Journal of Universal Computer Science, vol. 10, no. 9 (2004), 1250-1271
submitted: 7/4/04, accepted: 9/6/04, appeared: 28/9/04 © J.UCS

Rewriting Tissue P Systems

Madhu Mutyam
(International Institute of Information Technology
Hyderabad - 500019, India
madhu_mutyam@iiit.net)

Vaka Jaya Prakash, Kamala Krithivasan
(Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai - 600036, India
prakash@cs.iitm.ernet.in, kamala@iitm.ernet.in)

Abstract: By considering string-objects and rewriting rules, we propose a variant of tissue P systems, namely, *rewriting tissue P systems*. We show the computational efficiency of rewriting tissue P systems by solving the Satisfiability and the Hamiltonian path problems in linear time. We study the computational capacity of rewriting tissue P systems and show that rewriting tissue P systems with at most *two* cells and *four* states are computationally universal. We also show the universality result of rewriting tissue P systems with at most *one* cell and *five* states. Finally we propose some new directions for future work.

Key Words: Tissue P systems, rewriting tissue P systems, computational universality, matrix grammars
Category: F.4.2, F.4.3

1 Introduction

P systems [Păun 2000], a field of current research, is motivated from the structure of the cell and the functioning of membranes. The three fundamental features of the cell which we use in this computing model are *membrane structure*, *objects*, and *evolution rules*. In a cell, objects can be considered as being *atomic*, as in the case of *P systems with symbol-objects*, or they can be associated with a structure, as in the case of DNA molecules, which can be described by a string. This leads one to consider *P systems with string-objects* [Păun 2000, Martín-Vide and Păun 2000, Ferretti et al. 2003]. Depending on the way in which string-objects being processed, P systems can be further divided into *rewriting P systems* [Dassow and Păun 1999, Păun 2000, Martín-Vide and Păun 2000] and *splicing P systems* [Păun and Yokomori 1999, Păun 2000]. We process string-objects in rewriting P systems with rules of the form $X \to v(tar)$, where $X \to v$ is a context-free rule and $tar \in \{here, in, out\}$ is a target indication specifying the region where the result of rewriting should go. All strings in a region are processed in parallel, but each single string is rewritten by only one rule. In other words, the parallelism is maximal at the level of strings and rules, but the rewriting is sequential at the level of the symbols from each string.

One common feature in all the variants of P systems is the tree-like representation of the membrane structure. A new way of looking at P systems was considered in *tissue P systems* [Martín-Vide et al. 2003], where the membrane structure was represented in a graph-like structure. Tissue P systems are based on the ideas of inter-cellular communication and the way the neurons cooperate and process impulses in a complex net established by *synapses* [Arbib 1978]. Tissue P systems consist of several *cells* which are related by protein channels. Each cell has a state from a given finite set and can process multisets of objects, represented by symbols from a given alphabet. The rules are of the form $sM \to s'M'$, where s, s' are states and M, M' are multisets of symbols. A single rule can be applied to one occurrence of M (the *minimal* mode) or to all possible occurrences of M (the *parallel* mode) or we can apply a maximal package of rules of the form $sM_i \to s'M_i'$, $1 \le i \le k$, that is involving the same states s and s', which can be applied to the current multiset (the *maximal* mode). The processed objects can be communicated to one of the neighboring cells (the *one* mode) or to all the neighboring cells (the *repl* mode) or to a subset of the neighboring cells (the *spread* mode).

In this paper, we propose *rewriting tissue P systems* by combining the ideas of tissue P systems and rewriting P systems. Instead of multisets of objects, here we consider string-objects and hence use rewriting rules, in particular, context-free rules, for processing string-objects. Like in tissue P systems, we consider *three* modes of processing string-objects and *three* modes for transmitting the processed strings, hence obtain *nine* possible ways of working with this model. Our model is computationally efficient in a sense that using this model we can solve the Satisfiability and the Hamiltonian path problems in linear time. From the computational capacity point of view, we show that our model, working in all possible modes, is computationally universal.

In the next section we give some language prerequisites. Section 3 defines our new model. We give algorithms for solving the Satisfiability problem and the Hamiltonian path problem in Section 4 and illustrate the algorithms with examples. We study the computational universality of our model in Section 5 and conclude the paper with a brief note in Section 6.

2 Prerequisites

Before proceeding to the main section of the paper, we recall the definition of matrix grammar.

A context-free *matrix* grammar with appearance checking is a 5-tuple $G = (N, T, S, M, F)$, where N and T are disjoint sets of nonterminals and terminals, respectively, $S \in N$ is the start symbol, M is a finite set of matrices, i.e., sequences of the form $(A_1 \to z_1, \ldots, A_n \to z_n)$, $n \geq 1$, of context-free rules, and F is a set of occurrences of rules in M.

For $w, z \in (N \cup T)^*$ we write $w \to z$ if there is a matrix $(A_1 \to x_1, \ldots, A_n \to x_n)$ in M and the strings $w_i \in (N \cup T)^*$, $1 \leq i \leq n+1$, such that $w = w_1$, $z = w_{n+1}$, and, for all i, $1 \leq i \leq n$, either (1) $w_i = w_i' A_i w_i''$, $w_{i+1} = w_i' x_i w_i''$, for some w_i', $w_i'' \in (N \cup T)^*$, or (2) $w_i = w_{i+1}$, A_i does not appear in w_i, and the rule $A_i \to x_i$ appears in F.

The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied. Thus, if a rule not in F is met, then it has to be used. If a rule from F is met, then we have two case: if it *can* be applied, then it *must* be applied; if it cannot be applied (the nonterminal from its left-hand side member is not present in the current string), then the rule may be skipped. If these conditions do not hold, the matrix is not applicable. If F is the empty set, a matrix grammar without appearance checking is presented.

We denote by MAT_{ac} the family of languages generated by matrix grammars with appearance checking. We omit the lower index *ac*, if we consider only matrix grammars without appearance checking.

A matrix grammar with appearance checking $G = (N, T, S, M, F)$ is said to be in *binary normal form*, if $N = N_1 \cup N_2 \cup \{S, \dagger\}$, with these three sets are mutually disjoint, and the matrices in M are in one of the following forms:

1. $(S \to XA)$, with $X \in N_1$, $A \in N_2$;

2. $(X \to Y,\ A \to x)$, with $X, Y \in N_1$, $A \in N_2$, $x \in (N_2 \cup T)^*$;

3. $(X \to Y,\ A \to \dagger)$, with $X, Y \in N_1$, $A \in N_2$;

4. $(X \to \lambda,\ A \to x)$, with $X \in N_1$, $A \in N_2$, $x \in T^*$.

Moreover, there is only one matrix of type 1 and F consists exactly all rules $A \to \dagger$ appearing in matrices of type 3; $\dagger$ is called a trap symbol, because once introduced, it is never removed. A matrix of type 4 is used only once, in the last step of the derivation.

According to [Dassow and Păun 1989], for each matrix grammar there is an equivalent matrix grammar in binary normal form. For an arbitrary matrix grammar $G = (N, T, S, M, F)$, let us denote by $ac(G)$, the cardinality of the set $\{A \in N \mid A \to \alpha \in F\}$. If a matrix grammar G is in binary normal form and $ac(G) \leq 2$, it is said to be in *strong binary normal form*. In [Freund and Păun 2001], it was proved that each recursively enumerable language can be generated by a matrix grammar G such that $ac(G) \leq 2$.

We now formally define rewriting tissue P systems.

