Tissue P systems with parallel rules on channels *

Xu Xian **

(Department of Computer Science and Technology, Shanghai Jiao Tong University, Shanghai 200240, China)

Abstract A new variant of P systems as an improvement of the original design is introduced. The P systems here allow parallelism in rules application on synapses (or links, channels), rendering the systems more efficient. We describe the execution mechanism of our P systems, make analysis on its computation power, and illustrate the running by an example.

Keywords: tissue P systems, parallel rules.

Cell-like P systems[1-3], of which kind the most typical one is the transition P system, are a kind of distributed computing device. The thought is inspired from the organization of living cells. Tree-like nested membrane structure (no linking), reaction rules inside membranes and objects communicating among membranes are characteristics of cell-like P systems. With the maximal parallelism in rules application, cell-like P systems, together with a wealth of variants, possess the computation power of Turing machines.

Tissue-like P systems are a different kind of systems that mimic the interaction among cells constituting tissues of various sorts. They emphasize the collaboration rather than the process within the cells. So tissue-like P systems have graph-like membrane linking (no nested structure). The characteristics of tissue-like P systems are as follows: (1) The primary rules are symport and antiport rules[4]. (2) The membranes (cells) can have states to help enrich the evolving of the objects in them[5]. (3) The links can have states to control the communication between cells[6].

On links, rules are applied in a sequential manner. The loss in computational power caused by eliminating parallelism is somewhat compensated by the state-on-link mechanism. At the system level (among cells/links) the rules are applied in parallel.

a) Two cells, sufficient states and small-weight antiport rules lead to Turing completeness. b) One cell (any number of states and however rules are designed) results in Parikh images of languages generated by matrix grammars without appearance checking.

See [5, 6] for more specific explanations of tissue-like P systems.

1 Motivation

We extend the parallelism to the rule application on links (or channels) among cells, that is, from on-channel-sequentiality and inter-channel-parallelism to on- and inter-channel-parallelism. We call it the largest parallelism to distinguish it from the maximal parallelism, which means that no futher rules can be applied to the remaining objects in the current membrane, and this can be seen from the definition and algorithm below in this paper. The extension can be justified in two aspects.

(1) From the biological point of view, the reactions of different kinds usually take place in a manner that a reaction is applied whenever possible. The communication between cells is an automatic process whenever some condition (such as concentration and temperature) is met. And in fact it is frequent that several communications are enabled at the same time in some state of the links. In other words, the highly randomized biological processes concerning communication in tissues are surely not confined to sequential functioning. We can mention the signaling pathways as a kind of examples. In a typical pathway (for instance the Wnt pathway, a signaling pathway related to hematopoiesis), the triggering of the initial signal on the surface of a cell can be caused by several kinds of molecules, resulting in several parallel sub-pathways to the nucleus.

* Supported by National Natural Science Foundation of China (Grant Nos. 60473006 and 60225012), BDCC (03DZ14025), MSRA, and The BoShiDian Research Fund (20010248033)

** E-mail: xuxian@sjtu.edu.cn

(2) From the theoretical viewpoint of a computing device, parallelism is a very significant element in computation concering various scientific areas, such as physics, astrology, etc.. It well transcends sequential computation in the sense that it can provide a much neater, more natural and intuitive computational model in a lot of cases. A body of theories (parallel computing theory for example) and technologies (grid computing for example) have been studied and designed to exploit the strength in parallelism.

2 Tissue P systems with parallel rules on channels

The basic idea of the (largest) parallelism is the thought of semi-steps, that is, to place a pool for each channel in the dynamically executing system, as rules are applied in parallel among different channels. Under the system global clock, each step is divided into two halves, or semi-steps.

(1) The first semi-step serves as a synchronizer to collect all the rules possible to the effect into the pool associated with the channel. Notice that the rules put into the pool must have some parallel characteristics, that is they should not depend on each other to grant them the ability to execute simultaneously. Otherwise, they have to be applied in sequentiality, as in the original tissue P systems with channel states in [6].

(2) The second semi-step is simple. Just execute all the rules in the pool, which empties the pool for another full clock step.

In the original tissue P systems with channel states, all the states of the channels in the system are from the same states set K, but actually this is somewhat of little realistic sense, since in real biological systems the possible states of different communicating channels are probably distinct from one another, maybe only interleaving in some part of them. So it is better generalized to that each channel has its own states set, for example K_{ij} for all possible states of channel from cell i to cell j.

