ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume 21, Number 3, 2018, 298–309

Accepting H-Array Splicing Systems and Their Properties

D. K. SHEENA CHRISTY[1], V.MASILAMANI[2], D. G. THOMAS[3], Atulya K. NAGAR[4], and T. ROBINSON[5]

[1]Department of Mathematics, SRM University, Kattankulathur, Chennai - 603 203, India
[2]Department of Computer Science and Engineering, IIITD&M Kanchipuram, Chennai - 600 036, India
[3]Department of Science and Humanities, Saveetha School of Engineering, Chennai - 602 105, India
[4]Department of Mathematics and Computer Science, Liverpool Hope University
[5]Department of Mathematics, Madras Christian College, Tambaram, Chennai - 600 059, India
E-mail: sheena.lesley@gmail.com, masila@iiitdm.ac.in, dgthomasmcc@yahoo.com, nagara@hope.ac.uk, robinson@mcc.edu.in

Abstract. Mitrana et al. (2010) have introduced accepting splicing systems working in uniform and non-uniform way as a counterpart of the well investigated generating splicing systems. In this paper, we have extended these notions to H-array splicing systems studied by Helen Chandra et al. (2004) using parallel splicing on rectangular arrays. We have compared the accepting H-array splicing systems with existing two dimensional grammars such as local array systems and (Regular : Regular) array grammars in terms of their generative powers.

Key-words: Two-dimensional language, H-array splicing system, Iterated Splicing.

1. Introduction

There has been a lot of interest in the study of formal language theory in the frame work of DNA computing. The splicing operation has been used to model specific recombinant behaviours of DNA molecules which consist of "cutting" DNA sequences and then "pasting" the fragments again, under the action of restriction enzymes and ligases. In [3], Tom Head defined splicing systems motivated by this behaviour of DNA sequences. The splicing system makes use of a new operation, called splicing on strings of symbols. Gh. Păun et al. [9, 10] extended the definition of Head and defined extended H systems which are computationally universal. For more details of the study of splicing systems on strings, we refer to [4].

V. Mitrana et al. [8] proposed a novel view of splicing systems and defined the concept of an accepting splicing system. Two ways of iterating the splicing operation in the generating case

and the computational power of generating splicing systems defined by these operations were investigated [8]. Furthermore, two variants of accepting splicing systems were considered. All the accepting models have been compared with each other with respect to their computational power.

In syntactic approaches to generation and recognition of images or pictures considered as digitized arrays, several two-dimensional grammars have been proposed and studied [2, 11]. Krithivasan et al. [6] made an extension of the concept of splicing to arrays and defined array splicing systems. In [5] a new method of applying the splicing operation on rectangular arrays is introduced. This model is different from the model considered in [6]. The idea here is that each of two rectangular arrays is "cut" between two specified columns (respectively rows) and the "left" ("upper") part of the first array is "pasted" with the "right" ("lower") part of the second array, resulting in a new array and the "cut" and "paste" operations are according to a set of domino splicing rules.

The resulting model called H-array splicing system is compared with other generative mechanisms of picture languages [2, 11, 12]. Some closure results under geometric operations and language theoretic operations are considered. Dassow and Mitrana [1] investigated a very simple and natural restriction on the splicing operation, namely the cross-over rule applicable only to identical strings, in trying to capture certain features of the recombination of genes in a chromosome. In [5], the authors introduced self cross-over array systems and studied related properties.

In this paper, we propose the notion of accepting H-array splicing systems and study the computational power of these systems. We compare these systems with some existing models of two dimensional array grammars such as local array systems and (Regular : Regular) array grammars in terms of their generative powers.

2. Preliminaries

In this section, we deal with the basic concepts of picture languages and array splicing systems that generate picture languages.

Let V be a finite alphabet. A picture (or an image or an array) is an $m \times n$ matrix with entries from V and this picture is said to be of size $m \times n$, $m, n \geq 0$. If p is a picture then the number of rows and the number of columns of p are respectively denoted by $|p|_r$ and $|p|_c$. The collection of all pictures over V is denoted by V^{++}, and $V^{++} \cup \{\Lambda\}$ is denoted by V^{**}, where Λ denotes empty image with either m or $n = 0$. The set of all (one-dimensional or linear or horizontal) strings over V is denoted by V^+ and $V^+ \cup \{\lambda\}$ is denoted by V^*, where λ denotes empty string. V_* denotes the set of all vertical sequences of letters over V and $V_+ = V_* - \{\Lambda\}$.

We use the symbol $\ominus$ to denote the row concatenation for arrays and it is defined as follows:

if $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$ and $B = \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ b_{21} & \cdots & b_{2n} \\ \cdots & \cdots & \cdots \\ b_{p1} & \cdots & b_{pn} \end{bmatrix}$ then

$$A \ominus B = \begin{array}{|ccc|}\hline a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots \\ a_{m1} & \cdots & a_{mn} \\ b_{11} & \cdots & b_{1n} \\ b_{21} & \cdots & b_{2n} \\ \cdots & \cdots & \cdots \\ b_{p1} & \cdots & b_{pn} \\ \hline \end{array}$$

In similar way, we use the symbol $\bigoplus$ to denote column concatenation for arrays and it is defined as follows:

if $A = \begin{array}{|ccc|}\hline a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots \\ a_{m1} & \cdots & a_{mn} \\ \hline \end{array}$ and $B = \begin{array}{|ccc|}\hline b_{11} & \cdots & b_{1p} \\ b_{21} & \cdots & b_{2p} \\ \cdots & \cdots & \cdots \\ b_{m1} & \cdots & b_{mp} \\ \hline \end{array}$ then

$$A \bigoplus B = \begin{array}{|cccccc|}\hline a_{11} & \cdots & a_{1n} & b_{11} & \cdots & b_{1p} \\ a_{21} & \cdots & a_{2n} & b_{21} & \cdots & b_{2p} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1} & \cdots & a_{mn} & b_{m1} & \cdots & b_{mp} \\ \hline \end{array}$$

Note that the row concatenation for two images is defined only when the numbers of columns of those two images are equal, and the column concatenation for two images is defined only when the numbers of rows of those two images are the same.

