Available online at www.sciencedirect.com

SCIENCE DIRECT®

Theoretical Computer Science 334 (2005) 161–175

Theoretical Computer Science

www.elsevier.com/locate/tcs

Rewriting P systems: improved hierarchies

Madhu Mutyam

International Institute of Information Technology, Gachibowli, Hyderabad, 500 019, Andhra Pradesh, India

Received 1 May 2003; received in revised form 29 December 2003; accepted 10 June 2004

Communicated by G. Rozenburg

Abstract

Generally, for proving universality results about rewriting P systems one considers matrix grammars in the strong binary normal form. Such grammars contain both matrices with rules used in the appearance checking mode and matrices without appearance checking rules. In the proofs of most of the universality theorems reported in the literature, appearance checking matrices are simulated by using only two membranes, while four membranes are used for simulating matrices without appearance checking rules. Thus, a way to improve these theorems is to diminish the number of membranes used for simulating matrices without appearance checking rules. In this paper we address this problem, and give first a general improved result about simulating matrix grammars without appearance checking: three membranes are shown to suffice. This result is then used to improve several universality results from various membrane computing papers, for instance, about P systems with replicated rewriting, with leftmost rewriting, with conditional communication, as well as for hybrid P systems with finite choice.

Keywords: P systems; Rewriting P systems; Recursively enumerable languages; Computational universality; Leftmost rewriting; Replicated rewriting; Matrix languages; Conditional communication; Penttonen normal form; Hybrid P systems

1. Introduction

Natural computing is a field of research which tries to imitate nature's way of computing.

E-mail address: madhu_mutyam@iiit.net.

0304-3975/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.06.015

P systems [12] is a branch of natural computing which abstracts from the structure and the functioning of living cells. Cells contain several compartments and these compartments contain chemical compounds. The chemical compounds are processed by using chemical reactions. In correspondence to the structure and the functioning of a cell, a P system has a membrane structure and each membrane contains objects and evolution rules. Using evolution rules objects can be created or destroyed or even sent to neighboring membranes. Based on the type of objects and the type of evolution rules several variants of P systems are defined. In this work we concentrate on *rewriting P systems* [11], where we consider string-objects and context-free rules for processing these objects. The reader is assumed to be familiar with formal language theory basic elements, for instance, from [14], as well as with basics of membrane computing, for instance from [13].

A *rewriting P system* of degree n, $n \geqslant 1$, is a construct

$$\Pi = (V, T, \mu, L_1, \ldots, L_n, R_1, \ldots, R_n),$$

where:

- V is the total alphabet of the system;
- $T \subseteq V$ is the terminal alphabet;
- μ is a membrane structure;
- L_i, $1 \leqslant i \leqslant n$, are finite languages over V, representing the strings initially present in the regions $1, \ldots, n$ of μ;
- R_i, $1 \leqslant i \leqslant n$, are finite sets of rewriting rules of the form $X \to v(tar)$, where $X \in V$, $v \in V^*$, and $tar \in \{here, out, in\}$.

We process string-objects in rewriting P systems with rules of the form $X \to v(tar)$, where $X \to v$ is a usual 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 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.

The configurations and the computations of Π are defined in the usual manner (for detailed explanation one can refer to [11]). A computation is *successful* if and only if it halts, i.e., there is no rule applicable to the strings present in the last configuration.

The result of a successful computation consists of the strings over T which can be ejected from the skin membrane. We denote by $L(\Pi)$ the language computed by Π in the way described above and by $RP_n(i/o)$ the family of languages generated by rewriting P systems of degree at most n. If there is no bound on the degree of membrane system, we replace n with $*$.

2. Prerequisites

Before proceeding to the main section of the paper, we recall the definition of matrix grammars and Penttonen normal form.

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 $(A_1 \to z_1, \ldots, A_n \to z_n)$, $n \geqslant 1$, of context-free rules, and F is a set of occurrences of rules in M. For a sentential form x, an element $m = (r_1, r_2, \ldots, r_n)$ is executed by applying productions $r_1, r_2, \ldots, r_n$ one after another, in the strict order they are listed, with the exception that if some r_i, $1 \leqslant i \leqslant n$, cannot be applied, then it has to be an element of F and the next production r_{i+1} has to be taken into consideration. If these conditions do not hold, the matrix is not applicable. The resulting sentential form y is said to be the string directly derived from x. 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 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. Matrices of type 1, 2 and 4 are called matrices without appearance checking and those of type 3 are called appearance checking matrices.

According to [2], 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 $ac(G) \leqslant 2$, it is said to be in *strong binary normal form*. In [4], it was proved that each recursively enumerable language can be generated by a matrix grammar G such that $ac(G) \leqslant 2$.

In the literature of formal language theory, there are several normal forms for *type* 0 grammars. One such normal form is Penttonen normal form. Any *type* 0 grammar $G = (N, T, S, R)$ is in *Penttonen normal form* if each rule in R is one of the following forms:

$$A \to \lambda,\ A \to a,\ A \to BC,\ AB \to AC, \quad \text{for } A, B, C \in N \text{ and } a \in T.$$

3. Improved results

In this section we give some improved results of rewriting P systems. First we show that only three membranes are enough for generating matrix languages. Based on this result, we can improve the universality result for P systems with replicated rewriting. If we take leftmost rewriting into consideration, then matrix grammars can be simulated

with rewriting P systems of degree 2. This result improves the universality result for P systems with leftmost rewriting.

3.1. Simulating matrix grammars

From [3], we know that rewriting P systems of degree 4 generate all matrix languages. The following theorem improves this result.