3 Rewriting Tissue P Systems

A rewriting tissue P system, of degree m, $m \geq 1$, is a construct

$$\Pi = (V, T, \sigma_1, \sigma_2, ..., \sigma_m, syn, i_{out}),$$

where:

- V is the total alphabet of the system;
- $T \subseteq V$ is the terminal alphabet;
- $syn \subseteq \{1, ..., m\} \times \{1, ..., m\}$ (synapses among cells);
- $i_{out} \in \{1, ...m\}$ indicates the output cell;
- $\sigma_i, \cdots, \sigma_m$ are cells, of the form

  $$\sigma_i = (Q_i, s_{i,0}, L_{i,0}, P_i), 1 \leq i \leq m,$$

  where:

  - Q_i is a finite set of states;
  - $s_{i,0} \in Q_i$ is the initial state;
  - $L_{i,0} \in V^*$ is the initial set of strings;
  - P_i is a finite set of rules of the form $sX \rightarrow s'w(tar)$, where $s, s' \in Q_i$, $X \in (V - T)$, $w \in V^*$, and $tar \in \{here, go\}$ for $i \neq i_{out}$ and $tar \in \{here, go, out\}$ for $i = i_{out}$. When presenting the rules, the indication *here* is in general omitted.

Any m-tuple of the form $(s_1L_1, \cdots, s_mL_m)$, with $s_i \in Q_i$ and $L_i \in V^*$, $1 \leq i \leq m$, is called a *configuration* of Π; thus, $(s_{1,0}L_{1,0}, \cdots, s_{m,0}L_{m,0})$ is the *initial configuration* of Π.

We can change the configuration of the system by using rules from the sets P_i, $1 \leq i \leq m$. To define *transitions* among the configurations of the system, first we need to define various modes for both processing strings and transmitting the processed strings from a cell to another cell.

For processing strings, we define min, par, and max modes as follows:

$\underline{min\ mode}$:- We choose one string at random from all available strings in a cell and apply a possible rule from the cell on it. Application of the rule results in replacing only one occurrence of the symbol (the left-hand side member of the rule) with the right-hand side member of the rule and changing the state of the cell.

In a formal way,

$$sL_i \Rightarrow_{min} s'L_i' \text{ iff there is } x \in L_i \text{ and } sX \to s'w(tar) \in P_i \text{ such that}$$
$$x = x'Xx'' \text{ and } L_i' = (L_i - \{x\}) \cup \{x'wx''\}.$$

par mode:- We choose a rule from all available rules in a cell and apply the rule simultaneously to all possible strings in the cell. Simultaneous application of the rule results in replacing only one occurrence of the left-hand side member of the rule with the right-hand side member of the rule in *all possible* strings. As usual, the state change will occur.

That is,

$$\begin{aligned} sL_i \Rightarrow_{par} s'L_i' \text{ iff there is } & sX \to s'w(tar) \in P_i \text{ such that} \\ & L_i' = (L_i - \{x_1, x_2, \cdots, x_k\}) \cup \{y_1, y_2, \cdots, y_k\}, \text{ where} \\ & \{x_1, x_2, \cdots, x_k\} \subseteq L_i, x_j = x_j'Xx_j'', y_j = x_j'wx_j'', 1 \le j \le k, \\ & \text{and if } x \in L_i - \{x_1, x_2, \cdots, x_k\}, \text{ then } |x|_X = 0. \end{aligned}$$

max mode:- We choose a set of rules which use the current state of the cell and introduce the same new state after processing strings and apply the set of rules on all possible strings present in the cell. Here each string is being processed by one rule from the set, and moreover only one occurrence of the symbol from left-hand side of the rule is processed in each string.

Formally,

$$\begin{aligned} sL_i \Rightarrow_{max} s'L_i' \text{ iff } & L_i' = (L_i - \{x_1, x_2, \cdots, x_k\}) \cup \{y_1, y_2, \cdots, y_k\}, \text{ where} \\ & \{x_1, x_2, \cdots, x_k\} \subseteq L_i, \text{ there are the rules} \\ & sX_1 \to s'w_1(tar_1), \cdots, sX_k \to s'w_k(tar_k) \in P_i, \text{ such that} \\ & x_j = x_j'Xx_j'', y_j = x_j'wx_j'', 1 \le j \le k, \text{ and if} \\ & x \in L_i - \{x_1, x_2, \cdots, x_k\}, \text{ then } |x|_X = 0 \text{ for all rules} \\ & sX \to s'z(tar) \in P_i. \end{aligned}$$

Based on the tar value, processed strings can be remained in the same cell (if $tar = here$) or sent to other cells (if $tar = go$) or environment (if $tar = out$). Rules with $tar = out$ can be applied only in the output cell, whereas rules with $tar = \{here, go\}$ can be applied in any cell. Since a cell can be connected to many other cells, we consider three modes, i.e., *repl*, *one*, and *spread*, for transmitting strings which are obtained by using rules with $tar = go$ from one cell to other cells.

For a language $L_i \in V^*$, we denote by $L_{i,no}$ the set of strings which are not processed by any rule from P_i, $L_{i,here}$ the set of strings which are processed by rules with $tar = here$ from P_i, $L_{i,go}$ the set of strings which are processed by rules with $tar = go$ from P_i, and $L_{i,out}$ the set of strings which are processed by rules with $tar = out$ from P_i. If cell i is not the output cell, then $L_{i,out} = \emptyset$. Furthermore, for a node i in the graph defined by syn we denote the ancestor of node i as $anc(i) = \{j \mid (j,i) \in syn\}$.

We now define the transition among the configurations of the system Π as follows:

Let $C_1 = (s_1L_1, \cdots, s_mL_m)$ and $C_2 = (s'_1L''_1, \cdots, s'_mL''_m)$ be any two configurations of Π. We then write $C_1 \Rightarrow_{\alpha,\beta} C_2$, for $\alpha \in \{min, par, max\}$, $\beta \in \{repl, one, spread\}$, if there are finite languages $L'_1, \cdots, L'_m$ over V such that

$$s_iL_i \Rightarrow_\alpha s'_iL'_i, 1 \le i \le m, \text{ where each } L'_i = L_{i,no} \cup L_{i,here} \cup L_{i,go} \cup L_{i,out}$$

and

– for $\beta = repl$ we have:

$$L''_i = L_{i,no} \cup L_{i,here} \cup \bigcup_{j \in anc(i)} L_{j,go};$$

– for $\beta = one$ we have:

$$L''_i = L_{i,no} \cup L_{i,here} \cup \bigcup_{j \in I_i} L_{j,go},$$

where $I_i \subseteq anc(i)$ such that the set $anc(i)$ was partitioned into $I_1, \cdots, I_m$; at this transition, all non-empty sets of strings of the form $\bigcup_{j \in I_k} L_{j,go}$, $1 \le k \le m$, should be sent to receiving cells (added to L''_l, $1 \le l \le m$);

– for $\beta = spread$ we have:

$$L''_i = L_{i,no} \cup L_{i,here} \cup L^1_i,$$

where $L^1_i \subseteq \bigcup_{j \in anc(i)} L_{j,go}$ such that $L^1_1, \cdots, L^1_m$ are sets with the property $\bigcup_{j=1}^m L^1_j = \bigcup_{j \in anc(i)} L_{j,go}$, and such that all $L^1_1, \cdots, L^1_m$ are sent to receiving cells (added to L''_l, $1 \le l \le m$);

Note that in the case of the output cell we also remove from L'_i all strings from the set $L_{i,out}$.

If there is only one string in a cell, all the three modes of processing strings are same. Similarly, If the system has at most two cells, all the three modes of transmitting the processed strings from a cell to other cells are same.

During any transition, some cells can do nothing: if no rule is applicable to the available strings in the current state, a cell waits until it gets new strings from its ancestor cells. We assume that each transition lasts one time unit so that the work of the net is synchronized, and the same clock marks the time for all cells.

