For example, we show a simple R system Λ, which is defined as follows.

$$\begin{aligned} S &= \{1, 2, 3, 4\} \\ A &= \{a, b, c\} \\ a &= (\{3\}, \{1, 2\}, \{1, 2\}) \\ b &= (\{3, 4\}, \{2\}, \{4\}) \\ c &= (\{1, 4\}, \{3\}, \{2, 3\}) \end{aligned}$$

Figure1 shows an execution of the R system in case that a set $\{3, 4\}$ is given as input. In this case, applicable reactions are a and b for the input set, and these two rules are applied simultaneously. Since a result of the transition is an union of obtained objects, an obtained set of objects is $\{1, 2, 4\}$ after the first transition.

Next, an applicable reaction for $\{1, 2, 4\}$ is only c, and an obtained set of objects is $\{2, 3\}$ after the second transition. Then, a set of objects becomes empty after the third transition since no reaction is applicable for $\{2, 3\}$.

III. Data structure for binary numbers

In this subsection, we describe a unified data structure for a binary number using objects in the R system. Data structure for Boolean values has been proposed in [2], and we improve the data structure in this paper. In the data structure, one object corresponds to one bit of a binary number. Therefore, we use $O(mn)$ objects to denote n binary numbers of m bits. In addition, the data structure enables the addressing feature, that is, each binary number is stored in a given address.

Let $V_{i,j} \in \{0, 1\}$ be a j-th Boolean value stored in address i. Then, the value is denoted using the following object on the R system.

$$\langle A_i, B_j, V_{i,j} \rangle$$

We call the above object ***a memory object*** for Boolean values.

In case of a binary number, let $V_{i,m-1}, V_{i,m-2}, \cdots, V_{i,0}$ be m Boolean values stored in address i. Then, a non-negative integer V_i stored in address i satisfies the following condition.

$$V_i = \sum_{j=1}^{m} V_{i,j} \times 2^{j-1}$$

The above binary number is represented by m memory objects given below.

$$\langle A_i, B_{m-1}, V_{i,m-1} \rangle, \langle A_i, B_{m-2}, V_{i,m-2} \rangle, \cdots, \langle A_i, B_0, V_{i,0} \rangle$$

For example, the binary number 1010 stored in the address 6 is represented by the following four objects.

$$\langle A_6, B_3, 1 \rangle, \langle A_6, B_2, 0 \rangle, \langle A_6, B_1, 1 \rangle, \langle A_6, B_0, 0 \rangle$$

IV. R systems for logical operations

A. Input and output

In this section, we propose a simple R system that executes a two-input logical operation, such as AND, OR, and XOR. We assume that input of the logical operation is a pair of Boolean values x, y. The x, y are denoted by a pair of following two memory objects on the R system.

$$\langle A_0, B_0, V_{0,0} \rangle, \quad \langle A_1, B_0, V_{1,0} \rangle$$

We also assume that an output of the logical operation is a Boolean value z, which is denoted by the following memory object.

$$\langle A_2, B_0, V_{2,0} \rangle$$

B. R system for logical operations

In this subsection, we show an R system for the logical operation. Any two-input logical operation is defined in the truth table in Table I. For example, $z_0 = z_1 = z_2 = 0$ and $z_3 = 1$ in case of AND operation. We propose an R system for any two-input logical operation based on the truth table.

C. Details of the R system for the logical operation

An output of the logical operation is determined according to the truth table. Thus, we encode the truth table into the R system as reactions. We now formally define the R system Λ_{LO} for any logical operations in the following.