2.1 Definition

Now we give the formal definition.

Definition 1. A tissue P System with parallel rules on channels is a construct.

$$\Pi = (O, T, Syn, (\omega_i)_{1\leqslant i\leqslant m}, E, (K_{ij})_{(i,j)\in Syn}, (s_{ij})_{(i,j)\in Syn}, (R_{ij})_{(i,j)\in Syn}, i_0)$$

where m is the degree of the system, that is, the number of membranes; O, the finite objects set, or alphabet; $T\subseteq O$, the terminal objects; Syn is synapses (another word for "links"), $Syn\subseteq\{(i,j)\mid i,j=0,1,2,\cdots,m\}$ (0 for the environment). It is an irreflexive, asymmetric and ordered relation on cells. $(\omega_i)\in O^*$ is initial objects (multiset) in each cell of the system; $E\subseteq O$ are objects (in sufficient supply) in the environment; K_{ij} is the finite set of states of synapse (i,j); s_{ij}, the initial state of synapse (i,j); and R_{ij}, the finite set of rules on synapse (i,j). The rules are of the following form:

$$(s_1, u/v, s_2),\quad s_1,s_2\in K_{ij},\quad u,v\in O^*$$

It means that in state s_1, the synapse (i,j) can perform an antiport action, that is, transport objects u from cell i to cell j in exchange for objects v from cell j, and then shift its state to s_2. And $i_0\in\{1,2,\cdots,m\}$ is the membrane for outputing results of computation.

Remark 1. The graph defined by Syn is directed, but the rules grant cells to communicate undirectedly, due to the antiport rules (in general form).

For convenience, we sometimes use $P_{ij}\subseteq R_{ij}$ as the pool of the synapse (i,j) in transition steps of the system. The pool is used to collect the rules possible to be applied simultaneously in the current state of the synapse.

Here, we give more definitions about the P system in Definition 1. The size of the P system defined in Definition 1 can be examined in three aspects.

Number of cells: m; Number of states: $k\triangleq\max_{(i,j)\in Syn}|K_{ij}|$; Size of rules: $r\triangleq\max_{(i,j)\in Syn}\{|u|+|v| \mid (s_1,u/v,s_2)\in R_{ij}\}$.

2.2 The execution of the P system

Below we describe the algorithm (or mechanism) of one time step in the execution of a P system defined in Definition 1. We focus on one synapse (i,j), as all the synapses have the same mechanism. Suppose the synapse (i,j) is currently in state s_1.

(1) In the first semi-step: selecting.

(a) Choose all the rules $(s_1, u/v, s)(s\in K_{ij})$ on the synapse and put them into P_{ij}.

(b) Classify the rules in P_{ij} by the state s. That is, divide the rules by the equlvalence relation saying that two rules are equivalent if their goal states are the same. Thus we obtain several equivalence classes of rules.

(c) Choose the equivalence class of the largest size, that is, the equivalence class holding the largest number of rules, and eliminate other rules from P_{ij}. If there are more than one such class, choose one randomly. Suppose the winning equivalence class is RC (rules class). We use $|RC|$ to denote the size of the rules class.

Here we can see that both the largest parallelism and nondeterminism principles apply, with the strethening that the largest number of rules are selected at present.

(2) In the second semi-time-step: executing.

(a) Remove from RC those rules that cannot be applied, for example, the desired objects do not exist.

(b) If the rules in RC (possibly only one) have no conflict (whose meaning we will make clear below), then we are done before executing. Rename RC to RC' and go to (d).

(c) Some conflict exists, for example (typically)[1], several rules in RC compete for the same object (s) in cell i or j. In this case, the successful application of one of the conflicting rules may fail again for application. Then RC can be divided again by the following equivalence relation: Two rules are equivalent if they compete for the same objects u (in i or j).

What we get are several classes of conflicting rules with respect to some objects, and there is no conflict between the classes[2].

Now we can choose one rule from each conflicting rules class as the lucky one, and this choice is not important since rules are deemed equivalent in the same class (for now no priority on rules is assumed in the system). So we gain a new set of rules, the number of which is the number of conflicting classes. We denote this set of rules as RC'.

(d) Execute the rules in RC' simultaneously and pass the state of the synapse (i, j) to the goal state indicated by the rules in it.

And this finishes one time step of execution of the P system.

Remark 2. Any rule that can be applied in a step (clock unit) must be applied, that is, the system is evolving whenever possible. If no rule is applicable on one synapse, then in this step nothing happens on that synapse, and the state remains.