A sequence of transitions among configurations of the rewriting tissue P system Π is called a *computation* of Π. A computation is said to be successful if it ends in an *halting configuration*, i.e., a configuration where no rule in any cell of the system can be used. The output of a computation is the set of all strings which are ejected from the output cell to the environment. But we consider the output of a successful computation. The set of all strings over T from the output of a successful computation is said to be the language generated by the rewriting tissue P system. We denote by $L_{\alpha,\beta}(\Pi)$, $\alpha \in \{min, par, max\}$, $\beta \in \{repl, one, spread\}$, the language generated by a rewriting tissue P system Π in the (α, β) mode. The family of all languages generated by rewriting tissue P systems with at most m, $m \geq 1$, cells, each of which using at most r, $r \geq 1$, states, is denoted by $RtP_{m,r}(\alpha, \beta)$. When one of the parameters m, r is not bounded, replace it with $*$.

We now discuss the computational efficiency of our model by giving linear time algorithms for solving two NP-complete problems.

4 Solving NP-Complete Problems

We consider two well-known NP-complete problems, i.e., the *Satisfiability problem* and the *Hamiltonian path problem*, and give linear time algorithms for solving them. In order to solve these problems, we do some pre-processing in terms of building the system. In other words, we construct the system which is specific to the given instance of the problem in polynomial time. We then run the algorithm on the constructed system and check whether or not the problem has a solution in linear time. The problem has a solution if and only if at least one string is sent out of the system. We also illustrate the algorithms with examples.

4.1 Solving the Satisfiability Problem

The Satisfiability problem is to check whether or not a given propositional formula in the conjunctive normal form is satisfiable.

A propositional formula C is in the conjunctive normal form if it is a conjunction of disjunctions, i.e.,

$$C = C_1 \wedge C_2 \wedge \cdots \wedge C_m,$$

where each C_i, $1 \leq i \leq m$, is a *clause* of the form

$$C_i = y_{i,1} \vee y_{i,2} \vee \cdots \vee y_{i,p_i},$$

with each literal y_j being either a propositional variable, x_s, or its negation, $\overline{x_s}$.

Theorem 1. *The Satisfiability problem can be solved in linear time with respect to the number of variables and the number of clauses by a rewriting tissue P system working in the (max, repl) mode.*

Proof. Let us consider a propositional formula $C = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, with $C_i = y_{i,1} \vee y_{i,2} \vee \cdots \vee y_{i,p_i}$, for some $m \geq 1, p_i \geq 1$, and $y_{i,j} \in \{x_k, \overline{x_k} \mid 1 \leq k \leq n\}$, $1 \leq i \leq m$, $1 \leq j \leq p_i$.

We construct the rewriting tissue P system

$$\Pi = (V, \emptyset, \sigma_1, \cdots, \sigma_{2n+m}, syn, 2n+m),$$

where:

- $V = \{x_i, \overline{x_i}, T_i, F_i \mid 1 \leq i \leq n\}$;
- $syn = \{(2i-1, 2i+1), (2i, 2i+1), (2i-1, 2i+2), (2i, 2i+2) \mid 1 \leq i \leq n-1\} \cup \{(2n-1, 2n+1), (2n, 2n+1)\} \cup \{(i, i+1) \mid 2n+1 \leq i \leq 2n+m-1\}$;
- Each $\sigma_i = (\{s\}, s, L_i, P_i)$, $1 \leq i \leq 2n+m$, where:
  - $L_1 = L_2 = \{x_1 x_2 \cdots x_n \overline{x_1}\ \overline{x_2} \cdots \overline{x_n}\}$;
  - $L_i = \emptyset$, $3 \leq i \leq 2n+m$;
  - $P_{2i-1} = \{s\overline{x_i} \rightarrow sF_i(go)\}$, $1 \leq i \leq n$;
  - $P_{2i} = \{sx_i \rightarrow sT_i(go)\}$, $1 \leq i \leq n$;
  - $P_{2n+i} = \{sT_j \rightarrow sT_j(go) \mid x_j \in C_i, 1 \leq j \leq n\} \cup \{sF_j \rightarrow sF_j(go) \mid \overline{x_j} \in C_i, 1 \leq j \leq n\}$, $1 \leq i \leq m-1$;
  - $P_{2n+m} = \{sT_j \rightarrow sT_j(out) \mid x_j \in C_m, 1 \leq j \leq n\} \cup \{sF_j \rightarrow sF_j(out) \mid \overline{x_j} \in C_m, 1 \leq j \leq n\}$.

The system works as follows:

The system has $2n+m$ cells and all these cells are connected as shown in Figure 1. The system works in two phases, i.e., generating phase and checking phase. In generating phase, distinct strings will be generated with all possible truth assignments. Cells from σ_1 to σ_{2n} are involved in the generating phase. In checking phase, each cell σ_{2n+i}, $1 \leq i \leq m$, checks whether or not the truth assignments in strings of σ_{2n+i} satisfy the clause C_i.

In the initial configuration, the system has a string $x_1 x_2 \cdots x_n \overline{x_1} \cdots \overline{x_n}$ in both σ_1 and σ_2. In the next step, both σ_3 and σ_4 get two strings, i.e., $T_1 x_2 \cdots x_n \overline{x_1}\ \overline{x_2} \cdots \overline{x_n}$ and $x_1 x_2 \cdots x_n F_1 \overline{x_2} \cdots \overline{x_n}$, from their ancestor cells. Since the system is working in the $(max, repl)$ mode and each cell σ_i, $3 \leq i \leq 2n+1$, has two ancestor cells, after n successive configurations, σ_{2n+1} gets 2^n distinct

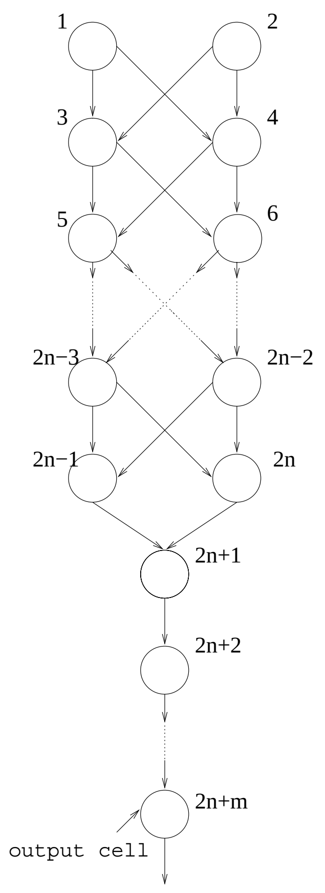

Figure 1: SAT network

strings each of which represents one of the 2^n possible n-variable truth assignments. Cells involved in the checking phase filter all available strings based on the truth assignments associated with each string. So, σ_{2n+1} sends only those strings which correspond to the truth assignments satisfying the clause C_1 to σ_{2n+2}. Similarly σ_{2n+2} sends only those strings which correspond to the truth assignments satisfying the clause C_2 to σ_{2n+3}. Thus the truth assignments of the strings present in σ_{2n+3} satisfy both C_1 and C_2. In this way, the filtering process continues for cells from σ_{2n+3} to σ_{2n+m}. It is clear that the strings present in a cell σ_{2n+i}, $2 \leq i \leq m$, correspond to the truth assignments satisfying $C_1 \wedge C_2 \cdots \wedge C_{i-1}$. Hence, the strings come out of σ_{2n+m} correspond to the truth assignments satisfying $C_1 \wedge C_2 \wedge \cdots \wedge C_m$. Thus, the propositional formula is found to be satisfiable if and only if the output cell sends at least one string to the environment.