Theorem 3.1. $MAT = RP_3(i/o)$.

Proof. Let $G = (N, T, S, M)$ be a matrix grammar without appearance checking in binary normal form with $(S \to XA)$ as the first matrix. Here $N = N_1 \cup N_2 \cup \{S\}$ and $M = \{m_i \mid 1 \leqslant i \leqslant k\}$, where each m_i, $1 \leqslant i \leqslant k$, is a matrix. We replace every production of the form $X \to \lambda$ with $X \to f$, where f is a new symbol.

We now construct the P system

$$\Pi = (V, T, [_1[_2[_3\ ]_3]_2]_1, \{XA\}, \emptyset, \emptyset, R_1, R_2, R_3),$$

where:

- $V = N_1 \cup N_2 \cup \{X', X_i \mid X \in N_1, 1 \leqslant i \leqslant k\} \cup \{f, f', f_i \mid 1 \leqslant i \leqslant k\} \cup \{A_i, A_i' \mid A \in N_2, 1 \leqslant i \leqslant k\}$;
- R_1 contains the following rules:
  (1) $A \to xA_i(in)$, for $m_i : (X \to Y, A \to x)$, $1 \leqslant i \leqslant k$;
  (2) $Y' \to Y(here)$, $Y \in N_1 \cup \{f\}$;
  (3) $f \to \lambda(out)$;
- R_2 contains the following rules:
  (1) $X \to Y_i(in)$, for $m_i : (X \to Y, A \to x)$, $Y \in N_1 \cup \{f\}$, $1 \leqslant i \leqslant k$;
  (2) $A_i' \to A_{i-1}'(in)$, $1 < i \leqslant k$;
  (3) $A_1' \to \lambda(out)$;
- R_3 contains the following rules:
  (1) $A_i \to A_i'(here)$, $1 \leqslant i \leqslant k$;
  (2) $Y_i \to Y_{i-1}(out)$, $Y \in N_1 \cup \{f\}$, $1 < i \leqslant k$;
  (3) $Y_1 \to Y'(out)$, $Y \in N_1 \cup \{f\}$.

The initial configuration of the system is $[_1XA[_2[_3\ ]_3]_2]_1$. Let us suppose that at a particular instance we have a configuration of the form $[_1Xw_1Aw_2[_2[_3\ ]_3]_2]_1$. In order to simulate a matrix $m_i : (X \to Y, A \to x)$, we apply a rule $R_1(1)$, so that the configuration of the system is changed to $[_1[_2Xw_1xA_iw_2[_3\ ]_3]_2]_1$. In membrane 2 we can only apply a rule of type $R_2(1)$. After applying such a rule, the configuration of the system becomes $[_1[_2[_3Y_jxA_iw_2]_3]_2]_1$. In membrane 3 we can first apply either $R_3(1)$ or $R_3(2)$. If we apply $R_3(2)$ first, then the configuration becomes $[_1[_2Y_{j-1}w_1xA_iw_2[_3\ ]_3]_2]_1$, and we cannot proceed further. Otherwise, i.e., if we apply $R_3(1)$ first, the resulting string will remain in the same membrane, so that in the next step we can apply $R_3(2)$. After this step of rewriting, the configuration of the system will be $[_1[_2Y_{j-1}w_1xA_i'w_2[_3\ ]_3]_2]_1$. Now in membrane 2, we can apply $R_2(2)$ (if $i \neq 1$) so that the configuration is changed to $[_1[_2[_3Y_{j-1}xA_{i-1}w_2]_3]_2]_1$. We repeat this process to decrease the subscript values of

Y and A until one of these subscripts becomes 1. If $j \neq i$, then we have the following cases:

- if $j > i$, then at some stage we reach a configuration of the form

  $$[_1[_2 Y_{j-i} w_1 x A'_1 w_2[_3\ ]_3]_2]_1.$$

  Here we can only apply $R_2(3)$, so that the configuration is changed to $[_1 Y_{j-i} w_1 x w_2[_2[_3\ ]_3]_2]_1$. If there is no nonterminal $B \in N_2$ in $w' = w_1 x w_2$, then we cannot proceed further. On the other hand, if any nonterminal $B \in N_2$ is present in w', then with $R_1(1)$ we can send the resulting string w'' to membrane 2. In membrane 2 no rule can be applied, so that we cannot proceed further.
- if $j < i$, then at some stage we reach a configuration of the form

  $$[_1[_2[_3 Y_1 w_1 x A'_{i-j+1} w_2]_3]_2]_1.$$

  Here we can only apply $R_3(3)$, so that the configuration is changed to $[_1[_2 Y' w_1 x A'_{i-j+1} w_2[_3\ ]_3]_2]_1$. Since $j < i$, we can only apply $R_2(2)$, so that the configuration becomes $[_1[_2[_3 Y' w_1 x A'_{i-j} w_2]_3]_2]_1$ and the system will be halted (since no rule from R_3 can be applied), and no result is obtained.