Now we recall the row and column concatenations of two picture languages, and the H-array splicing system discussed in [5].

Let L_1 and L_2 be two picture languages over V. The row concatenation of L_1 and L_2 is defined as

$$L_1 \ominus L_2 = \{A \ominus B \mid A \in L_1, B \in L_2\}$$

The column concatenation of L_1 and L_2 is defined as

$$L_1 \bigoplus L_2 = \{A \bigoplus B \mid A \in L_1, B \in L_2\}$$

Definition 2..1. *Let V be an alphabet, # and \$ are two symbols that are not in V. A column domino is of the form* $\begin{array}{|c|}\hline a \\ \hline b \\ \hline \end{array}$*, and a row domino is of the form* $\begin{array}{|c|c|}\hline a & b \\ \hline \end{array}$*, where $a, b \in V$.*

A domino column splicing rule over V is of the form $\alpha : y_1 \# y_2 \$ y_3 \# y_4$, where $y_i =$ $\begin{array}{|c|}\hline a_l \\ \hline b_l \\ \hline \end{array}$ *or* $\begin{array}{|c|}\hline \lambda \\ \hline \lambda \\ \hline \end{array}$*, $1 \le i \le 4$, $a_l, b_l \in V$.*

A domino row splicing rule over V is of the form $\beta : x_1 \# x_2 \$ x_3 \# x_4$, where $x_i =$ $\begin{array}{|c|c|}\hline a_j & b_j \\ \hline \end{array}$ *or $x_i =$* $\begin{array}{|c|c|}\hline \lambda & \lambda \\ \hline \end{array}$*, $1 \le i \le 4$, $a_j, b_j \in V$.*

Let $X = \begin{array}{|cccccc|}\hline a_{11} & \cdots & a_{1,j} & a_{1,j+1} & \cdots & a_{1p} \\ a_{21} & \cdots & a_{2,j} & a_{2,j+1} & \cdots & a_{2p} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1} & \cdots & a_{m,j} & a_{m,j+1} & \cdots & a_{mp} \\ \hline \end{array}$

and $Y = \begin{array}{|cccccc|}\hline b_{11} & \dots & b_{1,k} & b_{1,k+1} & \dots & b_{1q} \\ b_{21} & \dots & b_{2,k} & b_{2,k+1} & \dots & b_{2q} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{m1} & \dots & b_{m,k} & b_{m,k+1} & \dots & b_{mq} \\ \hline \end{array}$

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

$$p_i = \begin{array}{|c|}\hline a_{i,j} \\ \hline a_{i+1,j} \\ \hline \end{array} \# \begin{array}{|c|}\hline a_{i,j+1} \\ \hline a_{i+1,j+1} \\ \hline \end{array} \$ \begin{array}{|c|}\hline b_{i,k} \\ \hline b_{i+1,k} \\ \hline \end{array} \# \begin{array}{|c|}\hline b_{i,k+1} \\ \hline b_{i+1,k+1} \\ \hline \end{array}$$

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

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

$Z' = \begin{array}{|cccccc|}\hline b_{11} & \dots & b_{1,k} & a_{1,j+1} & \dots & a_{1p} \\ b_{21} & \dots & b_{2,k} & a_{2,j+1} & \dots & a_{2p} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{m1} & \dots & b_{m,k} & a_{m,j+1} & \dots & a_{mp} \\ \hline \end{array}$

Let $U = \begin{array}{|cccc|}\hline a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{i,1} & a_{i,2} & \dots & a_{i,n} \\ a_{i+1,1} & a_{i+1,2} & \dots & a_{i+1,n} \\ \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pn} \\ \hline \end{array}$ and $V = \begin{array}{|cccc|}\hline b_{11} & b_{12} & \dots & b_{1n} \\ \dots & \dots & \dots & \dots \\ b_{k,1} & b_{k,2} & \dots & b_{k,n} \\ b_{k+1,1} & b_{k+1,2} & \dots & b_{k+1,n} \\ \dots & \dots & \dots & \dots \\ b_{q1} & b_{q2} & \dots & b_{qn} \\ \hline \end{array}$.

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

$$q_i = \begin{array}{|c|c|}\hline a_{i,j} & a_{i,j+1} \\ \hline \end{array} \# \begin{array}{|c|c|}\hline a_{i+1,j} & a_{i+1,j+1} \\ \hline \end{array} \$ \begin{array}{|c|c|}\hline b_{k,j} & b_{k,j+1} \\ \hline \end{array} \# \begin{array}{|c|c|}\hline b_{k+1,j} & b_{k+1,j+1} \\ \hline \end{array}$$

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

$W = \begin{array}{|cccc|}\hline a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{i,1} & a_{i,2} & \dots & a_{i,n} \\ b_{k+1,1} & b_{k+1,2} & \dots & b_{k+1,n} \\ \dots & \dots & \dots & \dots \\ b_{q1} & b_{q2} & \dots & b_{qn} \\ \hline \end{array}$ and $W' = \begin{array}{|cccc|}\hline b_{11} & b_{12} & \dots & b_{1n} \\ \dots & \dots & \dots & \dots \\ b_{k,1} & b_{k,2} & \dots & b_{k,n} \\ a_{i+1,1} & a_{i+1,2} & \dots & a_{i+1,n} \\ \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pn} \\ \hline \end{array}$

In [5] the definition of column (row) splicing operation of two arrays X and Y yields only one array Z (i.e., Z is obtained by taking the left (upper) part of X pasted with the right (lower) part of Y). But in this paper, we consider the definition of splicing in the usual way which yields two outputs after splicing [8, 9].

