International Journal of Computer Information Systems and Industrial Management Applications.
www.mirlabs.net/ijcisim/index.html

# Flat Splicing Array Grammar Systems Generating Picture Arrays

**G. Samdanielthompson[1], N. Gnanamalar David[1], Atulya K. Nagar[2] and K.G. Subramanian[2]**

[1]Department of Mathematics, Madras Christian College,
Tambaram, Chennai 600059 India
{*samdanielthompson,ngdmcc*}*@gmail.com*

[2]Department of Mathematics and Computer Science, Faculty of Science,
Liverpool Hope University, Liverpool, L16 9JD, UK
*nagara@hope.ac.uk*, *kgsmani1948@yahoo.com*

***Abstract***: **While studying the recombinant behaviour of DNA molecules, Head (1987) introduced a new operation, called splicing on words or strings, which are finite sequences of symbols. There has been intensive research using the concept of splicing on strings in the context of DNA computing, establishing important theoretical results on computational universality. A particular class of splicing, known as flat splicing on strings was recently considered and this operation was extended to provide picture array generating two-dimensional models. Making use of the operation of flat splicing on arrays, we propose here a grammar system, called flat splicing regular array grammar system (FSRAGS), as a new model of picture generation. The components of a FSRAGS generate picture arrays working in parallel using the rules of a two-phase grammar called 2RLG and with two different components of the FSRAGS communicating using the array flat splicing operations on columns and rows of the arrays. We establish some comparison results bringing out the generative power of FSRAGS and also exhibit the power of FSRAGS in generating certain "floor designs".**

***Keywords***: Flat splicing, Picture array, Picture language, Formal languages, Grammars, Grammar systems

## I. Introduction

In the quest for creating alternatives for the classical silicon-based computing, the field of DNA computing [13, 14], besides others, was born with Adleman solving a small instance of Hamilton path problem [1] based on the use of DNA molecules and operations on DNA sequences. In the study of modelling the DNA recombination process under the action of restriction enzymes and a ligase, Head [8] abstracted this phenomenon in his seminal paper on a formal model of the recombinant behaviour of DNA molecules and defined the operation of splicing of words, which are strings of symbols. This operation of splicing of two words $u = u_1u_2$ and $v = v_1v_2$ involves "cutting" $u$, between $u_1$ and $u_2$ and $v$, between $v_1$ and $v_2$, as dictated by certain rules known as "splicing rules" and "pasting" the prefix $u_1$ of $u$ with the suffix $v_2$ of $v$ yielding a new word $w = u_1v_2$. Subsequently, this opened up several theoretical investigations [6, 10, 11] giving rise to important and deep language-theoretic results in the area of formal language theory, which is widely accepted to be the backbone [5, 12] of theoretical computer science.

Based on the fact that DNA occur in circular form, Head [9, 10] introduced the operation of splicing on circular words. Motivated by a specific form of splicing on circular words, recently, a different form of splicing, called flat splicing on words has been introduced in [2]. The operation of flat splicing on a pair of words $(\alpha, \beta)$ involves "cutting" $\alpha$ and "inserting" $\beta$ into it, as directed by a rule, called flat splicing rule.

Inspired by problems in image processing, several generative models of two-dimensional picture arrays based on language theory, have been proposed and studied (see, for example, [7, 15, 16, 21, 24]). Recently, the operation of flat splicing on words has been extended to picture arrays [23] and an array flat splicing system $(AFS)$ has been defined to describe picture array languages.

On the other hand, the notion of a grammar system consisting of several grammars or other language identifying mechanisms cooperating according to some well-defined protocol, was introduced as a formal framework for modelling distributed complex systems [3]. Based on different motivations, a variety of grammar system models have been proposed and intensively investigated. A different type of grammar system, called Splicing Grammar System was introduced in [4] with the component grammars working in parallel and communication between components being done by the bio-inspired operation of splicing on strings of symbols introduced by Head [8]. This system was extended to arrays in [22] by introducing splicing array grammar system.