In order to get a successful computation, the value of i must be equal to that of j. In such a case, at a certain stage we have the following configuration:

$$[_1[_2[_3 Y_1 w_1 x A'_1 w_2]_3]_2]_1.$$

We now apply $R_3(3)$, so that the configuration becomes $[_1[_2 Y' w_1 x A'_1 w_2[_3\ ]_3]_2]_1$. In membrane 2, we apply $R_2(3)$ and obtain $[_1 Y' w_1 x w_2[_2[_3\ ]_3]_2]_1$. In membrane 1, we can first apply either $R_1(1)$ or $R_1(2)$. If we apply $R_1(1)$ first, then the system will be halted. In order to proceed further, we first apply $R_1(2)$ and then repeat the same process as above simulating another matrix of G. Whenever the symbol f is introduced in the sentential form (this indicates the last step of the derivation in G), the string will be sent out using $R_1(3)$. We consider terminal strings which are ejected from the system as the result of the computation. Therefore, $L(G) = L(\Pi)$, hence $MAT \subseteq RP_3(i/o)$. The inclusion $RP_*(i/o) \subseteq MAT$ was proved in [3], and thus we have the equality $MAT = RP_3(i/o)$. □

3.2. Replicated rewriting

Here we consider a variant of rewriting P systems, i.e., *P systems with replicated rewriting* [6], where the evolution rules are replicated rewriting rules. A replicated rewriting rule is of the form $X \to v_1(tar_1) \| \cdots \| v_n(tar_n)$. To apply this rule to a string w one replaces one occurrence of X in w by $v_1, \ldots, v_n$, in a context-free manner. Thus, this rewriting yields n strings, $w_1 v_1 w_2, \ldots, w_1 v_n w_2$, where $w = w_1 X w_2$. As usual, these n strings are sent to regions indicated by the targets $tar_1, \ldots, tar_n$, respectively.

We denote by $RRP_n(i/o)$, $n \geqslant 1$, the family of languages generated by P systems with replicated rewriting of degree at most n.

From [10], we know that the universality for P systems with replicated rewriting can be achieved with *six* membranes. We improve this result and show that universality can be achieved with *five* membranes.

Theorem 3.2. $RE = RRP_5(i/o)$.

Proof. We prove only the inclusion $RE \subseteq RRP_5(i/o)$, the reverse inclusion can be proved in a straightforward manner. Let us consider a matrix grammar with appearance checking, $G = (N, T, S, M, F)$, in strong binary normal form with $N = N_1 \cup N_2 \cup \{S, \dagger\}$. Assume that $ac(G) = 2$, and let $B^{(1)}$ and $B^{(2)}$ be the two objects in N_2 for which we have rules $B^{(j)} \to \dagger$ in matrices of M. Let us assume that we have h matrices of the form $(X \to Y, B^{(j)} \to \dagger)$, $X, Y \in N_1$, $j \in \{1, 2\}$, $1 \leqslant i \leqslant h$, and k matrices of the form $m_i : (X \to Y, A \to x)$, $X \in N_1$, $A \in N_2$, $Y \in N_1 \cup \{\lambda\}$, and $x \in (N_2 \cup T)^*$, $1 \leqslant i \leqslant k$. Each matrix of the form $(X \to \lambda, A \to x)$, $X \in N_1$, $A \in N_2$, $x \in T^*$, is replaced by $(X \to f, A \to x)$, where f is a new object. We continue to label the obtained matrix in the same way as the original one. The matrices of the form $(X \to Y, B^{(j)} \to \dagger)$, $X, Y \in N_1$ are labeled by m'_i, with $i \in lab_j$, for $j \in \{1, 2\}$, such that lab_1, lab_2 and $lab_0 = \{1, 2, \ldots, k\}$ are mutually disjoint sets.

We construct the P system

$$\Pi = (V, T, \mu, L_1, \ldots, L_5, R_1, \ldots, R_5),$$

where:

- $V = N_1 \cup N_2 \cup T \cup \{X', X^{(1)}, X^{(2)}, X_i \mid X \in N_1, 1 \leqslant i \leqslant k\} \cup \{A_i, A'_i \mid A \in N_2, 1 \leqslant i \leqslant k\} \cup \{Z, f, f', \dagger\} \cup \{f_i \mid 1 \leqslant i \leqslant k\}$;
- $\mu = [_1[_2[_3\ ]_3]_2[_4\ ]_4[_5\ ]_5]_1$;
- $L_1 = \{XA\}$;
- $L_i = \emptyset$, $2 \leqslant i \leqslant 5$;
- R_1 contains the following rules:
  (1) $A \to xA_i(in)$, for $m_i : (X \to Y, A \to x)$, $1 \leqslant i \leqslant k$;
  (2) $Y' \to Y(here)$, $Y \in N_1 \cup \{f\}$;
  (3) $X \to Y(here)||Y^{(1)}(in)$, for $m'_i : (X \to Y, B^{(1)} \to \dagger)$;
  (4) $X \to Y(here)||Y^{(2)}(in)$, for $m'_i : (X \to Y, B^{(2)} \to \dagger)$;
  (5) $f \to \lambda(out)$;
- R_2 contains the following rules:
  (1) $X \to Y_i(in)$, for $m_i : (X \to Y, A \to x)$, $Y \in N_1 \cup \{f\}$, $1 \leqslant i \leqslant k$;
  (2) $A'_i \to A'_{i-1}(in)$, $1 < i \leqslant k$;
  (3) $A'_1 \to \lambda(out)$;
  (4) $Y^{(1)} \to \dagger(here)$;
  (5) $Y^{(2)} \to \dagger(here)$;
  (6) $\dagger \to \dagger(here)$;
- R_3 contains the following rules:
  (1) $A_i \to A'_i(here)$, $1 \leqslant i \leqslant k$;
  (2) $Y_i \to Y_{i-1}(out)$, $Y \in N_1 \cup \{f\}$, $1 < i \leqslant k$;
  (3) $Y_1 \to Y'(out)$, $Y \in N_1 \cup \{f\}$;

- R_4 contains the following rules:
  (1) $B^{(1)} \to \dagger(here)$;
  (2) $A_i \to \dagger(here)$, $A \in N_2$, $1 \leqslant i \leqslant k$;
  (3) $Y^{(2)} \to \dagger(here)$;
  (4) $\dagger \to \dagger(here)$;
- R_5 contains the following rules:
  (1) $B^{(2)} \to \dagger(here)$;
  (2) $A_i \to \dagger(here)$, $A \in N_2$, $1 \leqslant i \leqslant k$;
  (3) $Y^{(1)} \to \dagger(here)$;
  (4) $\dagger \to \dagger(here)$.

