The P System Design Method based on the P Module

Ping Guo, Xixi Peng, Lian Ye
College of Computer Science
Chongqing University
Chongqing,400044
China

***Abstract*—Membrane computing is a kind of biocomputing model. At present, the main research areas of membrane computing are computational models and P system design. With the expansion of the P system scale, how to rapidly construct the P system has become a prominent issue. Designing P system based on P module is a P system design method proposed in recent years. This method provides information hiding and can build P system through recursive combination. However, the current P module design lacks a unified design method and lacks the standard process of building P system from P module. This paper studies the structural characteristics of cell-like P systems, and proposes an improved P module design method and a process for assembling P systems through P modules. In order to fully expound the design method of P module, the P system for the square root of the large number was analyzed and designed. And the correctness of the P system based on the P module design method was verified by an instance.**

Keywords*—P module; P System; P system design; membrane computing; biocomputing models

I. Introduction

Membrane computing, also known as P system, is a branch of natural computing [1]. The models of P system are mainly divided into three types, namely, the cell-like P system [2], the tissue-like P system [3] and the neural-like P system [4]. They have been applied to solve the problems such as NP problems [5]–[8], image processing [9], [10], arithmetic operations [11]–[14] and so on.

In our previous work, most energy are put to implement the arithmetic operations in cell-like P systems: Ref. [11] firstly proposed an arithmetic P systems to implement the arithmetic operation in 2001; in [12] proposes an algorithm and builds expression P systems without priority rules for evaluating arithmetic expression; in [13] designed the P systems for addition, subtraction and multiplication; in [14] proposes a family of systems for solving Matrix-Vector Multiplication. Although we have obtained many excellent research results in the cell-like P system, the difficulty for constructing the P system has continued to increase due to the increasingly complex algorithm. As a result, some experts and scholars began to propose some new models of modular constructing the P system. In 2009, Romero-Campero et al. proposed a biology model for modular combination cells [15] and Serbanuta et al. proposed K systems embedded in P system which can develop new extensions of P system [16]. In 2010, Păun et al. proposed the dp system [17], which contains ideas for modularity.

Based on the above models, the modularized construction model, the P module [18], is proposed for simplifing the computing system structure and improving the reusability. At present, this model has been applied to some areas of research. In [19] proposed an improved generic version of P modules, an extensible framework for recursive composition of P systems. It proposed an solution to solve Byzantine agreement problem by P module. In [20] presented an improved deterministic solution for Flow-shop Scheduling problem. In [21] extended the P module theoretically and proposes the P module to solve the stereo matching problem in the application. Besides, it realized the discovering neighbors and Echo Algorithm. In [22] studied on the problem which aims to find out a point-disjoint and edge-disjoint path between source point and target point. All of these literature researches are related to the algorithms application of the P module. However, due to the lack of a unified design method, the P system based on the P module have low design efficiency and high error rate. In order to improve such problems, this paper designs a well-structured P module by combining the structural design methods in design methodology. The correctness of the dynamic execution of the P system is ensured with a good structure, making the P system easy to understand, easy to debug, and easy to maintain.

In this paper, the cell-like P system and the P module are introduced in Section 2. Section 3 improves the P module and proposes the design and assembly of the P module. With the method of structural design in design methodology, four methods for constructing P module are proposed to design well-structured P system based on P modules in Section 4. Section 5 gives an instance to show the working mechanism of P module by using the related definitions and design methods of the P module. Section 6 summarizes the research work and presents a deeper level of research in the future.

II. Foundations and Related Works

A. Cell-like P System

The cell-like P system is a class of P system constructed by biochemical reactions in abstract biological cells. In the cell-like P system, the substances in the cells are abstracted as computational objects and the biochemical reactions within the cells are abstracted as object evolutionary rules. A cell-like P system containing five membranes is shown by Fig. 1.

Fig. 1 is a schematic representation of a cell-like P system. A cell-like P system consists of the membranes (elementary membrane and combination membrane), the membrane regions

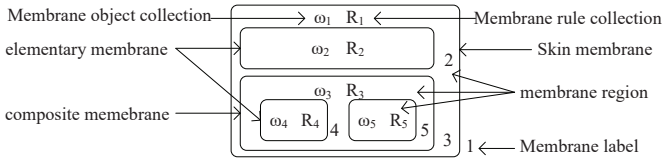

Fig. 1. The structure of cell-like P system.

surrounded by membranes, the membrane object collection in the regions and membrane rule collection.Formally, a cell-like P system (of degree $m \geq 1$) can be defined as form [23]: In the general model, the structure is in the form of nested membrane, which is not easy to be modularized, componentized and expanded. The rules in the cell-like P system can lead to high coupling degree.

$$\Pi = (O, \mu, \omega_1, \cdots, \omega_m, R_1, \cdots, R_m, i_o) \tag{1}$$

In the general model, the structure is in the form of nested membrane, which is not easy to be modularized, componentized and expanded. The rules in the cell-like P system can lead to high coupling degree.

B. The Related Works

The P module is a model that modularizes the biochemical reaction of a group of cells and supports information hiding. Formally, a P module can be defined as form:

$$\Pi = (O, K, \delta, P) \tag{2}$$

1) O is a finite non-empty alphabet of objects;
2) K is a finite set of cells, where each cell, $\delta \in K$, has the form $\delta = (Q, s_0, \omega_0, R)$ where,
   a) Q is a finite set of states;
   b) $s_0 \in Q$ is the initial state;
   c) $\omega_0 \in O^*$ is the initial multiset of objects;
   d) R is a finite ordered set of multiset rewriting rules of the general form:

$$sx \rightarrow_\alpha s'x'(u)_{\beta_\gamma}; \tag{3}$$

   where,
   (i) $s, s' \in Q$;
   (ii) $x, x' \in O^*$,$u \in O^*$;
   (iii) α is a rewriting operator, $\alpha \in \{min, max\}$, The rewriting operator $\alpha = min$ indicates that the rewriting is applied once, if the rule is applicable; and $\alpha = max$ indicates that the rewriting is applied as many times as possible, if the rule is applicable. When $\alpha = max$, α can be omitted in the rule.
   (iv) $\beta \in \{\uparrow, \downarrow, \updownarrow\}$;
   (v) $\gamma \in \{one, spread, repl\}$;
3) δ is a binary relation on K, i.e. a set of parent-child structural arcs, representing $duplex$ or $simplex$ communication channels between cells;
4) P is a subset of K,indicating the $port$ cells, i.e. the only cells can be connected to other modules.

P module is a modular combination model of cells. It mainly uses the characteristics of its recursive combination to realize the hidden functions of internal information and internal structure, so as to facilitate the construction of a complex P system.

III. Design and Assembly of the P Module

This section improves the P module with high encapsulation, information hiding, modular combination and high concurrency. Its special external definition, external reference and assembly mechanism make it highly independent, realize the reuse of modules and speed up the construction of P system.

A. P Module Improvement

The P module is a model of cell-like P system, which abstracts a cell into a P module. It has the characteristics of module encapsulation and the inheritance of rules and objects. Each module is independent and several P modules can be combined into the combination P module by a structured way. Formally, a P module (of degree $m \geq 1$) can be defined as form:

$$\Pi = (O, K, \delta, Q, D_\uparrow, D_\downarrow, R_\uparrow, R_\downarrow) \tag{4}$$

1) O is a finite non-empty alphabet of objects, O=$O_1 \cup O_2$. For each submodule, they contain the public objects from the parent module and their own objects.
   a) O_1 is a subset of O, which represents private objects.
   b) O_2 is a subset of O, disjoint of O_1, which represents public objects.
2) K is a finite set of P modules.
3) δ is a subset of $(K \times K) \cup (K \times R_\downarrow) \cup (R_\uparrow \times K)$, i.e. a set of parent-child structural arcs, representing duplex or simplex communication channels, between two existing P modules or between an existing P modules and an external reference.
4) Q is a subset of O_2, which is the generic synchronizing object set that P modules finally output.
5) $D_\uparrow$ is a subset of K, representing $def_\uparrow$ definitions, e.g. $def_{\uparrow_{\Pi_i}}$ represents that the entrance module of this P module is Π_i; $D_\downarrow$ is a subset of K, representing $def_\downarrow$ definitions, e.g. $def_{\downarrow_{\Pi_i}}$ represents that the export module of this P module is Π_i.
6) $R_\uparrow$ is a finite set, disjoint of K, representing $ref_\uparrow$ references, e.g. $ref_{\uparrow_{a_i}}$ represents that the entrance arc of this P module is a_i; $R_\downarrow$ is a finite set, disjoint of K, representing $ref_\downarrow$ references, e.g. $ref_{\uparrow_{b_i}}$ represents that the export arc of this P module is b_i.
7) Each cell, $\sigma \in K$, has the form $\sigma = (L, S, s_0, \omega_0, R)$. where,
   a) L represents the inheritance rights of the rules; $L = \{\Gamma, \Delta, \Phi\}$; Γ represents this rule as a public rule, Δ represents this rule as a protected rule, Φ represents this rule as a private rule(this can be omitted).
   b) S is a finite set of states;
   c) $s_0 \in S$ is the initial state;
   d) $\omega_0 \in O^*$ is the initial multiset of objects;
   e) R is a finite ordered set of rules:

$$lsx \rightarrow_\alpha s'/x'; (id) \tag{5}$$

   where,
   (i) $l \in L$;

(ii) $s, s' \in S$;
(iii) $x \in O^*$;
(iv) α is a rewriting operator, $\alpha \in \{min, max\}$, The rewriting operator $\alpha = min$ indicates that the rewriting is applied once, if the rule is applicable; and $\alpha = max$ indicates that the rewriting is applied as many times as possible, if the rule is applicable. When $\alpha = max$, α can be omitted in the rule.
(v) id is a number identified the sequence of rule execution. Prior and preference can be given high-ranking. If the rules in the same state are in the same priority, id can be omitted in the rule.

