


## Article

# P System with Fractional Reduction

Hai Nan <sup>1</sup>, Yumeng Kong <sup>1</sup>, Jie Zhan <sup>1</sup>, Mingqiang Zhou <sup>2,\*</sup>  and Ling Bai <sup>1,\*</sup>

<sup>1</sup> College of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China; stillwater@cqut.edu.cn (H.N.); kkym@stu.cqut.edu.cn (Y.K.); zhanjie@stu.cqut.edu.cn (J.Z.)

<sup>2</sup> College of Computer Science, Chongqing University, Chongqing 400044, China

\* Correspondence: zmqmail@cqu.edu.cn (M.Z.); bl8344@cqut.edu.cn (L.B.)

**Abstract:** Membrane computing is a branch of natural computing, which is a new computational model abstracted from the study of the function and structure of living biological cells. The study of numerical computation based on membrane computation has received increasing attention in recent years, where maximum parallelism in the execution of evolutionary rules plays an important role in improving the efficiency of numerical computation. Numbers in numerical computation are usually represented as decimals or fractions, and this paper investigates the fundamental problem in fraction representation and operations—fraction simplification. By improving the parallelization of two traditional fractional reduction algorithms, we design the corresponding fractional reduction class cells P System  $\Pi_1$  and P System  $\Pi_2$ . Combining these two P Systems, this paper designs P System  $\Pi_3$ . The feasibility and effectiveness of the P System designed in this paper are verified experimentally with the simulation software UPSimulator, and the characteristics and application scenarios of the three P Systems are analyzed.

**Keywords:** more phase derogation algorithm; division algorithm; fraction simplification; membrane computation; parallel computation



**Citation:** Nan, H.; Kong, Y.; Zhan, J.; Zhou, M.; Bai, L. P System with Fractional Reduction. *Appl. Sci.* **2023**, *13*, 8514. <https://doi.org/10.3390/app13148514>

Academic Editor: Rocco Zaccagnino

Received: 10 June 2023

Revised: 11 July 2023

Accepted: 21 July 2023

Published: 23 July 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Membrane computing (also known as P System) is a new branch of natural computing [1] introduced by Prof. Gh. Păun in a 1998 research paper and is an abstraction of a new model of computing based on the study of the function and structure of living cells in living organisms. The skin of a whole organism is equivalent to an entire computational System, which contains individual computational units. The membrane model of computing divides each biological cell into a hierarchy of regions, each of which is called a membrane; i.e., regions correspond to membranes. Each region contains a multiset of objects that evolve according to the evolutionary rules within the region, and the evolutionary rules vary from region to region. Membrane computation is based on the abstraction of living cells, and each computational unit is able to perform its own computation. Due to the extremely large number of cell membranes and the small amount of energy required to drive them, one of the greatest advantages of membrane computation is that it allows for maximum parallelism in the computation. Although computers nowadays have a profound impact on all aspects of human society, they still have limitations in their ability to process information. Due to their shortcomings, such as small storage capacity, slow computing speed and low intelligence, electronic computers are powerless to deal with many real-life challenges, such as NP problems. In contrast, membrane computing brings a finite, discrete, distributed, parallel, and hierarchical or mesh-based model of information processing to computer science [2]. Gh. Păun has demonstrated in the literature [3] that membrane computing has the equivalent computing power to Turing machines, and its powerful parallel computing capability can effectively solve the bottlenecks currently faced by electronic computers.

There is a proliferation of international research in the field of membrane computing, which has greatly facilitated the intersection and penetration of mathematics, computer science and biology. The research on membrane computing and membrane computers is developing rapidly, and great progress has been made in such areas as membrane computing applied to the solution of some NP-complete problems, membrane computing in biological laboratories, language Systems for membrane computers, associative memory problems, and the application of membrane computing to cryptography. For example, in solving NP problems, [4] proposed a new solution based on membrane computing and the dynamic window approximation algorithm, which enables the model to calculate the optimal motion command of the robot in logarithmic time and avoid obstacles in real time. Membrane computing is abstracted from living cells in organisms and is also applied to biology. In [5], it improved the implementation of biological processes, including Mitogen-activated protein kinase (MAPK), by virtue of its biological characteristics. In [6], it introduced the application of membrane computing in systems and Synthetic biology. Ref. [7] provided a detailed explanation of the extensive applications of membrane computing in biology, computer science, and linguistics for scholars to conduct in-depth learning and explore new ideas.

However, to achieve a general-purpose computer, it must also be able to implement basic arithmetic operations, such as the four operations of addition, subtraction, multiplication and division. Arithmetic operations are considered the basic operations for many complex operations, and therefore, to solve more complex problems, arithmetic operations need to be fully investigated first. Ref. [8] implemented arithmetic operations based on a designed arithmetic P System, but its membrane System structure was complex and did not make full use of the maximum parallelism of membrane computation; literature [9] designed a natural coding-based arithmetic P System to implement arithmetic operations, which greatly simplified the membrane System structure; Ref. [10] designed a multi-layer membrane P System to implement unsigned quadratic operations, which reduced the computational complexity; Ref. [11] designed a single-layer membrane P System to implement arithmetic operations, further simplifying the membrane structure and improving computational efficiency; Ref. [12] designed a multi-layer membrane P System to implement arithmetic operations with signed numbers, improving the application range and execution efficiency of basic operations; Refs. [13–15] designed a single-layer membrane P System to implement expression evaluation in the domain of integers; Ref. [16] used the P System implemented basic arithmetic operations in the domain of rational numbers, expanding the scope of application of arithmetic operations in P System to further enhance the computing power of biological computers.

Since all numbers in the Rational number field can be expressed as fractions, the results of arithmetic operations in the Rational number field may not be in the simplest form. If these results continue to be used in the new operation process, the fraction needs to be initialized first and reduced to its simplest form. In addition, from the perspective of mathematical rigor, the numerical meanings of  $1/2$  and  $1349/2698$  are the same. In the P System, the latter uses much more objects than the former. If used in other arithmetic P Systems, the large number calculation process will also be more complex, reducing system efficiency. In precise calculations, using  $1/3$  will be more accurate than  $0.33333$  and easier to represent in the P System. For this reason, scholars have conducted research on the problem of fractional simplification in order to simplify the process and obtain more accurate results when applying the simplification results to other arithmetic operations [17]. In [18], the authors studied a hybrid code-based arithmetic P System based on a cell-like P System and used fractional simplification as an example to simplify the arithmetic operations on large operands in the P System and to achieve fractional simplification after arithmetic operations on real numbers expressed as fractions in the P System. Its method used multi-layer membrane nesting, and the objects [19] inside each layer of membrane are related to the objects generated by its parent membrane, so this is a highly serialized P System that does not fully utilize the maximum parallelism of membrane computing. However, the solution

to the problem of fractional simplification using membrane computing is not sufficient, and currently, there is no other literature that can fully utilize the characteristics of membrane computing to design a more suitable P System. In response to this phenomenon, we also study the issue of fractional reduction in this article, considering partial improvements on the basis of the more phase derogation method and attempting to find suitable solutions in the two traditional reduction methods of more phase derogation and division to further improve the parallelism of the P System, thereby improving the overall efficiency of arithmetic operations.

To this end, this paper designs a fraction simplification P System based on rule priorities, and the specific article is structured as follows:

Section 1 first introduces research on membrane computation in arithmetic operations, then presents the purpose and significance of this paper, and finally gives the organization of this paper.

Section 2 begins with an introduction to the biological basis of membrane computation, followed by a definition of the cell-like P System.

Section 3 introduces the principles and algorithms of two methods, namely, the more phase derogation algorithm and the division algorithm, respectively, and improves some parts of the more phase derogation algorithm by combining them with applications in the P System.

Section 4 designs the corresponding cell-like P System based on the more phase derogation algorithm and division algorithm methods, respectively, and proposes a new method to combine the two to achieve fractional simplification. Detailed rules are introduced for these three P Systems, and the execution flow of the rules is detailed with specific examples.

Section 5 verifies the execution efficiency of the three from an experimental point of view, using the UPSimulator simulation tool for a large amount of data. Firstly, experiments are conducted for the  $\xi$  values in the combined method, and the optimal solution is obtained and then compared with the two original methods of the more phase derogation algorithm and the division algorithm, and finally, the experimental conclusions are drawn.

Section 6 summarizes the work accomplished in this paper and presents the problems that need to be improved in the future.

## 2. Fundamentals of Membrane Computing

### 2.1. Biological Basis of Membrane Computation

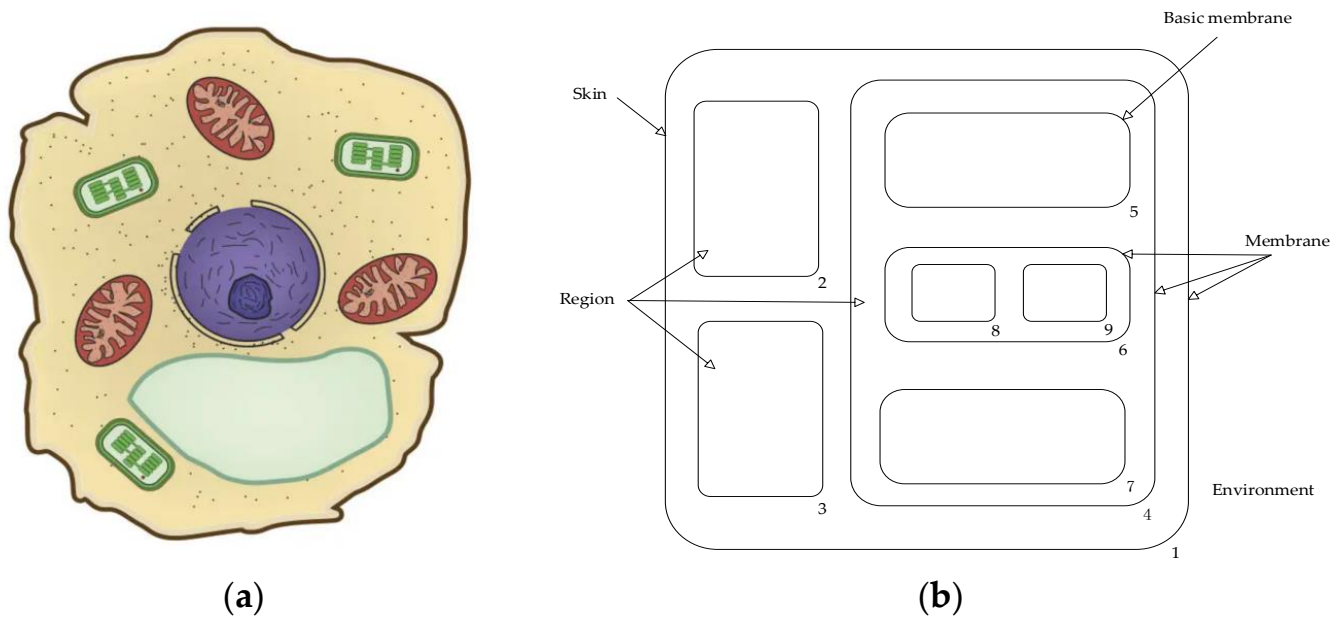
The work in this paper is based on the cell-like P System [19], so the basics of the cell-like P System are introduced first. The structure of the cell-like P System is shown in Figure 1. A schematic diagram of a cell is shown in Figure 1a, from which the membrane structure is abstracted. The membrane structure, as shown in Figure 1b, consists of the skin and the membranes within the skin arranged in a hierarchical structure. These membranes divide the interior of the cell into regions with a hierarchical structure, with the boundaries of each region being called membranes. The skin is the outermost membrane, which separates the P System from the external environment. The area outside the skin is called the environment. A basic membrane is a membrane that contains no other membrane inside.

Membrane calculations do not simply simulate the structure and function of the cell but abstract the most basic essentials. Membranes play a very important role in the structure and function of the cell by:

- (1) The skin separates the cell from its external environment, creating an enclosed area inside the cell that contains the internal space of the cell.
- (2) The inner membrane divides the cell into regions with a hierarchical structure, within each of which local biochemical reactions take place.
- (3) The cell membrane acts as a communication channel between regions.

There are two general representations of membrane structure: the generalized table and the tree structure.

Generalized table: A membrane is represented by  $[ ]$  and the ordinal number of the membrane (represented by a number) is indicated by a subscript of  $[ ]$ , i.e.,  $[ ]_i$  can represent basic membrane  $i$ ;  $[ [ ]_i ]_j$  indicates that membrane  $j$  contains membrane  $i$  within it.



**Figure 1.** Structure of the cell-like P System. (a) Cell membrane; (b) Abstraction of membranes.

Tree structure: the nested properties of membranes can be conveniently described by means of a tree structure, i.e., the skin membrane is represented by the root of the tree, and the leaves of the tree represent the basic membrane.

## 2.2. Definition of the Cell-Like P System

The cell-like P System is one of the most fundamental and earliest proposed models for membrane computation [19]. A cell-like P System  $\Pi$  of degree  $m$  ( $m \geq 1$ ) is defined as:

$$\Pi = (V, O, H, \mu, \omega_1, \dots, \omega_m, R_1, \dots, R_m, i_0) \quad (1)$$

Among them,

- (1)  $V$  is a finite, non-empty alphabet whose elements are objects;
- (2)  $O \subseteq V$  is the set of output objects;
- (3)  $H$  is the set of membrane markers,  $H = \{1, 2, \dots, m\}$ ;
- (4)  $\mu$  is the membrane structure containing  $m$  membranes, where  $m$  is called the degree of  $\Pi$ ;
- (5)  $\omega_i \in V^* (1 \leq i \leq m)$ , denotes the multiset of objects contained inside membrane  $i$ . For example, if membrane  $i$  contains 5  $a$ -objects and 3  $b$ -objects, then we have  $\omega_i = a^5b^3$ .  $V^*$  is the set of arbitrary strings consisting of the characters in  $V$ . If  $\omega_i = \lambda$ , then it means that no objects exist inside membrane  $i$ .
- (6)  $R_i (1 \leq i \leq m)$  is a finite set of evolutionary rules inside region  $i$  in membrane structure  $\mu$ . The evolutionary rules are a binary set  $(u, v)$ , usually written  $u \rightarrow v$ , where  $u$  is a string in  $V^*$ ,  $v = v'$  or  $v = v'\delta$ , where  $v'$  is a string on the set  $\{a_{here}, a_{out}, a_{in_j} \mid a \in V, 1 \leq i \leq m\}$ , and here indicates that object  $v'$  is still in the the region where the rule is used; out indicates that object  $v'$  is moved out of the current region into the outer membrane containing the current membrane; and  $in_j$  indicates that object  $v'$  will be moved into membrane  $j$ , which is directly contained by membrane  $i$ .  $\delta$  is a special character not belonging to  $V$ . When a rule contains  $\delta$ , the membrane is dissolved after execution of the rule.
- (7)  $i_0$  is the output region of the membrane structure that is used to hold the results of the computation,  $i \in H$ .

In addition to the rules introduced above, it is necessary to consider rules for creating membranes of the form:  $e \rightarrow [W]_i$ , where  $e \in V, W \in V^+, i$  is a number on a given list of

ordinal numbers. This rule indicates that the object  $e$  creates a membrane labelled  $i$  and the objects inside the membrane are denoted by  $W$  [20].

In every membrane structure, the implementation of evolutionary rules will be guided by two principles:

(1) Non-determinism

The P System will follow the non-determinism principle when executing evolutionary rules, which means that when multiple rules can be satisfied simultaneously, the rules chosen by the P System to execute are non-deterministic [3].

(2) Maximum parallelism

In the P System, each step of the computation follows the principle of maximum parallelism, meaning that all rules that can be executed must be executed at the same time.

### 3. Principles and Algorithms of Fraction Simplification

Fractional reduction is an indispensable part of arithmetic operations. This paper designs the membrane system based on two traditional methods of fractional reduction: the more phase derogation algorithm and the division algorithm method, and Section 3 focuses on the principles of both and their algorithms.

#### 3.1. The More Phase Derogation Algorithm

##### 3.1.1. Principle

The more phase derogation algorithm is from the ancient Chinese mathematical treatise “the Nine Chapters on the Mathematical Art”, and can be used to find the greatest common divisor of two numbers.

The exact steps of this algorithm are as follows:

Step 1: Given any two positive integers, first determine if they are both even; if so, use 2 to approximate; if not, perform step 2.

Step 2: The larger number in the numerator denominator is subtracted from the smaller number; the resulting difference is compared to the smaller number; and the number is reduced by the larger number. Continue this operation until the resulting subtraction and difference are equal.

The rule  $a \mid b$  means that  $a$  divides  $b$  ( $a$  is a divisor of  $b$ ).  $\gcd(a, b)$  means the greatest common divisor of  $a, b$ .

$a \bmod b$  means that  $a$  divides  $b$  by the remainder.

It may be useful to set  $A > B$ . Let the greatest common divisor of  $A$  and  $B$  be  $X$  so that  $A = aX$  and  $B = bX$ , where  $a$  and  $b$  are both positive integers and  $a > b$ .  $C = A - B$ , then we have  $C = aX - bX = (a - b)X$ .

Since both  $a$  and  $b$  are positive integers,  $C$  is also divisible by  $X$ , i.e., the greatest common divisor of  $A, B$  and  $C$  is  $X$ . So  $\gcd(A, B) = \gcd(B, A - B)$ , which means that the final result is its greatest common divisor.

Then the numerator and denominator are divided by the maximum common divisor  $X$ , which is obtained by the more phase derogation method, to obtain the final result.

##### 3.1.2. Improvement of the More Phase Derogation Algorithm

Although it is relatively simple to find the maximum convention by the approximate method of the more phase derogation algorithm, the steps of multiplication and simplification after finding the maximum convention are improved in this paper because the membrane rules and the complexity required for multiplication and division in the P System are much greater than those for addition and subtraction.

The specific steps are as follows:

Step 1: Given any two positive integers, subtract the smaller number from the larger number in the numerator denominator, then compare the resulting difference with the smaller number and record the relationship using  $z$ . If the difference is greater than the minus number, then  $z$  is recorded as 0, and if the difference is less than the minus number,

then it is recorded as 1, and the larger number is used to reduce the number. Continue this procedure until the subtractor and the difference are equal.

Step 2: Design two other sequences  $\{p_i\}$  and  $\{q_i\}$ , starting with the last ordinal number  $n$  of the array  $z$ , so that the initial value of  $p_n$  is 2 and the initial value of  $q_n$  is 1. Then gradually assign values to the sequences:  $p_{n-1} = p_n + q_n$ , if  $z$  is 0, then  $q_{n-1} = q_n$ ; if  $z$  is 1, then  $q_{n-1} = p_n$ . Keep repeating this step, and the final approximate result is  $p_1/q_1$ .

### 3.1.3. Algorithms

Suppose we want to reduce the fraction  $\frac{x}{y}$  ( $0 < y < x$ ). Based on the principle of Section 3.1.1 and the refinement of Section 3.1.2, the process of fraction simplification by a more phase derogation algorithm is described as follows:

- (i) Input  $x, y$  ( $0 < y < x$ );
- (ii) Calculate  $\{x_i\}, \{y_i\}, \{u_i\}, \{z_i\}$ , where  $i = 2, 3, \dots, t$ , and  $x_1 = x, y_1 = y, u_1 = x - y, z_1 = 0$ ;
- (iii) Calculate  $\{p_i\}, \{q_i\}$ , where  $i = t, t - 1, \dots, 1$ ;
- (iv) Output  $p_1, q_1$ .

The Algorithm 1 using the more phase derogation technique is as follows:

---

**Algorithm 1:** Algorithm for fraction simplification using more phase derogation.

---

**Input:**  $x, y$  ( $0 < y < x$ )

**Output:**  $p_1, q_1, p_1/q_1$  is the simplest form of  $x/y$  fractions.

**procedure**

**Input:**  $x, y$  ( $0 < y < x$ )

**Output:**  $p_1, q_1, p_1/q_1$  is the simplest form of  $x/y$  fractions.

**procedure**

```

1   $x_1 \leftarrow x, y_1 \leftarrow y;$ 
2   $z_1 \leftarrow 0;$ 
3   $u_1 \leftarrow x - y;$ 
4  while  $y_i \neq u_i$ 
5    if  $u_i < y_i$  then  $x_{i+1} \leftarrow y_i;$ 
6     $y_{i+1} \leftarrow u_i;$ 
7     $z_{i+1} \leftarrow 1;$ 
8    end if
9    if  $u_i > y_i$  then  $x_{i+1} \leftarrow u_i;$ 
10    $y_{i+1} \leftarrow y_i;$ 
11    $z_{i+1} \leftarrow 0;$ 
12   end if
13    $i \leftarrow i + 1;$ 
14    $u_i \leftarrow x_i - y_i;$ 
15 end while
16  $p_i \leftarrow 2; q_i \leftarrow 1;$ 
17 repeat
18  $i \leftarrow i - 1;$ 
19  $p_i \leftarrow p_{i+1} + q_{i+1};$ 
20 if  $z_{i+1} == 1$  then  $q_i = p_{i+1};$ 
21 end if
22 else then  $q_i = q_{i+1};$ 
23 end else
24 until  $i = 1;$ 
25 end procedure
```

---

In this algorithm, the more phase derogation algorithm is split into two parts: the complexity of the algorithm is  $O(1)$  when  $x$  is twice as long as  $y$ . For a general  $\{x_i\}, \{y_i\}$ , the time complexity of the first part of the algorithm is  $O(t)$  if the algorithm performs  $t$  subtraction operations, at which point  $u_t = y_t$ , and the first part of the more phase



derogation algorithm ends. It is easy to know that the length of the sequence  $\{z_i\}$  is the same as the number of times the first part of the algorithm is executed, so the length of the sequence  $z$  is  $t$ . After  $t$  times of execution, the value of the more phase derogation algorithm  $i$  goes to 0, and the second part of the more phase derogation algorithm ends. That is, the time complexity of this algorithm in the second part is  $O(t)$  and its total time complexity is  $O(t)$ . That is, in the average case, the time complexity of this algorithm is  $O(t)$ . If the difference between  $x$  and  $y$  is large, then the worst time complexity is  $O(\max(x, y))$ .

### 3.2. Division Algorithm

#### 3.2.1. Principle

The division algorithm, also known as Euclid's algorithm [21,22] for finding the greatest common divisor of two numbers, is considered to be the earliest algorithm in the world (300 BCE). It first appeared in Euclid's *Principia Geometrica* (Book VII, Propositions I and II), and in China, it can be traced back to the *Nine Chapters of Arithmetic*, which appeared in the Eastern Han Dynasty.

The greatest common divisor of two natural numbers is the largest positive integer that can divide them simultaneously. The division algorithm is based on the principle that the greatest common divisor of two integers is equal to the greatest common divisor of the smaller number and the remainder of the division of the two numbers.

For example, the greatest common divisor of 1254 and 390 is 6 ( $1254 = 6 \times 209$ ;  $390 = 6 \times 65$ ); the process of deriving the greatest common divisor using these two numbers is as follows:

$$1254 \% 390 = 84$$

$$390 \% 84 = 54$$

$$84 \% 54 = 30$$

$$54 \% 30 = 24$$

$$30 \% 24 = 6$$

$$24 \% 6 = 0$$

So the greatest common divisor of these two numbers is 6.

The proof of this algorithm is as follows:

Let the two numbers be  $a$  and  $b$  ( $b < a$ ) and denote by  $\gcd(a, b)$  the greatest common divisor of  $a$  and  $b$ .  $r = a \bmod b$  is the remainder of  $a$  divided by  $b$ , and  $k$  is the quotient of  $a$  divided by  $b$ . The division algorithm method is to prove that  $\gcd(a, b) = \gcd(b, r)$ .

Step 1: Let  $c = \gcd(a, b)$ , then let  $a = mc$  and  $b = nc$ ;

Step 2: It follows from the premise that  $r = a - kb = mc - knc = (m - kn)c$ ;

Step 3: It follows from the result of step 2 that  $c$  is also a factor of  $r$ ;

Step 4: It can be concluded that  $m - kn$  and  $n$  are mutually exclusive; otherwise, one can set  $m - kn = xd$  and  $n = yd$  ( $d > 1$ ), then  $m = kn + xd = kyd + xd = (ky + x)d$ , and then  $a = mc = (ky + x)dc$  and  $b = nc = ycd$ , so the maximum convention of  $a$  and  $b$  becomes  $cd$ , not  $c$ , contradicting the previous conclusion.

