Journal of Universal Computer Science, vol. 18, no. 13 (2012), 1802-1820
submitted: 15/11/11, accepted: 30/3/12, appeared: 1/7/12 © *J.UCS*

Array P System with Shuffle on Trajectories

A.S. Prasanna Venkatesan
(Department of Mathematics, B.S. Abdur Rahman University
Chennai - 600 048, India
prasannaram@bsauniv.ac.in)

D.G. Thomas, T. Robinson
(Department of Mathematics, Madras Christian College
Tambaram, Chennai - 600 059, India
dgthomasmcc@yahoo.com, robin.mcc@gmail.com)

Atulya K Nagar
(Department of Computer Science, Liverpool Hope University
Hope Park, Liverpool, L16 9JD, United Kingdom
nagara@hope.ac.uk)

Abstract: In this paper, we introduce a new concept of trajectory array P system which consists of a membrane structure in which the objects are arrays and the evolutionary rules are given in terms of trajectories. We present some properties of trajectory array P system and compare with certain families of picture languages. We consider a variant of trajectory array P system and show that the languages generated by the trajectory P system and its variant have common intersection.

Key Words: Membrane computing, P system, array languages, shuffle on trajectories
Category: F.4.3

1 Introduction

Membrane computing is a branch of natural computing, the broad area of research concerned with computation taking place in nature with human-designed computing inspired by nature. Membrane computing abstracts computing models from the architecture and the functioning of living cells, as well as from the organization of cells in tissues, organs or other higher order structures such as colonies of cells (e.g. bacteria). The initial goal was to learn from cell biology something possibly useful to computer science, and the area quickly developed in this direction. Several classes of computing models - called P systems - were defined in this context, inspired from biological facts or motivated from mathematical or computer science point of view [Păun 1998, Păun 2002]. The main ingredients of a P system are (i) membrane structure, (ii) multisets of objects and (iii) evolutionary rules. Very briefly, compartments defined by a hierarchical arrangement of membranes have multisets of objects together with evolutionary rules associated with the membranes.

On the other hand, parallel composition of words and languages seems to be a fundamental operation in parallel computation and in the theory of concurrency. This operation is modeled by the shuffle operation [Păun et al. 1995] or restrictions of this operation such as literal shuffle and insertion. A trajectory defines how to skip from a word to another word during the shuffle operation. Trajectory P system has been introduced in [Annadurai et al. 2008] with words as objects. We extend this concept to arrays by introducing trajectory array P system where the objects are arrays.

The paper is organized as follows: Section 2 deals with preliminary concepts related to P system and array languages. In section 3 we define the new concept of trajectory array P system and examine its generative power. In section 4 we compare the generative power of trajectory array P system with that of array rewriting P system. In section 5 we define a variant of trajectory array P system and study its generative power.

2 Preliminaries

In this section we recall some of the basic concepts of P system [Păun 1998, Păun 2002] and trajectories [Annadurai et al. 2008, Giammarresi and Restivo 1997]. P system [Păun 1998] is a new computability model of a distributed parallel type based on the notion of a membrane structure. Such a structure consists of computing cells which are organized hierarchically by the inclusion relation. Each cell is enclosed by its membrane. Each cell is an independent computing agent with its own computing program, which produces objects. The interaction between cells consists of the exchange of objects through membranes. The membranes are labelled in a one-to-one manner. Each membrane identifies a region determined by it and the membranes placed directly inside it (if any). Membranes are represented by matching parentheses. For notions and notations of P systems, we refer to [Păun 1998].

A transition P system of degree n, $n \geq 1$ is a construct
$\Pi = (V, \mu, w_1, w_2, \ldots, w_n, (R_1, \rho_1), (R_2, \rho_2), \ldots, (R_n, \rho_n), i_0)$ where

1. V is an alphabet and its elements are called objects.

2. μ is a membrane structure of degree n, with the membranes and the regions labeled in a one-to-one manner with elements in a given set V. Here we use the labels $1, 2, \ldots, n$.

3. w_i, $1 \leq i \leq n$ are strings from V^* representing multisets over V associated with the regions $1, 2, \ldots, n$ of μ.

4. R_i, $1 \leq i \leq n$ are finite sets of evolution rules over V associated with the regions $1, 2, \ldots, n$ of μ. ρ_i is a partial order relation over R_i, $1 \leq i \leq n$ specifying a priority relation over R_i. An evolution rule is a pair (u, v), which is written in the form $u \to v$ where u is a string over V and $v = v'$ or $v = v'\delta$ where v' is a string over $(V \times \{here, out\}) \cup (V \times \{in_j | 1 \leq j \leq n\})$ and δ is a special symbol not in V. The length of μ is called the radius of the rule $u \to v$.

5. i_0 is a number between 1 and n which specifies the output membrane of μ.

This basic P system has been modified later and many variants of P system have been extensively investigated [Mutyam et al. 2004, Păun et al. 2010, Păun et al. 2000].

Definition 1. Let V be a finite alphabet. V^* is the set of all words over V including the empty word λ. A picture A over V is a rectangular $m \times n$ array of elements of the form

$$A = \begin{matrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{matrix} = [a_{ij}]_{m \times n}$$

The set of all pictures or arrays over V is denoted by V^{**}. A picture or an array language over V is a subset of V^{**}.

Definition 2. Let $A = [a_{ij}]_{m \times p}$, $B = [a_{ij}]_{n \times q}$. The column concatenation $A\Phi B$ of A and B is defined only when $m = n$ and is given by

$$A\Phi B = \begin{matrix} a_{11} & \ldots & a_{1p} & b_{11} & \ldots & b_{1q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mp} & b_{n1} & \ldots & b_{nq} \end{matrix}$$

Similarly, the row concatenation $A\Theta B$ of A and B is defined only when $p = q$ and is given by

$$A\Theta B = \begin{matrix} a_{11} & \ldots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mp} \\ b_{11} & \ldots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{n1} & \ldots & b_{nq} \end{matrix}$$

The empty array is denoted by Λ, $\Lambda\Phi P = P\Phi\Lambda = P$ and $\Lambda\Theta P = P\Theta\Lambda = P$ for any $P \in V^{**}$.