In this model, the connection relationship between P modules is a parent-child relationship, and theirs inheritance can be reflected by objects, O, and rules, R. Through the inheritance of the P module, we can organize system structure more effectively, clarify the relationship between modules, and make full use of existing modules to achieve more complex and deeper development.

B. P Module Assembly Mechanism

According to the definition of P module introduced above, a P system is a P module which is constructed by nested P modules. The nested P module is expressed by the combination P module which is constructed by P modules in the same layer. Given an arbitrary finite set of disjoint P modules, we can construct a combination P module by instantiating some of their external references to some of their external definitions, which implicitly instantiates the relationship of P modules in the same layer. When the parent P module is executed, the submodule will inherit the public objects and public rules of the parent P module to further initialize the internal structure of the module and start to work. The siblings can be executed in parallel, this shows the powerful computing power of the whole system. The combination P module can encapsulate the details of the interior, users only pay attention to their input and output.

Considering of a finite family of n P modules, $\Psi = \{ \Pi_i | i \in [1,n] \}$, where $\Pi_i = (O_i, K_i, \delta_i, Q_i, D_{\uparrow_i}, D_{\downarrow_i}, R_{\uparrow_i}, R_{\downarrow_i})(i \in [1,n])$, the result of a composition P module depends on one kind of actual instantiation that the external reference and the definition are matched. The external reference is matched to external definition by two partial mappings, $\rho_\uparrow : \cup_{i\in[1,n]} R_{\uparrow_i} \rightarrow \cup_{i\in[1,n]} D_{\uparrow_i}$, $\rho_\downarrow : \cup_{i\in[1,n]} R_{\downarrow_i} \rightarrow \cup_{i\in[1,n]} D_{\downarrow_i}$. A previously uninstantiated arc (σ, r), where $(\sigma \in K_i, r \in R\downarrow_i | i \in [1,n])$, is instantiated as $(\sigma, \rho_{\downarrow_{(\sigma_j)}})$, and a previously uninstantiated arc (r, σ), where $(\sigma \in K_i, r \in R_{\uparrow_i} | i \in [1,n])$, is instantiated as $(\rho_{\uparrow_{(\sigma_i)}}, \sigma)$.

Based on what has been described above, the P module family Ψ can be expressed as the form, $\Pi = (O, K, \delta, Q, D_\uparrow, D_\downarrow, R_\uparrow, R_\downarrow)$, when $\rho_\uparrow, \rho_\downarrow$ are the partial mappings that define the instantiation (as previously introduced), if:

1) Ψ is cell-disjoint;
2) $O = \cup_{i\in[1,n]} O_i$;
3) $K = \cup_{i\in[1,n]} K_i$;
4) $\delta = \{ (\tilde{\rho}_{\uparrow_{(\sigma)}}, \tilde{\rho}_{\downarrow_{(\sigma)}}) | \cup_{i\in[1,n]} \sigma_i \}$, where $\tilde{\rho}_{\uparrow_{(\sigma)}} = \sigma \in Dom(\rho_\uparrow) ? \rho_{\uparrow_{(\sigma)}} : \sigma, \tilde{\rho}_{\downarrow_{(\sigma)}} = \sigma \in Dom(\rho_\downarrow) ? \rho_{\downarrow_{(\sigma)}} : \sigma$;
5) $Q = \cup_{i\in[1,n]} Q_i / \cup_{i\in[1,n-1]} Q_i = Q_n$;($Q_n$ is the ouput onjects as the exit of the combination P module)
6) $D_\uparrow \subseteq \cup_{i\in[1,n]} D_{\uparrow_i}, D_\downarrow \subseteq \cup_{i\in[1,n]} D_{\downarrow_i}$;
7) $R_\uparrow = \cup_{i\in[1,n]} R_{\uparrow_i} \setminus Dom(\rho_\uparrow), R_\downarrow = \cup_{i\in[1,n]} R_{\downarrow_i} \setminus Dom(\rho_\downarrow)$;

As described above, we can know the concrete the construction and assembly mechanism of P modules in P system, which includes the nested combination principle of the P module in the same layer and the perfect encapsulation mechanism. The construction and assembly mechanism also make a detailed definition of δ, D, R as a way of communication. The P modules construction and assembly mechanism facilitates the design of P system for complex algorithms, where every P module provides encapsulation and information hiding to other P modules.

IV. Basic Structure of the P System

The P system is constructed by layer upon layer encapsulation using P module. P modules in the same layer are assembled by the construction and assembly mechanism and encapsulated into a combination P module. The construction and assembly methods of a combination P module include four ways, i.e. the sequence method, the branch method, the cycle method and the parallel method, which show the four structures of the P module, respectively.

A. Sequential Method

Since the specific implementation rules of each P module will be determined by the function to be performed, the definition of the rules in each P module need to be abstracted to be a form,which is shown below through the two rules. The general design definition of a elementary P module perform a series of calculations on the initial object set, and finally output the result set. As shown below, here are two rules to represent this process, r_1 represents a series of operations on the initial object set, which are a series of operations except the output of the result set, and the evolution from the initial set of objects x to y is accomplished by multiple rules in the specific implementation. r_2 represents the calculated set of objects is evolved into the set of public objects required by the submodule, so that the submodule can inherit from it to obtain a complete initial set of objects.

r_1: $\Gamma/\Delta/\Phi S_0 x_0, \ldots, x_{i_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_0}$

r_2: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_0} \rightarrow_{min/max} S_0 z_0, \ldots, z_{k_0}$

Fig. 2 illustrates a combined P module through the sequential modular composition of two elementary P modules.

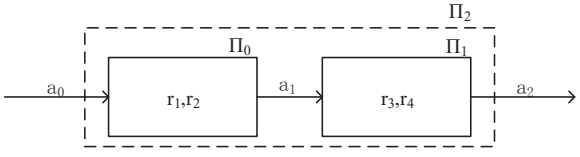

Fig. 2. The sequential structure based on the P module.

In $\Pi_0 < def_{\downarrow_{\Pi_0}}, ref_{\downarrow_{a_1}} >$, using the ruleset following this paragraph objects $\alpha_i (i \in [1, k_0])$ can be obtained by inputting objects, $x_i (i \in [1, i_0])$.

r_1: $\Gamma/\Delta/\Phi\ S_0 x_0, \ldots, x_{i_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_0}$

r_2: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_0} \rightarrow_{min/max} S_0\alpha_0, \ldots, \alpha_{k_0}$

In $\Pi_1 < def_{\downarrow_{\Pi_1}}, ref_{\downarrow_{a_2}} >$following this paragraph, using the ruleset following this paragraph, objects $\beta_i (i \in [1, k_1])$ can be obtained by inputting objects, $\alpha_i (i \in [1, k_0])$.

r_3: $\Gamma/\Delta/\Phi\ S_0\alpha_0, \ldots, \alpha_{k_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_1}$

r_4: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_1} \rightarrow_{min/max} S_0\beta_0, \ldots, \beta_{k_1}$

The combination P module, $\Pi_2 < def_{\downarrow_{\Pi_0}}, ref_{\downarrow_{a_2}} >$, contains two P modules, Π_0 and Π_1, which also appears as an external $def_{\downarrow}$ definition, and makes external $ref_{\downarrow}$ references to a unspecified P module, a_2. There is a definition of Π_2 following this paragraph.

$$\Pi_2 = (O, K, \delta, Q, D_{\uparrow}, D_{\downarrow}, R_{\uparrow}, R_{\downarrow})$$

where,

1) $O = O_1 \cup O_2$

$$\begin{aligned} &= \{\ x_0, \ldots, x_{i_0}, y_0, \ldots, y_{j_0}, x_0, \ldots, x_{i_1}, y_0, \ldots, y_{j_1} \\ &\quad \} \cup \{\ \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_1}\ \} \\ &= \{\ x_0, \ldots, x_{i_0}, y_0, \ldots, y_{j_0}, x_0, \ldots, x_{i_1}, y_0, \ldots, y_{j_1}, \\ &\quad \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_1}\ \} \end{aligned}$$

2) $K = \{\ \Pi_0, \Pi_1\ \}$
3) $\delta = \{\ (\Pi_0, ref_{\downarrow_{a_1}} \rightarrow def_{\downarrow_{\Pi_1}})\ \}$
4) $Q = \{\ \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_1}\ \} / \{\ \alpha_0, \ldots, \alpha_{k_0}\ \} = \{\ \beta_0, \ldots, \beta_{k_1}\ \}$
5) $D_{\downarrow} = \{\ def_{\downarrow_{\Pi_0}}\ \}, D_{\uparrow} = \{\ \}$
6) $R_{\downarrow} = \{\ ref_{\downarrow_{a_2}}\ \}, R_{\uparrow} = \{\ \}$

We can connect Π_0 and Π_1 by the generic instantiation: $(\Pi_0, ref_{\downarrow_{a_1}} \rightarrow def_{\downarrow_{\Pi_1}})$.