2.3 The computation of the P system

The computation description is as follows:

(1) Initial configuration. m cells, with cell i having the objects ω_i in it.

(2) Discrete the computing procedure (assuming a system global clock).

In each step, apply the algorithm described above (Section 2.2) and thus update the (cells in the) system.

(3) Halting condition. The system halts if there is no rule applicable on any synapse.

(4) Results. The result of the computation should be collected in the cell i_0. And the result can be of two kinds: one is the vector representing the multiplicity of objects in the cell i_0. Those objects in $O - T$ are not counted. We denote this kind of result as $\mathcal{P}s_v(\Pi_{m,k,r})$ (with the meaning of m, k, r explained before); the other one is multiset of objects from T or O in cell i_0. This is more direct and can be used when the P system is in the generative mode. We denote this kind of result as $\mathcal{P}s_s(\Pi_{m,k,r})$ (with the meaning of m, k, r explained before).

3 An example

To illustrate the execution of the P systems defined in Definition 1, we give an example as follows.

1) In fact, this is the only kind of conflicts, as the rules are likely to be applied in one parallel step.

2) This can be achieved by a conflict-detecting algorithm starting from the single object, merging the conflicting sets and corresponding objects into multisets respectively, since the alphabet is finite, so are the multisets in the cells. We do not describe it here to maintain succinctness.

$$\Pi_1 = (O, T, Syn, \omega_1, \omega_2, \omega_3, E, (K_{ij})_{(i,j)\in syn}, (s_{ij})_{(i,j)\in syn}, (R_{ij})_{(i,j)\in syn}, (P_{ij})_{(i,j)\in syn}, i_0)$$

where $m=3$, $O=\{a,b\}$, $T=\{a,b\}$, $Syn=\{s_1, s_2, s_3\}$, in which $s_1:(0,1)$, $s_2:(1,2)$, $s_3:(1,3)$, $\omega_1=\omega_2=\omega_3=\lambda$. For E, we assume that the input was in the environment initially, to simplify the problem. $K_{01}=\{s, s'\}$, $K_{12}=\{t\}$, $K_{13}=\{p\}$; $s_{01}=s$, $s_{12}=t$, $s_{13}=p$; $R_{01}=\{r_{01}^1, r_{01}^2, r_{01}^3, r_{01}^4\}$, $R_{12}=\{r_{12}^1, r_{12}^2, r_{12}^3, r_{12}^4\}$, $R_{13}=\{r_{13}^1\}$, where

$r_{01}^1:(s, a/\lambda, s)$	$r_{12}^1:(t, a/\lambda, t)$	$r_{13}^1:(p, ab/\lambda, p)$
$r_{01}^2:(s, b/\lambda, s)$	$r_{12}^2:(t, b/\lambda, t)$	
$r_{01}^3:(s, a/\lambda, s')$	$r_{12}^3:(t, \lambda/a, t)$	
$r_{01}^4:(s, b/\lambda, s')$	$r_{12}^4:(t, \lambda/b, t)$	

and $i_0=3$.

The system is presented graphically in Fig. 1, where circles stand for cells. And the doubly encircled one is the output cell; the objects in each cell indicate the initial multiset of objects in it; arrows respresent synapses, with the rules associated with it along; 0 refers to the environment.

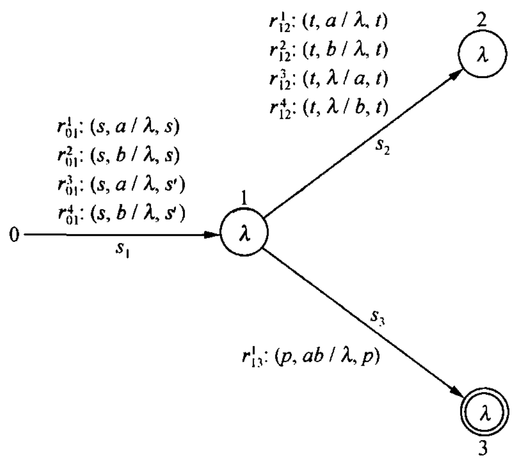

Fig. 1. System Π_1 (cells, synapses, rules and initial configuration).