Definition 3. The column shuffle operation on arrays P and Q denoted by $\sqcup\!\sqcup^c$ is defined recursively by

$$\begin{aligned} P \sqcup\!\sqcup^c Q &= ((A\Phi X) \sqcup\!\sqcup^c (B\Phi Y)) \\ &= A\Phi(X \sqcup\!\sqcup^c (B\Phi Y)) \cup B\Phi((A\Phi X) \sqcup\!\sqcup^c Y) \end{aligned}$$

where $P = A\Phi X$ and $Q = B\Phi Y$, $P, Q \in V^{**}$, A is the first column of P and B is the first column of Q. The operation is defined only when the number of rows in P and the number of rows in Q are equal. If A is empty then $X = P$. Likewise if B is empty then $Y = Q$. Also $P \sqcup\!\sqcup^c \Lambda = \Lambda \sqcup\!\sqcup^c P = P$.

The row shuffle operation on arrays P and Q denoted by $\sqcup\!\sqcup^r$ is defined recursively by

$$\begin{aligned} P \sqcup\!\sqcup^r Q &= ((A\Theta X) \sqcup\!\sqcup^r (B\Theta Y)) \\ &= A\Theta(X \sqcup\!\sqcup^r (B\Theta Y)) \cup B\Theta((A\Theta X) \sqcup\!\sqcup^r Y) \end{aligned}$$

where $P = A\Theta X$, $Q = B\Theta Y$ and $P, Q \in V^{**}$, A is the first row of P and B is the first row of Q.

Example 1. If $P = \begin{array}{c} a\ b \\ b\ a \end{array}$ and $Q = \begin{array}{c} c\ d \\ d\ c \end{array}$ then

$$P \sqcup\!\sqcup^c Q = \left\{ \begin{array}{l} a\ b\ c\ d \\ b\ a\ d\ c \end{array}, \begin{array}{l} a\ c\ b\ d \\ b\ d\ a\ c \end{array}, \begin{array}{l} a\ c\ d\ b \\ b\ d\ c\ a \end{array}, \begin{array}{l} c\ a\ b\ d \\ d\ b\ a\ c \end{array}, \begin{array}{l} c\ a\ d\ b \\ d\ b\ c\ a \end{array}, \begin{array}{l} c\ d\ a\ b \\ d\ c\ b\ a \end{array} \right\}$$

and

$$P \sqcup\!\sqcup^r Q = \left\{ \begin{array}{l} a\ b \\ c\ d \\ b\ a \\ d\ c \end{array}, \begin{array}{l} a\ b \\ c\ d \\ d\ c \\ b\ a \end{array}, \begin{array}{l} a\ b \\ b\ a \\ c\ d \\ d\ c \end{array}, \begin{array}{l} c\ d \\ d\ c \\ a\ b \\ b\ a \end{array}, \begin{array}{l} c\ d \\ a\ b \\ d\ c \\ b\ a \end{array}, \begin{array}{l} c\ d \\ a\ b \\ b\ a \\ d\ c \end{array} \right\}$$

Notations If X is an array, then $|X|_c$ denotes the number of columns in X, $|X|_r$ denotes the number of rows in X. If t is a word, then $|t|$ denotes length of t, and $|t|_x$ denotes the number of occurrences of x in t.

Definition 4. Let V be a finite alphabet, t a string over $\{r, u\}$, $v \in \{r, u\}$ and $P, Q \in V^{**}$. The column shuffle of P with Q on the trajectory vt, denoted by $P \sqcup\!\sqcup^c_{vt} Q$, is recursively defined as follows:
If $P = A\Phi X$ and $Q = B\Phi Y$ where $A, B, X, Y \in V^{**}$, A and B are the first columns of P and Q respectively, then

$$\begin{aligned} P \sqcup\!\sqcup^c_{vt} Q &= (A\Phi X) \sqcup\!\sqcup^c_{vt} (B\Phi Y) \\ &= \begin{cases} A\Phi(X \sqcup\!\sqcup^c_t (B\Phi Y)) \text{ if } v = r \\ B\Phi((A\Phi X) \sqcup\!\sqcup^c_t Y) \text{ if } v = u \end{cases} \end{aligned}$$

$$\text{If } P = \Lambda, \text{then } \ \Lambda \sqcup\!\sqcup^c_{vt}(B\Phi Y) = \begin{cases} \phi & \text{if } v = r \\ B\Phi(\Lambda \sqcup\!\sqcup^c_t Y) & \text{if } v = u \end{cases}$$

$$\text{If } Q = \Lambda, \text{then } \ (A\Phi X) \sqcup\!\sqcup^c_{vt}\Lambda = \begin{cases} A\Phi(X \sqcup\!\sqcup^c_t \Lambda) & \text{if } v = r \\ \emptyset & \text{if } v = u \end{cases}$$

$$\text{and } \Lambda \sqcup\!\sqcup^c_t \Lambda = \begin{cases} \Lambda \text{ if } t = \lambda \\ \emptyset \ \text{ otherwise} \end{cases}$$

where the symbol $\emptyset$ denotes the empty set and λ denotes the empty string.

The row shuffle of P with Q on the trajectory vt, $v \in \{l, d\}$, $t \in \{l, d\}^*$ is defined in a similar way except that the column catenation Φ is replaced by the row catenation Θ and r is replaced by l and u is replaced by d. Also if $|P|_c \neq |t|_r$ or $|Q|_c \neq |t|_u$, then $P \sqcup\!\sqcup^c_t Q = \emptyset$. Similarly, if $|P|_r \neq |t|_l$ or $|Q|_r \neq |t|_d$, then $P \sqcup\!\sqcup^r_t Q = \emptyset$.

Example 2. Let $V = \{a, b\}$,

$$P = \begin{matrix} a & a & a & a \\ a & a & a & a \\ a & a & a & a \end{matrix}, \ Q = \begin{matrix} b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \end{matrix} \ \text{and } R = \begin{matrix} c & c & c & c \\ c & c & c & c \\ c & c & c & c \end{matrix}.$$

$$\text{Then } P \sqcup\!\sqcup^c_t Q = \begin{matrix} a & b & b & a & b & a & b & b & a \\ a & b & b & a & b & a & b & b & a \\ a & b & b & a & b & a & b & b & a \end{matrix} \ \text{if } t = ur^2urur^2u$$

$$\text{and } P \sqcup\!\sqcup^r_t R = \begin{matrix} a & a & a & a \\ c & c & c & c \\ c & c & c & c \\ a & a & a & a \\ c & c & c & c \\ c & c & c & c \\ a & a & a & a \end{matrix} \ \text{if } t = ld^2ld^2l.$$

Definition 5. A Siromoney matrix grammar [Siromoney et al. 1972] is a 2-tuple (G_1, G_2) where $G_1 = (H_1, I_1, P_1, S)$ is a regular, a context-free or a context-sensitive grammar

- H_1 is a finite set of horizontal non-terminals
- $I_1 = \{S_1, S_2, \ldots, S_k\}$, a finite set of intermediates, $H_1 \cap I_1 = \emptyset$
- P_1 is a finite set of production rules called horizontal production rules

- $S \in H_1$, is the start symbol.