Here we explain the procedure to simulate appearance checking matrices, because for other type of matrices the procedure is same as the one explained in the proof of Theorem 3.1.

In order to simulate a matrix $m'_i : (X \to Y, B^{(1)} \to \dagger)$ of type $3(a)$, we apply the replicated rewriting rule $X \to Y(here)||Y^{(1)}(in)$, so that the string Yw remains in the same membrane and the other one $Y^{(1)}w$ is sent to any one of the three inner membranes. If $Y^{(1)}w$ is sent to either membrane 2 or 5, then a trap symbol is introduced and the computation never halts. In order to get result of a computation, the string $Y^{(1)}w$ is sent to membrane 4. If the string $Y^{(1)}w$ contains any symbol $B^{(1)}$, then a trap symbol is introduced and the computation never halts. Otherwise, the string $Y^{(1)}w$ remains as it is and we cannot consider it for result of a computation and the computation continues with the other string. In this way we can simulate the matrix. Similar procedure can be given for a matrix of type $3(b)$.

These operations can be iterated, hence any derivation in G can be simulated by a computation in Π and, conversely, the computations in Π correspond to correct derivations in G. Thus, the equality $L(\Pi) = L(G)$ follows. □

3.3. Leftmost rewriting

A restriction in the use of rules of *rewriting P systems* was considered in *P systems with leftmost rewriting* [3], where any string is rewritten in the leftmost position which can be rewritten by a rule from its region. In order to apply a rule, we examine the symbols of the string, step by step, from left to right and the first one which can be rewritten by a rule from the region of the string is rewritten. If there are several rules with same left-hand side symbol, we can select one of them nondeterministically.

We denote by $RP_n(left)$, $n \geqslant 1$, the family of languages generated by P systems with leftmost rewriting of degree at most n.

Theorem 3.3. $MAT \subset RP_2(left)$.

Proof. As in the proof of Theorem 3.1, we consider a matrix grammar without appearance checking, $G = (N, T, S, M)$, in binary normal form. We now construct the P system

$$\Pi = (V, T, [_1[_2\ ]_2]_1, \{XA\}, \emptyset, R_1, R_2),$$

where:

- $V = N_1 \cup N_2 \cup \{g, f, f', \dagger\} \cup \{X', X_i \mid X \in N_1, 1 \leqslant i \leqslant k\} \cup \{A_i, A' \mid A \in N_2, 1 \leqslant i \leqslant k\} \cup \{f_i \mid 1 \leqslant i \leqslant k\}$;
- R_1 contains the following rules:
  (1) $A \to A'(here)$, $A \in N_2$;
  (2) $A \to A_i x(in)$, for $m_i : (X \to Y, A \to x)$, $1 \leqslant i \leqslant k$;
  (3) $A_i \to A_{i-1}(in)$, $1 < i \leqslant k$, $A \in N_2$;
  (4) $A_1 \to g(in)$, $A \in N_2$;
  (5) $Y' \to Y(here)$, $Y \in N_1$;
  (6) $f' \to \lambda(out)$;
  (7) $g \to \dagger(here)$;
  (8) $\dagger \to \dagger(here)$;
- R_2 contains the following rules:
  (1) $X \to Y_i(out)$, for $m_i : (X \to Y, A \to x)$, $Y \in N_1 \cup \{f\}$, $1 \leqslant i \leqslant k$;
  (2) $Y_i \to Y_{i-1}(out)$, $Y \in N_1 \cup \{f\}$, $1 < i \leqslant k$;
  (3) $Y_1 \to Y'(here)$, $Y \in N_1 \cup \{f\}$;
  (4) $A' \to A(here)$, $A \in N_2$;
  (5) $g \to \lambda(out)$;
  (6) $A_i \to \dagger(here)$, $A \in N_2$, $1 < i \leqslant k$;
  (7) $\dagger \to \dagger(here)$.

Initially we have XA in membrane 1. Let us suppose that at a particular instance we have the following configuration:

$$[_1 X w_1 A w_2 [_2\ ]_2]_1.$$

We assume that $A \notin w_1$. If we want to simulate a matrix $m_i : (X \to Y, A \to x)$, we first change all nonterminals from N_2 in w_1 to their primed version by using $R_1(1)$. We now apply $R_1(2)$ so that the configuration is changed to $[_1 [_2 X w_1' A_i x w_2]_2]_1$. Note that w_1' indicates the primed version of w_1. In membrane 2 only one rule, $R_1(1)$, can be applied. After applying the rule, the configuration of the system is changed to $[_1 Y_j w_1' A_i x w_2 [_2\ ]_2]_1$. We now decrease the subscript values of both Y and A. As explained in the proof of the previous theorem, if the values of i and j are not equal, then we get no result of the computation. Otherwise, at some stage we get the following configuration:

$$[_1 Y_1 w_1' A_1 x w_2 [_2\ ]_2]_1.$$

We now apply $R_1(4)$ so that g is introduced in the sentential form and the resulting string is send to membrane 2. In membrane 2 we apply $R_2(3)$. Since the target of $R_2(3)$ is $here$, the string remains in the same membrane so that we can convert all primed nonterminals to their original form by repeatedly applying $R_2(4)$. We now erase g and sent the resulting string to membrane 1. In membrane 1 we apply $R_1(5)$ so that Y' becomes Y and this completes the correct simulation of the matrix m_i. We repeat this process until the symbol f' is introduced in the sentential form. By applying $R_1(8)$, we erase the symbol f' and send the resulting string to the environment. Since the result of the computation is the correct simulation of the

matrix grammar, we have $L(G) = L(\Pi)$, hence $MAT \subseteq RP_2(left)$. The inclusion is proper, because $EOL \subseteq RP_2(left)$ [3] and $EOL - MAT \neq \emptyset$. □

In [9], it was proven that the universality for a rewriting P system with leftmost rewriting can be achieved with *five* membranes. Among the five membranes, three membranes were used for simulating matrices of type 2 and 4 and other two membranes were used for simulating matrices of type 3. But from the above theorem we know that only two membranes are enough for simulating matrices of type 2 and 4. Hence, we can get the following result:

Theorem 3.4. $RE = RP_4(left)$.

4. Conditional communication

A variant of rewriting P systems was considered in [1], where restrictions are imposed on the communication of string-objects through membranes. Each membrane in a membrane system is associated with both *permitting* and *forbidding* conditions. The conditions can be of the following forms:
Empty: no restriction is imposed on strings, they can freely exit the current membrane or enter any of the directly inner membranes; an empty permitting condition is denoted by $(true, tar)$, $tar \in \{in, out\}$; an empty forbidding condition is denoted by $(false, notar)$, $tar \in \{in, out\}$;
Symbol checking: each permitting condition is of the form (a, tar) and each forbidding condition is of the form $(a, notar)$, where $tar \in \{in, out\}$ and a is a symbol from the total alphabet of the system; a string w can go to a lower membrane only if there is a pair (a, in) as a permitting condition in the current membrane with $a \in w$, and for each $(b, notin)$ in the forbidding condition set of the current membrane we have $b \notin w$; In a similar way, we can explain the procedure to send a string to outer membranes;
Substring checking: instead of single symbols, here we consider substrings for both permitting and forbidding conditions.

The above conditions can be represented as *empty*, *symb*, and sub_k, respectively, where k is the length of the longest string in all permitting and forbidding conditions.

Computations can be defined in the usual way. After rewriting a string, the resulting string can be checked against the permitting and forbidding conditions of the current membrane. If it fulfills the requested conditions, it will be immediately sent out of the membrane or to an inner membrane, if any exists; if it fulfills both in and out conditions, it is sent either out of the membrane or to a lower membrane. If a string cannot be rewritten, it is directly checked against the communication conditions, and, as above, it leaves the membrane (or remains inside forever) depending on the result of this checking. That is, the rewriting has priority over communication: we first try to rewrite a string and only after that do we try to communicate the result of the rewriting or the string itself if no rewriting is possible on it.

The result of a computation consists of all terminal strings sent out of the system. We denote by $RP_n(rw, \alpha, \beta)$, $n \geqslant 1$, $\alpha, \beta \in \{empty, symb\} \cup \{sub_k \mid k \geqslant 2\}$, the family of lan-

guages generated by P systems of degree at most n and with permitting conditions of type α and forbidding conditions of type β.

In [1], it was proven that P systems of degree 4 with permitting conditions of type sub_2 and forbidding conditions of type *symb* are computationally universal. We improve the result and show that $three$ membranes are enough for achieving the universality.

Theorem 4.1. $RE = RP_3(rw, sub_2, symb)$.

Proof. Let us consider a *type* 0 grammar $G = (N, T, S, P)$, in Penttonen normal form, with the noncontext-free rules from P labelled in a one-to-one manner, and construct the system

$$\Pi = (V, T, [_1[_2[_3\ ]_3]_2]_1, \{S\}, \emptyset, \emptyset, (R_1, P_1, F_1), (R_2, P_2, F_2), (R_3, P_3, F_3)),$$

with the following components:

$$\begin{aligned}
V &= N \cup T \cup \{A', A'' \mid A \in N\};\\
R_1 &= \{A \to x \mid A \to x \in P\}\\
&\quad \cup \{B \to (B, r) \mid r : AB \to AC \in P\}\\
&\quad \cup \{A \to A \mid A \in N\};\\
P_1 &= \{((A, r), in) \mid A \in N\};\\
&\quad \cup \{(true, out) \mid A \in N\};\\
F_1 &= \{(A, notout), ((A, r), notout) \mid A \in N\};\\
R_2 &= \{A' \to A \mid A \in N\}\\
&\quad \cup \{(A, r) \to (A, r) \mid A \in N\};\\
P_2 &= \{(A(B, r), in) \mid r : AB \to AC\}\\
&\quad \cup \{(true, out)\};\\
F_2 &= \{((A, r), notout), (A', notout) \mid A \in N\};\\
R_3 &= \{(B, r) \to C' \mid r : AB \to AC \in P\};\\
P_3 &= \{(A', out) \mid A \in N\};\\
F_3 &= \{(false, notout)\}.
\end{aligned}$$

The system works as follows:

The initial configuration of the system is $[_1S[_2[_3\ ]_3]_2]_1$. The context-free rules from P are present in R_1 as rewriting rules, hence we can simulate them without any difficulty. Let us assume that we have a string w_1ABw_2 in membrane 1. In order to simulate a rule $r : AB \to AC \in P$, we apply the rule $B \to (B, r)$ on the string so that the resulting string is sent to membrane 2. The string is sent to membrane 3 only if it has a substring of the form $A(B, r)$ such that $r : AB \to AC \in P$. Otherwise, by repeated application of the rule $(B, r) \to (B, r)$ in membrane 2, the computation never halts. In membrane 3, we replace the symbol (B, r) with C' and send the resulting string to membrane 2.