$$\Lambda_{LO} = (S, A)$$

TABLE I
A TRUTH TABLE OF A TWO-INPUT LOGICAL OPERATION

Input x	Input y	Output z
0	0	z_0
0	1	z_1
1	0	z_2
1	1	z_3

$$
\begin{aligned}
S &= \{\langle A_0, B_0, V_{0,0}\rangle, \langle A_1, B_0, V_{1,0}\rangle, \langle A_2, B_0, V_{2,0}\rangle,\\
&\qquad \langle E\rangle | V_{0,0}, V_{1,0}, V_{2,0} \in \{0,1\}\}\\
A &= \{a_1, a_2, a_3, a_4\}\\
a_1 &= (\{\langle A_0, B_0, 0\rangle, \langle A_1, B_0, 0\rangle\}, \{\langle E\rangle\}, \{\langle A_2, B_0, z_0\rangle\})\\
a_2 &= (\{\langle A_0, B_0, 0\rangle, \langle A_1, B_0, 1\rangle\}, \{\langle E\rangle\}, \{\langle A_2, B_0, z_1\rangle\})\\
a_3 &= (\{\langle A_0, B_0, 1\rangle, \langle A_1, B_0, 0\rangle\}, \{\langle E\rangle\}, \{\langle A_2, B_0, z_2\rangle\})\\
a_4 &= (\{\langle A_0, B_0, 1\rangle, \langle A_1, B_0, 1\rangle\}, \{\langle E\rangle\}, \{\langle A_2, B_0, z_3\rangle\})
\end{aligned}
$$

In the above R system, $\langle E\rangle$ is an object that denotes an empty set, and z_0, z_1, z_2, z_3 are Boolean values given in Table I.

For a simple example of the above R system, we assume that two input objects $\langle A_0, B_0, 0\rangle$ and $\langle A_1, B_0, 1\rangle$ are given to the R system. Then, a reaction a_2 is applied, and an object $\langle A_2, B_0, z_1\rangle$ is obtained as an output of the R system.

Since computation of the above R system Λ_{OL} is completed by only one transaction, we obtain the following theorem for Λ_{OL}.

Theorem 1: The R system Λ_{OL}, which computes any two-input logical operations, works in $O(1)$ parallel steps using $O(1)$ types of objects and reactions. □

V. COMPARE-AND-SWAP

In this section, we present an R system for the compare-and-swap operation of two binary numbers of m bits. The compare-and-swap operation compares two input values x_{in} and y_{in}, and assigns the smaller and larger values to the variable x_{out} and y_{out}, respectively.

A. Input and output

Two input binary numbers are denoted by the following sets of memory objects on the R system.

$$
\begin{aligned}
x_{in} &: \quad \{\langle A_0, B_j, V_{0,j}\rangle \mid 0 \le j \le m-1\}\\
y_{in} &: \quad \{\langle A_1, B_j, V_{1,j}\rangle \mid 0 \le j \le m-1\}
\end{aligned}
$$

In addition to the above input, we assume that the following object is given to the R system. The object, which denotes bit-position of the comparison, starts the R system.

$$\langle C, m-1\rangle$$

In addition, two output binary numbers are also denoted by the following sets of memory objects.

$$
\begin{aligned}
x_{out} &: \quad \{\langle A_2, B_j, V_{2,j}\rangle \mid 0 \le j \le m-1\}\\
y_{out} &: \quad \{\langle A_3, B_j, V_{3,j}\rangle \mid 0 \le j \le m-1\}
\end{aligned}
$$

B. An overview of the R system

Let $x_{in,j}$ and $y_{in,j}$ be j-th bit of two input binary numbers. The R system for the compare-and-swap consists of the following two steps. First, in Step 1, each bit of the two input binary numbers, x_{in} and y_{in}, are compared from a higher bit to a lower bit. We assume that $x_{in,k}$ and $y_{in,k}$ are the first pair of bits such that the bits are different in the comparison. Then, x_{in} is greater than y_{in} if $x_{in,k} = 1, y_{in,k} = 0$, otherwise, x_{in} is less than y_{in}, and $x_{in,k} = 0, y_{in,k} = 1$. An object that denotes a result of the comparison is created after the comparison.

Second, in Step 2, two binary numbers are exchanged and outputted to x_{out} and y_{out} in case of $x_{in} < y_{in}$. Otherwise, x_{in} and y_{in} are copied into x_{out} and y_{out}, respectively.