B. Branch Method

Fig. 3 illustrates a combined P module through the branch modular composition of four P modules.

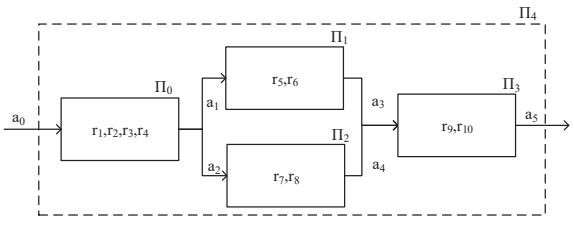

Fig. 3. The branch structure based on the P module.

In $\Pi_0 < def_{\downarrow_{\Pi_0}}, ref_{\downarrow_{a_5}} >$, using the ruleset following this paragraph, there is no material input to determine the object m (m is a set of objects), if there is a material object, get $\alpha_i (i \in [1, k_0])$; otherwise, get $\beta_i (i \in [1, k_0])$.

r_1: $\Gamma/\Delta/\Phi S_0 x_0, \ldots, x_{i_0}, m \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_0}; 1$

r_2: $\Gamma/\Delta/\Phi S_0 x_0, \ldots, x_{i_0}, \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_0}; 2$

r_3: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_0} \rightarrow_{min/max} S_0\alpha_0, \ldots, \alpha_{k_0}$

r_4: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_0} \rightarrow_{min/max} S_0\beta_0, \ldots, \beta_{k_0}$

In $\Pi_1 < def_{\downarrow_{\Pi_1}}, ref_{\downarrow_{a_3}} >$, using the ruleset following this paragraph, objects $\phi_i (i \in [1, k_1])$ can be obtained by inputting objects, $\alpha_i (i \in [1, k_0])$.

r_5: $\Gamma/\Delta/\Phi S_0\alpha_0, \ldots, \alpha_{k_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_1}$

r_6: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_1} \rightarrow_{min/max} S_0\phi_0, \ldots, \phi_{k_1}$

In $\Pi_2 < def_{\downarrow_{\Pi_2}}, ref_{\downarrow_{a_4}} >$, using the ruleset following this paragraph, objects $\phi_i (i \in [1, k_1])$ can be obtained by inputting objects, $\beta_i (i \in [1, k_0])$.

r_7: $\Gamma/\Delta/\Phi S_0\beta_0, \ldots, \beta_{k_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_2}$

r_8: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_2} \rightarrow_{min/max} S_0\phi_0, \ldots, \phi_{k_1}$

In $\Pi_3 < def_{\downarrow_{\Pi_3}}, ref_{\downarrow_{a_5}} >$, using the ruleset following this paragraph, objects $\gamma_i (i \in [1, k_2])$ can be obtained by inputting objects, $\phi_i (i \in [1, k_1])$.

r_7: $\Gamma/\Delta/\Phi S_0\phi_0, \ldots, \phi_{k_1} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_2}$

r_8: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_2} \rightarrow_{min/max} S_0\gamma_0, \ldots, \gamma_{k_2}$

The combined P module, $\Pi_4 < def_{\downarrow_{\Pi_0}}, ref_{\downarrow_{a_5}} >$, contains four P modules, Π_0, Π_1, Π_2 and Π_3, which also appears as an external $def_{\downarrow}$ definition, and makes one external $ref_{\downarrow}$ references to one unspecified P module, a_5. There is a definition of Π_4 following this paragraph.

$$\Pi_4 = (O, K, \delta, Q, D_{\uparrow}, D_{\downarrow}, R_{\uparrow}, R_{\downarrow})$$

where,

1) $O = O_1 \cup O_2$

$$\begin{aligned} &= \{\ x_0, \ldots, x_{i_0}, y_0, \ldots, y_{j_0}, m, x_0, \ldots, x_{i_1}, y_0, \\ &\quad \ldots, y_{j_1}\ \} \cup \{\ \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_0}, \phi_0, \ldots, \\ &\quad \phi_{k_1}, \gamma_0, \ldots, \gamma_{k_2}\ \} \\ &= \{\ x_0, \ldots, x_{i_0}, y_0, \ldots, y_{j_0}, m, x_0, \ldots, x_{i_1}, y_0, \\ &\quad \ldots, y_{j_1}, \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_0}, \phi_0, \ldots, \phi_{k_1}, \\ &\quad \gamma_0, \ldots, \gamma_{k_2}\ \} \end{aligned}$$

2) $K = \{\ \Pi_0, \Pi_1, \Pi_2, \Pi_3\ \}$
3) $\delta = \{\ (\Pi_0, ref_{\downarrow_{a_1}} \rightarrow def_{\downarrow_{\Pi_1}}), (\Pi_0, ref_{\downarrow_{a_2}} \rightarrow def_{\downarrow_{\Pi_2}}), (\Pi_1, ref_{\downarrow_{a_3}} \rightarrow def_{\downarrow_{\Pi_3}}), (\Pi_2, ref_{\downarrow_{a_4}} \rightarrow def_{\downarrow_{\Pi_3}})\ \}$
4) $Q = \{\ \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_0}, \phi_0, \ldots, \phi_{k_1}, \gamma_0, \ldots, \gamma_{k_2}\ \} / \{\ \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_0}, \phi_0, \ldots, \phi_{k_1}\ \} = \{\ \gamma_0, \ldots, \gamma_{k_2}\ \}$
5) $D_{\downarrow} = \{\ def_{\downarrow_{\Pi_0}}\ \}, D_{\uparrow} = \{\ \}$
6) $R_{\downarrow} = \{\ ref_{\downarrow_{a_5}}\ \}, R_{\uparrow} = \{\ \}$

We can connect Π_0 and Π_1 by the generic instantiation: $(\Pi_0, ref_{\downarrow_{a_1}} \rightarrow def_{\downarrow_{\Pi_1}})$, Π_0 and Π_2 by the generic instantiation: $(\Pi_0, ref_{\downarrow_{a_2}} \rightarrow def_{\downarrow_{\Pi_2}})$, Π_1 and Π_3 by the generic instantiation:$(\Pi_1, ref_{\downarrow_{a_3}} \rightarrow def_{\downarrow_{\Pi_3}})$ and connect Π_2 and Π_3 by the generic instantiation: $(\Pi_2, ref_{\downarrow_{a_4}} \rightarrow def_{\downarrow_{\Pi_3}})$.

C. Cycle Method

Fig. 4 illustrates a combined P module through the cycle modular composition of two P modules which construct do-while model.

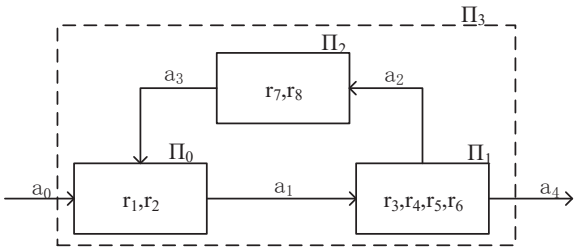

Fig. 4. The cycle structure based on the P module.