From this, it follows that  $\gcd(b, r) = c$ , and subsequently  $\gcd(a, b) = \gcd(b, r)$ .

#### 3.2.2. Algorithm

Suppose we want to approximate the fraction  $\frac{x}{y}$ . According to the principle of Section 3.2.1, the process to achieve fractional simplification by the division algorithm is described as follows:

- (i) Input  $x, y (0 < y < x)$ ;
- (ii) Calculate  $\{a_i\}, \{b_i\}, \{r_i\}$ , where  $i = 2, 3, \dots, t$ , and  $a_1 = x, b_1 = y, r_1 = m \% n$ ;
- (iii) Output  $k, l$ .

The division Algorithm 2 is as follows:

---

**Algorithm 2:** Algorithm for fractional reduction using the division algorithm.

---

**Input:**  $x, y (0 < y < x)$

**Output:**  $k, l, k/l$  is the simplest form of  $x/y$  fractions.

**procedure**

```

1    $X \leftarrow x, Y \leftarrow y;$ 
2    $a_1 \leftarrow X, b_1 \leftarrow Y;$ 
3   if  $a_1 < b_1$  then  $t \leftarrow a_1;$ 
4    $a_1 \leftarrow b_1;$ 
5    $b_1 \leftarrow t;$ 
6   end if
7    $t_1 \leftarrow 1, a_1 \% b_1;$ 
8   while  $r_i \neq 0$ 
9    $a_{i+1} \leftarrow b_i;$ 
10   $b_{i+1} \leftarrow r_i;$ 
11   $i \leftarrow i + 1;$ 
12   $r_i \leftarrow a_i \% b_i;$ 
13 end while
14 end procedure
```

---

In this algorithm, the complexity of the algorithm is  $O(1)$  when  $m$  is a multiple of  $n$ . Generally, for the sequence  $\{r_i\}$ , we know that  $r_{t+1} = 0$  if the algorithm performs  $t$  coset operations. Comparing the sequence  $\{r_{t+1}\}$  with the Fibonacci sequence  $\{F_i\}$ , we have  $F_0 = 1 \leq r_t$  and  $F_1 = 1 \leq r_{t-1}$ .  $r_k \geq r_{k+1} + r_{k+2}$  can be obtained by  $r_k \bmod r_{k+1} = r_{k+2} (0 \leq k \leq t-1)$ . Therefore,  $r_k \geq F_{t-k}$  can be concluded by mathematical induction. In addition, we can obtain  $m = n_0 \geq F_t$  and  $n = n_1 \geq F_{t-1}$ . That is, if our algorithm performs the remainder operation  $t$  times, then  $n$  must not be less than  $F_{t-1}$  and vice versa. According to the characteristics of the Fibonacci sequence, we have  $F_{t-1} \geq (1.618)^t / \sqrt{5}$ , which gives us  $n \geq (1.618)^t / \sqrt{5}$  and  $t \leq \log_{1.618}(\sqrt{5}n)$ . Therefore, in the worst case, the complexity of the algorithm is  $O(\log n)$ .

#### 4. Fractional Simplification P System

In membrane computing, the execution of rules is parallel. If the system meets the execution conditions of multiple rules simultaneously and there are no conflicts between the rules, these rules will be executed synchronously. After execution, continue to judge whether all rules are met in the new state. If the system still has rules that are met, continue to execute, and if not, stop running. If there is a conflict between rules (referring to the situation where an object has multiple rules that can be executed), the system will choose to execute the rules indefinitely. At this point, we can specify the priority so that the rules are executed in the order of priority. For this purpose, we designed three priority-based cellular P systems. This section will define, explain rules, and analyze examples of the three P Systems  $\Pi_1$ ,  $\Pi_2$  and  $\Pi_3$ .

In order to aid scholars when verifying the efficiency of the system, during manual simulation and simulation using simulation software UPS [23], the execution time of each rule is recorded as a time slice, and the efficiency of each P system is judged based on the number of time slices. In each time slice, the system will execute all the rules that can be executed once, until there are no rules to execute and the system stops running. At this point, the number of time slices can reflect the time consumed by the P system, thus comparing the efficiency of each P system.

In addition, when explaining the rule execution process and instance analysis in detail in the P System, if multiple objects execute a rule at the same time or if a rule generates multiple objects at the same time, in order to make the object representation clearer, we use “,” or “and” to partition between objects without corner markers. Among them, the objects with practical effects will be explained in detail in this article, while the process objects with auxiliary effects will be omitted or briefly explained.



#### 4.1. P System for Fractional Simplification of More Phase Derogation Algorithm

This section first introduces the definition and initial pattern of the P System for the more phase derogation algorithm, and then provides a detailed explanation of the complete rules of the P System and provides specific examples to demonstrate the execution process of the rules.

##### 4.1.1. Definition of the P System for More Phase Derogation Algorithm

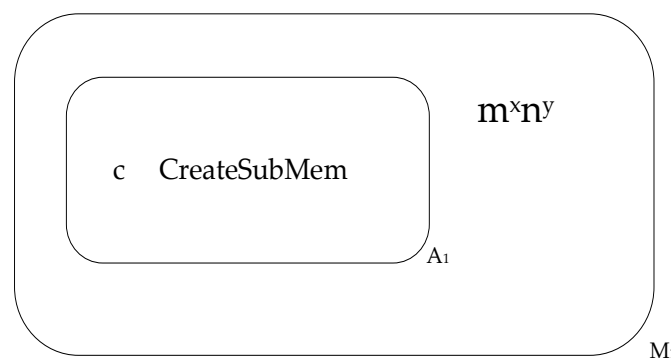
The more phase derogation P System is formalized as:

$$\Pi_1 = (V, O, H, \mu, \omega_{M_1}, \omega_{A_1}, \dots, \omega_{A_t}, R_{M_1}, R_{A_1}, \dots, R_{A_t}, i_0), \quad (2)$$

Among them,

- (1)  $V = \{m, n, c, \text{CreateSubMem}, \text{Num}, a, b, g, x, y, l, u, k, f, v\}$ ;
- (2)  $H = \{M_1, A_1, \dots, A_t\}$ , in this article, for clarity, membrane labels are expressed in the form of a combination of membrane type and subscripts.
- (3)  $\mu = \{ [ [ [ [ ]_{A_t} ] \dots ]_{A_1} ]_{M_1} \}$ , in the initial state, the P System only contains membrane  $M_1$  and membrane  $A_1$ , and membrane  $A_1$  is included in membrane  $M_1$ .  $A_i (2 \leq i \leq t)$  is dynamically generated during the execution of membrane rules, and any  $A_i$  always includes  $A_{i-1}$  in;
- (4)  $\omega_{M_1} = m^x n^y, \omega_{A_1} = c \text{CreateSubMem}$ , and  $x$  and  $y$  represent the numerator and denominator of fractions, respectively;
- (5) The  $R_{M_1}, R_{A_1}, \dots, R_{A_t}$  evolution rule finite set will be explained in Section 4.1.2;
- (6)  $i_0 = M_1$  indicates that when the entire System stops, the final result can be obtained in the membrane  $M_1$ ;

For the convenience of description, there is the following agreement in the remaining content of this section: the newly created class A membrane is named  $A_2, A_3, \dots, A_t$ , and the rules within these membranes are consistent with those within membrane  $A_1$ . Therefore, there will be no further explanation in the remaining content of this section. The initial pattern of the fraction simplification P System based on the more phase derogation algorithm is shown in Figure 2.



**Figure 2.** Initial Pattern of the More Phase Derogation Algorithm P System.

Figure 2 depicts the initial pattern of the P System using the more phase subtraction method for fractional reduction: membrane  $M_1$  is mainly used to compare the sizes of molecule  $x$  and denominator  $y$ , transfer the larger and smaller values into object  $a$  and object  $b$  of membrane  $A_1$ , respectively, and save the final calculation results and dissolve other objects transferred from membrane  $A_1$ . Membrane  $A_1$  and its dynamically generated sub membranes  $A_2, A_3, \dots, A_t$  are used for more phase reduction operations, and the values of variables such as sequence  $\{x_i\}, \{y_i\}, \{u_i\}, \{z_i\}, \{p_i\}, \{q_i\}$  are calculated (where  $i = 1, 2, \dots, t$ ). Except for membrane  $M_1$  and membrane  $A_1$ , other required membrane structures are dynamically created during the reduction process based on the actual conditions of the fraction, and the newly created membrane rules are completely consistent with the rules in membrane  $A_1$ .

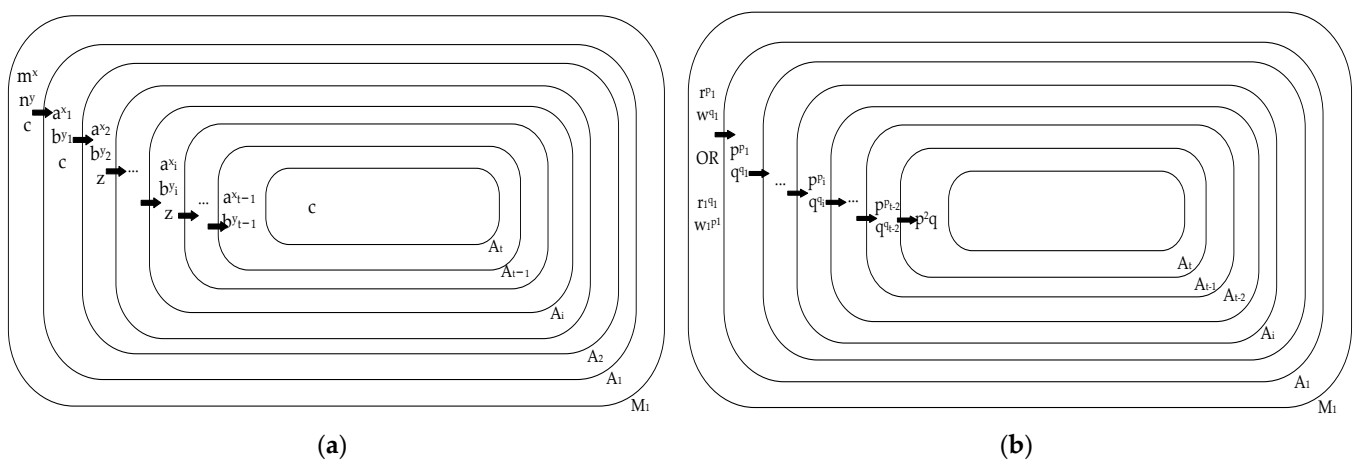
In Figure 2, objects  $m$  and  $n$  are used to represent the numerator and denominator, respectively;  $x$  and  $y$  are the numbers of objects  $m$  and  $n$ , and object  $c$  is used to trigger the execution of rules in membrane  $A_1$  and determine whether a new membrane needs to be generated based on different conditions. Membrane  $A_1$  and other newly created submembranes are used together to calculate variables in the rules. These newly created membranes are nested layer by layer: membrane  $A_2$  is created within membrane  $A_1$ , membrane  $A_3$  is created within membrane  $A_2$ , and finally, membrane  $A_t$  is created within membrane  $A_{t-1}$ .

The process of calculating sequences  $\{x_i\}$ ,  $\{y_i\}$ ,  $\{u_i\}$  is described below, and the specific execution process of membrane rules will be explained in detail in Section 4.1.2.

- (i) In the initial state, a partial rule is executed in membrane  $M_1$  to determine the magnitude of the numerator  $x$  and denominator  $y$ , passing the larger and smaller values to object  $a$  and object  $b$  in membrane  $A_1$ , respectively. For e.g., if  $x = 12$  and  $y = 9$ , then the rule is executed to generate a multiset  $a^{12}b^9$  in membrane  $A_1$ , i.e.,  $x_1 = 12$  and  $y_1 = 9$  in the corresponding sequence.
- (ii) In membrane  $A_1$ , the multiset  $a^{x_1}b^{y_1}$  is consumed by partial execution of the rule, a judgment is made as to whether  $z_1$  is produced while a new membrane  $A_2$  is produced, and the multiset  $a^{x_2}b^{y_2}$  is produced by passing some material into the submembrane. This step is repeated until no new membranes are generated when the stopping condition for more phase loss is met in membrane  $A_t$ , and the computation of the sequence  $\{x_i\}$ ,  $\{y_i\}$ ,  $\{u_i\}$  is completed.
- (iii) In general, within the membrane  $A_i$  ( $2 \leq i < t - 1$ ), the multisets  $a^{x_i}$  and  $b^{y_i}$  generated by substances transported by the membrane  $A_{i-1}$  are then transformed into new multisets into the submembrane according to the corresponding rules, respectively, until the end when  $y_t$  is equal to  $u_t$ .

#### 4.1.2. Membrane Rules for the More Phase Derogation Algorithm

According to Section 4.1.1, the fraction simplification using the more phase derogation algorithm consists of calculating the sequences  $\{x_i\}$ ,  $\{y_i\}$ ,  $\{u_i\}$ ,  $\{z_i\}$ ,  $\{p_i\}$ ,  $\{q_i\}$ . The value of each sequence indicates the number of corresponding objects in each layer of the membrane. For example,  $x_1 = 12$  means the number of  $x$  in membrane  $A_1$  is 12. Combined with the idea of Algorithm 1 in Section 3.1.3, it can be seen that membrane  $A_1$  and the dynamically created submembrane only need to use the addition and subtraction method to complete the calculation, and the specific process is shown in Figure 3.



**Figure 3.** Flowchart of the fraction simplification solution of the more phase derogation algorithm P System. (a) Flowchart of the sequence  $\{x_i\}, \{y_i\}$  by more phase derogation algorithm; (b) Flowchart of the sequence  $\{p_i\}, \{q_i\}$  by more phase derogation algorithm.

The following is a specific description of the membrane rules.

## (1) Compare the values of the numerator and denominator

According to the principle and algorithm of the more phase derogation algorithm, it is necessary to ensure that the subtracted number is larger than the subtracted number in the class A membrane, so the size of the numerator  $x$  and the denominator  $y$  are compared in the membrane  $M_1$  first. The multiplicity set in the initial state is  $m^x n^y$ , and the comparison rules are as follows:

$$\begin{aligned} r_1 : (m \ n \rightarrow g, 1) & \quad r_2 : (m \rightarrow a(a \text{ Num}, \text{in all})|_g, 2) \\ r_3 : (n \rightarrow b(a \text{ Num}, \text{in all})|_g, 2) & \quad r_4 : (g \rightarrow (a \ b \text{ Num}, \text{in all}), 3) \end{aligned}$$

- (i) When  $x > y$ , the membrane rules are executed in the following order:  $r_1 \rightarrow r_2 \rightarrow r_4$ . Execution of rule  $r_1$ , which corresponds to converting object  $n$  to  $g$ , also leaves  $x - y \ m$  in the membrane System. Executing  $r_2$  under the condition that both  $g$  and  $m$  are present backs up the difference of  $x - y$  in the membrane as a multiset  $a^{x-y}$ , while generating a multiset  $a^{x-y} \text{Num}^{x-y}$  to pass into the submembrane  $A_1$ .
- (ii) When  $x < y$ , the membrane rules are executed in the following order:  $r_1 \rightarrow r_3 \rightarrow r_4$ . Executing rule  $r_1$ , there is no remaining  $m$  in the membrane, so executing  $r_3$  backs up the excess  $n$  in the membrane as multiset  $b^{y-x}$ , while generating multiset  $a^{y-x} \text{Num}^{y-x}$  to pass into submembrane  $A_1$ .

The execution of  $r_4$  continues in both cases, passing the smaller of the two,  $g$ , to  $a$  and  $b$  in the submembrane, so that the number of objects  $a$  in membrane  $A_1$ ,  $x_1$ , is equivalent to performing a calculation of  $x - y + y$  or  $y - x + x$ . The number of objects  $b$  in membrane  $A_1$ ,  $y_1$  is the smaller of the numerator and denominator,  $a$  and  $b$  in membrane  $M_1$  still represent the numerator and denominator.

(2) Compute the sequence  $\{x_i\}, \{y_i\}, \{u_i\}, \{z_i\}$ 

After first comparing the size of the numerator denominator in  $M_1$ , the multiset  $a^{x_1} b^{y_1}$  is placed in membrane  $A_1$ , at which point the complete multiset within the membrane is  $a^{x_1} b^{y_1} c \text{Num}^{x_1} \text{CreateSubMem}$ . Within membrane  $A_1$  and other dynamically generated submembranes, object  $c$  evolves into  $k$  and  $f$ , which is used to control the loop; each layer of the generated membrane is initialized with  $a$  and  $b$ . To avoid confusion, the sequences  $\{x_i\}$  and  $\{y_i\}$  are designed to hold the number of  $a$  and  $b$  in the  $i$ th layer of the membrane, respectively, with  $x_1$  and  $y_1$  representing the initial larger and smaller values.

The rules for computing the sequence  $\{x_i\}, \{y_i\}, \{u_i\}, \{z_i\}$  are as follows:

$$\begin{aligned} r_1 : (a \ b \rightarrow x \ y, 1) \quad r_2 : (a \rightarrow l \ u|_c, 2) \quad r_3 : (c \rightarrow k \ f|_u, 2) \\ r_4 : (y \ u \rightarrow v, 3) \quad r_5 : (k \rightarrow (c, \text{in all})|_v, 4) \quad r_6 : (y \ f \rightarrow i \ (z, \text{in all})|_v, 4) \\ r_7 : (u \ f \rightarrow h \ |_v, 4) \quad r_8 : (x \rightarrow (a, \text{in all})|_i, 4) \quad r_9 : (x \rightarrow (b, \text{in all})|_h, 4) \\ r_{10} : (l \rightarrow (b, \text{in all})|_i, 4) \quad r_{11} : (l \rightarrow (a, \text{in all})|_h, 4) \\ r_{12} : (\text{CreateSubMem} \ \text{Num} \rightarrow A : a\{ \} \ |_v, 3) \\ r_{13} : (\text{Num} \rightarrow (\text{Num}, \text{in all})|_{\text{CreateSubMem}}, 3) \end{aligned}$$

The membrane rule is executed in the following order when  $x_i > y_i$ :  $r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow r_{12} \rightarrow \{r_5, r_6, r_{13}\} \rightarrow \{r_8, r_{10}\}$  or  $r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow r_{12} \rightarrow \{r_5, r_7, r_{13}\} \rightarrow \{r_9, r_{11}\}$  (for convenience, if the rule  $r_{j_1}, r_{j_2}, \dots, r_{j_k}$  can be executed simultaneously, it is denoted as  $\{r_{j_1}, r_{j_2}, \dots, r_{j_k}\}$ ).

- (i) Execution of rule  $r_1$ : In  $A_i (1 \leq i \leq t - 1)$ , objects  $a$  and  $b$  are consumed to generate the multiset  $x^{y_i} y^{y_i}$ , indicating that the numerator and denominator values are consumed in both  $y_i$  copies, leaving  $u_i (u_i = x_i - y_i)$  copies of object  $a$ . Matter  $x$  and  $y$  are used to back up the subtractor  $b$  and to compare with the difference  $u_i$ , respectively. Rule  $r_2$  is executed under the condition of the existence of object  $c$ . The multiset  $l^{u_i} u^{u_i}$  is generated by executing  $r_2$ , where  $l$  is used to backup the difference  $u_i$ , and  $u$  is used to compare the size with the subtraction  $y$ .
- (ii) Execute rule  $r_3$  under the condition that there is an object  $u$ :  $c$  generates object  $k$  and  $f$ ,  $k$  is used to control the loop, and  $f$  is used to check the size of  $y_i$  and  $u_i$ . If  $y_i > u_i$ ,

the rules are executed in the first order. If  $y_i < u_i$ , the rules are executed in the second order. When  $y_i = u_i$ , the dynamic generation of submembranes has ended, and will be discussed in the subsequent calculation of sequence  $\{p_i\}$ ,  $\{q_i\}$ , which will not be explained in detail here.

- (iii) Execute rule  $r_4$  and  $y$  and  $u$  evolve into  $v$ . Compare the size of  $y_i$  and  $u_i$  by observing which object remains within the environment. If there is still  $y$  in the membrane after evolving into  $v$ , then  $x_{i+1} = y_i$ ,  $y_{i+1} = u_i$ , and  $z_{i+1} = 1$  in the child membrane  $A_{i+1}$ ; otherwise,  $x_{i+1} = u_i$ ,  $y_{i+1} = y_i$ , and  $z_{i+1} = 0$  (i.e., there is no substance  $z$  in the child membrane). So when  $y$  is still present within the environment, rules  $r_5$  and  $r_6$  are executed with object  $v$  as catalyst.
- (iv) Before executing  $r_5$ , we must first execute  $r_{12}$  to consume *CreateSubMem* and a *Num* in the membrane to generate a child membrane, and then we can subsequently pass the object into the child membrane. Each class A membrane is initialized with a *CreateSubMem*, and *Num* is passed from the parent membrane to the submembrane; to ensure sufficient *Num*, the number of initialized *Num* is  $x_1$ . Execute  $r_5$  and  $r_6$ : object  $k$  generates a new  $c$  and sends it to the submembrane; if  $y_{i+1} \neq y_i$ , then  $y$  and  $f$  evolves to  $i$  and generates a new object  $z$  and sends it to the submembrane; then, execute  $r_{13}$  after consuming a *Num*; the remaining *Num* is passed into the submembrane for the submembrane to dynamically generate membranes.
- (v) If object  $i$  exists, it is shown that both  $a$  and  $b$  values of the submembrane need to be changed, for which rules  $r_8$  and  $r_{10}$  are executed:  $r_8$  passes the subtractive backup  $x$  to the  $a$  of the submembrane (i.e.,  $x_{i+1} = y_i$ ) and  $r_{10}$  passes the differential backup  $l$  to the  $b$  of the submembrane (i.e.,  $y_{i+1} = u_i$ ). If  $u$  is left in the environment, rules  $r_5$  and  $r_7$  are executed with object  $v$  as the catalyst. When  $h$  is generated in the environment, it is proved that  $y_{i+1} = y_i$  in the submembrane  $A_{i+1}$ . Therefore,  $r_9$  and  $r_{11}$  are executed to pass the subtractive backup to  $x$  to the submembrane  $y_{i+1}$  and the differential backup  $l$  to  $x_{i+1}$  in the submembrane, respectively. The above steps are executed several times until the dynamic generation ends when  $x_{t-1} = 2y_{t-1}$ .

When the submembrane  $A_{t-1}$  is executed to  $x_{t-1} = 2y_{t-1}$ , the order of rule execution is:  $r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow r_{12} \rightarrow \{r_5, r_{13}\}$ . The rule execution is the same as the parent membrane, at this time  $u_{t-1} = y_{t-1}$ , so the rest of the rules in this part are not eligible for execution, and the innermost membrane  $A_t$  no longer executes any rules and starts the second part of the calculation.

(3) Compute the sequence  $\{p_i\}$ ,  $\{q_i\}$

When both  $y$  and  $u$  in membrane  $A_{t-1}$  are consumed, i.e., when the condition of termination of the more phase derogation algorithm is reached (the decrement is equal to the difference), the calculation of object  $z$  is ended and no new submembrane is generated in membrane  $A_t$ . At this point, the sequence  $\{p_i\}$ ,  $\{q_i\}$  is started, i.e., the execution of the improved part of the more phase derogation algorithm is started. According to the description of the improved step of the more phase derogation algorithm in Section 3.1.2, it is known that, by several calculations, object  $z$  is generated in part of the membrane, and object  $z$  is a marker of whether the value of  $q$  in the parent membrane is changed or not. By the idea of the more phase derogation algorithm, it is known that the subtraction operation ends when the subtraction and difference are equal, and based on this conclusion, the following rule is designed:

$$\begin{array}{ll} r_{14} : (f \rightarrow p^2 q \mid !y \& !u \& !a \& !b, 6) & r_{15} : (p \rightarrow p1(r, out), 5) \\ r_{16} : (q \rightarrow q1(w, out), 5) & r_{17} : (z \rightarrow (o, out), 5) \\ r_{18} : (r \rightarrow p \mid q, 5) & r_{19} : (w \rightarrow p \mid o, 5) \\ r_{20} : (w \rightarrow p \mid q, 6) & r_{21} : (r \rightarrow p, 6) \end{array}$$

- (i) The order of the execution of the rules in membrane  $A_{t-1}$  is  $r_{14} \rightarrow \{r_{15}, r_{16}, r_{17}\}$ . To avoid confusion, the following explanation is given: in the membrane rule,  $p^2$  means

that the membrane contains 2 objects  $p$ , and  $p1$  means that the System contains an object named “ $p1$ ”.

