Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 858527, 10 pages
http://dx.doi.org/10.1155/2014/858527

Research Article

Fraction Reduction in Membrane Systems

Ping Guo,[1,2] Hong Zhang,[1] Haizhu Chen,[3] and Ran Liu[1]

[1] *College of Computer Science, Chongqing University, Chongqing 400030, China*
[2] *Chongqing Key Laboratory of Software Theory & Technology, Chongqing 400044, China*
[3] *Department of Software Engineering, Chongqing College of Electronic Engineering, Chongqing 401331, China*

Correspondence should be addressed to Ping Guo; guoping_cqu@163.com

Received 18 September 2013; Accepted 12 February 2014; Published 16 March 2014

Academic Editors: H. Cheng and B. Nagy

Fraction reduction is a basic computation for rational numbers. P system is a new computing model, while the current methods for fraction reductions are not available in these systems. In this paper, we propose a method of fraction reduction and discuss how to carry it out in cell-like P systems with the membrane structure and the rules with priority designed. During the application of fraction reduction rules, synchronization is guaranteed by arranging some special objects in these rules. Our work contributes to performing the rational computation in P systems since the rational operands can be given in the form of fraction.

1. Introduction

Membrane computing (also called P systems) is a branch of natural computing introduced by Păun in 1998 which abstracts computing models from the architecture and the functioning of living cells [1]. Membrane computing model takes the living cell as multihierarchical structural regions which are referred to as the membranes [2]. In the compartments defined by membranes there are objects that can evolve to other objects and pass through the membranes. After Păun proposed and proved that P systems based on membrane division can solve SAT problems in polynomial time [3], many variants of P systems, including cell-like [4, 5], tissue-like [6], and neural-like ones [7], have been successfully used to design solutions for NP-complete problems. The introductions of the complexity, parallelism, decomposition of membrane, and hierarchical structure can be found in [8, 9].

Based on cell-like P systems which are one kind of common systems in membrane computing, Atanasiu firstly constructs arithmetic P systems to implement arithmetic operations [10]. Reference [11] designs multilayer P systems without priority rules to lower the complexity of the operations. And, in [12], the membrane structure is simplified greatly and efficiency of the computations is also improved owing to arithmetic operations being performed in a single membrane without priority rules. Furthermore, multimembrane P systems are constructed for signed arithmetic operations [13] and the operational range of P system can be extended to the whole integer field. In [14] arithmetic expression is evaluated with primary arithmetical operations implemented in single membranes. Reference [15] proposes an algorithm and builds expression P systems without priority rules for evaluating arithmetic expression. And [16] implements primary arithmetic operations of fractions in P systems and builds a bridge between rational numbers and membrane computing. In cell-like P systems, the operands of the arithmetic operations are represented by multiset, which is composed by the objects and their cardinalities. The rational number can be given by the form of fraction, whose numerator and denominator can be represented by multisets, respectively, so the fraction can be a bridge for implementing the calculations of the rational numbers in P systems.

However, [16] has not further processed the computation results which need to be reduced to lighten the load of the subsequent fraction computations. This paper proposes a suitable method of fraction reduction and implements it in P systems. The rest of this paper is organized as follows: Section 2 introduces cell-like P systems, and Section 3 proposes and proves the fraction reduction method. In Section 4, based on cell-like P system, the rules for implementing fraction reduction are described in detail with the membrane structure designed. The conclusions are drawn in the final section.

2. Foundations

2.1. Cell-Like P Systems. Our work in this paper is based on cell-like P systems, and such system (of degree $m \geq 1$) can be defined formally as [1, 2]

$$\Pi = (O, \mu, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, \rho_1, \ldots, \rho_m, i_o), \quad (1)$$

where

(i) O is the alphabet of the system. Each symbol represents one kind of object. O^* is the finite and nonempty multiset over O where λ is empty string; $O^+ = O^* - \{\lambda\}$;

(ii) μ is a membrane structure with m membrane, labeled by $1, 2, \ldots, m$;

(iii) ω_i $(1 \leq i \leq m)$ is string over O representing the multiset of objects placed in membrane i. For example, there are 5 copies of object a and 3 copies of object b in membrane i; then we have $\omega_i = a^5 b^3$; $\omega_i = \lambda$ means that there is no object in membrane i;

(iv) i_o is output region of the system and it saves the final results;

(v) $R_1, R_2, \ldots, R_m$ are finite sets of possible evolution rules over O associated with the regions $1, 2, \ldots, m$ of μ. The rules in R_i $(1 \leq i \leq m)$ are of the form $U \rightarrow V|_a$, with $a \in O$, $U \in O^+$, $V = (V', \xi)$, or $V = (V', \xi)\ \delta$, $V' \in O^*$ and $\xi = \{\text{here}, \text{out}, \text{in}_j \mid 1 \leq j \leq m\}$: here means the product V' remains in the same region; out means V' goes out of the region and enters into another membrane which includes membrane i as its submembrane; and in_j means V' goes into membrane j which is a submembrane of membrane i. Specifically, when $\xi = \text{here}$, (V', ξ) can be abbreviated as V'. δ is a special symbol not in O, and it means that the membrane which includes it will be dissolved and the contents of this membrane will be left in the outer membrane. Object a is a promoter in the rule $U \rightarrow V|_a$; this rule can only be applied in the presence of object a;

(vi) ρ_i $(1 \leq i \leq m)$ defines a partial order relation among the rules in R_i. If $\rho_i = \{a \rightarrow b > c \rightarrow d\}$ and both objects a and c are available, then only $a \rightarrow b$ can be applied although the two rules do not compete for any objects.

Beside the above rules, we also consider rules for membrane creation, which is of the form $e \rightarrow [\,_i V\,]_i$, with $e \in O$, $V \in O^*$, and i is a number from a given list of the labels; the idea is that the object e creates a new membrane labeled by i, including multiset V and associated with evolution rules [17].

In each membrane, rules are applied according to the following principles.

(i) Nondeterminism. Suppose n rules compete for the reactants which can only support m $(m < n)$ rules to be applied; then the m rules are chosen nondeterministically.

(ii) Maximal parallelism. All of the rules that can be applied must be applied simultaneously.