Time Complexity: Our algorithm takes $(n+m)$ steps to determine whether or not a given propositional formula that contains n variables and m clauses is satisfiable. Note that here we excluded the time for constructing the system. $\square$

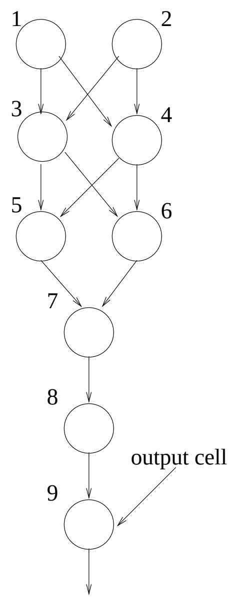

Figure 2: SAT network for the example

Example 1. Checking whether or not the following propositional formula γ is satisfiable.

$$\gamma = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}).$$

We construct the rewriting tissue P system

$$\Pi = (V, \emptyset, \sigma_1, \cdots, \sigma_9, syn, 9),$$

where:

– $V = \{x_i, \overline{x_i}, T_i, F_i \mid 1 \leq i \leq 3\}$;

– syn is as shown in Figure 2;

– Each $\sigma_i = (\{s\}, s, L_i, P_i)$, $1 \leq i \leq 9$, where:

- $L_1 = L_2 = \{x_1x_2x_3\overline{x_1}\ \overline{x_2}\ \overline{x_3}\}$;
- $L_i = \emptyset$, $3 \leq i \leq 9$;
- Each P_i, $1 \leq i \leq 9$, contains the rules as given in the above theorem.

step 0 : $(s\{x_1x_2x_3\overline{x_1}\ \overline{x_2}\ \overline{x_3}\}, s\{x_1x_2x_3\overline{x_1}\ \overline{x_2}\ \overline{x_3}\}, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset)$;

step 1 : $(s\emptyset, s\emptyset, s\{x_1x_2x_3F_1\overline{x_2}\ \overline{x_3}, T_1x_2x_3\overline{x_1}\ \overline{x_2}\ \overline{x_3}\}, s\{x_1x_2x_3F_1\overline{x_2}\ \overline{x_3},$
$T_1x_2x_3\overline{x_1}\ \overline{x_2}\ \overline{x_3}\}, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset)$;

step 2 : $(s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\{x_1x_2x_3F_1F_2\overline{x_3}, T_1x_2x_3\overline{x_1}F_2\ \overline{x_3}, x_1T_2x_3F_1\overline{x_2}\ \overline{x_3},$
$T_1T_2x_3\overline{x_1}\ \overline{x_2}\ \overline{x_3}\}, s\{x_1x_2x_3F_1F_2\overline{x_3}, T_1x_2x_3\overline{x_1}F_2\ \overline{x_3}, x_1T_2x_3F_1\overline{x_2}\ \overline{x_3},$
$T_1T_2x_3\overline{x_1}\ \overline{x_2}\ \overline{x_3}\}, s\emptyset, s\emptyset, s\emptyset)$;

step 3 : $(s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\{x_1x_2x_3F_1F_2F_3, T_1x_2x_3\overline{x_1}F_2F_3, x_1T_2x_3F_1\overline{x_2}F_3,$
$T_1T_2x_3\overline{x_1}\ \overline{x_2}F_3, x_1x_2T_3F_1F_2\overline{x_3}, T_1x_2T_3\overline{x_1}F_2\ \overline{x_3}, x_1T_2T_3F_1\overline{x_2}\ \overline{x_3},$
$T_1T_2T_3\overline{x_1}\ \overline{x_2}\ \overline{x_3}\}, s\emptyset, s\emptyset)$;

step 4 : $(s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\{x_1x_2x_3F_1F_2F_3\}, s\{T_1x_2x_3\overline{x_1}F_2F_3, x_1T_2x_3F_1\overline{x_2}F_3,$
$T_1T_2x_3\overline{x_1}\ \overline{x_2}F_3, T_1x_2T_3\overline{x_1}F_2\ \overline{x_3}, x_1x_2T_3F_1F_2\overline{x_3}, x_1T_2T_3F_1\overline{x_2}\ \overline{x_3},$
$T_1T_2T_3\overline{x_1}\ \overline{x_2}\ \overline{x_3}\}, s\emptyset)$;

step 5 : $(s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\{x_1x_2x_3F_1F_2F_3\}, s\{x_1T_2x_3F_1\overline{x_2}F_3\},$
$s\{T_1x_2x_3\overline{x_1}F_2F_3, T_1T_2x_3\overline{x_1}\ \overline{x_2}F_3, T_1x_2T_3\overline{x_1}F_2\ \overline{x_3}, x_1x_2T_3F_1F_2\overline{x_3},$
$x_1T_2T_3F_1\overline{x_2}\ \overline{x_3}, T_1T_2T_3\overline{x_1}\ \overline{x_2}\ \overline{x_3}\})$;

step 6 : $(s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\emptyset, s\{x_1x_2x_3F_1F_2F_3\}, s\{x_1T_2x_3F_1\overline{x_2}F_3\},$
$s\{T_1x_2T_3\overline{x_1}F_2\ \overline{x_3}\})T_1x_2x_3\overline{x_1}F_2F_3, T_1T_2x_3\overline{x_1}\ \overline{x_2}F_3, x_1x_2T_3F_1F_2\overline{x_3},$
$x_1T_2T_3F_1\overline{x_2}\ \overline{x_3}, T_1T_2T_3\overline{x_1}\ \overline{x_2}\ \overline{x_3}$;

The ejected strings (from the output cell) indicate that the given formula is satisfiable for the truth assignments TFF, TTF, FFT, FTT, and TTT.

4.2 Solving the Hamiltonian Path Problem

Given an undirected graph $G = (U, E)$ with $|U| = n$, $n \geq 2$, where U is the set of nodes and E, the set of edges, the Hamiltonian path problem is to determine whether or not there exists a path that passes through all the nodes in U exactly once. The Hamiltonian path problem for undirected graphs is known to be NP-complete. We solve this problem in linear time with respect to the number of nodes of a graph by using rewriting tissue P systems.

Theorem 2. *The Hamiltonian path problem for undirected graphs can be solved in linear time with respect to the number of nodes by a rewriting tissue P system working in the* (α, *repl*) *mode, where* $\alpha \in \{par, max\}$.

Proof. Let $G = (U, E)$ be an undirected graph with $|U| = n$, $n \geq 2$. Let $U = \{a_1, \cdots, a_n\}$. We construct the rewriting tissue P system

$$\Pi = (V, \emptyset, \sigma_1, \cdots, \sigma_{2n}, syn, 2n),$$

where:

- $V = \{a_i, h_i \mid 1 \leq i \leq n\}$;
- $syn = \{(i,j), (j,i) \mid (a_i, a_j) \in E\} \cup \{(i, n+1) \mid 1 \leq i \leq n\} \cup \{(n+i, n+i+1) \mid 1 \leq i \leq n-1\}$;
- Each $\sigma_i = (\{s\}, s, L_i, P_i)$, $1 \leq i \leq 2n$, where:
  - $L_i = \{a_1 a_2 \cdots a_n\}$, $1 \leq i \leq n$;
  - $L_{n+i} = \emptyset$, $1 \leq i \leq n$;
  - $P_i = \{s a_i \rightarrow s h_i(go)\}$, $1 \leq i \leq n$;
  - $P_{n+i} = \{s h_i \rightarrow s h_i(go)\}$, $1 \leq i \leq n-1$;
  - $P_{2n} = \{s h_n \rightarrow s h_n(out)\}$.