In the above two steps, it is worth while noticing that memory objects must be copied repeatedly because of non-persistency of the object.

C. Details of the R system

We now explain each step of the R system. In Step 1, each bit of the two input binary numbers, x_{in} and y_{in}, are compared from a higher bit to a lower bit. The comparison is executed using the following sets of reactions.

$$
\begin{aligned}
A_{1,1} &= \{\ (\{\langle A_0, B_k, 0\rangle, \langle A_1, B_k, 1\rangle, \langle C, k\rangle\}, \{\langle E\rangle\},\\
&\qquad \{\langle A_0, B_k, 0\rangle, \langle A_1, B_k, 1\rangle, \langle LT\rangle\})\\
&\qquad\ \mid 0 \le k \le m-1\}\\
&\quad \cup \{\ (\{\langle A_0, B_k, 1\rangle, \langle A_1, B_k, 0\rangle, \langle C, k\rangle\}, \{\langle E\rangle\},\\
&\qquad \{\langle A_0, B_k, 1\rangle, \langle A_1, B_k, 0\rangle, \langle GT\rangle\})\\
&\qquad\ \mid 0 \le k \le m-1\}\\
A_{1,2} &= \{\ (\{\langle A_0, B_k, 0\rangle, \langle A_1, B_k, 0\rangle, \langle C, k\rangle\}, \{\langle E\rangle\},\\
&\qquad \{\langle A_0, B_k, 0\rangle, \langle A_1, B_k, 0\rangle, \langle C, k-1\rangle\})\\
&\qquad\ \mid 1 \le k \le m-1\}\\
&\quad \cup \{\ (\{\langle A_0, B_k, 1\rangle, \langle A_1, B_k, 1\rangle, \langle C, k\rangle\}, \{\langle E\rangle\},\\
&\qquad \{\langle A_0, B_k, 1\rangle, \langle A_1, B_k, 1\rangle, \langle C, k-1\rangle\})\\
&\qquad\ \mid 1 \le k \le m-1\}
\end{aligned}
$$

In case of j-th bit of two binary numbers are different, reactions in $A_{1,1}$ are applied, and one of objects, $\langle LT\rangle$ or $\langle GT\rangle$, which denotes $x_{in} < y_{in}$ or $x_{in} > y_{in}$, is created. Otherwise, two bits are copied, and the comparison is moved to the next bit-position using reactions in $A_{1,2}$.

In addition to the above, the other memory objects are copied repeatedly, due to non-persistency of the object, using the following sets of reactions.

$$
\begin{aligned}
A_{1,3} &= \{\ (\{\langle A_0, B_k, V_{0,k}\rangle\}, \{\langle C, k\rangle, \langle LT\rangle, \langle GT\rangle, \langle EQ\rangle\},\\
&\qquad \{\langle A_0, B_k, V_{0,k}\rangle\}) \mid 0 \le k \le m-1, V_{0,k} \in \{0,1\}\}\\
&\quad \cup \{\ (\{\langle A_1, B_k, V_{1,k}\rangle\}, \{\langle C, k\rangle, \langle LT\rangle, \langle GT\rangle, \langle EQ\rangle\},\\
&\qquad \{\langle A_1, B_k, V_{1,k}\rangle\}) \mid 0 \le k \le m-1, V_{1,k} \in \{0,1\}\}
\end{aligned}
$$

In case of $x_{in} = y_{in}$, reactions in the following $A_{1,4}$ is applied after comparisons of all bits, and an object $\langle EQ\rangle$ is created.

$$
\begin{aligned}
A_{1,4} \;=\; & \{\ (\{\langle A_0, B_0, 0\rangle, \langle A_1, B_0, 0\rangle, \langle C, 0\rangle\}, \{\langle E\rangle\}, \\
& \ \{\langle A_0, B_0, 0\rangle, \langle A_1, B_0, 0\rangle, \langle EQ\rangle\})\ \} \\
& \cup \{\ (\{\langle A_0, B_0, 1\rangle, \langle A_1, B_0, 1\rangle, \langle C, 0\rangle\}, \{\langle E\rangle\}, \\
& \ \{\langle A_0, B_0, 1\rangle, \langle A_1, B_0, 1\rangle, \langle EQ\rangle\})\ \}
\end{aligned}
$$