From now on we only deal with cell-like P systems with membrane creation and call them P systems for brevity.

2.2. Fraction Arithmetic Operations. Reference [16] discusses how to perform fraction arithmetic operations by P systems based on multiple membranes. In [16], the fraction arithmetical operations are written in the form as

$$\frac{(+/-)\, m_1}{m_2}\, op\, \frac{(+/-)\, n_1}{n_2}, \quad op \in \{+, -, \times, \div\}, \quad (2)$$

where m_1, m_2, n_1, and n_2 are all integers; $m_2 > 0$, $n_2 > 0$, $m_1 \geq 0$, and $n_1 \geq 0$.

Fraction operands are converted into the format which the integer arithmetic requires when the operation is processed. The process of initialization makes the fraction operand be represented in a unified form and it simplifies the operation rules since different operands can be represented by the same objects in the P systems. After initialization, [16] designs four kinds of fraction arithmetic P systems to implement primary arithmetic operations of fractions (namely, addition, subtraction, multiplication, and division).

The computation results obtained by the systems in [16] are not in reduced form. So they are required to be processed further for lightening the load of the subsequent fraction computations. However, the current methods for fraction reductions are not available in P systems. In this paper, we propose a method of fraction reduction and discuss how to carry it out in cell-like P systems.

3. A Method for Fraction Reduction

The goal of fraction reduction is to obtain the simplest fraction, and it means the numerator and denominator are coprimes. Generally, fractions can be reduced by the following methods.

(i) Numerator and denominator are divided by the prime factors that they share until their common factor is 1.

(ii) Numerator and denominator are divided directly by their greatest common factor.

These two methods are simple, but both of them are not suitable for being implemented in P systems owing to the following.

(i) For the first one, we need to enumerate the primes, such as $2, 3, 5, 7, \ldots$, to find out the common prime factors of the numerator and denominator. This method involves large calculation and cannot be processed in parallel. If it is implemented in P system, a rule or several rules should be designed for testing whether a prime is their common factor. It means that the more prime factors they share, the more rules are designed and the more complex membrane structure is.

(ii) For the second one, the greatest common factor of the numerator and denominator should be calculated by Euclidean algorithm, but it cannot be performed efficiently in P systems.

For designing a set of generally universal rules to implement fraction reduction in P systems, we present a new fraction reduction method, based on which the designed system works independently on the size of the input. In this section, some theories on the new method are given and the corresponding algorithm is proposed subsequently with its correctness ensured by the present theories.

3.1. The Principles for Fraction Reduction. Assume that we have integers m, n ($0 < n < m$) and let $n_0 = m$, $n_1 = n$, the sequences $\{a_i\}$, $\{n_i\}$, and $\{k_i\}$ can be constructed as

$$a_i = \frac{n_i}{n_{i-1}} = \frac{1}{k_i + (n_{i+1}/n_i)}, \quad 0 \le n_{i+1} < n_i,\ i \ge 1, \tag{3}$$

where, for $i \ge 1$,

$$\begin{gathered} k_i = n_{i-1} \operatorname{div} n_i, \\ n_{i+1} = n_{i-1} \bmod n_i. \end{gathered} \tag{4}$$

For $\{n_0, n_1, \ldots, n_\rho\}$, we have the following.

Theorem 1. *Integer sequence $\{n_0, n_1, \ldots, n_\rho\}$ is monotone decreasing, and there is an integer $v > 0$, such that $n_v = 0$.*

Proof. (i) Obviously, $\{n_0, n_1, \ldots, n_\rho\}$ is monotone decreasing according to the procedure of the construction.

(ii) Assume that n_t is the minimum in $\{n_0, n_1, \ldots, n_\rho\}$ and $n_t > 0$. From the construction, we have

$$n_{t+1} = n_{t-1} \bmod n_t. \tag{5}$$

It is easy to see that $0 \le n_{t+1} < n_t$. According to the assumption, we obtain $n_{t+1} = 0$ and let $v = t + 1$, namely, $n_v = 0$. □

From (3), we have

$$\begin{gathered} a_t = \frac{n_t}{n_{t-1}} = \frac{1}{k_t + (n_{t+1}/n_t)} = \frac{1}{k_t}, \\ a_i = \frac{1}{k_i + a_{i+1}}, \quad 0 \le i < t-1. \end{gathered} \tag{6}$$

The sequence $\{f_i\}$ can be constructed as follows:

$$\begin{gathered} f_{t+1} = 1, \\ f_t = k_t, \\ f_{t-1} = k_{t-1} \times f_t + 1. \end{gathered} \tag{7}$$

Generally,

$$f_i = k_i \times f_{i+1} + f_{i+2}, \quad 1 \le i < t-2. \tag{8}$$

So,

$$\begin{aligned} a_t &= \frac{1}{f_t}, \\ a_{t-1} &= \frac{1}{k_{t-1} + a_t} = \frac{1}{k_{t-1} + (1/f_t)} = \frac{f_t}{k_{t-1} \times f_t + 1} \\ &= \frac{f_t}{f_{t-1}}, \\ a_{t-2} &= \frac{1}{k_{t-2} + a_{t-1}} = \frac{1}{k_{t-2} + (f_t/f_{t-1})} = \frac{f_{t-1}}{k_{t-2} \times f_{t-1} + f_t} \\ &= \frac{f_{t-1}}{f_{t-2}}. \end{aligned} \tag{9}$$

Generally, for $0 < i < t$,

$$\begin{aligned} a_i &= \frac{1}{k_i + a_{i+1}} = \frac{1}{k_i + (f_{i+2}/f_{i+1})} = \frac{f_{i+1}}{k_i \times f_{i+1} + f_{i+2}} \\ &= \frac{f_{i+1}}{f_i}. \end{aligned} \tag{10}$$

Specifically,

$$a_1 = \frac{1}{k_1 + a_2} = \frac{f_2}{k_1 \times f_2 + f_3} = \frac{f_2}{f_1}. \tag{11}$$

Theorem 2. *f_1 and f_2 are coprimes, and f_2/f_1 is the simplest proper fraction of n/m.*

Proof. let ξ be the common factor of f_1 and f_2. According to the construction of $\{f_i\}$, we can obtain $f_1 = k_1 \times f_2 + f_3$, so ξ is also the factor of f_3. Similarly, ξ is the common factor of f_{t-1} and f_t. It means that there are ξ_1 and ξ_2, such that $f_{t-1} = \xi \times \xi_1$ and $f_t = \xi \times \xi_2$. According to the definition of f_{t-1}, we have

$$f_{t-1} = \xi \times \xi_1 = k_{t-1} \times (\xi \times \xi_2) + 1. \tag{12}$$

That is,

$$\xi \times (\xi_1 - k_{t-1} \times \xi_2) = 1. \tag{13}$$

Namely, ξ is the factor of 1. Hence, $\xi = 1$. So f_1 and f_2 are coprimes.

