Computing with Membranes and Picture Arrays

A.S. Prasanna Venkatesan[a], D.G. Thomas[b], T. Robinson[b], Atulya K Nagar[c]

[a] *Department of Mathematics, B.S. Abdur Rahman University*
Chennai - 600 048, India
[b] *Department of Mathematics, Madras Christian College*
Tambaram, Chennai - 600 059, India
[c] *Department of Computer Science, Liverpool Hope University*
Hope Park, Liverpool, L16 9JD, United Kingdom

Abstract

Splicing systems were introduced by Tom Head [3] on biological considerations to model certain recombinant behaviour of DNA molecules. An effective extension of this operation to images was introduced by Helen Chandra et al. [5] and H array splicing systems were considered. A new method of applying the splicing operation on images of hexagonal arrays was introduced by Thomas et al. [12] and generated a new class of hexagonal array languages HASSL. On the other hand, P systems, introduced by Paun [6] generating rectangular arrays and hexagonal arrays have been studied in the literature, bringing together the two areas of theoretical computer science namely membrane computing and picture languages. P system with array objects and parallel splicing operation on arrays is introduced as a simple and effective extension of P system with operation of splicing on strings and this new class of array languages is compared with the existing families of array languages. Also we propose another P system with hexagonal array objects and parallel splicing operation on hexagonal arrays is introduced and this new class of hexagonal array languages is compared with the existing families of hexagonal array languages.

Keywords: Membrane Computing, P system, Rectangular arrays, Hexagonal arrays, Parallel splicing.

Email addresses: prasannaram@bsauniv.ac.in (A.S. Prasanna Venkatesan), dgthomasmcc@yahoo.com (D.G. Thomas), robin.mcc@gmail.com (T. Robinson), nagara@hope.ac.uk (Atulya K Nagar)

1. Introduction

Models based on biological phenomena that were introduced in the literature enriched both formal language theory and life science with major developments. Splicing system is one such model introduced by Head [3] based on biological considerations. The splicing systems make use of a new operation, called splicing on strings of symbols. Helen Chandra et al. [5] extended this operation and introduced a new method of splicing on images of rectangular arrays.

On the other hand, P systems introduced by Paun [6] are a class of distributed parallel computing devices of biochemical inspiration. These systems are based on a structure of finitely many cell membranes which are hierarchically arranged. All cell membranes are embedded in a main membrane called skin membrane. The membranes delimit regions where objects, elements of a finite alphabet, and evolution rules present. Evolution rules may contain target indicators; here indicates that the resulting object remains in the same membrane where it is produced; out indicates that the resulting object is sent to the region surrounding the membrane in which it is produced; in indicates that the resulting object is sent to a membrane which is contained in that membrane. Many variants of P systems have been introduced and extensively studied which can be seen in [7].

Also, in the study of picture generation, several grammars were introduced in the literature to generate various classes of pictures. One such grammar is the hexagonal kolam array grammar (HKAG) introduced by Siromoneys [10], generating a class of hexagonal arrays (HKAL). A new method of applying the parallel splicing operation on images of hexagonal arrays that involve 2×1 or 1×2 dominoes along x or y directions are introduced in [12].

In this paper, we introduce a new P system called array splicing P system where the objects are rectangular arrays and the evolution rules are parallel splicing rules as introduced in [5]. We compare the family of languages generated by this system with existing families of array languages like local two-dimensional array languages. Also we propose another P system called parallel splicing Hexagonal Array P System where the objects are hexagonal arrays and the evolution rules are parallel splicing rules as introduced in [12]. We compare the family of hexagonal array languages generated by this system with existing families of hexagonal array languages like hexagonal local picture languages.

2. Basic Definitions

In this section, we recall the basic definition of array languages, hexagonal picture languages and parallel splicing rules over array and hexagonal pictures as in [5, 12]. For the basic definition of P system and its variants, we refer to [6, 7].

Definition 2.1. *Let V be a finite alphabet. A picture A over V is a rectangular $m \times n$ array of elements of the form*

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

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 2.2. *Let V be an alphabet, # and \$ are two symbols that are not in V. A vertical domino is of the form* $\boxed{\begin{matrix} a \\ b \end{matrix}}$, *and a row domino is of the form* $\boxed{a \mid b}$, *where $a, b \in V \cup \{\lambda\}$.*

A domino column splicing rule over V is of the form $p : y_1 \# y_2 \$ y_3 \# y_4$, where $y_i = \boxed{\begin{matrix} a \\ b \end{matrix}}$ or $\boxed{\begin{matrix} \lambda \\ \lambda \end{matrix}}$, $1 \le i \le 4$.

A domino row splicing rule over V is of the form $q : x_1 \# x_2 \$ x_3 \# x_4$, where $x_i = \boxed{a \mid b}$ or $x_i = \boxed{\lambda \mid \lambda}$, $1 \le i \le 4$.

$$\text{Let } X = \begin{matrix} a_{11} & \dots & a_{1,j} & a_{1,j+1} & \dots & a_{1p} \\ a_{21} & \dots & a_{2,j} & a_{2,j+1} & \dots & a_{2p} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & \dots & a_{m,j} & a_{m,j+1} & \dots & a_{mp} \end{matrix}$$

$$\text{and } Y = \begin{matrix} b_{11} & \dots & b_{1,k} & b_{1,k+1} & \dots & b_{1q} \\ b_{21} & \dots & b_{2,k} & b_{2,k+1} & \dots & b_{2q} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{m1} & \dots & b_{m,k} & b_{m,k+1} & \dots & b_{mq} \end{matrix}$$

We write $(X, Y) \vdash_c Z$ if there exist column splicing rules $p_1, p_2, \dots, p_{m-1}$, not all necessarily different, such that

$$p_i = \boxed{\begin{matrix} a_{i,j} \\ a_{i+1,j} \end{matrix}} \# \boxed{\begin{matrix} a_{i,j+1} \\ a_{i+1,j+1} \end{matrix}} \$ \boxed{\begin{matrix} b_{i,k} \\ b_{i+1,k} \end{matrix}} \# \boxed{\begin{matrix} b_{i,k+1} \\ b_{i+1,k+1} \end{matrix}}$$

for all i, $(1 \leq i \leq m-1)$, and for some j, k $(1 \leq j \leq p-1,\ 1 \leq k \leq q-1)$, and

$$Z = \begin{matrix} a_{11} & \dots & a_{1,j} & b_{1,k+1} & \dots & b_{1q} \\ a_{21} & \dots & a_{2,j} & b_{2,k+1} & \dots & b_{2q} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & \dots & a_{m,j} & b_{m,k+1} & \dots & b_{mq} \end{matrix}$$

In a similar way, row splicing operation of two images U and V of sizes $p \times n$ and $q \times n$ using row splicing rules will produce an image W as given below:

$$\text{Let } U = \begin{matrix} a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{i,1} & a_{i,2} & \dots & a_{i,n} \\ a_{i+1,1} & a_{i+1,2} & \dots & a_{i+1,n} \\ \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pn} \end{matrix}$$

$$\text{and } V = \begin{matrix} b_{11} & b_{12} & \dots & b_{1n} \\ \dots & \dots & \dots & \dots \\ b_{k,1} & b_{k,2} & \dots & b_{k,n} \\ b_{k+1,1} & b_{k+1,2} & \dots & b_{k+1,n} \\ \dots & \dots & \dots & \dots \\ b_{q1} & b_{q2} & \dots & b_{qn} \end{matrix}.$$