Next, in Step 2, two input binary numbers, x_{in} and y_{in}, are exchanged and outputted to x_{out} and y_{out} in case that there exists an object $\langle GT\rangle$. The swap and copy are executed using the following set of reactions $A_{2,1}$.

$$
\begin{aligned}
A_{2,1} \;=\; & \{\ (\{\langle A_0, B_k, V_{0,k}\rangle, \langle GT\rangle\}, \{\langle E\rangle\}, \\
& \ \{\langle A_3, B_k, V_{0,k}\rangle\}) \mid 0 \le k \le m-1, V_{0,k} \in \{0,1\}\} \\
& \cup \{\ (\{\langle A_1, B_k, V_{1,k}\rangle, \langle GT\rangle\}, \{\langle E\rangle\}, \\
& \ \{\langle A_2, B_k, V_{1,k}\rangle\}) \mid 0 \le k \le m-1, V_{1,k} \in \{0,1\}\}
\end{aligned}
$$

On the other hand, two input binary numbers, x_{in} and y_{in}, are copied to x_{out} and y_{out}, respectively, in case that there exists $\langle LT\rangle$ or $\langle EQ\rangle$. The copy is executed using the following sets of reactions, $A_{2,2}$ and $A_{2,3}$.

$$
\begin{aligned}
A_{2,2} \;=\; & \{\ (\{\langle A_0, B_k, V_{0,k}\rangle, \langle LT\rangle\}, \{\langle E\rangle\}, \\
& \ \{\langle A_2, B_k, V_{0,k}\rangle\}) \mid 0 \le k \le m-1, V_{0,k} \in \{0,1\}\} \\
& \cup \{\ (\{\langle A_1, B_k, V_{1,k}\rangle, \langle LT\rangle\}, \{\langle E\rangle\}, \\
& \ \{\langle A_3, B_k, V_{1,k}\rangle\}) \mid 0 \le k \le m-1, V_{1,k} \in \{0,1\}\} \\
A_{2,3} \;=\; & \{\ (\{\langle A_0, B_k, V_{0,k}\rangle, \langle EQ\rangle\}, \{\langle E\rangle\}, \\
& \ \{\langle A_2, B_k, V_{0,k}\rangle\}) \mid 0 \le k \le m-1, V_{0,k} \in \{0,1\}\} \\
& \cup \{\ (\{\langle A_1, B_k, V_{1,k}\rangle, \langle EQ\rangle\}, \{\langle E\rangle\}, \\
& \ \{\langle A_3, B_k, V_{1,k}\rangle\}) \mid 0 \le k \le m-1, V_{1,k} \in \{0,1\}\}
\end{aligned}
$$

We now summarize details of the R system Λ_{CS}, which executes the compare-and-swap operation.

$$\Lambda_{CS} = (S, A)$$

$$
\begin{aligned}
S \;=\; & \{\langle A_i, B_j, V_{0,j}\rangle \mid 0 \le i \le 3, 0 \le j \le m-1\} \\
& \cup \{\langle C, k\rangle \mid 0 \le k \le m-1\} \\
& \cup \{\langle GT\rangle, \langle LT\rangle, \langle EQ\rangle, \langle E\rangle\} \\
A \;=\; & A_{1,1} \cup A_{1,2} \cup A_{1,3} \cup A_{1,4} \cup A_{2,1} \cup A_{2,2} \cup A_{2,3}
\end{aligned}
$$

Figure 2 illustrates an execution of the R system Λ_{CS}. In the example, $m = 3$, and an input is a pair of two binary numbers, $x_{in} = 1100$ and $y_{in} = 1011$. In the example, at first, reactions in $A_{1,2}$ and $A_{1,3}$ are applied because the highest bits of the two numbers are same. Next, reactions in $A_{1,1}$ and $A_{1,3}$ are applied because the next bits of the two numbers are different, and an object $\langle GT\rangle$ is created according to the comparison. Finally, the output values are set to x_{out} and y_{out} using reactions in $A_{2,1}$.