According to the construction of $\{a_i\}$ and (11), we have $f_2/f_1 = a_1 = n/m$, so f_2/f_1 is the simplest proper fraction of n/m. □

The proofs of the above theories show that the proposed method is feasible for fraction reduction; namely, the simplest proper fraction can be obtained by this method for any fraction.

```
Input: $n, m (m \geq n > 0)$;
Output: $f_1, f_2$ ($f_2/f_1$ is the simplest proper fraction of $n/m$);
Procedure:
    //calculate $\{n_i\}, \{k_i\}$
        $i \leftarrow 1$;
        $n_1 \leftarrow n, n_0 \leftarrow m$;
        repeat
            $k_i \leftarrow n_{i-1}$ div $n_i$;
            $n_{i+1} \leftarrow n_{i-1}$ mod $n_i$;
            $i \leftarrow i + 1$;
        until $n_{i+1} = 0$;
    //calculate $\{f_i\}$;
        $i \leftarrow i - 1$;
        $f_{i+1} \leftarrow 1$;
        $f_i \leftarrow k_i$;
        while $i > 1$ {
            $f_{i-1} \leftarrow k_{i-1} * f_i + f_{i+1}$;
            $i \leftarrow i - 1$;
        }
End.
```

Algorithm 1: Fraction reduction.

3.2. The Algorithm for Fraction Reduction. Assume that we want to reduce the fraction n/m ($0 < n < m$); from the discussion in Section 3.1, the procedure for fraction reduction can be described as follows:

(i) input n, m ($0 < n \leq m$);

(ii) compute $\{n_i\}, \{k_i\}, i = 2, 3, \ldots u$, and $n_1 = n, n_0 = m$;

(iii) compute $\{f_i\}, i = t, t-1, \ldots, 1$;

(iv) output f_1,f_2.

We can present an algorithm for fraction reduction in Algorithm 1.

In this algorithm, the complexity of the algorithm is $O(1)$ when m is a multiple of the n. Generally, for sequence $\{n_i\}$, we know $n_{t+1} = 0$ if the algorithm performs mod operation for t times. Comparing sequence $\{n_i\}$ with Fibonacci sequence $\{F_i\}$, we have $F_0 = 1 \leq n_t$ and $F_1 = 1 \leq n_{t-1}$. And $n_k \geq n_{k+1} + n_{k+2}$ can be obtained due to n_k mod $n_k + 1 = n_k + 2$ $(0 \leq k \leq t-1)$. So $n_k \geq F_{t-k}$ can be concluded by mathematical induction. Furthermore, we can obtain $m = n_0 \geq F_t$ and $n = n_1 \geq F_{t-1}$. That is to say, n must be not less than F_{t-1} if our algorithm performs mod operation for t times and vice versa. We have $F_{t-1} \geq (1.618)^t/\sqrt{5}$ according to the feature of Fibonacci sequence; namely, $n \geq (1.618)^t/\sqrt{5}$ and $t \leq \log_{1.618}(\sqrt{5}n)$, so the complexity of the algorithm is $O(\log n)$ in the worst case.

4. Fraction Reduction in P Systems

In this section, a kind of P systems is designed for fraction reduction based on the algorithm proposed in Section 3.2.

4.1. The P Systems for Fraction Reduction. The P systems for fraction reduction can be defined as the form of (1) given in Section 2.1, where:

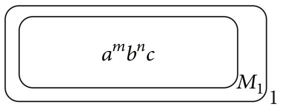


Figure 1: The initial configuration of P system for fraction reduction.

(i) O is the (finite and nonempty) alphabet of objects which occur in the rules in the designed P system;

(ii) μ is the structure of the system and it can be decided by the rules presented subsequently;

(iii) $i_o = 1$, and it means that the final result can be found in membrane 1 when the whole system halts.

Figure 1 describes the initial configuration of this P system: membrane 1 is responsible for keeping the final results and dissolving the other objects coming from membrane M_1, while fraction reduction is processed in membrane M_1. Except the two membranes, other membranes will be dynamically created during the reduction, and the rules in the new membranes are the same as the ones in membrane M_1. In Figure 1, objects a and b are used to label the denominator and numerator, respectively; m and n, which are the cardinalities of a and b, represent the absolute value of the denominator and numerator, respectively; object c is used to trigger the rules in membrane M_1.

The algorithm proposed in Section 3.2 can be implemented by the P system as shown in Figure 2. In Figure 2, membrane M_1 and the created membranes are responsible for computing $\{n_i\}$, $\{k_i\}$, and $\{f_i\}$. These membranes are nested one by one: the new membrane M_2 is created in membrane M_1, and another new membrane M_3 is created in membrane M_2. Finally, membrane M_t is created in membrane M_{t-1} (t is decided in Section 3.2).