In $\Pi_0 < def_{\downarrow_{\Pi_0}}, ref_{\downarrow_{a_1}} >$,using the ruleset following this paragraph, objects $\alpha_i (i \in [1, k_0])$ can be obtained by inputting objects, $x_i (i \in [1, i_0])$.

r_1: $\Gamma/\Delta/\Phi S_0 x_0, \ldots, x_{i_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_0}$

r_2: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_1} \rightarrow_{min/max} S_0 \alpha_0, \ldots, \alpha_{k_0}$

In $\Pi_1 < def_{\downarrow_{\Pi_1}}, ref_{\downarrow_{a_2}}, ref_{\downarrow_{a_4}} >$, there is no material input to determine the object m (m is a set of objects), if there is a material object, get $\beta_i (i \in [1, k_1])$; otherwise, get $\phi_i (i \in [1, k_0])$.

r_3: $\Gamma/\Delta/\Phi S_0 \alpha_0, \ldots, \alpha_{k_0}, m \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_1}; 1$

r_4: $\Gamma/\Delta/\Phi S_0 \alpha_0, \ldots, \alpha_{k_0}, \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_1}; 2$

r_5: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_1} \rightarrow_{min/max} S_0 \beta_0, \ldots, \beta_{k_1}$

r_6: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_1} \rightarrow_{min/max} S_0 \phi_0, \ldots, \phi_{k_1}$

In $\Pi_2 < def_{\downarrow_{\Pi_2}}, ref_{\downarrow_{a_3}} >$, using the ruleset following this paragraph, objects $\gamma_i (i \in [1, k_2])$ can be obtained by inputting objects,$\beta_i (i \in [1, k_1])$.

r_7: $\Gamma/\Delta/\Phi S_0 \beta_0, \ldots, \beta_{k_1} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_2}$

r_8: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_2} \rightarrow_{min/max} S_0 \gamma_0, \ldots, \gamma_{k_2}$

The combined P module, $\Pi_3 < def_{\downarrow_{\Pi_0}}, ref_{\downarrow_{a_4}} >$, contains three P modules, Π_0, Π_1 and Π_2, which also appears as an external $def_{\downarrow}$ definition, and makes two external $ref_{\downarrow}$ references to one unspecified P modules, a_4. There is a definition of Π_3 following this paragraph.

$$\Pi_3 = (O, K, \delta, Q, D_{\uparrow}, D_{\downarrow}, R_{\uparrow}, R_{\downarrow})$$

where,

1) $O = O_1 \cup O_2$

$= \{ x_0, \ldots, x_{i_0}, m, y_0, \ldots, y_{j_0}, y_0, \ldots, y_{j_1}, y_0, \ldots, y_{j_2} \} \cup \{ \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_1}, \phi_0, \ldots, \phi_{k_1}, \gamma_0, \ldots, \gamma_{k_2} \}$

$= \{ x_0, \ldots, x_{i_0}, m, y_0, \ldots, y_{j_0}, y_0, \ldots, y_{j_1}, y_0, \ldots, y_{j_2}, \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_1}, \phi_0, \ldots, \phi_{k_1}, \gamma_0, \ldots, \gamma_{k_2} \}$

2) $O = O_1 \cup O_2 = \{ x_0, \ldots, x_{i_0}, m, y_0, \ldots, y_{j_0}, y_0, \ldots, y_{j_1}, y_0, \ldots, y_{j_2} \} \cup \{ \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_1}, \phi_0, \ldots, \phi_{k_1}, \gamma_0, \ldots, \gamma_{k_2} \} = \{ x_0, \ldots, x_{i_0}, m, y_0, \ldots, y_{j_0}, y_0, \ldots, y_{j_1}, y_0, \ldots, y_{j_2}, \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_1}, \phi_0, \ldots, \phi_{k_1}, \gamma_0, \ldots, \gamma_{k_2} \}$
3) $K = \{ \Pi_0, \Pi_1, \Pi_2 \}$
4) $\delta = \{ (\Pi_0, ref_{\downarrow_{a_1}} \rightarrow def_{\downarrow_{\Pi_1}}), (\Pi_1, ref_{\downarrow_{a_2}} \rightarrow def_{\downarrow_{\Pi_2}}), (\Pi_2, ref_{\downarrow_{a_3}} \rightarrow def_{\downarrow_{\Pi_0}}) \}$
5) $Q = \{ \alpha_0, \ldots, \alpha_{k_0}, \beta_0, \ldots, \beta_{k_1}, \phi_0, \ldots, \phi_{k_1}, \gamma_0, \ldots, \gamma_{k_2} \} / \{ \alpha_0, \ldots, \alpha_{k_0}, , \beta_0, \ldots, \beta_{k_1}, \phi_0, \ldots, \phi_{k_1} \} = \{ \gamma_0, \ldots, \gamma_{k_2} \}$
6) $D_{\downarrow} = \{ def_{\downarrow_{\Pi_0}} \}, D_{\uparrow} = \{ \}$
7) $R_{\downarrow} = \{ ref_{\downarrow_{a_3}}, ref_{\downarrow_{a_4}} \}, R_{\uparrow} = \{ \}$

We can connect Π_0 and Π_1 by the generic instantiation: $(\Pi_0, ref_{\downarrow_{a_1}} \rightarrow def_{\downarrow_{\Pi_1}})$ and connect Π_1 and Π_2 by the generic instantiation: $(\Pi_1, ref_{\downarrow_{a_2}} \rightarrow def_{\downarrow_{\Pi_2}})$ and connect Π_2 and Π_0 by the generic instantiation: $(\Pi_2, ref_{\downarrow_{a_3}} \rightarrow def_{\downarrow_{\Pi_0}})$.

D. Parallel Method

The parallel structure can be seen as a variant of the branch structure, but the operation rules in this structure are very different from the branch structure due to the large number of P modules in parallel computation and its unique parallelism. Fig. 5 shows the generic parallel structure of parallel computing modules of degree m(m is a variable).

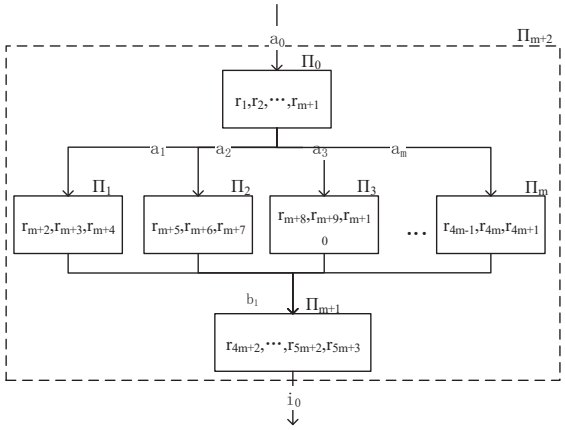

Fig. 5. The parallel structure based on the P module.

The parallelism of the parallel structure is embodied in the $1 - m$ P module. They equally inherit the public object set from Π_0, and Π_{m+1} inherits the object set of m parallel P modules.

For the parallel structure, each parallel module first inherits the same object set form Π_0, then uses some of the inheriting object sets to execute the respective calculation rules, and finally outputs the result set and remaining inherited objects to Π_{m+1}. Π_{m+1} inherits the object set of all parallel P modules, Π_{m+1} module needs to process these inheriting objects in order to run correctly .

Due to the existence of multiple inheritance and the existence of a special case of 1 pair n and n pair 1, it becomes more complicated to maintain the consistency of the data. Some inheriting object set still remain the submodule,because the parallel execution P modules use only part of the inheriting object set. In the remaining inherited objects, one is the object set as the public global variables that need pass to the child module, and the other is the redundant object set. For the first case, the submodules of the parallel modules inherits the object sets of the m P modules and results in a multiple of the

number of object sets. Therefore, it is necessary to divide the public global variables in the submodule (i.e. the public global variables/the number of the parallel P modules). In the second case, since it is useless data, each parallel submodule needs to destroy the redundant object set (because the redundant object set does not have a general purpose, so specific problems need to be specifically designed). To make the above situation clear, set the public global variable to be ω for the rule design. The following is the general structure design of the parallel structure.

In $\Pi_0 < def_{\downarrow_{\Pi_0}}, ref_{\downarrow_{a_1}}, ref_{\downarrow_{a_2}}, ref_{\downarrow_{a_3}}, \ldots, ref_{\downarrow_{a_m}} >$, the original objects include $x_0, \ldots, x_{i_0}$ and ω(it is the public global variable), using the ruleset following this paragraph, objects $\alpha_i, \beta_i, \ldots, \phi_i (i \in [1, k_0])$ can be obtained by inputting objects, $x_i (i \in [1, i_0])$.

r_1: $\Gamma/\Delta/\Phi S_0 x_0, \ldots, x_{i_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_0}$

r_2: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_0} \rightarrow_{min/max} S_0 \alpha_0, \ldots, \alpha_{k_0}$

r_3: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_0} \rightarrow_{min/max} S_0 \beta_0, \ldots, \beta_{k_0}$

$\ldots$

r_{m+1}: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_0} \rightarrow_{min/max} S_0 \phi_0, \ldots, \phi_{k_0}$

In $\Pi_1 < def_{\downarrow_{\Pi_1}}, ref_{\downarrow_{b_1}} >$, using the ruleset following this paragraph, objects $\alpha'_i (i \in [1, p_0])$ can be obtained by inputting objects,$\alpha_i (i \in [1, k_0])$. For the remaining objects, i.e. $(\beta_0, \ldots, \beta_{k_0})$, they need to be removed by the rule named r_{m+4} .

r_{m+2}: $\Gamma/\Delta/\Phi S_0 \alpha_0, \ldots, \alpha_{k_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_1}$

r_{m+3}: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_1} \rightarrow_{min/max} S_0 \alpha'_0, \ldots, \alpha'_{p_0}$

r_{m+4}: $\Gamma/\Delta/\Phi S_1 \beta_0, \ldots, \beta_{k_0}, \ldots, \phi_0, \ldots, \phi_{k_0} \rightarrow_{max} S_0$

In $\Pi_2 < def_{\downarrow_{\Pi_2}}, ref_{\downarrow_{b_2}} >$, using the ruleset following this paragraph, objects $\beta'_i (i \in [1, p_0])$ can be obtained by inputting objects,$\beta_i (i \in [1, k_0])$.

r_{m+5}: $\Gamma/\Delta/\Phi S_0 \beta_0, \ldots, \beta_{k_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_1}$

r_{m+6}: $\Gamma/\Delta/\Phi S_1 y_0, \ldots, y_{j_1} \rightarrow_{min/max} S_0 \beta'_0, \ldots, \beta'_{p_0}$

r_{m+7}: $\Gamma/\Delta/\Phi S_1 \alpha_0, \ldots, \alpha_{k_0}, \ldots, \phi_0, \ldots, \phi_{k_0} \rightarrow_{max} S_0$

Due to the uncertainty in the number of the parallel P modules, we will not list rules of other parallel P modules here. Their difference is that the use of different initialization object sets, the output set of different object sets and the remaining object sets needed to be removed.