Here we make use of the array flat splicing operation and introduce a grammar system called flat splicing regular array grammar system $(FSRAGS)$ and examine the power of this system in generating picture array languages. The components of the $FSRAGS$ involve the rules of the two-

phase grammar $2RLG$ [7, 19] working in parallel and the array flat splicing operation [23] is used for communication among two different components in terms of "insertion" of an array generated in a component into an array generated in a different component. We establish some comparison results bringing out the power of $(FSRAGS)$. We also exhibit the use of $(FSRAGS)$ in generating certain "floor designs". A preliminary version of this paper appears in [18].

## II. Preliminaries

For notions and results related to formal string languages and array languages, we refer to [7, 17].

A linear word or simply, a word $w = a_1a_2\cdots a_n$ is a finite sequence of symbols $a_i$, $1 \leq i \leq n$, taken from a finite alphabet $\Sigma$. The empty word with no symbols, is denoted by $\lambda$. The set of all words over $\Sigma$ is denoted by $\Sigma^*$. The length $|w|$ of a word $w$ is the number of symbols in the word, including repetitions of symbols. For example,the length of the word $abaab$ is 5. For any word $w$, we denote by ${}^tw$ the word $w$ written vertically. For example, if $w = abbab$ over $\{a, b\}$, then

$$ {}^tw = \begin{matrix} a \\ b \\ b \\ a \\ b \end{matrix}. $$

A picture array or simply, an array $M$ of size $m \times n$ over an alphabet $\Sigma$ is a rectangular array with $m$ rows and $n$ columns and is of the form

$$ M = \begin{matrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{matrix} $$

where each symbol $a_{ij} \in \Sigma, 1 \leq i \leq m, 1 \leq j \leq n$. We denote by $|M|_r$ and $|M|_c$, the number of rows and the number of columns of $M$ respectively. The set of all rectangular arrays over $\Sigma$ is denoted by $\Sigma^{**}$, which includes also the empty array $\lambda$ with no symbols. $\Sigma^{++} = \Sigma^{**} - \{\lambda\}$. A picture language is a subset of $\Sigma^{**}$. The column catenation $X \circ Y$ of an array $X$ of size $p \times q$ and an array $Y$ of $r \times s$ is defined only when $p = r$ and likewise the row catenation $X \diamond Y$ is defined only when $q = s$. In other words, in column catenation $X \circ Y$, $Y$ is attached to the right of $X$ while in row catenation $X \diamond Y$, $Y$ is attached below $X$. Also $X \circ \lambda = \lambda \circ X = X \diamond \lambda = \lambda \diamond X = X$, for every array $X$. For example, if

$$ X = \begin{matrix} a & a & b & b \\ a & b & a & b \\ b & a & b & a \end{matrix}, \quad Y = \begin{matrix} a & b & b \\ a & b & a \\ b & a & b \end{matrix}, \quad Z = \begin{matrix} b & b & a \\ a & a & b \end{matrix} $$

then

$$ X \circ Y = \begin{matrix} a & a & b & b & a & b & b \\ a & b & a & b & a & b & a \\ b & a & b & a & b & a & b \end{matrix}, \quad Y \diamond Z = \begin{matrix} a & b & b \\ a & b & a \\ b & a & b \\ b & b & a \\ a & a & b \end{matrix}, $$

An array grammar model of the non-isometric variety generating rectangular arrays of symbols is the two-dimensional right-linear grammar (originally called 2D matrix grammar [19]), which we call here as a two-dimensional right-linear grammar, consistent with the terminology used in [7].

**Definition 1** *A two-dimensional right-linear grammar (2RLG) [7, 19] is*
$G = (V_h, V_v, V_i, T, S, R_h, R_v)$ *where* $V_h, V_v, V_i$ *are finite sets of symbols, respectively called as horizontal nonterminals, vertical nonterminals and intermediate symbols;* $V_i \subset V_v$; $T$ *is a finite set of terminals;* $S \in V_h$ *is the start symbol;* $R_h$ *is a finite set of horizontal rules of the form* $X \to AY, X \to A\ X, Y \in V_h, A \in V_i$; $R_v$ *is a finite set of vertical rules of the form* $X \to aY$ *or* $X \to a$, $X, Y \in V_i, a \in T \cup \lambda$.