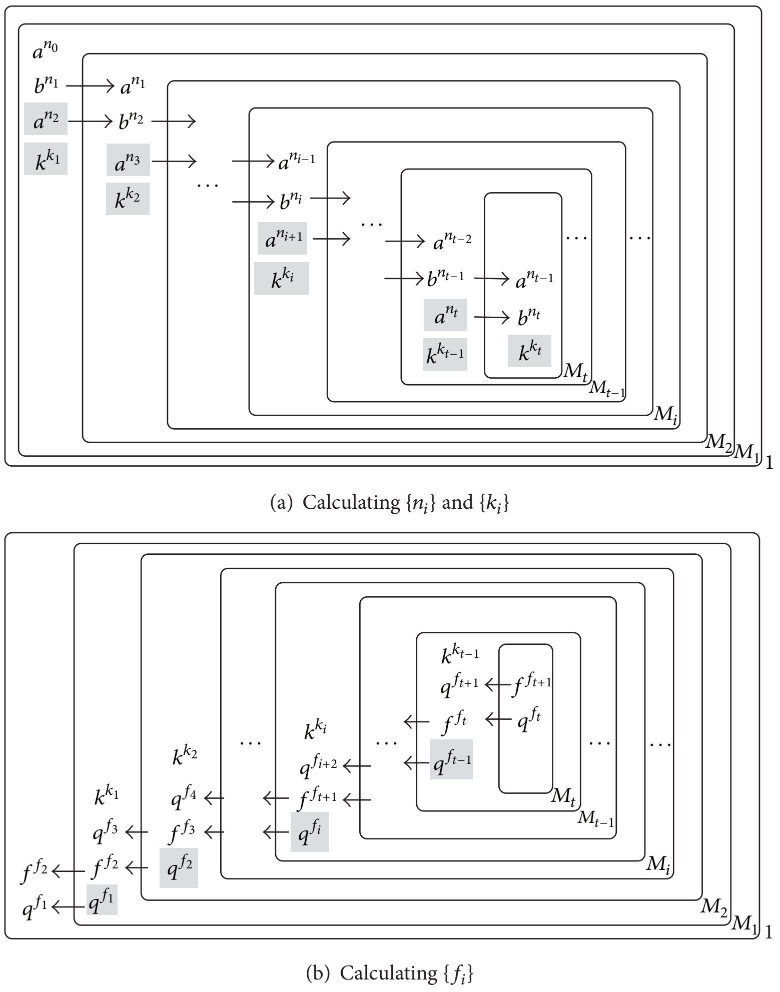


FIGURE 2: Schematic diagrams for the algorithm in Section 3.2 being implemented by the P systems.

As shown in Figure 2(a), the procedure of calculating $\{n_i\}$ and $\{k_i\}$ is as follows (the cardinalities of the objects are the items in $\{n_i\}$ and $\{k_i\}$).

(i) In membrane M_1, multiset a^{n_0} is consumed with new multisets a^{n_2} and k^{k_1} produced by applying several rules. Furthermore, a^{n_2} and b^{n_1} are sent into membrane M_2, and k^{k_1} is kept in M_1.

(ii) When a^{n_2} and b^{n_1} are sent into M_2, they are converted to b^{n_2} and a^{n_1}, respectively. In membrane M_2, a^{n_1} is consumed with new multisets a^{n_3} and k^{k_2} produced. Furthermore, a^{n_3} and b^{n_2} are sent into membrane M_3, and k^{k_2} is kept in M_2.

(iii) Generally, in membrane M_i $(2 \le i \le t-1)$, multisets a^{n_i} and $b^{n_{i-1}}$ coming from M_{i-1} are transformed to b^{n_i} and $a^{n_{i-1}}$, respectively. And then $a^{n_{i-1}}$ is consumed with the new multisets $a^{n_{i+1}}$ and k^{k_i} produced. Furthermore, $a^{n_{i+1}}$ and b^{n_i} are sent into membrane M_{i+1}, and k^{k_i} is kept in M_i.

(iv) Finally, a^{n_t} and $b^{n_{t-1}}$ leave membrane M_{t-1} and they are transferred to b^{n_t} and $a^{n_{t-1}}$, respectively, after arriving in membrane M_t. In membrane M_t, $a^{n_{t-1}}$ is consumed with the k^{k_t} produced.

As shown in Figure 2(b), the procedure of calculating $\{f_i\}$ is as follows (the cardinalities of the objects are the items in $\{f_i\}$).

(i) In membrane M_t, multiset $f^{f_{t+1}}$ $(f_{t+1} = 1)$ is produced and k^{k_t} (k^{k_t} is kept in membrane M_t previously) is transformed to q^{f_t} (here, $f_t = k_t$). Then, $f^{f_{t+1}}$ and q^{f_t} are sent into membrane M_{t-1}.

(ii) When $f^{f_{t+1}}$ and q^{f_t} are sent into M_{t-1}, they are converted to $q^{f_{t+1}}$ and f^{f_t}, respectively. In membrane M_{t-1}, $q^{f_{t+1}}$ and $k^{k_{t-1}}$ ($k^{k_{t-1}}$ is kept in membrane M_{t-1} previously) are consumed with multisets $q^{f_{t-1}}$ produced. Then, $q^{f_{t-1}}$ and f^{f_t} are sent into membrane M_{t-2}.

(iii) Generally, membrane M_{i+1} sends $q^{f_{i+1}}$ and $f^{f_{i+2}}$ into membrane M_i. $q^{f_{i+1}}$ and $f^{f_{i+2}}$ are transformed to $f^{f_{i+1}}$ and $q^{f_{i+2}}$ after arriving in membrane M_i. Then $q^{f_{i+2}}$

and k^{k_i} (k^{k_i} is kept in membrane M_i previously) are consumed with new multisets q^{f_i} produced.

(iv) When membrane M_1 sends q^{f_1} and f^{f_2} into membrane 1, the cardinalities of objects f and q compose the result of the reduction, namely, f_2/f_1.

For convenience, we have some conventions in the rest of the paper as follows.

(i) The rules should have priority, and they are described as the form $(U \rightarrow V, \varphi)$, where $U \rightarrow V$ is rewritten rule, and φ indicates the priority. The smaller value φ is set, higher priority the corresponding rule will have. When $\varphi = 1$, the corresponding rule will have the highest priority. For example, there are two rules in membrane $M_1 : r_1 : (ab \rightarrow x, 1)$ and $r_2 : (ay \rightarrow ad, 2)$, and the priority of r_1 is higher than r_2, so the r_1 will be applied firstly when both of them can be applied.

(ii) The created membranes named $M_2, M_3, \ldots$, and M_t and the rules in all of them are the same as the ones in membrane M_1.

(iii) The objects appearing in the rest of the paper have the same meaning, so they will not be explained any more once they are introduced previously.