- (a) Since the final step of the method of finding the maximum convention by the more phase derogation algorithm is that the subtraction is equal to the difference, and this equal value is the sought maximum convention, in the absence of objects  $y, u, a$ , and  $b$ , which is the case when proving that the subtractors are equal to the difference after executing rule  $r_4$ , the object  $f$  evolves to object  $p^2q$  (i.e.,  $p_{t-1} = 2, q_{t-1} = 1$ ) by executing rule  $r_{14}$ , which means that the subtracted number in membrane  $A_{t-1}$  is two times the subtracted number.
- (b) Then execute rules  $r_{15}$  and  $r_{16}$ :  $p$  and  $q$  evolve into  $p_1$  and  $q_1$ , respectively, for saving  $p_{t-1}$  and  $q_{t-1}$  in this membrane, while generating objects  $r$  and  $w$  into the parent membrane for backup. Execute rule  $r_{17}$ : when there is  $z$  in the membrane, evolve  $z$  to object  $o$  and release it to the parent membrane.
- (ii) Execution order of the rule in membrane  $A_i (1 \leq i < t - 1)$ :  $\{r_{18}, r_{19}\}$  or  $(\{r_{20}, r_{21}\}) \rightarrow \{r_{15}, r_{16}, r_{17}\}$ 
  - (a) Under the condition that an object  $o$  exists, prove the existence of  $z$  in the submembrane, which requires changing  $q_i$ . Execute rules  $r_{18}, r_{19}$ : the multiset  $r^{p_{i+1}}$  evolves into the multiset  $p^{p_{i+1}}q^{q_{i+1}}$  and the multiset  $w^{q_{i+1}}$  evolves into the multiset  $p^{p_i}$  (at this time  $p_i = p_{i+1} + q_{i+1}, q_i = p_{i+1}$ ). Under the condition that there is no object  $o$ , execute rules  $r_{20}$  and  $r_{21}$ : the multiset  $w^{q_{i+1}}$  evolves into the multiset  $p^{q_{i+1}}q^{q_{i+1}}$ , and the multiset  $r^{p_{i+1}}$  evolves into the object  $p^{p_i}$  (at this time  $p_i = p_{i+1} + q_{i+1}, q_i = q_{i+1}$ ).
  - (b) Then execute  $r_{15}$  and  $r_{16}$  to pass  $p_i$  and  $q_i$  in this membrane to the parent membrane, and if there is  $z$  in this membrane, execute  $r_{17}$  to generate  $o$  to pass to the parent membrane.
  - (c) The above steps are repeatedly executed until membrane  $A_1$  finally generates the results  $p_1$  and  $q_1$ . Execute rules  $r_{15}$  and  $r_{16}$  to output the multiset  $r^{p_1}w^{q_1}$ , where  $p_i$  and  $q_i$  are the values after simplification of the larger and smaller values in the numerator denominator, respectively.
  - (d) Finally, the comparison is performed in membrane  $M_1$ : if the numerator  $\geq$  denominator, the result  $p_i/q_i$  is directly derived (i.e., the quantity ratio of objects  $r$  and  $w$ ), and if the numerator  $<$  denominator,  $r^{p_1}$  and  $w^{q_1}$  are replaced in membrane  $M_1$  as  $w_1^{p_1}$  and  $r_1^{q_1}$ , respectively, and the final result is  $q_i/p_i$  (i.e., the quantity ratio of objects  $r_1$  and  $w_1$ ).

#### 4.1.3. Example of Fraction Simplification by the More Phase Derogation Method

In this subsection, we will give an example to describe in detail the process of implementing fractional reduction using the more phase derogation algorithm of Section 4.1.1. For example, the fraction simplification process for  $6/15$  is shown in Figure 4:

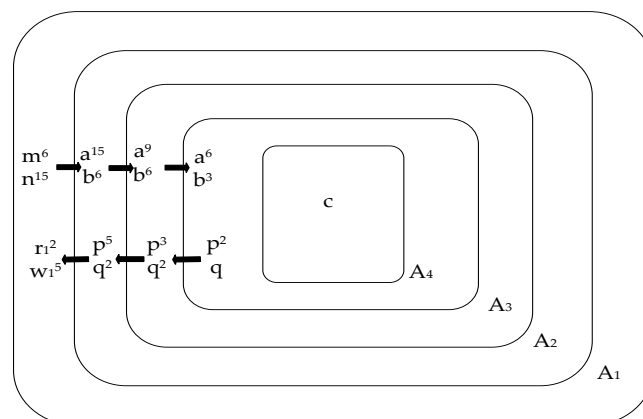


Figure 4. Flowchart of the more phase derogation algorithm example.

The following details the execution process of rules in this P System.

(1) Initial pattern

First, put the multiset  $m^6n^{15}$  in the membrane  $M_1$ , execute the rule  $r_{16}$  times to generate the multiset  $g^6$ ; execute  $r^3$  under the condition that both  $g$  and  $n$  exist, generate  $b^9$  in the membrane, and at the same time generate the multiset  $a^9Num^9$  to pass to the submembrane  $A_1$ ; execute  $r_4$  to generate the multiset  $a^6b^6Num^6$  into the submembrane  $A_1$ ; at this time, the multiset in the membrane  $M_1$  is  $b^6$ , and the initialized multiset of the membrane  $A_1$  is  $a^{15}b^6cNum^{15}CreateSubMem$ .

(2) Calculate the sequence  $\{x_i\}, \{y_i\}, \{u_i\}, \{z_i\}$

(i) The rules available in membrane  $A_1$  are executed in the following order:  $r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow r_{12} \rightarrow \{r_5, r_7, r_{13}\} \rightarrow \{r_9, r_{11}\}$ .

(a) Execute the rule  $r_1$  6 times in the membrane  $A_1$  until  $b^6$  is completely consumed, generating the multiset  $x^6y^6$  and  $a^9$  remains; execute the rule  $r_2$  under the condition that the object  $c$  exists, generating  $l^9u^9$ .

(b) At this point, only the rule  $r_3$  can be executed: evolve  $c$  to the multisets  $k$  and  $f$  under the condition that the object  $u$  exists; execute the rule  $r_4$  six times to generate the multiset  $v^6$ , at which point, the multiset in the membrane  $A_1$  is  $x^6u^3l^9v^6kf$ . Execute the rule  $r_{12}$ , as described in Section 4.1.2, consuming the *CreateSubMem* and a *Num* in this membrane to generate the membrane  $A_2$ .

(c) Execute the rules  $\{r_5, r_7, r_{13}\}$  within the same time slice. Execute the rule  $r_5$  to generate a new object  $c$  from  $k$  into the control loop in membrane  $A_2$ ; execute the rule  $r_7$  so that the multiset  $u$  and  $f$  is consumed to generate object  $h$ , and execute  $r_{13}$  to pass the remaining *Num* into the submembrane for subsequent continuation of submembrane generation.

(d) The next time slice executes  $\{r_9, r_{11}\}$  at the same time. Execute the rule  $r_9$  6 times to evolve  $x^6$  into the multiset  $b^6$  and send  $b^6$  into the membrane  $A_2$ , execute  $r_{11}$  at the same time to evolve  $l^9$  into the multiset  $a^9$  and send it into the membrane  $A_2$ .

(ii) At this time, there are multiple  $a^9b^6cNum^{14}$  sets in membrane  $A_2$ . Execute the rules in the following order:  $r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow r_{12} \rightarrow \{r_5, r_6, r_{13}\} \rightarrow \{r_8, r_{10}\}$

(a) After executing the membrane rule in the order of  $r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow r_{12}$ , the multiset within the membrane is  $l^3y^3x^6v^3kf$ , and a new submembrane  $A_3$  is generated.

(b) As described in Section 4.1.2, the next time slice simultaneously executes the rule  $\{r_5, r_6, r_{13}\}$ : execute the rule  $r_5$ , object  $k$  evolves into object  $c$  to be sent to membrane  $A_3$ ; while executing the rule  $r_6$ , object  $y$  and object  $f$  evolves into object  $i$  and sends  $z$  to membrane  $A_3$ , and executes the rule  $r_{13}$  to pass the remaining *Num* to membrane  $A_3$ .

(c) Execute the rules  $\{r_8, r_{10}\}$  under the condition that object  $i$  exists. The rule  $r_8$  is executed 6 times, and the multiset  $x^6$  evolves into the multiset  $a^6$  into the membrane  $A_3$ , while the rule  $r_{10}$  is executed 3 times, and the  $l^3$  evolves into the multiset  $b^3$  into the membrane  $A_3$ .

At this point, the membrane  $A_3$  contains the multiset  $a^3b^3czNum^{13}$ . Similarly in  $A_2$ , the available rules are executed in order  $r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow r_{12}$ , and after execution, the multiset in membrane  $A_3$  is  $l^3x^3v^3kfzNum^{12}$ . This result indicates that the subtraction and difference are equal, and the first part of the more phase derogation algorithm is over.

(3) Compute the sequence  $\{p_i\}, \{q_i\}$

(i) At this time, the multiset in the membrane  $A_3$  is  $l^3, x^3, v^3, k, f, z$ . The P System executes the rules in the following order:  $r_{14} \rightarrow \{r_{15}, r_{16}, r_{17}\}$ .

(a) Execute the rule  $r_{14}$ ,  $f$  generate the multiset  $p^2q$ ; at this point, the multiset in the membrane  $A_3$  is  $l^3, x^3, v^3, k, z, p^2, q$ ;

(b) The next time slice simultaneously executes the rules  $\{r_{15}, r_{16}, r_{17}\}$ : execute  $r_{15}$  to evolve  $p^2$  into the multiset  $p_1^2$ , and generate the multiset  $r^2$  into the membrane



- $A_2$ ; execute the rule  $r_{16}$  to evolve  $q$  into the multiset  $q_1$ , generate the object  $w$  into the membrane  $A_2$ , and execute the rule  $r_{17}$  to generate the multiset  $o$  of object  $z$  to output into  $A_2$ .
- (ii) At this time, the multiset in membrane  $A_2$  is  $o, i, y^2, v^3, r^2, w$  and the order of execution of the rules is  $\{r_{18}, r_{19}\} \rightarrow \{r_{15}, r_{16}\}$ .
- (a) Under the condition that the object  $o$  exists, execute the rules  $\{r_{18}, r_{19}\}$ :  $w$  generates the multiset  $p$ ,  $r^2$  generates the multiset  $p^2q^2$ , at which point, the multiset in  $A_2$  is  $o, i, y^2, v^3, p^3, q^2$ ;
- (b) Next, execute the rule  $\{r_{15}, r_{16}\}$  to generate the multiset  $p_1^3q_1^2$  within the membrane  $A_2$ , and also generate the multiset  $r^3w^2$  to pass into the membrane  $A_1$ .
- (iii) At this time, the multiset in membrane  $A_1$  is  $r^3, w^2, h, u^2, v^6$ , and the order of execution of the rules is  $\{r_{20}, r_{21}\} \rightarrow \{r_{15}, r_{16}\}$ .
- (a) Execute the rule  $\{r_{20}, r_{21}\}$ , the multiset  $r^3$  generates the multiset  $p^3$ , and the multiset  $w^2$  generates the multiset  $p^2q^2$ . At this time, the multiset in the membrane is  $p^5, q^2, h, u^2, v^6$ .
- (b) Then execute  $\{r_{15}, r_{16}\}$  to pass out  $r$  and  $w$ , respectively. In the membrane,  $M_1$  obtains  $r^5w^2$ , corresponding with the numerator and denominator, and then converts it to  $r_1^2w_1^5$ . After that there is no rule to execute, so the whole System stops. So the final simplification of  $6/15$  results in  $2/5$ .

The specific execution process is divided according to the time slice, as shown in Table 1. To facilitate experimental verification, all the membranes generated do not dissolve within the system. The number of membranes is large and the substances are complex, so only multiset that have changed is listed in the table. If a membrane is not listed in one of the rows of the table, it means that the material in this membrane has not changed during that time slice and the multiset in that membrane is the same as the last time it was listed.

**Table 1.** Time slice process of the more phase derogation algorithm procedure.

Time Slice	Rules for Implementation	Results of the Implementation
Initial Status	None	$M_1: m^6, n^{15}$ ; $A_1: c, \text{CreateSubMem};$
1	$r_1$	$M_1: g^6, n^9$ ;
2	$r_3, r_4$	$M_1: b^9$ ; $A_1: a^{15}, b^6, c, \text{Num}^6, \text{CreateSubMem};$
3	$r_1, r_2$	$A_1: c, \text{Num}^6, u^9, x^6, y^6, \text{CreateSubMem}, l^9$ ;
4	$r_3, r_4$	$A_1: \text{Num}^6, u^3, f, v^6, x^6, k, \text{CreateSubMem}, l^9$ ;
5	$r_5, r_7, r_{12}$	$A_1: \text{Num}^5, u^2, v^6, x^6, h, l^9$ ; $A_2: c, \text{CreateSubMem};$
6	$r_8, r_{10}, r_{13}$	$A_1: u^2, v^6, h$ ; $A_2: a^9, b^6, c, \text{Num}^5, \text{CreateSubMem};$
7	$r_1, r_2$	$A_2: c, \text{Num}^5, u^3, x^6, y^6, \text{CreateSubMem}, l^3$ ;
8	$r_3, r_4$	$A_2: \text{Num}^5, f, v^3, x^6, y^3, k, \text{CreateSubMem}, l^3$ ;
9	$r_5, r_7, r_{12}$	$A_2: \text{Num}^4, v^3, x^6, y^2, i, l^3$ ; $A_3: c, z, \text{CreateSubMem};$
10	$r_8, r_{10}, r_{13}, r_{17}$	$A_2: v^3, y^2, i, o$ ; $A_3: a^6, b^3, c, \text{Num}^4, \text{CreateSubMem};$
11	$r_1, r_2$	$A_3: c, \text{Num}^4, u^3, x^3, y^3, \text{CreateSubMem}, l^3$ ;
12	$r_3, r_4$	$A_3: \text{Num}^4, f, v^3, x^3, k, \text{CreateSubMem}, l^3$ ;
13	$r_5, r_{12}, r_{14}$	$A_3: q, \text{Num}^3, v^3, x^3, l^3, p^2$ ; $A_4: c, \text{CreateSubMem};$
14	$r_{13}, r_{15}, r_{16}$	$A_3: p_1^2, v^3, x^3, l^3, q_1$ ; $A_4: c, \text{Num}^3, \text{CreateSubMem};$
15	$r_{18}, r_{19}$	$A_3: p_1^2, v^3, x^3, l^3, q_1$ ;
16	$r_{15}, r_{16}$	$A_2: p_1^3, v^3, y^2, i, o, q_1^2$ ;

Table 1. Cont.

Time Slice	Rules for Implementation	Results of the Implementation
17	$r_{20}, r_{21}$	$A_1: q^2, u^2, v^6, h, p^5;$ $A_2: p_1^3, v^3, y^2, i, o, q_1^2;$
18	$r_{15}, r_{16}$	$M_1: b^9, r^5, w^2; A_1: p_1^5, u^2, v^6, h, q_1^2;$
19	$r_{20}, r_{21}$	$M_1: b^9, w_1^5, r_1^2;$
20	No enforceable rules	

From the above analysis process, it can be seen that using the P System  $\Pi_1$  for fractional reduction on 6/15 requires a total of 19 time slices.

#### 4.2. P System for Fraction Simplification in the Division Algorithm

##### 4.2.1. Definition of the P System for Rolling Division

The definition of the P System based on the division algorithm is as follows (3):

$$\Pi_2 = (V, O, H, \mu, \omega_{M_1}, \omega_{A_1}, \dots, \omega_{A_t}, R_{M_1}, R_{A_1}, \dots, R_{A_t}, i_0), \quad (3)$$

Among them,

- (1)  $V = \{m, n, c, t, CreateSubMem, Num, a, b, g, x, y, d, e, h, z, o, i, l, u, k, f, w, v\};$
- (2) The definitions of  $H$  and  $\mu$  are similar to those in Section 4.1.1, so this section does not provide detailed explanations;
- (3)  $\omega_{M_1} = m^x n^y c t$ ,  $\omega_{A_1} = c CreateSubMem$ , and  $x$  and  $y$  represent the numerator and denominator of fractions, respectively;
- (4) The  $R_{M_1}, R_{A_1}, \dots, R_{A_t}$  evolution rule finite set will be explained in Section 4.2.2;
- (5)  $i_0 = M_1$  indicates that when the entire System stops, the final result can be obtained in the membrane  $M_1$ .

Figure 5 depicts the initial layout of the division algorithm P System: membrane  $M_1$  is mainly used to compare the magnitudes of the numerator  $x$  and the denominator  $y$ . The larger and smaller values are passed to object  $a$  and object  $b$  in membrane  $A_1$ , respectively, and after obtaining the maximum convention  $x$  from membrane  $A_1$  and the dynamically generated class A membrane, the numerator and denominator are divided by  $x$ , respectively, saving the final result of the operation  $k/l$ . Membrane  $A_1$  is used to perform the division algorithm operation and calculate the value of the maximum convention,  $x$ . With the exception of membrane  $M_1$  and membrane  $A_1$ , all other required membrane structures are created dynamically during the reduction process based on the actual situation of the partition, and the rules within the newly created membrane are identical to those within membrane  $A_1$ . In Figure 5, objects  $m$  and  $n$  are used to represent the numerator and denominator, respectively;  $x$  and  $y$  are the numbers of objects  $m$  and  $n$ . Object  $c$  is used to trigger the execution of the rules in the membrane, and  $CreateSubMem$  determines whether new membranes need to be generated according to different conditions. Membrane  $A_1$  and other newly created submembranes are used to calculate the parameters in the rules. These newly created membranes are nested in a layer: membrane  $A_2$  is created within membrane  $A_1$ , membrane  $A_3$  is created within membrane  $A_2$ , and finally, membrane  $A_t$  is created within membrane  $A_{t-1}$ .

##### 4.2.2. Membrane Rule of Division Algorithm

Combining the design idea of the P System and the principle of Section 3.2, the P System of fraction simplification by division algorithm mainly consists of calculating the sequence of larger values  $\{a_i\}$ , the sequence of smaller values  $\{b_i\}$ , the maximum convention  $x_1$ , the numerator  $k$  after reduction, and the denominator  $l$  after reduction. The initial state of the multiset  $m^x, n^y, c, t$  is first put into the membrane  $M_1$ , where  $x$  denotes the number of numerators and  $n$  denotes the number of denominators. Then, after comparing the size of the numerator and denominator in membrane  $M_1$ , the larger and smaller values are passed into membrane  $A_1$ , respectively, and the maximum convention number  $x_1$  is

calculated according to the rule, and then  $x_1$  is passed into membrane  $M_1$  to calculate the final approximate result  $k/l$ . Following this idea, a specific flow chart can be obtained, as shown in Figure 6.

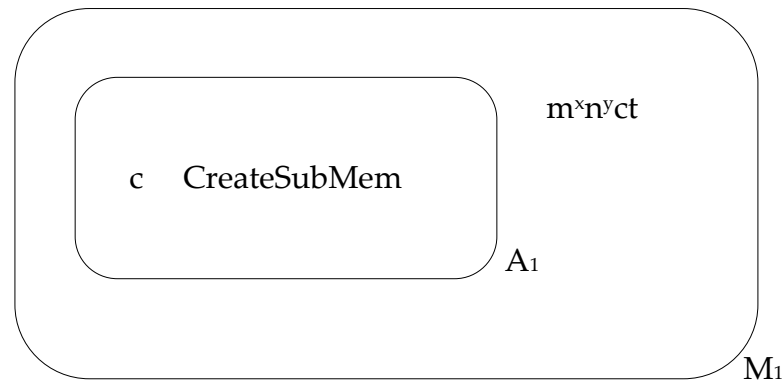


Figure 5. Initial pattern of the P System in the division algorithm.

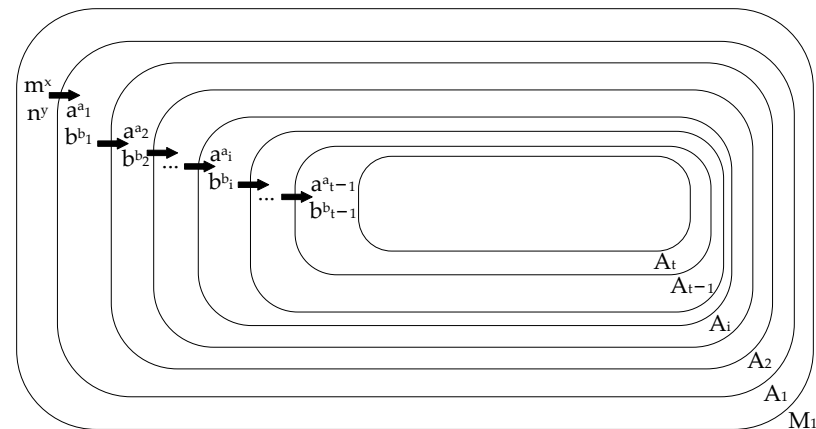


Figure 6. Flow diagram of the division algorithm P System.

The following is a specific description of the membrane rules.

(1) Calculate the sequence  $\{a_i\}$ ,  $\{b_i\}$  and the maximum convention  $x_1$

$m^x n^y ct$  is the initial state of membrane  $M_1$ . The numerator  $x$  and denominator  $y$  are compared in membrane  $M_1$  and then the larger and smaller values are passed to object  $a$  and object  $b$  in membrane  $A_1$ , respectively. The specific conversion rules are similar to membrane  $M_1$  in the more phase derogation algorithm, which can be found in Section 4.1.1. The membrane rule within this System improves the division into multiple additions and subtractions; the initialized multiset in membrane  $A_1$  is  $a^{a_1} b^{b_1} c Num^{a_1} CreateSubMem$ , where  $a_1$  is the value of the larger number in the numerator denominator, and  $b_1$  is the value of the smaller number.

The rule for calculating the maximum convention  $x_1$  is as follows:

- |   |   |  |
|---|---|--|
| $r_1 : (a \ b \rightarrow x, 1)$                          | $r_2 : (c \rightarrow y _x, 1)$               | $r_3 : (a \ y \rightarrow a \ d, 2)$     |
| $r_4 : (x \rightarrow b _d, 3)$                           | $r_5 : (d \rightarrow c, 4)$                  | $r_6 : (b \ y \rightarrow b \ e \ g, 2)$ |
| $r_7 : (CreateSubMem \ Num \rightarrow A : a\{\}   b, 3)$ |   | $r_8 : (g \rightarrow h, 3)$             |
| $r_9 : (x \rightarrow (b, in) _h, 4)$                     | $r_{10} : (b \rightarrow (a, in) _h, 4)$      | $r_{11} : (h \rightarrow (c, in) _h, 5)$ |
| $r_{12} : (y \rightarrow z \ e, 3)$                       | $r_{13} : (x \rightarrow (z \ x, out) _z, 4)$ |  |
| $r_{14} : (Num \rightarrow (Num, in) _{CreateSubMem})$    |   |  |

- (i) When  $a_i - nb_i > b_i (i = 1, 2, \dots, t-1, n = 0, 1, \dots, t-1)$  the rules are executed in the following order:  $\{r_1, r_7\} \rightarrow \{r_2, r_{14}\} \rightarrow r_3 \rightarrow r_4 \rightarrow r_5$ .