There are two phases of derivation in a $2RLG$. In the first horizontal phase, starting with $S$ the horizontal rules are applied (as in a regular grammar) generating strings over intermediates, which act as terminals of this phase. In the second vertical phase each intermediate in such a string serves as the start symbol for the second phase. The vertical rules are applied in parallel in this phase for generating the columns of the rectangular arrays over terminals. In this phase at a derivation step, either all the rules applied are of the form $X \to aY, a \in T$ or all the rules are of the form $X \to Y$ or all the rules are the terminating vertical rules of the form $X \to a, a \in T$ or all of the form $X \to \lambda$. When the vertical generation halts, the array obtained over $T$, is collected in the picture language generated by the 2RLG. Note that the picture language generated by a 2RLG consists of rectangular arrays of symbols. We denote by $L(2RLG)$ the family of array languages generated by two-dimensional right-linear grammars.

We illustrate the working of $2RLG$ with an example.

**Example 1** *Consider the* $2RLG$ $G_L$ *with regular horizontal rules* $S \to AX, X \to BX, X \to B$ *where* $A, B$ *are the intermediate symbols and the vertical rules* $A \to aA, A \to a, B \to bB, B \to a$ *where* $a, b$ *are the terminal symbols. In the first phase, as in a string grammar, the horizontal rules generate words of the form* $AB^n$, $n \geq 1$. *For example, the generation of* $AB^3$ *is as given below:*

$$ S \Rightarrow AX \Rightarrow ABX \Rightarrow ABBX \Rightarrow ABBB = AB^3 $$

*on applying the three horizontal rules one after another, with the rule* $X \to BX$ *applied twice. In the second phase every symbol in such a word* $AB^n$ *is rewritten in parallel either by using the rules* $A \to aA, B \to bB$ *in which case the derivation can be likewise continued adding rows of the form* $ab\cdots b$ *or by using the rules* $A \to a, B \to a$ *in which case the derivation terminates adding a row of the form* $aa\cdots a$, *thus generating picture arrays, one member of which is shown in Fig. 1. In fact, for this picture array, the derivation in the second phase is as follows:*

$$ \begin{matrix} A & B & B & B \end{matrix} \Rightarrow \begin{matrix} a & b & b & b \\ A & B & B & B \end{matrix} \Rightarrow \begin{matrix} a & b & b & b \\ a & b & b & b \\ A & B & B & B \end{matrix} $$

$$\Rightarrow \begin{array}{cccc} \mathtt{a} & \mathtt{b} & \mathtt{b} & \mathtt{b} \\ \mathtt{a} & \mathtt{b} & \mathtt{b} & \mathtt{b} \\ \mathtt{a} & \mathtt{b} & \mathtt{b} & \mathtt{b} \\ A & B & B & B \end{array}$$

*on applying the nonterminal rules $A \to aA, B \to bB$. When the terminal rules $A \to a, B \to bB, B \to a$ are applied together, derivation terminates yielding the array in Fig. 1.*

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

**Figure. 1**: A picture array generated in Example 1

An extension of the $2RLG$, which we call here as two-dimensional tabled right-linear grammar ($2TRLG$) (originally referred to as 2D tabled matrix grammar in [20]) was introduced in [20] to generate picture languages that cannot be generated by any $2RLG$. A $2TRLG$ is defined similar to a $2RLG$ with the following difference: In the second phase, the right-linear rules are grouped into different sets, called tables such that $i$) all the rules in a table are nonterminal rules of the form either $A \to aB$ only or $A \to B$ only or $ii$) all are terminal rules of the form either $A \to a$ only or $A \to \lambda$ only and at a time rules from only one table are used in the second phase. We denote by $L(2TRLG)$ the family of array languages generated by two-dimensional tabled right-linear grammars.