We write $(U, V) \vdash_r W$ if there exist row splicing rules $q_1, q_2, \dots, q_{n-1}$, not all necessarily different, such that

$$q_j = \boxed{a_{i,j} \mid a_{i,j+1}} \# \boxed{a_{i+1,j} \mid a_{i+1,j+1}} \$ \boxed{b_{k,j} \mid b_{k,j+1}} \# \boxed{b_{k+1,j} \mid b_{k+1,j+1}}$$

for all j, $(1 \leq j \leq n-1)$, and for some i, k $(1 \leq i \leq p-1,\ 1 \leq k \leq q-1)$, and

$$W = \begin{matrix} a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{i,1} & a_{i,2} & \dots & a_{i,n} \\ b_{k+1,1} & b_{k+1,2} & \dots & b_{k+1,n} \\ \dots & \dots & \dots & \dots \\ b_{q1} & b_{q2} & \dots & b_{qn} \end{matrix}$$

Example 2.1. *Let* $X = \begin{bmatrix} a & a & a \\ b & b & b \\ a & a & a \end{bmatrix}$, $Y = \begin{bmatrix} a & a & a \\ a & a & a \\ a & a & a \end{bmatrix}$,

$$P_1 = \begin{array}{|c|}\hline a \\ \hline a \\ \hline \end{array} \# \begin{array}{|c|}\hline a \\ \hline a \\ \hline \end{array} \$ \begin{array}{|c|}\hline a \\ \hline b \\ \hline \end{array} \# \begin{array}{|c|}\hline a \\ \hline b \\ \hline \end{array}$$

$$P_2 = \begin{array}{|c|}\hline a \\ \hline a \\ \hline \end{array} \# \begin{array}{|c|}\hline a \\ \hline a \\ \hline \end{array} \$ \begin{array}{|c|}\hline b \\ \hline a \\ \hline \end{array} \# \begin{array}{|c|}\hline b \\ \hline a \\ \hline \end{array}$$

$$P_3 = \begin{array}{|c|}\hline a \\ \hline \lambda \\ \hline \end{array} \# \begin{array}{|c|}\hline a \\ \hline \lambda \\ \hline \end{array} \$ \begin{array}{|c|}\hline a \\ \hline \lambda \\ \hline \end{array} \# \begin{array}{|c|}\hline a \\ \hline \lambda \\ \hline \end{array}$$

then applying the parallel splicing rules P_1, P_2, P_3 *we get*

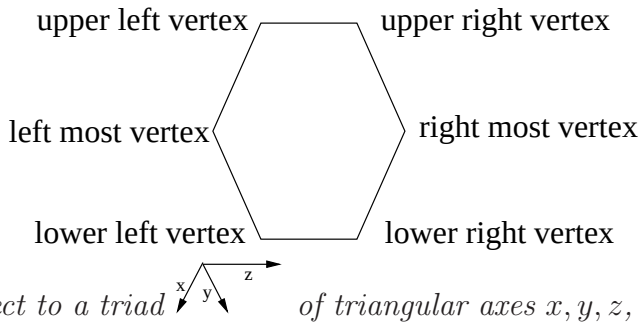
$$Z = \begin{bmatrix} a & a & a & a \\ b & b & a & a \\ a & a & a & a \end{bmatrix}$$

Definition 2.3. *Let* V *be a finite alphabet of symbols. A hexagonal picture* p *over* V *is a hexagonal array of symbols of* V*. For example, a hexagonal picture over the alphabet* $\{a, b\}$ *is :*

$$\begin{array}{ccccc} & a & & a & \\ a & & b & & b \\ & b & & a & \end{array}$$

The set of all hexagonal arrays over the alphabet V *is denoted by* V^{**H}*. A hexagonal picture language* L *over* V *is a subset of* V^{**H}*.*

We consider hexagons of the type :



With respect to a triad of triangular axes x, y, z*, the coordinates of each element of a hexagonal picture can be fixed.*

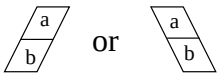
Definition 2.4. *Given a picture* $p \in V^{**H}$*, let* $l_1(p)$ *denote the number of elements in the border of* p *from upper left vertex to left most vertex in the direction* $\swarrow$ *called* x *direction,* $l_2(p)$ *denote the number of elements in the*

border of p from upper right vertex to right most vertex in the direction $\searrow$ called y direction and $l_3(p)$ denote the number of elements in the border of p from upper left vertex to upper right vertex in the direction $\rightarrow$ called z direction.

The directions are fixed with origin of reference as the upper left vertex, having coordinates $(1,1,1)$. The triple $(l_1(p), l_2(p), l_3(p))$ is called the size of the picture p.

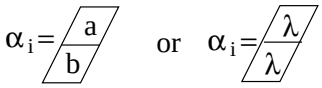
Given a hexagonal picture p of size (l,m,n), for $g \leq l, h \leq m$ and $k \leq n$, we denote by $B_{g,h,k}(p)$ the set of all hexagonal subpictures (called hexagonal blocks) of p of size (g,h,k). Each member of $B_{2,2,2}(p)$ is called a hexagonal tile.

Definition 2.5. *Let V be an alphabet. Let $¢, \$$ be two special symbols, not in V. A domino over V is of the form*

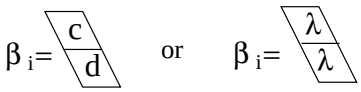


for $a, b \in V \cup \{\lambda\}$.

A domino x direction splicing rule over V is of the form $r = \alpha_1 ¢ \alpha_2 \$ \alpha_3 ¢ \alpha_4$ where each

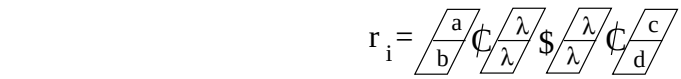


for some $a, b \in V \cup \{\lambda\}$ where λ is the empty word ($1 \leq i \leq 4$).
A domino y direction splicing rule over V is of the form $r = \beta_1 ¢ \beta_2 \$ \beta_3 ¢ \beta_4$ where each



for some $a, b \in V \cup \{\lambda\}$, $(1 \leq i \leq 4)$.

Definition 2.6. *Given two hexagonal arrays X and Y of sizes (l,m,n) and (l,m,n') respectively, i.e., $X = \langle a_{ijk} \rangle$, $1 \leq i \leq l, 1 \leq j \leq m, 1 \leq k \leq n$; $Y = \langle b_{i'j'k'} \rangle, 1 \leq i' \leq l, 1 \leq j' \leq m, 1 \leq k' \leq n'$, we write $(X,Y) \underset{x}{\vdash} W$ if there exist x direction splicing rules $r_1, r_2, \ldots, r_p$ (not all different) such that each r_i is of the form*



and $W = \langle c_{ijk'} \rangle$, $1 \leq i \leq l$, $1 \leq j \leq m$, $1 \leq k' \leq n''$, $n'' \geq n'$ and n.

We now say that W is obtained from X and Y by domino x direction splicing in parallel.