Now we follow the execution algorithm described above to see what the system can do. First we compute the equivalent rule classes on each synapse. For synapse $s_1((0,1))$, the rule classes are

$$\{\{r_{01}^1, r_{01}^2\}, \{r_{01}^3, r_{01}^4\}\}$$

for synapse $s_2((1, 2))$, the rule classes are

$$\{\{r_{12}^1, r_{12}^2, r_{12}^3, r_{12}^4\}\}$$

for synapse $s_3((1,3))$, the rule class is

$$\{\{r_{13}^1\}\}$$

Then we compute RC and RC'.

(1) On synapse $s_1((0,1))$. In state s. RC can be $\{r_{01}^1, r_{01}^2\}$, then RC' is just RC; in state s'. RC can be $\{r_{01}^3, r_{01}^4\}$, then RC' is just RC.

(2) On synapse $s_2((1,2))$. In state t. RC can be $\{r_{12}^1, r_{12}^2, r_{12}^3, r_{12}^4\}$. Then RC' can be any subset of RC. For instance, when the computation is expected to halt, RC' may be $\{r_{12}^1, r_{12}^2\}$, $\{r_{12}^3, r_{12}^4\}$, RC itself, or even $\emptyset$. When the computation fails to halt eventually, RC' may be $\{r_{12}^1, r_{12}^3\}$ or $\{r_{12}^2, r_{12}^4\}$.

(3) On synapse $s_3((1,3))$. As there is only one rule on this synapse, parallelism is nullified here. The only rule is applied whenever there exists a pair of a, b in cell 1.

The actual run of the system starts from the initial state of each synapse, continues rules selection and application, each step of which involves the analysis above based on the parallel rules application algorithm (Section 2.2), and gathers the result in membrane 3. From the computation, we can see that:

(1) In state s, cell 1 brings inside a number of copies of a and b, and this process ceases when the state s shifts to state s'. At that time there has been some fixed number of a and b, say, n a and m b in cell 1.

(2) When, in cell 1, a and b are equal in number, there is a (successful) computation that the objects in cell 1 are transferred to cell 3 pair by pair, which is ab, using rule r_{13}^1 on synapse $s_3((1,3))$. This will lead to a halting condition of the system.

(3) When, in cell 1, a and b differ in number, then when ab pairs are exhausted (taken to cell 3), the residual object (a or b) in cell 2 will cause the system to run forever, by cyclically (and simultaneously) applying two of the rules on synapse $s_2((1, 2))$, that is, $\{r_{12}^1, r_{12}^3\}$ in case of a surplus a or $\{r_{12}^2, r_{12}^4\}$ in case of a surplus b.

Note in the last case, the rule applications like this case on synapse $s_2((1,2))$ also exist, but this is not relevant to the result of the overall computation.

The function of the system Π_1 is obvious now. It computes (accepts) the two-dimentional (natural number) vectors with the two components equal to each other, that is

$$\mathcal{P}s_v(\Pi_1) = \{(n, n) \mid n \geqslant 1\}$$

Note at least one copy of a is fetched in, so n is greater than 0.

The example shows that the parallelism of rules application on synapses is useful and effective in some sort of systems. In fact, more examples can be worked on, such as the mathematical computation and modeling of biological processes.

4 Computational capability

Due to the parallelism we place in our P system, we think that apparently the computational power of the P system defined in this paper is no less than that in [6], for what the latter can surely be computed by our P systems, simply through some technical approach to degrade parallel executing manner to sequential manner in rules applying, as sequentiality can be seen as a special case of parallelism in some sense. We maintain this as a possibly small open problem for future work and do not give the details here.

In our tissue-like P system, besides commincating rules by antiport, we can also add multiset rewriting rules to cells (inside)[1)]. Thus the system behavior can be extended to be able to describe more intricate procedures such as gene regulation in biology. But the design of the two kinds of rules should be careful enough to accommodate them cooperatively, that is, to avoid unexpected combination of rules executions. What the rules perform ought to be collaborating to achieve some defined function.

5 Discussions

Actually, the rules selecting algorithm in Section 2.2 can be improved a little. For example, we may choose some RC resulting in a smaller RC', then one can immediately argue that if we choose another equivalence class when choosing RC, it may generate a better RC′ from the viewpoint of the largest parallelism. That is, because we merely use the size metric of an equivalence class, the algorithm may violate the principle of largest parallelism a little. However, we still think that the mechanism we designed above is reasonable and concise in a sense. For instance, it saves much time in deciding which set of rules to be applied, which is critical if the system being modeled is of real-time type.