We constructed the system to check whether or not there exists an Hamiltonian path in a given graph of n nodes. The system has $2n$ cells and all these cells are connected as shown in Figure 3. Cells from the set $\{\sigma_1, \cdots, \sigma_n\}$ are connected based on the connectivity of all n nodes of the given graph. All these cells are also connected to σ_{n+1} so that any processed string from these cells can also be sent to σ_{n+1}. In the initial configuration, each cell from the set $\{\sigma_1, \cdots, \sigma_n\}$ contains a string of the form $a_1 a_2 \cdots a_n$. From the set of rules given in the algorithm, it is clear that any cell σ_i, $1 \leq i \leq n$, can send a string x such that $|x|_{a_i} = 1$ to all its ancestor cells by replacing a_i with h_i. Generally σ_{n+1} gets strings over $\{a_i, h_i \mid 1 \leq i \leq n\}$. A string from σ_{n+1} can be sent out from the output cell (i.e., σ_{2n}) via $\sigma_{n+2}, \cdots, \sigma_{2n-1}$, if and only if it is of the form $h_1 h_2 \cdots h_n$. That means, if a string in σ_{n+1} contains at least one symbol from the set $\{a_1, \cdots, a_n\}$, the string cannot be sent out from the output cell. σ_{n+1} gets a string of the form $h_1 h_2 \cdots h_n$ (after $(n+1)^{th}$ configuration) from one or more of the cells from the set $\{\sigma_1, \cdots, \sigma_n\}$ if and only if all the cells from the set $\{\sigma_1, \cdots, \sigma_n\}$ are connected. But we know that the cells from the set $\{\sigma_1, \cdots, \sigma_n\}$ are connected in the same way as the nodes of the given graph are connected. This indicates that σ_{n+1} gets a string of the form $h_1 h_2 \cdots h_n$ if and only if all the

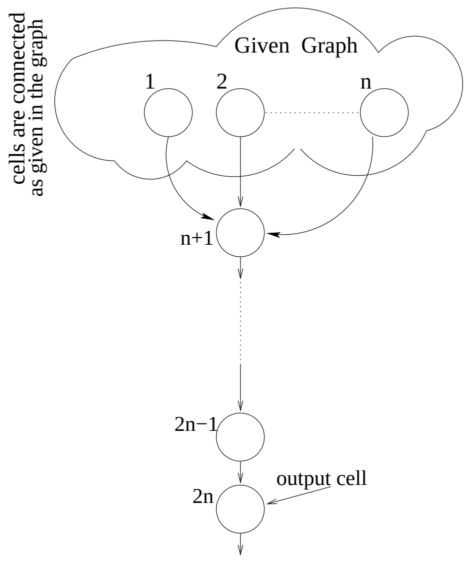

Figure 3: HPP network

nodes in the graph are connected. In other words, a string is sent out from the output cell if and only if there exists an Hamiltonian path in the given graph.

Time Complexity: Our algorithm takes $2n$ steps to determine whether or not there exits an Hamiltonian path in a given graph that contains n nodes. Like in the previous theorem, we are not considering the pre-processing cost. □

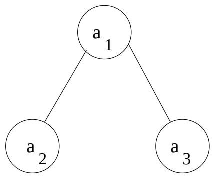

Figure 4: Example graph

Example 2. Checking whether or not the graph as shown in Figure 4 contains an Hamiltonian path.

We construct the rewriting tissue P system

$$\Pi = (V, \emptyset, \sigma_1, \cdots, \sigma_6, syn, 6),$$

where:

- $V = \{a_i, h_i \mid 1 \leq i \leq 3\}$;
- syn is as shown in Figure 5;
- Each $\sigma_i = (\{s\}, s, L_i, P_i)$, $1 \leq i \leq 6$, where:
  - $L_1 = L_2 = L_3 = \{a_1a_2a_3\}$;
  - $L_4 = L_5 = L_6 = \emptyset$;
  - Each P_i, $1 \leq i \leq 6$, contains the rules as given in the above theorem.

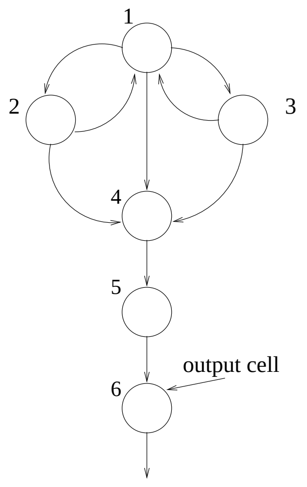

Figure 5: HPP network for the example graph

step 0 : $(s\{a_1a_2a_3\}, s\{a_1a_2a_3\}, s\{a_1a_2a_3\}, s\emptyset, s\emptyset, s\emptyset)$;
step 1 : $(s\{a_1h_2a_3, a_1a_2h_3\}, s\{h_1a_2a_3\}, s\{h_1a_2a_3\}, s\{h_1a_2a_3, a_1h_2a_3, a_1a_2h_3\},$
$s\emptyset, s\emptyset)$;
step 2 : $(s\{h_1h_2a_3, h_1a_2h_3\}, s\{h_1h_2a_3, h_1a_2h_3\}, s\{h_1h_2a_3, h_1a_2h_3\},$

$$\begin{aligned}
&s\{h_1h_2a_3, h_1a_2h_3, a_1h_2a_3, a_1a_2h_3\}, s\{h_1a_2a_3\}, s\emptyset);\\
\text{step } 3 : &(s\{h_1h_2h_3, h_1h_2h_3, h_1h_2a_3, h_1a_2h_3\}, s\{h_1h_2a_3\}, s\{h_1a_2h_3\},\\
&s\{h_1h_2h_3, h_1h_2h_3, a_1h_2a_3, a_1a_2h_3\}, s\{h_1h_2a_3, h_1a_2h_3, h_1a_2a_3\}, s\emptyset);\\
\text{step } 4 : &(s\{h_1h_2h_3, h_1h_2h_3, h_1h_2a_3, h_1a_2h_3\}, s\{h_1h_2a_3\}, s\{h_1a_2h_3\},\\
&s\{a_1h_2a_3, a_1a_2h_3\}, s\{h_1h_2h_3, h_1h_2h_3, h_1a_2h_3, h_1a_2a_3\}, s\{h_1h_2a_3\});\\
\text{step } 5 : &(s\{h_1h_2h_3, h_1h_2h_3, h_1h_2a_3, h_1a_2h_3\}, s\{h_1h_2a_3\}, s\{h_1a_2h_3\},\\
&s\{a_1h_2a_3, a_1a_2h_3\}, s\{h_1a_2h_3, h_1a_2a_3\}, s\{h_1h_2h_3, h_1h_2h_3, h_1h_2a_3\});\\
\text{step } 6 : &(s\{h_1h_2h_3, h_1h_2h_3, h_1h_2a_3, h_1a_2h_3\}, s\{h_1h_2a_3\}, s\{h_1a_2h_3\},\\
&s\{a_1h_2a_3, a_1a_2h_3\}, s\{h_1a_2h_3, h_1a_2a_3\}, s\{h_1h_2a_3\})h_1h_2h_3, h_1h_2h_3;
\end{aligned}$$

Two strings are ejected from the output cell. This indicates that the graph contains *two* Hamiltonian paths.

Having discussed the computational efficiency, we now discuss the computational capacity of our system.

5 Computational Universality

In this section we study the computational universality of our variant. First we consider rewriting tissue P systems working in the maximal mode and show the universality result. Later we obtain the universality of rewriting tissue P systems working in the minimal mode. From the proof of the universality theorem of rewriting tissue P systems working in the minimal mode, we show that rewriting tissue P systems working in parallel mode are computationally universal.