$G_2 = (G_{21}, G_{22}, \ldots, G_{2k})$ where
$G_{2i} = (V_{2i}, T, P_{2i}, S_i)$, $1 \leq i \leq k$ are regular grammars,

- V_{2i} is a finite set of vertical non-terminals, $V_{2i} \cap V_{2j} = \emptyset$, $i \neq j$
- T is a finite set of terminals
- P_{2i} is a finite set of right linear production rules of the form $X \rightarrow aY$ or $X \rightarrow a$ where $X, Y \in V_{2i}$, $a \in T$
- $S_i \in V_{2i}$ is the start symbol of G_{2i}.

The type of G_1 gives the type of G; so we speak about regular, context-free or context-sensitive Siromoney matrix grammars if G_1 is regular, context-free or context-sensitive respectively.

Derivations are defined as follows: First a string $S_{i1}S_{i2}\ldots S_{in} \in I_1^*$ is generated horizontally using the horizontal production rules of P_1 in G_1. i.e., $S \Rightarrow S_{i1}S_{i2}\ldots S_{in} \in I_1^*$.

Vertical derivations proceed as follows: we write

$$\begin{array}{c} A_{i1}\ldots A_{in} \\ \Downarrow \\ a_{i1}\ldots a_{in} \\ B_{i1}\ldots B_{in} \end{array}$$

if $A_{ij} \rightarrow a_{ij}B_j$ are rules in P_{2j}, $1 \leq j \leq n$. The derivation terminates if $A_j \rightarrow a_{mj}$ are all terminal rules in G_2.

The set $L(G)$ of picture arrays generated by G consists of all $m \times n$ arrays $[a_{ij}]$ such that $1 \leq i \leq m$, $1 \leq j \leq n$ and $S \Rightarrow_{G_1}^* S_{i1}S_{i2}\ldots S_{im} \Rightarrow_{G_2}^* [a_{ij}]$.

We denote the picture language classes of regular, context-free, context-sensitive Siromoney matrix grammars by $RML, CFML, CSML$ respectively.

Definition 6. Given a picture p of $size(m, n)$, let $h \leq m$, $k \leq n$. We denote by $B_{h,k}(\hat{p})$, the set of all blocks (or sub-pictures of p of $size(h, k)$). We call a square picture of $size(2, 2)$ as a tile.

Definition 7. Let Γ be a finite alphabet. A two-dimensional language $L \subseteq \Gamma^{**}$ is local if there exists a finite set θ of tiles over the alphabet $\Gamma \cup \{\#\}$ such that $L = \{p \in \Gamma^{**} | B_{2,2}(\hat{p}) \subseteq \theta\}$. The family of local picture languages will be denoted by LOC [Giammarresi and Restivo 1997]. We now give an example of a local two-dimensional picture language.

Example 3. Let $\Sigma = \{a\}$ be a one-letter alphabet and let L be the subset of Σ^{**} that contains all the rectangular pictures over Σ.

Consider the set

$$\theta = \left\{ \begin{array}{|c|c|}\hline \# & a \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & \# \\ \hline a & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline a & a \\ \hline \end{array} \right\}$$

The language $L = L(\theta)$ is the language of all the rectangular pictures over Σ. Hence L is a local language.

Definition 8. A tiling system (TS) is a 4-tuple $T = (\Sigma, \Gamma, \theta, \pi)$ where Σ and Γ are two finite alphabets, θ is a finite set of tiles over the alphabet $\Gamma \cup \{\#\}$ and $\pi : \Gamma \to \Sigma$ is a projection.

We say that a language $L \subseteq \Sigma^{**}$ is recognizable by a tiling system (or tiling recognizable) if there exists a tiling system $T = (\Sigma, \Gamma, \theta, \pi)$ such that $L = L(T)$. This family of tiling recognizable picture languages is denoted by REC [Giammarresi and Restivo 1997].

Example 4. Let $\Sigma = \{a\}$ be a one letter alphabet. The set of pictures of a's with three columns is a recognizable language which is not local.

i.e., $L = \left\{ a\ a\ a, \begin{array}{c} a\ a\ a \\ a\ a\ a \end{array}, \begin{array}{c} a\ a\ a \\ a\ a\ a \\ a\ a\ a \end{array}, \ldots \right\} \in REC.$

3 Trajectory Array P system

In this section we introduce a new concept of trajectory array P system.

Definition 9. A trajectory array P system is defined as
$\Pi = (V, T, \mu, (L_1, T_1, tar), (L_2, T_2, tar), \ldots, (L_n, T_n, tar))$ where

1. V is an alphabet; its elements are called objects
2. $T = T_c \cup T_r$ is the control alphabet where $T_c = \{r, u\}$, $T_r = \{l, d\}$.
3. μ is a membrane structure consisting of n membranes $\mu_1, \mu_2, \ldots, \mu_n$.
4. $L_i \subset V^{**}$, $T_i = T_{ci} \cup T_{ri}$ where $T_{ci} \subset T_c^*$, $T_{ri} \subset T_r^*$ and $tar \in \{\{here, out\}, \{here, in_j\}, here, out, in_j | 1 \leq j \leq n\}$.

For each i, $1 \leq i \leq n$, let (L_i, T_i) be the content of i^{th} membrane. If i is an elementary membrane then $L_i' = L_i^{T_i}$ is the language computed in the i^{th} membrane by applying the trajectory rule T_i on L_i and depending on the target it will be sent out of the membrane and no object will remain in that membrane if $tar = out$ or it will remain in that membrane if $tar = here$.