4.2. The Rules for Fraction Reduction. In this subsection the rules in the P systems for fraction reduction will be discussed in detail. There are two kinds of rules: one is in membrane 1 and the other is in membranes $M_1, M_2, M_3, \ldots$, and M_t.

According to Section 3, we know that the fraction reduction mainly includes calculating $\{n_i\}$, $\{k_i\}$, and $\{f_i\}$. So membrane M_1 and the created membranes $M_2, M_3, \ldots$, and M_t should carry out the computations including division (for calculating $\{n_i\}$ and $\{k_i\}$), multiplication and addition (for calculating $\{f_i\}$).

4.2.1. The Rules in Membrane M_1

(i) Calculating $\{n_i\}$ and $\{k_i\}$. Firstly multiset $a^m b^n c$ is put in membrane M_1. In this membrane and the created membranes, object c evolves to y for controlling the division which is used to calculate $\{n_i\}$, $\{k_i\}$, and objects a and b label the denominator and numerator: in membrane M_1, objects a and b label n_0 and n_1, respectively; in membrane M_2, objects a and b label n_1 and n_2, respectively; in membrane M_3, objects a and b label n_2 and n_3, respectively; ...; in membrane M_t, objects a and b label n_{t-1} and n_t, respectively.

The rules for calculating $\{n_i\}$ and $\{k_i\}$ should include

$$\begin{aligned}
&r_1 : (ab \longrightarrow x, 1), \qquad r_2 : (c \longrightarrow y, 1), \\
&r_3 : (ay \longrightarrow ad, 2), \qquad r_4 : (x \longrightarrow b|_d, 3), \\
&r_5 : (d \longrightarrow kc, 4), \qquad r_6 : (by \longrightarrow beg, 2), \\
&r_7 : (e \longrightarrow [i]_i|_b, 3), \qquad r_8 : (g \longrightarrow h, 3), \\
&r_9 : (x \longrightarrow (ab, \text{in})|_h, 4), \qquad r_{10} : (b \longrightarrow (a, \text{in})|_h, 4), \\
&r_{11} : (h \longrightarrow (c, \text{in}), 5), \qquad r_{12} : (y \longrightarrow kze, 3), \\
&r_{13} : (x \longrightarrow \lambda|_z, 3), \qquad r_{14} : (z \longrightarrow \lambda, 4).
\end{aligned} \tag{14}$$

If $m > n$, r_1, r_2, r_3, r_4, and r_5 should be applied in the order of $\{r_1, r_2\} \rightarrow r_3 \rightarrow \{r_4, r_5\}$ (for convenience, the rules $r_{j1}, r_{j2}, \ldots$, and r_{jk} will be represented as $\{r_{j1}, r_{j2}, \ldots, r_{jk}\}$, if they can be executed simultaneously). a and b are consumed by r_1 with x produced; it means that the values of the numerator and denominator are subtracted by n simultaneously and $(m - n)$ copies of object a will remain. Object y, which is produced by r_2, evolves to d with the help of a by applying r_3. Once d occurs, x evolves to b by applying r_4 and it means that the numerator is restored for the next division. d evolves to kc by applying r_5. This procedure may be repeated for several times. Finally, the cardinality of object k represents the quotient of the current division.

If $m < n$, r_1, r_2, r_6, r_7, r_8, r_9, r_{10}, and r_{11} should be applied in the order of $\{r_1, r_2\} \rightarrow r_6 \rightarrow \{r_7, r_8\} \rightarrow \{r_9, r_{10}, r_{11}\}$. r_1 and r_2 are applied as described previously, and $(n - m)$ copies of object b will remain. Object y, which is produced by r_2, evolves to eg with the help of b by applying r_6. By applying r_7, e triggers to a new membrane to be created in the current membrane in the presence of object b and the new membrane will be used to calculate new items in $\{n_i\}$ and $\{k_i\}$. In the presence of object h which is produced by r_8, r_9, and r_{10} will be applied: by r_9, x evolves to ab (they will be sent into the new membrane), and it means that the numerator and denominator consumed by r_1 will be restored in the created membrane; by r_{10}, b evolves to a (a will be sent into the new membrane). The applications of r_9 and r_{10} mean that the numerator becomes the new denominator and the denominator becomes the new numerator in the new membrane for the next division since the division rules will not be triggered in the case of $m < n$ (there is multiset $a^n b^m$ in the new membrane). Object c will be produced and sent into the new membrane by applying r_{11}.

If $m = n$, r_1, r_2, r_{12}, r_{13}, and r_{14} should be applied in the order of $\{r_1, r_2\} \rightarrow r_{12} \rightarrow r_{13} \rightarrow r_{14}$. y evolves to kze by applying r_{12}. In the presence of z, x will be dissolved by applying r_{13}. Then z will be dissolved by applying r_{14}. When e occurs in the innermost membrane, it means that the calculations of $\{n_i\}$ and $\{k_i\}$ will be ended.

(ii) Calculating $\{f_i\}$. The calculations of $\{n_i\}$ and $\{k_i\}$ will be terminated when object s appears and b does not appear in the innermost membrane. At this moment, the operations of multiplication and addition should be triggered to calculate $\{f_i\}$.

Concerning (10) in Section 3, we know that in membrane M_i in Figure 2(b), the copies of objects q and f will be produced at several steps. So we can design the rules to realize that multisets $f^{f_{i-1}}$ and q^{f_i} can be produced in membrane M_{i-1} while multisets f^{f_i} and $q^{f_{i+1}}$ are produced in membrane M_i. Based on this consideration, we design the rules for calculating $\{f_i\}$ and they can be applied in several membranes simultaneously. For example, in membrane M_i in Figure 2(b), several copies of objects q and f are sent into membrane M_{i-1} once they are produced and they will trigger the rules in membrane M_{i-1}. It means that the rules in membranes M_i and M_{i-1} will be applied together at the subsequent steps. Maximal parallelism is implemented in the P systems for fraction reduction.