$\Pi_{m+1} < def_{\downarrow_{\Pi_{m+1}}}, ref_{\downarrow_{i_0}} >$ can inherit the same object set,ω, from n P modules and result in the error of ω, so need to get the correct object set,ω, by ω=ω/m(rule:r_{4m+2}). Then perform corresponding calculations on different kinds of object sets. Its rule set is as listed below.

r_{4m+2}: $\Gamma/\Delta/\Phi S_0 \omega^m \rightarrow_{max} S_1 \omega$;

r_{4m+3}: $\Gamma/\Delta/\Phi S_0 \alpha'_0, \ldots, \alpha'_{p_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_2}$;

r_{4m+4}: $\Gamma/\Delta/\Phi S_0 \beta'_0, \ldots, \beta'_{p_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_2}$;

$\ldots$

r_{5m+2}: $\Gamma/\Delta/\Phi S_0 \phi'_0, \ldots, \phi'_{p_0} \rightarrow_{min/max} S_1 y_0, \ldots, y_{j_2}$;

r_{5m+3}: $\Gamma/\Delta/\Phi S_0 y_0, \ldots, y_{j_2} \rightarrow_{min/max} S_1 \gamma_0, \ldots, \gamma_{k_1}$;

The combined P module, $\Pi_{m+2} < def_{\downarrow_{\Pi_0}}, ref_{\downarrow_{i_0}} >$, contains three kinds of P modules, i.e. the control P module, the parallel P modules and the merge P module. The control P module is Π_0, the parallel P module are $\Pi_i (i \in [1, m])$,the merge P module is Π_{m+1}. There is a definition of Π_{m+2} following this paragraph.

$$\Pi_{m+2} = (O, K, \delta, Q, D_\uparrow, D_\downarrow, R_\uparrow, R_\downarrow)$$

where,

1) O =$O_1 \cup O_2$=$\{ x_0, \ldots, x_{i_0}, y_0, \ldots, y_{j_0}, y_0, \ldots, y_{j_1}, y_0, \ldots, y_{j_2} \} \cup \{ \omega, (\alpha_0, \ldots, \alpha_{k_0}), (\beta_0, \ldots, \beta_{k_1}), \ldots, (\phi_0, \ldots, \phi_{k_1}), (\gamma_0, \ldots, \gamma_{k_2}), (\alpha'_0, \ldots, \alpha'_{p_0}), (\beta'_0, \ldots, \beta'_{p_0}), \ldots, (\phi'_0, \ldots, \phi'_{p_0}) \}$=$\{ x_0, \ldots, x_{i_0}, y_0, \ldots, y_{j_0}, y_0, \ldots, y_{j_1}, y_0, \ldots, y_{j_2}, (\alpha_0, \ldots, \alpha_{k_0}), (\beta_0, \ldots, \beta_{k_1}), \ldots, (\phi_0, \ldots, \phi_{k_1}), (\gamma_0, \ldots, \gamma_{k_2}), (\alpha'_0, \ldots, \alpha'_{p_0}), (\beta'_0, \ldots, \beta'_{p_0}), \ldots, (\phi'_0, \ldots, \phi'_{p_0}), \omega \}$
2) $K = \{ \Pi_i (i \in [1, m]) \}$
3) δ=$\{ (\Pi_0, ref_{\downarrow_{a_i}} \rightarrow def_{\downarrow_{\Pi_i}})(i \in [1, m]), (\Pi_i, ref_{\downarrow_{b_1}} \rightarrow def_{\downarrow_{\Pi_{m+1}}})(i \in [1, m]) \}$
4) $Q = \{ \omega, (\alpha_0, \ldots, \alpha_{k_0}), (\beta_0, \ldots, \beta_{k_1}), \ldots, (\phi_0, \ldots, \phi_{k_1}), (\gamma_0, \ldots, \gamma_{k_2}), (\alpha'_0, \ldots, \alpha'_{p_0}), (\beta'_0, \ldots, \beta'_{p_0}), \ldots, (\phi'_0, \ldots, \phi'_{p_0}) \} / \{ (\alpha_0, \ldots, \alpha_{k_0}), (\beta_0, \ldots, \beta_{k_1}), \ldots, (\phi_0, \ldots, \phi_{k_1}), (\alpha'_0, \ldots, \alpha'_{p_0}), (\beta'_0, \ldots, \beta'_{p_0}), \ldots, (\phi'_0, \ldots, \phi'_{p_0}) \} = \{ \omega, \gamma_0, \ldots, \gamma_{k_2} \}$
5) $D_\downarrow = \{ def_{\downarrow_{\Pi_0}} \}, D_\uparrow = \{ \}$
6) $R_\downarrow = \{ ref_{\downarrow_{i_0}} \}, R_\uparrow = \{ \}$

We can connect Π_0 and the parallel P modules,$\Pi_i (i \in [1, m])$ by the generic instantiation: $(\Pi_0, ref_{\downarrow_{a_i}} \rightarrow def_{\downarrow_{\Pi_i}})(i \in [1, m])$ and connect $\Pi_i (i \in [1, m])$ and Π_{m+1} by the generic instantiation: $(\Pi_i, ref_{\downarrow_{b_1}} \rightarrow def_{\downarrow_{\Pi_{m+1}}})(i \in [1, m])$.

V. An Example: P System Design based on P Module

Based on the high computational complexity of calculating the arithmetic square root of a large number, this section proposes an efficient algorithm to reduce its computational complexity, and implement the algorithm in the P system by using the P module and four P module construction methods.

A. Square Root Algorithm of a Large Number

Two algorithms for calculating the square root of a large number are introduced here. One is a square root estimation algorithm for estimating the scope of the square root. This algorithm is illustrated by Table I.

The algorithm is applied to estimate the square root, including four steps and the time complexity of $Sqrte(n)$ is about $O(d)$ (d is the number of digits of n).

The other is a square root algorithm through m bisection calculation algorithm. This algorithm is an improved algorithm of 2-points, which is illustrated by Table II.

According Table II, the complexity of $Mbisection$ (b,n,a,m) is about $O(log_m n)$ (m is the number of the interval splited, n is the interval size).

TABLE I. ALGORITHM:$EstimateSqr(\mathrm{x})$

Input: x
Output: the left interval point of estimated square root and the interval size of estimated square root
Steps:
(1) Circularly dividing 100 with no remainder,then get the high-value of the inputting number expressed as the numbers, y,and the number of cycles.
(2) Select the square root of the high-value according to the formula, i.e. the high-value $>[81, 64, 49, 36, 25, 16, 9, 4, 1]$, then the high-value square root result is $[10, 9, 8, 7, 6, 5, 4, 3, 2]$.
(3) Combinate the square root of the high-value and the cycles.
(4) Output: the left interval point of estimated square root and the size of estimated square root.
End

TABLE II. $Mbisection(b, l, x, m)$

Input: b, l, x
x :the original number to be squared;
b: the left interval point of $\sqrt{x}$;
l: the size of the interval of$\sqrt{x}$;
m: the quantity of equidistant intervals
original interval: $[b, b + l]$
Output: the square root
Steps:
(1) Splite the interval into m equidistant intervals and obtain $m + 1$ copies of endpoint number in the interval, $x_0, x_1, \ldots, x_m$.
(2) Parallelly calculate $f(x_i) = x_i^2 - x (i \in [0, m])$, then output these key value pairs $(x_i, f(x_i))$ $(i \in [0, m])$.
(3) Filter these key value pairs, if there is the number, $f(x_i) = 0$, then output x_i as the final result; otherwise, output two adjacent numbers, x_i, x_{i+1} when $f(x_i) < 0$ and $f(x_{i+1}) > 0$.
(4)If $x_{i+1} - x_i >= 2$, x_{i+1} and x_i are as a new round of inputting and enter (1); otherwise, output $x_i + 1$ as the final result.
End

B. Square Root Algorithm of a Large Number based on the P Module

The previous chapter proposed two algorithms for solving the square root of a large number, but the computational efficiency of each algorithm is not optimized. To reduce the computational complexity, the two algorithms can be combined to form an efficient algorithm named $Bigsplite$, which integrates the square root estimation algorithm, $EstimateSqr(\mathrm{x})$, and the square root algorithm through m bisection calculation algorithm, $Mbisection(b, l, x, m)$. By using P modules, the P system, $Bigsplite$, has high powerful parallel execution capabilities, high reusability and low coupling.