We can similarly define domino y direction splicing in parallel.

Let R_x^z and R_y^z denote the finite set of domino x direction splicing rules and domino y direction splicing rules respectively. By applying x direction splicing rules and y direction splicing rules we can generate the language of hexagonal arrays p of sizes $(\ell_1(p), \ell_2(p), \ell_3(p))$ where $\ell_1(p)$ and $\ell_2(p)$ are fixed and $\ell_3(p)$ varies.

3. Array Splicing *P* System

In this section, we define an array splicing P system and examine the power of this system.

Definition 3.1. *An array splicing P system (of degree m, $m \geq 1$) is a construct $\Pi = (V \cup \{\#, \$\}, \mu, L_1, L_2, \ldots, L_m, (R_1, \rho_1), (R_2, \rho_2), \ldots, (R_m, \rho_m), i_0)$ where*

(i) V is an alphabet; $\#, \$$ are special symbols not in V.

(ii) μ is a membrane structure consisting of m membranes (labeled with $1, 2, \ldots, m$).

(iii) $L_i \subseteq V^{**}$, $(1 \leq i \leq m)$ are finite arrays called axiom arrays over V associated with the regions $1, 2, \ldots, m$ of μ.

(iv) R_i, $1 \leq i \leq m$ are finite sets of evolution rules associated with the regions $1, 2, \ldots, m$ of μ given in the following form: $(\{r\}, tar)$ where $\{r = \alpha_1 \# \alpha_2 \$ \alpha_3 \# \alpha_4\}$ is a set of domino column splicing rules or a domino row splicing rules as given in Definition 2.2 and $tar \in \{here, out\} \cup \{in_j / 1 \leq j \leq m\}$. ρ_i is a partial order relation over R_i, $1 \leq i \leq m$ specifying a priority relation among rules of R_i.

(v) i_0 is the output membrane.

When an object is present in a region of the system, it is assumed to appear in arbitrarily many copies.

Any m-tuple $(M_1, M_2, \ldots, M_m)$ of array languages over V is called a configuration of Π. For any two configurations $(M_1, M_2, \ldots, M_m)$ and $(M_1^*, M_2^*, \ldots, M_m^*)$, we write $(M_1, M_2, \ldots, M_m) \vdash^* (M_1^*, M_2^*, \ldots, M_m^*)$ if we can pass from $(M_1, M_2, \ldots, M_m)$ to $(M_1^*, M_2^*, \ldots, M_m^*)$ by applying the splicing rules from each region of μ in parallel to all possible arrays of the corresponding regions and the target indications associated with the rules. More precisely, if

$x, y \in M_i \subset V^{**}$ and $\{\{r = \alpha_1\#\alpha_2\$\alpha_3\#\alpha_4\}, tar\} \in R_i$ such that we can have $(x, y) \vdash_r w$, then w will go to the region indicated by tar. If (i) $tar = here$, then the generated array remains in i^{th} membrane, (ii) $tar = out$, then the generated array is moved to the region immediately outside the membrane i, (iii) $tar = in_j$, then the generated array is moved to membrane j, provided that this is immediately inside i^{th} membrane; if not, the rule cannot be applied. We note that the generated arrays x and y are still available in the region i, but if the generated array w is sent out of region i, then no copy of it remains here.

A sequence of transitions between configurations of a given P system Π, starting from the initial configuration $(L_1, L_2, \ldots, L_m)$ is called a computation with respect to Π. The result of a computation consists of all arrays over V which are sent to the membrane i_0 (output membrane) at any time during the computation. Instead of output membrane we can also collect the output which are sent out of the skin membrane. We denote by $L(\Pi)$, the language of all of this type. We say that $L(\Pi)$ is generated by Π.

We denote by $SPLHA$, the family of languages generated by array splicing P systems as above.

Example 3.1. *Consider the array splicing P system*

$$\Pi = (V \cup \{\#, \$\}, \mu, L_1, L_2, L_3, (R_1, \rho_1), (R_2, \rho_2), (R_3, \rho_3), 3)$$

where
$V = \{a, b, c, x\}$, $\mu = [_3[_2[_1]_1]_2]_3$

$$L_1 = \left\{ I_1 = \begin{bmatrix} a & x & b \\ a & x & b \\ a & x & b \end{bmatrix}, \quad I_2 = \begin{bmatrix} b & x & c \\ b & x & c \\ b & x & c \end{bmatrix} \right\}$$

$$R_1 = \left\{ \left(r_1 : \left\{ p_1 : \boxed{\begin{matrix}\lambda\\x\end{matrix}} \# \boxed{\begin{matrix}\lambda\\b\end{matrix}} \$ \boxed{\begin{matrix}\lambda\\a\end{matrix}} \# \boxed{\begin{matrix}\lambda\\x\end{matrix}} ; p_2 : \boxed{\begin{matrix}x\\x\end{matrix}} \# \boxed{\begin{matrix}b\\b\end{matrix}} \$ \boxed{\begin{matrix}a\\a\end{matrix}} \# \boxed{\begin{matrix}x\\x\end{matrix}} ; \right.\right.\right.$$

$$\left.\left.\left. p_3 : \boxed{\begin{matrix}x\\\lambda\end{matrix}} \# \boxed{\begin{matrix}b\\\lambda\end{matrix}} \$ \boxed{\begin{matrix}a\\\lambda\end{matrix}} \# \boxed{\begin{matrix}x\\\lambda\end{matrix}} \right\}, \{here, out\} \right),$$

$$\left(r_2 : \left\{ p_4 : \boxed{\begin{matrix}\lambda\\x\end{matrix}} \# \boxed{\begin{matrix}\lambda\\c\end{matrix}} \$ \boxed{\begin{matrix}\lambda\\b\end{matrix}} \# \boxed{\begin{matrix}\lambda\\x\end{matrix}} ; p_5 : \boxed{\begin{matrix}x\\x\end{matrix}} \# \boxed{\begin{matrix}c\\c\end{matrix}} \$ \boxed{\begin{matrix}b\\b\end{matrix}} \# \boxed{\begin{matrix}x\\x\end{matrix}} ; \right.\right.$$

$$p_6 : \left[\begin{array}{c} x \\ \lambda \end{array}\right] \# \left[\begin{array}{c} c \\ \lambda \end{array}\right] \$ \left[\begin{array}{c} b \\ \lambda \end{array}\right] \# \left[\begin{array}{c} x \\ \lambda \end{array}\right] \Big\}, \{here, out\}\Big)\Big\}$$

$\rho_1 = \phi$
$L_2 = \phi$

$$R_2 = \Big\{ r_1 : \Big\{ p_7 : \left[\begin{array}{c} \lambda \\ b \end{array}\right] \# \left[\begin{array}{c} \lambda \\ \lambda \end{array}\right] \$ \left[\begin{array}{c} \lambda \\ \lambda \end{array}\right] \# \left[\begin{array}{c} \lambda \\ b \end{array}\right] ; p_8 : \left[\begin{array}{c} b \\ b \end{array}\right] \# \left[\begin{array}{c} \lambda \\ \lambda \end{array}\right] \$ \left[\begin{array}{c} \lambda \\ \lambda \end{array}\right] \# \left[\begin{array}{c} b \\ b \end{array}\right] ;$$

$$p_9 : \left[\begin{array}{c} b \\ \lambda \end{array}\right] \# \left[\begin{array}{c} \lambda \\ \lambda \end{array}\right] \$ \left[\begin{array}{c} \lambda \\ \lambda \end{array}\right] \# \left[\begin{array}{c} b \\ \lambda \end{array}\right] \Big\}, \{here, out\}\Big\}$$

$\rho_2 = \phi, \quad L_3 = \phi$
$R_3 = \phi, \quad \rho_3 = \phi.$

In membrane 1, the two axiom arrays I_1 and I_2 are present in multiple copies. By applying the domino column splicing rules of r_1 in parallel on the axiom array I_1 with itself we get a new array $\begin{bmatrix} a & x & x & b \\ a & x & x & b \\ a & x & x & b \end{bmatrix}$.