The rules for the operations of multiplication and addition can be designed as follows:

$$\begin{aligned}
&r_{15}:(e \longrightarrow pr,4), \quad r_{16}:(r \longrightarrow (f,\text{out}),2),\\
&r_{17}:\left(k \longrightarrow (q,\text{out})|_p,1\right), \quad r_{18}:(p \longrightarrow (ojw,\text{out})\,\delta,2),\\
&r_{19}:(w \longrightarrow (ojw,\text{out}),3), \quad r_{20}:(f \longrightarrow (q,\text{out}),3),\\
&r_{21}:\left(k \longrightarrow k\,(q,\text{out})|_q,3\right), \quad r_{22}:\left(o \longrightarrow v|_q,3\right),\\
&r_{23}:(qj \longrightarrow j\,(f,\text{out}),3), \quad r_{24}:(v \longrightarrow o,4),\\
&r_{25}:(k \longrightarrow \lambda|_o,4), \quad r_{26}:(j \longrightarrow \lambda|_o,4),\\
&r_{27}:\ (o \longrightarrow \delta,5).
\end{aligned} \tag{15}$$

Rule r_{15} is only applied in the innermost membrane M_t and is responsible for e evolving to pr. Object r evolves to f by applying r_{16}, and it means that 1 is assigned to f_{t+1} in membrane M_t as shown in Figure 2(b). f and q represent the numerator and denominator, respectively. Object k evolves to q and q is sent into the outer membrane in the presence of p by applying r_{17} (it is only applied in membrane M_t); it means that k_t is assigned to f_t in membrane M_t as shown in Figure 2(b). Rule r_{18} is only applied in membrane M_t and is responsible for sending out multiset ojw to the outer membrane M_{t-1}. Simultaneously, membrane M_t is dissolved because of the occurrence of the special symbol δ when r_{18} is applied.

Except for the rules $r_{15} \sim r_{18}$, the rest of the rules are available in M_{t-1}, $M_{t-2}, \ldots,$ and M_1. By applying r_{19}, object w evolves to ojw and ojw is sent into the outer membrane for triggering rules r_{19}, r_{22}, and r_{23} in the outer membrane. Object f coming from the inner membrane evolves to q and q is sent into the outer membrane by applying r_{20}. Rules r_{21} and r_{22} are applied to generate objects q, k, and v both in the presence of object q and the new generated object q is sent into the outer membrane. Rule r_{23} is applied to generate objects j and f (f is sent into the outer membrane). Rules r_{17}, r_{20}, r_{21}, and r_{23} are responsible for calculating $\{f_i\}$ as shown in Figure 2(b). Object v evolves to o by applying r_{24} for triggering rules $r_{25} \sim r_{27}$. Rules r_{25} and r_{26} are applied to consume objects k and j completely in the presence of object o. Rule r_{27} is applied to dissolve the current membranes.

4.2.2. *The Rule in Membrane 1.*

There is only one rule in membrane 1 and it is responsible for keeping the final results and dissolving the objects coming from membrane M_1. The rule can be designed as

$$r_{28}:(ojw \longrightarrow \lambda,1), \tag{16}$$

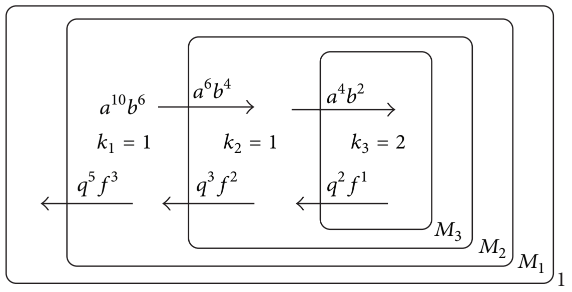


Figure 3: Schematic diagram for reducing 6/10 by the P system.

where objects o, j, and w sent from membrane M_1 are dissolved.

Owing to maximal parallelism, the complexity of the P systems for fraction reduction must be not more than $O(\log n)$.

4.3. *The Instance for Fraction Reduction.*

In this subsection, we will give an instance to show how to implement the fraction reduction in the P system designed previously. For example, 6/10 can be reduced by the P system as shown in Figure 3.

The rules in this P system can be applied as follows.