If i is not an elementary membrane, but contains a membrane j, and if $tar = \{in_j, here\}$ then one copy of the generated language will remain in the same membrane i and another copy will be sent to j^{th} membrane. If $tar = \{in_j, out\}$ then one copy of the generated language will be sent out of the membrane and another copy will be sent to j^{th} membrane. If $tar = \{here, out\}$ then one copy of the generated language will be sent out of the membrane and another copy will remain in the same membrane. This process is repeated till the language is sent to the skin membrane and the language obtained by the computations in the skin membrane is denoted by $L(\Pi)$. The family of languages generated by the trajectory P system is denoted by $TAPSL$.

We now give a trajectory array P system which generates the set of all digitized pictures of token I.

Example 5. Consider the trajectory array P system
$\Pi = (V, T, \mu, (L_1, T_1, tar), (L_2, T_2, tar), \ldots, (L_6, T_6, tar))$ where
$V = \{\#\} \cup \{X\}$, # represents blank symbol,
$T = T_c \cup T_r$ with $T_c = \{r, u\}$, $T_r = \{l, d\}$
$\mu = [_6[_5[_4]_4[_3[_2[_1]_1]_2]_3]_5]_6$ is the membrane structure consisting of six membranes with skin membrane labeled 6.

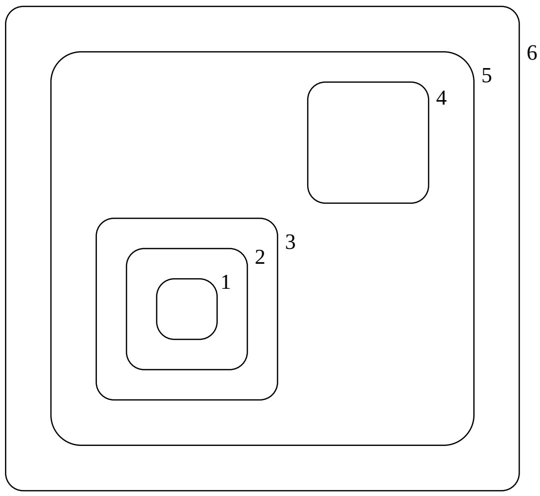

Figure 1: Membrane structure of Π in Example 5

$L_1 = \{\#\}$, $T_1 = T_{c1} \cup T_{r1}$, $T_{c1} = \{r^n u^m | n, m \geq 1\}$,
$T_{r1} = \{l^n d^n | n \geq 1\}$, $tar = \{here, out\}$.
$L_2 = \emptyset$, $T_2 = T_{c2} \cup T_{r2}$, $T_{c2} = \emptyset$,

$T_{r2} = \{l^n d^m | n, m \geq 1\}$, $tar = \{in_1, out\}$.
$L_3 = \{X\}$, $T_3 = T_{c3} \cup T_{r3}$, $T_{c3} = \{r^n u r^n | n \geq 1\}$,
$T_{r3} = \{l^n d^m | n, m \geq 1\}$, $tar = \{here, out\}$.
$L_4 = \left\{ \begin{matrix} X \\ X \end{matrix} \right\}$, $T_4 = T_{c4} \cup T_{r4}$,
$T_{c4} = \{r^n u^m | n, m \geq 1\}$, $T_{r4} = \emptyset$, $tar = \{here, out\}$.
$L_5 = \emptyset$, $T_5 = T_{c5} \cup T_{r5}$, $T_{c5} = \emptyset$,
$T_{r5} = \{l d^m l | m \geq 1\}$, $tar = \{out\}$.
$L_6 = \emptyset$, $T_6 = \emptyset$.

$$\begin{matrix} X & X & X & X & X & X & X \\ \# & \# & \# & X & \# & \# & \# \\ \# & \# & \# & X & \# & \# & \# \\ \# & \# & \# & X & \# & \# & \# \\ X & X & X & X & X & X & X \end{matrix}$$

Figure 2: I in Example 5

In membrane 1, by applying rule T_{c1} (when $n = 1, m = 1$), we get $L'_{T_{c1}} = \{\# \ \#\}$ and by applying rule T_{r1} ($n = 1$), we get $L'_{T_{r1}} = \left\{ \begin{matrix} \# \\ \# \end{matrix} \right\}$. A copy of this will be sent to membrane 2 where by applying rule T_{r2} ($n = 2, m = 2$) we get $L'_{T_{r2}} = \left\{ \begin{matrix} \# \ \# \\ \# \ \# \end{matrix}, \begin{matrix} \# \\ \# \\ \# \\ \# \end{matrix} \right\}$. Again a copy of the output will be sent to membrane 1. By applying the rules in membrane 1 and membrane 2 iteratively we can generate a $m \times n$ picture $\begin{pmatrix} \# & \cdots & \# \\ \vdots & \ddots & \vdots \\ \# & \cdots & \# \end{pmatrix}_{m \times n}$ which will be sent out of membrane 2 to membrane 3. At the same time, in membrane 3 by applying T_{r3} iteratively we can get $m \times 1$ picture of the form $\begin{pmatrix} X \\ X \\ \vdots \\ X \end{pmatrix}_{m \times 1}$. In the same membrane, by using one of the rules in T_{c3}, we can get pictures of the form $\begin{matrix} \# \ \# \ X \ \# \ \# \\ \# \ \# \ X \ \# \ \# \\ \# \ \# \ X \ \# \ \# \end{matrix}$. These pictures will be sent out of membrane 3 to membrane 5. Similarly, in membrane

4 by applying T_{c4} iteratively we can get a picture of the form $\begin{pmatrix} X & X & \cdots & X \\ X & X & \cdots & X \end{pmatrix}_{2\times m}$ which will be sent to membrane 5. In membrane 5 by applying one of the rules in T_{r5} a row containing the letter X is attached to top and bottom of the picture obtained in membrane 3 and will be sent out to membrane 6. Hence the above system generates the set of all digitalized pictures of the token I of different proportions which is a CFML [Siromoney et al. 1972, Siromoney et al. 1973].