As an illustration, we give an example of a $2TRLG$.

**Example 2** *Consider the $2TRLG$ $G_H$ with regular horizontal rules $S \to AX, X \to BX, X \to BY, Y \to A$ where $A, B$ are the intermediate symbols and the tables of vertical rules are $t_1 = \{A \to aA, B \to bB\}$, $t_2 = \{A \to aC, B \to cD\}$, $t_3 = \{C \to aC, D \to bD\}$, $t_4 = \{C \to a, D \to b\}$ where $a, b, c$ are the terminal symbols. In the first phase, the horizontal rules generate words of the form $AB^nA$, $n \geq 1$. In the second phase every symbol in such a word $AB^nA$ is rewritten in parallel by using the tables of rules. Application of the table $t_1$ certain number of times, followed by $t_2$ once which is then followed by $t_3$ again certain number of times and finally terminating the derivation in this second phase, generates picture arrays, one member of which is shown in Fig. 2.*

$$\begin{array}{cccccc} \mathtt{a} & \mathtt{b} & \mathtt{b} & \mathtt{b} & \mathtt{b} & \mathtt{a} \\ \mathtt{a} & \mathtt{b} & \mathtt{b} & \mathtt{b} & \mathtt{b} & \mathtt{a} \\ \mathtt{a} & \mathtt{b} & \mathtt{b} & \mathtt{b} & \mathtt{b} & \mathtt{a} \\ \mathtt{a} & \mathtt{c} & \mathtt{c} & \mathtt{c} & \mathtt{c} & \mathtt{a} \\ \mathtt{a} & \mathtt{b} & \mathtt{b} & \mathtt{b} & \mathtt{b} & \mathtt{a} \\ \mathtt{a} & \mathtt{b} & \mathtt{b} & \mathtt{b} & \mathtt{b} & \mathtt{a} \end{array}$$

**Figure. 2**: A picture array generated in Example 2

An operation, called flat splicing on linear words, is considered by Berstel et al. [2]. A flat splicing rule $r$ is of the form $(\alpha|\gamma - \delta|\beta)$, where $\alpha, \beta, \gamma, \delta$ are words over a given alphabet $\Sigma$. For two words $u = x\alpha\beta y$, $v = \gamma z\delta$, an application of the flat splicing rule $r = (\alpha|\gamma - \delta|\beta)$ to the pair $(u, v)$ yields the word $w = x\alpha\gamma z\delta\beta y$. In other words, the second word $v$ is inserted between $\alpha$ and $\beta$ in the first word $u$ as a result of applying the rule $r$.

The notion of flat splicing on words [2] has been extended to arrays in [23], by introducing two kinds of flat splicing rules, called column flat splicing rule and row flat splicing rule and thus a new model of picture generation, called array flat splicing system is introduced in [23].

**Definition 2** *[23] Let $V$ be an alphabet.*

*(i) A column flat splicing rule is of the form $({}^t(a_1a_2)|^t(x_1x_2) - {}^t(y_1y_2)|^t(b_1b_2))$ where $a_1, a_2, b_1, b_2 \in \Sigma \cup \{\lambda\}$ with $|a_1| = |a_2|$ and $|b_1| = |b_2|$, $x_1, x_2, y_1, y_2 \in \Sigma \cup \{\lambda\}$ with $|x_1| = |x_2|$ and $|y_1| = |y_2|$.*

*(ii) A row flat splicing rule is of the form $(c_1c_2|u_1u_2 - v_1v_2|d_1d_2)$ where $c_1, c_2, d_1, d_2 \in \Sigma \cup \{\lambda\}$ with $|c_1| = |c_2|$ and $|d_1| = |d_2|$, $u_1, u_2, v_1, v_2 \in \Sigma \cup \{\lambda\}$ with $|u_1| = |u_2|$ and $|v_1| = |v_2|$.*