Similarly by applying the domino column splicing rules of r_2 in parallel on the axiom array I_2 with itself we get another new array $\begin{bmatrix} b & x & x & c \\ b & x & x & c \\ b & x & x & c \end{bmatrix}$. Now the new array generated can be sent out of the membrane to the membrane 2.

In membrane 2, by applying the domino column splicing rules in parallel to the arrays got from membrane 1, we get a new array $\begin{bmatrix} a & x & x & b & b & x & x & c \\ a & x & x & b & b & x & x & c \\ a & x & x & b & b & x & x & c \end{bmatrix}$ which can be represented as $(a_3^1 x_3^2 b_3^2 x_3^2 c_3^1)$ where subscript represents the number of rows and superscript represents the number of columns in which that element appears. This will be sent to the outer membrane 3.

If we apply the rules in membrane 1, iteratively we get two arrays of the form $(a_3^1 x_3^m b_3^1)$ and $(b_3^1 x_3^n c_3^1)$, $m, n \geq 1$.

At any step, these 2 arrays can be sent to membrane 2 and applying the splicing rules in membrane 2, we get an array $(a_3^1 x_3^m b_3^2 x_3^n c_3^1)$.

Hence the language generated by the system Π consists of arrays with three rows and any number of columns with left border of a's, right border of c's and middle part with bb and inner part with x's.

i.e., $L(\Pi) = \{a_3^1 x_3^m b_3^2 x_3^n c_3^1 | m, n \geq 1\}$.

Example 3.2. *Consider the array splicing P system*

$$\Pi = (V \cup \{\#, \$\}, \mu, L_1, L_2, (R_1, \rho_1), (R_2, \rho_2), 2)$$

where
$V = \{a, b\}$,

$\mu = [_2[_1]_1]_2$

$$L_1 = \left\{ \begin{bmatrix} a & b \\ a & a \end{bmatrix} \right\}$$

$$R_1 = \left\{ \left(\left(r_1 : \left\{ \begin{array}{l} q_1 : \boxed{\lambda}\boxed{a} \# \boxed{\lambda}\boxed{a} \$ \boxed{\lambda}\boxed{\lambda} \# \boxed{\lambda}\boxed{a} ; \\ q_2 : \boxed{a}\boxed{b} \# \boxed{a}\boxed{a} \$ \boxed{\lambda}\boxed{\lambda} \# \boxed{a}\boxed{b} ; \\ q_3 : \boxed{b}\boxed{b} \# \boxed{a}\boxed{a} \$ \boxed{\lambda}\boxed{\lambda} \# \boxed{b}\boxed{b} ; \\ q_4 : \boxed{b}\boxed{\lambda} \# \boxed{a}\boxed{\lambda} \$ \boxed{\lambda}\boxed{\lambda} \# \boxed{a}\boxed{\lambda} ; \end{array} \right\}, \{here, out\} \right), \right.$$

$$\left. \left(r_2 : \left\{ \begin{array}{l} p_1 : \begin{matrix}\boxed{\lambda}\\\boxed{b}\end{matrix} \# \begin{matrix}\boxed{\lambda}\\\boxed{\lambda}\end{matrix} \$ \begin{matrix}\boxed{\lambda}\\\boxed{a}\end{matrix} \# \begin{matrix}\boxed{\lambda}\\\boxed{b}\end{matrix} ; \\ p_2 : \begin{matrix}\boxed{b}\\\boxed{b}\end{matrix} \# \begin{matrix}\boxed{\lambda}\\\boxed{\lambda}\end{matrix} \$ \begin{matrix}\boxed{a}\\\boxed{a}\end{matrix} \# \begin{matrix}\boxed{b}\\\boxed{b}\end{matrix} ; \\ p_3 : \begin{matrix}\boxed{b}\\\boxed{a}\end{matrix} \# \begin{matrix}\boxed{\lambda}\\\boxed{\lambda}\end{matrix} \$ \begin{matrix}\boxed{a}\\\boxed{a}\end{matrix} \# \begin{matrix}\boxed{b}\\\boxed{a}\end{matrix} ; \\ p_4 : \begin{matrix}\boxed{a}\\\boxed{\lambda}\end{matrix} \# \begin{matrix}\boxed{\lambda}\\\boxed{\lambda}\end{matrix} \$ \begin{matrix}\boxed{a}\\\boxed{\lambda}\end{matrix} \# \begin{matrix}\boxed{a}\\\boxed{\lambda}\end{matrix} \end{array} \right\}, \{here, out\} \right) \right\}$$

$\rho_1 = \phi$
$L_2 = \phi$
$R_2 = \phi$
$\rho_2 = \phi$.

In membrane 1, the initial array $\begin{bmatrix} a & b \\ a & a \end{bmatrix}$ will be present in multiple number of copies. By applying the parallel row splicing rules r_1 once, we

get a new array $\begin{bmatrix} a & b \\ a & b \\ a & a \end{bmatrix}$. This new array will be sent out the membrane if $tar = out$ and it will remain in the same membrane if we use $tar = here$.

On the other hand, if we apply the parallel column splicing rules r_2 first, we get a new array $\begin{bmatrix} a & b & b \\ a & a & a \end{bmatrix}$. This will be sent out of the membrane if $tar = out$ and it will remain in the same membrane if $tar = here$.

In this way, by applying the parallel row splicing rules r_1 iteratively, the row size can be increased and by applying the parallel column splicing rules r_2 iteratively, the column size of the array can be increased.

Hence the above system Π generates the picture language of all $m \times n$ arrays $(m \geq 2, n \geq 2)$ describing the token L of a's (where b's are interpreted as blanks) as shown in Figure 1.

$$\begin{array}{ccccc} a & b & b & b & b \\ a & b & b & b & b \\ a & b & b & b & b \\ a & a & a & a & a \end{array}$$

Figure 1: Array describing token L

Definition 3.2. *For $p \in V^{**}$, let $p^{\#}$ be the rectangular array obtained by surrounding p with a special boundary symbol $\# \notin V$.*

For example if $p = \begin{array}{ccc} a & a & a \\ a & a & a \\ a & a & a \end{array}$ *, then*

$$p^{\#} = \begin{array}{ccccc} \# & \# & \# & \# & \# \\ \# & a & a & a & \# \\ \# & a & a & a & \# \\ \# & a & a & a & \# \\ \# & \# & \# & \# & \# \end{array}.$$

Theorem 3.1. *The family $SPLHA$ and the classes LOC of local array languages are incomparable but not disjoint.*

Proof. The picture language consisting of all $m \times n$ arrays $(m \geq 2, n \geq 2)$ describing the token L of a's as in Example 3.2 is a local array language [2].