Theorem 10. *The class TAPSL intersects the class RML.*

Proof. Consider the trajectory array P system
$\Pi = (V, T, \mu, (L_1, T_1, tar), (L_2, T_2, tar), \ldots, (L_6, T_6, tar))$ where
$V, T, \mu, (L_1, T_1, tar), (L_2, T_2, tar)$ are same as in Example 5.
$L_3 = \{X\}$, $T_3 = T_{c3} \cup T_{r3}$, $T_{c3} = \{r^n u^m | n, m \geq 1\}$,
$T_{r3} = \{l^n d | n \geq 2\}$, $tar = \{here, out\}$.
$L_4 = \{X\}$, $T_4 = T_{c4} \cup T_{r4}$,
$T_{c4} = \emptyset$, $T_{r4} = \{l^n d^m | n, m \geq 1\}$, $tar = \{here, out\}$.
$L_5 = \emptyset$, $T_5 = T_{c5} \cup T_{r5}$, $T_{c5} = \{ru^n | n \geq 2\}$,
$T_{r5} = \emptyset$, $tar = \{out\}$.
$L_6 = \emptyset$, $T_6 = \emptyset$.

By applying rules in membrane 1 and 2 (as in Example 5) iteratively to get the pictures of the form $\begin{pmatrix} \# & \cdots & \# \\ \vdots & \ddots & \vdots \\ \# & \cdots & \# \end{pmatrix}_{m\times n}$. These pictures will be sent out to membrane 3 using $tar = out$. At the same time, in membrane 3, using rules of T_{c3} iteratively we get a row matrix of the form $(\, X\ X\ \ldots\ X\,)_{1\times m}$ containing the letter X only. At any time, we can apply the rules of T_{r3} which will attach this row matrix containing the letter X at the bottom of the matrix obtained from membrane 2 and thus we get pictures of the form $\begin{matrix} \# & \# & \# & \# & \# \\ \# & \# & \# & \# & \# \\ \# & \# & \# & \# & \# \\ X & X & X & X & X \end{matrix}$. These pictures will be sent to membrane 5 using $tar = out$.

Similarly in membrane 4, the rules of T_{r4} are applied iteratively to the initial axiom present in membrane 4 to get a column matrix containing the letter X only and it will be sent to membrane 5. In membrane 5, using a suitable rule of T_{c5}, a column matrix containing the letter X only is attached as a first column to the matrix of the form $\begin{matrix} \# & \# & \# & \# & \# \\ \# & \# & \# & \# & \# \\ \# & \# & \# & \# & \# \\ X & X & X & X & X \end{matrix}$ and will be sent out of membrane 5. Thus, in membrane 6 we get arrays describing the token L (Figure 4) which is a RML [Siromoney et al. 1972, Siromoney et al. 1973].

$$\begin{array}{ccccc} X & \# & \# & \# & \# \\ X & \# & \# & \# & \# \\ X & \# & \# & \# & \# \\ X & \# & \# & \# & \# \\ X & \# & \# & \# & \# \\ X & X & X & X & X \end{array}$$

Figure 3: L in Theorem 10

Theorem 11. *The class TAPSL intersects the class CFML - RML.*

Proof. Consider the trajectory array P system
$\Pi = (V, T, \mu, (L_1, T_1, tar), (L_2, T_2, tar), \ldots, (L_6, T_6, tar))$ where
$V, T, \mu, (L_1, T_1, tar), \ldots, (L_3, T_3, tar)$ are same as in Example 5.
$L_4 = \{X\}$, $T_4 = T_{c4} \cup T_{r4}$, $T_{c4} = \{r^n u^m | n, m \geq 1\}$,
$T_{r4} = \emptyset, tar = \{here, out\}$.
$L_5 = \emptyset$, $T_5 = T_{c5} \cup T_{r5}$, $T_{c5} = \emptyset$,
$T_{r5} = \{ld^m | m \geq 2\}, tar = \{out\}$.
$L_6 = \emptyset$, $T_6 = \emptyset$.

Working of this system is similar to the system given in Example 5 except that in membrane 5 when the rule T_{r5} is applied a row containing the letter X only is attached to the top of the matrix of the form $\begin{array}{ccc} \# & X & \# \\ \# & X & \# \\ \# & X & \# \end{array}$ which is obtained in membrane 3 by applying rule T_{c3}. Hence the above system generates $m \times n$ arrays describing the token T (Figure 3) which is a CFML.

$$\begin{array}{ccccc} X & X & X & X & X \\ \# & \# & X & \# & \# \\ \# & \# & X & \# & \# \\ \# & \# & X & \# & \# \\ \# & \# & X & \# & \# \end{array}$$

Figure 4: T in Theorem 11

Theorem 12. *The class TAPSL intersects the class CSML-CFML.*

Proof. Consider the trajectory array P system
$\Pi = (V, T, \mu, (L_1, T_1, tar), (L_2, T_2, tar), \ldots, (L_7, T_7, tar))$ where

$V, T, (L_1, T_1, tar), (L_2, T_2, tar)$ are same as in Example 5.
$\mu = [_7[_6[_4[_2[_1]_1]_2[_3]_3]_4[_5]_5]_6]_7$
$L_3 = \{\, X\ X\ X\ X \,\}$, $T_3 = T_{c3} \cup T_{r3}$, $T_{c3} = \emptyset$,
$T_{r3} = \{l^n d^m | n, m \geq 1\}, tar = \{here, out\}$.
$L_4 = \emptyset$, $T_4 = T_{c4} \cup T_{r4}$, $T_{c4} = \{ru^n ru^n ru^n r | n \geq 2\}$,
$T_{r4} = \emptyset, tar = \{here, out\}$.
$L_5 = \{X\}$, $T_5 = T_{c5} \cup T_{r5}$, $T_{c5} = \{r^n u^m | n, m \geq 1\}$,
$T_{r5} = \emptyset, tar = \{here, out\}$.
$L_6 = \emptyset$, $T_6 = T_{c6} \cup T_{r6}$, $T_{c6} = \emptyset$,
$T_{r6} = \{ld^m | m \geq 3\}, tar = \{out\}$.
$L_7 = \emptyset$, $T_7 = \emptyset$.

$$
\begin{matrix}
X & X & X & X & X & X & X & X & X & X \\
X & \# & \# & X & \# & \# & X & \# & \# & X \\
X & \# & \# & X & \# & \# & X & \# & \# & X \\
X & \# & \# & X & \# & \# & X & \# & \# & X \\
X & \# & \# & X & \# & \# & X & \# & \# & X \\
X & \# & \# & X & \# & \# & X & \# & \# & X \\
X & \# & \# & X & \# & \# & X & \# & \# & X
\end{matrix}
$$