*(iii) Let $r_1, r_2, \cdots, r_{m-1}$ be a sequence of $(m-1)$ column flat splicing rules given by*

$$r_i = ({}^t(\alpha_i\alpha_{i+1})|^t(\gamma_i\gamma_{i+1}) - {}^t(\delta_i\delta_{i+1})|^t(\beta_i\beta_{i+1})),$$

*for $1 \leq i \leq (m-1)$. Let $X, Y$ be two picture arrays, each with $m$ rows, for some $m \geq 1$, and given by*

$$X = X_1 \circ {}^t(\alpha_1\alpha_2 \cdots \alpha_m) \circ {}^t(\beta_1\beta_2 \cdots \beta_m) \circ X_2,$$

$$Y = {}^t(\gamma_1\gamma_2 \cdots \gamma_m) \circ Y' \circ {}^t(\delta_1\delta_2 \cdots \delta_m),$$

*where $X_1, X_2, Y'$ are arrays over $\Sigma$ with $m$ rows, $\alpha_i, \beta_i, \in \Sigma \cup \{\lambda\}$ $(1 \leq i \leq m)$, with $|\alpha_1| = |\alpha_2| = \cdots = |\alpha_m|$, $|\beta_1| = |\beta_2| = \cdots = |\beta_m|$, $\gamma_i, \delta_i$, $(1 \leq i \leq m)$, $\in \Sigma \cup \{\lambda\}$ with $|\gamma_1| = |\gamma_2| = \cdots = |\gamma_m|$, $|\delta_1| = |\delta_2| = \cdots = |\delta_m|$. An application of the column flat splicing rules $r_1, r_2, \cdots, r_{m-1}$ to the pair of arrays $(X, Y)$ yields the array $Z$*

$$= X_1 \circ {}^t(\alpha_1\alpha_2 \cdots \alpha_m) \circ {}^t(\gamma_1\gamma_2 \cdots \gamma_m) \circ$$

$$Y' \circ {}^t(\delta_1\delta_2 \cdots \delta_m) \circ {}^t(\beta_1\beta_2 \cdots \beta_m) \circ X_2.$$

*The pair $(X, Y)$ yielding $Z$ is then denoted by $(X, Y) \vdash_c Z$.*

*(iv) Let $s_1, s_2, \cdots, s_{n-1}$ be a sequence of $(n-1)$ row flat splicing rules given by*

$$s_j = (\eta_j\eta_{j+1}|(\mu_j\mu_{j+1}) - (\nu_j\nu_{j+1})|\theta_j\theta_{j+1}),$$

*for $1 \leq j \leq (n-1)$. Let $U, V$ be two picture arrays, each with $n$ columns, for some $n \geq 1$, and given by*

$$U = U_1 \diamond (\eta_1\eta_2 \cdots \eta_n) \diamond (\theta_1\theta_2 \cdots \theta_n) \diamond U_2,$$

$$V = (\mu_1\mu_2 \cdots \mu_n) \diamond V' \diamond (\delta_1\delta_2 \cdots \delta_n)$$

*where $U_1, U_2, V'$ are arrays over $\Sigma$ with $n$ columns, $\eta_j, \theta_j$, $(1 \leq j \leq n), \in \Sigma \cup \{\lambda\}$ with $|\eta_1| = |\eta_2| = \cdots = |\eta_n|$, $|\theta_1| = |\theta_2| = \cdots = |\theta_n|$, $\mu_j, \nu_j$, $(1 \leq j \leq n), \in \Sigma \cup \{\lambda\}$ with $|\mu_1| = |\mu_2| = \cdots = |\mu_n|$, $|\nu_1| = |\nu_2| = \cdots = |\nu_n|$. An application of the row flat splicing rules $s_1, s_2, \cdots, s_{n-1}$ to the pair of arrays $(U, V)$ yields the array W*

$$= U_1 \diamond (\eta_1\eta_2 \cdots \eta_n) \diamond (\mu_1\mu_2 \cdots \mu_n) \diamond V'$$

$$\diamond(\delta_1\delta_2 \cdots \delta_n) \diamond (\theta_1\theta_2 \cdots \theta_n) \diamond U_2.$$