Henceforth, unless otherwise stated, we denote by $G = (N, T, S, M, F)$ a matrix grammar with appearance checking, in strong binary normal form, with rules of the four forms mentioned in Section 2, and with $ac(G) = 2$. The nonterminal alphabet N is of the form $N = N_1 \cup N_2 \cup \{S, \dagger\}$. Let $B^{(1)}$ and $B^{(2)}$ be the two symbols in N_2 for which we have rules $B^{(j)} \to \dagger$, $j \in \{1, 2\}$, in the matrices of M. Let us assume that we have

- k matrices of the form $m_i : (X \to \alpha, A \to x), X \in N_1, A \in N_2, \alpha \in N_1 \cup \{\lambda\}$, and $x \in (N_2 \cup T)^*, 1 \leq i \leq k$;
- h matrices of the form $m'_i : (X \to Y, B^{(j)} \to \dagger), X, Y \in N_1, j \in \{1, 2\}$; these matrices are labeled by m'_i, with $i \in lab_j$, for $j \in \{1, 2\}$, such that lab_1, lab_2, and $lab_0 = \{1, \cdots, k\}$ are mutually disjoint sets.

Theorem 3. *$RE = RtP_{1,5}(max, \gamma)$, for $\gamma \in \{repl, one, spread\}$.*

Proof. We prove the inclusion $RE \subseteq RtP_{1,5}(max, \gamma)$, $\gamma \in \{repl, one, spread\}$, the reverse inclusion can be obtained in a straightforward manner. We start by considering a matrix grammar $G = (N, T, S, M, F)$ in strong binary normal form, and construct the rewriting tissue P system

$$\Pi = (V, T, \sigma_1, \emptyset, 1),$$

where:

- $V = N_1 \cup N_2 \cup \{X' \mid X \in N_1\} \cup \{Z\} \cup \{X_i, A_i \mid X \in N_1, A \in N_2, 0 \leq i \leq k+1\}$;
- $\sigma_1 = (\{s, s_1, s_2, s^{(1)}, s^{(2)}\}, s, \{X, A\}, P_1)$, where P_1 contains the following rules:
  1. $sX \rightarrow s_1X_0$ and $sA \rightarrow s_1A_0, X \in N_1, A \in N_2$;
  2. $s_1X_j \rightarrow s_1X_{j+1}$ and $s_1A_j \rightarrow s_1A_{j+1}, 0 \leq j \leq k$;
  3. $s_1X_i \rightarrow s_2Y'$ and $s_1A_i \rightarrow s_2x$, for a matrix $m_i : (X \rightarrow Y, A \rightarrow x)$;
  4. $s_2Y' \rightarrow sY$;
  5. $s_2A_i \rightarrow sZ(out), A \in N_2, 1 \leq i \leq k$;
  6. $s_2X_i \rightarrow sZ(out), X \in N_1, 1 \leq i \leq k$;
  7. $s_1X_i \rightarrow s$ and $s_1A_i \rightarrow sx(out)$, for a matrix $m_i : (X \rightarrow \lambda, A \rightarrow x)$;
  8. $sX \rightarrow s^{(j)}Y'$, for a matrix $m'_i : (X \rightarrow Y, B^{(j)} \rightarrow \dagger)$;
  9. $s^{(j)}Y' \rightarrow sY$ and $s^{(j)}B^{(j)} \rightarrow sZ(out)$, for a matrix $m'_i : (X \rightarrow Y, B^{(j)} \rightarrow \dagger)$;
  10. $s_1X_{k+1} \rightarrow s_1Z(out), X \in N_1$;
  11. $s_1A_{k+1} \rightarrow s_1Z(out), A \in N_2$.

The system works as follows:

The system has only one cell. Since it has only one cell, all the three modes of transmitting the processed strings from a cell to other cells are same. Initial configuration of the system is $(s\{X, A\})$. Assume that at a particular instant the configuration of the system is $(s\{X, w_1Aw_2\})$, where $w_1, w_2 \in (N_2 \cup T)^*$. We apply two rules $sX \rightarrow s_1X_0$ and $sA \rightarrow s_1A_0$ on two strings X and w_1Aw_2, respectively, in parallel, so that the configuration of the system is changed to $(s_1\{X_0, w_1A_0w_2\})$. We now increment the subscripts of both X and A simultaneously. Let the configuration of the system be $(s\{X_i, w_1A_iw_2\})$. In order to simulate a matrix $m_i : (X \rightarrow Y, A \rightarrow x)$, we can apply the rules $s_1X_i \rightarrow s_2Y'$ and $s_1A_i \rightarrow s_2x$. If we apply only one of these two rules, a trap symbol Z is introduced in the next step. On the other hand, if we apply both the rules simultaneously, X_i and A_i are replaced with Y' and x, respectively, and the state of the cell becomes s_2. In the next step, Y' is replaced with Y and the state changes to s. Now the configuration of the system becomes $(s\{Y, w_1xw_2\})$ which represents the correct simulation of the matrix. The same procedure is applied for simulating a matrix $m_i : (X \rightarrow \lambda, A \rightarrow x)$, except the last step. In the last step, we apply the rules $s_1X_i \rightarrow s$ and $s_1A_i \rightarrow sx(out)$ on X_i and A_i, respectively, so that the configuration of the system becomes $(s\{\lambda\})$ and the string w_1xw_2 is sent to the environment. If w_1xw_2 contains any nonterminal, it is not considered as a result.

Let $(s\{X, w\})$ be the configuration of the system. For simulating a matrix $m'_i : (X \rightarrow Y, B^{(1)} \rightarrow \dagger)$, we apply the rule $sX \rightarrow s^{(1)}Y'$ so that the configuration of the system becomes $(s^{(1)}\{Y', w\})$. If w contains $B^{(1)}$, a trap symbol Z is introduced. Otherwise, Y' is replaced with Y and the configuration of the system becomes $(s\{Y, w\})$ which represents the correct simulation of the matrix. Similar procedure can be given for simulating a matrix $m'_i : (X \rightarrow Y, B^{(2)} \rightarrow \dagger)$.

The procedure for simulating a matrix of type 2 (type 3) can be iterated for all the matrices of type 2 (type 3). Finally the simulation process ends by simulating a type 4 matrix. So any derivation in G can be simulated by a computation in Π and, conversely, the result of successful computations in Π corresponds to the terminal derivations in G. Thus, the equality $L(\Pi) = L(G)$ follows. □

We now show that the universality of rewriting tissue P systems can be achieved with only four states but at the cost of one extra cell.

Theorem 4. *$RE = RtP_{2,4}(max, \gamma)$, for $\gamma \in \{repl, one, spread\}$.*

Proof. In order to prove the theorem we make use of a matrix grammar $G = (N, T, S, M, F)$ in strong binary normal form. We now construct the rewriting tissue P system

$$\Pi = (V, T, \sigma_1, \sigma_2, \{(1, 2), (2, 1)\}, 1),$$

where:

- $V = N_1 \cup N_2 \cup \{X' \mid X \in N_1\} \cup \{E, Z\} \cup \{X_i, A_i \mid X \in N_1, A \in N_2, 0 \leq i \leq k + 1\}$;
- $\sigma_1 = (\{s, s_1, s_2, s_3\}, s, \{X, A\}, P_1)$, where P_1 contains the following rules:
  1. $sX \rightarrow s_1X_0$ and $sA \rightarrow s_1A_0, X \in N_1, A \in N_2$;
  2. $s_1X_j \rightarrow s_1X_{j+1}$ and $s_1A_j \rightarrow s_1A_{j+1}, 0 \leq j \leq k$;
  3. $s_1X_i \rightarrow s_2Y'$ and $s_1A_i \rightarrow s_2x$, for a matrix $m_i : (X \rightarrow Y, A \rightarrow x)$;
  4. $s_2Y' \rightarrow sY$;
  5. $s_2A_i \rightarrow sZ(out), A \in N_2, 1 \leq i \leq k$;