Definition 2..2. *A generating H-array splicing scheme is a triple $\sigma = (V, R_c, R_r)$ where V is an alphabet, R_c is a set of domino column splicing rules and R_r is a set of domino row splicing rules.*

Definition 2..3. *For a given generating H-array splicing scheme $\sigma = (V, R_c, R_r)$ and a language $L \subseteq V^{**}$, if $X, Y \in L$ then,*
$\sigma(X,Y) = \{Z, Z' \in V^{**} \mid (X,Y) \vdash_c \{Z, Z'\}$ *or* $(X,Y) \vdash_r \{Z, Z'\}\}$ *and*
$\sigma(L) = \bigcup_{X,Y \in L} \sigma(X,Y)$.

For two array languages L_1, L_2, we define $\sigma(L_1, L_2) = \bigcup_{X \in L_1, Y \in L_2} \sigma(X,Y)$.

The iterated application of σ is defined as follows:
$\sigma^0(L) = L$;
$\sigma^{i+1}(L) = \sigma^i(L) \cup \sigma(\sigma^i(L))$ *for* $i \geq 0$;
$\sigma^*(L) = \bigcup_{i=0}^{\infty} \sigma^i(L)$.

This method of defining iterated splicing can be called as uniform.

An H-array splicing system is defined by $S = (\sigma, A)$ where $\sigma = (V, R_c, R_r)$ and A is a subset of V^{**}, called the initial language. The column and row splicing operations use rules from R_c and R_r respectively. The language of S defined by $L(S)$ is $\sigma^*(A)$ and we call it as a splicing array language.

We denote the class of all H-array splicing systems by HAS and the family of all splicing array languages of H-array splicing systems by $\mathcal{L}(HAS)$.

3. Non-uniform Variant

In [5], the authors considered the splicing operation on images of rectangular arrays and studied the properties and the generative power of H-array splicing system. The iterated splicing defined using the system is such that the splicing at any step occurs between any two arrays generated in the previous step. We consider a different method of iterated splicing which we call as a non-uniform variant of iterated splicing using H-array splicing system. In this method the splicing is done only with axioms and splicing at any step occurs between a generated array in the previous step and an axiom. This method is analogous to the one considered by Mitrana et al. [8] in the case of strings and is useful to investigate the computational power of the accepting H-array splicing system. Now, we explain the method.

Definition 3..1. *For an H-array splicing system $S = (\sigma, A)$, the non-uniform variant of iterated splicing is defined as follows:*
$\tau^0(A) = A$;
$\tau^{i+1}(A) = \sigma(\tau^i(A), A)$, $i \geq 0$,
$\tau^*(A) = \bigcup_{i=0}^{\infty} \tau^i(A)$.

The language generated by H-array splicing system in a non-uniform way is $\tau^*(A)$ and the family of languages generated by a non-uniform H-array splicing system is denoted by $\mathcal{L}_n(HAS)$.

We illustrate a non-uniform variant of H- array splicing system with an example.

Example 3..1. *Let $S = (\sigma, A)$ be a non-uniform H-array splicing system with $\sigma = (V, R_c, R_r)$ where $V = \{a, x, b\}$, $A = \left\{ \begin{array}{|ccc|}\hline a & x & b \\ a & x & b \\ a & x & b \\ \hline \end{array} \right\}$.*

$R_c = \left\{ p_1 : \begin{array}{|c|}\hline x \\ \hline x \\ \hline \end{array} \# \begin{array}{|c|}\hline b \\ \hline b \\ \hline \end{array} \$ \begin{array}{|c|}\hline a \\ \hline a \\ \hline \end{array} \# \begin{array}{|c|}\hline x \\ \hline x \\ \hline \end{array} \; ; \; p_2 : \begin{array}{|c|}\hline b \\ \hline b \\ \hline \end{array} \# \begin{array}{|c|}\hline \lambda \\ \hline \lambda \\ \hline \end{array} \$ \begin{array}{|c|}\hline b \\ \hline b \\ \hline \end{array} \# \begin{array}{|c|}\hline \lambda \\ \hline \lambda \\ \hline \end{array} \right\}$ *and*

$R_r = \left\{ q_1 : \begin{array}{|cc|}\hline a & x \\ \hline \end{array} \# \begin{array}{|cc|}\hline \lambda & \lambda \\ \hline \end{array} \$ \begin{array}{|cc|}\hline \lambda & \lambda \\ \hline \end{array} \# \begin{array}{|cc|}\hline a & x \\ \hline \end{array} \; ; \right.$

$\left. q_2 : \begin{array}{|cc|}\hline x & b \\ \hline \end{array} \# \begin{array}{|cc|}\hline \lambda & \lambda \\ \hline \end{array} \$ \begin{array}{|cc|}\hline \lambda & \lambda \\ \hline \end{array} \# \begin{array}{|cc|}\hline x & b \\ \hline \end{array} \right\}$