*The pair $(U, V)$ yielding $W$ is then denoted by $(U, V) \vdash_r W$.*

*(v) An array flat splicing rule is either a column flat splicing rule or a row flat splicing rule. The notation $\vdash$ denotes either $\vdash_c$ or $\vdash_r$.*

*(vi) For a picture language $L \subseteq \Sigma^{**}$ and a set $R$ of array flat splicing rules, we define*

$$f(L) = \{M \in \Sigma^{**} \mid (X, Y) \vdash M, for X, Y \in L,$$

$$and\ some\ rule\ in\ R\}.$$

*(vii) An array flat splicing system $(AFS)$ is $\mathcal{A} = (\Sigma, M, R_c, R_r)$ where $\Sigma$ is an alphabet, $M$ is a finite set of arrays over $\Sigma$, called initial set, $R_c$ is a finite set of column flat splicing rules and $R_r$ is a finite set of row flat splicing rules. The picture language $L(\mathcal{A})$ generated by $\mathcal{A}$ is iteratively defined as follows:*

$$f^0(M) = M;\ For\ i \geq 0,$$

$$f^{i+1}(M) = f^i(M) \cup f(f^i(M));$$

$$L(\mathcal{A}) = f^*(M) = \cup_{i \geq 0} f^i(M).$$

*The family of picture languages generated by array flat splicing systems is denoted by $L(AFS)$.*

An illustration of the application of column and row flat splicing rules as well as the working of an array flat splicing system is given in the following example.

**Example 3** *Consider the array flat splicing system $\mathcal{A}_R$ with alphabet $\{a, b\}$ and the initial set consisting of the array $M = \{ \begin{matrix} a & b \\ a & b \end{matrix} \}$. The column flat splicing rule is $c$ where $c = ( \begin{matrix} a \\ a \end{matrix} \mid \begin{matrix} a \\ a \end{matrix} - \begin{matrix} b \\ b \end{matrix} \mid \begin{matrix} b \\ b \end{matrix} )$. The row flat splicing rules are $r_1, r_2, r_3$ where $r_1 = (aa|aa - aa|aa)$, $r_2 = (ab|ab - ab|ab), r_3 = (bb|bb - bb|bb)$.*

*The column flat splicing rule $c$ is applicable to the pair $(M, M)$. Note that both the components in the pair being the same initial array $M$, the requirement of equal number of rows for the application of column flat splicing rule $c$ is satisfied. Also the second array in the pair begins with the column $\begin{matrix} a \\ a \end{matrix}$ and ends with the column $\begin{matrix} b \\ b \end{matrix}$, as required in the rule $c$. The first array is "cut" between the columns $\begin{matrix} a \\ a \end{matrix}$ and $\begin{matrix} b \\ b \end{matrix}$ while the second array is "inserted" between them, yielding the array $\begin{matrix} a & a & b & b \\ a & a & b & b \end{matrix}$. On the other hand, the row flat splicing rules $r_1, r_2$ can be used to expand the array rowwise. For example, the sequence of rules $r_1, r_2, r_3$ could be applied to the pair of arrays with both components having the same array $\begin{matrix} a & a & b & b \\ a & a & b & b \end{matrix}$. Again note that both the components in the pair being the same array, the requirement of equal number of columns for the application of a sequence of row flat splicing rules, is satisfied. In fact, the first array is "cut" between the first row $a\ \ a\ \ b\ \ b$ and the second row which is also $a\ \ a\ \ b\ \ b$. The second array satisfies the requirements of the sequence of rules $r_1, r_2, r_3$. The second array therefore can be "inserted" into the first array to yield the array $\begin{matrix} a & a & b & b \\ a & a & b & b \\ a & a & b & b \\ a & a & b & b \end{matrix}$. Proceeding like this, we compute the successive terms $f^0(M), f^1(M), \cdots$. In fact*