Provided that input α, the square root algorithm of a large number based on the P module, $Bigsplite(\alpha)$, is illuminated by Fig. 6. While Fig. 6 shows that how to construct the P system by using P modules. In the beginning of the system construction, the initial objects structure of the elementary modules only contains one object, c, except for $\Pi_1 \ldots \Pi_9$. The objects that corresponds to $\Pi_1 \ldots \Pi_9$ are $b^4c^4q^2s$, $b^9c^5q^3s$,$b^{16}c^7q^4s$, $b^{25}c^9q^5s$, $b^{36}c^{11}q^6s$, $b^{49}c^{13}q^7s$,$b^{64}c^{15}q^8s$, $b^{81}c^{17}q^9s$, $b^{100}c^{19}q^{10}s$. The specific rules of this algorithm in the P system are shown in the appendix. According to the flow chart and the rule sets, the detailed implementation process is described in detail below.

1) Firstly, copy α to τ for saving global parameters,α. Through circularly dividing the data, τ, by 100, the quotient of the cycle calculation value- δ and the number of cycle calculations-θ are obtained after the cycle ends. At present, there are public objects in Π_0, i.e. the original objects, α, the highest-segment value of α, δ, and the bits of $\sqrt{\alpha}$ except the highest-bit, θ. Go to the step 2.
2) In $\Pi_1, \ldots, \Pi_9$, these P modules parallel compute the square root of the high-segment value, δ, inheriting from Π_0. Through finding a P module of $\Pi_1, \ldots, \Pi_9$ that make the value, q, satisfying the arithmetic formula: $\sqrt{\delta} < q < \sqrt{\delta} + 1$ and converting q to be $\beta + \gamma(\gamma = 1)$, $\sqrt{\delta}$ can be expressed by $\beta < \sqrt{\delta} < \beta + \gamma(\gamma = 1)$. For other P modules, they need to remove inherited public objects, α and θ. Finally, there are public objects in $\Pi_i(\Pi_i is the selected cell)$, i.e. the original objects, α, the bits of $\sqrt{\alpha}$ except the highest-bit, θ, the left interval point of $\sqrt{\delta}$, β, and the size of $\sqrt{\delta}$, γ. Go to step (3). (Parallel Computing)
3) In Π_{10}, combinate of digits, θ, and the valuation interval of the square root of high-segment value, $[\beta, \beta+\gamma]$, then get the valuation range of α, i.e. $[\beta * 10^\theta, (\beta + \gamma) * 10^\theta]$ which is replaced by $[\beta, \beta + \gamma](\gamma = 1)$. Finally, public objects have the original objects, α, the left valuation inteval point, β, and the valuation interval size, γ. Go to step (4).
4) In Π_{12}, the valuation interval, $[\beta, \beta + \gamma](\gamma = 1)$, is divided into m intervals on average, which are showed as $m + 1$ endpoints, $\varphi_0, \varphi_1, \ldots, \varphi_m$(they are also $\psi_0, \psi_1, \ldots, \psi_m$). Public objects include the original objects, α, $\varphi_0, \varphi_1, \ldots, \varphi_m$ and $\psi_0, \psi_1, \ldots, \psi_m$. Go to step (5).
5) In $\Pi_{30+i}(i \in [0, m])$, they respectively calculate $f(\varphi_i, \psi_i) = \varphi_i * \psi_i - \alpha$ by the objects they want to use, i.e. $\varphi_i, \psi_i(i =$the id of P module$-30)$, and respectively destroy other objects, i.e. $\varphi_i, \psi_i(i \neq$the id of cell$-30)$. If $f(\varphi_i, \psi_i) == 0$, producing $\gamma_i(\gamma_i = \varphi_i)$; else if $f(\varphi_i, \psi_i) < 0$, producing $\alpha_i(\alpha_i = \varphi_i)$; else if $f(\varphi_i, \psi_i) > 0$, producing $\beta_i(\beta_i = \varphi_i)$. There must be

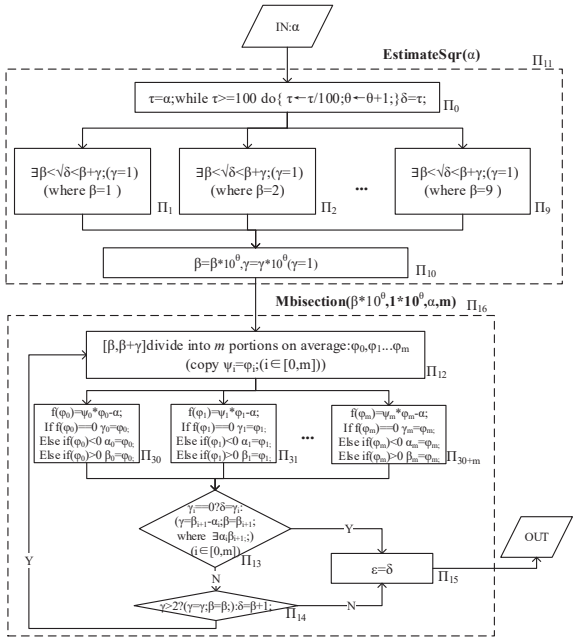

Fig. 6. The square root algorithm of large number based on the P module.

two kinds of objects for each module, one is one of the $\alpha_i, \beta_i, \gamma_i (i =$the id of cell$-30)$, the second is that each module has a common global variable, α. Go to step (6).(Parallel Computing)

6) In Π_{13}, determine whether there is an accurate result from the results of the calculation, γ_i, if so, then the square root value, $\delta = \gamma_i$, inherited by Π_{15} and go to step(7); otherwise, find the interval of the square root, i.e. $[\beta, \beta + \gamma]$=$[\alpha_i, \beta_{i+1}]$. At present, there is either δ and α or β, γ and α. Go to step (8).
7) In Π_{15},output the final result $\varepsilon(\varepsilon = \delta)$.
8) In the cell Π_{14}, determine whether the size of the interval, i.e. γ, is more than 2, if so, the result, $\delta = \beta + 1$ is given to Π_{15} and go to step (7); otherwise,pass β, γ, α to Π_{12} and go to step (4).

The square root algorithm of a large number based on the P module, $Bigsplite(\alpha)$, can pass through two phases, $EstimateSqr(\alpha)$ and $Mbisection(\beta, \gamma, \alpha, m)$, when the result interval of the square root is $[\beta, \beta + \gamma]$ by $EstimateSqr(\alpha)$. The size of the second parameter of the algorithm named $Mbisection$ is much smaller than the original input number, α, so the total number of the recursive computing is decreased, but the time complexity of $Mbisection(\beta, \gamma, \alpha, m)$ is still $O(log_m n)$. So the time complexity of the square root algorithm of large number based on the P module is $O(log_m n)$.

C. Structure of P Module for Calculating the Square Root of a Large Number

Fig. 7 is a P module flow chart illustrating the modular combination of P modules that calculates the square root of a large number.

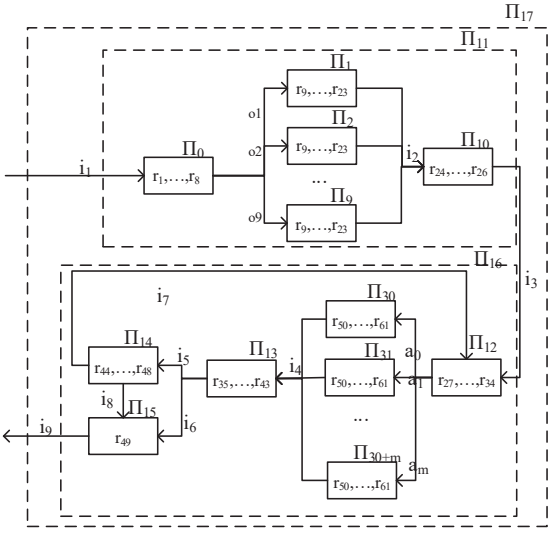

Fig. 7. A P module flow of a parallel algorithm.

As is shown Fig. 7, the number of P modules in the P system are $(19 + m)$. The P system can be expressed as the biggest combination P module $\Pi_{17} < def_{\downarrow_{\Pi_0}}, ref_{\downarrow_{i_9}} >$. The entire definition and operation mechanism of the system is mainly represented by public object sets, rule sets, and construction and assembly mechanism. The description of the public object set can not only show the evolutionary direction and flow of the entire system calculation, but also more easily incorporate the rule set in the appendix to ensure the correctness of the rule design of P system. The following is a simple description of the public and private objects in the evolution of the rules of each module.