Now, $\tau^0(A) = A$,

$$\tau^1(A) = \sigma(\tau^0(A), A)$$

$$= \sigma \left(\begin{array}{|ccc|}\hline a & x & b \\ a & x & b \\ a & x & b \\ \hline \end{array}, \begin{array}{|ccc|}\hline a & x & b \\ a & x & b \\ a & x & b \\ \hline \end{array} \right)$$

$$= \left\{ \Lambda, \begin{array}{|ccc|}\hline a & x & b \\ \hline \end{array}, \begin{array}{|ccc|}\hline a & x & b \\ a & x & b \\ \hline \end{array}, \begin{array}{|cc|}\hline a & b \\ a & b \\ a & b \\ \hline \end{array}, \begin{array}{|ccc|}\hline a & x & b \\ a & x & b \\ a & x & b \\ \hline \end{array}, \begin{array}{|cccc|}\hline a & x & x & b \\ a & x & x & b \\ a & x & x & b \\ \hline \end{array}, \right.$$

$$\left. \begin{array}{|ccc|}\hline a & x & b \\ a & x & b \\ a & x & b \\ a & x & b \\ \hline \end{array}, \begin{array}{|ccc|}\hline a & x & b \\ a & x & b \\ a & x & b \\ a & x & b \\ a & x & b \\ \hline \end{array}, \begin{array}{|ccc|}\hline a & x & b \\ a & x & b \\ a & x & b \\ a & x & b \\ a & x & b \\ a & x & b \\ \hline \end{array} \right\}$$

It is not hard to find that $\tau^*(A)$ generates the language $L = P \cup S \cup Q \cup B$, where $P = \{p \in V^{**} \mid p$ is a $3 \times n$ array, where $n \geq 3$, with all positions in first column carry symbol a, all positions in the last column carry symbol b and all other positions carry symbol $x\}$, $S = \{s \in V^{**} \mid s$ is an $m \times 3$ array, $m \geq 3$ with all positions in first column carry symbol a, all positions in the third column carry symbol b and all positions in the second column carry symbol $x\}$, $Q = \{q \in V^{**} \mid q$ is an $m \times 2$ array, $m \geq 3$ with all positions in the first column carry symbol a and all positions in the second column carry symbol $b\}$ and $B = \left\{ \Lambda, \begin{array}{|ccc|}\hline a & x & b \\ \hline \end{array}, \begin{array}{|ccc|}\hline a & x & b \\ a & x & b \\ \hline \end{array} \right\}$.

4. Accepting H-Array Splicing System

In this section we introduce the notion of accepting H-array splicing system with two methods of iterated splicing namely uniform and non-uniform methods. We study the generative power of this system and compare with that of well-known systems such as local array systems [5] and array rewriting systems $(R : R)AG$ and $(CF : CF)AG$ [12].

Definition 4..1. *An accepting H-array splicing system (AHASS) is a pair $\Gamma = (S, P)$ where $S = (\sigma, A)$ is a H-array splicing system and P is a finite subset of V^{**}. Let $w \in V^{**}$. We define the uniform method (usual way) of the iterated array splicing of Γ as follows:*

$\sigma^0(A, w) = \{w\}$;

$\sigma^{i+1}(A, w) = \sigma^i(A, w) \cup \sigma(\sigma^i(A, w) \cup A)$, $i \geq 0$.

$\sigma^*(A, w) = \bigcup_{i=0}^{\infty} \sigma^i(A, w)$.

The language accepted by an accepting H-array splicing system Γ is

$L(\Gamma) = \{w \in V^{**} \mid \sigma^*(A, w) \cap P \neq \phi\}$.

The class of languages accepted by accepting H-array splicing system is denoted by $\mathcal{L}(AHASS)$.

Let $\Gamma = (S, P)$ be an accepting H-array splicing system and $w \in V^{**}$. The non-uniform variant of iterated splicing of Γ is defined as follows.

$\tau^0(A, w) = \{w\}$;

$\tau^{i+1}(A, w) = \tau^i(A, w) \cup \sigma(\tau^i(A, w), A)$, $i \geq 0$,

$\tau^*(A, w) = \bigcup_{i=0}^{\infty} \tau^i(A, w)$.

An array $w \in V^{**}$ is said to be accepted by Γ in a non-uniform way if $\tau^*(A, w) \cap P \neq \phi$.

The language accepted by Γ in a non-uniform way is

$\mathcal{L}_n(\Gamma) = \{w \in V^{**} \mid \tau^*(A, w) \cap P \neq \phi\}$.

The class of all languages accepted by AHASS in non-uniform way is denoted by $\mathcal{L}_n(AHASS)$.

We illustrate an accepting H-array splicing system with a following example.

Example 4..1. *Let $\Gamma = (S, F)$ be an AHASS where $S = (\sigma, A)$, $\sigma = (V, R_c, R_r)$ with $V = \{a, b\}$, $P = \left\{ \boxed{\begin{matrix} a & b \\ b & a \end{matrix}} \right\}$ and $A = \{\Lambda\}$.*

$$R_c = \left\{ p_1 : \boxed{\begin{matrix} a \\ b \end{matrix}} \# \boxed{\begin{matrix} \lambda \\ \lambda \end{matrix}} \$ \boxed{\begin{matrix} \lambda \\ \lambda \end{matrix}} \# \boxed{\begin{matrix} b \\ a \end{matrix}} \; ; \; p_2 : \boxed{\begin{matrix} b \\ a \end{matrix}} \# \boxed{\begin{matrix} \lambda \\ \lambda \end{matrix}} \$ \boxed{\begin{matrix} \lambda \\ \lambda \end{matrix}} \# \boxed{\begin{matrix} a \\ b \end{matrix}} \right\}$$

$R_r = \phi$.