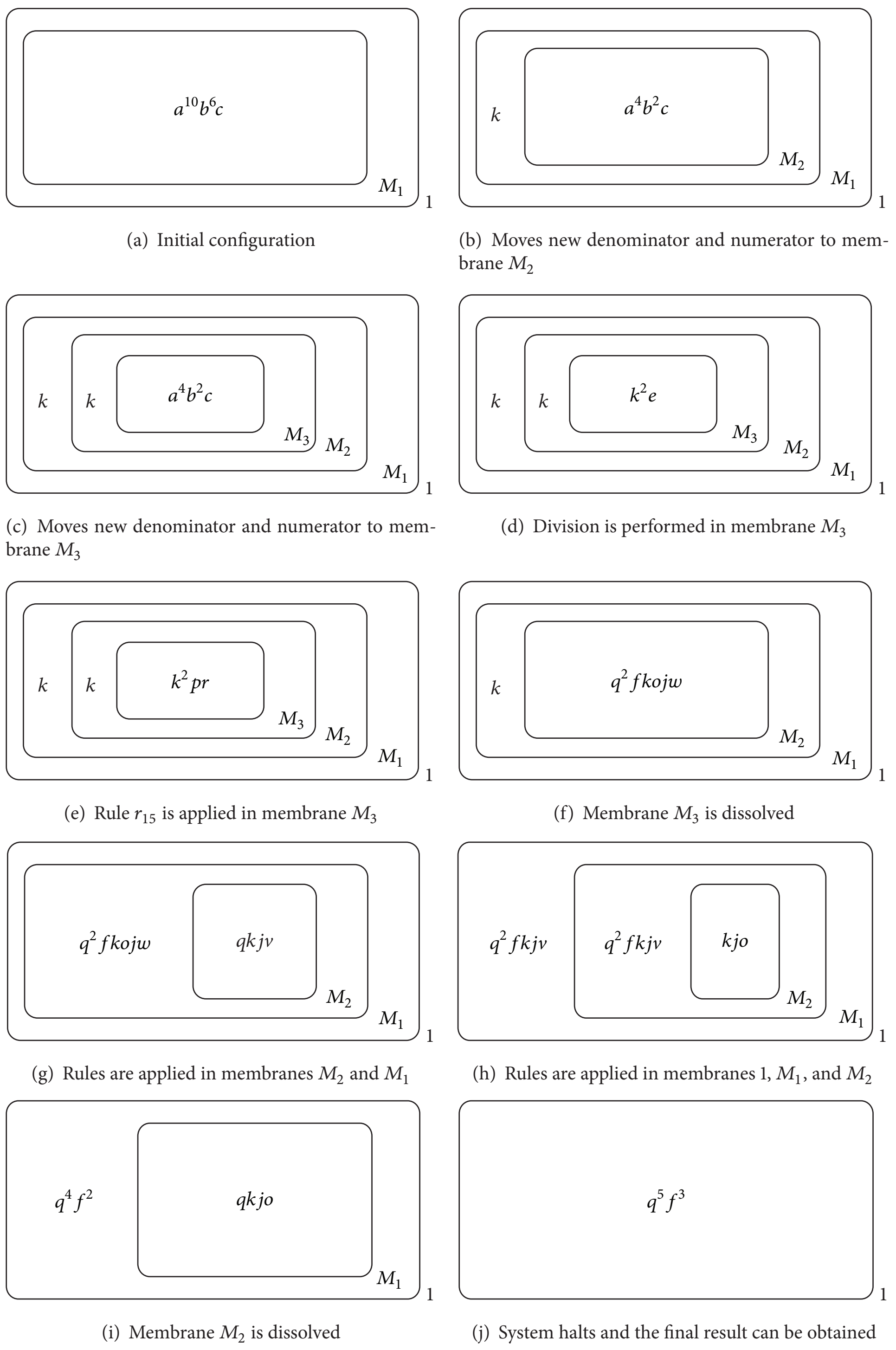
4.3.1. *Initial Configuration.*

Firstly multiset $a^{10}b^6c$ is put in membrane M_1, as Figure 4(a) shows: the cardinality of object a is 10, and it is the denominator of the fraction; the cardinality of object b is 6 and it is the numerator.

4.3.2. *Calculating $\{n_i\}$ and $\{k_i\}$*

(i) In Figure 4(a), rule r_1 is applied 6 times to generate multiset x^6 until b^6 is consumed completely, and rule r_2 is applied to generate y at the same time. Then only rule r_3 can be applied to generate multiset ad. Rule r_4 is applied to restore x^6 to b^6 in the presence of object d. At the same time, r_5 is applied to generate multiset ck and there is multiset a^4b^6ck in membrane M_1. In this case, due to the fact that the cardinality of a is less than the one of b, the available rules are applied in the order of $\{r_1, r_2\} \rightarrow r_6 \rightarrow \{r_7, r_8\} \rightarrow \{r_9, r_{10}, r_{11}\}: r_1$ is applied 4 times to consume a^4 completely; r_6 is applied to generate multiset beg; by applying r_7, a new membrane is created with the label M_2; by applying r_9, r_{10}, and r_{11}, x^4 and b^2 are sent into the new created membrane as the new numerator and denominator with object h also sent into the new membrane (see Figure 4(b)).

(ii) At this moment, there is multiset a^6b^4c in membrane M_2 (see Figure 4(b)). Hence, the available rules are applied in the order of $\{r_1, r_2\} \rightarrow r_3 \rightarrow \{r_4, r_5\}$ to generate multiset a^2b^4ck. Owing to the fact that the cardinality of a is less than the one of b, the available rules are applied in the order of $\{r_1, r_2\} \rightarrow r_6 \rightarrow \{r_7, r_8\} \rightarrow \{r_9, r_{10}, r_{11}\}$: by applying r_7, a new



(a) Initial configuration

(b) Moves new denominator and numerator to membrane M_2

(c) Moves new denominator and numerator to membrane M_3

(d) Division is performed in membrane M_3

(e) Rule r_{15} is applied in membrane M_3

(f) Membrane M_3 is dissolved

(g) Rules are applied in membranes M_2 and M_1

(h) Rules are applied in membranes 1, M_1, and M_2

(i) Membrane M_2 is dissolved

(j) System halts and the final result can be obtained

FIGURE 4: The procedure of reducing 6/10 in the designed P system.

membrane is created with the label M_3; by applying r_9, r_{10}, and r_{11}, x^2 and b^2 are sent into the new created membrane as the new numerator and denominator with object h also sent into the new membrane (see Figure 4(c)).

(iii) There is multiset a^4b^2c in membrane M_3; the available rules are applied in the order of $\{r_1, r_2\} \rightarrow r_3 \rightarrow \{r_4, r_5\}$ to generate multiset a^2b^2ck. Due to the fact that the cardinality of a equals the one of b, the available rules are applied in the order of $\{r_1, r_2\} \rightarrow r_{12} \rightarrow r_{13} \rightarrow r_{14}$: r_{12} is applied to generate multiset kze; r_{13} is applied 2 times to consume x^2 completely; then z will be dissolved by applying r_{14} (see Figure 4(d)).

4.3.3. Calculating $\{f_i\}$

(i) There is multiset k^2e in membrane M_3, and r_{15} is applied to generate multiset pr (see Figure 4(e)).

(ii) There is multiset k^2pr in membrane M_3 now. At this moment, r_{16} is applied to generate f and f is sent into the outer membrane. Then r_{17} is applied 2 times to generate q^2 and q^2 is sent into the outer membrane in the presence of object p. Rule r_{18} is applied to generate multiset $ojw\delta$ and ojw is sent into the outer membrane. Simultaneously, membrane M_3 is dissolved (see Figure 4(f)).

(iii) There is multiset q^2fkojw in membrane M_2. The following rules can be applied in a step: r_{19} and r_{20} are applied to generate objects q, o, j, and w (all of them are sent into the outer membrane); r_{21} and r_{22} are applied to generate objects q, k, and v both in the presence of object q and the new generated object q is sent into the outer membrane; r_{23} is applied to generate objects j and f (f is sent into the outer membrane) (see Figure 4(g)).