6. $s_2X_i \to sZ(out), X \in N_1, 1 \leq i \leq k$;
7. $s_1X_i \to s$ and $s_1A_i \to sx(out)$, for a matrix $m_i : (X \to \lambda, A \to x)$;
8. $sX \to s_3Y^{(j)}(go)$ and $sA \to s_3A(go)$, $A \in N_2$, for a matrix $m'_i : (X \to Y, B^{(j)} \to \dagger)$;
9. $s_3E \to s$ and $s_3Z \to sZ(out)$;
10. $s_1X_{k+1} \to s_1Z(out), X \in N_1$;
11. $s_1A_{k+1} \to s_1Z(out), A \in N_2$;
12. $s_3X \to s_3Z(out), X \in N_1$;

– $\sigma_2 = (\{s, s_1, s_2, s_3\}, s, \emptyset, P_2)$, where P_2 contains the following rules:
  1. $sY^{(j)} \to s_jY'$, for a matrix $m'_i : (X \to Y, B^{(j)} \to \dagger)$;
  2. $s_jY' \to s_3Y$ and $s_jB^{(j)} \to s_3Z$, for a matrix $m'_i : (X \to Y, B^{(j)} \to \dagger)$;
  3. $s_3A \to sA(go), A \in N_2$;
  4. $s_3Y \to sYE(go), Y \in N_1$.

The system works as follows:

Here the system has two cells σ_1 and σ_2. We know that if the system has at most two cells, all the three modes of transmitting the processed strings from a cell to other cells are same. Initial configuration of the system is $(s\{X, A\}, s\emptyset)$. The procedure for simulating matrices of type 2 and type 4 is same as the one given in the proof of Theorem 3. Let us suppose that at a particular instant, the configuration of the system is $(s\{X, w\}, s\emptyset)$, where $w \in (N_2 \cup T)^*$.

For simulating a matrix $m'_i : (X \to Y, B^{(1)} \to \dagger)$, we apply the rules $sX \to s_3Y^{(i)}(go)$ and $sA \to s_3A(go)$ so that the configuration of the system becomes $(s_3\emptyset, s\{Y^{(i)}, w\})$. We now apply the rule $sY^{(1)} \to s_1Y'$ and change the configuration of the system to $(s_3\emptyset, s_1\{Y', w\})$. In the next step, we replace Y' with Y and change the state to s_3. At the same time, if w contains $B^{(1)}$, a trap symbol Z is introduced. Otherwise, in the next step we apply the rules $s_3A \to sA(go)$ and $s_3Y \to sYE(go)$ so that the configuration of the system becomes $(s_3\{YE, w\}, s\emptyset)$. We complete the simulation of the matrix by applying the rule $s_3E \to s$. Similar procedure can be given for simulating a matrix $m'_i : (X \to Y, B^{(2)} \to \dagger)$.

We repeat the procedure for simulating a matrix of type 2 (or type 3) for all the matrices of type 2 (or type 3) and end the process of simulation by simulating a type 4 matrix. So any derivation in G can be simulated by a computation in Π and, conversely, the result of successful computations in Π corresponds to the terminal derivations in G. Thus, the equality $L(\Pi) = L(G)$ follows. □

We now show the universality of our system working in the minimal mode.

Theorem 5. *$RE = RtP_{2,4}(min, \gamma)$, for $\gamma \in \{repl, one, spread\}$.*

Proof. We start by considering a matrix grammar $G = (N, T, S, M, F)$ in strong binary normal form, and construct the rewriting tissue P system

$$\Pi = (V, T, \sigma_1, \sigma_2, \{(1,2),(2,1)\}, 1),$$

where:

– $V = N_1 \cup N_2 \cup \{X' \mid X \in N_1\} \cup \{E, E_1, Z\} \cup \{X_i, A_i, A'_i \mid X \in N_1 \cup \{f\}, A \in N_2, 0 \le i \le k+1\}$;

– $\sigma_1 = (\{s, s_1, s_2, s_3\}, s, \{X, A\}, P_1)$, where P_1 contains the following rules:

1. $sX \to s_1 Y_i(go)$, for a matrix $m_i : (X \to Y, A \to x)$;
2. $s_1 A \to sxA_i(go)$, for a matrix $m_i : (X \to Y, A \to x)$;
3. $sX \to s_1 f_i(go)$, for a matrix $m_i : (X \to \lambda, A \to x)$;
4. $s_1 A \to sxA'_i(go)$, for a matrix $m_i : (X \to \lambda, A \to x)$;
5. $sY' \to s_2 Y$, $Y \in N_1$;
6. $s_2 E \to s$;
7. $sf' \to s_2$;
8. $s_2 E_1 \to s\lambda(out)$;
9. $sX \to s_3 Y^{(i)}(go)$, for a matrix $m'_j : (X \to Y, B^{(i)} \to \dagger)$;
10. $s_3 A \to s_1 A(go)$;
11. $s_1 Y \to sY'(go)$;
12. $sE \to s_2 Z(out)$;
13. $sZ \to sZ(out)$;
14. $sA_i \to sZ(out), 0 \le i \le k$;

– $\sigma_2 = (\{s, s_1, s_2, s_3\}, s, \emptyset, P_2)$, where P_2 contains the following rules:

1. $sY_i \to s_3 Y_{i-1}, Y \in N_1 \cup \{f\}, 1 \le i \le k$;
2. $s_3 A_i \to sA_{i-1}, A \in N_2, 1 \le i \le k$;
3. $s_3 A'_i \to sA'_{i-1}, A \in N_2, 1 \le i \le k$;

4. $sY_0 \to s_3Y'(go), Y \in N_1 \cup \{f\}$;
5. $s_3A_0 \to sE(go), A \in N_2$;
6. $s_3A'_0 \to sE_1(go), A \in N_2$;
7. $sY^{(i)} \to s_iY(go), Y \in N_1$;
8. $s_iB^{(i)} \to s_iZ(go)$, for a matrix $m'_j : (X \to Y, B^{(i)} \to \dagger)$;
9. $s_iY' \to sY'(go)$;
10. $sA \to sAE(go)$.

The system works as follows:

Here also the system has two cells so that all the three modes of transmitting the processed strings from a cell to other cells are same. Since the system is working in the minimal mode, only one rule can be applied on one of the available strings. Initial configuration of the system is $(s\{X, A\}, s\emptyset)$. Let us suppose that at a particular instant, the configuration of the system is $(s\{X, w_1Aw_2\}, s\emptyset)$, where $w_1, w_2 \in (N_2 \cup T)^*$.

In order to simulate a matrix $m_i : (X \to Y, A \to x)$, we apply the rule $sX \to s_1Y_i(go)$ so that the configuration of the system becomes $(s_1\{w_1Aw_2\}, s\{Y_i\})$. We now simultaneously apply the rules $s_1A \to sxA_j(go)$ in σ_1 and $sY_i \to s_3Y_{i-1}$ in σ_2 and change the configuration of the system to $(s\emptyset, s_2\{Y_{i-1}, w_1xA_jw_2\})$. In the next step, we change the state of σ_2 to s by decreasing the subscript of A. Now the configuration of the system is $(s\emptyset, s\{Y_{i-1}, w_1xA_{j-1}w_2\})$. The process of decreasing the subscripts of both Y and A (each in subsequent steps) is repeated until one of the subscripts becomes 0. Based on the subscripts of Y and A, we have the following situations:

- if $i < j$, at a certain stage the configuration of the system becomes $(s\emptyset, s\{Y_0, w_1xA_{j-i}w_2\})$. By applying the rule $sY_0 \to s_3Y'(go)$ we change the configuration of the system to $(s\{Y'\}, s_3\{w_1xA_{j-i}w_2\})$. In the next step, the subscript of A is decreased and the state is changed to s. At the same time, in σ_1 we replace Y' with Y and change the state to s_2. Now the system reaches an halting configuration $(s_2\{Y\}, s\{w_1xA_{j-i-1}w_2\})$ due to non applicability of any rule so that we get no result.