As a matter of fact, the improvement is not so complicated. It is described as below: After substep b) in step 1 (Section 2.2):

Do the same computation as that in substeps a), b) and c) of step 2 for each equivalence class RC_i obtained that time, and get RC_i'; choose RC_i' with the largest size in rules number, denoted as RC'; and execute the rules in RC' within one time step.

We can see that this improvement increases the time complexity of choosing RC'. The details of the computing complexity remain to be examined closely.

One would also argue that the parallelism on the synapses seems not so necessary, as the rules applicable simultaneously on one synapse can be merged into one rule, by merging the antiport rules, for example, the following two rules on the same synapse

$$(s_1, u_1/v_1, s_2),\ (s_1, u_2/v_2, s_2)$$

can be substituted by the rule

$$(s_1, u_1u_2/v_1v_2, s_2)$$

This is really an alternative way to handle the parallelism, but only in some specific cases, that is in general this is not possible (in the sense that the obtained system may probably not has the same function). And even the merging is possible in designing the system on some occasions, choosing not to merge all the rules as above appears more reasonable and readable, moreover, it can make the system more modular and clear for understanding the behavior. Hence, more often the merging is not possible to conserve the original function of the system, such as the system Π_1 given as an example in Section 3. So the algorithm in Section 2.2 is indeed of much practical sense.

In fact, our algorithm implicitly offers a way to implement the merging of rules suitable for parallel executing whenever possible. And if necessary, it can

1) This is not included in the P systems defined in this paper.

be used to minimize the P system under consideration (details remain to be analyzed), though such minimizing would probably sacrifice some intuition in semantics.

6 Future work

The work here is initial since it is developed only on the definition level, including some basic analysis and a simple example. More future work can be tried based on this paper, either on the computation mechanism or applications.

In comparison with the cell-like P systems with active membranes[7,8], why not introduce active links in our tissue-like P systems, that is, the channel topology can be altering dynamically during system running (like in [9]). This may introduce more intriguing and convenient computation elements, rendering related applications (such as modeling biological processes) more smoothly.

A more general topic is the dynamically membranes changing (forming or dissolving) and linking dynamics as described above (the readers are referred to population P systems[9] for more description). At least the design of states on links and/or states in cells provides some preliminary thoughts. This is a little far away from the work here, but the behavior of the new kind of P systems might be in more accordance with that of living cells in tissues.

We consider it necessary to work out more intricate and practical examples to illustrate the effectiveness and reflect in a sense the expressive capability of our P systems. Although the example in Section 3 is typical, in some applications more sophisticated design is needed. For instance, when used to model signaling pathway in biological processes, a much larger and complicated interaction graph should be depicted, analyzed and described at a certain abstract level, and the model can be of similar complexity as an highly-integrated circuit. We are currently working on this topic.

In Section 4, we merely give some informal explanation on the computation power of our P system, it is correct, but indirect. We plan to provide a more formal and direct proof on the computational power. And it is anticipated that a more concise and efficient construct can be found making full use of the parallelism on links. After that, some specific computation instances may be used to exemplify the power of the system.

References

1 Păun Gh. Computing with membranes. Journal of Computer and System Sciences, 2000, 61(1): 108—143

2 Păun Gh. From Cells to Computers: Computing with Membranes (P Systems). BioSystems, 2001, 59(3): 139—158

3 Păun Gh. Membrane Computing. An Introduction. Berlin: Springer, 2002

4 Freund R and Oswald M. A short note on analysing P systems with antiport rules. Bulletin of the EATCS, 2002, 78: 231—236

5 Martín-vide C, Păun Gh, Pazos J, et al. Tissue P systems. Theoretical Computer Science, 2003, 296(2): 295—326

6 Freund R, Păun Gh and Pérez-Jiménez MJ. Tissue-like P systems with channel states. Theoretical Computer Science, 2005, 330: 101—116

7 Alhazov A, Freund R and Păun Gh. P systems with active membranes and two polarizations. Technical Report 01/04 of Research Group on Natural Computing, Sevilla University, Spain, 2004, 20—36

8 Alhazov A and Ishdorj T. Membrane Operations in P Systems with Active Membranes. Second Brainstorming Week on Membrane Computing, Sevilla, Spain, February 2—7, 2004, 37—44

9 Bernardini F and Gheorghe M. Population P systems. Journal of Universal Computer Science, 2004, 10(5): 509—539

10 Păun Gh, Riscos-Núñez A, Romero-Jiménez A, et al. Technical Report 01/04 of Research Group on Natural Computing, Sevilla University, Spain, 2004