Figure 5: "fork" in Theorem 12

As in Example 5, we use the rules in membrane 1 and membrane 2 iteratively to get pictures of the form $\begin{pmatrix} \# & \cdots & \# \\ \vdots & \ddots & \vdots \\ \# & \cdots & \# \end{pmatrix}_{m \times n}$. These pictures will be sent to membrane 4 by using $tar = out$. At the same time, in membrane 3, using the rules of T_{r3} iteratively on the initial axiom array $(X\ X\ X\ X)_{1\times 4}$, we get pictures of the form $\begin{pmatrix} X & X & X & X \\ \vdots & \vdots & \vdots & \vdots \\ X & X & X & X \end{pmatrix}_{n \times 4}$ which will be sent to membrane 4. Now in membrane 4, using one of the rules of T_{c4} we get pictures of the form $\begin{matrix} X & \# & \# & X & \# & \# & X & \# & \# & X \\ X & \# & \# & X & \# & \# & X & \# & \# & X \\ X & \# & \# & X & \# & \# & X & \# & \# & X \\ X & \# & \# & X & \# & \# & X & \# & \# & X \\ X & \# & \# & X & \# & \# & X & \# & \# & X \\ X & \# & \# & X & \# & \# & X & \# & \# & X \end{matrix}$ which will be sent to membrane 6. In membrane 5, we get a row matrix containing the letter X only is obtained using the rules of

T_{c5} and these pictures will be sent out to membrane 6. In membrane 6, using rules of T_{r5}, a row containing the letter X is attached at the top of the pictures which we got from membrane 4. These pictures will be sent to membrane 7 using $tar = out$. Thus in membrane 7, we get four-pronged fork (without handle) of all sizes and proportion (but retaining equal intervals between forks) which is a CSML [Siromoney et al. 1972].

Theorem 13. *The class TAPSL intersects the class LOC.*

Proof. Consider the trajectory array P system

$$\Pi = (V, T, \mu, (L_1, T_1, tar), (L_2, T_2, tar))$$

where $V = \{a\}$, $T = T_c \cup T_r$ where $T_c = \{r, u\}$ and $T_r = \{l, d\}$, $\mu = [_2[_1]_1]_2$ is a membrane structure consisting of two membranes with skin membrane labeled 2.
$L_1 = \{a\}$, $T_1 = T_{c1} \cup T_{r1}$, $T_{c1} = \{r^n u^m | n, m \geq 1\}$,
$T_{r1} = \{l^n d^m | n, m \geq 1\}$, $tar = \{here, out\}$.
$L_2 = \emptyset$, $T_2 = T_{c2} \cup T_{r2}$, $T_{c2} = \emptyset$, $T_{r2} = \emptyset$.
Then $L(\Pi)$ is the set of all rectangular pictures over $\{a\}$ as in Example 3. $L(\Pi) \in LOC$.

Theorem 14. *The class TAPSL intersects the class REC.*

Proof. Consider the trajectory array P system

$$\Pi = (V, T, \mu, (L_1, T_1, tar))$$

where $V = \{a\}$, $T = T_c \cup T_r$ where $T_c = \{r, u\}$ and $T_r = \{l, d\}$, $\mu = [_1]_1$ is a membrane structure consisting only skin membrane labeled 1.
$L_1 = \{a\ a\ a\}$, $T_1 = T_{c1} \cup T_{r1}$, $T_{c1} = \emptyset$,
$T_{r1} = \{l^n d^m | n, m \geq 1\}$, $tar = here$.
Then $L(\Pi)$ is the set of all pictures of a's with three columns as in Example 4. $L(\Pi) \in REC$.

Remark. The results proved in this paper are represented in Figure 6.

4 Comparison with Array-Rewriting *P* Systems

Ceterchi et al. [Ceterchi et al. 2003] initially proposed array-rewriting P systems by extending the notion of string-rewriting P systems [Păun et al. 2010]. The family of all array languages generated by the array-rewriting P system is denoted by $AP_m[\alpha]$ where $\alpha \in \{REG, CF, \#CF\}$. In this section, we compare trajectory array P systems with array-rewriting P systems.

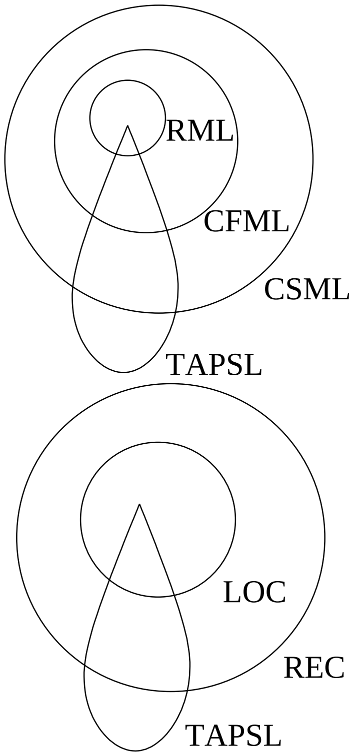

Figure 6: Representation of the results

Theorem 15. *The class $TAPSL$ intersects the class $AP_m[\alpha]$ where $\alpha \in \{REG, CF, \#CF\}$.*

Proof. Consider the trajectory array P system
$\Pi = (V, T, \mu, (L_1, T_1, tar), (L_2, T_2, tar), \ldots, (L_6, T_6, tar))$ where
$V = \{a, b\}$, $T = T_c \cup T_r$ where $T_c = \{r, u\}$,
$T_r = \{l, d\}$, $\mu = [_6[_5[_4]_4[_3[_2[_1]_1]_2]_3]_5]_6$ is the membrane structure consisting of six membranes with skin membrane labeled 6.

$L_1 = \{b\}$, $T_1 = T_{c1} \cup T_{r1}$, $T_{c1} = \{r^n u^m | n, m \geq 1\}$,
$T_{r1} = \{l^n d^n | n \geq 1\}$, $tar = \{here, out\}$.
$L_2 = \emptyset$, $T_2 = T_{c2} \cup T_{r2}$, $T_{c2} = \emptyset$,
$T_{r2} = \{l^n d^m | n, m \geq 1\}$, $tar = \{in_1, out\}$.
$L_3 = \{a \;\; a\}$, $T_3 = T_{c3} \cup T_{r3}$, $T_{c3} = \{r u^m r | m \geq 2\}$,
$T_{r3} = \{l^n d^m | n, m \geq 1\}$, $tar = \{here, out\}$.