D. Complexity of the R system

Since complexity of Step 1 in the above R system Λ_{CS} is $O(m)$, we obtain the following theorem for Λ_{CS}.

Theorem 2: The R system Λ_{CS}, which executes the compare-and-swap operation for two binary numbers of m bits, works in $O(m)$ parallel steps using $O(m)$ types of objects and reactions. □

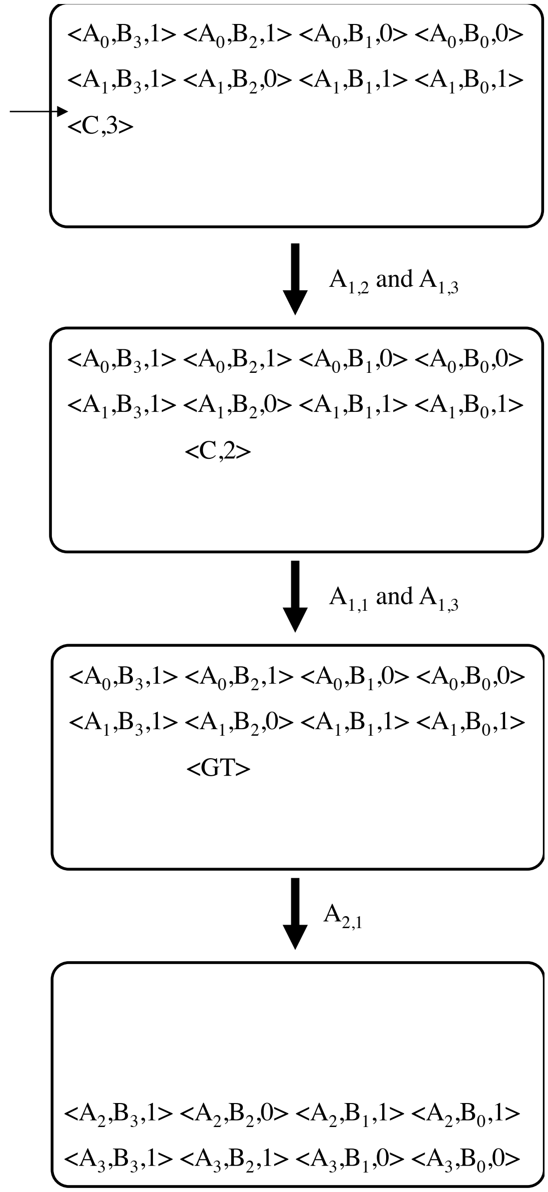

Fig. 2. An example of execution of Λ_{CS}

VI. Sorting

A. Input and output

In this section, we present an R system for sorting of n binary numbers of m bits using the R system. An input and an output of the R system is a set of binary numbers that are denoted by the following set of memory objects.

$$\{\langle A_i, B_j, V_{i,j}\rangle \mid 0 \le i \le n-1, 0 \le j \le m-1\}$$

B. An overview and complexity of the R system

The proposed P system is based on odd-even transposition sort [12], which is a well-known parallel sorting algorithm. Let

$(x_0, x_1, \ldots, x_{n-1})$ be an input of the sorting. A basic idea of the odd-even transposition sort is quite simple. At odd phases, we perform the compare-and-swap operations for each pair (x_{2i}, x_{2i+1}) $(0 \le i \le \frac{n}{2} - 1)$ in parallel. On the other hand, we perform the same operations for each pair (x_{2i-1}, x_{2i}) $(1 \le i \le \frac{n}{2} - 1)$ at even phases. It is proved that the input is sorted after $\frac{n}{2}$ repetition of the above two steps [12].