We illustrate the acceptance and rejection of arrays by Γ.

Let $w = \boxed{\begin{matrix} a & b & b \\ b & a & a \end{matrix}} \in V^{**}$.

Now $\sigma^0(A, w) = \{w\} = \left\{ \boxed{\begin{matrix} a & b & b \\ b & a & a \end{matrix}} \right\}$.

$\sigma^1(A, w) = \sigma^0(A, w) \cup \sigma(\sigma^0(A, w) \cup A)$.

$$= \left\{ \Lambda, \boxed{\begin{matrix} b \\ a \end{matrix}}, \boxed{\begin{matrix} a & b \\ b & a \end{matrix}}, \boxed{\begin{matrix} a & b & b \\ b & a & a \end{matrix}}, \boxed{\begin{matrix} a & b & b & b \\ b & a & a & a \end{matrix}}, \boxed{\begin{matrix} a & b & a & b & b \\ b & a & b & a & a \end{matrix}}, \right.$$

$$\left. \boxed{\begin{matrix} a & b & b & a & b & b \\ b & a & a & b & a & a \end{matrix}} \right\}$$

Clearly $\sigma^1(A, w) \cap P \neq \phi$. Hence w is accepted by Γ.

Let $u = \boxed{\begin{matrix} a & b & a \\ b & a & a \end{matrix}} \in V^{**}$.

Now $\sigma^0(A,u) = \{u\} = \left\{ \boxed{\begin{array}{ccc} a & b & a \\ b & a & a \end{array}} \right\}$

$$\sigma^1(A,u) = \left\{ \boxed{\begin{array}{c} a \\ a \end{array}}, \boxed{\begin{array}{ccc} a & b & a \\ b & a & a \end{array}}, \boxed{\begin{array}{ccccc} a & b & a & b & a \\ b & a & b & a & a \end{array}} \right\}$$

$$\sigma^2(A,u) = \left\{ \boxed{\begin{array}{c} a \\ a \end{array}}, \boxed{\begin{array}{ccc} a & b & a \\ b & a & a \end{array}}, \boxed{\begin{array}{ccccc} a & b & a & b & a \\ b & a & b & a & a \end{array}}, \boxed{\begin{array}{ccccccc} a & b & a & b & a & b & a \\ b & a & b & a & b & a & a \end{array}} \right\}$$

In general $\sigma^*(A,u) \cap P = \phi$. *Hence* u *is not accepted by* Γ.
The language accepted by this AHASS is

$$L = \{p \in V^{**} \mid p = \left(\boxed{\begin{array}{cc} a & b \\ b & a \end{array}} \right)^n, \left(\boxed{\begin{array}{ccc} a & b & b \\ b & a & a \end{array}} \right)^n, \boxed{\begin{array}{c} a \\ b \end{array}} \left(\boxed{\begin{array}{c} b \\ a \end{array}} \right)^n,$$
$$\left(\boxed{\begin{array}{c} a \\ b \end{array}} \right)^n \left(\boxed{\begin{array}{c} b \\ a \end{array}} \right), \left(\boxed{\begin{array}{cc} a & b \\ b & a \end{array}} \right)^n \boxed{\begin{array}{c} b \\ a \end{array}}, n \geq 1\}.$$

We now recall the definition of local array languages (LOC) [2] to compare the generative powers of AHASS and the systems that yield local array languages.

Given an array p of size $m \times n$, we denote by $B_{h,k}(p)$ for $h \leq m, k \leq n$, the set of all subpictures of p of size $h \times k$. We call a square picture of size 2×2 as a tile. Given a picture p of size $m \times n$, $\hat{p}$ denotes a picture of size $(m+2) \times (n+2)$ obtained by surrounding p with a special boundary symbol $\notin \notin V$.

A two dimensional language $L \subseteq V^{**}$ is local if there exists a finite set Θ of tiles over the alphabet $V \cup \{\not\epsilon\}$ such that $L = \{p \in V^{**} | B_{2,2}(\hat{p}) \subseteq \Theta\}$. The family of local picture languages is denoted by LOC.

Theorem 4..1. *(i)* $\mathcal{L}(AHASS) - LOC \neq \phi$

(ii) $LOC - \mathcal{L}(AHASS) \neq \phi$

Proof. (i) To prove that $\mathcal{L}(AHASS)$-$LOC \neq \phi$, let us consider the picture language L consists of all arrays over $V = \{x\}$ with three columns. This language is not in LOC [2] as it is not possible to fix the number of columns using only one symbol, i.e., the tile $\boxed{\begin{array}{cc} x & x \\ x & x \end{array}}$ can be moved without restriction on the number of columns. But it is accepted by an AHASS, $\Gamma = (S,P)$ where $S = (\sigma, A)$ with $\sigma = (V, R_c, R_r)$, $V = \{x\}$, $A = \{\Lambda\}$. $P = \{ \ x \quad x \quad x \ \}$.
$R_c = \phi$ and $R_r = \left\{ \boxed{\begin{array}{cc} x & x \end{array}} \# \boxed{\begin{array}{cc} \lambda & \lambda \end{array}} \$ \boxed{\begin{array}{cc} \lambda & \lambda \end{array}} \# \boxed{\begin{array}{cc} x & x \end{array}} \right\}$

(ii) Let us consider another example $L = \{p \in V^{**} \mid p$ is a square image in which diagonal positions carry 'a' but the remaining positions carry symbol 'b'$\}$. This language is in LOC [2]. But this array language cannot be accepted by any AHASS, with R_r being empty, since the row and column splicings are independently done and P is finite, there are many $w \in L$: $\sigma^*(A,w) \cap P = \phi$. To prove this statement, let us assume that there exists a AHASS with R_r empty, accepting L. Without loss of generality, let $\Gamma = (S,P)$ be an AHASS with $S = (\sigma, A)$, $\sigma = (V, R_c, R_r)$, $V = \{a,b\}$, $A \subseteq V^{**}$, $R_c \neq \phi$, $R_r = \phi$ and P is a finite subset of V^{**}.

