Acta Cybernetica **15** (2001) 53-74.

P Systems with Picture Objects

Shankara Narayanan Krishna * Raghavan Rama *

Kamala Krithivasan †

Abstract

New computability models called P systems, based on the evolution of objects in a membrane structure, were recently introduced. In this paper, we consider two variants of P systems having "complex objects" like pictures as the underlying data structure. The first variant is capable of generating pictures with interesting patterns. We also investigate the generative power of this variant by comparing it with the families of two dimensional matrix languages. The second variant has some applications in pattern generation.

1 Introduction

The name 'picture processing' is generally used to describe that area of computer science which is concerned with the analysis and generation of pictures. Pioneering work in suggesting and applying a linguistic model for the solution of nontrivial problems in picture processing was done by Narasimhan [4]. Narasimhan has also proposed and implemented schemes for the recognition of handprinted letters of the English alphabet and for the generation of poster pictures [5, 6].

Various classes of pictures have also been generated using grammars [11]. A matrix model to describe digital pictures viewed as matrices ($m \times n$ rectangular arrays of terminals) is given in [13]. There has been considerable interest in applying the methods of mathematical linguistics to picture generation and description. The concept of substitution of regular sets into languages of Chomsky type are defined in [13] and the concept of substitution of regular sets into L-systems are defined in [7]. When a picture is described as a rectangular array of terminals, it is advantageous to assign attribute values to members of the array, the typical attribute values being intensity or grey level, color and opaqueness or transparency [12].

P systems, introduced by Gh.Paun, [8] form a new class of biologically inspired distributed computing models. P systems can be used as a support for a computing device based on any type of objects and any type of evolution rules associated with them. In basic variants of P systems, the objects are represented either by symbols

*Department of Mathematics Indian Institute of Technology, Madras E-mail:ramar@iitm.ac.in

†Department of Computer Science and Engineering, Indian Institute of Technology, Madras E-mail:kamala@iitm.ernet.in

from a given alphabet or by strings over a given alphabet. In the case of string objects, the objects can evolve in many ways defined by string processing rules [8], [10] such as rewriting (sequential and parallel)[2], point mutations and so on.

Generalizing the passing from symbol objects to string objects, an immediate further step is to consider still more complex objects, such as trees, graphs of arbitrary forms, arrays etc, [9]. Such generalizations are classic in language theory (graph grammars, array grammars and even picture grammars are well developed domains). So, it is natural to start considering P systems with complex object descriptions. With such motivation we introduce picture objects into P systems and evolve the pictures by specific rules.

In the following section, we give a few preliminary notions and notations. In Section 3, a variant of P systems with picture objects is introduced and in the next section, certain examples are given. In Section 5, the generative power of this variant is investigated by comparing it with the existing families of matrix languages $PSML, CSML, CFML$ and RML. In Section 6, it is proved that an online tesselation automata can be simulated using this variant. Another variant of P systems with picture objects is introduced in Section 7, and the application of this variant in pattern generation is illustrated with a few examples.

2 Preliminaries

In this section, we give some preliminaries which will be useful in subsequent sections.

A two dimensional string (or a picture) over an alphabet Σ is a two dimensional rectangular array of elements of Σ. The set of all two dimensional strings over Σ is denoted as Σ^{**}.

Given a picture $p \in \Sigma^{**}$, let $l_1(p)$ denote the number of rows of p and $l_2(p)$ denote the number of columns of p. The pair $(l_1(p), l_2(p))$ is called the size of the picture p. The empty picture is the only picture of size (0,0) and it will be denoted by λ. Pictures of size $(0,n)$ or $(n,0)$ where $n > 0$ are not defined. The set of all pictures over Σ of size (m,n) with $m,n > 0$ will be indicated by $\Sigma^{m \times n}$. Furthermore, if $1 \leq i \leq l_1(p)$ and $1 \leq j \leq l_2(p)$, $p_{i,j}$ denotes the symbol in p with coordinates (i,j).

Now we will give some simple examples of two dimensional languages.

1. Let $\Sigma = \{a\}$ be a one letter alphabet. The set of pictures of a's with three columns is a two dimensional language over Σ. It can be formally defined as $L = \{p \mid p \in \Sigma^{**} \text{ and } l_2(p) = 3\}$.

2. Take a two letter alphabet $\Gamma = \{0,1\}$, and consider the set of squares in which all letters in the main diagonal are 1, whereas the remaining positions carry letter 0. An example is the following one:

$$\begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

Two dimensional online tesselation automata ($2OTA$) introduced by Inoue and Nakamura is a particular model of two dimensional cellular automata ($2CA$) [1]. A $2OTA$ is a restricted type of $2CA$ in which cells do not make transitions at every time-step: rather a "transition wave" passes once diagonally across the array. Each cell changes its state depending on the two neighbors to the top and to the left, respectively.

A non-deterministic (deterministic) two-dimensional on-line tesselation automaton, referred to as $2OTA$ ($2DOTA$), is defined by $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$, where:
— Σ is the input alphabet;
— Q is a finite set of states;
— $I \subset Q$ ($I = \{i\} \subseteq Q$) is the set of "initial" states;
— $F \subseteq Q$ is the set of "final" (or "accepting") states;
— $\delta : Q \times Q \times \Sigma \to 2^Q$ ($\delta : Q \times Q \times \Sigma \to Q$) is the transition function.

#	#	#	#	#	#	#	#
#							#
#							#
#					$p_{i-1\ j}$		#
#				$p_{i\ j-1}$	$p_{i\ j}$		#
#							#
#	#	#	#	#	#	#	#

A run of $\mathcal{A}$ on a picture p consists of associating a state from Q to each position (i, j) of p. Such a state is given by the transition function δ and depends on the states already associated with positions $(i, j-1)$ and $(i-1, j)$ and on the symbol $p_{i,j}$. At time $t = 0$ an initial state q_0 is associated with all positions of the first row and of the first column of p. The computation consists of $l_1(p) + l_2(p) - 1$ steps. It starts at time $t = 1$ by reading $p_{1,1}$ and associating the state $\delta(q_0, q_0, p_{1,1})$ with position (1,1). At time $t = 2$, states are simultaneously associated with positions (1,2) and (2,1), and so on, to the next diagonals. At time $t = k$, states are simultaneously associated with each position (i, j) such that $i + j - 1 = k$. A $2OTA$ recognizes a picture p if there exists a run of $\mathcal{A}$ on p such that the state assigned to position $(l_1(p), l_2(p))$ is a final state.

Now we give an example of a language recognized by a $2OTA$. Let $\Sigma = \{a\}$ and let $L \subseteq \Sigma^{**}$ be the language of all pictures over Σ with an odd number of columns. That is, $L = \{p \mid l_2(p) \text{ is odd }\}$. A 2 OTA can recognize pictures of L by associating states "1" and "2" with the positions of each odd and even column, respectively. A picture is accepted if positions of the rightmost column contain state "1". More formally, L is recognized by the $2OTA$ $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ defined as follows:

$Q = \{0,1,2\}$, $I = \{0\}$, $F = \{1\}$,
$\delta(0,0,a) = \delta(0,2,a) = \delta(1,0,a) = \delta(1,2,a) = 1$, $\delta(0,1,a) = \delta(2,1,a) = 2$.

A *matrix grammar with appearance checking* is a construct $G = (N,T,S,M,F)$, where N,T are disjoint alphabets, $S \in N$, M, a finite set of sequences of the form $(A_1 \to x_1,\ldots,A_n \to x_n)$, $n \geq 1$, of context-free rules over $N \cup T$ (with $A_i \in N, x_i \in (N \cup T)^*$, in all cases), and F, a set of occurrences of rules in M (we say that N is the nonterminal alphabet, T is the terminal alphabet, S is the axiom, while the elements of M are called matrices).

For $w,z \in (N \cup T)^*$ we write $w \Longrightarrow z$ 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 $1 \leq i \leq n$, either $w_i = w_i' A_i w_i'', w_{i+1} = w_i' x_i w_i''$, for some $w_i', w_i'' \in (N \cup T)^*$, or $w_i = w_{i+1}$, 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; we say that these rules are applied in the *appearance checking* mode.) If $F = \emptyset$, then the grammar is said to be without appearance checking (and F is no longer mentioned). We denote by $\Longrightarrow^*$ the reflexive and transitive closure of the relation $\Longrightarrow$. The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$. The family of languages of this form is denoted by MAT_{ac}. When we use grammars without appearance checking, then the family obtained is denoted by MAT. It is known that $MAT \subset MAT_{ac} = RE$ and that each one-letter language in the family MAT is regular.