Using the R system described in Section V, we can realize the odd-even transposition sort on the R system as follows.

A basic idea of R system for sorting

Repeat the following steps $\frac{n}{2}$ times.

Step 1: Execute the compare-and-swap operations, which is described in Section V, for pairs (x_{2i}, x_{2i+1}) $(0 \le i \le \frac{n}{2} - 1)$ in parallel.

Step 2: Execute the same compare-and-swap operations for pairs (x_{2i-1}, x_{2i}) $(1 \le i \le \frac{n}{2} - 1)$ in parallel.

Since all reactions on the R system can be applied in parallel, the above idea is implemented as an R system Λ_{SORT}, which sorts n binary number of m bits, with a modification of the R system proposed in Section V. Although the main difficulty for the implementation is synchronization of the compare-and-swap operations, we can solve the difficulty by using object that works as a global counter. (The precise description of the R system is omitted because an implementation of reactions is redundant.)

We now consider complexity of the R system Λ_{SORT}. The above two steps are repeated by $\frac{n}{2}$ times, and each compare-and-swap operation is executed in $O(m)$ steps. Then, we obtain the following theorem for the R system Λ_{SORT}.

Theorem 3: The R system Λ_{SORT}, which sorts n binary numbers of m bits, works in $O(mn)$ parallel steps using $O(mn)$ types of objects and reactions. □

VII. Conclusions

In the present paper, we proposed R systems for logical operations and sorting of binary numbers. We first proposed an R system that executes any two-input logical operation, and showed that the R system works in $O(1)$ parallel steps and using $O(1)$ types of objects and reaction rules. We next proposed an R system for a compare-and-swap operation of two binary numbers of m bits, and showed that the R system works in $O(m)$ parallel steps and using $O(m)$ types of objects and reaction rules. We finally proposed an R system for sorting of n binary numbers of m bits, and also showed that the R system works in $O(mn)$ parallel steps and using $O(mn)$ types of objects and reaction rules.

As future work, we are considering reduction of the numbers of types of objects and reactions in the R systems.

Acknowledgments

This research was partially supported by JSPS KAKENHI, Grand-in-Aid for Scientific Research (C), 24500019.

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, no. 1, pp. 127–146, 2006.

[3] L. Pan and A. Alhazov, "Solving HPP and SAT by P systems with active membranes and separationrules," *Acta Informatica*, vol. 43, no. 2, pp. 131–145, 2006.

[4] G. Păun, "P system with active membranes: Attacking NP-complete problems," *Journal of Automata, Languages and Combinatorics*, vol. 6, no. 1, pp. 75–95, 2001.

[5] C. Zandron, G. Rozenberg, and G. Mauri, "Solving NP-complete problems using P systems with active membranes," *Proceedings of the Second International Conference on Uncoventional Models of Computation*, pp. 289–301, 2000.

[6] H. Tagawa and A. Fujiwara, "Solving SAT and Hamiltonian cycle problem using asynchronous p systems," *IEICE Transactions on Information and Systems (Special section on Foundations of Computer Science)*, vol. E95-D, no. 3, 2012.

[7] K. Tanaka and A. Fujiwara, "Asynchronous p systems for hard graph problems," *International Journal of Networking and Computing*, vol. 4, no. 1, pp. 2–22, 2014.

[8] 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.

[9] G. Păun, "Bridging P and R," *Research Topics in Membrane Computing*, vol. 1, no. 1, pp. 53–57, 2012.

[10] ——, "Towards fypercomputations (in membrane computing)," *Lecture Notes in Computer Science*, vol. 7300, pp. 207–220, 2012.

[11] M. M. R. Brijder, A. Ehrenfeucht, "A tour of reaction systems," *International Journal of Foundations of Computer Science*, vol. 22, no. 1, pp. 1499–1518, 2011.

[12] N. Haberman, "Parallel neighbor-sort (or the glory of the induction principle)," AD-759 248, National Technical Information Service, US Department of Commerce, Tech. Rep., 1972.