Since $R_r = \phi$, for any $w \in L$, each member α of $\sigma^*(A, w)$ satisfies the condition $|\alpha|_r = |w|_r$. In order to have $w \in L(\Gamma)$, we need to take a member of $\sigma^*(A, w)$ to be in P. Since L contains pictures with arbitrary large number of rows, P should be infinite, which is a contradiction. Hence $L \neq L(\Gamma)$. Hence $LOC\text{-}\mathcal{L}(AHASS) \neq \phi$. □

Note: (ii) holds if AHASS is such that $R_c = \phi$.

We now recall the definition of a class of array grammars $(R : R)AG$ and its corresponding languages $(R : R)AL$ [12] and give a comparison result with non-uniform variant of AHASS.

Let $G = (V, I, P, S)$ be an array (rewriting) grammar (AG), where $V = V_1 \cup V_2$, V_1 a finite set of non-terminals, V_2 a finite set of intermediates, I is a finite set of terminals, $P = P_1 \cup P_2 \cup P_3$, P_1 is the finite set of non-terminal rules, P_2 the finite set of intermediate rules and P_3 the finite set of terminal rules. $S \in V_1$ is the start symbol. P_1 is a finite set of ordered pairs (u, v) (written $u \to v$), u and v are in $(V_1 \cup V_2)^+$ or u and v are in $(V_1 \cup V_2)_+$.

P_1 is called regular if $u \in V_1$ and v of the form $U \oplus V$, U in V_1 and V in V_2 or U in V_2 and V in V_1.

P_2 is a set of ordered pairs (u, v), u and v is in $(V_2 \cup \{x_1, \ldots, x_p\})^+$ or u and v in $(V_2 \cup \{x_1, \ldots, x_p\})_+$; $x_1, \ldots, x_p$ in I^{++} have same number of rows in the first case and same number of columns in the second case, i.e., the finite set of intermediate rules involve only intermediates and a finite number of fixed arrays in I^{++}. Further P_2 is such that each intermediate in V_2 generates either a language (called intermediate matrix language) whose terminals are a finite number of arrays with the same number of rows or the transpose of such a language.

P_3 the finite set of terminal rules are ordered pairs (u, v), u in $(V_1 \cup V_2)$ and v in I^{++}.

An array grammar (AG) is called $(R : R)AG$ if the nonterminals rules are regular and all the intermediate languages are regular.

$L = \{X \mid S \Rightarrow^*_G X, X \in I^{++}\}$ is a (regular:regular) array language ($(R : R)AL$) if there exists a $(R : R)AG$, G such that $L = L(G)$.

We now give a result connecting $\mathcal{L}_n(AHASS)$ and $(R : R)AL$. For this we require the notions of H-tokens with equal arms and H-tokens with unequal arms. An example of H-token with equal arm is $\begin{matrix} x & \cdot & x \\ x & \cdot & x \\ x & x & x \\ x & \cdot & x \\ x & \cdot & x \end{matrix}$, while $\begin{matrix} x & \cdot & x \\ x & x & x \\ x & \cdot & x \\ x & \cdot & x \end{matrix}$ is an example of a H-token with unequal arms.

We need the following notation to use in the next theorem. If X is a symbol, then X^v_u stands for a new array formed by X where subscript u denotes the number of rows and superscript v denotes the number of columns.

If Z is a set of symbols of V, then Z^v_u is the set of $u \times v$ arrays whose entries are in Z.

Theorem 4..2. *(i)* $\mathcal{L}_n(AHASS) - (R : R)AL \neq \phi$

(ii) $(R : R)AL - \mathcal{L}_n(AHASS) \neq \phi$ *if AHASS is such that* $R_r = \phi$.

Proof. (i) Let us consider the picture language $L' = \{p = \{0, 1\}^n_u, u = 2$ and $n \geq 1$ with at least one column of p must have the entries as 0 in first row and 1 in second row$\}$. This language cannot be generated by $(R : R)AG$, since each array in L' is such that a pattern (subarray) does not appear periodically in the array. But it can be accepted by the following non-uniform variant of AHASS.

Let us consider a variant AHASS $\Gamma = (S, P)$ where $S = (\sigma, A)$, $\sigma = (V, R_c, R_r)$, $V = \{1, 0\}$, $A = \left\{ \boxed{\begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix}} \right\}$ and $P = \left\{ \boxed{\begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix}} \right\}$.

$R_c = \left\{ \boxed{\begin{matrix} 0 \\ 1 \end{matrix}} \# \boxed{\begin{matrix} \lambda \\ \lambda \end{matrix}} \$ \boxed{\begin{matrix} 1 \\ 1 \end{matrix}} \# \boxed{\begin{matrix} 0 \\ 1 \end{matrix}} \right\}$ and

$R_r = \phi$.

Let $w = \boxed{\begin{matrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{matrix}} \in V^{**}$.