From membrane 2, the string is sent to membrane 1 by applying the rule $A' \to A$. In this way we complete the simulation of the rule.

The process can be iterated until no nonterminal is present in the sentential form. Hence, each derivation in G can be simulated in Π and, conversely, all halting computations in Π correspond to correct derivations in G. Therefore, the computation in Π can stop only after reaching a terminal string with respect to G. Thus, we have $L(G) = L(\Pi)$. □

In [1], it was proven that P systems of degree 6 with both permitting and forbidding conditions of type $symb$ are computationally universal. We improve the result and show that $five$ membranes are enough for achieving the universality.

Theorem 4.2. $RE = RP_5(rw, symb, symb)$.

Proof. Let us consider a matrix grammar $G = (N, T, S, M, F)$, in strong binary normal form. We now construct the system

$$\Pi = (V, T, [_1[_2[_3\ ]_3]_2[_4\ ]_4[_5\ ]_5]_1, \{XA\}, \emptyset, \emptyset, \emptyset, \emptyset, (R_1, P_1, F_1), \ldots, (R_5, P_5, F_5)),$$

with the following components:

$$\begin{aligned}
V &= N \cup T \cup \{A_i, A_i' \mid A \in N_2, 1 \leqslant i \leqslant k\} \\
&\quad \cup \{X', X_i, X_i' \mid X \in N_1 \cup \{f\}, 1 \leqslant i \leqslant k\} \\
&\quad \cup \{A' \mid A \in N_2\} \cup \{X^{(1)}, X^{(2)}, X' \mid X \in N_1\}; \\
R_1 &= \{A \to xA_i \mid m_i : (X \to Y, A \to x), Y \in N_1 \cup \{f\}, 1 \leqslant i \leqslant k\} \\
&\quad \cup \{X \to Y^{(1)} \mid m_i' : (X \to Y, B^{(1)} \to \dagger)\} \\
&\quad \cup \{X \to Y^{(2)} \mid m_i' : (X \to Y, B^{(2)} \to \dagger)\} \\
&\quad \cup \{X' \to X \mid X \in N_1\} \\
&\quad \cup \{f' \to \lambda\}; \\
P_1 &= \{(A_i, in) \mid A \in N_2, 1 \leqslant i \leqslant k\} \\
&\quad \cup \{(X^{(j)}, in) \mid X \in N_1, j \in \{1, 2\}\} \\
&\quad \cup \{(true, out)\}; \\
F_1 &= \{(X, notout) \mid X \in N_1 \cup N_2\} \\
&\quad \cup \{(false, notin)\}; \\
R_2 &= \{X \to Y_i \mid m_i : (X \to Y, A \to x), Y \in N_1 \cup \{f\}, 1 \leqslant i \leqslant k\} \\
&\quad \cup \{X_i' \to X_i \mid X \in N_1 \cup \{f\}, 1 \leqslant i \leqslant k\} \\
&\quad \cup \{A_i' \to A_{i-1} \mid A \in N_2, 1 < i \leqslant k\} \\
&\quad \cup \{A_1' \to \lambda \mid A \in N_2\}
\end{aligned}$$

$$\cup \{X' \to X' \mid X \in N_1 \cup \{f\}\}$$
$$\cup \{X_i \to X_i \mid X \in N_1 \cup \{f\}, 1 \leqslant i \leqslant k\}$$
$$\cup \{X^{(j)} \to X^{(j)} \mid X \in N_1, j \in \{1,2\}\};$$

$$P_2 = \{(X_i, in) \mid X \in N_1, 1 \leqslant i \leqslant k\}$$
$$\cup \{(true, out)\};$$

$$F_2 = \{(X, notin), (X, notout) \mid X \in N_1\}$$
$$\cup \{(X', notin) \mid X \in N_1 \cup \{f\}\}$$
$$\cup \{(X_i, notout), (X_i', notout), (X_i', notin) \mid X \in N_1 \cup N_2 \cup \{f\}, 1 \leqslant i \leqslant k\}$$
$$\cup \{(X^{(j)}, notin), (X^{(j)}, notout) \mid X \in N_1\};$$

$$R_3 = \{A_i \to A_i' \mid A \in N_2, 1 \leqslant i \leqslant k\}$$
$$\cup \{X_i \to X_{i-1}' \mid X \in N_1 \cup \{f\}, 1 < i \leqslant k\}$$
$$\cup \{X_1 \to X' \mid X \in N_1 \cup \{f\}\};$$

$$P_3 = \{(true, out)\};$$

$$F_3 = \{(X_i, notout) \mid X \in N_1 \cup N_2 \cup \{f\}, 1 \leqslant i \leqslant k\};$$

$$R_4 = \{B^{(1)} \to B^{(1)} \mid B \in N_2\}$$
$$\cup \{X^{(1)} \to X \mid X \in N_1\}$$
$$\cup \{X^{(2)} \to X^{(2)} \mid X \in N_1\}$$
$$\cup \{A_i \to A_i \mid A \in N_2, 1 \leqslant i \leqslant k\};$$