A matrix grammar $G = (N,T,S,M,F)$ with appearance checking is said to be in the *binary normal form* if $N = N_1 \cup N_2 \cup \{S,\#\}$, with these three sets mutually disjoint, and the matrices in M are 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. $(X \to Y, A \to \#)$, with $X,Y \in N_1, A \in N_2$,
4. $(X \to \lambda, A \to x)$, with $X \in N_1, A \in N_2$, and $x \in T^*$.

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

A 2D-matrix grammar is a 2-tuple

$$G = (G_1, G_2)$$

where

$G_1 = (H_1, I_1, P_1, S)$ *is a grammar,*
H_1 *is a finite set of horizontal nonterminals,* $H_1 \cap I_1 = \emptyset$,
$I_1 = \{S_1, S_2, \ldots, S_k\}$, *a finite set of intermediates,*
P_1 *is a finite set of production rules called horizontal production rules,*

S is the start symbol, $S \in H_1$,
$G_2 = (G_{2,1}, G_{2,2}, \ldots, G_{2,k})$,
$G_{2i} = (V_{2i}, I_2, P_{2i}, S_i)$, $1 \leq i \leq k$ are regular grammars,
I_2 is a finite set of terminals,
V_{2i} is a finite set of vertical nonterminals, $V_{2i} \cap V_{2j} = \emptyset, i \neq j$,
S_i is the start symbol,
P_{2i} is a finite set of right linear production rules.

The type of G_1 gives the type of G, so we speak about regular, context-free, context-sensitive, recursively enumerable 2D-matrix grammars if G_1 is a regular, context-free, context sensitive or arbitrary respectively. Derivations are defined as follows: First a string $S_{i_1} S_{i_2} \ldots S_{i_k} \in I_1^*$ is generated horizontally using the horizontal production rules P_1 in G_1. That is, $S \Longrightarrow S_{i_1} S_{i_2} \ldots S_{i_k} \in I_1^*$. Vertical derivations proceed as follows: We write

$$\boxed{A_{i_1} \ldots A_{i_n}} \Downarrow \boxed{\begin{matrix} a_{i1} \ldots a_{in} \\ B_1 \ldots B_n \end{matrix}}$$

if $A_{ij} \rightarrow a_{ij} B_j$ are rules in $P_{2,j}$, $1 \leq j \leq n$. The derivation terminates if $A_j \rightarrow a_{mj}$ are all terminal rules in G_2. The set $M(G)$ of all matrices generated by G is defined to be the set of $m \times n$ *arrays* $[a_{ij}]$ such that $1 \leq i \leq m, 1 \leq j \leq n$ and $S \Longrightarrow_{G_1}^* S_1 \ldots S_n \Longrightarrow_{G_2}^* [a_{ij}]$. Now we shall recall the definition of P systems with string objects from [8]. A rewriting P system of degree $m, m \geq 1$, is a construct

$$\Pi = (V, T, \mu, M_1, M_2, \ldots, M_m, (R_1, \rho_1), (R_2, \rho_2), \ldots, (R_m, \rho_m))$$

where V is the total alphabet, $T \subseteq V$ is the output alphabet, μ is a membrane structure consisting of m membranes labeled with $1, 2, \ldots, m$, $M_1, \ldots, M_m$, are finite languages over V associated with the regions $1, 2, \ldots, m$ of μ, $R_1, \ldots, R_m$ are finite sets of developmental rules over V associated with the regions $1, 2, \ldots, m$ of μ and $\rho_1, \ldots, \rho_m$ are partial order relations over $R_1, \ldots, R_m$, specifying a priority relation among the rules. The rules in R_i are of the form $X \rightarrow (v, tar)$, where $tar \in \{here, out, in_j\}$; j is the label of a membrane. In each step, each string which can be rewritten is rewritten (in a context-free manner) by a rule from its region. The rule is nondeterministically chosen, and the resulting string will be moved to the membrane indicated by the target *tar* associated with the rule: *here* means that the string remains in the same membrane, *out* means that the string exits the membrane, and in_j means that the string goes to the membrane with label j providing that it is directly inside the membrane where the rule $X \rightarrow (v, in_j)$ is applied. This way, we pass from a given configuration to another one; a sequence of transitions forms a computation. We consider as successful computations only the halting ones (computations which reach a configuration where no further rule

can be applied); the result of a halting computation consists of all strings over T^* which are sent out of the system during the computation.

3 Basic definitions

Here, we directly define the system which we are going to work with.
Definition 3.1 A P System of degree n, $n \geq 1$ with picture objects is a construct

$$\Pi = (V, T, \mu, S', M_1, M_2, \ldots, M_n, (R_1, \rho_1), (R_2, \rho_2), \ldots, (R_n, \rho_n)),$$

where

- V is the total alphabet of the system, $V = \bigcup_{i=1}^{n} H_i \cup \bigcup_{i=1}^{n} I_i \cup \bigcup_{i=1}^{n} V_i \cup U \cup T$, where H_j, I_j and V_j are the set of horizontal nonterminals, intermediates and vertical nonterminals associated with membranes j, $1 \leq j \leq n$. U is a set of nonterminals disjoint from any of H_j, I_j, V_j and T.
- $T \subseteq V$ is the output alphabet.
- μ is the membrane structure of degree n, with membranes labeled by $1, 2, \ldots, n$.
- S' is the set of horizontal and vertical scanners. If a membrane has scanners in it, then any picture in the membrane can be scanned. The presence of scanners in a membrane is optional.
- Each M_j is the union of three finite sets, $M_j = S'_j \cup L'_j \cup U_j$, where S'_j is the set of scanners associated with membrane j, L'_j is a finite language over V associated with membrane j and U_j is a multiset of objects over U associated with membrane j.
- R_j and ρ_j are the set of rewriting rules and partial order relations associated with the regions j, $1 \leq j \leq n$, of μ; the form of rules will be specified below.

$M_1, M_2, \ldots, M_n$ can be empty and the same is valid for $R_1, R_2, \ldots, R_n$ and their associated priority relations $\rho_1, \rho_2, \ldots, \rho_n$. Now we shall explain how the scanners work.

The scanners are of two types : horizontal and vertical. The horizontal scanners scan the rows and the vertical scanners scan the columns. Suppose that there are m horizontal scanners $h_1, h_2, \ldots, h_m$ and n vertical scanners $v_1, v_2, \ldots, v_n$ in a membrane k. If p is a picture of size $m \times n$, present in k, then h_i scans the ith row of p and v_j scans the jth column. The horizontal scanner h_i starts scanning the ith row from the left boundary. That is from position $(i, 1)$. Similarly, the vertical scanner v_j starts scanning the jth column from the position $(1, j)$.

At any point of time, the horizontal scanners h_i and vertical scanners v_j hold information about the current element under scan and the previous element scanned. By h_i (or v_j) scanning the ith row (jth column), we mean that h_i (or v_j) is reading

the elements of the ith row (jth column) starting from the left end (top) and proceeds in a systematic way reading one element in each step till it reaches the last element in the ith row (jth column). The scanners $h_1, \ldots, h_m, v_1, \ldots, v_n$ can start working simultaneously unless there are priority relations specifying the order in which they should work.

To specify the position in the picture where the scanning is done, we denote by $h_i(\lambda, p_{i,1})$ (or $v_j(\lambda, p_{1,j})$), the scanners h_i and v_j, during the first step of scanning. This indicates that the current elements under scan are the elements at positions $(i,1)(\ (1,j)\ )$. The scanners h_i, v_j can continue the scanning in the next step, provided there are no rules (horizontal, vertical or transition) having a higher priority. If the scanning is continued in the next step, $h_i(\lambda, p_{i,1})$ $(v_j(\lambda, p_{1,j}))$ is changed to $h_i(p_{i,1}, p_{i,2})$ $(v_j(p_{1,j}, p_{2,j}))$. This means that $p_{i,2}(\ p_{2,j}\ )$ and $p_{i,1}(\ p_{1,j}\ )$ are the current (previous) elements scanned by h_i and v_j respectively.