- (a) Execution of rule  $r_1$  within membrane  $A_i$ : consume the multiset  $a^{b_i}b^{b_i}$ , generating  $x^{b_i}$ , which means that the values of the denominator and numerator are consumed in  $b_i$  copies at the same time, leaving  $a_i - b_i$  copies of the object  $a$ , while executing  $r_7$  to generate a new submembrane  $A_{i+1}$ ; execution of rule  $r_2$ : object  $c$  evolves into  $y$  under the condition that object  $x$  exists; this step is used to control the loop, while execution  $r_{14}$  passes the remaining Num into the submembrane  $A_{i+1}$ ;
  - (b) Then execute  $r_3$ , the object  $y$  evolves into object  $d$ , where  $a$  plays a catalytic role; under the condition that  $d$  exists, execute rule  $r_4$ ; the object  $x$  evolves into object  $b$  and returns  $x^{b_i}$  to  $b^{b_i}$ . At this time, the existing multiset in the membrane is  $a^{a_i-b_i}b^{b_i}dNum^{a_i}$ , and the System satisfies the execution conditions of  $r_1$  and continues to generate  $x$ .
  - (c) Execute  $r_5$ , object  $d$  evolves into object  $c$ ; at this time, there are  $x$  generated in the membrane  $A_i$ , so the execution conditions of  $r_2$  are satisfied again. Because there is only one *CreateSubMem* object in the membrane,  $r_7$  and  $r_{14}$  in a membrane can only be executed once, but the rule  $r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow r_5$  may be cyclically executed  $n$  times, until  $a_i - nb_i < b_i$  stops the cycle of execution, or  $a_i - nb_i = b_i$  proves that the remainder is 0; then, the System has found the maximum convention  $x$  and stops the cycle of execution.
- (ii) When  $a_i - nb_i < b_i$  in the membrane  $A_i$ , the rules are executed in the following order:  $r_1 \rightarrow r_2 \rightarrow r_6 \rightarrow \{r_7, r_8\} \rightarrow \{r_9, r_{10}, r_{11}, r_{14}\}$ .
- (a) Execution rule  $r_1$ : the multiset  $a^{a_i-nb_i}b^{a_i-nb_i}$  is consumed to generate the multiset  $x^{a_i-nb_i}$ . This means that the values of the denominator and numerator are consumed in both  $a_i - nb_i$  copies, leaving  $b_i - (a_i - nb_i)$  copies of object  $b$ . At this time, the number of  $x$   $a_i - nb_i$  is the remainder of  $a_i$  divided by  $b_i$ .
  - (b) Under the condition that object  $x$  exists, rule  $r_2$  is executed to evolve object  $c$  into  $y$ . This step is used to control the loop; rule  $r_6$  is executed, and object  $y$  evolves into objects  $e$  and  $g$  under the action of catalyst  $b$ .
  - (c) Execute rules  $\{r_7, r_8\}$ : consume *CreateSubMem* with a *Num* to generate a new submembrane  $A_{i+1}$  under the condition that object  $b$  exists, while  $g$  evolves into object  $h$  for synchronization control.
  - (d) Then execute the rules  $\{r_9, r_{10}, r_{11}, r_{14}\}$ : under the condition that the object  $h$  exists, convert the multiset  $x^{a_i-nb_i}$  to multiset  $b^{b_{i+1}}$  directly into the submembrane  $A_{i+1}$  and the multiset  $b^{b_i-(a_i-nb_i)}$  to multiset  $a^{a_{i+1}}$  into the submembrane; at this time,  $b_{i+1} = a_i - nb_i$  and  $a_{i+1} = b_i - (a_i - nb_i)$ , while the object  $h$  evolves into  $c$  into the membrane  $A_{i+1}$  to continue the execution. In membrane  $A_{i+1}$ , we continue to judge the size relationship between  $a_{i+1}$  and  $b_{i+1}$  and execute the corresponding rules.
- (iii) When  $a_{t-1} - nb_{t-1} = b_{t-1}$ , the rules are executed in the following order:  $r_1 \rightarrow r_2 \rightarrow r_{12} \rightarrow r_{13}$ .
- (a) Rule  $r_1, r_2$  is executed as above. At this time, there is no excess  $a$  or  $b$  in the membrane, proving that the two numbers have no remainder, which also means that they have been integregated; therefore, the execution of rule  $r_{12}$ : the object  $y$  evolves into  $z$  and  $e$ . Under the condition that there is  $z$ , the execution of rule  $r_{13}$  converts the multiset  $x^{b_{t-1}}$  into the multiset  $z^{x_1}x^{x_1}$  directly into the parent membrane, i.e.,  $b_{t-1}$  is the maximum common divisor.
  - (b) In membranes  $A_{t-2}, A_{t-3}, \dots, A_1$ , after receiving the  $z^{x_1}x^{x_1}$  of the submembrane, rule  $r_{13}$  is executed to output the maximum common divisor, and finally output it from membrane  $A_1$  to membrane  $M_1$ . Calculate  $k$  and  $l$  according to the rules in membrane  $M_1$ .

(2) Calculate  $k, l$ 

The calculation of  $k$  and  $l$  begins when membrane  $M_1$  receives the maximum convention  $x_1$  from membrane  $A_1$ . The specific calculation rules are as follows:

$$\begin{array}{lll} r_5 : (x \rightarrow w \vee) & r_6 : (a \ w \rightarrow i, 1) & r_7 : (c \rightarrow y|_i, 1) \\ r_8 : (a \ y \rightarrow a \ d, 2) & r_9 : (i \rightarrow w|_d, 3) & r_{10} : (d \rightarrow k \ c, 4) \\ r_{11} : (y \rightarrow k, 3) & r_{12} : (b \ v \rightarrow f, 1) & r_{13} : (t \rightarrow e|_f, 1) \\ r_{14} : (b \ e \rightarrow b \ h, 2) & r_{15} : (f \rightarrow v|_h, 3) & r_{16} : (h \rightarrow l \ t, 4) \\ r_{17} : (e \rightarrow l, 3) \end{array}$$

The main job of membrane  $M_1$  (at this point) is to divide the numerator and denominator by the maximum common divisor, respectively, and finally arrive at the result. Firstly, rule  $r_5$  is executed: the multiset  $x^{x_1}$  is converted into a multiset  $w^{x_1}v^{x_1}$  for backup, so that the numerator and denominator can be synchronously simplified and parallelism can be improved.

- (i) The rules involved in simplifying the molecule  $x$  and the order of execution are as follows:  $r_6 \rightarrow r_7 \rightarrow r_8 \rightarrow r_9 \rightarrow r_{10} \rightarrow r_{11}$ .
  - (a) Execute rule  $r_6$ : The multiset  $a^{x_1}w^{x_1}$  evolves into  $i^{x_1}$ . At this point, object  $i$  retains the number of conventions  $x_1$ , and there are  $x - x_1$  objects  $a$  remaining in the System. Because  $x_1$  is the largest convention of the numerator denominator, object  $a$  must be an integer multiple of  $x_1$ .
  - (b) In the condition that object  $i$  exists, execute rule  $r_7$ : object  $c$  evolves into object  $y$ . This step is used to control the loop; execute  $r_8$ : object  $y$  evolves into object  $d$  under the action of catalyst  $a$ . This step proves that there is no end to integer division in the System, at this time;
  - (c) Execute rule  $r_9$  under the condition that object  $d$  exists: object  $i$  evolves into object  $w$ , and then the value of the convention number is passed to  $w$ ; at this point,  $r_6$  execution conditions are met, and rule  $r_{10}$  is executed while continuing to execute  $r_6$ : object  $d$  evolves into objects  $k$  and  $c$ , i.e., the convention number is subtracted once for one more  $k$ , which is equivalent to a counter.
  - (d) The above steps can be executed several times. Until there is no object  $a$  in the System after executing  $r_6$  again, it indicates that the molecules have been completely divided. Execution rule  $r_7$  and  $r_{11}$ : object  $y$  evolves into object  $k$ . The last number of  $k$  is the value of the numerator after the approximate simplification, and also, the number of times the subtraction is used.
- (ii) The rules involved in simplifying the denominator and the order of execution are as follows:  $r_{12} \rightarrow r_{13} \rightarrow r_{14} \rightarrow r_{15} \rightarrow r_{16} \rightarrow r_{17}$ . The execution rules are similar to the steps for dividing the numerator by the greatest common divisor, and the final number of  $l$  is the value of the denominator after simplification. Due to the maximum parallelism of the membrane calculation, the number of steps required to implement the division function is the number of steps in the larger value simplification.

The final result of the approximate fractional simplification in the division algorithm P System is  $k/l$ .

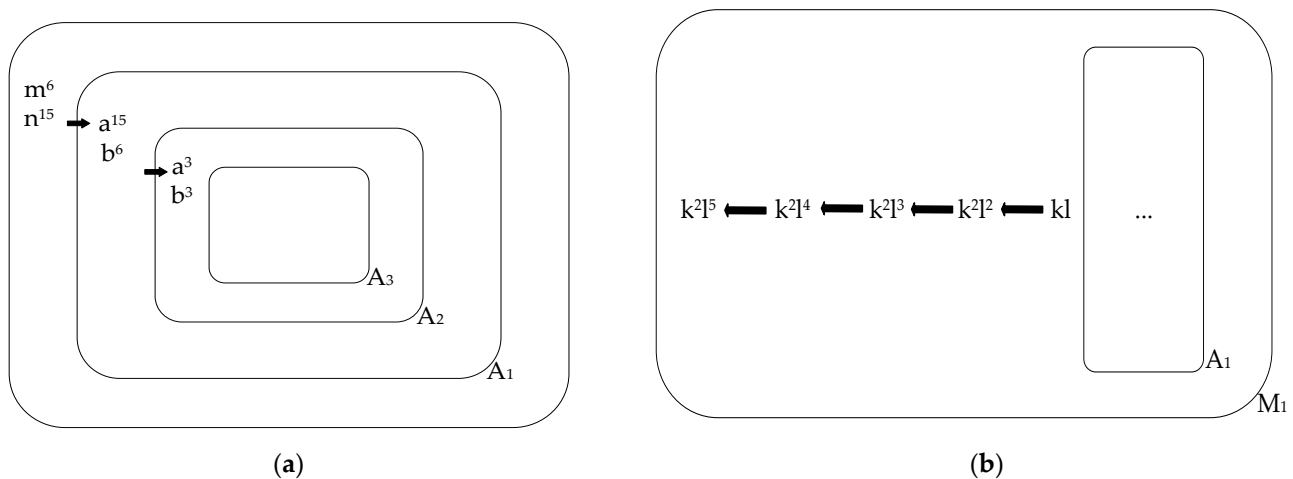
#### 4.2.3. Example of Fraction Simplification by the Division Algorithm

In this subsection, we will give an example to describe in detail how to implement fractional simplification using the division algorithm P System. For example, the reduction process for  $6/15$  is shown in Figure 7:

The following details the execution process of rules in this P System.

##### (1) Initial pattern

As shown in Figure 7a, the initial multiset  $m^6n^{15}ct$  of this System is first put into the membrane  $M_1$ , and the membrane  $A_1$  has the initial object  $c$  and the object *CreateSubMem*. The object  $n$  is used to represent the denominator of the fractional equation with the number of 15, and the object  $m$  is used to represent the numerator of the fractional equation with the number of 6.



**Figure 7.** Process diagram for solving the example of the division algorithm P System. (a) Flowchart for finding the greatest common divisor  $x$  in the division algorithm; (b) Flowchart for simplifying the result of the division algorithm example.

First execute the rule  $r_1$  six times, and the multi-set  $m^6n^6$  is completely consumed to generate the object  $g^6$ ; then execute the rule  $r_3$  nine times under the condition of having the object  $g$ . The remaining object  $n^9$  evolves into  $b^9$  and generates the objects  $a^9$  and  $Num^9$  into the membrane  $A_1$ , while  $g^6$  generates the multi-set  $a^6b^6$  and passes  $a^6b^6Num^6$  to the membrane  $A_1$ . At this time, there is no other executable rule in the membrane  $M_1$ , so wait for the end of the execution of the membrane of class A to continue the execution. The initialized multiset in membrane  $A_1$  is  $a^{15}b^6cNum^{15}CreateSubMem$ .

(2) Calculate the maximum common divisor  $x_1$

- (i) Execute rules in the following order:  $\{r_1, r_7\} \rightarrow \{r_2, r_{14}\} \rightarrow r_3 \rightarrow r_4$ . In membrane  $A_1$ , execute rule  $r_1$  6 times to generate object  $x^6$ , while executing rule  $r_7$ , under the condition of having  $b$  directly consume  $CreateSubMem$  and a  $Num$  to generate submembrane  $A_2$  for backup. The existing object in Membrane  $A_1$  is  $a^9x^6c$ . Execute rule  $r_2$  to evolve object  $c$  to  $y$  under the condition that object  $x$  exists, while executing  $r_{14}$  to pass the remaining  $Num$  value into submembrane  $A_2$ .

At this point, only the rule  $r_3$  can be executed, in the role of catalyst  $a$  object  $y$  evolved to  $d$ ; according to the conditions of the implementation of the rule  $r_4$ , the object  $x^6$  evolved to  $b^6$ ; implementation of the rule  $r_5$ , the object  $d$  evolved to  $c$ ; at this time, the  $r_1$  condition is met again, and the multiset in the membrane  $A_1$  is  $a^3b^6c$ , after the rules are executed in the order of  $r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4$ .

- (ii) At this time  $a < b$ , the rules are executed in the order of  $r_1 \rightarrow r_2 \rightarrow r_6 \rightarrow \{r_7, r_8\} \rightarrow \{r_9, r_{10}, r_{11}\}$ ; execute the rule  $r_1$  three times to generate the multiset  $x^3$ ; execute the rule  $r_2$ , and the object  $c$  evolves into the object  $y$ ; thus, at this time the multiset in the membrane  $A_1$  is  $x^3b^3y$ ; this case can only execute the rule  $r_6$ , converting the multiset  $b$  and  $y$  to the multiset  $b, e, g$ ; then execute the rule  $r_8$ , converting the object  $g$  to  $h$ ; under the condition that  $h$  exists at the same time, execute the rules  $r_9, r_{10}, r_{11}$ , generating  $a^3b^3c$  passed to membrane  $A_2$ .
- (iii) At this time, the multiset in membrane  $A_2$  is  $a^3b^3cNum^{14}CreateSubMem$ , i.e.,  $a_2 = b_2 = 3$ , according to  $\{r_1, r_7\} \rightarrow \{r_2, r_{14}\} \rightarrow r_{12} \rightarrow r_{13}$ ; execute the rule  $r_1$  three times, consume all the  $a$  and  $b$  to generate  $x^3$ , while executing  $r_7$  to generate submembrane  $A_3$ ; execute the rule  $r_2$ , the multiset  $c$  evolves into the multiset  $y$ , and at this time, the multiset in the membrane  $A_2$  is  $x^3y$ , while executing the rule  $r_{14}$  to pass the remaining  $Num$  values to the submembrane  $A_3$ . In this environment, one can only execute the rule  $r_{12}$  to evolve  $y$  into the multiset  $z$  and  $e$ ; execute under the condition that the multiset  $z$  exists, and rule  $r_{13}$  transforms  $x^3$  into  $z^3x^3$  and passes it to membrane  $A_1$ ; rule  $r_{13}$  continues in membrane



$A_1$  and passes the multiset  $z^3x^3$  to membrane  $M_1$ , where the number of objects  $x$  is the maximum convention  $x_1$ . This concludes the computation of the maximum convention  $x_1$ .

(3) Calculate  $k, l$

At this point, the multiset in the membrane  $M_1$  is  $a^6b^{15}z^3x^3ct$ .

- (i) Execute rule  $r_5$  in membrane  $M_1$ , backing up the maximum convention as  $w^3v^3$  for simultaneous simplification of the numerator and denominator; execute rule  $r_6$  three times, consuming the multiset  $a^3w^3$  to generate  $i^3$ , and execute rule  $r_{12}$  three times simultaneously, consuming the multiset  $b^3v^3$  to generate  $f^3$ ;
- (ii) Execute rule  $r_7$  to evolve object  $c$  into  $y$  under the condition that object  $i$  exists, and the execution of the rule  $r_{13}$  involving the evolution of multiset  $f^3$  to multiset  $v^3$  under the condition that object  $h$  exists;
- (iii) Simultaneous execution of the rules  $r_8, r_{14}$ , evolving  $y$  to  $d$  in the presence of catalyst  $a$  and  $e$  to  $h$  in the presence of catalyst  $b$ .
- (iv) Simultaneously execute the rules  $r_9, r_{15}$ , under the condition that  $d$  exists, the object  $i^3$  evolves into  $w^3$ , and  $f^3$  evolves into  $v^3$ . Then simultaneously execute the rules  $r_{10}, r_{16}$ , evolve  $d$  into  $k, c$  and  $h$  into  $l, t$ , respectively; at this time,  $k$  and  $l$  start to accumulate, and the multiset in the membrane  $M_1$  is  $z^3, a^3, b^{12}, w^3, v^3, c, t, k, l$ .
- (v) After the numerator is repeated once in the order  $\{r_6, r_{11}\}$  and the denominator is repeated once in the order  $\{r_{12}, r_{17}\}$ , the multiset inside the membrane is  $z^3i^3b^9v^3k^2l^2t$ . Since  $a$  has been completely consumed, the rule  $\{r_6, r_{11}\}$  is stopped and the denominator continues to be executed. After repeating the execution three times in sequence, the multiset within the membrane is now  $z^3i^3f^3k^2l^4e$ , at which point, only rule  $r_{17}$  can be executed and object  $e$  evolves to  $l$ . At this time, the multiset in the membrane is  $z^3i^3f^3k^2l^5$ .

The execution up to this point has no rules to execute, so the whole system stops. The number of objects  $k$  and  $l$  denote the values of the numerator and denominator after simplification, respectively, so  $6/15$  ends up with a simplification result of  $2/5$ .

The specific execution process is divided according to the time slice, as shown in Table 2. This table is also partially omitted.

**Table 2.** Time slice process for the division algorithm.

Time Slice	Rules for Implementation	Results of the Implementation
Initial Status	None	$M_1: m^6, n^{15}, c, t; A_1: c, \text{CreateSubMem};$
1	$r_1$	$M_1: g^6, n^9, c, t;$
2	$r_3, r_4$	$M_1: a^6, b^{15}, c, t;$
3	$r_1, r_7$	$A_1: a^{15}, b^6, c, \text{Num}^{15}, \text{CreateSubMem};$ $A_1: a^9, c, \text{Num}^{14}, x^6; A_2: \text{CreateSubMem};$
4	$r_2, r_{14}$	$A_1: a^9, x^6, y;$ $A_2: \text{Num}^{14}, \text{CreateSubMem};$
5	$r_3$	$A_1: a^9, x^6, d;$
6	$r_4, r_5$	$A_1: a^9, b^6, c;$
7	$r_1$	$A_1: a^3, x^6, c;$
8	$r_2$	$A_1: a^3, x^6, y;$
9	$r_3$	$A_1: a^3, x^6, d;$
10	$r_4$	$A_1: a^3, b^6, c;$
11	$r_1$	$A_1: b^3, x^3, c;$
12	$r_2$	$A_1: b^3, x^3, y;$
13	$r_6$	$A_1: b^3, x^3, e, g;$
14	$r_8$	$A_1: b^3, x^3, e, h;$
15	$r_9, r_{10}$	$A_1: e;$ $A_2: a^3, b^3, c, \text{Num}^{14}, \text{CreateSubMem};$
16	$r_1, r_7$	$A_2: c, \text{Num}^{13}, x^3; A_3: \text{CreateSubMem};$
17	$r_2, r_{14}$	$A_2: x^3, y; A_3: \text{Num}^{13}, \text{CreateSubMem};$

Table 2. Cont.

Time Slice	Rules for Implementation	Results of the Implementation
18	r <sub>12</sub>	A <sub>2</sub> : e, x <sup>3</sup> , z;
19	r <sub>13</sub>	A <sub>1</sub> : e, x <sup>3</sup> , z <sup>3</sup> ; A <sub>2</sub> : e, z;
20	r <sub>13</sub>	M <sub>1</sub> : a <sup>6</sup> , b <sup>15</sup> , c, t, x <sup>3</sup> , z <sup>3</sup> ; A <sub>1</sub> : e, z <sup>3</sup> ; A <sub>2</sub> : e, z;
21	r <sub>5</sub>	M <sub>1</sub> : a <sup>6</sup> , b <sup>15</sup> , c, t, v <sup>3</sup> , w <sup>3</sup> , z <sup>3</sup>
22	r <sub>6</sub> , r <sub>12</sub>	M <sub>1</sub> : a <sup>3</sup> , b <sup>12</sup> , c, t, f <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> ;
23	r <sub>7</sub> , r <sub>13</sub>	M <sub>1</sub> : a <sup>3</sup> , b <sup>12</sup> , e, f <sup>3</sup> , i <sup>3</sup> , y, z <sup>3</sup> ;
24	r <sub>8</sub> , r <sub>14</sub>	M <sub>1</sub> : a <sup>3</sup> , b <sup>12</sup> , d, f <sup>3</sup> , h, i <sup>3</sup> , z <sup>3</sup> ;
25	r <sub>9</sub> , r <sub>10</sub> , r <sub>15</sub> , r <sub>16</sub>	M <sub>1</sub> : a <sup>3</sup> , b <sup>12</sup> , c, t, v <sup>3</sup> , w <sup>3</sup> , z <sup>3</sup> , k, l;
26	r <sub>6</sub> , r <sub>12</sub>	M <sub>1</sub> : b <sup>9</sup> , c, t, f <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> , k, l;
27	r <sub>7</sub> , r <sub>13</sub>	M <sub>1</sub> : b <sup>9</sup> , e, f <sup>3</sup> , i <sup>3</sup> , y, z <sup>3</sup> , k, l;
28	r <sub>11</sub> , r <sub>14</sub>	M <sub>1</sub> : b <sup>9</sup> , f <sup>3</sup> , h, i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l;
29	r <sub>15</sub> , r <sub>16</sub>	M <sub>1</sub> : b <sup>9</sup> , t, v <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>2</sup> ;
30	r <sub>12</sub>	M <sub>1</sub> : b <sup>6</sup> , t, f <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>2</sup> ;
31	r <sub>13</sub>	M <sub>1</sub> : b <sup>6</sup> , e, f <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>2</sup> ;
32	r <sub>14</sub>	M <sub>1</sub> : b <sup>6</sup> , f <sup>3</sup> , h, i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>2</sup> ;
33	r <sub>15</sub> , r <sub>16</sub>	M <sub>1</sub> : b <sup>6</sup> , t, v <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>3</sup> ;
34	r <sub>12</sub>	M <sub>1</sub> : b <sup>3</sup> , t, f <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>3</sup> ;
35	r <sub>13</sub>	M <sub>1</sub> : b <sup>3</sup> , e, f <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>3</sup> ;
36	r <sub>14</sub>	M <sub>1</sub> : b <sup>3</sup> , f <sup>3</sup> , h, i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>3</sup> ;
37	r <sub>15</sub> , r <sub>16</sub>	M <sub>1</sub> : b <sup>3</sup> , t, v <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>4</sup> ;
38	r <sub>12</sub>	M <sub>1</sub> : t, f <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>4</sup> ;
39	r <sub>13</sub>	M <sub>1</sub> : e, f <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>4</sup> ;
40	r <sub>17</sub>	M <sub>1</sub> : f <sup>3</sup> , i <sup>3</sup> , z <sup>3</sup> , k <sup>2</sup> , l <sup>5</sup> ;
41	No enforceable rules	

From the above process analysis, it is clear that the total time slice for this example is 40.

#### 4.3. P System Combined with the More Phase Derogation and the Tossing and Division Algorithm

##### 4.3.1. Definition of a Combinatorial P System

The P System of a combinatorial more phase derogation algorithm and the tossing and division algorithm is defined as Equation (4):

$$\Pi_3 = (V, O, H, \mu, \omega_{M_1}, \omega_{A_1}, \dots, \omega_{A_{t_1}}, \omega_{B_1}, \dots, \omega_{B_{t_2}}, R_{M_1}, R_{A_1}, \dots, R_{A_{t_1}}, R_{B_1}, \dots, R_{B_{t_2}}, i_0), \quad (4)$$

Among them,

(1)  $V = \{m, n, c, t, \text{CreateSubMem}, \text{Num}, a, b, g, x, y, d, e, h, z, o, i, l, u, k, f, w, v\}$ ;

(2)  $H = \{M_1, A_1, \dots, A_{t_1}, B_1, \dots, B_{t_2}\}$ , when the P System is in its initial state; it only contains membrane  $M_1$ , membrane  $A_1$ , and membrane  $B_1$ . During the execution process, membranes  $A_2, \dots, A_{t_1}, B_2, \dots, B_{t_2}$  are generated according to different conditions.