$$f^0(M) = M = \{ \begin{matrix} a & b \\ a & b \end{matrix} \},$$

$$f^1(M) = M \cup f(M) = \{ \begin{matrix} a & b \\ a & b \end{matrix}, \begin{matrix} a & a & b & b \\ a & a & b & b \end{matrix}, \begin{matrix} a & b \\ a & b \\ a & b \\ a & b \end{matrix}, \},$$

$$\cdots .$$

*Thus the picture language $L(\mathcal{A}_R) = f^*(M)$ consists of rectangular arrays of even side length over the symbols $a, b$. Also if the number of columns in such an array is $2n$, then the first $n$ columns are over $a$ while the next $n$ columns are over $b$.*

```
a a a b b b
a a a b b b
a a a b b b
a a a b b b
```

**Figure. 3**: A member of the language of $\mathcal{A}_R$

One such picture array is shown in Fig. 3.

## III. Flat Splicing Regular Array Grammar System

We now introduce a new model of picture generation, called flat splicing array gramnmar system.

**Definition 3** *A flat splicing regular array grammar system $(FSAGS)$ is a construct $G_{as} = (V_h, V_v, V_i, T, (S_1, R_1^h, R_1^v), ..., (S_n, R_n^h, R_n^v), F)$ where $V_h, V_v, V_i$ are respectively, the finite sets of horizontal, vertical and intermediate nonterminals; $V_i \subseteq V_v$; $T$ is the terminal alphabet; $S_i, 1 \leq i \leq n$ is the start symbol of the corresponding component; $R_i^h, 1 \leq i \leq n$ is a finite set of horizontal rules, which are regular of the forms $X \to AY, X \to A$, $X, Y \in V_h, A \in V_i$; $R_i^v, 1 \leq i \leq n$ is a finite set of right-linear vertical rules of the forms $A \to aB, A \to B, A \to a$, $A, B \in V_v, a \in T$; $F$ is a finite set of array flat splicing rules The derivations take place in*

*two phases as follows :*

*Each component grammar generates a word called intermediate word, over intermediates starting from its own start symbol and using its horizontal rules ; the derivations in this phase are done with the component grammars working in parallel.*

*In the second phase any of the following steps can take place:*

*(i) each component grammar can rewrite as in a two dimensional matrix grammar using the vertical rules, starting from its own intermediate word generated in the first phase. (The component grammars rewrite in parallel and the rules are applied together). Note that the component grammars together terminate with all rules used in the form $A \to a$ or together continue rewriting in the vertical direction with all rules used either in the form $A \to aB$ or in the form $A \to B$.*

*(ii) At any instant the picture array $X$ generated in the $i^{th}$ component for some $i$, $1 \leq i \leq n$ and the picture array $Y$ generated in the $j^{th}$ component for some $j$, $1 \leq j \leq n$ can be flat spliced using array flat splicing rules, either column or row flat splicing rules as in definition 2, thus yielding picture array $Z$ in $i^{th}$ component. In any other component (other than $i^{th}$ component), the arrays generated at this instant will remain unchanged during this flat splicing process.*

*There is no priority between steps (i) and (ii).*

*The language $L_i(G_{as})$ generated by the $ith$ component of $G_{as}$ consists of all picture arrays, generated over $T$, by the derivations described above. This language will be called the individual picture array language of the system and we may choose this to be the language of the first component. The family of individual picture array languages generated by $FSRAGS$ with at most $n$ components is denoted by $L_n(FSRAGS)$.*
We illustrate the working of $FSRAGS$ with an example.