$L_4 = \left\{ \begin{matrix} a \\ a \end{matrix} \right\}$, $T_4 = T_{c4} \cup T_{r4}$, $T_{c4} = \{r^n u^m | n, m \geq 1\}$,
$T_{r4} = \emptyset$, $tar = \{here, out\}$.
$L_5 = \emptyset$, $T_5 = T_{c5} \cup T_{r5}$, $T_{c5} = \emptyset$,
$T_{r5} = \{ld^m l | m \geq 2\}$, $tar = \{out\}$.
$L_6 = \emptyset$, $T_6 = \emptyset$.

By applying the rules in membrane 1 and membrane 2 (as in Example 5) iteratively, we can generate a $m \times n$ picture $\begin{pmatrix} b & \dots & b \\ \vdots & \ddots & \vdots \\ b & \dots & b \end{pmatrix}_{m \times n}$ which will be sent to membrane 3. At the same time, in membrane 3 by applying T_{r3} iteratively we can get $n \times 2$ pictures of the form $\begin{pmatrix} a & a \\ a & a \\ \vdots & \vdots \\ a & a \end{pmatrix}$. Now using the rules of T_{c3} a column matrix containing the letter a is attached to the first and last column of the array $\begin{pmatrix} b & \dots & b \\ \vdots & \ddots & \vdots \\ b & \dots & b \end{pmatrix}_{m \times n}$ and we get pictures of the form $\begin{matrix} a\,b\,b\,b\,b\,a \\ a\,b\,b\,b\,b\,a \\ a\,b\,b\,b\,b\,a \\ a\,b\,b\,b\,b\,a \\ a\,b\,b\,b\,b\,a \end{matrix}$. These pictures will be sent out to membrane 5. Similarly, in membrane 4 by applying T_{c4} iteratively we can get a pictures of the form $\begin{pmatrix} a & a & \dots & a \\ a & a & \dots & a \end{pmatrix}_{2 \times n}$ which will be sent to membrane 5. Now in membrane 5 by applying one of the rules of T_{r5}, a row containing the letter a is attached to the top and bottom of the matrix $\begin{matrix} a\,b\,b\,b\,b\,a \\ a\,b\,b\,b\,b\,a \\ a\,b\,b\,b\,b\,a \\ a\,b\,b\,b\,b\,a \\ a\,b\,b\,b\,b\,a \end{matrix}$ so that we get a solid rectangle as shown in Figure 7. These pictures will be sent out to membrane 6 which is the skin membrane.

$$\begin{matrix} a & a & a & a & a & a \\ a & b & b & b & b & a \\ a & b & b & b & b & a \\ a & b & b & b & b & a \\ a & a & a & a & a & a \end{matrix}$$

Figure 7: Solid rectangle

The above system generates all solid rectangles marked as suggested in Figure 7: the edges are marked with a and all the internal pixels are marked with b.

We note that the pictures of all solid rectangles can also be generated by an array-rewriting P system [Ceterchi et al. 2003].

Remark. The trajectory array P system cannot generate the family of all L-shaped angles with equal arms, each arm being of length at least three which is generated by an array-rewriting P system [Ceterchi et al. 2003], since the equality in number of rows and columns cannot be maintained by the shuffle operation.

Hence there is a language which is in $AP_m[\alpha]$ but not in $TAPSL$. Also the language given in Theorem 12 cannot be generated by an array-rewriting P system with regular rules.

5 A Variant of Trajectory Array P System

In this section, we introduce a variant of trajectory array P system by extending the definition of P system with catalytic-like rules given in [Niu et al. 2010]. Definition 9 uses trajectories as evolution rules. In this section, we modify that definition and compare the languages generated by this new system with TAPSL.

Definition 16. An array P system with shuffle operation is defined as

$$\Pi = (V, T, \mu, L_1, L_2, \ldots, L_m, T_1, T_2, \ldots, T_m, R_1, R_2, \ldots, R_m, i_0)$$

where

1. V is an alphabet; its elements are called objects
2. $T = T_c \cup T_r$ where $T_c = \{r, u\}$, $T_r = \{l, d\}$, is the control alphabet.
3. μ is a membrane structure consisting of m membranes $\mu_1, \mu_2, \ldots, \mu_n$.
4. $L_i \subset V^{**}$, $1 \leq i \leq m$ are initial sets of array languages respectively associated with regions $1, 2, \ldots, m$ of μ.
5. $T_i = T_{ci} \cup T_{ri}$ where $T_{ci} \subset T_c^*$, $T_{ri} \subset T_r^*$ are sets of trajectories respectively associated with region $1, 2, \ldots, m$ of μ.
6. R_i is a finite set of evolution rules associated with region i of μ. Each evolution rule is in any one of the following forms:
   (i) $A_{h_1} B_{h_2} \rightarrow (C, tar)$, or
   (ii) $A_{h_1} B_{h_2} \rightarrow (A, here)(C, tar)$ or
   (iii) $A_{h_1} B_{h_2} \rightarrow (C, tar)(B, here)$

where A, B, C are array languages over V, $h_1, h_2 \in T_{ci}$ or $h_1, h_2 \in T_{ri}$, $h_1 \neq h_2$ and $tar \in \{here, out, in_j | 1 \leq j \leq m\}$. If there exist $A', B' \in L_i$ with $A' \in A$, $B' \in B$ and $A' ⧢_{T_i} B' \neq \emptyset$, then A', B' can be assigned to one rule of the three forms (i to iii) non-deterministically and $C = A' ⧢_{T_i} B'$ is produced. The rule $A_{h_1} B_{h_2} \to (C, tar)$ means as follows: Case (i): Let $h_1, h_2 \in T_{ci}$. Then there exist $A', B' \in L_i$, with $A' \in A$ and $B' \in B$ such that $A' ⧢_{T_{ci}} B' = C$. Similarly we have other cases to obtain C.

7. i_0 is the output membrane.

The symbols $here, out, in_j$, $1 \leq j \leq m$ are called target indications. The objects can be transported through membranes due to targets. If $(C, here)$ is present, the array object C is placed in the same region i where the rule is applied. If (C, out) is present, the object C is moved to the region immediately outside membrane i. In the same way other target indicators work as in Definition 9.