The picture language L of all images over an alphabet $V = \{a\}$ with three columns (as shown in Figure 2) is not in LOC.

But this can be generated by a splicing array P system given by

$$\Pi = (V \cup \{\#, \$), \mu, L_1, L_2, (R_1, \rho_1), (R_2, \rho_2), 2)$$

where
$V = \{a\}$, $\mu = [_2[_1]_1]_2$
$L_1 = \{[a \;\; a \;\; a]\}$

$$R_1 = \left\{ r_1 : \left\{ \begin{array}{l} q_1 : \boxed{\lambda}\boxed{a} \# \boxed{\lambda}\boxed{\lambda} \$ \boxed{\lambda}\boxed{\lambda} \# \boxed{\lambda}\boxed{a} \; ; \\ q_2 : \boxed{a}\boxed{a} \# \boxed{\lambda}\boxed{\lambda} \$ \boxed{\lambda}\boxed{\lambda} \# \boxed{a}\boxed{a} \; ; \\ q_3 : \boxed{a}\boxed{\lambda} \# \boxed{\lambda}\boxed{\lambda} \$ \boxed{\lambda}\boxed{\lambda} \# \boxed{a}\boxed{\lambda} \end{array} \right\}, \{here, out\} \right\}$$

$\rho_1 = \phi$, $L_2 = \phi$
$R_2 = \phi$, $\rho_2 = \phi$.

In membrane 1, by applying the parallel row splicing rules r_1 iteratively, we get the picture language L of all images over the singleton $\{a\}$ with three columns.

$$\begin{array}{ccc} a & a & a \\ a & a & a \\ a & a & a \end{array}$$

Figure 2: An image over $\{a\}$ with three columns

It is known that the picture language of square images in which diagonal positions carry the symbol a but the remaining positions carry the symbol b (as shown in Figure 3) is in LOC [2]. But this array language cannot be generated by a splicing array P system since it is clear that the equality in the number of rows and columns cannot be maintained as the row and column splicing are independently done.

□

Definition 3.3. *[5] A parallel internal contextual array grammar is a system $G = (V, B, C, R, \varphi_c, \varphi_r)$ where V is an alphabet, B is a finite subset of V^{**} called the base of G, C is a finite subset of $V^{**}\$_c V^{**}$ called column array*

$$\begin{array}{cccc} a & b & b & b \\ b & a & b & b \\ b & b & a & b \\ b & b & b & a \end{array}$$

Figure 3:

contexts, R is a finite subset of $V^{**}\$_c V^{**}$ called row array contexts, $\varphi_c : V^{**} \to 2^C$ and $\varphi_r : V^{**} \to 2^R$ are the choice mappings which perform the parallel internal column and row contextual operations. When φ_c and φ_r are omitted, we call G as a parallel internal contextual array grammar without choice.

The direct derivation with respect to G is a binary relation $\Rightarrow$ on V^{**} and is defined as $X \Rightarrow_{in} Y$, where $X, Y \in V^{**}$ if and only if $X = X_1 \bigoplus X_2 \bigoplus X_3$, $Y = X_1 \bigoplus W \bigoplus X_2 \bigoplus Z \bigoplus X_3$ or $X = X_1 \ominus X_2 \ominus X_3$, $Y = X_1 \ominus W \ominus X_2 \ominus Z \ominus X_3$ for some $X_1, X_2, X_3 \in V^{**}$, and W, Z are contexts obtained by the parallel internal column or row contextual operations according to the choice mappings. $\Rightarrow^*_{in}$ is the reflexive and transitive closure of the relation $\Rightarrow_{in}$. Let $G = (V, B, C, R, \varphi_c, \varphi_r)$ be a parallel internal contextual array grammar. The language generated by G, denoted by $L(G)$ is defined as

$$L(G) = \{Y \in V^{**} / \text{ there exists } X \in B \text{ such that } X \Rightarrow^*_{in} Y\}$$

We write $X \overset{\$_c}{\Rightarrow} Y$ (respectively $X \overset{\$_c}{\Rightarrow} Y$) when column (respectively row) operations are performed on X to yield Y.

The family of all parallel internal contextual array languages is denoted by PIAC.

Theorem 3.2. *The family $SPLHA$ intersects the family $PIAC$.*

Proof. Consider the splicing array P system

$$\Pi = (V \cup \{\#, \$\}, \mu, L_1, L_2, (R_1, \rho_1), (R_2, \rho_2), 2)$$

where
$V = \{a, x, b\}$
$\mu = [_2[_1]_1]_2$

$$L_1 = \begin{bmatrix} b & a & b \\ a & x & a \\ a & x & a \\ b & a & b \end{bmatrix}$$

$$R_1 = \Big\{\Big(r_1 : \Big\{p_1 : \begin{bmatrix}\lambda\\a\end{bmatrix} \# \begin{bmatrix}\lambda\\b\end{bmatrix} \$ \begin{bmatrix}\lambda\\b\end{bmatrix} \# \begin{bmatrix}\lambda\\a\end{bmatrix};$$
$$p_2 : \begin{bmatrix}a\\x\end{bmatrix} \# \begin{bmatrix}b\\a\end{bmatrix} \$ \begin{bmatrix}b\\a\end{bmatrix} \# \begin{bmatrix}a\\x\end{bmatrix};$$
$$p_3 : \begin{bmatrix}x\\x\end{bmatrix} \# \begin{bmatrix}a\\a\end{bmatrix} \$ \begin{bmatrix}a\\a\end{bmatrix} \# \begin{bmatrix}x\\x\end{bmatrix};$$
$$p_4 : \begin{bmatrix}x\\a\end{bmatrix} \# \begin{bmatrix}a\\b\end{bmatrix} \$ \begin{bmatrix}a\\b\end{bmatrix} \# \begin{bmatrix}x\\a\end{bmatrix};$$
$$p_5 : \begin{bmatrix}a\\\lambda\end{bmatrix} \# \begin{bmatrix}b\\\lambda\end{bmatrix} \$ \begin{bmatrix}b\\\lambda\end{bmatrix} \# \begin{bmatrix}a\\\lambda\end{bmatrix} \Big\}, \{here, out\}\Big),$$

$$\Big(r_2 : \Big\{p_6 : \begin{bmatrix}\lambda\\a\end{bmatrix} \# \begin{bmatrix}\lambda\\a\end{bmatrix} \$ \begin{bmatrix}\lambda\\b\end{bmatrix} \# \begin{bmatrix}\lambda\\a\end{bmatrix};$$
$$p_7 : \begin{bmatrix}a\\x\end{bmatrix} \# \begin{bmatrix}a\\x\end{bmatrix} \$ \begin{bmatrix}b\\a\end{bmatrix} \# \begin{bmatrix}a\\x\end{bmatrix};$$
$$p_8 : \begin{bmatrix}x\\x\end{bmatrix} \# \begin{bmatrix}x\\x\end{bmatrix} \$ \begin{bmatrix}a\\a\end{bmatrix} \# \begin{bmatrix}x\\x\end{bmatrix};$$
$$p_9 : \begin{bmatrix}x\\a\end{bmatrix} \# \begin{bmatrix}x\\a\end{bmatrix} \$ \begin{bmatrix}a\\b\end{bmatrix} \# \begin{bmatrix}x\\a\end{bmatrix};$$
$$p_{10} : \begin{bmatrix}a\\\lambda\end{bmatrix} \# \begin{bmatrix}a\\\lambda\end{bmatrix} \$ \begin{bmatrix}b\\\lambda\end{bmatrix} \# \begin{bmatrix}a\\\lambda\end{bmatrix} \Big\}, \{here, out\}\Big)\Big\}$$

$\rho_1 = \{r_1 < r_2\}$
$L_2 = \phi$,
$R_2 = \phi$,
$\rho_2 = \phi$.