Since each scanner has information about the current element and its previous one, two operations : *p extract* and *c extract* are defined on them. *p extract* stands for extracting the previous element scanned by the scanner and *c extract* stands for extracting the current element scanned. These are denoted as follows: The operation *p extract* denoted as $h_i^p(p_{i,j}, p_{i,j+1})$ $(v_i^p(p_{j,i}, p_{j+1,i}))$ gives $p_{i,j}$ $(p_{j,i})$, and the operation *c extract* denoted as $h_i^c(p_{i,j}, p_{i,j+1})$ $(v_i^c(p_{j,i}, p_{j+1,i}))$ gives $p_{i\ j+1}$ $(p_{j+1\ i})$. These operations are useful for accessing and changing any particular element in a given picture. For instance, if we want to change the element at position (i,j) in a picture, there are two ways: (1) One way is to use the horizontal scanner h_i and wait till it reaches the jth element. Once it reaches the jth element, $h_i^c(p_{i,j-1}, p_{i,j})$ contains the element at position (i,j). At this step, if an assignment of the form $h_i^c(p_{i,j-1}, p_{i,j}) \to t$ is made, then the element at position (i,j) is changed to t. A similar way is to use the rule $h_i^p(p_{i,j}, p_{i,j+1}) \to t$. (2) Using the vertical scanner v_j also, this can done. Rules of the form $h_i^p(\lambda, p_{i,1}) \to t$ are not valid as there is no element to the left of the first element in any row. Similarly, rules of the form $v_j^p(\lambda, p_{1,j}) \to t$ are also not valid as there are no elements on top of the first element in any column. The work of each scanner h_i (*or* v_j) comes to an end after it has finished scanning all the elements in the ith row (jth column). After this it can start scanning again from the first position of each row (column).

Now we shall explain the three kinds of rules: horizontal, vertical and transition.

Each membrane j has a set I_j of intermediates, horizontal nonterminals H_j and vertical nonterminals V_j and rules associated with each of them. In addition, we also have a set of nonterminals U which can be associated with some or all of the membranes. Also, U is disjoint from all H_j, V_j and I_j. Note that $H_j \cap V_j = \emptyset, H_j \cap I_j = \emptyset$ for all j but it is not necessary that $H_i \cap I_j = \emptyset$ or $H_i \cap V_j = \emptyset$ or $H_i \cap H_j = \emptyset$ or $V_i \cap V_j = \emptyset$ or $I_i \cap V_j = \emptyset$ or $I_i \cap I_j = \emptyset$ for $i \neq j$. Horizontal rules are associated with nonterminals, vertical rules are associated with the intermediates and vertical nonterminals and transition rules are associated with the elements of U. By default, preference is given to the horizontal rules over the vertical rules in each membrane. A string over the intermediates is generated using the horizontal rules after which, the vertical rules are applied. The transition rules are applicable always

unless they are controlled by priority relations. With suitable priority relations, the transition rules can be used to control rules involving scanners; viz., rules of the form $h_i^c(a_{i,j}, a_{i,j+1}) \to t\,(h_i^p(a_{i,j}, a_{i,j+1}) \to t)$ or $v_j^c(a_{i-1,j}, a_{i,j}) \to t\,(v_j^p(a_{i-1,j}, a_{i,j}) \to t)$.

The horizontal rules in each membrane j are of the following two forms:

1. Rules of the form $\alpha \to (\beta_1\gamma_1, tar_1)\dots(\beta_n\gamma_n, tar_n)$ or $\alpha \to (\beta_1, tar_1)\dots(\beta_n, tar_n)$ where $\alpha, \gamma_i \in H_j, \beta_i \in I_j^*$ and tar_i is one of $\{here, out, in_k\}$, k is a membrane adjacent to j. Rules of this form are called right linear rules. The elements of I_j formed during the evolution are terminals with respect to horizontal derivation rules, since only vertical rules are applicable to them.

2. Rules of the form $\alpha \to (\beta_1, tar_1)\dots(\beta_n, tar_n)$ where $\alpha \in H_j$ and $\beta_i \in (H_j \cup I_j)^*$. Rules of this form are called context free rules. As above, tar_i has to be one of *here*, *out* or in_k for some membrane k adjacent to j.

If $n > 1$ in the above rules of types 1, 2, we say that the horizontal rules are replication rules. The horizontal rules are applied sequentially: we apply exactly one rule to a string at a time. If the number of horizontal nonterminals in a string is more than one and if there are applicable rules to each one of them, then a horizontal nonterminal is chosen nondeterministically and one occurrence of it is replaced. The parallelism of the system refers to applying rules to strings in all membranes. If a rule of the form $\alpha \to (\beta\gamma, tar)$ is applied, the string moves as a whole to the membrane indicated by tar after replacing α by $\beta\gamma$. Since preference is given to the horizontal rules in each membrane and each horizontal rule gives rise to some intermediates, a string over intermediates is obtained when the horizontal derivations come to a halt.

The intermediates are the start symbols for vertical rules in all membranes. The vertical rules are applied in parallel: that is, all the intermediates or vertical nonterminals which can be replaced in a step should be replaced. The vertical rules in a membrane j are of the form $\alpha \to (\beta\gamma, tar)/(\beta, tar)$ where $\alpha \in I_j$ or V_j, $\beta \in T^*, \gamma \in V_j$ or I_j and tar is one of $\{here, out, in_k\}$ where k is a membrane adjacent to j. Hence, the vertical rules are always right linear.

The vertical rules result in the string growing downward; that is if there is a string XYZ in a membrane and if there are vertical rules $X \to +++, Y \to .++, Z \to ..+$, then we get the picture

$$\begin{array}{ccc} + & . & . \\ + & + & . \\ + & + & + \end{array}$$

Assume that we rewrite in this way n symbols, by rules which have associated targets of the form $here, out, in_j$ of several types and that there are n_{tar} targets of each type. Then, the structure obtained by rewriting is sent to the membrane indicated by the target with the maximal n_{tar} (the target which appears the maximal number of times in the used rules). As usual, when several targets have the same maximal number of occurrences, then one of them is nondeterministically

chosen. For instance, if we have the structure given below in some membrane k and rules $A_1 \to (\varpi, in_1), A_2 \to (\varsigma, in_2), A_3 \to (\gamma, in_3), A_4 \to (\kappa, in_4)$ where $1, 2, 3, 4$ are membranes adjacent to k, then

$$\begin{array}{cccc} a & A_2 & c & d \\ e & & f & h \\ A_1 & & A_3 & i \\ & & & A_4 \end{array} \quad \Longrightarrow \quad \begin{array}{cccc} a & \varsigma & c & d \\ e & & f & h \\ \varpi & & \gamma & i \\ & & & \kappa \end{array}$$

and the resultant structure can be moved to any of the membranes since each target has occurred an equal number of times.

The priority among rules in any membrane can be summarized as follows: All horizontal rules have higher priority than vertical rules; once all horizontal derivations are over and a string over intermediates is obtained, all symbols in the string which can be rewritten are rewritten according to priority relations that exist between the vertical rules.

In any membrane i, the transition rules for objects of U are of the form $u \to v$, where $u \in U$ and $v = v'$ or $v = v'\delta$, where v' is a string over $(U \times \{here, out\}) \cup (U \times \{in_j \mid 1 \leq j \leq n\})$, j is a membrane adjacent to i and δ is a special symbol not in V. The strings u, v are understood as representations of multisets over U. We use these rules mainly for controlling the action of the scanners. Thus, the horizontal and vertical rules are rewriting rules whereas, the transition rules are evolution rules [8].

Thus, this variant of P systems has a combination of symbol objects described by U and string objects over $H_j, 1 \leq j \leq n$. By applying transition rules to objects of U, horizontal rules to strings over H_j and then vertical rules to objects over $I_j \cup V_j$, pictures are developed. If a picture purely over T is obtained by applying rules of the above type and if it is sent out of the system at the end of a halting configuration, we list it in the language generated by the system. Any picture not over T or any non rectangular array over T sent out of the system are not listed in the language. Similarly, pictures over T remaining in the system after a halting configuration will not be included in the language generated.

If the horizontal rules in all membranes are right linear, then we say that the P system with picture objects uses right linear rules for its horizontal derivations. If the horizontal rules used in some membranes are right linear and the horizontal rules used in some membranes are context free, we say that the P system with picture objects uses context free rules for its horizontal derivations.

The work of this class of systems can be summarized as follows: in each time unit, each string which can be rewritten is rewritten using the horizontal rules; once a string over intermediates is formed, vertical rules are applied simultaneously to all symbols which can be rewritten. In addition, transition rules can be applied to symbols of U present in all the membranes. We thus pass on from one configuration to another one. A sequence of transitions form a computation, and we consider as successful computations the halting ones (the computations which reach a configuration where no rule can be applied); the result of a halting configuration consists of all pictures over T sent out of the system during the computation.

We denote by $P(\Pi)$ the language generated by a P system Π with picture objects; the family of all languages of this type, generated by systems with at most m membranes, using right linear rules (context free rules) for its horizontal derivations, is denoted by $RLP_m(\Pi)(CFP_m(\Pi))$; if no bound on the number of membranes is considered, then the subscript m is replaced by $*$.