(3)  $\mu = \left\{ \left[ \left[ \left[ \left[ A_{t_1} \right] \dots \right] A_1 \right] \left[ \left[ \left[ B_{t_2} \right] \dots \right] B_1 \right] M_1 \right\}$ , in the initial state, membrane  $A_1$  and membrane  $B_1$  are siblings, both included in membrane  $M_1$ ;  $A_i (2 \leq i \leq t_1)$  and  $B_k (2 \leq k \leq t_2)$  are dynamically generated during the execution of the membrane rules; any  $A_i$  is always included in  $A_{i-1}$ , and any  $B_k$  is always included in  $B_{k-1}$ .

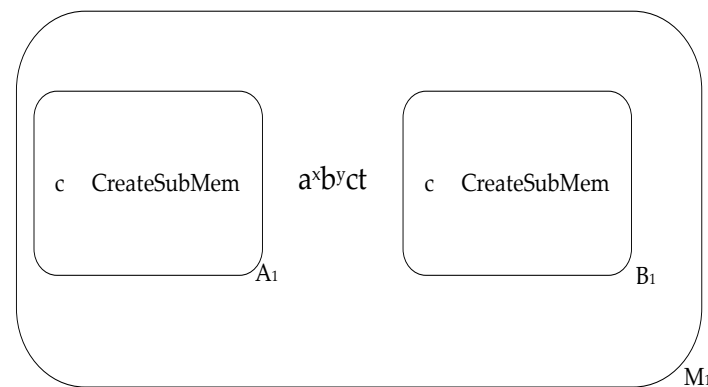
(4)  $\omega_{M_1} = a^m b^n c t$ ,  $\omega_{A_1} = c \text{CreateSubMem}$ , and  $\omega_{B_1} = c \text{CreateSubMem}$ ;  $m$  and  $n$  represent the numerator and denominator of fractions, respectively;

(5) The  $R_{M_1}, R_{A_1}, \dots, R_{A_{t_1}}, R_{B_1}, \dots, R_{B_{t_2}}$  evolution rule finite set will be explained in Section 4.3.2;

(6)  $i_0 = M_1$  indicates that when the entire System stops, the final result can be obtained in the membrane  $M_1$ .

As shown in Figure 8, this diagram shows the initial pattern of a fraction simplification P System with a combination of the more phase derogation and division algorithms. The order of execution of the various types of membranes in this P System is: membrane  $M_1 \rightarrow$  membrane  $A_1$  and submembranes  $\rightarrow$  membrane  $M_1 \rightarrow$  membrane  $B_1$  and submembranes  $\rightarrow$

membrane  $M_1 \rightarrow$  membrane  $A_1$  and submembranes. The function of each type of membrane and the complete process of the P System are briefly described as follows: membrane  $A_1$  is first used to perform a more phase derogation algorithm to determine the relationship with  $\xi$ , and when the difference is greater than  $\xi$ , it continues to generate submembrane for a more phase derogation algorithm until the difference is less than or equal to  $\xi$ ; the intermediate truncation results  $m, n$  are obtained and passed to membrane  $M_1$  and then from membrane  $M_1$  to membrane  $B_1$ . Membrane  $B_1$  receives  $m, n$  and the division algorithm to calculate the maximum convention  $x$  and passes it to membrane  $M_1$ . Membrane  $M_1$  finds the approximate result of the intermediate  $m, n$  and passes it to membrane  $A_1$ . Membrane  $A_1$  passes the result of the more phase derogation algorithm  $r/w$  to membrane  $M_1$ . Finally, within membrane  $M_1$ ,  $r$  and  $w$  correspond to the numerator and denominator: if the numerator  $>$  denominator, the final result is directly  $r/w$ , otherwise  $r$  and  $w$  are “interchanged”, and the final result is  $w_1/r_1$  with exception to membrane  $M_1$  and membranes  $A_1$  and  $B_1$ , all other required membrane structures are created dynamically during the reduction process, and the rules in the newly created class A membranes,  $A_2, A_3, \dots, A_{t_1}$ , are identical to those in membrane  $A_1$ ; class B membranes  $B_2, B_3, \dots, B_{t_2}$  have exactly the same rules as those in membrane  $B_1$ .



**Figure 8.** Process diagram for solving the example of the rolling phase division P System.

#### 4.3.2. Membrane Rules for Combining the Two Methods

As can be seen from Section 4.1 of the more phase derogation algorithm P System, if the difference between the numerator and denominator is too large, the number of dynamically generated membranes is too large, and if Section 4.2 of the division algorithm is used, each division is equivalent to multiple subtractions, making the number of subtraction steps too heavy for a membrane System. For this reason, this sub-section discusses in detail the fraction simplification of the P System using a combination of the idea of a more phase derogation algorithm and a division algorithm.

The process of fraction simplification using the combination of the two methods consists of finding the sequence  $\{a_i\}, \{b_i\}, \{u_i\}, \{z_i\}$ , the intermediate values  $m, n$  of the method transformation, the greatest common divisor  $x_1$ , the result of the reduction of the intermediate values  $k_1, l_1$ , and the sequence  $\{p_i\}, \{q_i\}$ . Combining the design ideas of the two P Systems in Sections 4.1 and 4.2, it can be seen that all membrane structures require only addition and subtraction to complete the calculation of fractional simplification, as shown in Figure 9.

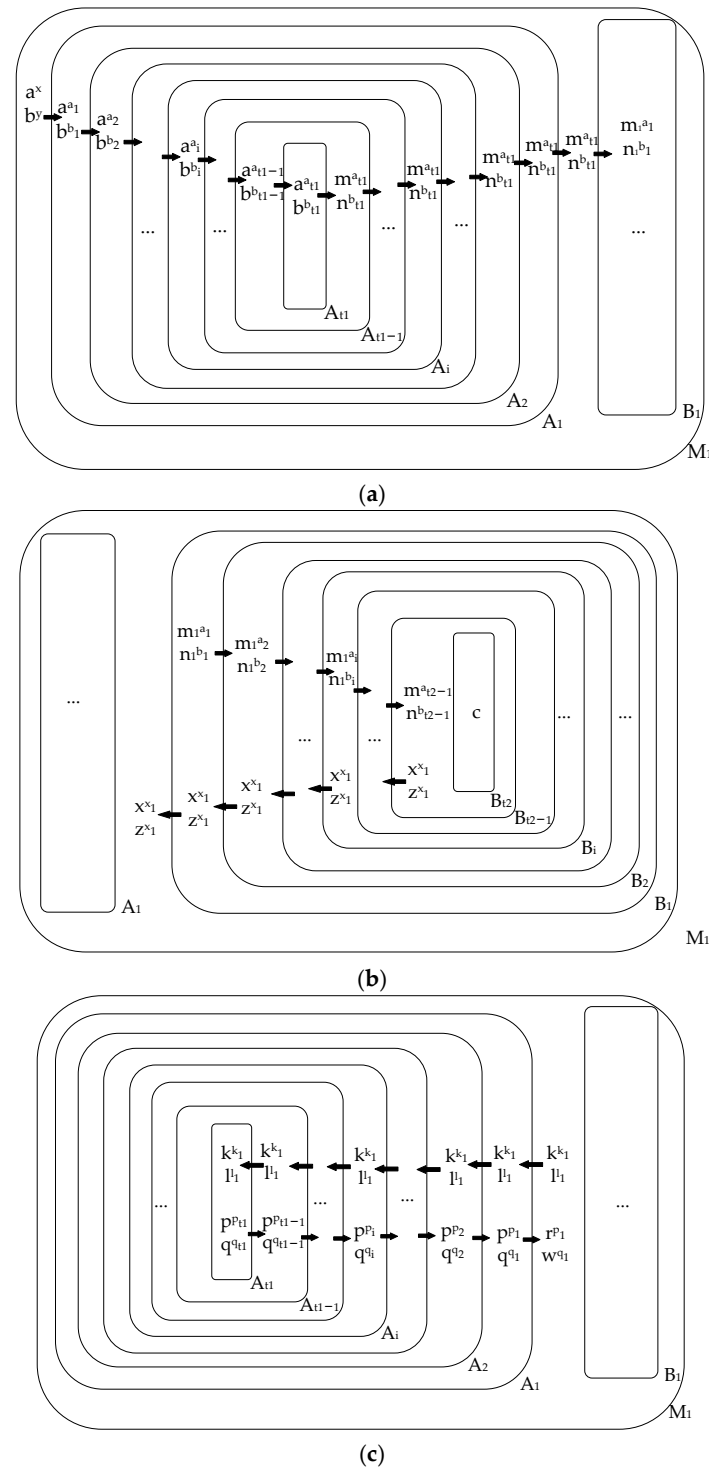
The following is a specific description of the membrane rules.

(1) Calculate the larger value of the partition  $a$  and the smaller value of  $b$ .

The multiset  $a^x b^y ct$  (here  $x$  is the numerator value and  $y$  is the denominator value) is put into the membrane  $M_1$ , and the magnitudes of the numerator and denominator are compared in the membrane  $M_1$ . When the numerator is greater than the denominator, the number of objects  $a$  and  $b$  are directly passed into the membrane  $A_1$ ; otherwise, the rules are executed to convert the object  $b$  to  $a$  and directly passed to the membrane  $A_1$  and swapping the order of numerators and denominators in membrane  $A_1$ . At this point,

there are no other rules to execute in membrane  $M_1$ , so we wait for membrane  $A_1$  to finish calculating the intermediate values  $m, n$  and then continue the execution.

Now the number of objects  $a$  and  $b$  in each layer of class A membrane is designed as a sequence  $\{a_i\}, \{b_i\}$ . For example, the number of object  $a$  in membrane  $A_1$  is  $a_1$ , and the number of  $b$  in membrane  $A_{t_1}$  is  $b_{t_1}$ .



**Figure 9.** Flow chart for solving the combined method P System. (a) Flow chart for the combination of the two methods to find the intermediate values  $m, n$ ; (b) Flow chart for finding the maximum common divisor by the division algorithm; (c) Flow chart for the sequence  $\{p_i\}, \{q_i\}$  by the more phase derogation algorithm.

Initialize the membrane  $A_1$  in  $a_1, b_1$ . The rules are as follows:

$$\begin{aligned} r_1 : (a \ b \rightarrow g, 1) & \quad r_2 : (a \rightarrow a2(a \text{ Num}, \text{in all}) \mid g, 2) \\ r_3 : (b \rightarrow b2(a \text{ Num}, \text{in all}) \mid g, 2) & \quad r_4 : (g \rightarrow (a \ b \text{ Num}, \text{in all}), 3) \end{aligned}$$

- (i) When  $x > y$ , the rules within membrane  $M_1$  are executed in the following order:  $r_1 \rightarrow r_2 \rightarrow r_4$ .
- (a) Execution rule  $r_1$ : objects  $a$  and  $b$  are consumed and the multiset  $a^y b^y$  generates the multiset  $g^y$ , with  $a^{x-y}$  remaining. This means that the numerator and denominator values are consumed in  $y$  copies at the same time, leaving  $x - y$  of object  $a$ ;
  - (b) Execute the rule  $r_2$  under the condition that object  $g$  exists, passing the remaining  $a$  to object  $a$  in the submembrane;
  - (c) Execute  $r_4$  to convert the multiset  $g^y$  into  $a^y b^y$  and transfer it to the submembrane; at this time, the number of  $a$  in membrane  $A_1$  and membrane  $B_1$  is equivalent to performing a cumulative addition to obtain  $a_1 = x, b_1 = y$ . The multiplicity set,  $Num^x$ , is used to generate membranes dynamically.
- (ii) When  $x < y$ , the rules in membrane  $M_1$  are executed in the following order:  $r_1 \rightarrow r_3 \rightarrow r_4$ .
- (a) Execution rule  $r_1$ : objects  $a$  and  $b$  are consumed and the multiset  $a^x b^x$  generates the multiset  $g^x$ , with  $b^{y-x}$  remaining. This means that  $x$  copies of the numerator and denominator are consumed simultaneously, leaving  $y - x$  objects,  $b$ ;
  - (b) Execute rule  $r_3$  in the presence of object  $g$ , passing the remaining  $b$  to object  $a$  in the submembrane;
  - (c) Execute  $r_4$  to convert the multiset  $g^x$  into  $a^x b^x$  and transfer it to the submembrane; at this time, the object  $a$  in membrane  $A_1$  and membrane  $B_1$  is also equivalent to performing an addition, obtaining  $a_1 = y, b_1 = x$ . The multiset  $Num^y$  is used for the dynamic generation of membranes.

(2) Calculate the sequence  $\{a_i\}, \{b_i\}, \{u_i\}, \{z_i\}$  and the intermediate values  $m, n$

The process is performed by the membrane  $A_1$  and the dynamically generated class A submembrane. Initially,  $a^{a_1} b^{b_1} c d^{\xi} Num^{a_1} CreateSubMem$  is placed in membrane  $A_1$ . Within membrane  $A_1$  and other newly created submembranes, object  $c$  evolves into  $k, e, j$ , and  $k$  is used to control the execution of any dynamical membrane;  $e$  is involved in generating sequences  $\{z_i\}$ ; object  $j$  is used to determine the special case where  $b_i$  and difference are directly equal;  $a_1$  and  $b_1$  denote the larger and smaller values in the fractional equation, respectively; the number of  $d$  is the value of the parameter  $\xi$  in the conversion of the more phase derogation algorithm to division algorithm, and  $CreateSubMem$  is used for the dynamic generation of submembranes.

Calculate the sequence  $\{a_i\}, \{b_i\}, \{z_i\}$  and the intermediate values  $m, n$  by the following rules:

$$\begin{aligned} r_1 : (a \ b \rightarrow x \ y, 1); \quad r_2 : (c \rightarrow k \ e \ j, 1); \quad r_3 : (a \rightarrow a_1 u \ i, 2) \\ r_4 : (d \ i \rightarrow g, 1); \quad r_5 : (i \ g \rightarrow f, 1); \quad r_6 : (y \ u \rightarrow \lambda \mid f); \\ r_7 : (CreateSubMem \ Num \rightarrow A : a \{ \} \mid \lambda); \\ r_8 : (Num \rightarrow (Num, \text{in all}) \mid CreateSubNum); \quad r_9 : (y \ e \rightarrow \delta(z, \text{in all}) \mid \lambda, 4) \\ r_{10} : (k \rightarrow (c, \text{in all}) \mid \lambda); \quad r_{11} : (x \rightarrow (a, \text{in all}) \mid \delta, 4); \quad r_{12} : (a_1 \rightarrow (b, \text{in all}) \mid \delta, 4) \\ r_{13} : (u \ e \rightarrow h \mid \lambda); \quad r_{14} : (x \rightarrow (b, \text{in all}) \mid h, 4); \quad r_{15} : (a_1 \rightarrow (a, \text{in all}) \mid h, 4) \\ r_{16} : (j \rightarrow j_1 \mid y \ \& \ !u \ \& \ \lambda); \quad r_{17} : (j_1 \rightarrow p^2 q \mid !y \ \& \ !u \ \& \ \lambda); \quad r_{18} : (g \rightarrow o \mid !i \ \& \ !f, 2) \\ r_{19} : (x \rightarrow (m \ n, \text{out}) \mid o); \quad r_{20} : (a_1 \rightarrow (m, \text{out}) \mid o) \\ r_{21} : (m \rightarrow (m, \text{out}) \mid !o); \quad r_{22} : (n \rightarrow (n, \text{out}) \mid !o) \end{aligned}$$

- (i) In class A membranes, the rules are executed in the following order:  $\{r_1, r_2\} \rightarrow r_3 \rightarrow r_4 \rightarrow r_5$ . Taking membrane  $A_1$  as an example, the multiset  $x^{b_1} y^{b_1}$  generated by executing rule  $r_1$  is the value used for temporary storage  $b_1$  and is used to compare with  $\xi$  to determine whether to convert;  $r_2$  converts  $c$  to  $k, e, j$ ; execute  $r_3$  to convert the

remaining multiset  $a^{a_1-b_1}$  to multiset  $a_1^{u_1}u^{u_1}i^{u_1}$  for backup ( $u_1 = a_1 - b_1$ ):  $a_1$  passes the difference  $u_1$  to the submembrane and does not participate in other comparisons; object  $u$  compares size with  $y$  and decides how to pass  $A_2$  and  $b_2$  to submembrane  $a_2$ ; and object  $i$  compares value with  $d$  and determines whether to end the more phase derogation algorithm and start using the division algorithm. Execute  $r_4$  to compare the difference between  $u_1$  and  $\xi$ , at this point.

- (ii) When  $u_1 > \xi$  (i.e., after executing  $r_4$ , object  $i$  remains in the membrane after the execution), the execution of  $r_5$  converts  $i$  and  $g$  to  $f$  and continues using the more phase derogation algorithm, the order of execution in this case is  $r_6 \rightarrow \{r_7, r_{10}\} \rightarrow \{r_8, r_9\} \rightarrow \{r_{11}, r_{12}\}$  or  $r_6 \rightarrow \{r_7, r_{10}\} \rightarrow \{r_8, r_{13}\} \rightarrow \{r_{14}, r_{15}\}$ .
  - (a) Execute rule  $r_6$  to convert  $y$  and  $u$  into object  $\lambda$  to determine the magnitude of the subtraction and the difference. When there is  $\lambda$  in the membrane,  $r_7$  is executed to dynamically generate Class A submembrane. After the object *CreateSubMem* is consumed, it indicates that the submembrane has been generated. At the same time,  $r_{10}$  is executed to transfer  $c$  into the submembrane  $A_2$ .
  - (b) As the *CreateSubMem* in the membrane is consumed,  $r_8$  is then executed to pass the remaining multiset,  $Num^{a_1-1}$ , into the submembrane for continued dynamic generation of the membrane. At this point, if the object  $y$  is still remaining within the membrane, it indicates that the minus number is larger and that both the larger and smaller values in the submembrane need to be replaced. Execution of  $r_9$  generates  $\delta$  within this membrane and also generates  $z$  to be passed into the inner membrane  $A_2$ , i.e.,  $z_2 = 1$ .
  - (c) Object  $\delta$  serves as the condition for executing  $r_{11}$ , passing the subtractive backup  $x^{b_1}$  of the membrane to the subtractive  $a$  of the submembrane, and executing  $r_{12}$  to transfer the differential backup  $a_1^{u_1}$  to the subtractive  $b$  of the submembrane, i.e.,  $a_2 = b_1$ ,  $b_2 = u_1$ .
  - (d) If there is still object  $u$  in membrane  $A_1$ , it proves that only the subtracted number  $a$  needs to be replaced in the submembrane  $A_2$ , and the value of  $b$  is equal to that of the membrane  $A_1$ , i.e.,  $a_2 = u_1$ ,  $b_2 = b_1$ , and  $z_2 = 0$  (indicating that there is no object  $z$  inside the submembrane  $A_2$  and no need to execute the rule); at this time, there are still objects  $u$  and  $e$  in the membrane, and executing  $r_{13}$  generates  $h$ ; under the catalysis of  $h$ , execute  $r_{14}$  to transfer the backup subtracted number  $x^{b_1}$  to the subtracted number  $b^{b_2}$  in the submembrane, and execute  $r_{15}$  to transfer the difference backup  $a_1^{u_1}$  to the subtracted number  $a^{a_2}$  in the submembrane.

$\{r_{16}, r_{17}\}$  is aimed at the special case where the subtraction  $b_i$  is directly equal to the difference  $u_i$ , and there is no need for tossing and dividing. They directly generate  $p^2q$  to start the more phase decrement improvement part. For details, please refer to Section 4.1.2 of this paper.

- (iii) When  $u_1 \leq \xi$ , the order of rule execution in this membrane, at this time, is  $r_{18} \rightarrow \{r_{19}, r_{20}\}$ . When the object  $g$  exists and both  $i$  and  $f$  do not exist, execute  $r_{18}$  to convert  $g$  to  $o$ . The presence of  $o$  indicates that the P System is to be converted from the more phase derogation algorithm to the division algorithm; stop generating new membranes and execute  $r_{19}$  and transfer the backup subtractor  $x^{b_1}$  to  $m$  and  $n$  in the parent membrane; execute  $r_{20}$  and transfer the backup difference  $a_1^{u_1}$  to  $m$  in the parent membrane. Execute  $r_{20}$ , pass the backup difference  $a_1^{u_1}$  to  $m$  in the parent membrane; at which point, the number of  $m$  in the parent membrane is  $a_1$ .

The above rules take membrane  $A_1$  as an example. In addition to membrane  $A_1$  requiring initialization of  $a_1b_1$ , other rules also apply to membranes  $A_2, A_3, \dots, A_{t_1}$ . All membranes except membrane  $A_1$  in type A membrane execute  $\{r_{21}, r_{22}\}$  after receiving  $m, n$  values, and transmit the received multiset,  $m^m n^n$ , directly to the parent membrane, and finally from membrane  $A_1$  to membrane  $M_1$ .

- (3) Calculation of the maximum convention  $x_1$



The relevant rules for implementing this function in class B membranes are as follows:

$$\begin{aligned}
 r_1 : (m_1 \ n_1 \rightarrow x, 1) \quad r_2 : (c \rightarrow y \mid_x, 1) \quad r_3 : (m_1 \ y \rightarrow m_1 \ d, 2) \\
 r_4 : (x \rightarrow n_1 \mid_d, 3) \quad r_5 : (d \rightarrow c, 4) \quad r_6 : (n_1 \ y \rightarrow n_1 \ e \ g, 2) \\
 r_7 : (\text{CreateSubMem Num} \rightarrow B : b\{\} \mid_{n_1}, 3) \\
 r_8 : (\text{Num} \rightarrow (\text{Num}, \text{in all}) \mid_{\text{CreateSubMem}}) \\
 r_9 : (g \rightarrow h, 3) \quad r_{10} : (x \rightarrow (n_1, \text{in all}) \mid_h, 4) \quad r_{11} : (n_1 \rightarrow (m_1, \text{in all}) \mid_h, 4) \\
 r_{12} : (h \rightarrow (c, \text{in all}) \mid_h, 5) \quad r_{13} : (y \rightarrow z \mid_{m_1} \ \& \ !_{n_1}, 3) \quad r_{14} : (x \rightarrow (z \ x, \text{out}) \mid_z, 4) \\
 r_{15} : (a \rightarrow \lambda, 1) \quad r_{16} : (b \rightarrow \lambda, 1)
 \end{aligned}$$

The procedure for finding the greatest common divisor  $x_1$  of intermediate values  $m$  and  $n$  in a class B membrane is the same as that for finding the greatest common divisor  $x$  in the division algorithm P System, as described in Section 4.2.2.