- if $i > j$, at a certain stage the configuration of the system becomes $(s\emptyset, s\{Y_{i-j}, w_1xA_0w_2\})$. We now decrease the subscript of Y and change the state to s_3. In the next step, by applying the rule $s_3A_0 \to sE(go)$ we change the configuration of the system to $(s\{w_1xEw_2\}, s\{Y_{i-j-1}\})$. In σ_1, a trap symbol Z is introduced by applying the rule $sE \to s_2Z(out)$. Hence we get no result.

– if $i = j$, at a certain stage the configuration of the system becomes $(s\emptyset, s\{Y_0, w_1xA_0w_2\})$ so that by applying the rule $sY_0 \rightarrow s_3Y'(go)$ we change the configuration of the system to $(s\{Y'\}, s_3\{w_1xA_0w_2\})$. In the next step, we simultaneously apply the rules $s_3A_0 \rightarrow sE(go)$ in σ_2 and $sY' \rightarrow s_2Y$ in σ_1 so that the configuration of the system becomes $(s_2\{Y, w_1xEw_2\}, s_3\emptyset)$. In the next step we erase the symbol E by applying the rule $s_2E \rightarrow s$.

So when both the subscripts are equal, we get the correct simulation of the matrix. We can explain the simulation of matrices of type 4 in a similar way. Whenever we simulate a matrix of type 4, at a certain stage the configuration of the system becomes $(s\{f', w_1xE_1w_2\}, s\emptyset)$. By applying the rule $sf' \rightarrow s_2$, we erase f' and change the state to s_2. In the next step we apply the rule $s_2E_1 \rightarrow s\lambda(out)$ so that the configuration of the system becomes $(s\{\lambda\}, s\emptyset)$ and the string w_1xw_2 is sent to the environment. If this string contains any nonterminal, it is not considered as a result.

For simulating a matrix $m'_i : (X \rightarrow Y, B^{(1)} \rightarrow \dagger)$, we apply the rule $sX \rightarrow s_3Y^{(1)}(go)$ and change the configuration of the system to $(s_3\{w_1Aw_2\}, s\{Y^{(1)}\})$. We now simultaneously apply the rules $s_3A \rightarrow s_1A(go)$ in σ_1 and $sY^{(1)} \rightarrow s_1Y(go)$ in σ_2 so that the configuration becomes $(s_1\{Y\}, s_1\{w_1Aw_2Y^{(1)}\})$. If w_1Aw_2 contains $B^{(1)}$, a trap symbol Z is introduced. Otherwise, no rule is applied on the string. At the same time, we apply the rule $s_1Y \rightarrow sY'(go)$ in σ_1 and send the string Y' to σ_2. We now apply the rule $s_1Y' \rightarrow sY'(go)$ so that the string Y' is sent to σ_1. Once the state of σ_2 becomes s, we apply the rule $sA \rightarrow sAE(go)$ so that the string w_1AEw_2 is sent to σ_1. At the same time, we replace Y' with Y in σ_1 and change the state to s_2. Finally we apply the rule $s_2E \rightarrow s$ so that the symbol E is erased. In this way we can simulate the matrix. Similar procedure can be given for a matrix of the form $m' : (X \rightarrow Y, B^{(2)} \rightarrow \dagger)$.

The procedure for simulating a matrix of type 2 (type 3) can be iterated for all the matrices of type 2 (type 3). Finally the simulation process ends by simulating a type 4 matrix. So any derivation in G can be simulated by a computation in Π and, conversely, the result of successful computations in Π corresponds to the terminal derivations in G. Thus, the equality $L(\Pi) = L(G)$ follows. □

In the proof of the above theorem, we consider the minimal mode so that only one rule can be applied at any time on the available strings. But we consider two strings X and w such that $X \in N_1$ and $w \in (N_2 \cup T)^*$. Since the two strings are over disjoint alphabet, a rule which can be applied on one string cannot be applied on the other string. From this observation we can easily prove the following theorem.

Theorem 6. *$RE = RtP_{2,4}(par, \gamma)$, for $\gamma \in \{repl, one, spread\}$.* □

6 Conclusion

We proposed a variant of tissue P systems and studied its efficiency and computational capacity. Interestingly for all modes we obtained the universality with at most *two* membranes and *four* states. Currently we are investigating the power of tissue P systems with leftmost rewriting [Mutyam and Krithivasan 2004]. Like the descriptional complexity for rewriting P systems [Mutyam 2004], one can define the descriptional complexity for rewriting tissue P systems. For rewriting tissue P systems, in addition to the descriptional complexity measures given in [Mutyam 2004], one can consider descriptional complexity measures such as number states, type of communication (i.e., one-way or two-way), and membrane structure representation (i.e., star, tree, chain, cycle). Finally, it is an interesting research problem to consider parallel rewriting [Krishna and Rama 2000] to define a new mode of processing string-objects and study the computational capacity of such systems.

References

[Arbib 1978] Arbib, M.A.: "Brains, Machines, and Mathematics"; Springer, Berlin, 2nd edition.

[Dassow and Păun 1989] Dassow, J., Păun, G.: "Regulated Rewriting in Formal Language Theory" ; Springer-Verlag, Berlin.

[Dassow and Păun 1999] Dassow, J., Păun, G.: "On the power of membrane computing"; Journal of Universal Computer Science 5, 2 (1999), 33-49.

[Ferretti et al. 2003] Ferretti, C., Mauri, G., Păun, Gh., Zandron, C.: "On three variants of rewriting P systems"; Theoretical Computer Science 301, 1-3 (2003), 201-215.

[Freund and Păun 2001] Freund, R., Păun, Gh.: "On the number of non-terminal symbols in graph-controlled, programmed, and matrix grammars"; In Margenstern, M., Rogozhin, Y. (eds.): Machines, Computations, and Universality. 3^{rd} International Conference, LNCS 2055, Springer-Verlag, Berlin (May 2001), 214-225.

[Krishna and Rama 2000] Krishna, S.N., Rama, R.: "On the power of P systems with sequential/parallel rewriting"; International Journal of Computer Mathematics 76, 1-2 (2000), 317-330.

[Mutyam 2004] Mutyam, M.: "Descriptional complexity of rewriting P systems" ; Journal of Automata, Languages and Combinatorics 9, 2 (2004) (to appear).

[Mutyam and Krithivasan 2004] Mutyam, M., Krithivasan, K.: "Tissue P systems with leftmost rewriting"; Submitted.

[Martín-Vide and Păun 2000] Martín-Vide, C., Păun, Gh.: "String objects in P systems"; In Proceedings of Workshop on Algebraic Systems, Formal Languages and Computations, RIMS Kokyuroku, Kyoto university, Kyoto (2000), 161-169.

[Martín-Vide et al. 2003] Martín-Vide, C., Păun, Gh., Pazos, J., Rodríguez-Patón, A.: "Tissue P systems"; Theoretical Computer Science 296, 2 (2003), 295-326.

[Păun 2000] Păun, Gh.: "Computing with membranes"; Journal of Computer and System Sciences 61, 1 (2000), 108-143.

[Păun and Yokomori 1999] Păun, Gh., Yokomori, T.: "Membrane computing based on splicing"; In Winfree, E., Gifford, D.(eds.): Preliminary Proceedings of 5^{th} International Meeting on DNA Based Computers, MIT, USA (1999), 213-227.