$$P_4 = \{(X, out) \mid X \in N_1\};$$

$$F_4 = \{(B^{(1)}, notout) \mid B \in N_2\}$$
$$\cup \{(X^{(2)}, notout) \mid X \in N_1\}$$
$$\cup \{(A_i, notout) \mid A \in N_2, 1 \leqslant i \leqslant k\};$$

$$R_5 = \{B^{(2)} \to B^{(2)} \mid B \in N_2\}$$
$$\cup \{X^{(2)} \to X \mid X \in N_1\}$$
$$\cup \{X^{(1)} \to X^{(1)} \mid X \in N_1\}$$
$$\cup \{A_i \to A_i \mid A \in N_2, 1 \leqslant i \leqslant k\};$$

$$P_5 = \{(X, out) \mid X \in N_1\};$$

$$F_5 = \{(B^{(2)}, notout) \mid B \in N_2\}$$
$$\cup \{(X^{(1)}, notout) \mid X \in N_1\}$$
$$\cup \{(A_i, notout) \mid A \in N_2, 1 \leqslant i \leqslant k\}.$$

The system works as follows:

The initial configuration of the system is $[_1 X A [_2 [_3 \,]_3]_2 [_4 \,]_4 [_5 \,]_5]_1$. Let us suppose that at a particular instance we have a configuration $[_1 X w_1 A w_2 [_2 [_3 \,]_3]_2 [_4 \,]_4 [_5 \,]_5]_1$. In order to simulate a matrix $m_i : (X \to Y, A \to x)$, we apply the rule $A \to x A_i$ so that the resulting

string can be sent to one of the inner membranes. If the string is sent to either membrane 4 or 5, the computation never halts. Otherwise, in membrane 2, we apply a rule of the form $X \to Y_j$ and send the resulting string to membrane 3. In membrane 3, we replace A_i with A'_i and Y_j with Y'_{j-1} and send the resulting string to membrane 2. In membrane 2, we replace A'_i with A_{i-1} and Y'_j with Y_j and send the resulting string to membrane 3. We repeat this process till the subscript of either A or Y becomes 1. If the subscript of A is smaller than that of Y, after certain steps of computation we have a configuration $[_1[_2 Y_{j-i} w_1 x A'_1 w_2 [_3\ ]_3]_2[_4\ ]_4[_5\ ]_5]_1$. We now erase the symbol A'_1 but the resulting string cannot be sent to membrane 1 due to the presence of Y_{j-i} and the computation never halts. On the other hand, if the subscript of Y is smaller than that of A, after certain steps of computation we have a configuration $[_1[_2 Y' w_1 x A_{i-j} w_2 [_3\ ]_3]_2[_4\ ]_4[_5\ ]_5]_1$. The string cannot be sent to membrane 3 due to the presence of Y' and the computation never halts. The only way in which we get the correct simulation is to have the same subscripts for both A and Y. In such a case, after certain steps of computation we have a configuration $[_1[_2 Y' w_1 x A'_1 w_2 [_3\ ]_3]_2[_4\ ]_4[_5\ ]_5]_1$. We now erase the symbol A'_1 and send the string to membrane 1, where we replace Y' with Y. In this way we complete the simulation of the matrix. The same procedure can be applied for simulating a matrix of type 4 except the last step, where we erase the symbol f'. If the resulting string contains any nonterminal, the computation never halts. Otherwise, the string is sent out.

In order to simulate a matrix $m'_i : (X \to Y, B^{(1)} \to \dagger)$, we apply the rule $X \to Y^{(1)}$ so that the resulting string can be sent to one of the inner membranes. If the string is sent to membrane 2 or 5, the computation never halts. Otherwise, in membrane 4, we replace the symbol $Y^{(1)}$ with Y. The resulting string can be sent out, only if it does not contain the symbol $B^{(1)}$. Hence we get the correct simulation of the matrix. In a similar way we can explain the simulation of matrices of the form $m' : (X \to Y, B^{(2)} \to \dagger)$.

The process can be iterated until no nonterminal is present in the sentential form. Hence, each derivation in G can be simulated in Π and, conversely, all halting computations in Π correspond to correct derivations in G. Therefore, the computation in Π can stop only after reaching a terminal string with respect to G. Thus, we have $L(G) = L(\Pi)$. □

5. Contextual processing

Instead of context-free rules for processing string-objects, contextual rules were considered in [7], where the derivations are taking place depending on the contexts. Combining context-free rules and contextual rules for processing string-objects, a new class of P systems, called *hybrid P systems*, were introduced in [5] and studied in [8]. In [8], it was shown that *four* membranes are needed for proving the universality for hybrid P systems with *finite* choice.

We improve the universality result for hybrid P systems with $finite$ choice and show that only $three$ membranes are enough. The improvement is achieved by considering a *type* 0 grammar in Penttonen normal form, instead of Kuroda normal form considered in [8].

Theorem 5.1. $RE = HyP_3(FIN, in)$.