4 Examples

Example 4.1 In this example, we generate a four pronged fork without handle, which is known to be generated only by a $CSMG$ [13]. Consider a P system with picture objects, having no priorities and using only right linear rules for its horizontal derivations,

$$\Pi = (V, T, \mu, \lambda, M_1, M_2, M_3, M_4, M_5, (R_1, \phi), (R_2, \phi), (R_3, \phi), (R_4, \phi), (R_5, \phi))$$

with

$$\begin{aligned}
V &= \{S, X, S_1', Y_1, S_1'', Y_2, Y_3, \lambda', \lambda'', S_1, S_2, Y, Y', Y'', V_1', V_2'\} \cup \{\star, .\},\\
T &= \{\star, .\},\\
M_1 &= \{S\},\ M_i = \lambda, i \neq 1,\\
\mu &= [_1\ [_2]_2\ [_3[_4[_5]_5]_4]_3\ ]_1,\\
H_1 &= \{S, X, S_1', Y_1, S_1'', Y_2, Y_3, \lambda', \lambda''\},\\
I_1 &= \{S_1, S_2, Y, Y', Y''\},\ V_1 = \{V_1', V_2'\},\\
H_2 &= \{Y', Y''\}, I_2 = \{\lambda\},\ V_2 = \{\lambda\},\\
H_3 &= \{Y, Y_1\}, I_3 = \{S_2\}, V_3 = \{\lambda\},\\
H_4 &= \{Y', Y_1'\}, I_4 = \{S_2\}, V_4 = \{\lambda\},\\
H_5 &= \{Y''\}, I_5 = \{S_2\}, V_5 = \{\lambda\},\\
R_1 &= \{S \to S_1X,\ X \to YS_1',\ S_1' \to S_1Y_1,\ Y_1 \to Y'S_1'', S_1'' \to S_1Y_2\}\\
&\cup \{Y_2 \to (Y''S_1, in_3),\ Y_3 \to (\lambda, in_2),\ \lambda' \to (\lambda, in_2),\ \lambda'' \to \lambda\}\\
&\quad (horizontal\ rules)\\
&\cup \left\{S_1 \to \begin{matrix}\star\\ V_1'\end{matrix},\ V_1' \to \begin{matrix}\star\\ V_1'\end{matrix},\ S_2 \to \begin{matrix}\star\\ V_2'\end{matrix},\ V_2' \to \begin{matrix}.\\ V_2'\end{matrix}\right\}\\
&\cup \{V_1' \to (\star, out), V_2' \to (., out)\}(vertical\ rules),\\
R_2 &= \{Y' \to (\lambda', out),\ Y'' \to (\lambda'', out)\}(horizontal\ rules),\\
R_3 &= \{Y \to (S_2Y_1, in_4),\ Y_1 \to (Y_3, out),\ Y_1 \to Y\}(horizontal\ rules),\\
R_4 &= \{Y' \to (S_2Y_1', in_5),\ Y_1' \to (Y', out)\}(horizontal\ rules),\\
R_5 &= \{Y'' \to (S_2Y'', out)\}(horizontal\ rules).
\end{aligned}$$

In membrane two, the string $S_1S_2^nS_1S_2^nS_1S_2^nS_1$ is generated as a result of horizontal derivations. This string is passed on to the skin membrane where as a result of vertical derivations, we get the four pronged fork without handle.

A four pronged fork without handle

Example 4.2 Consider the P system with picture objects having no priorities, no scanners and using right linear rules for its horizontal derivations,

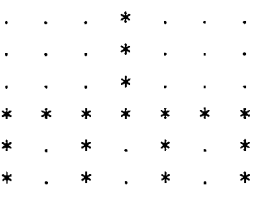
$$\Pi = (V, T, \mu, \lambda, M_1, M_2, \ldots, M_{13}, (R_1, \phi), (R_2, \phi), \ldots (R_{13}, \phi))$$

with

$$\begin{aligned}
V &= \{S, X, Y', Y_2', Y_2'', Y_2''', Y_2^4, Y_2^5, S_1, S_2, S_3, S_1', S_2', S_3', Y'', Y_1'', Y''', Y_1''', Y^4, \\
&\quad Y_1^4, Y^5, Y_1^5, Y^6\}, \\
T &= \{., \star\}, \\
\mu &= [_1\ [_2[_3[_4[_5[_6\ ]_6]_5]_4]_3]_2\ [_7\ ]_7[_8\ ]_8[_9[_{10}[_{11}[_{12}[_{13}\ ]_{13}]_{12}]_{11}]_{10}]_9\ ]_1, \\
M_1 &= \{S\}, M_i = \lambda, i \neq 1, \\
H_1 &= \{S, X, Y', Y_2'\}, I_1 = \{S_1, S_2, S_3, S_1', S_2', S_3'\}, V_1 = \{\lambda\}, \\
H_2 &= \{Y'', Y_1''\}, I_2 = \{S_3\}, V_2 = \{\lambda\}, \\
H_3 &= \{Y''', Y_1'''\}, I_3 = \{S_3\}, V_3 = \{\lambda\}, \\
H_4 &= \{Y^4, Y_1^4\}, I_4 = \{S_3\}, V_4 = \{\lambda\}, \\
H_5 &= \{Y^5, Y_1^5\}, I_5 = \{S_3\}, V_5 = \{\lambda\}, \\
H_6 &= \{Y^6\}, I_6 = V_6 = \{\lambda\}, \\
H_7 &= \{\lambda\}, I_7 = \{S_1, S_2, S_3\}, V_7 = \{S_1', S_2', S_3'\}, \\
H_8 &= \{X, S_3', Y_1'', S_1', Y_1''', S_2', Y_1^4, S_1'', Y_1^5, S_3'', Y_1^6\}, \\
I_8 &= \{Y', S_3, Y'', S_1, Y''', S_2, Y^4, Y^5, Y^6\}, V_8 = \{\lambda\}, \\
H_9 &= \{Y'', Y_2''\}, I_9 = V_9 = \{\lambda\}, \\
H_{10} &= \{Y''', Y_2'''\}, I_{10} = V_{10} = \{\lambda\}, \\
H_{11} &= \{Y^4, Y_2^4\}, I_{11} = V_{11} = \{\lambda\}, \\
H_{12} &= \{Y^5, Y_2^5\}, I_{12} = V_{12} = \{\lambda\}, \\
H_{13} &= \{Y^6\}, I_{13} = V_{13} = \{\lambda\}, \\
R_1 &= \{S \to (S_1 X, in_8),\ Y' \to (S_3 Y_1', in_2),\ Y' \to (Y_2', in_9),\ Y_2' \to \lambda\} \\
&\quad (\textit{horizontal rules}) \\
&\cup \left\{S_1 \to \begin{matrix} \cdot \\ S_1 \end{matrix},\ S_3 \to \begin{matrix} \cdot \\ S_3 \end{matrix},\ S_2 \to \begin{matrix} \star \\ S_2 \end{matrix}\right\} \\
&\cup \left\{S_1 \to \begin{pmatrix} \star \\ S_1 \end{pmatrix}, in_7\right), S_3 \to \left(\begin{pmatrix} \star \\ S_3 \end{pmatrix}, in_7\right), S_2 \to \left(\begin{pmatrix} \star \\ S_2 \end{pmatrix}, in_7\right)\right\}
\end{aligned}$$

$$
\begin{aligned}
&\cup \quad \{S_2' \to (.,out),\ S_3' \to (.,out),\ S_1' \to (\star,out)\}(\textit{vertical rules}),\\
R_2 &= \{Y'' \to (S_3Y_1'',in_3),\ Y_1'' \to (Y'',out)\}(\textit{horizontal rules}),\\
R_3 &= \{Y''' \to (S_3Y_1''',in_4),\ Y_1''' \to (Y''',out)\}(\textit{horizontal rules}),\\
R_4 &= \{Y^4 \to (S_3Y_1^4,in_5),\ Y_1^4 \to (Y^4,out)\}(\textit{horizontal rules}),\\
R_5 &= \{Y^5 \to (S_3Y_1^5,in_6),\ Y_1^5 \to (Y^5,out)\}(\textit{horizontal rules}),\\
R_6 &= \{Y^6 \to (S_3Y^6,out)\}(\textit{horizontal rules}),\\
R_7 &= \left\{S_1 \to \begin{matrix}\star\\ S_1'\end{matrix},\ S_2 \to \begin{matrix}.\\ S_2'\end{matrix},\ S_3 \to \begin{matrix}.\\ S_3'\end{matrix}\right\}\\
&\cup \quad \left\{S_1 \to \begin{pmatrix}\star & , & out\\ S_1' & & \end{pmatrix}, S_2 \to \begin{pmatrix}. & , & out\\ S_2' & & \end{pmatrix}, S_3 \to \begin{pmatrix}. & , & out\\ S_3' & & \end{pmatrix}\right\}\\
&\quad (\textit{vertical rules}),\\
R_8 &= \{X \to Y'S_3',\ S_3' \to S_3Y_1'',\ Y_1'' \to Y''S_1',\ S_1' \to S_1Y_1'''\}\\
&\cup \quad \{Y_1''' \to Y'''S_2', S_2' \to S_2Y_1^4,\ Y_1^4 \to Y^4S_1'',\ S_1'' \to S_1Y_1^5\}\\
&\cup \quad \{Y_1^5 \to Y^5S_3'',\ S_3'' \to S_3Y_1^6,\ Y_1^6 \to (Y^6S_1,out)\}(\textit{horizontal rules}),\\
R_9 &= \{Y'' \to (Y_2'',in_{10}), Y_2'' \to (\lambda,out)\}(\textit{horizontal rules}),\\
R_{10} &= \{Y''' \to (Y_2''',in\ 11), Y_2''' \to (\lambda,out)\}(\textit{horizontal rules}),\\
R_{11} &= \{Y^4 \to (Y_2^4,in_{12}), Y_2^4 \to (\lambda,out)\}(\textit{horizontal rules}),\\
R_{12} &= \{Y^5 \to (Y_2^5,in_{13}), Y_2^5 \to (\lambda,out)\}(\textit{horizontal rules}),\\
R_{13} &= \{Y^6 \to (\lambda,out)\}(\textit{horizontal rule}).
\end{aligned}
$$