Rules from sets of R_i, $1 \leq i \leq m$ are applied to sets of array languages in corresponding region of μ synchronously, in non-deterministic maximally parallel manner. The m-tuple of sets of languages present at any moment in the m regions of Π constitutes the configuration of the system at that moment. The m-tuple $(L_1, L_2, \ldots, L_m)$ is the initial configuration of Π. A transition between configuration is governed by application of evolution rules all of which are based on shuffle operation. A sequence of transitions between configurations of a given system Π is called a computation with respect to Π. A computation is successful if and only if it halts. i.e., there is no rule applicable to the objects present in the last configuration. The language generated by the system is the union of objects present in the output membrane i_0. The family of languages generated by this system with at most m membranes is denoted by $STAPL_m(type)$ where $type \in \{cat, \lambda\}$.

Example 6. Consider the array P system with shuffle operation without catalytic-like rules

$$\Pi = (V, T, \mu, L_1, L_2, L_3, T_1, T_2, T_3, R_1, R_2, R_3, 3)$$

where
$V = \{X, \#\}$,
$T = \{r, u\} \cup \{l, d\}$,
$\mu = [_1[_2[_3]_3]_2]_1$,
$L_1 = \{X(\#)^n | n \geq 1\}$,
$L_2 = \{(X)^n | n \geq 2\}$,
$T_1 = T_{r_1} = \{l^n d | n \geq 1\}$,
$R_1 = \{X(\#)^n | n \geq 1\}_l \{X(\#)^n | n \geq 1\}_d \to \{C', \{here, in\}\}$,
$T_2 = T_{r_2} = \{l^n d | n \geq 1\}$,
$R_2 = \{X(\#)^n | n \geq 1\}_l \{(X)^n | n \geq 2\}_d \to \{C'', in_3\}$,

$L_3 = \emptyset$,
$R_3 = \emptyset$,
$T_3 = \emptyset$.

Initially, arrays of size $1 \times n$ will be present in membrane 1. According to T_1, rule R_1 can be applied to get a $2 \times n$ array C' of the form $\begin{matrix} X\ (\#)^n \\ X\ (\#)^n \end{matrix}$. A copy of this remains in membrane 1 and a copy is sent to membrane 2. If it is sent to membrane 2, according to T_{r_2}, rule R_2 can be applied to get a L shaped array C'' of the form $\begin{matrix} X\ (\#)^n \\ X\ (\#)^n \\ X\ (X)^n \end{matrix}$ which will be sent to membrane 3.

By applying the rules R_1 iteratively we get different size of L shaped arrays.

$$\begin{matrix} X & \# & \# & \# & \# \\ X & \# & \# & \# & \# \\ X & \# & \# & \# & \# \\ X & \# & \# & \# & \# \\ X & \# & \# & \# & \# \\ X & X & X & X & X \end{matrix}$$

Figure 8: L in Example 6

Theorem 17. *The class TAPSL intersects the class $STALP_3$.*

Proof follows from Theorem 10 and Example 6.

6 Conclusion

In this paper, we have introduced trajectory array P system and given examples which generate certain members of CSML, CFML, RML [Siromoney et al. 1972, Siromoney et al. 1973], LOC and REC [Giammarresi and Restivo 1997]. Trajectory array P systems cannot generate square pictures over singleton $\{a\}$, since the equality in number of rows and columns cannot be maintained by the shuffle operation. We have compared this P system with Siromoney matrix grammars, tiling systems and array-rewriting P systems. Further studies like generative power, descriptional complexity, comparison results and universality results can be done. A preliminary version of a part of this work has appeared in [Prasanna Venkatesan et al. 2010].

References

[Annadurai et al. 2008] Annadurai, S., Kalyani, T., Dare, V.R., Thomas, D.G: "Trajectory P System"; Progress in Natural Science, 18 (2008), 611–616.

[Ceterchi et al. 2003] Ceterchi, R., Mutyam, M., Păun, Gh., Subramanian, K.G.: "Array-rewriting P systems"; Natural Computing, 2 (2003), 229–249.

[Giammarresi and Restivo 1997] Giammarresi, D., Restivo, A.: "Two-dimensional languages"; In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, Volume 3 (1997), 215–267, Springer-Verlag.

[Mateescu et al. 1998] Mateescu, A., Rozenberg, G., Salomaa, A.: "Shuffle on trajectories: syntactic constraints"; Theoretical Computer Science, 197 (1998), 1–56.

[Mutyam et al. 2004] Mutyam, M., Jaya Prakash, V., Krithivasan, K.: "Rewriting tissue P systems"; Journal of Universal Computer Science, 10(9) (2004), 1250–1271.

[Niu et al. 2010] Niu, Y., Xu, J., Subramanian, K.G.: "P systems with shuffle operation"; Proceedings of the Fifth IEEE International Conference on Bio-Inspired Computing: Theories and Applications, 2 (2010), 1482–1486.

[Păun 1998] Păun, Gh.: "Computing with membranes"; Journal of Computer System Sciences, 61(1) (1998), 108–143.

[Păun 2002] Păun, Gh.: "Membrane Computing. An Introduction"; Berlin, Germany; Springer-Verlag, 2002.

[Păun et al. 1995] Păun, Gh., Rozenberg, G., Salomaa, A., "Grammars based on shuffle operation"; Journal of Universal Computer Science, 1(1) (1995), 67–82.

[Păun et al. 2010] Păun, Gh., Rozenberg, G., Salomaa, A. (Eds.): "The Oxford Handbook of Membrane Computing"; Oxford Univ. Press., (2010).

[Păun et al. 2000] Păun, Gh., Yokomori, T.: "Simulating H systems by P systems"; Journal of Universal Computer Science, 6(1) (2000), 178–193.

[Prasanna Venkatesan et al. 2010] Prasanna Venkatesan, A.S., Thomas, D.G., Robinson, T., Atulya K Nagar: "Trajectory array P system"; Proceedings of the Fifth IEEE International Conference on Bio-Inspired Computing: Theories and Applications, 2 (2010), 1543–1549.

[Siromoney et al. 1972] Siromoney, G., Siromoney, R., Krithivasan, K.: "Abstract Families of Matrices and Picture Languages"; Computer Graphics Image Processing, 1 (1972), 234–307.

[Siromoney et al. 1973] Siromoney, G., Siromoney, R., Krithivasan, K.: "Picture Languages with array rewriting rules"; Information and Control, 22 (1973), 447–470.