Now $\tau^0(A, w) = \boxed{\begin{matrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{matrix}}$.

$$\tau^1(A, w) = \tau^0(A, w) \cup \sigma(\tau^0(A, w), A)$$
$$= \left\{ \boxed{\begin{matrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{matrix}}, \boxed{\begin{matrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}}, \boxed{\begin{matrix} 1 & 0 \\ 1 & 0 \end{matrix}} \right\}$$

$$\tau^2(A, w) = \left\{ \boxed{\begin{matrix} 1 \\ 1 \end{matrix}}, \boxed{\begin{matrix} 1 & 0 \\ 1 & 0 \end{matrix}}, \boxed{\begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix}}, \boxed{\begin{matrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{matrix}}, \boxed{\begin{matrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}}, \boxed{\begin{matrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix}} \right\}$$

Hence it is easy to see that $\tau^2(A, w) \cap P \neq \phi$. In general, for all $w \in L'$, $\tau^*(A, w) \cap P \neq \phi$. Therefore L' is accepted by AHASS in a non-uniform way.
Hence $\mathcal{L}_n(AHASS) - (R : R)AL \neq \phi$.

(ii) To prove $(R : R)AL - \mathcal{L}_n(AHASS) \neq \phi$ if $R_r = \phi$, let us consider the picture language L consisting of arrays describing token H with equal arms. It cannot be accepted by any AHASS with R_r being empty in a non-uniform way, since P is finite and there are many $w \in L$ (infinite in number); $\tau^*(A, w) \cap P = \phi$. The proof of this statement is similar to the one discussed in Theorem 4..1. But L can be generated by a $(R : R)AG$ [12]. □

Theorem 4..3. *Let Γ be an AHASS with $R_r = \phi$. There exists a positive integer k such that if $p \in L(\Gamma)$ with $|p|_c \geq k$ then $p \bigoplus y \bigoplus p \in L(\Gamma)$ for any $y \in V^{**}$ with $|y|_r = |p|_r$.*

Proof. Let $\Gamma = (S, P)$ be an AHASS with $S = (\sigma, A)$ where $\sigma = (V, R_c, R_r)$, A an initial language, P is a finite subset of V^{**}, $R_c \neq \phi$. Let $k = \max\{|x|_c : x \in P\} + 1$. Let us assume that $p \in L(\Gamma)$ and $|p|_c \geq k$. Then $p \in (L(\Gamma) - P)$. We now prove that for any array $y \in V^{**}$ with $|y|_r = |p|_r$, $p \bigoplus y \bigoplus p \in L(\Gamma)$. In fact we show that $(\sigma^i(A, p) - \{p\}) \subseteq \sigma^i(A, p \bigoplus y \bigoplus p)$ for all $i \geq 1$. This is done by induction on i. The result is true for $i = 1$. Let $s \in (\sigma^{i+1}(A, p) - \{p\})$ for $i \geq 1$. If s is obtained by applying rules in R_c to a pair of arrays from $\sigma^i(A, p) - \{p\}$, then $s \in \sigma^{i+1}(A, p \bigoplus y \bigoplus p)$ by induction hypothesis. If s is obtained by applying column splicing rules to a pair of distinct arrays (p, x) (or) (x, p) then s can also be obtained by applying the same column splicing rules to the pair of arrays $(p \bigoplus y \bigoplus p, x)$ or $(x, p \bigoplus y \bigoplus p)$ respectively. Similarly if s is obtained by applying column splicing rule to pair (p, p), then s can also be derived by applying the same column splicing rules to the pair $(p \bigoplus y \bigoplus p, p \bigoplus y \bigoplus p)$. This proves the theorem. □

Similar to theorem 4..3, we have

Theorem 4..4. *Let Γ be an AHASS with $R_c = \phi$. There exists a positive integer s such that if $p \in L(\Gamma)$ with $|p|_r \geq s$, then $p \bigominus y \bigominus p \in L(\Gamma)$ for any $y \in V^{**}$ with $|p|_c = |y|_c$.*

Similar results hold good for languages accepted by AHASS in a non-uniform way. In other words, we have:

Theorem 4..5. *Let Γ be an AHASS with $R_r = \phi$. There exists an integer $k > 0$ such that if $w \in \mathcal{L}_n(\Gamma)$, with $|w|_c \geq k$ then either $w \bigoplus y \in \mathcal{L}_n(\Gamma)$ or $y \bigoplus w \in \mathcal{L}_n(\Gamma)$, where $y \in V^{**}$ and $|y|_r = |w|_r$.*

Theorem 4..6. *Let Γ be an AHASS with $R_c = \phi$. There exists an integer $s > 0$ such that if $w \in \mathcal{L}_n(\Gamma)$, with $|w|_r \geq s$ then either $w \ominus z \in \mathcal{L}_n(\Gamma)$ or $z \ominus w \in \mathcal{L}_n(\Gamma)$, where $z \in V^{**}$ and $|z|_c = |w|_c$.*

Remark 4..1. *In the definition 2..1, we make use of domino column and row splicing rules over V. We can extend the structures of these rules by considering arrays instead of dominoes, in order to obtain pictures of desired patterns.*

Theorem 4..7. $\mathcal{L}(AHASS) - \mathcal{L}_n(AHASS) \neq \phi$.

Proof. Let us consider the language $L = \{x \bigoplus p \bigoplus x / x = \boxed{\begin{matrix} b \\ b \end{matrix}}$ and $p \in Q$ where Q is a set of arrays over $\{a, b\}$ of sizes $2 \times n$, $n \geq 1\}$.