This system generates a four pronged fork with handle which cannot be generated even by a $PSMG$, [13]. Initially, we have the start symbol S in the skin. Then, $S_1Y'S_3Y''S_1Y'''S_2Y^4S_1Y^5S_3Y_1^6$ is generated in membrane 8 and in the next step, the string $S_1Y'S_3Y''S_1Y'''S_2Y^4S_1Y^5S_3Y^6S_1$ is sent to the skin by applying the rule $Y_1^6 \to (Y^6S_1,out)$. This string then passes through membranes 2,3,4, 5 and 6 in order, applying rules for $Y, Y', \ldots, Y^6$ and comes back to the skin. This cycle can then be repeated or terminated by applying the rule $Y' \to (Y_2',in_9)$. The string then passes through membranes 9, 10, 11, 12, 13 and comes back to the skin in the form $S_1S_3^{2n+1}S_1S_3^nS_2S_3^nS_1S_3^{2n+1}S_1$. Now vertical rules can be applied suitably to get a four pronged fork with width n in between the prongs.



A four pronged fork with handle

5 The Generative Power

In this section, we compare the power of P systems with picture objects with the families of 2D matrix languages.

Theorem 5.1 *A P system with picture objects, having horizontal scanners h_1, no priorities and using only right linear rules for its horizontal derivations can generate any picture which belongs to the families of $PSML$, $CSML$, $CFML$ and RML.*

Proof. Since all the vertical derivations are right linear by definition, and the highest family of matrix languages is obtained using a $PSMG$ where the horizontal rules used are of type-0, we prove here that a P system with picture objects having horizontal scanners and which uses only right linear rules for its horizontal derivations can generate as a result of its horizontal rules any string belonging to a type-0 language. A P system with picture objects with the above property can then simulate any $PSMG$, the vertical rules of the P system being the same as that of the $PSMG$.

Now we construct a P system with picture objects, having horizontal scanners, no priorities and using only right linear horizontal rules to generate any type-0 language by its horizontal derivations. Let $G = (N, T, S, P)$ be a type-0 grammar in Kuroda normal form. So the rules in G are of the following forms:

(1) $X \to AB, X, A, B \in N$

(2) $XY \to AB, X, Y, A, B \in N$

(3) $X \to a, X \in N, a \in T$

(4) $X \to \lambda, X \in N$

For any $X \in N$, let k be the maximum number of rules of the form $XY \to AB$. For $X \in N$, if there exist nonterminals $X_1, X_2, \ldots, X_k$ in G having rules $XX_i \to \alpha$, $\alpha \in N^+, \mid \alpha \mid = 2$, then we associate to X the membrane structure $\mu_X = [x[x_{1X}]x_{1X}\ [x_{2X}]x_{2X} \cdots [x_{kX}]x_{kX}]x$. Construct the P system with picture objects Π having degree less than or equal to $n(k+1)+2$, $\mid N \mid = n$, given by

$(V, T', \mu, S', M_1, M_{1'}, M_X, M_{X_{1X}}, \ldots, M_{X_{kX}}, \ldots, M_{Z_{kZ}},$
$(R_1, \phi), (R_{1'}, \phi), (R_X, \phi), \ldots, (R_{Z_{kZ}}, \phi))$
with

$V = N \cup T \cup T' \cup \{X', A_X \mid A, X \in N, XY \to AB \in P\},$

$T' = T \cup V_1,$

$\mu = [_1\ \mu_X\ \mu_Y \ldots \mu_Z\ ]_{1'}\ ]_1$, *the number of sub membrane constructs μ_X embedded within μ depends on the number of nonterminals X in the given grammar that have rules of the form $XY \to AB$,*

$S' = \{h_1\},$

$M_1 = \{S\}, M_X = \{h_1 \mid X \in N\},\ M_i = \{\lambda\},\ \forall$ *other* $i,$

$$
\begin{aligned}
H_1 &= \{X \mid X \in N\},\\
I_1 &= \{a \mid a \in T\} \cup \{A' \mid A \in N\} \cup \{A_X \mid XY \to AB \in P, A, X \in N\},\\
V_1 &= \{Any\ set \mid H_1 \cap V_1 = I_1 \cap V_1 = \emptyset\},\\
H_{1'} &= \{A' \mid A \in N\},\ I_{1'} = \{A \mid A \in N\},\ V_{1'} = \{\lambda\},\\
H_X &= H_1 \cup \{A' \mid A \in N\},\ I_X = \{A \mid A \in N\},\ V_X = \{\lambda\},\\
H_{Y_X} &= \{A_X \mid XY \to AB \in P, A, X \in N\},\ I_{Y_X} = \{A' \mid A \in N\},\ V_{Y_X} = \{\lambda\},\\
R_1 &= \{X \to a, \lambda \mid X \to a, \lambda \in P, X \in N, a \in T\}\\
&\cup \{X \to (A'B, in_{1'}) \mid X \to AB \in P\} \cup \{X \to (A_X, in_X) \mid XY \to AB \in P\}\\
&\cup \{X \to X \mid X \in N\ and\ has\ no\ rules\ in P\}(horizontal\ rules)\\
&\cup \{Arbitrary\ vertical\ rules\},\\
R_{1'} &= \{A' \to (A, out) \mid A \in N\},\\
R_X &= \{Y \to (B, in_{Y_X}) \mid h_1^p(an\ element\ of\ V, Y) = A_X\ and\ XY \to AB \in P\}\\
&\cup \{A' \to (A, out) \mid A \in N\},\\
R_{Y_X} &= \{A_X \to (A', out)\}.
\end{aligned}
$$

Initially, the start symbol S is in the skin membrane. Rules in G of the form $X \to a, \lambda$ where $a \in T$ are retained as such in R_1. A rule of the form $X \to AB$ is simulated by applying $X \to (A'B, in_{1'})$ where $A' \in I_1$. This makes the rule right linear with respect to the skin membrane. The rule $A' \to (A, out)$ is then applied to the string in membrane $1'$. Thus the rule $X \to AB$ in G is simulated correctly and the string is back in the skin membrane. To simulate the rule $XY \to AB$, the rule $X \to (A_X, in_X)$ is applied first. In membrane X, we have the horizontal scanner h_1. The rule $Y \to (B, in_{Y_X})$ is now applied only if the element to the left of Y in the string is A_X. The scanner scans the whole string and if there is atleast one configuration in which $h_1^p(a_{1n}, Y)$ equals A_X, that occurrence of Y is replaced and the string moves to membrane Y_X. Here, we convert the A_X into A' and send the string out. Back in membrane X, the rule $A' \to (A, out)$ is applied and the string goes back to the skin membrane after simulating the rule $XY \to AB$ correctly. Proceeding in this way, we get a string over I_1 in the skin membrane (Corresponding to getting a string over T in G). Now any vertical rule can be applied to generate a picture. Thus, any picture generated by the matrix languages $PSML, CSML, CFML$ and RML can be generated by a P system using horizontal scanners and right linear rules for its horizontal derivations. □

Theorem 5.2 *A P system with picture objects, no scanners, no priorities and using context free replicated rules for its horizontal derivations can generate any picture belonging to the families of $PSML, CSML, CFML$ and RML.*

Proof. It is proved in [3] that any string can be generated by a replicated rewriting P system with no priorities. In proving this result we considered a matrix grammar $G = (N, T, S, M, F)$ in the binary normal form with matrices $m_1, \ldots, m_k$ of type 2 or 4 and $m_{k+1}, \ldots, m_l$ of type 3 and constructed a P system having the membrane structure $\mu = [_1\ [_2\ ]_2 \ldots [_k\ ]_k\ [_{k+1}]_{k+1} \ldots [_l\ ]_l\ ]_1$.