(4) Calculating the intermediate value simplification results  $k_1, l_1$

The procedure for finding the  $m$  and  $n$  simplification results  $k_1/l_1$  in membrane  $M_1$  is the same as that for simplifying  $k/l$  in the division algorithm P System, as described in Section 4.2.2

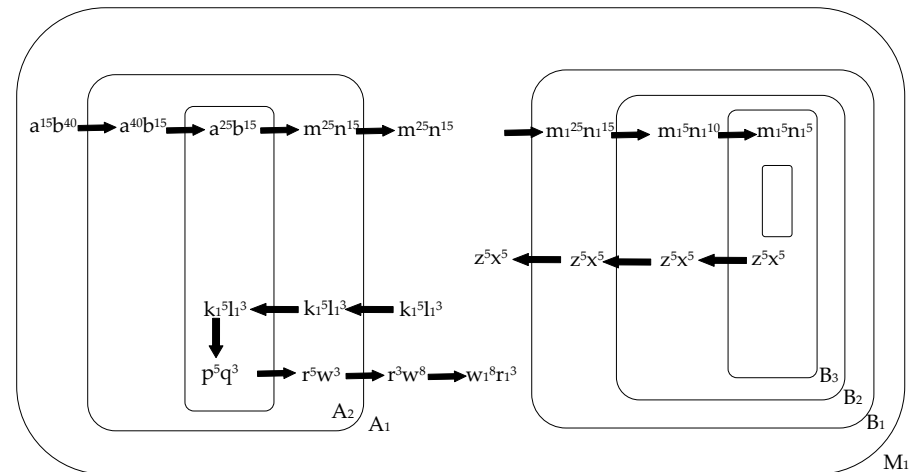
(5) Compute the sequence  $\{p_i\}, \{q_i\}$

$$\begin{aligned}
 r_{23} : (k_1 \rightarrow (k_1, \text{in all}) \mid_{!o}, 1); \quad r_{24} : (l_1 \rightarrow (l_1, \text{in all}) \mid_{!o}, 1); \quad r_{25} : (k_1 \rightarrow p, 2); \\
 r_{23} : (k_1 \rightarrow (k_1, \text{in all}) \mid_{!o}, 1); \quad r_{24} : (l_1 \rightarrow (l_1, \text{in all}) \mid_{!o}, 1); \quad r_{25} : (k_1 \rightarrow p, 2); \\
 r_{26} : (l_1 \rightarrow q, 2); \quad r_{27} : (p \rightarrow (r, \text{out})); \quad r_{28} : (q \rightarrow (w, \text{out})); \\
 r_{29} : (z \rightarrow (o_1, \text{out})); \quad r_{30} : (r \rightarrow p \mid_{o_1}); \quad r_{31} : (w \rightarrow p \mid_{o_1}); \\
 r_{32} : (w \rightarrow p \mid_{!o_1}); \quad r_{33} : (r \rightarrow p \mid_{!o_1});
 \end{aligned}$$

- (i) Firstly, the membrane  $M_1$  sends  $k_1, l_1$  to membrane  $A_1$  and then executes  $\{r_{23}, r_{24}\}$  to gradually transfer  $k_1$  and  $l_1$  to the submembrane into the submembrane. When  $k_1$  and  $l_1$  reaches the innermost membrane, the order of execution in the innermost class A substratum is  $\{r_{25}, r_{26}\} \rightarrow \{r_{27}, r_{28}, r_{29}\}$ .
  - (a) Simultaneous execution of rules  $r_{25}$  and  $r_{26}$  translate the quantities of  $k_1$  and  $l_1$  into the initial values  $p_{t_1}$  and  $q_{t_1}$ ;
  - (b) Under the condition that multiset of  $p^{p_i} q^{q_i}$  exists within membrane  $A_i (i = t_1, \dots, 2, 1)$ , simultaneous execution of  $r_{27}, r_{28}$  conversion to  $r^{p_i} w^{q_i}$  is passed directly to the parent membrane  $A_{i-1}$ , with  $r_{29}$  deciding whether or not to execute based on the presence or absence of  $z$  in each layer, and. if the layer has  $z$ , convert  $z$  into  $o_1$  and output it to the parent membrane.
- (ii) The order of execution of all class A membranes is  $\{r_{30}, r_{31}\} \rightarrow \{r_{27}, r_{28}, r_{29}\}$  or  $\{r_{32}, r_{33}\} \rightarrow \{r_{27}, r_{28}, r_{29}\}$ .
  - (a) If  $o_1$  is received from the parent membrane, it means that the minus number changes to the minus number during the more phase derogation algorithm, so it is necessary to convert the multiset  $r^{p_i}$  to  $p^{p_i} q^{p_i}$  and then the multiset  $w^{q_i}$  to the object  $p$ . At this point,  $p_{i-1} = p_i + q_i, q_{i-1} = p_i$ , i.e., the execution  $\{r_{30}, r_{31}\}$ . If no  $o_1$  is passed in the substratum, it means that the decrement is not replaced. Execute  $r_{32}, r_{33}$  directly, converting objects  $w$  to  $p, q$  and  $r$  to  $p$ , i.e.,  $p_{i-1} = p_i + q_i, q_{i-1} = q_i$ .
  - (b) Any class A membrane with  $\{r_{27}, r_{28}, r_{29}\}$  performs the same process as the innermost submembrane, and finally membrane  $A_1$  performs  $\{r_{27}, r_{28}\}$  to output the multiset  $r^{p_1} w^{q_1}$  to  $M_1$  in the membrane, which is then assigned according to the size of the numerator and denominator: when the numerator  $>$  denominator, the fraction simplification result is  $r^{p_1} / w^{q_1}$ ; when the numerator  $<$  denominator, the objects  $r$  and  $w$  are swapped in membrane  $M_1$  and transformed into  $w_1$  and  $r_1$ , respectively, and the final fraction simplification result is  $r_1^{q_1} / w_1^{p_1}$ .

#### 4.3.3. Examples of Fractional Simplification by Combining the Two Methods

In this subsection, we will give an example to describe in detail how to implement fractional simplification in a P System using a combination of the more phase derogation algorithm and the division algorithm. For example, the reduction process for a P System taking a manifold of  $15/40$  with  $\xi = 20$  is shown in Figure 10:



**Figure 10.** Simplified flow chart for approximate differentiation of the two combined examples.

The following details the execution process of rules in this P System.

##### (1) Initial pattern

First, put the multiset  $a^{15}b^{40}ct$  in the membrane  $M_1$ ; where: the number of object  $a$  is 15, which is used to represent the numerator of the partition; the number of object  $b$  is 40, which is used to represent the denominator. Preprocessing in membrane  $M_1$ , executing rules in the order  $r_1 \rightarrow r_3 \rightarrow r_4$ : execute rule  $r_1$  fifteen times until object  $a$  is fully consumed, generating  $g^{15}$ ; execute rule  $r_3$  twenty-five times in the presence of object  $g$ , converting multiset  $b^{25}$  to multiset  $b_2^{25}$  and generating multiset  $a^{25}Num^{25}$  to be fed into submembranes  $A_1, B_1$ ; after executing rule  $r_4$ , the multiset  $g^{15}$  generates the multiset  $a^{15}b^{15}Num^{15}$  and feeds it into submembrane  $A_1, B_1$ ; at this point, the multiset in membrane  $M_1$  is  $b_2^{25}ct$ .

##### (2) More phase derogation algorithm

- (i) At this time, the multiset in membrane  $A_1$  is  $a^{40}b^{15}d^{20}cNum^{40}$  *CreateSubMem*, and the order of execution of the rules is:  $\{r_1, r_2\} \rightarrow r_3 \rightarrow r_4 \rightarrow r_5 \rightarrow r_6 \rightarrow \{r_7, r_{13}\} \rightarrow \{r_8, r_{10}, r_{14}, r_{15}\}$ .
  - (a) Execute the rule  $r_1$ , and the multiset  $a^{15}b^{15}$  evolves to  $x^{15}y^{15}$ ; while executing the rule  $r_2$ , the object  $c$  evolves to  $k, e, j$ ;
  - (b) Execute the rule  $r_3$ , and the multiset  $a^{25}$  evolved to  $a_1^{25}u^{25}i^{25}$ ; then the rule  $r_4$  is executed to compare the numerical values of 25 and  $\xi$ , and generate the multiset  $g^{20}$ ;
  - (c) Because of the remaining  $i^5$ , the conditional execution rule  $r_5$  is met and the multiset  $i^5g^5$  evolves to  $f^5$ ;
  - (d) Under the condition that the object  $f$  exists, execute the rule  $r_6$ , and the multiset  $y^{15}u^{15}$  evolves to generate the multiset  $\lambda^{15}$ ;
  - (e) If object  $\lambda$  exists, the System executes the rule  $r_7$  to generates a new submembrane  $A_2$ , while executing  $r_{13}$  to evolve  $u$  and  $e$  into object  $h$ ;
  - (f) Simultaneous execution of rules  $r_8, r_{10}, r_{14}, r_{15}$  and  $a^{25}b^{15}Num^{39}c$  is passed directly into the newly created submembrane  $A_2$ . At which point, the remaining multiset in membrane  $A_1$  is  $\lambda^{15}u^9f^5g^{15}jh$ .
- (ii) At this point, the initialized multiset in membrane  $A_2$  is  $a^{25}b^{15}Num^{39}c$  *CreateSubMem*, and the rules are executed in the order  $\{r_1, r_2\} \rightarrow r_3 \rightarrow r_4 \rightarrow r_{18} \rightarrow \{r_{19}, r_{20}\}$ .

- (a) Execute the rule  $r_1$ , and the multiset  $a^{15}b^{15}$  evolves to multiset  $x^{15}y^{15}$ ; while executing the rule  $r_2$ ,  $c$  evolves to multiset  $k,e,j$ ;
- (b) Execute the rule  $r_3$ , and the multiset  $a^{10}$  is evolved to  $a_1^{10}u^{10}i^{10}$ ; then execute the rule  $r_4$  to evolve  $d^{10}i^{10}$  to multiset  $g^{10}$ ; then the multiset in the membrane  $A_2$  is  $x^{15}, y^{15}, a_1^{10}, u^{10}, g^{10}, d^{10}, k, e, j, Num^{39}, CreateSubMem$ .
- (c) Execute the rule  $r_{18}$  in the absence of object  $i$  and object  $f$ . The multiset  $g^{10}$  evolves into  $o^{10}$ ; by executing rule  $r_{19}, r_{20}$  in the presence of  $o$ , the multisets  $x^{15}$  and  $a_1^{10}$  evolve into the multisets  $m^{15}n^{15}$  and  $m^{10}$ , respectively, which are transferred to the parent membrane  $A_1$ .

At this point, the multiset in membrane  $A_2$  is  $y^{15}, u^{10}, o^{10}, d^{10}, k, e, j, Num^{39}, CreateSubMem$ , and the multiset in membrane  $A_1$  is  $m^{25}, n^{15}, \lambda^{15}, u^9, f^5, g^{15}j, h$ ; execute the rules  $r_{21}, r_{22}$  simultaneously without object  $o$ ; the multiset  $m^{25}n^{15}$  is fed into the membrane  $M_1$ .

### (3) Division algorithm

- (i) The multiset in membrane  $B_1$  is  $m_1^{25}n_1^{15}cNum^{40}CreateSubMem$ , and the rules are executed in the order  $r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow r_5$ .
  - (a) Implementation rules  $r_1$ , multiset  $m_1^{15}n_1^{15}$  evolves to  $x^{15}$ ; then execute the rule  $r_2$  that evolves object  $c$  to  $y$ , subject to the existence of object  $x$ ;
  - (b) Implementation rules  $r_3$ , multiset  $m_1y$  evolves to  $m_1d$ ; execute the rule  $r_4$  under the condition that object  $d$  exists, so that  $x^{15}$  evolves into the multiplicity set  $n_1^{15}$ ; then execute the rule  $r_5$ , and the object  $d$  evolves to object  $c$ ;
- (ii) At this time, the multiset in membrane  $B_1$  is  $m_1^{10}n_1^{15}cNum^{40}CreateSubMem$ , and the rules are executed in the order  $r_1 \rightarrow r_2 \rightarrow r_6 \rightarrow \{r_7, r_9\} \rightarrow \{r_8, r_{10}, r_{11}\} \rightarrow r_{12}$ .
  - (a) Execute the rule  $r_1$  that evolves  $m_1^{10}n_1^{10}$  to the multiset  $x^{10}$ ; execute the rule  $r_2$  to evolve  $c$  into  $y$  under the condition that object  $x$  exists; then execute the rule  $r_6$  to evolve the multiset  $n_1, y$  to  $n_1, e, g$ ;
  - (b) Execution rules  $r_7, r_9$ , consume  $CreateSubMem$  and a  $Num$  to generate a sub-membrane  $B_2$ , while the object  $g$  evolves to  $h$ ;
  - (c) Under the condition that the multiset  $h$  exists, execute the rule  $r_8, r_{10}, r_{11}$  that transfers the multiset  $Num^{39}n_1^{10}m_1^5$  to the membrane  $B_2$ ;
  - (d) Execute the rule  $r_{12}$ , and the multiset  $h$  evolves into  $c$  and feeds into the membrane  $B_2$ .
- (iii) The initialized multiset in membrane  $B_2$  is  $m_1^5n_1^{10}cNum^{39}CreateSubMem$ . The rules are executed in the same order  $r_1 \rightarrow r_2 \rightarrow r_6 \rightarrow \{r_7, r_9\} \rightarrow \{r_8, r_{10}, r_{11}\} \rightarrow r_{12}$ . After the rules are executed, the remaining multiset in membrane  $B_2$  is  $e$ . The multiset obtained from membrane  $B_3$  is  $m_1^5n_1^5cNum^{38}$ .
- (iv) At this point, the initialized multiset in  $B_3$  is  $m_1^5n_1^5cNum^{38}CreateSubMem$ . And the rules are executed in the order  $r_1 \rightarrow r_2 \rightarrow r_{13} \rightarrow r_{14}$ .
  - (a) Implementation rules  $r_1$ , multiset  $m_1^5n_1^5$  evolves to  $x^5$ , and then, execute the rule  $r_2$  to evolve  $c$  into the object  $y$ ;
  - (b) Under the condition that there are no objects  $m_1$  and  $n_1$ , execute rules  $r_{13}$  that evolves  $y$  to  $z$ ; at this point, the condition is met to execute  $r_{14}$ , the multiset  $x^5$  evolves to  $z^5x^5$  and is fed into membrane  $B_2$ ;

In membrane  $B_2$ , continue to execute the rule  $r_{14}$  to generate the multiset  $z^5x^5$  in membrane  $B_1$ ; membrane  $B_1$  also executes the rule  $r_{14}$ , and finally sends the multiset  $z^5x^5$  into the membrane  $M_1$ ; at this point, the number of  $x$  is the maximum number of conventions  $x_1$ .

### (4) Calculate the median simplification result $k_1, l_1$

At this point, the multiset in membrane  $M_1$  is  $a_1^{25}b_1^{15}b_2^{25}z^5x^5ct$ . The execution process of the rule is the same as the approximate partitioning process of the example in Section 4.2.3, and will not be described too much here. After the rule is executed, the

multiset in membrane  $M_1$  is  $l_1^3 k_1^5 f^5 i^5 b_2^{25}$ , and the multiset passed into membrane  $A_1$  is  $l_1^3 k_1^5$ .

(5) Calculate the final result

- (i) At this point, the multiset in membrane  $A_1$  is  $\lambda^{15} u^9 g^{15} f^5 h j l_1^3 k_1^5$ , and the condition is met to execute rule  $r_{23}, r_{24}$  to pass the multiset  $l_1^3 k_1^5$  into membrane  $A_2$ , respectively.
- (ii) When membrane  $A_2$  receives a multiset  $l_1^3 k_1^5$  from the parent membrane, the rules are executed in the order  $\{r_{25}, r_{26}\} \rightarrow \{r_{27}, r_{28}\}$ .
  - (a) First executing rule  $r_{25}, r_{26}$  in membrane  $A_2$ , converts the multisets  $k_1^5$  and  $l_1^3$  into  $p^5$  and  $q^3$ , respectively;
  - (b) Then executing rule  $r_{27}, r_{28}$  converts the multiset  $p^5 q^3$  into a multiset  $r^5 w^3$  to pass directly into membrane  $A_1$ ;
- (iii) The order of execution in membrane  $A_1$ , at this time, is:  $\{r_{32}, r_{33}\} \rightarrow \{r_{27}, r_{28}\}$ .
  - (a) In the absence of object  $o_1$ , rule  $r_{32}, r_{33}$  is executed at the same time, and the multiset  $r^5 w^3$  evolves into  $p^8 q^3$ ;
  - (b) Execute rule  $r_{27}, r_{28}$  to convert the multiset  $p^8 q^3$  to  $r^8 w^3$  and feed it into the membrane  $M_1$ .
- (iv) When the membrane  $M_1$  receives the multiset  $r^8 w^3$ , the object  $b_2$  is generated when the rule is executed because the numerator is smaller than the denominator, so the corresponding rule is executed to evolve the multiset  $r^8 w^3$  into the multiset  $w_1^8 r_1^3$ . At this time, the multiset in the membrane  $M_1$  is  $l_1^3 k_1^5 f^5 i^5 b_2^{25} w_1^8 r_1^3$ , and there is no rule to execute in the System, so the whole System is stopped. The number of objects  $w_1$  and  $r_1$  represent the values of denominator and numerator, respectively, so the final reduction result of  $15/40$  is  $3/8$ .

Table 3 shows the partial process of simplifying  $15/40$  in a combinatorial P System  $\Pi_3$ , including system initialization, obtaining the maximum convention  $x$ , calculating the intermediate value reduction results  $k_1$ ,  $l_1$ , and other key time slices. The complete execution process is shown in Appendix A.

From the above process analysis, it is clear that the total time slice for this example is 61.

**Table 3.** Partial time slice of a combinatorial P System  $\Pi_3$ .

Time Slice	Rules for Implementation	Results of the Implementation
Initial Status	None	$M_1: a^{15}, b^{40}, c, t;$ $A_1: c, d^{20}, \text{CreateSubMem};$ $B_1: c, \text{CreateSubMem};$
2	$M_1: r_3, r_4;$	$M_1: c, t, b_2^{25};$ $A_1: a^{40}, b^{15}, d^{20}, e, \text{Num}^{40}, j, k, \text{CreateSubMem};$ $B_1: a^{40}, b^{15}, c, \text{Num}^{40}, \text{CreateSubMem};$
7	$A_1: r_{13}, r_7, r_{10};$	$M_1: c, t, b_2^{25};$ $A_1: \text{Num}^{39}, u^9, f^5, g^{15}, x^{15}, h, j, \lambda^{15}, a_1^{25};$ $A_2: c, d^{20}, \text{CreateSubMem};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$
13	$A_1: r_{30}, r_{31};$	$M_1: c, t, m^{25}, n^{15}, b_2^{25};$ $A_1: u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, \text{Num}^{39}, e, u^{10}, y^{15}, j, k, \text{CreateSubMem}, o^{10};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$
14	$M_1: r_6, r_5;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, \text{Num}^{39}, e, u^{10}, y^{15}, j, k, \text{CreateSubMem}, o^{10};$ $B_1: c, n_1^{15}, m_1^{25}, \lambda^{55}, \text{Num}^{40}, \text{CreateSubMem};$

Table 3. Cont.

Time Slice	Rules for Implementation	Results of the Implementation
34	B <sub>1</sub> : r <sub>12</sub> ;	M <sub>1</sub> : c, t, b <sub>1</sub> <sup>15</sup> , b <sub>2</sub> <sup>25</sup> , a <sub>1</sub> <sup>25</sup> , x <sup>5</sup> , z <sup>5</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , u <sup>9</sup> , f <sup>5</sup> , g <sup>15</sup> , h, j, λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : e, λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
54	M <sub>1</sub> : r <sub>20</sub> , r <sub>24</sub> ; A <sub>2</sub> : r <sub>23</sub> ;	M <sub>1</sub> : l <sub>1</sub> <sup>3</sup> , k <sub>1</sub> <sup>5</sup> , f <sup>5</sup> , i <sup>5</sup> , w <sub>1</sub> <sup>6</sup> , r <sub>1</sub> <sup>3</sup> , b <sub>2</sub> <sup>25</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , f <sup>5</sup> , k <sub>1</sub> , g <sup>15</sup> , h, j, r, u <sup>9</sup> , λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : e, l <sub>1</sub> <sup>3</sup> , k <sub>1</sub> <sup>5</sup> , λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
61	M <sub>1</sub> : r <sub>24</sub>	M <sub>1</sub> : l <sub>1</sub> <sup>3</sup> , k <sub>1</sub> <sup>5</sup> , f <sup>5</sup> , i <sup>5</sup> , w <sub>1</sub> <sup>8</sup> , r <sub>1</sub> <sup>3</sup> , b <sub>2</sub> <sup>25</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , f <sup>5</sup> , g <sup>15</sup> , h, j, u <sup>9</sup> , λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : e, l <sub>1</sub> <sup>3</sup> , k <sub>1</sub> <sup>5</sup> , λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
62	No enforceable rules	

## 5. Experimental Results

To determine the appropriate parameters  $\xi$  for the combination of the two methods, in this experiment, we randomly selected 100 sets of data and used the P System simulation software UPSimulator for validation analysis. We compared the steps obtained when introducing different parameters into the data. The fewer steps, the shorter the time used, and the better the rules.

### 5.1. Simulation Experiments

Various P System simulation tools are listed in the literature [24]. In this paper, we make use of the P System simulation software UPSimulator (UPS for short), which is used to read the P System model described by UPL and to perform simulation experiments on the model. Therefore, this section first introduces the relevant knowledge of UPLanguage, then introduces the UPSimulator simulation software, and finally provides an example to demonstrate the simulation process.

#### 5.1.1. UPSimulator: P System Simulation Software

In order to more conveniently describe the simulation model, literature [23] introduced the language UPLanguage (UPL for short) for describing the P System, which is used to describe the membrane structure, membrane properties, initial object multiset and rule sets of the P System. It integrates object-oriented thinking and can define a membrane with a certain function as a membrane class. Membranes can be used as a type of membrane, and when used, only one statement is needed to instantiate a membrane of that type. Membranes can also be inherited by another membrane, and the inherited membrane will have all the content of that membrane and can also have its own unique content. Meanwhile, in UPL, rules are described as having two parts: conditions and results, allowing the characteristics of the rules to be arbitrarily combined.

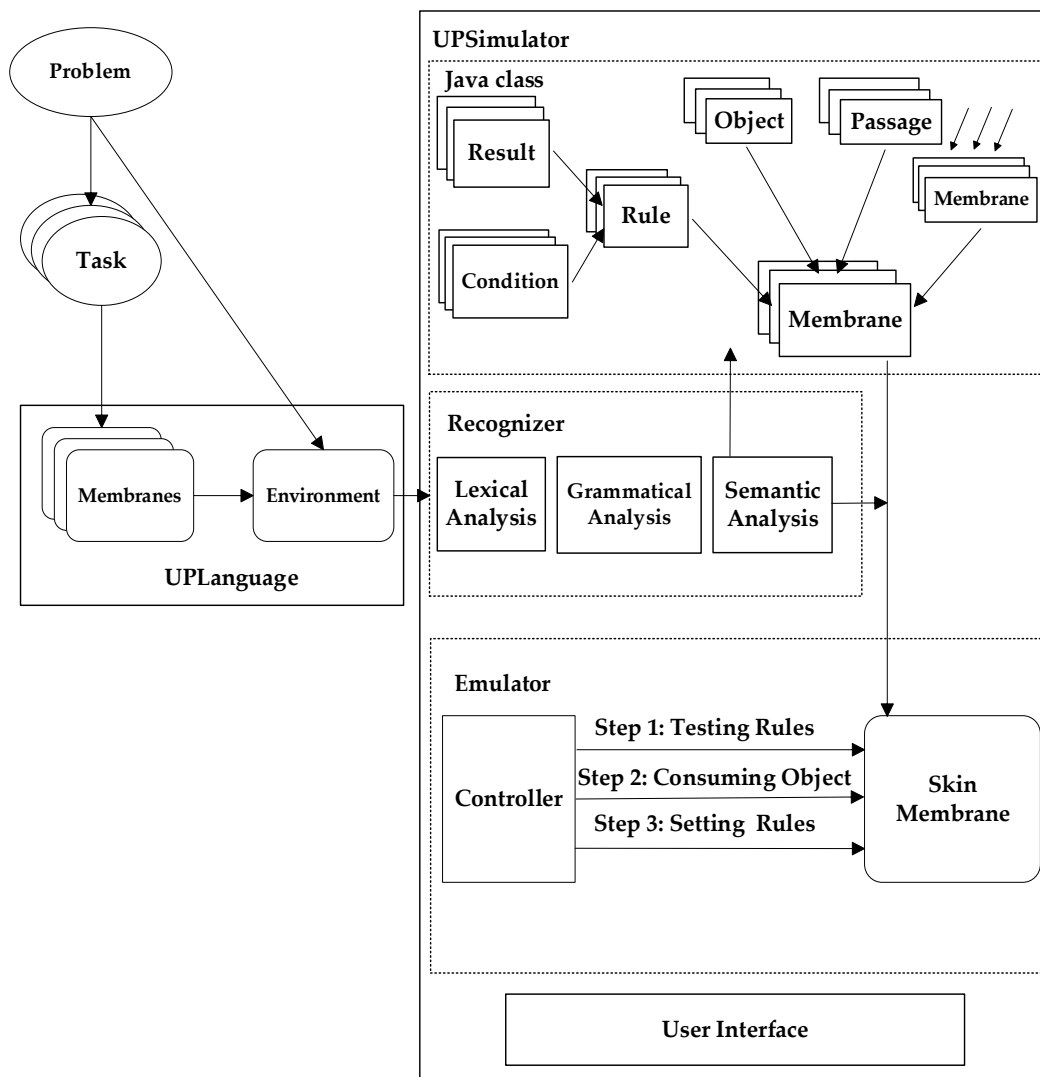
UPL will be described using the BNF paradigm. In UPL, define a membrane class, starting with the keyword ‘Membrane’, followed by the type of membrane (a combination of letters), and the membrane content enclosed in parentheses:

$\langle \text{MembraneDef} \rangle ::= \text{“Membrane”} \langle \text{MemType} \rangle [ \text{“extends”} \langle \text{MemType} \rangle \{ \text{“,”} \langle \text{MemType} \rangle \} ] \{ \{ \langle \text{Submembrane} \rangle \mid \langle \text{Objects} \rangle \mid \langle \text{RuleDef} \rangle \mid \langle \text{Properties} \rangle \} \text{“} \}$

$\langle \text{MemType} \rangle ::= \langle \text{Letters} \rangle$

The ‘extensions’ section declares which membrane classes this membrane class inherits from, and this section is optional.