1) In Π_0, O_1={ τ, b, c } and O_2={ α, θ, δ };
2) In the parallel P modules of Π_{10}, O_1={ b, c, q, s, e } and O_2={ $\alpha, \theta, \delta, \beta, \gamma$ };
3) In Π_{11}, O_1={ θ, c } and O_2={ α, β, γ };
4) In Π_{13}, O_1={ $c, \gamma, \beta, d_i, b_i$ } and O_2={ $\alpha, \varphi_i, \psi_i$ };$(i \in [0, n])$
5) In the parallel P modules of Π_{14}, O_1={ c, v, r, ϕ_i, ψ_i } and O_2={ $\alpha, \alpha_i, \beta_i, \gamma_i$ };$(i \in [0, n])$
6) In Π_{15}, O_1={ $\alpha_i, \beta_i, \gamma_i, c, p_i$ } and O_2={α, δ, γ, β };$(i \in [0, n])$
7) In Π_{16}, O_1={ c } and O_2={ $\alpha, \delta, \gamma, \beta$ };
8) In Π_{17}, O_1={ c, δ } and O_2={ α, ε };

As a whole, the biggest P module,$\Pi_{17} < def_{\downarrow_{\Pi_0}}, ref_{\downarrow_{i_9}} >$, includes a closely related combination of two functional combination P modules, Π_{11}<$def_{\downarrow_{\Pi_0}}$,$ref_{\downarrow_{i_3}}$> and Π_{16}<$def_{\downarrow_{\Pi_{12}}}$,$ref_{\downarrow_{i_9}}$>. Π_{11} implements the algorithm $EstimateSqr(x)$ and Π_{19} implements the algorithm $Mbisection(b, x, a, m)$. These two P modules form a sequential structure which describes that narrow the computing scope in the first step and accurately calculate to obtain the final result in the next step.

The combination P module Π_{11}<$def_{\downarrow_{\Pi_0}}, ref_{\downarrow_{i_3}}$> mainly describes a square root estimation algorithm for estimating the scope of the square root. Π_{11} is a parallel structure which is combinated by 11 P modules $\Pi_i (i \in [0, 10])$. $\Pi_i (i \in [1, 9])$ are the core parallel modules.

In the elementary P module Π_0 <$def_{\downarrow_{\Pi_0}}$, $ref_{\downarrow_{o_1}}$, $ref_{\downarrow_{o_2}}$, $ref_{\downarrow_{o_3}}$, $ref_{\downarrow_{o_4}}$, $ref_{\downarrow_{o_5}}$, $ref_{\downarrow_{o_6}}$, $ref_{\downarrow_{o_7}}$, $ref_{\downarrow_{o_8}}$,$ref_{\downarrow_{o_9}}$>, mainly obtain the high-value and the number of digits other than the high-value by processing the original data in parallel. The rules in $\Pi_i < def_{\downarrow_{\Pi_i}}, ref_{\downarrow_{i_2}} > (i \in [1, 9])$ are exactly the same. $\Pi_{10} < def_{\downarrow_{\Pi_{10}}}, ref_{\downarrow_{i_3}} >$ outputs the estimated square root result.

The combination P module $\Pi_{16} < def_{\downarrow_{\Pi_{12}}}, ref_{\downarrow_{i_9}} >$ mainly describes a square root algorithm through m bi-section calculation approach. In a whole, Π_{16} is a cycle structure, do-while. Two ways which include $(\Pi_{13} \rightarrow \Pi_{14} \rightarrow \Pi_{15})$ and $(\Pi_{13} \rightarrow \Pi_{15})$ can end the cycle. $(\Pi_{13} \rightarrow \Pi_{14} \rightarrow \Pi_{15})$ is executed when the size of the interval is not greater than 2. $(\Pi_{13} \rightarrow \Pi_{15})$ is executed when the exact square root value are obtained. If the size of interval is more than 2, the cycle continue. $\Pi_{12} < def_{\downarrow_{\Pi_{12}}}, ref_{\downarrow_{a_0}}, ref_{\downarrow_{a_1}}, \ldots, ref_{\downarrow_{a_m}} >$ splite the interval into m equidistant intervals and obtain $m + 1$ copies of endpoint number in the interval,$\varphi_0, \varphi_1, \ldots, \varphi_m (\psi_0, \psi_1, \ldots, \psi_m)$ Also, Π_{16} is a parallel structure. $\Pi_{30+i} (i \in [0, m])$ is the parallel P modules in the combination P module Π_{16}, which calculate a formula, $f(\varphi_i, \psi_i) = \varphi_i * \psi_i - \alpha (i \in [0, m])$.

D. Calculate Instance

We assume that the inputting data is 69399 and the number of parallel computing P module is 11, namely, m=11. So 69399 copies of objects into the membrane system to evolve.

1) The membrane structure after the original data, Π_0 includes O_2=$\{\ \alpha^{69399}\ \}$and O_1=$\{\ c\ \}$. Then obtaining the set of public objects, O_2= $\{\ \delta^6, \theta^2, \alpha^{69399}\ \}$.
2) The parallel P modules in Π_{10}, namely, $\Pi_1, \ldots, \Pi_9$, inherit the public objects from Π_0 and start the core calculation of $EstimateSqr(69399)$. After the parallel calculation, Π_{11} inherits β^2, $\gamma^1, \theta^2, \alpha^{69399}$ from Π_{10}. Then Π_{11} gets β^{200}, $\gamma^{100}$and α^{69399} after calculation.
3) Start $Mbisection(200, 69399, 100, 11)$. Π_{13} inherits the objects from Π_{11}, so the inputting objects in Π_{19} are β^{200}, γ^{100} and α^{69399}. By the rules, Π_{13} gets objects, O_2=$\{\ \phi_i^{200+i*10}(i \in [0,10]), \psi_i^{200+i*10}(i \in [0,10]), \alpha^{69399}\ \}$. Then $\Pi_{30+i}(i \in [0,10])$ correspondingly inherit O_2 in Π_{13}. Namely, in $\Pi_{30+i}(i \in [0,10])$, O_2=$\{\ \phi_i^{200+i*10}(i \in [0,10]), \psi_i^{200+i*10}(i \in [0,10]), \alpha^{69399}\ \}$.
4) After the first cycle of the algorithm, $Mbisection(200, 69399, 100, 11)$, Π_{16} gets α_6^{260},β_7^{270} and α^{69399} from Π_{36} and Π_{37}. Then Π_{16} gets β^{260}, γ^{10} and α^{69399} from Π_{15}.
5) After the second cycle of the algorithm, $Mbisection(260, 69399, 10, 11)$, Π_{16} gets α_1^{262},β_2^{264} and α^{69399} from Π_{31} and Π_{32}. Then Π_{16} gets β^{262}, γ^2 and α^{69399} from Π_{15} and ends, while Π_{17} inherits δ^{263} from $\Pi_{15}(\delta^{263} = \beta^{262} + c)$ and saved as ε^{263}. The objects ε^{263} represents the square root of 69399. That means the final result is the number, 263.

VI. Conclusion

In this paper, the design and assembly of the P module is formed to provide a framework for constructing a P system by recursive combined P modules, so that the P module combines the three characteristics of packaging, information hiding and modular combination. By designing a well-structured P module, the correctness of the dynamic execution of the P system is ensured with a good structure, the efficiency of software development is improved, and the error rate is reduced.

As the application of this paper, we use the definition and design method of P module to solve the large number square root problem. By designing a P system for solving the square root of a large number, the correctness and high efficiency of the P system design method based on the P module are clarified.

The design method in this paper is mainly aimed at the cell-like P system and further work can apply it to other models of P systems, such as the tissue-like P system and neural-like P system.

References

[1] C.MartÍn-Vide, Gh.Păun, G.Rozenberg. "Membrane systems with carriers". Theoretical Computer Science, 2002, 270 (l): 779-796

[2] A.Vitale, G.Mauri, C.Zandron. "Simulation of abounded symport/antiport P system with Brane calculi". Biosystems, 2008, 91(3): 558-571.

[3] R.Freund, Gh.Păun, M.J.Pérez-Jiménez. "Tissue P systems with channel states". Theoretical Computer Science, 2005, 330(l): 101-116.

[4] O.H.Ibarra, A.Păun, et al. "Normal forms for spiking neural P systems". Theoretical Computer Science, 2007, 372(2): 196-217.

[5] Guo P, Dai Y, Chen H. "A p system for Hamiltonian cycle problem". Optik - International Journal for Light and Electron Optics, 2016, 127(20):8461-8468.

[6] Guo P, Zhu J, Zhou M. "A Family of Uniform P Systems for All-Satisfiability Problem". Journal of Computational & Theoretical Nanoscience, 2016, 13(1):135-142.