The language L can be accepted by the uniform variant of AHASS in the following way:

Let $\Gamma = (V, A, R_c, R_r, P)$ be a uniform variant of AHASS. Here $V = \{a, b\}$, $A = \{\Lambda\}$, $P = \begin{pmatrix} b & b \\ b & b \end{pmatrix}$, $R_c = \left\{ \boxed{\begin{matrix} b \\ b \end{matrix}} \# \boxed{\begin{matrix} \lambda \\ \lambda \end{matrix}} \$ \boxed{\begin{matrix} \lambda \\ \lambda \end{matrix}} \# \boxed{\begin{matrix} b \\ b \end{matrix}} \right\}$ and $R_r = \phi$.

L cannot be accepted by the non-uniform variant of any AHASS by Theorems 4..5 and 4..6. □

Theorem 4..8. *$(\mathcal{L}(HAS) \cap \mathcal{L}_n(HAS)) - (\mathcal{L}(AHASS) \cup \mathcal{L}_n(AHASS)) \neq \phi$, when either R_c or R_r is empty.*

Proof. Let us consider the array language $L = \{X_u^n \bigoplus Y_u^m \mid n, m \geq 0, u \geq 2\} - \{\Lambda\}$. This language lies in both $\mathcal{L}(HAS)$ and $\mathcal{L}_n(HAS)$. In fact, the following HAS $S = (\sigma, A)$ where $\sigma = (V, R_c, R_r)$; $V = \{X, Y\}$;

$$R_c = \left\{ p_1 = \boxed{\begin{matrix} X \\ X \end{matrix}} \# \boxed{\begin{matrix} Y \\ Y \end{matrix}} \$ \boxed{\begin{matrix} \lambda \\ \lambda \end{matrix}} \# \boxed{\begin{matrix} X \\ X \end{matrix}}, p_2 = \boxed{\begin{matrix} Y \\ Y \end{matrix}} \# \boxed{\begin{matrix} \lambda \\ \lambda \end{matrix}} \$ \boxed{\begin{matrix} X \\ X \end{matrix}} \# \boxed{\begin{matrix} Y \\ Y \end{matrix}} \right\},$$

$R_r = \phi$, $A = \left\{ \boxed{\begin{matrix} X & X & Y \\ X & X & Y \end{matrix}}, \boxed{\begin{matrix} X & Y & Y \\ X & Y & Y \end{matrix}} \right\}$ generates L both in uniform and non-uniform way. But it does not belong to either $\mathcal{L}(AHASS)$ or $\mathcal{L}_n(AHASS)$. This follows from theorems 4..3, 4..4, 4..5 and 4..6. □

5. Conclusion

In this paper, the study of parallel splicing on arrays [5] has been continued. We define the accepting H-array splicing systems both uniform and non-uniform, similar to that of strings done by Mitrana et al. [8] and studied its generative power of accepting H-array splicing system. We have the following question: whether $\mathcal{L}_n(AHASS)$ is strictly included in the class

of $\mathcal{L}(AHASS)$. Other related questions when both R_r and R_c are non-empty have to be analyzed. Certain decision problems such as membership problem and equivalence problem are also interesting. A preliminary version of this paper has appeared in [7].

References

[1] J. DASSOW and V. MITRANA, *Self cross-over system*, In Computing with Bio-Molecules: Theory and Experiments, Ed. Gh. Păun, Springer series in Discrete Mathematics and Theoretical Computer Science, pp. 283–294, 1998.

[2] D. GIAMMARRESI and A. RESTIVO, *Two-dimensional languages*, in Handbook of Formal Languages, eds. A. Salomaa and G. Rozenberg, **3**, Springer-Verlag, pp. 215–267, 1997.

[3] T. HEAD, *Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours*, Bulletin of Mathematical Biology, **49**, pp. 735–759, 1987.

[4] T. HEAD, Gh. PĂUN and D. PIXTON, *Language theory and molecular genetics: generative mechanisms suggested by DNA recomination*, in Handbook of Formal Languages, eds. G. Rozenberg and A. Salomaa, **2**(7), Springer-Verlag, pp. 296–358, 1997.

[5] P. Helen CHANDRA, K. G. SUBRAMANIAN and D. G. THOMAS, *Parallel splicing on images*, International Journal of Pattern Recognition and Artificial Intelligence, **18**, pp. 1071–1091, 2004.

[6] K. KRITHIVASAN, V. T. CHAKARAVARTHY and R. RAMA, *Array splicing systems*, in New Trends in Formal Languages, Control Cooperation and Combinatorics, Eds. Gh. Păun and A. Salomaa, Lecture Notes in Computer Science, Springer-Verlag, **1218**, pp. 346–365, 1997.

[7] V. MASILAMANI, D. K. SHEENA CHRISTY, D. G. THOMAS, A. K. NAGAR and R. THAMBURAJ, *Accepting H-array splicing systems*, Communications in Computer and Information Science, Springer, **472**, pp. 313–317, 2014.

[8] V. MITRANA, I. PETRE and V. ROGOJIN, *Accepting splicing systems*, Theoretical Computer Science, **411**, pp. 2414–2422, 2010.

[9] Gh. PĂUN, G. ROZENBERG and A. SALOMAA, *Computing by splicing*, Theoretical Computer Science, **168**, pp. 321–336, 1996.

[10] Gh. PĂUN, G. ROZENBERG and A. SALOMAA, *DNA Computing New Computing Paradigms*, Springer, Berlin, 1998.

[11] A. ROSENFELD and R. SIROMONEY, *Picture languages - A Survey*, Languages of Design, **1**, pp. 229–245, 1993.

[12] G. SIROMONEY, R. SIROMONEY and K. KRITHIVASAN, *Picture languages with array rewriting rules*, Information and Control, **22**, pp. 447–470, 1973.