After introducing UPL to describe the simulation model, the author also designed UPS simulation software, whose overall architecture is shown in Figure 11. When using the P System to solve problems, the problem can be decomposed into a series of subtasks. For each subtask, we can use one or more membranes to complete it, and each membrane that completes these tasks is a membrane class. After constructing all the required membrane classes, construct validation examples based on the problem, which is the simulation environment. Then use a recognizer to identify the environment and the string composed of all the membranes used. When the recognizer recognizes, it will continuously construct conditions, results, rules, objects, and membranes based on the identified content and place them inside the correct membranes. Finally, based on the identified simulation environment, a skin membrane is constructed. The skin membrane and its internal submembranes will evolve according to rules under the control of the controller, whose main function is to ensure parallelism and randomness during membrane execution. The user interface will display the status of the skin membrane in a timely manner, and users can also control the entire simulation process through the interface. The specific design of the P System description language and simulation software can be referred to in reference [23].



**Figure 11.** Software structure of UPS.

### 5.1.2. Simulation Examples

Taking 576/378 as an example, when  $\xi = 8$ , the initial interface is shown in Figure 12.

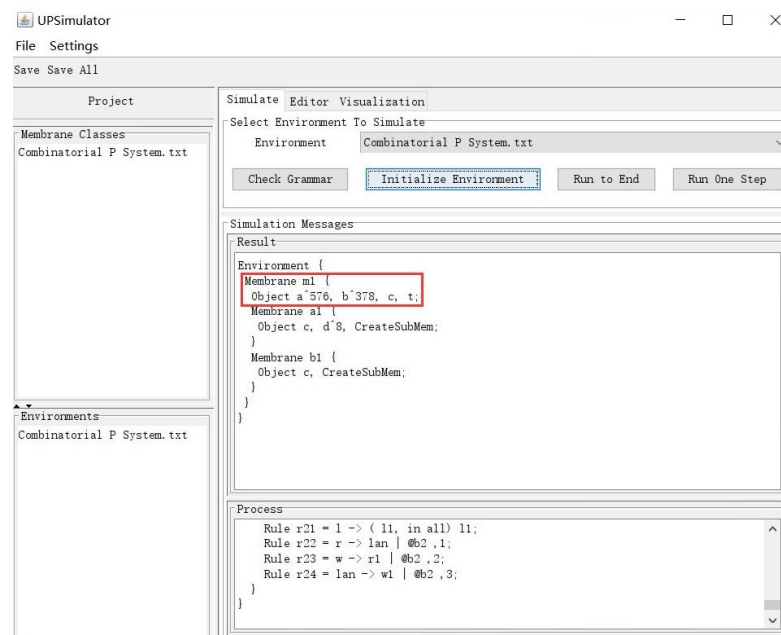


Figure 12. Initial state of the instance.

From Figure 12, it can be seen that the initial state in membrane m1 sets molecule 576 as the initial value of substance a, denominator 378 as the initial value of b (highlighted in red box in Figure 12), assigns the number of experimental parameters  $\xi$  to substance d in membrane a1, creates CreateSubMem to determine whether a new membrane is generated, and both c and t are used to control the cycle. The final experimental results are shown in Figure 13.

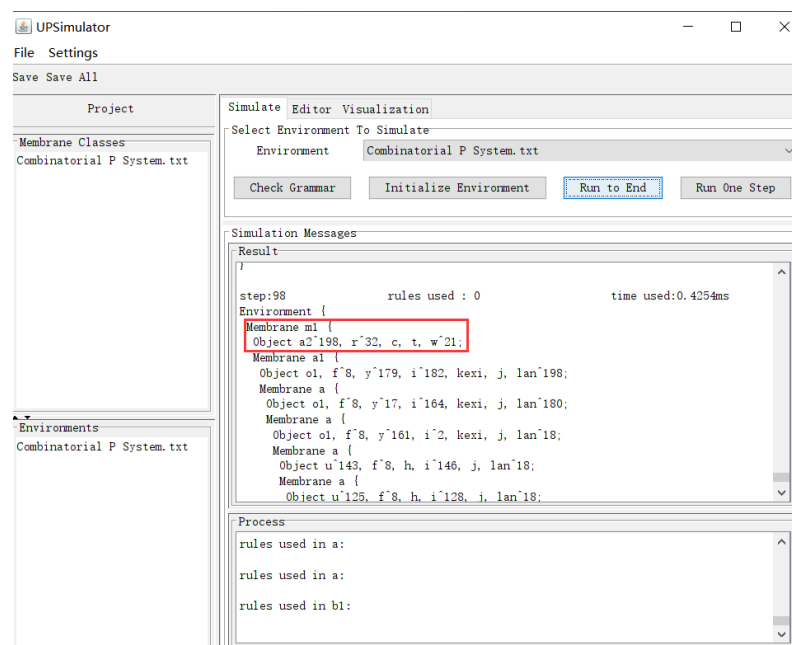


Figure 13. Example simulation results.

When the numerator is greater than or equal to the denominator (i.e.,  $a \geq b$ ), the experimental result is  $r/w$  in membrane m1. When the numerator is less than the denominator (i.e.,  $a < b$ ), the experimental result is  $r_1/w_1$  in membrane  $M_1$ . In this example, the reduction result of  $576/378$  is  $32/21$  (highlighted in red box in Figure 13), and the total number of steps consumed is 98.



## 5.2. Parameter Experiments

In this experiment, we randomly selected 100 sets of data and conducted validation analysis using UPSimulator to compare the steps (i.e., the number of time slices) obtained when introducing different parameters into the data. Group the digits with the larger value in the numerator and denominator, with a minimum digit of one and a maximum digit of six. A specific example of digit distribution is shown in Table 4.

**Table 4.** Range of experimental data values.

Serial Number	Maximum Number of Digits	Value Range	Number of Groups
1	4	0–9999	40
2	5	100–99,999	30
3	6	1000–999,999	30

Among them, for the first group with a maximum number of digits of four, we take the values of parameter  $\xi$  as 8, 20, 50, 100, 500, and 1000, respectively. For the second set of data in Table 4, add the parameter  $\xi = 10,000$ . Add an additional parameter  $\xi = 100,000$  to the third set of data. This experiment is to observe how different values of parameter  $\xi$  will affect the execution efficiency of the system. Therefore, each parameter is randomly obtained within different digits from small to large and has no specific meaning. In each P System that combines the two, parameter  $\xi$  exists in the form of substance  $d$  in class A membranes, and the membrane rules are completely the same except for the different quantities of substance  $d$  in the initial state. Due to the fact that the intermediate value of each class A film needs to be compared with parameter  $\xi$  to determine whether to switch from a more phase derogation algorithm to a division algorithm. Therefore, in class A membranes, it is not allowed to transfer substance  $d$  to the parent or daughter membranes to prevent changes in the  $x$  value in each layer of the membrane. For each generation of a class A membrane,  $\xi$   $d$  objects are fixed in the membrane and initialized. Due to the different ranges of parameter values, three sets of data were statistically analyzed separately.

### 5.2.1. Four-Digit Experimental Result

For forty randomly selected sets of data, each set of data will have six parameters substituted for the experiment, and the experimental results will be tabulated for statistical analysis. Because the number of steps may vary greatly when different parameters are used for experiments, in order to make the broken line of parameter  $\xi$  with the least number of steps more obvious, all the following line charts may be incomplete due to the long number of steps. This phenomenon is normal and not statistically incomplete.

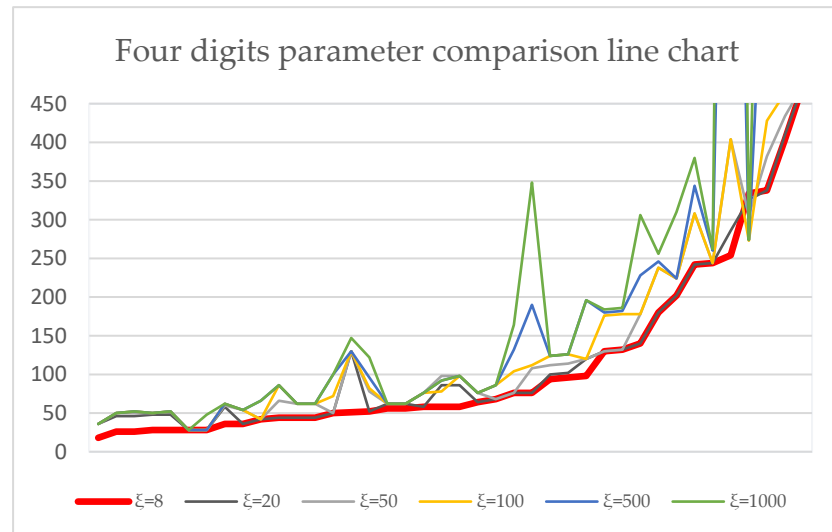
According to the discount Figure 14 made based on the experimental results, it can be seen that out of the 40 randomly selected sets of data, 39 sets of data required the least number of steps for the experiment when the parameter  $\xi = 8$ , and only one set of data had the least number of steps when  $\xi = 100$ , while the experimental steps for each set of data reached the highest value when  $\xi = 1000$ . Therefore, a preliminary conclusion is drawn that in the reduction process of fractions with a maximum of four digits, it is more appropriate to use the combination of the more phase derogation algorithm and the division algorithm with parameter  $\xi = 8$ .

### 5.2.2. Five-Digit Experimental Result

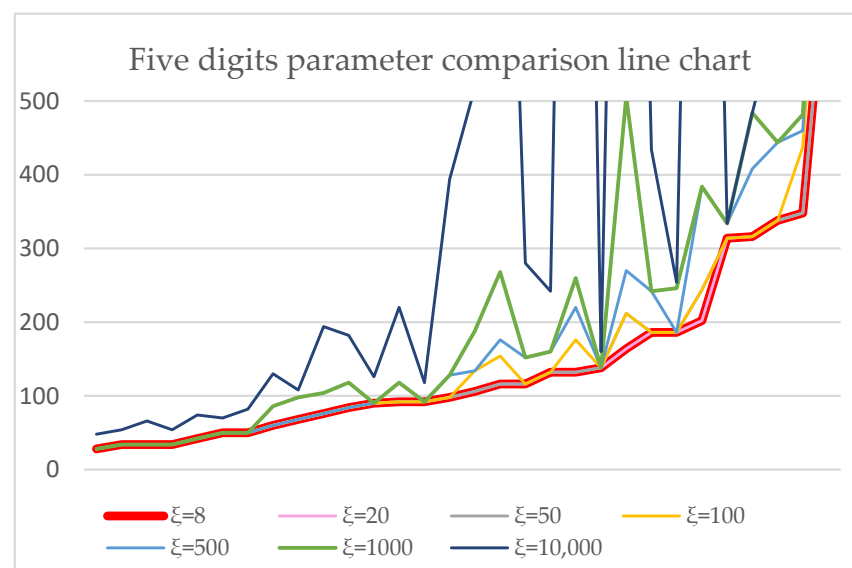
For thirty randomly selected sets of data, each set of data will have seven parameters substituted for the experiment, and the experimental results will be tabulated for statistical analysis.

According to the discount Figure 15 made based on the experimental results, it can be seen that among the 30 randomly selected sets of data, each set of data requires the least number of experimental steps when the parameter  $\xi = 8$ , while the experimental steps for each set of data reach the highest value when  $\xi = 10,000$ . Therefore, a preliminary

conclusion is drawn that in the reduction process of fractions with a maximum number of five digits, it is more appropriate to use the combination of the more phase derogation algorithm and the division algorithm and also take the parameter  $\xi = 8$ .



**Figure 14.** Four-digit parameter comparison line chart.

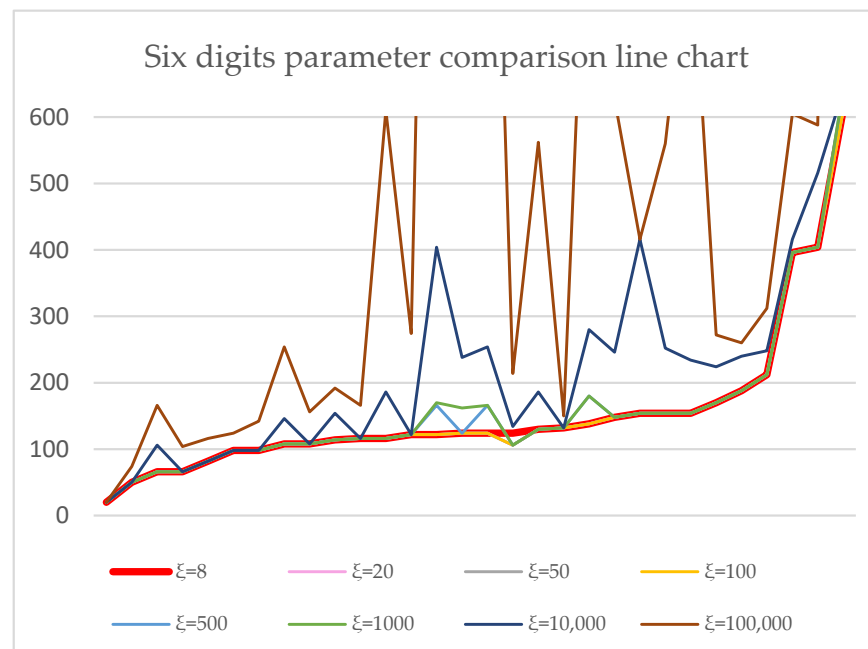


**Figure 15.** Five-digit parameter comparison line chart.

### 5.2.3. Six-Digit Experimental Result

For thirty randomly selected sets of data, each set of data will have eight parameters substituted for the experiment, and the experimental results will be tabulated for statistical analysis.

According to the discount Figure 16 made based on the experimental results, it can be seen that out of the 30 randomly selected sets of data, 29 required the least number of steps for the experiment when the parameter  $\xi = 8$ , and only one set of data had the least number of steps when  $\xi = 20$ , while the experimental steps for each set of data reached the highest value when  $\xi = 100,000$ . Therefore, a preliminary conclusion is drawn that in the reduction process of fractions with a maximum of six digits, it is more appropriate to use the combination of the more phase derogation algorithm and the division algorithm and also take the parameter  $\xi = 8$ .



**Figure 16.** Six-digit parameter comparison line chart.

#### 5.2.4. Summary of Parameter Experiments

Based on the comparison results of the above experiments, we can draw a specific conclusion: for the vast majority of data sets, when the parameter  $\xi$  value is 8, the steps required for the UPS experiment are the least, while the larger the parameter  $\xi$  value, the more steps required for the UPS experiment. When the  $\xi$  value exceeds a certain range, the steps required for the experiment do not change.

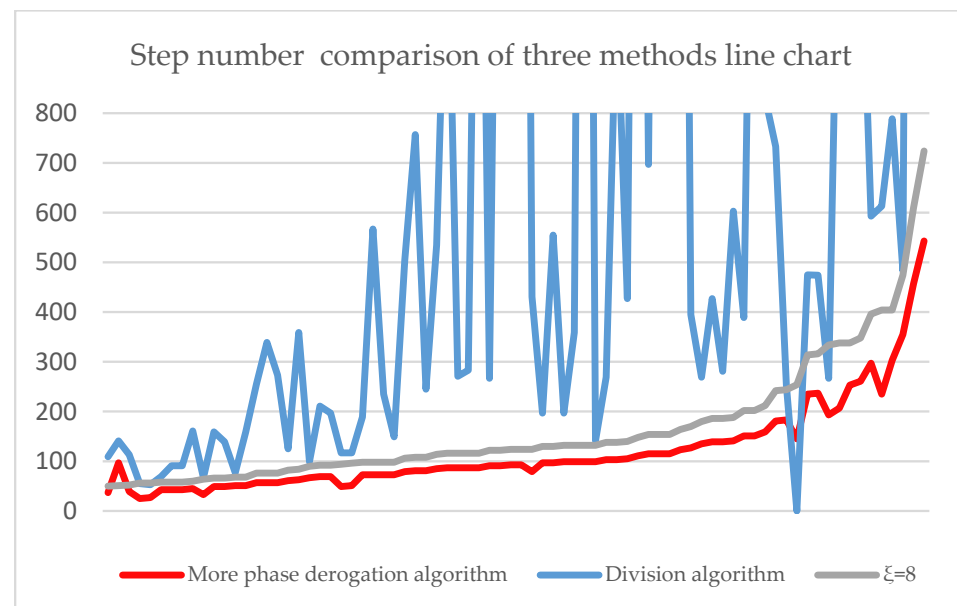
Based on the properties and functions of parameter  $\xi$ , the smaller the value of parameter  $\xi$ , the longer it takes for the system to transition from more phase derogation to division. That is to say, when conducting experiments on the same set of data, the larger the proportion of execution time that is reduced, the higher the efficiency of the system. From the perspective of limit, when  $\xi$  gets smaller and smaller until it approaches zero, it means that only when the difference between the larger value and the smaller value is less than zero can it be converted to the Euclidean algorithm. For the positive integer fractional reduction membrane system designed in this article without considering symbols, there is no case where the difference is less than zero. Therefore, in the combination of the two methods, only the first part of the more phase derogation method is used, and only the rules in membrane  $M_1$  and class A membranes are executed. There is no need to use class B membranes for rolling and dividing to obtain the results. However, when  $\xi$  becomes larger and larger and the difference between the numerator and denominator is even larger, only membrane  $A_1$  of type A films normally executes the subtraction rule. After judging the difference and the size of  $\xi$ , directly use type B films to use the Euclidean algorithm method. At this time, the initial value of membrane  $B_1$  is the initial value of the numerator and denominator, which means that in this case, only the second part of the Euclidean algorithm is performed.

Through the above analysis on the limit, we have a new guess: when comparing the more phase derogation method, the Euclidean algorithm method, and the combination of the two methods, is the number of time slices for the more phase derogation the least?

#### 5.3. Comparative Experiment of Three Methods

We will substitute all 100 sets of data into the combination of the more phase derogation algorithm, the division algorithm, and the optimal parameter  $\xi = 8$  obtained from the experiment in Section 5.2 for experiments. The experimental results will be tabulated for statistical analysis.

According to the discount Figure 17 made based on the experimental results, it can be seen that out of the 100 randomly selected sets of data, 99 sets of data required the least number of steps when using the more phase subtraction method for the experiment, and only one set of data consumed the least number of steps when using the division algorithm. Ninety-nine percent of the data steps reached the highest value when using the division algorithm for the experiment. The number of steps consumed by  $\xi = 8$  is mostly in the middle of the steps consumed by the more phase derogation algorithm and the division algorithm, and in a few cases, it overlaps with the division algorithm.



**Figure 17.** Step number comparison of three methods in a line chart.

Analyzing the case of  $\xi = 8$  in conjunction with Section 5.2.4: In the method of combining the two, regardless of the data values of the numerator and denominator, when the difference is less than eight, the P System needs to convert from a more phase derogation algorithm to a division algorithm for calculation. However, the implementation rules of a class A membrane in the first part are actually consistent with those of a class A membrane in the more phase derogation algorithm P System. If the maximum common divisor of the numerator and denominator is not less than eight, the final result can be obtained directly by the more phase derogation algorithm without passing out the intermediate value in the innermost membrane. At this point, the steps of combining the two in the P System are the same as the steps of reducing the loss of the P System, so it is necessary to focus on the impact of other rules on the time slices. However, the rule that multiset in all membranes realizes one-way transmission between membranes is only executed once, which has little impact on the total steps in the whole System. Therefore, the main reason for the increase in time slices is the rolling division.

According to the execution process of the Section 4.2.2 division membrane rules, it can be seen that our designed film rule actually converts division into subtraction: the process of dividing a larger value by a smaller value to obtain a remainder is converted into the process of subtracting a larger value by an integer multiple of the smaller value. When the P System determines that subtraction cannot continue, the remaining value is the remainder. Repeat this step until the maximum common divisor is found when there is no remainder. Therefore, only one subtraction is needed for each layer of class A membrane in the more phase derogation P System, and then the object is directly transferred to the submembrane, while the division P System is to subtract an integer multiple in each layer until the rules of this layer cannot be executed, then the Multiset can be transferred to the inner membrane, and then the same rules are executed in the inner membrane until the

maximum common divisor is found. For this reason, in theoretical applications, every time a division is performed, it actually requires multiple subtractions in the P System, resulting in a much higher cumulative number of subtractions compared to using the more phase derogation method alone.

Therefore, the final experimental conclusion is drawn: in the process of fractional reduction, the efficiency is higher when using a more phase derogation algorithm P System. Or, to put it another way, it is also correct: in a combined P system, when the parameter  $\xi$  is set to 0, the efficiency of the system is the highest.

## 6. Conclusions

Membrane computing is a computational model abstracted from the functions and structures of living biological cells and the tissues or organs made up of cells and is a branch of biological computing. It is distributed, parallel and non-deterministic in nature. Membrane computing has developed very rapidly, and its methods have been applied to fields such as theoretical computing, image processing and bioinformatics. In this paper, we study the fraction simplification problem in numerical computation, improve the traditional fraction simplification method from the perspective of computational parallelism, and design three fractional reduction P System  $\Pi_1$ ,  $\Pi_2$  and  $\Pi_3$ . Among them,  $\Pi_1$  is a P System design based on improved parallelization of the more phase derogation algorithm;  $\Pi_2$  is a P System design based on improved parallelization of the division algorithm, and  $\Pi_3$  is a P System designed by combining  $\Pi_1$  and  $\Pi_2$  Systems. The feasibility and effectiveness of these P System evolution rules are verified by experiments on the simulation software UPSimulator. From the comparison experiments of the three P Systems, it is clear that the P System  $\Pi_1$  based on the more phase derogation algorithm has higher computational efficiency.

From another point of view, both the more phase derogation P System and the P System combining the two methods are more efficient than the Euclidean P System in finding the greatest common divisor. However, the rolling division is widely used. This algorithm can be used to obtain the least common multiple of two numbers and the maximum common subsequence, and it even has some connection with the RSA algorithm in Cryptography. In future research, we can also optimize other problems that can be solved by the Euclid algorithm so that the P System can achieve higher efficiency.

Fractional reduction is a frequently used calculation in numerical computation, and improving its computational efficiency is one of the most effective ways to improve numerical computation efficiency. Designing P systems with more efficient fractional reduction methods and fewer evolutionary rules is still worth further research.

**Author Contributions:** Conceptualization, H.N. and J.Z.; methodology, Y.K.; validation, Y.K., J.Z. and L.B.; formal analysis, M.Z.; investigation, H.N.; resources, M.Z.; data curation, Y.K.; writing—original draft preparation, J.Z.; writing—review and editing, Y.K. and J.Z.; visualization, J.Z.; supervision, H.N.; project administration, L.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by the Science and Technology Research Program of Chongqing Municipal Education Commission (Grant No. KJQN201901133).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available in insert article.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A

**Table A1.** The complete process of simplifying 15/40 using the combined P system  $\Pi_3$  (complete data for Table 3).