In membrane 1, by applying the parallel splicing rules given in r_1 once, we get a rectangular picture $\begin{matrix} b & a & a & b \\ a & x & x & a \\ a & x & x & a \\ b & a & a & b \end{matrix}$. This will be sent out of the membrane

if $tar = out$ or it will remain in the same membrane if $tar = here$. Now we can apply rules in r_1 or r_2. Since we have a priority relation, we must use the splicing rules given in r_2 which increases the column size. By applying the rules in r_2 iteratively, we get a family of rectangular pictures with x's, the four sides of x's are surrounded by a's and the remaining by b's. We note that this language is also generated by $PIAC$ grammar [5]. □

4. Parallel Splicing Hexagonal Array P System

In this section, we introduce a new array P system with hexagonal arrays as objects and parallel splicing rules as evolution rules and compare the new class of hexagonal array languages generated by this proposed system with the existing families of hexagonal array languages.

Definition 4.1. *A hexagonal array splicing P system (of degree $m, m \geq 1$) is a construct*

$$\Pi = (V \cup \{\lambda, \$, \notin\}, \mu, L_1, L_2, \ldots, L_m, (R_1, \rho_1), (R_2, \rho_2), \ldots, (R_m, \rho_m), i_0)$$

where

(i) V is a finite alphabet; λ, $\$$, $\notin$ are special symbols not in V.

(ii) μ is a membrane structure consisting of m membranes (labeled with $1, 2, \ldots, m$).

(iii) $L_i \subseteq V^{**H}$, $(1 \leq i \leq m)$ are finite sets of hexagonal arrays called axiom arrays over V associated with regions $1, 2, \ldots, m$ of μ.

(iv) R_i, $1 \leq i \leq m$ are finite sets of evolution rules associated with the regions $1, 2, \ldots, m$ of μ given in the following form: $(\{r\}, tar)$ where $\{r = \alpha_1 \notin \alpha_2 \$ \alpha_3 \notin \alpha_4\}$ is a set of domino x direction splicing rules or domino y direction splicing rules as given in Definition 2.5 and $tar \in \{here, out\} \cup \{in_j | 1 \leq j \leq m\}$. ρ_i is a partial order relation over R_i, $1 \leq i \leq m$ specifying a priority relation among rules of R_i.

(v) i_0 is the output membrane.

When an object is present in a region of the system, it is assumed to appear in arbitrarily many copies.

Any m-tuple $(H_1, H_2, \ldots, H_m)$ of hexagonal array language over V is called a configuration of Π. For any two configurations $(H_1, H_2, \ldots, H_m)$ and $(H'_1, H'_2, \ldots, H'_m)$, we write $(H_1, H_2, \ldots, H_m) \vdash^* (H'_1, H'_2, \ldots, H'_m)$ if we

can pass from $(H_1, H_2, \ldots, H_m)$ *to* $(H'_1, H'_2, \ldots, H'_m)$ *by applying the parallel domino x direction splicing rules or y direction splicing rules from each region of* μ*, in parallel to all possible hexagonal arrays of the corresponding regions, and following the target indications associated with the rules. More precisely, if* $a, b \in H_i \subseteq V^{**H}$ *and* $\{\{r = \alpha_1 ¢ \alpha_2 \$ \alpha_3 ¢ \alpha_4\}, tar\} \subseteq R_i$ *such that* $(a, b) \vdash_r w$*, then* w *is moved to the region indicated by* tar*. If* $tar = here$*, then the generated array remains in the same* i^{th} *membrane where it is generated, if* $tar = out$*, then the generated array is moved to the region immediately outside the membrane* i *and if* $tar = in_j$*, then the generated array is moved to* j^{th} *membrane if* j *is immediately inside the membrane* i*. If not, the rule cannot be applied.*

A sequence of transitions between configurations of a given P system Π*, starting from the initial configuration* $(L_1, L_2, \ldots, L_m)$ *is called a computation with respect to* Π*. The result of a computation consists of all hexagonal arrays over* V *which are sent to the output membrane* i_0 *at any time during the computation.*

The set of all hexagonal arrays computed by a Parallel Splicing Hexagonal Array P System (PSHAPS) Π *is denoted by* $HASPL(\Pi)$*. The family of all hexagonal array languages* $HASPL(\Pi)$ *generated by such systems* Π*, with at most m membranes, is denoted by* $PSHAP_m$*.*

Example 4.1. *Consider the hexagonal array splicing P system*

$$\Pi = (V \cup \{\lambda, ¢, \$\}, [_2[_1]_1]_2, L_1, \phi, (R_1, \rho_1), (\phi, \phi), 2)$$

where $V = \{a\}$*,*

$$L_1 = \left\{ \begin{array}{ccccc} & a & & a & \\ a & & a & & a \\ & a & & a & \end{array} \right\}$$

$R_1 = \{\{ r_1 : \frac{a}{a} ¢ \frac{\lambda}{\lambda} \$ \frac{\lambda}{\lambda} ¢ \frac{a}{a} \}. \{here, out\}\}$

$\rho_1 = \phi$*.*

Initially, in membrane 1, the axiom hexagonal array given by L_1 *is present in arbitrarily many copies. Two copies of them will be spliced using the parallel domino x-direction splicing rule* r_1 *as follows:*

$$\left(\begin{array}{ccccccc} & a & & a & & \lambda & \\ a & & a & & \lambda & & a \\ & a & & \lambda & & a & \end{array} , \begin{array}{ccccccc} & a & & \lambda & & a & \\ a & & \lambda & & a & & a \\ & a & & a & & a & \end{array} \right) \vdash \begin{array}{ccccccc} & a & & a & & a & \\ a & & a & & a & & a \\ & a & & a & & a & \end{array}$$

The generated hexagonal array is sent to membrane 2 by using tar = out. It remains in the same membrane if tar = here. By applying the rule r_1 iteratively and sending the generated arrays to output membrane we get a language of hexagonal arrays given by

$$L(\Pi) = \left\{ \begin{array}{ccccc} & a & & a & \\ a & & a & & a \\ & a & & a & \end{array} \;,\; \begin{array}{ccccccc} & a & & a & & a & \\ a & & a & & a & & a \\ & a & & a & & a & \end{array} \;, \ldots \right\}.$$

Hence the system Π generates the language of hexagonal arrays of sizes $(2, 2, n)$, $n \geq 2$.