Here, we construct a P system with picture objects having the membrane structure $[_0\ \mu\ ]_0$, with $H_0 = H_1 = N, H_i = N \cup \{X_i \mid X \in N, k+1 \leq i \leq l\}, H_i = N \mid 1 \leq i \leq k, V_j = \emptyset, j \neq 0, I_j = T, \forall j$, where T and N are respectively the set of terminals and non terminals of the given grammar G. The set V_0 can be chosen to be disjoint from H_0 and I_0 and the vertical rules in the skin membrane can be chosen arbitrarily. $M_1 = \{S\}$, the start symbol of G, $M_i = \emptyset, i \neq 1$. We introduce the horizontal rules $S \to XA$ in membrane one and $\{A \to A \mid A \in N\}$ in the skin membrane; the horizontal rules in the rest of membranes are the same as in [3]. The strings reaching membrane zero are the strings which were sent out of the system in the above lemma. If a string which is not purely over T^* (here I_1^*) reaches membrane zero, the rule $A \to A$ is applied for all nonterminals A occurring in the string and hence the computation never stops. Only when a string over I_1^* reaches the skin membrane, can the vertical rules be applied. In this way, any picture belonging to the families of $PSML, CSML, CFML$ and RML can be generated. □

Theorem 5.3 *A P system with picture objects having priorities for horizontal rules, no scanners and using context free rules for horizontal derivations can generate any picture belonging to the families of $PSML, CSML, CFML$ and RML with just 3 membranes.*

Proof. We recall the result $RE = ERP_2(Pri)$ proved in [2]. In proving this result we consider a matrix grammar $G = (N, T, S, M, F)$ in the binary normal form with matrices $m_1, \ldots, m_k$ of type 2 or 4 and $m_{k+1}, \ldots, m_l$ of type 3 and construct a P system having the membrane structure $\mu = [_1\ [_2\ ]_2\ ]_1$.

Here, we construct a P system with picture objects having the membrane structure $[_0 \mu]_0$, $H_0 = H_1 = H_2 = N$, $I_0 = I_1 = I_2 = T$, $V_0 =$ any set disjoint from H_0 and I_0, $V_1 = V_2 = \{\lambda\}, M_0 = M_2 = \{\lambda\}, M_1 = \{S\}$, *the start symbol of* G. The horizontal rules and priorities for membranes one and two are the same rules and priorities used in the proof of the theorem $RE = ERP_2(Pri)$. As in the above theorem, we add rules $\{A \to A \mid A \in H_0\}$ for all nonterminals in the skin membrane. This ensures that vertical derivations start only from a string over I_0^*. The vertical rules may be chosen arbitrarily. This way, any picture in the families of $PSML, CSML, CFML$ and RML can be generated by a 3-degree P system with picture objects. □

6 P Systems and OTA

In this section we simulate the action of an OTA using P systems with picture objects. We prove that given a $2DOTA$ recognizing or not recognizing a given picture, there exists a P system with picture objects that works exactly in the same way as the $2DOTA$.

Theorem 6.1 *Given a $2DOTA$ recognizing or not recognizing a picture, we can*

construct a two degree P system with picture objects having scanners which can simulate the 2DOTA.

Proof. Let $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ be a $2DOTA$ and let p be a picture of size $m \times n$. $\mathcal{A}$ recognizes p if there exists a run on p such that the state associated to the position (m, n) is a final state.
We construct a two degree P system with picture objects which simulates $\mathcal{A}$.
Let

$$\Pi = (V, \{t\}, S', [_1\ [_2\ ]_2\ ]_1, \{\lambda\}, \{p, c_0, h_1, h_2, \ldots, h_m, v_1, v_2, \ldots, v_n\}, (R_1, \phi), (R_2, \rho_2))$$

where p is the given picture of size $m \times n$ for which transitions of the $2DOTA$ are defined. $h_1, \ldots, h_m$ and $v_1, \ldots, v_n$ are the horizontal and vertical scanners in membrane two, which depend on the size of the given picture p.

$$\begin{aligned}
V &= \Sigma \cup Q \cup \{\gamma_1, \gamma_2, t_1, d, \dagger\} \cup \{d_q \mid q \in Q\} \cup \{c_i \mid 0 \le i \le m+n\} \cup \{t, t_1\},\\
U &= \{\dagger, \gamma_1, \gamma_2, d, t, t_1\} \cup \{d_q, c_i \mid q \in Q, 0 \le i \le m+n\},\\
S' &= \{h_1, h_2, \ldots, h_m, v_1, v_2, \ldots, v_n\},\\
H_1 &= I_1 = V_1 = \{\lambda\},\\
H_2 &= I_2 = V_2 = \{\lambda\},\\
R_1 &= \{t_1 \to (t, out)\}(\textit{transition rule}),\\
R_2 &= \{c_0 \to c_1\gamma_1, c_i \to c_{i+1}, 1 \le i \le m+n-2, c_{m+n-2} \to c_{m+n-1}d\}\\
&\cup \{\gamma_1 \to \gamma_2, \gamma_2 \to \gamma_1, \dagger \to \dagger, d \to d_{\delta(v_n^p(p_{m-1,n}, p_{m,n}),\ h_m^p(p_{m,n-1}, p_{m,n}), p_{m,n})}\}\\
&\cup \{d_q \to t_1\delta \mid q \in F\} \cup \{d_q \to \dagger \mid q \notin F\}(\textit{transition rules}).
\end{aligned}$$

For $1 \le j \le m+n-1,\ 1 \le i \le m$, scanner rules:

$$h_i^c(p_{i,j-1}, p_{i,j}) \to \begin{cases}
\delta(v_j^p(p_{i-1,j}, p_{i,j}),\ h_i^p(p_{i,j-1}, p_{i,j}), p_{i,j}) & \text{if } v_j^p(p_{i-1,j}, p_{i,j}),\ h_i^p(p_{i,j-1}, p_{i,j}) \ne \lambda\\
\delta(q_0,\ h_i^p(p_{i,j-1}, p_{i,j}), p_{i,j}) & \text{if } v_j^p(p_{i-1,j}, p_{i,j}) = \lambda, h_i^p(p_{i,j-1}, p_{i,j}) \ne \lambda\\
\delta(v_j^p(p_{i-1,j}, p_{i,j}), q_0, p_{i,j}) & \text{if } h_i^p(p_{i,j-1}, p_{i,j}) = \lambda, v_j^p(p_{i-1,j}, p_{i,j}) \ne \lambda\\
\delta(q_0, q_0, p_{i,j}) & \text{otherwise}
\end{cases}$$

$$\begin{aligned}
\rho_2 = \{&h_{i-1} > h_i, v_{i-1} > v_i\ \forall i;\ \gamma_1 \to \gamma_2 > \textit{all scanning rules};\\
&\gamma_2 \to \gamma_1 > \begin{cases} d \to d_{\delta(\ v_n^p(p_{m-1,n},\ p_{m,n}),\ h_m^p(p_{m,n-1},\ p_{m,n}),\ p_{m,n})},\\ \textit{all rules with LHS } c_j \textit{ and } h_i^c(p_{i,l-1}, p_{i,l}) \end{cases}\\
&c_i \to c_{i+1} > \textit{all rules with LHS } h_j^c(p_{j,l-1}, p_{j,l}) \textit{ such that } i+1 \ne j+l\}.
\end{aligned}$$

We start the proof by giving the details of the rules applied during the first few steps.
Step 0 :We have an $m \times n$ picture and a counter c_0 in membrane two.
Step 1 : the scanners h_1 and v_1 start working and $c_0 \to c_1\gamma_1$ is applied. So we have in the second membrane $h_1(\lambda, p_{1,1})$ and $v_1(\lambda, p_{1,1})$ in addition to c_1 and γ_1.
Step 2 : the rules $\gamma_1 \to \gamma_2$, $c_1 \to c_2$ and $h_1^c(\lambda, p_{1,1}) \to \delta(q_0, q_0, p_{1,1})$ are applied. In

this step, we assign to $p_{1,1}$ the state $\delta(q_0, q_0, p_{1,1})$ of the $2DOTA$ by applying the rule $h_1^c(\lambda, p_{1,1}) \to \delta(q_0, q_0, p_{1,1})$.
Step 3 : $\gamma_2 \to \gamma_1$ and scanning rules $h_1(p_{1,1},\ p_{1,2})$, $v_1(p_{1,1},\ p_{2,1})$, $h_2(\lambda, p_{2,1})$, $v_2(\lambda, p_{1,2})$ are applied.
Step 4 : $\gamma_1 \to \gamma_2$, $c_2 \to c_3$, $h_1^c(p_{1,1}, p_{1,2}) \to \delta(q_0, h_1^p(p_{1,1}, p_{1,2}))$ and $h_2^c(\lambda, p_{2,1}) \to \delta(v_1^p(p_{1,1}, p_{2,1}), q_0, p_{2,1})$ are applied. In this step, states are assigned to the next diagonal elements $p_{1,2}$ and $p_{2,1}$ depending on $h_1^p(p_{1,1}, p_{1,2})$ and $v_1^p(p_{1,1}, p_{2,1})$. That is, depending on the state stored at the position to the left of (top of) $p_{1,2}$ and $p_{2,1}$.
Clearly, this is the same way the $2DOTA$ works; assigning states to diagonal elements depending on the states of the elements at the top and left.