Time Slice	Rules for Implementation	Results of the Implementation
Initial Status	None	$M_1: a^{15}, b^{40}, c, t;$ $A_1: c, d^{20}, \text{CreateSubMem};$ $B_1: c, \text{CreateSubMem};$
1	$M_1: r_1; A_1: r_2$	$M_1: b^{25}, c, t, g^{15};$ $A_1: d^{20}, e, j, k, \text{CreateSubMem};$ $B_1: c, \text{CreateSubMem};$
2	$M_1: r_3, r_4;$	$M_1: c, t, b_2^{25};$ $A_1: a^{40}, b^{15}, d^{20}, e, \text{Num}^{40}, j, k, \text{CreateSubMem};$ $B_1: a^{40}, b^{15}, c, \text{Num}^{40}, \text{CreateSubMem};$
3	$A_1: r_3, r_1;$ $B_1: r_{14}, r_{13};$	$M_1: c, t, b_2^{25};$ $A_1: d^{20}, e, \text{Num}^{40}, u^{25}, x^{15}, y^{15}, i^{25}, j, k, \text{CreateSubMem}, a_1^{25};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$
4	$A_1: r_4;$	$M_1: c, t, b_2^{25};$ $A_1: e, \text{Num}^{40}, u^{25}, g^{20}, x_{15}^{15}, y^{15}, i^{25}, j, k, \text{CreateSubMem}, a_1^{25};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$
5	$A_1: r_5;$	$M_1: c, t, b_2^{25};$ $A_1: e, \text{Num}^{40}, u^{25}, f^5, g^{15}, x^{15}, y^{15}, j, k, \text{CreateSubMem}, a_1^{25};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$
6	$A_1: r_6;$	$M_1: c, t, b_2^{25};$ $A_1: e, \text{Num}^{40}, u^{10}, f^5, g^{15}, x^{15}, j, \text{lan}^{15}, k, \text{CreateSubMem}, a_1^{25};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$
7	$A_1: r_{13}, r_7, r_{10};$	$M_1: c, t, b_2^{25};$ $A_1: \text{Num}^{39}, u^9, f^5, g^{15}, x^{15}, h, j, \lambda^{15}, a_1^{25};$ $A_2: c, d^{20}, \text{CreateSubMem};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$
8	$A_1: r_8, r_{14}, r_{15};$ $A_2: r_2;$	$M_1: c, t, b_2^{25};$ $A_1: u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: a^{25}, b^{15}, d^{20}, \text{Num}^{39}, e, j, k, \text{CreateSubMem};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$
9	$A_2: r_1, r_3;$	$M_1: c, t, b_2^{25};$ $A_1: u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{20}, \text{Num}^{39}, e, u^{10}, x^{15}, y^{15}, i^{10}, j, k, \text{CreateSubMem}, a_1^{10};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$
10	$A_2: r_4;$	$M_1: c, t, b_2^{25};$ $A_1: u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, \text{Num}^{39}, e, u^{10}, g^{10}, x^{15}, y^{15}, j, k, \text{CreateSubMem}, a_1^{10};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$
11	$A_2: r_{16};$	$M_1: c, t, b_2^{25};$ $A_1: u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, \text{Num}^{39}, e, u^{10}, x^{15}, y^{15}, j, k, \text{CreateSubMem}, o^{10}, a_1^{10};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$
12	$A_2: r_{17}, r_{18};$	$M_1: c, t, b_2^{25};$ $A_1: u^9, f^5, g^{15}, h, j, \lambda^{15}, m^{25}, n^{15};$ $A_2: d^{10}, \text{Num}^{39}, e, u^{10}, y^{15}, j, k, \text{CreateSubMem}, o^{10};$ $B_1: c, \text{Num}^{40}, \lambda^{55}, \text{CreateSubMem};$

Table A1. Cont.

Time Slice	Rules for Implementation	Results of the Implementation
13	$A_1: r_{30}, r_{31};$	$M_1: c, t, m^{25}, n^{15}, b_2^{25};$ $A_1: u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: c, Num^{40}, \lambda^{55}, CreateSubMem;$
14	$M_1: r_6, r_5;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: c, n_1^{15}, m_1^{25}, \lambda^{55}, Num^{40}, CreateSubMem;$
15	$B_1: r_1, r_{15};$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: c, m_1^{10}, x^{15}, Num^{39}, \lambda^{55};$ $B_2: CreateSubMem;$
16	$B_1: r_{16}, r_2;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: m_1^{10}, x^{15}, \lambda^{55}, y;$ $B_2: Num^{39}, CreateSubMem;$
17	$B_1: r_3;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: d, m_1^{10}, x^{15}, \lambda^{55};$ $B_2: Num^{39}, CreateSubMem;$
18	$B_1: r_5, r_4;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: n_1^{15}, c, m_1^{10}, \lambda^{55};$ $B_2: Num^{39}, CreateSubMem;$
19	$B_1: r_1;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: n_1^5, c, x^{10}, \lambda^{55};$ $B_2: Num^{39}, CreateSubMem;$
20	$B_1: r_2;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: n_1^5, x^{10}, y, \lambda^{55};$ $B_2: Num^{39}, CreateSubMem;$
21	$B_1: r_6;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: n_1^5, x^{10}, e, g, \lambda^{55};$ $B_2: Num^{39}, CreateSubMem;$
22	$B_1: r_7;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: n_1^5, x^{10}, e, h, \lambda^{55};$ $B_2: Num^{39}, CreateSubMem;$
23	$B_1: r_8, r_9, r_{10};$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, \lambda^{55};$ $B_2: n_1^{10}, c, m_1^5, Num^{39}, CreateSubMem;$



Table A1. Cont.

Time Slice	Rules for Implementation	Results of the Implementation
24	$B_2: r_1, r_{15};$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, \lambda^{55};$ $B_2: n_1^5, c, x^5, Num^{38};$ $B_3: CreateSubMem;$
25	$B_2: r_2, r_{16};$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, \lambda^{55};$ $B_2: n_1^5, x^5, y;$ $B_3: Num^{38}, CreateSubMem;$
26	$B_2: r_6;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, \lambda^{55};$ $B_2: n_1^5, x^5, e, g;$ $B_3: Num^{38}, CreateSubMem;$
27	$B_2: r_7;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, \lambda^{55};$ $B_2: n_1^5, x^5, e, h;$ $B_3: Num^{38}, CreateSubMem;$
28	$B_2: r_8, r_9, r_{10};$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, \lambda^{55};$ $B_2: e;$ $B_3: n_1^5, m_1^5, c, Num^{38}, CreateSubMem;$
29	$B_3: r_1, r_{15};$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, \lambda^{55}; B_2: e;$ $B_3: c, Num^{37}, x^5; B_4: CreateSubMem;$
30	$B_3: r_{16}, r_2;$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, \lambda^{55}; B_2: e; B_3: y, x^5;$ $B_4: Num^{37}, CreateSubMem;$
31	$B_3: r_{11};$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, \lambda^{55}; B_2: e;$ $B_3: z, x^5;$ $B_4: Num^{37}, CreateSubMem;$
32	$B_3: r_{12};$	$M_1: c, t, b_1^{15}, b_2^{25}, a_1^{25};$ $A_1: n_1^{15}, m_1^{25}, u^9, f^5, g^{15}, h, j, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, \lambda^{55};$ $B_2: e, x^5, z^5;$ $B_3: z; B_4: Num^{37}, CreateSubMem;$

Table A1. Cont.

Time Slice	Rules for Implementation	Results of the Implementation
33	B <sub>2</sub> : r <sub>12</sub> ;	M <sub>1</sub> : c, t, b <sub>1</sub> <sup>15</sup> , b <sub>2</sub> <sup>25</sup> , a <sub>1</sub> <sup>25</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , u <sup>9</sup> , f <sup>5</sup> , g <sup>15</sup> , h, j, λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : e, λ <sup>55</sup> , x <sup>5</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
34	B <sub>1</sub> : r <sub>12</sub> ;	M <sub>1</sub> : c, t, b <sub>1</sub> <sup>15</sup> , b <sub>2</sub> <sup>25</sup> , a <sub>1</sub> <sup>25</sup> , x <sup>5</sup> , z <sup>5</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , u <sup>9</sup> , f <sup>5</sup> , g <sup>15</sup> , h, j, λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : e, λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
35	M <sub>1</sub> : r <sub>7</sub> , r <sub>13</sub> ;	M <sub>1</sub> : c, t, f <sup>5</sup> , i <sup>5</sup> , b <sub>1</sub> <sup>10</sup> , b <sub>2</sub> <sup>25</sup> , a <sub>1</sub> <sup>20</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , u <sup>9</sup> , f <sup>5</sup> , g <sup>15</sup> , h, j, λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : e, λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
36	M <sub>1</sub> : r <sub>8</sub> , r <sub>14</sub> ;	M <sub>1</sub> : e, f <sup>5</sup> , i <sup>5</sup> , b <sub>1</sub> <sup>10</sup> , b <sub>2</sub> <sup>25</sup> , a <sub>1</sub> <sup>20</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , u <sup>9</sup> , f <sup>5</sup> , g <sup>15</sup> , h, j, λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : e, λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
37	M <sub>1</sub> : r <sub>9</sub> , r <sub>15</sub> ;	M <sub>1</sub> : d, f <sup>5</sup> , h, i <sup>5</sup> , b <sub>1</sub> <sup>10</sup> , b <sub>2</sub> <sup>25</sup> , a <sub>1</sub> <sup>20</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , u <sup>9</sup> , f <sup>5</sup> , g <sup>15</sup> , h, j, λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : e, λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
38	M <sub>1</sub> : r <sub>10</sub> , r <sub>11</sub> , r <sub>16</sub> , r <sub>17</sub> ;	M <sub>1</sub> : c, t, z <sup>5</sup> , x <sup>5</sup> , k, l, b <sub>1</sub> <sup>10</sup> , b <sub>2</sub> <sup>25</sup> , a <sub>1</sub> <sup>20</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , u <sup>9</sup> , f <sup>5</sup> , g <sup>15</sup> , h, j, λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : e, λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
39	M <sub>1</sub> : r <sub>7</sub> , r <sub>13</sub> , r <sub>20</sub> , r <sub>21</sub>	M <sub>1</sub> : c, t, f <sup>5</sup> , i <sup>5</sup> , k <sub>1</sub> , l <sub>1</sub> , b <sub>1</sub> <sup>5</sup> , b <sub>2</sub> <sup>25</sup> , a <sub>1</sub> <sup>15</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , u <sup>9</sup> , f <sup>5</sup> , g <sup>15</sup> , k <sub>1</sub> , l <sub>1</sub> , h, j, λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : k <sub>1</sub> , l <sub>1</sub> , e, λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
40	A <sub>1</sub> : r <sub>19</sub> , r <sub>20</sub> ; M <sub>1</sub> : r <sub>8</sub> , r <sub>14</sub> ;	M <sub>1</sub> : e, f <sup>5</sup> , i <sup>5</sup> , k <sub>1</sub> , l <sub>1</sub> , b <sub>1</sub> <sup>5</sup> , b <sub>2</sub> <sup>25</sup> , a <sub>1</sub> <sup>15</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , u <sup>9</sup> , f <sup>5</sup> , g <sup>15</sup> , h, j, λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , k <sub>1</sub> , l <sub>1</sub> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : k <sub>1</sub> , l <sub>1</sub> , e, λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
41	A <sub>2</sub> : r <sub>21</sub> , r <sub>22</sub> ; M <sub>1</sub> : r <sub>9</sub> , r <sub>15</sub> ;	M <sub>1</sub> : d, f <sup>5</sup> , i <sup>5</sup> , k <sub>1</sub> , l <sub>1</sub> , b <sub>1</sub> <sup>5</sup> , b <sub>2</sub> <sup>25</sup> , a <sub>1</sub> <sup>15</sup> ; A <sub>1</sub> : n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , u <sup>9</sup> , f <sup>5</sup> , g <sup>15</sup> , h, j, λ <sup>15</sup> ; A <sub>2</sub> : q, p, d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : k <sub>1</sub> , l <sub>1</sub> , e, λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;
42	A <sub>2</sub> : r <sub>24</sub> , r <sub>23</sub> ; M <sub>1</sub> : r <sub>10</sub> , r <sub>11</sub> , r <sub>16</sub> , r <sub>17</sub> ;	M <sub>1</sub> : c, t, k <sub>1</sub> , l <sub>1</sub> , x <sup>5</sup> , z <sup>5</sup> , k, l, b <sub>1</sub> <sup>5</sup> , b <sub>2</sub> <sup>25</sup> , a <sub>1</sub> <sup>15</sup> ; A <sub>1</sub> : r, n <sub>1</sub> <sup>15</sup> , m <sub>1</sub> <sup>25</sup> , u <sup>9</sup> , f <sup>5</sup> , g <sup>15</sup> , w, h, j, λ <sup>15</sup> ; A <sub>2</sub> : d <sup>10</sup> , Num <sup>39</sup> , e, u <sup>10</sup> , y <sup>15</sup> , j, k, CreateSubMem, o <sup>10</sup> ; B <sub>1</sub> : k <sub>1</sub> , l <sub>1</sub> , e, λ <sup>55</sup> , z <sup>5</sup> ; B <sub>2</sub> : e, z <sup>5</sup> ; B <sub>3</sub> : z; B <sub>4</sub> : Num <sup>37</sup> , CreateSubMem;

Table A1. Cont.

Time Slice	Rules for Implementation	Results of the Implementation
43	$M_1: r_7, r_{13}, r_{20}, r_{21};$ $A_1: r_{28}, r_{29};$	$M_1: c, t, k_1^2, l_1^2, f^5, i^5, b_2^{25}, a_1^{10};$ $A_1: n_1^{15}, m_1^{25}, l_1, k_1, f^5, g^{15}, h, j, p^2, q, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: k_1^2, l_1^2, e, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
44	$M_1: r_8, r_{14};$ $A_1: r_{19}, r_{20}, r_{23}, r_{24};$	$M_1: r^2, k_1^2, e, l_1^2, f^5, w, i^5, y, b_2^{25}, a_1^{10};$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, l_1, k_1, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: k_1^2, l_1^2, e, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
45	$M_1: r_9, r_{18}, r_{22}, r_{23};$ $A_2: r_{21}, r_{22};$	$M_1: d, l_1^2, k_1^2, f^5, i^5, \lambda^2, l, r_1, b_2^{25}, a_1^{10};$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, u^9, \lambda^{15};$ $A_2: q, d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10}, p;$ $B_1: k_1^2, l_1^2, e, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
46	$M_1: r_{10}, r_{11}, r_{21}, r_{24};$ $A_2: r_{23}, r_{24};$	$M_1: c, l_1^3, k_1^2, f^5, x^5, w_1^2, k, r_1, b_2^{25}, a_1^{10};$ $A_1: n_1^{15}, m_1^{25}, l_1, f^5, g^{15}, h, j, r, u^9, w, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: k_1^2, l_1^3, e, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
47	$M_1: r_7, r_{20};$ $A_1: r_{20}, r_{28}, r_{29};$	$M_1: c, l_1^3, k_1^3, f^5, i^5, w_1^2, r_1, b_2^{25}, a_1^5;$ $A_1: n_1^{15}, m_1^{25}, f^5, k_1, g^{15}, h, j, p^2, q, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, l_1, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, l_1^3, k_1^3, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
48	$M_1: r_8;$ $A_1: r_{19}, r_{23}, r_{24};$ $A_2: r_{22};$	$M_1: r^2, l_1^3, k_1^3, f^5, w, i^5, y, w_1^2, r_1, b_2^{25}, a_1^5;$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, u^9, \lambda^{15};$ $A_2: q, k_1, d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, l_1^3, k_1^3, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
49	$M_1: r_9, r_{22}, r_{23};$ $A_2: r_{21}, r_{24};$	$M_1: d, l_1^3, k_1^3, f^5, i^5, w_1^2, l, r_1^2, b_2^{25}, a_1^5;$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, u^9, w, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, l_1^3, k_1^3, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
50	$M_1: r_{10}, r_{11}, r_{24};$ $A_1: r_{28};$ $A_2: r_{23};$	$M_1: c, l_1^3, k_1^3, f^5, x^5, w_1^4, k, r_1^2, b_2^{25}, a_1^5;$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, p, q, r, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10}, p;$ $B_1: e, l_1^3, k_1^3, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
51	$M_1: r_7, r_{20};$ $A_1: r_{23}, r_{24}, r_{29};$	$M_1: r, c, l_1^3, k_1^4, f^5, w, i^5, w_1^4, r_1^2, b_2^{25};$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, k_1, h, j, p, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10}, p;$ $B_1: e, l_1^3, k_1^4, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
52	$M_1: r_8, r_{22}, r_{23};$ $A_1: r_{19}, r_{23};$	$M_1: r, l_1^3, k_1^4, f^5, i^5, y, w_1^4, \lambda, r_1^3, b_2^{25};$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, k_1, j, k, CreateSubMem, o^{10};$ $B_1: e, l_1^3, k_1^4, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$

Table A1. Cont.

Time Slice	Rules for Implementation	Results of the Implementation
53	$M_1: r_{12}, r_{22}, r_{24};$ $A_2: r_{21};$	$M_1: l_1^3, k_1^4, f^5, i^5, w_1^5, \lambda, k, r_1^3, b_2^{25};$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10}, p;$ $B_1: e, l_1^3, k_1^4, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
54	$M_1: r_{20}, r_{24};$ $A_2: r_{23};$	$M_1: l_1^3, k_1^5, f^5, i^5, w_1^6, r_1^3, b_2^{25};$ $A_1: n_1^{15}, m_1^{25}, f^5, k_1, g^{15}, h, j, r, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, l_1^3, k_1^5, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
55	$A_1: r_{19}, r_{29};$	$M_1: l_1^3, k_1^5, f^5, i^5, w_1^6, r_1^3, b_2^{25};$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, p, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, k_1, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, l_1^3, k_1^5, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
56	$A_1: r_{23};$ $A_2: r_{21};$	$M_1: r, l_1^3, k_1^5, f^5, i^5, w_1^6, r_1^3, b_2^{25};$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10}, p;$ $B_1: e, l_1^3, k_1^5, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
57	$M_1: r_{22};$ $A_2: r_{23};$	$M_1: l_1^3, k_1^5, f^5, i^5, w_1^6, r_1^3, b_2^{25}, \lambda;$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, r, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, l_1^3, k_1^5, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
58	$M_1: r_{24};$ $A_1: r_{29};$	$M_1: l_1^3, k_1^5, f^5, i^5, w_1^7, r_1^3, b_2^{25};$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, p, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, l_1^3, k_1^5, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
59	$A_1: r_{23};$	$M_1: r, l_1^3, k_1^5, f^5, i^5, w_1^7, r_1^3, b_2^{25};$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, l_1^3, k_1^5, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
60	$M_1: r_{22}$	$M_1: l_1^3, k_1^5, f^5, i^5, w_1^7, r_1^3, b_2^{25}, \lambda;$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, l_1^3, k_1^5, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
61	$M_1: r_{24}$	$M_1: l_1^3, k_1^5, f^5, i^5, w_1^8, r_1^3, b_2^{25};$ $A_1: n_1^{15}, m_1^{25}, f^5, g^{15}, h, j, u^9, \lambda^{15};$ $A_2: d^{10}, Num^{39}, e, u^{10}, y^{15}, j, k, CreateSubMem, o^{10};$ $B_1: e, l_1^3, k_1^5, \lambda^{55}, z^5;$ $B_2: e, z^5; B_3: z; B_4: Num^{37}, CreateSubMem;$
62	No enforceable rules	

## References

- de Castro, N.L. Fundamentals of natural computing: An overview. *Phys. Life Rev.* **2007**, *4*, 1–36. [[CrossRef](#)]
- Păun, G.; Mario, J.P.-J. Membrane computing: Brief introduction, recent results and applications. *Biosystems* **2006**, *85*, 11–22. [[CrossRef](#)] [[PubMed](#)]
- Păun, G. A quick introduction to membrane computing. *J. Log. Algebraic Program.* **2010**, *79*, 291–294. [[CrossRef](#)]

4. Pérez Hurtado de Mendoza, I.; Orellana Martín, D.; Martínez del Amor, M.Á.; Valencia Cabrera, L. A Membrane Computing Framework for Social Navigation in Robotics. *Comput. Electr. Eng.* **2021**, *95*, 107408. [CrossRef]
5. Mohan, B.S.; Mahmood, A.A.; Mohammed, M.Q.; Zaki, N.D. *Replicating the MAP Kinase Cascade in Membrane Computing*; IOP Publishing: Bristol, UK, 2021; Volume 1963, p. 012156.
6. Frisco, P.; Gheorghe, M.; Pérez-Jiménez, M.J. *Applications of Membrane Computing in Systems and Synthetic Biology*; Springer: Berlin/Heidelberg, Germany, 2014; ISBN 3-319-03191-0.
7. Ciobanu, G.; Păun, G.; Pérez-Jiménez, M.J. *Applications of Membrane Computing*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 17.
8. Atanasiu, A.; Carlos, M. Arithmetic with membranes. In Proceedings of the Workshop on Multiset Processing, Dubrovnik, Croatia, 29 June 2000; pp. 1–17.
9. Ciobanu, G. A Programming perspective of the membrane systems. *Int. J. Comput. Commun. Control* **2006**, *1*, 13–24. [CrossRef]
10. Guo, P.; Jing, C. Arithmetic operation in membrane system. In Proceedings of the 2008 International Conference on BioMedical Engineering and Informatics, Sanya, China, 27–30 May 2008; Volume 1, pp. 231–234.
11. Guo, P.; Zhang, H. Arithmetic operation in single membrane. In Proceedings of the 2008 International Conference on Computer Science and Software Engineering, Wuhan, China, 12–14 December 2008; Volume 3, pp. 532–535.
12. Guo, P.; Luo, M. Signed numbers arithmetic operation in multi-membrane. In Proceedings of the 2009 First International Conference on Information Science and Engineering, Nanjing, China, 26–28 December 2009; pp. 393–396.
13. Guo, P.; Liu, S.J. Arithmetic expression evaluation in membrane computing with priority. In *Advanced Materials Research*; Trans Tech Publications Ltd.: Stafa-Zurich, Switzerland, 2011; Volume 225, pp. 1115–1119.
14. Guo, P.; Chen, H.; Zheng, H. Arithmetic expression evaluations with membranes. *Chin. J. Electron.* **2014**, *23*, 55–60. Available online: <https://cje.ejournal.org.cn/article/id/8163> (accessed on 16 July 2022).
15. Guo, P.; Chen, H. Arithmetic expression evaluation by P systems. *Appl. Math* **2013**, *7*, 549–553. [CrossRef]
16. Guo, P.; Zheng, H.; Chen, H.; Chen, J. Fraction arithmetic operations performed by P systems. *Chin. J. Electron.* **2013**, *22*, 690–694.
17. Rich, A.D.; Stoutemyer, D.R. Representation. Simplification and Display of Fractional Powers of Rational Numbers in Computer Algebra. *arXiv* **2013**, arXiv:1302.2169.
18. Guo, P.; Zhang, H.; Chen, H.; Liu, R. Fraction reduction in membrane systems. *Sci. World J.* **2014**, *2014*, 858527. [CrossRef] [PubMed]
19. Păun, G. From cells to computers: Computing with membranes (P systems). *Biosystems* **2001**, *59*, 139–158. [CrossRef] [PubMed]
20. Martín-Vide, C.; Gheorghe, P.; Alfonso, R.-P. On P systems with membrane creation. *Comput. Sci. J. Mold.* **2001**, *9*, 26.
21. Backhouse, R.; Joao, F.F. On Euclid's algorithm and elementary number theory. *Sci. Comput. Program.* **2011**, *76*, 160–180. [CrossRef]
22. Rogers, H. The Euclidean Algorithm as a Means of Simplifying Fractions. *Arith. Teach.* **1970**, *17*, 657–662. [CrossRef]
23. Guo, P.; Quan, C.; Ye, L. UPSimulator: A general P system simulator. *Knowl.-Based Syst.* **2019**, *170*, 20–25. [CrossRef]
24. Raghavan, S.; Chandrasekaran, K. Tools and simulators for membrane computing—A literature review. In *Bio-Inspired Computing—Theories and Applications*; Gong, M., Pan, L., Song, T., Zhang, G., Eds.; BIC-TA 2016; Communications in Computer and Information Science; Springer: Singapore, 2016; Volume 681, pp. 249–277. [CrossRef]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.