(iv) There are multisets q^2fkojw and $qkjv$ in membranes M_1 and M_2, respectively. In membrane M_1, the following rules can be applied in a step: r_{19} and r_{20} are applied to generate objects q, o, j, and w (all of them are sent into the outer membrane); r_{21} and r_{22} are applied to generate objects q, k, and v both in the presence of object q and the new generated object q is sent into the outer membrane; r_{23} is applied to generate objects j and f (f is sent into the outer membrane). Simultaneously in membrane M_2, the available rules can be applied: r_{21} is applied to generate multiset kq in the presence of object q and the new generated object q is sent into the outer membrane; r_{23} is applied to generate multiset fj and f is sent into the outer membrane, and r_{24} is applied to generate object o at the same time (see Figure 4(h)).

(v) There are multisets q^2fojw, q^2fkjv, and kjo in membranes 1, M_1, and M_2, respectively. In membrane 1, rule r_{28} is applied to consume multiset ojw completely. In membrane M_1, the following rules can be applied in a step: r_{20} is applied to generate object q (q is sent into the outer membrane); r_{21} is applied to generate objects q and k in the presence of object q and the new generated object q is sent into the outer membrane; r_{23} is applied to generate objects j and f (f is sent into the outer membrane); r_{24} is applied to generate object o. In membrane M_2, rules r_{25} and r_{26} are applied to consume objects k and j completely; then membrane M_2 is dissolved after applying r_{27} (see Figure 4(i)). The aforementioned rules in membranes 1, M_1, and M_2 are applied simultaneously.

(vi) There are multisets q^4f^2 and $qkjo$ in membranes 1 and M_1, respectively. In membrane M_1, the following rules can be applied in a step: r_{20} is applied to generate object q (q is sent into the outer membrane); r_{21} and r_{22} are applied to generate objects q, k, and v both in the presence of object q and the new generated object q is sent into the outer membrane; r_{23} is applied to generate objects j and f (f is sent into the outer membrane). Then there is multiset kjv in membrane M_1, so the following rules can be applied: r_{24} is applied to generate object o; r_{25} and r_{26} are applied to consume objects k and j completely; membrane M_1 is dissolved after applying r_{27} (see Figure 4(j)).

(vii) There is multiset q^5f^3 in membrane 1 (see Figure 4(j)). At this time, no rules can be applied, so the whole system halts. The cardinalities of q and f represent the values of the denominator and numerator, respectively, so the final result of reducing 6/10 is 3/5.

5. Conclusions

Fraction (rational number) computing is foundational in most of the computing models and systems, and the computation results of the fractions often need to be reduced to lighten the load of the subsequent computations. This paper proposes and proves a new suitable reduction method and implements it in P systems. Furthermore, we give an instance to illustrate how to carry out the fraction reduction effectively in this system. For the fact that the rational number can be given by the form of fraction, whose numerator and denominator can be represented by multisets, respectively, our work will contribute to implementing the computation of the rational numbers in P systems. Further, we will research the signed fraction reduction in P systems and the fraction reduction in the case that the denominator or numerator is 0.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper is supported by the Fundamental Research Funds for the Central Universities (no. CDJZR13185502), the National Science Foundation for Young Scholars of China (Grant no. 61201347), and Natural Science Foundation Project of CQ CSTC (2012jjA40022, 2011jjA40027).

References

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

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

[3] G. Păun, "Computing with membranes: attacking NP-complete problems," in *Proceeding of the 2nd International Conference on Unconventional Models of Computation*, pp. 94–115, Springer, London, UK, 2001.

[4] A. Vitale, G. Mauri, and C. Zandron, "Simulation of a bounded symport/antiport P system with Brane calculi," *BioSystems*, vol. 91, no. 3, pp. 558–571, 2008.

[5] A. Alhazov, C. Bonchis, G. Ciobanu, and C. Izbaşa, "Encodings and arithmetic operations in P systems," in *Proceedings of the 4th Brainstorming Week on Membrane Computing*, pp. 84–611, Sevilla, Spain, 2006.

[6] C. Mart, G. Păun, J. Pazos et al., "Tissue P systems," *Theoretical Computer Science*, vol. 296, no. 2, pp. 295–326, 2003.

[7] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta Informaticae*, vol. 71, no. 2-3, pp. 279–308, 2006.

[8] G. Naranjo, A. Miguel, M. J. Pérez-Jiménez et al., "On the degree of parallelism in membrane systems," *Theoretical Computer Science*, vol. 372, no. 2, pp. 183–195, 2007.

[9] O. H. Ibarra, "On membrane hierarchy in P systems," *Theoretical Computer Science*, vol. 334, no. 1–3, pp. 115–129, 2005.

[10] A. Atanasiu and C. Martín-Vide, "Arithmetic with membranes," *Romanian Journal of Information Science and Technology*, vol. 4, no. 1, pp. 5–20, 2001.

[11] P. Guo and J. Chen, "Arithmetic operation in membrane system," in *Proceedings of the 1st International Conference on BioMedical Engineering and Informatics (BMEI '08)*, pp. 231–234, Sanya, China, May 2008.

[12] G. Ping and Z. Haiyan, "Arithmetic operation in single membrane," in *Proceedings of the International Conference on Computer Science and Software Engineering (CSSE '08)*, pp. 532–535, Wuhan, Hubei, China, December 2008.

[13] P. Guo and M. Luo, "Signed numbers arithmetic operation in multi-membrane," in *Proceedings of the 1st International Conference on Information Science and Engineering (ICISE '09)*, pp. 393–396, Nanjing, China, December 2009.

[14] P. Guo and S. Liu, "Arithmetic expression evaluation in membrane computing with priority," *Advanced Materials Research*, vol. 225-226, pp. 1115–1119, 2011.

[15] P. Guo, H. Z. Chen, and H. Zheng, "Arithmetic expression evaluations with membranes," *Chinese Journal of Electronics*, vol. 23, no. 1, pp. 55–60, 2014.

[16] P. Guo, H. Zhang, H. Z. Chen, and J. X. Chen, "Fraction arithmetic operations performed by P systems," *Chinese Journal of Electronics*, vol. 22, no. 4, pp. 689–694, 2013.

[17] C. Martín-Vide, G. Păun, A. Rodríguez-Patón et al., "On P systems with membrane creation," *Computer Science Journal of Moldova*, vol. 9, no. 2, article 26, 2001.