The rules $\gamma_1 \to \gamma_2$ and $\gamma_2 \to \gamma_1$ characterize the assigning phase and scanning phase respectively. If the rule $\gamma_1 \to \gamma_2$ is applied in a step, then values are assigned to the current elements scanned. No scanning is possible in this step due to the priorities. Similarly, when $\gamma_2 \to \gamma_1$ is applied, no values can be assigned to any of the elements; only the scanning takes place. The scanners are activated in the order $h_1, h_2, \ldots$ and $v_1, v_2, \ldots$. Due to this, at each step the elements under scan are the elements falling along a diagonal and the previous elements scanned by h_i and v_j are respectively the elements to the left and top of the currently scanned elements. This helps in assigning states to the currently scanned elements by referring to the states at the top and left positions.

In the kth assigning phase, we apply $c_k \to c_{k+1}$ and assign states to positions (i, j) for which $i + j - 1 = k$. Note that the priority $c_k \to c_{k+1} >$ all rules with LHS $h_i^c(p_{i,j-1}, p_{i,j})$ for which $i + j \neq k + 1$ ensures that at each step only the elements along the diagonal are assigned values. Proceeding in this manner, we assign to the element at the (m, n)th position the state $\delta(v_n^p(p_{m-1,n}, p_{m,n}), h_m^p(p_{m-1,n}, p_{m,n}))$. If this happens to be a final state, the rule $d_q \to t_1\delta$ is applied and the membrane dissolves. In the next step, the rule $t_1 \to (t, out)$ is applied. Thus, if the given picture is recognized by the $2DOTA$, a t will be sent out and the system halts. Otherwise, the rule $d_q \to \dagger$ is applied and the computations never halt. □

7 A Variant of P Systems with Picture Objects

In this section, we consider a variant of P systems with picture objects. Operations like reflection, rotation, magnification on an $m \times n$ picture produce another picture. To achieve the effect of such operations on a picture using membranes, we define a new class of P systems with picture objects. Because of the form of the basic operation it uses, we call such a system a block-rewriting P system. As we will see, this variant is useful for exemplifying the power of membrane structures. Formally, we give the definition as follows:

Definition 7.1 *A block-rewriting P system with picture objects of degree n is a*

construct

$$\Pi = (V, T, \mu, M_1, \ldots, M_n, R_1, \ldots, R_n)$$

where

- V *is the total alphabet of the system;*
- T *is the output alphabet;*
- μ *is the membrane structure;*
- $M_1, \ldots, M_n$ *are sets of pictures initially present in the regions* $1, \ldots, n$ *of* μ*;*
- $R_i, 1 \leq i \leq n$ *are rules associated with the regions* $1, \ldots, n$ *of* μ*;*

The objects considered here are pictures of the form $\begin{matrix} \# & \# & \ldots \# & \# \\ \# & a_{11} & \ldots a_{1n} & \# \\ \# & a_{21} & \ldots a_{2n} & \# \\ \vdots & \vdots & \ldots \vdots & \vdots \\ \# & a_{m1} & \ldots a_{mn} & \# \\ \# & \# & \ldots \# & \# \end{matrix}$. Rules are applied to sub pictures of any given picture. We consider a sub picture of a picture to be a rectangular block of the form $\boxed{\begin{matrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{matrix}}$ or $\boxed{p_1 \ldots p_n}$, $p_i \in (V \cup \{\#\})^*$, of size $1 \times n$ or $n \times 1$. Any block which contains the boundary marker $\#$ at one or both its ends is called an end block. We apply rules to blocks of size $1 \times n$ or $n \times 1$ replacing them by blocks of size $1 \times m$ or $m \times 1$, $m \geq n$. For blocks which are not end blocks, the replacement is made with a block of the same size, and for end blocks the replacement is done with blocks of the same or higher dimension. This is to take care of the shearing effect which occurs otherwise. For an end block, the increase in length of the resultant block does not affect the positions of any of the symbols in the picture. For instance, consider the block $\boxed{12 \ldots n\#}$ and the rule $12 \ldots n\# \rightarrow a_1 \ldots a_{n+1}\#\#$. On applying this rule, the block $\boxed{12 \ldots n\#}$ in the picture is replaced by $\boxed{a_1 a_2 \ldots a_{n+1}}$ and $\#\#$ is placed next to that, increasing the column size of the picture. Since the increase was made at the end, the positions of all other symbols in the picture are not distorted. Similar is the case when applying such rules to $\boxed{\begin{matrix} \# \\ 1 \\ \vdots \\ n \end{matrix}}$ or $\boxed{\begin{matrix} 1 \\ \vdots \\ n \\ \# \end{matrix}}$ or $\boxed{\#12 \ldots n}$. As in the case of rewriting P systems, we can attach targets to the rules. The rules are applied in a maximally parallel manner, that is, we replace all blocks in the picture for which there are rules available in the given region. If any two rules involve overlapping blocks

say for instance, $\boxed{\#a_{i,1}a_{i,2}\dots a_{i,n}}$ and $\boxed{\begin{matrix} a_{1,1} \\ \vdots \\ a_{i,1} \\ \vdots \\ a_{m,1} \end{matrix}}$ then we choose one of the blocks nondeterministically and apply the rule. Assume that we rewrite in this way n blocks, by rules which have associated targets of the form $here, out, in_j$ of several types and that there are n_{tar} targets of each type. Then, the picture obtained by rewriting is sent to the membrane indicated by the target with the maximal n_{tar} (the target which appears the maximal number of times in the used rules). The result of a computation consists of the set of pictures over T^* sent out of the skin membrane at the end of a halting configuration. We illustrate the effectiveness of this system with a few examples.

Example 7.1 *Consider the P system*

$$\Pi = (V, T, [_1[_2[_3[_4[_5\ ]_5]_4]_3]_2]_1, M_1, \dots, M_5, R_1, \dots, R_5)$$

with

$$\begin{aligned}
V &= \{a, b, c, d, \#, \#', \#'', \#_1, \#_2, \#_a, \#_b, \#_c, \#_b\}, \\
T &= \{a, b, c, d, \#\}, \\
M_1 &= \begin{matrix} \#'' & \# & \# & \#' \\ \# & a & b & \# \\ \# & c & d & \# \\ \#'' & \# & \# & \#' \end{matrix}, M_i = \lambda, i \neq 1, \\
R_1 &= \{\#\#' \to (\#\#\#_1, in_2), b\# \to (bb\#_b, in_2), d\# \to (dd\#_d, in_2)\} \\
&\cup \{\#' \to (\#, out), \#'' \to (\#, out), \#_b \to \#, \#_d \to \#\}, \\
R_2 &= \left\{ \begin{matrix} c \\ \# \end{matrix} \to \left(\begin{matrix} c \\ c \\ \#_c \end{matrix}, in_3 \right), \begin{matrix} d \\ \# \end{matrix} \to \left(\begin{matrix} d \\ d \\ \#_d \end{matrix}, in_3 \right) \right\} \\
&\cup \left\{ \begin{matrix} \# \\ \#'' \end{matrix} \to \left(\begin{matrix} \# \\ \# \\ \#_2 \end{matrix}, in_3 \right), \begin{matrix} \#_d \\ \#_1 \end{matrix} \to \left(\begin{matrix} \#_d \\ \#_d \\ \#_1 \end{matrix}, in_3 \right) \right\} \\
&\cup \{\#_a \to (\#, out), \#_c \to (\#, out)\}, \\
R_3 &= \{\#a \to (\#_a aa, in_4), \#c \to (\#_c cc, in_4), \#''\# \to (\#_2\#\#, in_4)\} \\
&\cup \{\#_2\#_c \to (\#_3\#_c\#_c, in_4), \#_3 \to (\#_3, out), \begin{matrix} \#_c \\ \#_3 \end{matrix} \to \left(\begin{matrix} \#_c \\ \#'' \end{matrix}, out \right)\}, \\
R_4 &= \left\{ \begin{matrix} \# \\ a \end{matrix} \to \left(\begin{matrix} \#_a \\ a \\ a \end{matrix}, in_5 \right), \begin{matrix} \# \\ b \end{matrix} \to \left(\begin{matrix} \#_b \\ b \\ b \end{matrix}, in_5 \right) \right\}
\end{aligned}$$

$$\cup \left\{ \begin{matrix} \#_2 \\ \#_a \end{matrix} \rightarrow \begin{pmatrix} \#_3 \\ \#_a \\ \#_a \end{pmatrix}, in_5 \right\}, \begin{matrix} \#_1 \\ \#_b \end{matrix} \rightarrow \begin{pmatrix} \#_4 \\ \#_b \\ \#_b \end{pmatrix}, in_5 \Bigg) \Bigg\}$$

$$\cup \left\{ \begin{matrix} \#_3 \\ \#_a \end{matrix} \rightarrow \begin{pmatrix} \#'' \\ \#_a \end{pmatrix}, out \right\},$$

$$R_5 = \{\#_d\#_1 \rightarrow (\#_d\#', out), \#_b\#_4 \rightarrow (\#_b\#', out)\}.$$

Clearly, this system scales up the given picture (excluding the boundary marker #) by factors of 2. Hence, the system generates all pictures of dimension $2^n \times 2^n, n \geq 1$ over a, b, c, d.