Proof. Let $G = (N, T, S, P)$ be a *type* 0 grammar in Penttonen normal form. Assume that all non context-free rules in P are labelled in a one-to-one manner. We construct the hybrid P system

$$\Pi = (V, T, [_1[_2[_3\ ]_3]_2]_1, \emptyset, \emptyset, \{S\}, R_1, R_2, R_3, 3),$$

where:

- $V = N \cup T \cup \{A' \mid A \in N\} \cup \{[B, r] \mid r : AB \to CD \in P\}$;
- R_1 contains the following rules:
  (1) $[B, r] \to C'(in)$, for $r : AB \to AC \in P$;
- R_2 contains the following rules:
  (1) $(A[B, r], (\lambda, \lambda), out)$, for $r : AB \to AC \in P$;
  (2) $[B, r] \to [B, r](here)$;
  (3) $C' \to C(in)$, $C \in N$;
- R_3 contains the following rules:
  (1) $A \to x(here)$, for $A \to x \in P$;
  (2) $B \to [B, r](out)$, for $r : AB \to AC \in P$.

The system works as follows:

The initial configuration of the system is $[_1[_2[_3S]_3]_2]_1$. The context-free rules from P are present in R_3 as rewriting rules, hence we can easily simulate them. Let us assume that we have a string w_1ABw_2 in membrane 3. The only way to move the string to membrane 2 is by using a rule of the form $B \to [B, r](out)$. Assume that we use the rule $B \to [B, r](out)$ corresponding to some rule $r : AB \to AC \in P$. The resulting string is sent to membrane 2, where we can apply either $[B, r] \to [B, r](here)$ or a rule of the form $(A[B, r], (\lambda, \lambda), out)$ such that $r : AB \to AC \in P$. If the string does not contain the symbol A (which helps in applying the rule $(A[B, r], (\lambda, \lambda), out)$), the computation never halts due to the repeated application of the rule $[B, r] \to [B, r](here)$. In order to proceed further, the string should contain the symbol A so that we can apply the latter rule and send the string to membrane 1. We can send the string from membrane 1 to 2 by replacing the symbol $[B, r]$ to C' such that $r : AB \to AC \in P$. In membrane 2, we apply the rule $C' \to C(in)$ so that the string is sent to membrane 3. In this way we get the correct result of simulating the rewriting of the string w_1ABw_2 by means of the rule $r : AB \to AC \in P$.

The process can be iterated until no nonterminal is present in the sentential form. Hence, each derivation in G can be simulated in Π and, conversely, all halting computations in Π correspond to correct derivations in G. Therefore, the computation in Π can stop only after reaching a terminal string with respect to G. Thus, we have $L(G) = L(\Pi)$. □

6. Conclusion

In this paper we gave some improved results in rewriting P systems. It is an open problem whether or not the results of Theorems 3.2 and 4.2 can be improved. We conjecture that the result of Theorem 3.4 cannot be improved further.

Acknowledgements

I am thankful to Department of Science and Technology, New Delhi, Committee on Science and Technology in Developing Countries, Chennai, IIT Madras and IIT Madras Alumni Association for providing me financial assistance to attend Brainstorming Week on Membrane Computing, Tarragona, Spain, 05-11, February, 2003. I am also thankful to Gh. Păun for his valuable comments on the earlier version of this paper.

References

[1] P. Bottoni, A. Labella, C. Martín-Vide, Gh. Păun, Rewriting P systems with conditional communication, in: W. Brauer, H. Ehrig, J. Karhumäki, A. Salomaa (Eds.), Formal and Natural Computing. Essays Dedicated to Grzegorz Rozenberg, Lecture Notes in Computer Science, Vol. 2300, Springer, Berlin, 2002, pp. 325–353.
[2] J. Dassow, Gh. Păun, Regulated Rewriting in Formal Language Theory, Springer, Berlin, 1989.
[3] C. Ferretti, G. Mauri, Gh. Păun, C. Zandron, On three variants of P systems with string-objects, Theoret. Comput. Sci. 301 (1–3) (2003) 201–215.
[4] R. Freund, Gh. Păun, On the number of non-terminal symbols in graph-controlled, programmed, and matrix grammars, in: M. Margenstern, Y. Rogozhin (Eds.), Machines, Computations, and Universality, 3rd Internat. Conf. Lecture Notes in Computer Science, Vol. 2055, Springer, Berlin, 2001, pp. 214–225.
[5] S.N. Krishna, K. Lakshmanan, R. Rama, Hybrid P systems, Romanian J. Inform. Sci. Technol. 4 (1) (2001) 111–124.
[6] S.N. Krishna, R. Rama, P systems with replicated rewriting, J. Automata, Languages Combinator. 6 (3) (2001) 345–350.
[7] M. Madhu, K. Krithivasan, Contextual P systems, Fund. Inform. 49 (1–3) (2002) 179–189.
[8] M. Madhu, K. Krithivasan, A note on hybrid P systems, Grammars 5 (3) (2002) 239–244.
[9] M. Madhu, K. Krithivasan, A survey on some variants of P systems, in: Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron (Eds.), Membrane Computing, International Workshop, WMC-CdeA 2002, Lecture Notes in Computer Science, Vol. 2597, Springer, Berlin, 2003, pp. 360–370.
[10] V. Manca, C. Martin-Vide, Gh. Păun, On P systems with replicated rewriting, J. Automata, Languages Combinator. 6 (3) (2001) 359–374.
[11] C. Martin-Vide, Gh. Păun, String objects in P systems, Proc. Algebraic Systems, Formal Languages and Computations Workshop, RIMS Kokyuroku, Kyoto University, Kyoto, 2000, pp. 161–169.
[12] Gh. Păun, Computing with membranes, J. Comput. System Sci. 61 (1) (2000) 108–143.
[13] Gh. Păun, Membrane Computing. An Introduction, Springer, Berlin, 2002.
[14] A. Salomaa, Formal Languages, Academic Press, New York, 1973.