[7] Ping Guo, Yuwen Zhong, Haizhu Chen, Mingqiang Zhou. "A P system for finding all solutions of the degree-constrained spanning tree problem". Pre-Proceedings of the Second Asian Conference on Membrane Computing (ACMC2015), 2015.

[8] Ping Guo, Junqi Xiang, Jingya Xie, Jinhang Zheng, "A P System for Solving All-Solutions of TSP", International Journal of Advanced Computer Science and Applications, Vol. 8, No. 9, 2017, 357-364

[9] Yahya, R.I., Shamsuddin, S.M., Hasan, S., Yahya, S.I., "Tissue-like P system for segmentation of 2D hexagonal images". ARO- Sci. J. Koya Univ. 4(1), 3542 (2016).

[10] Isawasan, P., Venkat, I., Subramanian, K., Khader, A., Osman, O., Christinal, H., "Region-based segmentation of hexagonal digital images using membrane computing". In: 2014 Asian Conference on Membrane Computing (ACMC). IEEE (2014).

[11] A.Atanasiu. "Arithmetic with membranes". Romanian Journal of Information Science and Technology, 2001, 4(1): 5-20.

[12] P.Guo, J.Chen. "Arithmetic operation in membrane system". In:Proceedings of International Conference on BioMedical Engineering and Informatics. Sanya, Hainan, China, 231-234(2008).

[13] Ruilong Yang, Ping Guo, Jia Li, Ping Gu, "Arithmetic operations with membranes based on arithmetic formula tables", Chinese Journal of Electronics, 2015, 24(2).

[14] Ping Guo, Lijiao Wei, Haizhu Chen. "A P Systems for Matrix-Vector Multiplication", Journal of Computational and Theoretical Nanoscience, Vol.12, No. 11, pp. 4279-4288, 2015.

[15] Francisco J. Romerocampero, Jamie Twycross, Miguel Camara, et al. "Modular assembly of cell systems biology models using P systems". International Journal of Foundations of Computer Science, 2009, 20(03):427-442.

[16] Şerbănuţă T, Ştefănescu G, Roşu G. "Defining and Executing P Systems with Structured Data in K". Membrane Computing. Springer-Verlag, 2009:374-393.

[17] Păun G, Pérez-Jiménez M D J. "Solving Problems in a Distributed Way in Membrane Computing: dP Systems". International Journal of Computers Communications & Control, 2010, 5(2):238-250.

[18] Dinneen M J, Kim YB, Nicolescu R. "P systems and the Byzantine agreement". Journal of Logic& Algebraic Programming, 2010, 79(6):334-349.

[19] Dinneen M J, Kim Y B, Nicolescu R. "A Faster P Solution for the Byzantine Agreement Problem". International Conference on Membrane Computing. Springer-Verlag, 2010:175-197.

[20] Dinneen M.J., Kim YB, Nicolescu R. "Synchronization in P Modules". International Conference on Unconventional Computation. Springer-Verlag, 2010:32-44.

[21] Nicolescu R. "Parallel and Distributed Algorithms in P Systems". International Conference on Membrane Computing. Springer-Verlag, 2011:35-50.

[22] Nicolescu R, Wu H. "BFS Solution for Disjoint Paths in P Systems". UC'11 Proceedings of the 10th international conference on Unconventional computation, Springer-Verlag, 2011:164-176.

[23] Guo P, Zhang H, Chen H, et al. "Fraction Reduction in Membrane Systems". The Scientific World Journal,2014(2):858527.

Appendix: RuleSet

1 The rules in Π_0
r_1: $S_0\alpha \rightarrow_{max} S_1\alpha\tau$;
r_2: $S_1\tau^{100}c \rightarrow_{min} S_2\tau^{100}c$; 1
r_3: $S_1\tau c \rightarrow_{min} S_5\tau c$;2
r_4: $S_2\tau^{100} \rightarrow_{max} S_3 b$;
r_5: $S_3\tau \rightarrow_{max} S_4$;
r_6: $S_3 b \rightarrow_{min} S_4 b\theta$;
r_7: $S_4 b \rightarrow_{min} S_1\tau$;
r_8: $S_5\tau \rightarrow_{min} S_0\delta$;

2 The rules in Π_1 to Π_9

r_9: $S_0\delta b \rightarrow_{max} S_1 d$;
r_{10}: $S_1 bs \rightarrow_{min} S_2 bs; 1$
r_{11}: $S_1 s \rightarrow_{min} S_4 s; 2$
r_{12}: $S_2 bc \rightarrow_{min} S_3 e$;
r_{13}: $S_3 bs \rightarrow_{min} S_4 bs; 1$
r_{14}: $S_3 s \rightarrow_{min} S_5 s; 2$
r_{15}: $S_4\delta \rightarrow_{max} S_0$;
r_{16}: $S_4 e \rightarrow_{max} S_0 bc; 1$
r_{17}: $S_4 d \rightarrow_{max} S_0 b; 2$
r_{18}: $S_4\alpha \rightarrow_{max} S_0$;
r_{19}: $S_4\theta \rightarrow_{max} S_0$;
r_{20}: $S_5 qs \rightarrow_{max} S_0\gamma; 1$
r_{21}: $S_5 q \rightarrow_{max} S_0\beta; 2$
r_{22}: $S_5 e \rightarrow_{max} S_0 bc; 2$
r_{23}: $S_5 d \rightarrow_{max} S_0 b; 2$

3 The rules in Π_{10}
r_{24}: $S_0\theta c \rightarrow_{max} S_1 c$
r_{25}: $S_1\beta \rightarrow_{max} S_0\beta^{10}$
r_{26}: $S_1\gamma \rightarrow_{max} S_0\gamma^{10}$;

4 The rules in Π_{12}
r_{27}: $S_0\gamma^n \rightarrow_{max} S_1 d_0 d_1 \dots d_n; 1$
r_{28}: $S_0\beta \rightarrow_{max} S_1 b_0 b_1 \dots b_n; 1$
r_{29}: $S_0\gamma c \rightarrow_{max} S_1\gamma c; 2$
r_{30}: $S_1\gamma c \rightarrow_{max} S_2 c d_0 d_1 \dots d_n; 1$
r_{31}: $S_1 c \rightarrow_{max} S_2 c; 2$
r_{32}: $S_1\gamma \rightarrow_{max} S_2; 3$
r_{33}: $S_2 d_i \rightarrow_{max} S_0\varphi_i^i\psi_i^i$;
r_{34}: $S_2 b_i \rightarrow_{max} S_0\varphi_i\psi_i$;

5 The rules in Π_{13}
r_{35}: $S_0\alpha_m \rightarrow_{max} S_1\alpha; 1$
r_{36}: $S_0\gamma_i c \rightarrow_{min} S_1 c\gamma_i; 1$
r_{37}: $S_0 c \rightarrow_{min} S_2 c; 2$
r_{38}: $S_1\gamma_i \rightarrow_{max} S_3\delta$;
r_{39}: $S_2\alpha_i\beta_{i+1} \rightarrow_{max} S_3 p_{i+1}\beta_{i+1}\beta$;
r_{40}: $S_3\beta_i p_i \rightarrow_{max} S_3\gamma; 1$
r_{41}: $S_3 p_i \rightarrow_{max} S_0; 2$
r_{42}: $S_3\alpha_i \rightarrow_{max} S_0; 2$
r_{43}: $S_3\beta_i \rightarrow_{max} S_0; 2$

6 The rules in Π_{14}
r_{44}: $S_0 c\gamma^3 \rightarrow_{min} S_0 c\gamma^3; 1$
r_{45}: $S_0 c \rightarrow_{min} S_1 c; 2$
r_{46}: $S_1\beta \rightarrow_{max} S_0\delta$;
r_{47}: $S_1 c \rightarrow_{max} S_0\delta c$;
r_{48}: $S_1\gamma \rightarrow_{max} S_0$;

7 The rules in Π_{15}
r_{49}: $S_0\delta \rightarrow_{max} S_0\varepsilon$;

8 The rules in Π_{30} to Π_{30+m}
r_{50}: $S_0\psi_i c \rightarrow_{min} S_1 c; 1$
r_{51}: $S_0 c \rightarrow_{min} S_2 c; 2$
r_{52}: $S_1\varphi_i \rightarrow_{max} S_0\varphi_i r$;
r_{53}: $S_2\alpha r \rightarrow_{max} S_3 v$;
r_{54}: $S_3\alpha v \rightarrow_{max} S_4\alpha v; 1$
r_{55}: $S_3 rv \rightarrow_{max} S_5 v; 2$
r_{56}: $S_3 v \rightarrow_{max} S_6 v; 3$
r_{57}: $S_4\varphi_i \rightarrow_{max} S_7\alpha_i$;
r_{58}: $S_5\varphi_i \rightarrow_{max} S_7\beta_i$;
r_{59}: $S_6\varphi_i \rightarrow_{max} S_7\gamma_i$;
r_{60}: $S_7 v \rightarrow_{max} S_0\alpha$;
r_{61}: $S_7 r \rightarrow_{max} S_0$;