Definition 4.2. *For $p \in V^{**H}$, let $p^{\#}$ be the hexagonal array obtained by surrounding p with a special boundary symbol $\# \notin V$.*

For example if $\begin{array}{ccc} & a & a \\ a & & b & & b \\ & b & a \end{array}$ *, then*

$$p^{\#} = \begin{array}{ccccccc} & \# & & \# & & \# & \\ \# & & a & & a & & \# \\ & a & & b & & b & \\ \# & & b & & a & & \# \\ & \# & & \# & & \# & \end{array}.$$

Definition 4.3. *A hexagonal picture language $L \subseteq V^{**H}$ is called local if there exists a finite set Δ of hexagonal tiles over $V \cup \{\#\}$ such that $L = \{p \in V^{**H} / B_{2,2,2}(p^{\#}) \subseteq \Delta\}$. L is denoted by $L(\Delta)$.*

The family of local hexagonal picture languages is denoted by $HLOC$.

Theorem 4.1. *The classes of $HLOC$ and $PSHAP_m$ are incomparable but not disjoint.*

Proof. Consider the hexagonal local array language L given in [1]. A member of L is given below:

$$\begin{array}{ccccccccc} & & 1 & & 0 & & 0 & & \\ & 0 & & 0 & & 1 & & 0 & \\ 0 & & 1 & & 0 & & 0 & & 1 \\ & 0 & & 0 & & 1 & & 0 & \\ & & 1 & & 0 & & 0 & & \end{array}$$

The above hexagonal array cannot be generated by a parallel splicing hexagonal array P system since the evolution rules are x direction splicing or y direction splicing rules and by definition they cannot generate only hexagonal arrays of sizes (ℓ, m, n) with $\ell = m = n$.

Hence $L \in HLOC$, but $L \notin PSHAP_m$.

The language given in Example 4.1 is in $PSHAP_m$, but is not in $HLOC$ [12].

Now we give a parallel splicing hexagonal array P system that generates a member of $HLOC$.

$$\Pi = (V \cup \{\lambda, ¢, \$\}, [_2[_1]_1]_2, L_1, \phi, (R_1, \rho_1), (\phi, \phi), 2)$$

where $V = \{1, 2, 3\}$

$$L_1 = \left\{ \begin{array}{ccccc} & 1 & & 1 & \\ 2 & & 2 & & 2 \\ & 3 & & 3 & \end{array} \right\}$$

R_1 = {{ r_1 : $\frac{1}{2}¢\frac{\lambda}{\lambda}\$\frac{\lambda}{\lambda}¢\frac{1}{2}$, r_2 : $\frac{2}{3}¢\frac{\lambda}{\lambda}\$\frac{\lambda}{\lambda}¢\frac{2}{3}$ }. {here, out}}

$\rho_1 = \phi$.

In membrane 1, by applying the domino x-direction splicing rules r_1 and r_2 parallely and using the target indicator *out* we generate the family of hexagonal arrays of sizes $(2, 2, n)$, $n \geq 2$. Hence

$$L(\Pi) = \left\{ \begin{array}{ccccccccccc} & 1 & & 1 & & & 1 & & 1 & & 1 & & \\ 2 & & 2 & & 2 & , & 2 & & 2 & & 2 & & 2 & , \ldots \\ & 3 & & 3 & & & 3 & & 3 & & 3 & & \end{array} \right\}$$

which is in $HLOC$. □

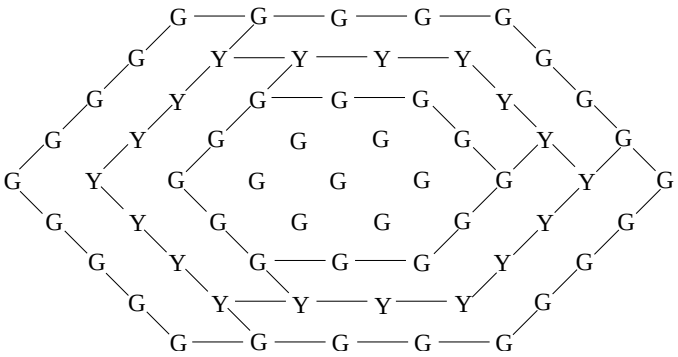
Definition 4.4. *[10] A hexagonal kolam array grammar G is a 5-tuple $G = (V, I, P, S, \mathcal{L})$ where $V = V_1 \cup V_2$, V_1 is a finite set of non-terminals and V_2 is a finite set of intermediates; I is a finite set of terminal letters; $P = P_1 \cup P_2$ where P_1 is a finite set of non-terminals rules of the form*
$S \to S_1 \nearrow a$ (upper right catenation),
$S \to S_1 \searrow b$ (lower right catenation),
$S \to S_1 \leftarrow c$ (left most catenation)
where $S, S_1 \in V_1, a, b, c \in V_2$ and $\nearrow$ is upper right arrow head catenation, $\searrow$ is

lower right arrow head catenation and $\overleftarrow{\oplus}$ *is leftmost arrow head catenation.* P_2 *is a terminal rule of the form* $S \to H$ *where* $S \in V_1$ *and* H *is a hexagonal array over* I*.* S *is a start symbol;* $\mathcal{L}$ *is a set of intermediate languages corresponding to each one of the* k *intermediates in* V_2*. These intermediate languages are regular or context-free or context-sensitive string languages written in the appropriate arrow head form. An arrow head is written in the form* $\{\cdots\langle v\rangle\cdots\}$ *where* $\langle v\rangle$ *denotes the vertex and arrow head is written in the clockwise direction. A hexagonal kolam array grammar is called* $(R:R)$*,* $(R:CF)$*,* $(R:CS)$ *accordingly as all the members of* $\mathcal{L}$ *are regular, or at least one of* $\mathcal{L}$ *is context-free or at least one of* $\mathcal{L}$ *is context-sensitive.*

Derivations proceed as follows: For the first stage of derivation, rules in P_1 *and* P_2 *are applied sequentially (introducing parentheses along with the arrow head direction) until all the non-terminals are replaced. For the second stage of derivation, starting from the innermost parentheses, each intermediate is replaced in parallel by an arrow head of the intermediate language.*

Theorem 4.2. *The classes of* $HKAGL$ *and* $PSHAP_m$ *are incomparable but not disjoint.*

Proof. Consider the following two hexagonal arrays



There exists a hexagonal kolam array grammar which generates a language consisting of the above two hexagonal arrays [10]. But this language

cannot be generated by a parallel splicing hexagonal array P system since the splicing rules cannot maintain the positions of G and Y.

The language given in Example 4.1 can be generated by a hexagonal kolam array grammar $G = (V, I, P, L, S)$ where $V = V_1 \cup V_2$ with $V_1 = \{S\}$, $V_2 = \{x\}$, $I = \{a\}$, $P = P_1 \cup P_2$ with $P_1 = \{S \to S \oplus x\}$, $P_2 = \left\{ S \to \begin{array}{ccccc} & & a & & a \\ & a & & a & \\ a & & a & & a \\ & & a & & a \end{array} \right\}$, $L = \{L_x\}$ where $L_x = \{a^n < a > a^n\}$, $n = 1$.