Example 7.2 *Consider the P system*

$$\Pi = (V, T, [_1[_2\ ]_2]_1, M_1, M_2, R_1, R_2)$$

$$V = \{., \star, \#, \#', \$\},$$

$$T = \{., \star, \#, \$\},$$

$$M_1 = \lambda,\ M_2 = \begin{matrix} \# & \# & \# & \# & \# & \#' \\ \$ & . & \star & . & . & \$ \\ \$ & . & \star & . & . & \$ \\ \$ & . & \star & . & . & \$ \\ \$ & . & \star & \star & \star & \$ \\ \# & \# & \# & \# & \# & \#' \end{matrix},$$

$$R_1 = \{.\$ \rightarrow (..\$, out), .\$ \rightarrow (..\$, in_2), \star\$ \rightarrow (\star\star\$, out), \star\$ \rightarrow (\star\star\$, in_2)\}$$

$$\cup \{\#\#' \rightarrow (\#\#\#, out), \#\#' \rightarrow (\#\#\#', in_2)\},$$

$$R_2 = \left\{ \begin{matrix} . \\ \# \end{matrix} \rightarrow \begin{pmatrix} . \\ . \\ \# \end{pmatrix}, out \right), \begin{matrix} \$ \\ \#' \end{matrix} \rightarrow \begin{pmatrix} \$ \\ \$ \\ \#' \end{pmatrix}, out \Bigg) \Bigg\}$$

$$\cup \left\{ \begin{matrix} \$ \\ \# \end{matrix} \rightarrow \begin{pmatrix} \$ \\ \$ \\ \# \end{pmatrix}, out \right), \begin{matrix} . \\ \star \\ \# \end{matrix} \rightarrow \begin{pmatrix} . \\ . \\ \star \\ \# \end{pmatrix}, out \Bigg) \Bigg\}$$

$$\cup \left\{ \begin{matrix} \star \\ \star \\ \# \end{matrix} \rightarrow \begin{pmatrix} \star \\ \star \\ \star \\ \# \end{pmatrix}, out \right) \Bigg\}.$$

Initially, membrane two contains a picture with an 'L' shape engraved in it. The system sends out pictures after enlarging the 'L' in both directions. Thus this system is capable of producing pictures with L's of any size engraved in it.

Example 7.3 *Consider the P system*

$$\Pi = (V, T, [_3[_1[_2\ ]_2]_1]_3, M_1, \lambda, \lambda, R_1, R_2, R_3)$$

with

$$
\begin{aligned}
V &= \{\$, \$', \$'', \#, a, a', a'', b, b', b'', c, c', c''\},\\
T &= \{\$, \#, a, b, c\},\\
M_1 &= \begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \$ & \# & a & b & c & \# \\ \$ & \# & b & c & a & \# \\ \$ & \# & c & a & b & \# \\ \# & \# & \# & \# & \# & \# \end{array},\\
R_1 &= \{\$\# \to \#\$, \{\$x \to \$'\$^x \mid x \in \{a,b,c\}\} \cup \{\$^x y \to y\$^x \mid x, y \in \{a,b,c\}\}\\
&\cup \{\$^x \# \to (x'\#, in_2) \mid x \in \{a,b,c\}\} \cup \{\$^y x'' \to (y'x'', in_2) \mid x, y \in \{a,b,c\}\}\\
&\cup \{\$'x'' \to x''\$'' \mid x \in \{a,b,c\}\} \cup \{\$''x'' \to x''\$'' \mid x \in \{a,b,c\}\}\\
&\cup \{\$''x \to \$'\$^x \mid x \in \{a,b,c\}\} \cup \{\$''x' \to x'\$'' \mid x \in \{a,b,c\}\}\\
&\cup \{\$''\# \to (\#\$, out)\},\\
R_2 &= \{\{yx' \to \$^y x'' \mid x, y \in \{a,b,c\}\} \cup \{x\$^y \to \$^y x \mid x, y \in \{a,b,c\}\}\\
&\cup \{x''\$^y \to (x''y, out) \mid x, y \in \{a,b,c\}\} \cup \{\$'x' \to (\$'x'', out) \mid x \in \{a,b,c\}\}\\
&\cup \{\$'\$^x \to (\$'x'', out) \mid x \in \{a,b,c\}\}\},\\
R_3 &= \{\{x'' \to (x, out) \mid x \in \{a,b,c\}\}\}.
\end{aligned}
$$

Given a picture or a set of pictures in membrane one, the system outputs those pictures which are obtained from the initial ones by taking a vertical reflection. In our case, because we start from $\begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \$ & \# & a & b & c & \# \\ \$ & \# & b & c & a & \# \\ \$ & \# & c & a & b & \# \\ \# & \# & \# & \# & \# & \# \end{array}$, *the system outputs the picture* $\begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \# & c & b & a & \# & \$ \\ \# & a & c & b & \# & \$ \\ \# & b & a & c & \# & \$ \\ \# & \# & \# & \# & \# & \# \end{array}$.

8 Conclusion

We have introduced a variant of P systems having pictures as the data structure. It has been shown that this variant is able to generate the family $PSML$ with no priorities, scanners and no bound on the number of membranes. The family $PSML$ can also be generated by two other variants of this system: one with replicated rewriting as the underlying control structure with no bound on the number of membranes, another one having three membranes and using rewriting and priorities among the rules. A different type of P systems with picture objects is defined in section 7, which is motivated by the idea of simulating an operation on a picture using membrane structures. A further study of this variant is worthwhile.

References

[1] K.Inoue and A.Nakamura, Some properties of two-dimensional on-line tesselation acceptors, *Information Sciences*, Vol. 13, 1977, 95–121.

[2] S.N.Krishna and R.Rama, On Power of P systems based on Sequential and Parallel Rewriting, *Intern. J. Comp. Math.*, Vol.76, 1–2, 2000, 317–330.

[3] S.N.Krishna, R.Rama P Systems with Replicated Rewriting, *Journal of Automata, Languages and Combinatorics*, to appear.

[4] R.Narasimhan, Labeling schemate and syntactic description of pictures, *Information and control*, 7, 1964, 151–179.

[5] R. Narasimhan,On the description, generation and recognition of classes of pictures, *In Automatic interpretation and classification of images*, Academic Press, 1969.

[6] R.Narasimhan and V.S.N.Reddy, A generative model for handprinted English letters and its computer implementation, *ICC Bulletin*, 6, 1967, 275–287.

[7] N.Nirmal and R.Rama, Picture generation and developmental matrix systems, *Computer Vision, Graphics and Image Processing*, Vol 43, pp 67-80, 1988.

[8] Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143 and *Turku Center for Computer Science-TUCS Report* No 208, 1998 (www.tucs.fi).

[9] Gh.P ăun, Computing with P Systems: Twenty Six Research Topics, *Auckland University, CDMTCS Report* No 119, 2000 (www.cs.auckland.ac.nz/CDMTCS).

[10] Gh.Păun, G. Rozenberg, A. Salomaa, Membrane computing with external output, *Fundamenta Informaticae*, 41, 3 (2000), 259–266, (www.tucs.fi).

[11] A.Rosenfeld, *Picture Languages*, Academic Press, New York, 1979.

[12] G.Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages*, Springer-Verlag, Hiedelberg, 1997.

[13] G.Siromoney, R.Siromoney and K.Krithivasan, Abstract families of matrices and picture languages, *Computer Graphics and Image Processing*, (1972), 1, (284-307)

Received January, 2001