Then

$$L(G) = \left\{ \begin{array}{ccccccccccc} & & a & & a & & & & a & & a \\ & a & & a & & a & & a & & a & \\ a & & a & & a & , & a & & a & & a \\ & & a & & a & & & & a & & a \end{array} , \ldots \right\}.$$

Hence the classes $HKAGL$ and $PSHAP_m$ are not disjoint. Consider the parallel splicing hexagonal array P system given by

$$\Pi = (V \cup \{\lambda, ¢, \$\}, [_2[_1]_1]_2, L_1, \phi, (R_1, \rho_1), (\phi, \phi), 2)$$

where $V = \{1, 2, 3\}$,

$$L_1 = \left\{ \begin{array}{ccccc} & 1 & & 1 & \\ 1 & & 2 & & 1 \\ & 3 & & 3 & \end{array} \right\}$$

$R_1 = \{\{ r_1 : \frac{1}{2}¢\frac{\lambda}{\lambda}\$\frac{\lambda}{\lambda}¢\frac{1}{2}$, $r_2 : \frac{2}{3}¢\frac{\lambda}{\lambda}\$\frac{\lambda}{\lambda}¢\frac{2}{3}$ \}. {here, out}\}

and $\rho_1 = \phi$.

By applying the splicing rules r_1 ad r_2 in parallel, the above system generates the hexagonal array language given by

$$L(\Pi) = \left\{ \begin{array}{ccccccccccc} & 1 & & 1 & & & 1 & & 1 & & 1 \\ 1 & & 2 & & 1 & , & 1 & & 2 & & 2 & & 1 \\ & 3 & & 3 & & & & 3 & & 3 & & 3 & \end{array} , \right.$$

$$\left. \begin{array}{ccccccccccc} & & & & 1 & & 1 & & 1 & & 1 \\ & & & 1 & & 2 & & 2 & & 2 & & 1 \\ & & & & 3 & & 3 & & 3 & & 3 & & \end{array} , \ldots \right\}$$

The above language cannot be generated by a $HKAG$ since if we apply the rule $S \to \begin{array}{ccc} & 1 & \\ 1 & & 1 \\ & 2 & \\ & 3 & \end{array}$ in P_2, other elements of the language of $HKAG$

are constructed either through the west arrow catenation or east arrow catenation and hence the first element of the language must be either a prefix or a suffix hexagonal array of the second element of the language. In general, the n^{th} element of the language must be a prefix or a suffix array of the $(n+1)^{th}$ element which is not in L. □

Definition 4.5. *[10] A controlled table 0L hexagonal array model is a 4-tuple $(V, \mathcal{P}, H_0, C)$ where V is a finite alphabet; $\mathcal{P}$ is a finite set of right up $\nearrow$, right down $\searrow$, left $\leftarrow$ tables $\{P_1, P_2, \ldots, P_k\}$, each table consisting of a finite set of rules of the form $a \to bc$, $a, b, c \in V$. H_0 is the axiom; C is the control language which may be either regular or context-free or context-sensitive and the model is called regular or CF or CS controlled according as C is regular or context-free or context-sensitive.*

The tables bear the appropriate arrow head sign to show the application in the proper direction. Derivations proceed in parallel along any one of the three pairs of edges of an arrow head by applying the rules in the appropriate table. Tables are applied sequentially according to control words in the control language C.

Theorem 4.3. *The classes of regular controlled table 0L hexagonal array languages (regular $CT0LHAL$) and $PSHAP_m$ are incomparable but not disjoint.*

Proof. Consider the language

$$L = \left\{ \begin{array}{ccccccccccccccccccc} & & & & & & & & a & & a & & a & & & & \\ & & a & & a & & & & a & & a & & a & & a & & \\ a & & & a & & a & , & a & & a & & a & & a & & a & , \ldots \\ & & a & & a & & & & a & & a & & a & & a & & \\ & & & & & & & & & a & & a & & a & & & \end{array} \right\}$$

which is a regular controlled table 0L hexagonal array language [12]. This cannot be generated by a parallel splicing hexagonal array P system since from the definition of domino x-direction and y-direction splicing rules, it is clear that hexagonal arrays of sizes (ℓ, m, n) with $\ell = m$ $(\ell \geq 2, m \geq 2, n \geq 2)$ only can be generated.

The language given in Example 4.1 can be generated by a regular controlled table 0L hexagonal array grammar [11].

Also $L(\Pi)$ given in Theorem 4.2 cannot be generated by any regular controlled table 0L hexagonal array grammar, since if we allow any of the rule $11 \leftarrow 1$ or $12 \leftarrow 1$, we get a different language. □

5. Conclusion

In this paper, a new P system on pictures with arrays as objects and parallel splicing rules as evolution rules has been proposed. The new class of array languages generated by the proposed system has been compared with the existing families of array languages like local two-dimensional array languages. Another variant on pictures with hexagonal arrays as objects and parallel splicing rules on hexagonal arrays as evolution rules has been proposed. The new class of hexagonal array languages generated by the proposed system has been compared with the existing families of hexagonal array languages like local hexagonal array languages, hexagonal kolam array languages, etc. Further studies like generative power, descriptional complexity, other comparison results and universality results can be done. A preliminary version of this paper can be seen in [9].

References

[1] Dersanambika, K.S., Krithivasan, K., Martin-Vide, C. and Subramanian, K.G.: Hexagonal Pattern Languages. Lecture Notes in Computer Science. **3322**. (2004) 52–64.

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

[3] Head, T.: Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours. Bull. Math. Biology. **49** (1987) 735–759.

[4] Head, T., Paun, Gh. and Pixton, D.: Language theory and molecular genetics: Generative mechanisms suggested by DNA recombination. Handbook of Formal Languages. **2**. Ch. 7. Eds. Rozenberg, G. and Salomaa, A. Springer-Verlag, (1997) 295–358.

[5] Helen Chandra, P., Subramanian, K.G. and Thomas, D.G.: Parallel splicing on images. Int. J. Pattern Recognition and Artificial Intelligence. World Scientific Publishing Company. **18**(6) (2004) 1071–1091.

[6] Paun, Gh.: Computing with membranes. Journal of Computer System Sciences. **61**(1) (1998) 108–143.

[7] Paun, Gh., Rozenberg, G. and Salomaa, A. (Eds.): The Oxford Handbook of Membrane Computing. Oxford Univ. Press. (2010).

[8] Prasanna Venkatesan, A.S. and Thomas, D.G.: Array Splicing P system. Proceedings of the National Conference on Recent Developments in Mathematics and its Applications. Excel India Publishers. (2011) 157–164.

[9] Prasanna Venkatesan, A.S. and Thomas, D.G: Computing with membranes and hexagonal arrays, Proceedings of the International Conference on Mathematics in Engineering and Business Management (ICMEB 2012), T.R. Publications Pvt. Ltd., **2** (2012), 5–8.

[10] Siromoney, G. and Siromoney, R.: Hexagonal Arrays and Rectangular Blocks. Computer Graphics and Image Processing. **5** (1976) 353–381.

[11] Subramanian, K.G.: Hexagonal Array Grammars. Computer Graphics and Image Processing. **10**(4) (1979) 388–394.

[12] Thomas, D.G., Begam, M.H. and David, N.G.: Hexagonal Array Splicing Systems. Ramanujan Mathematical Society. Lecture Notes Series, **3**, Eds. Krithivasan, K., Rama.R., (2007) 197–207.