**Example 4** *Consider the $FSAGS$ $G_1$ with two components given by*

$$(\{S_1, S_2, X, Y\}, \{A, C, D\}, \{A, C, D\}, \{a, c, d\},$$

$$(S_1, R_1^h, R_1^v), (S_2, R_2^h, R_2^v), F)$$

*where*

$$R_1^h = \{S_1 \to AX, X \to CX, X \to A, \},$$

$$R_2^h = \{S_2 \to Z, Z \to DZ, Z \to D\},$$

$$R_1^v = \{A \to aA, C \to cC, A \to a, C \to c\},$$

$$R_2^v = \{A \to aA, D \to dD, A \to a, D \to d\}$$

*and $F$ consists of the column flat splicing rule*

$$c_1 = ( \begin{matrix} a \\ a \end{matrix} \mid \begin{matrix} d \\ d \end{matrix} - \begin{matrix} d \\ d \end{matrix} \mid \begin{matrix} c \\ c \end{matrix} ).$$

```
a c c c a      d d d d
a c c c a      d d d d
a c c c a      d d d d
a c c c a      d d d d
```

$(a)$ $(b)$

**Figure. 4**: (a) Picture array generated in the first component of $G_1$ (b) Picture array generated in the second component of $G_1$

```
a d d d d c c c a
a d d d d c c c a
a d d d d c c c a
a d d d d c c c a
```

**Figure. 5**: A picture array generated by $G_1$

Starting from $S_1$, the horizontal regular rules in the first component generate a word of the form $AC^{n-1}A$ on applying the rule $S_1 \to AX$ once followed by the application of the rule $X \to CX$ $(n-1)$ times and finally the rule $X \to A$ once, terminating the derivation in the first component in the horizontal phase. At the same time in the second component derivations in the horizontal phase take place in parallel starting from $S_2$ and applying the rule $S_2 \to Z$ once followed by the application of the rule $Z \to DZ$ $(n-1)$ times and finally the rule $Z \to D$ once, yielding the word $D^n$ for the same $n$.

In the second phase vertical derivations in both the components take place in parallel. In the first component an $m \times (n+1)$ picture array as in Fig. 4(a) is generated while in the second component an $m \times n$ picture array as in Fig. 4(b) is generated. At this point, using the column flat splicing rule $c_1$ as many times as needed, the array in Fig. 4(b) is inserted in the array in Fig. 4(a) between the first column of $a's$ and the second column of $c's$ to yield a picture array of the form as in Fig. 5.

**Theorem 1**

$$L_2(FSRAGS) \setminus L(AFS) \neq \emptyset.$$

**Proof 1** *The picture array language $L$ generated by the $FSRAGS$ $G_1$ in Example 4 consists of picture arrays $M$ each of which has the following property $P$ : $M$ has exactly two columns of $a's$ with one, the leftmost column and the other, the rightmost column, besides other columns made of either $c's$ or $d's$. But $L$ cannot be generated by any $AFS$ since the flat splicing rules will "insert" one array of $L$ into another, thereby yielding picture arrays which will violate the property mentioned above.*

**Theorem 2**

$$L(2RLG) = L_1(FSRAGS) \subset L_2(FSRAGS).$$

**Proof 2** *The equality $L(2RLG) = L_1(FSRAGS)$ follows by noting that the $FSRAGS$ with one component will have rewriting rules as in a $2RLG$ and the set of array flat splicing rules can be taken to be an empty set as these rules*

*can be applied only when there are at least two components. The picture array language $L$ of Example 4, is generated by a $FSRAGS$ with two components. An array of $L$ has the property that there are $(n+1)$ consecutive columns of $d's$ (starting from the second column) followed by $n$ columns of $c's$. But this property which involves a proportion in the number of columns of $c's$ and the number of columns of $d's$ (analogous to the context-free string language $\{d^{(n+1)}c^n | n \geq 1\}$) cannot be maintained by regular horizontal rules of a $2RLG$ and hence $L$ cannot be generated by any $2RLG$. Since $L(2RLG) = L_1(FSRAGS)$ it follows that any $FSRAGS$ with only one component cannot generate the picture array language $L$. This proves the proper inclusion $L_1(FSRAGS) \subset L_2(FSRAGS)$.*

**Theorem 3** $L_2(FSRAGS) \setminus L(2TRLG) \neq \emptyset$

**Proof 3** *As noted in the proof of Theorem 2, the picture array language $L$ of Example 4, which is generated by a $FSRAGS$ with two components, has the feature that an array of $L$ involves a proportion in the number of columns of $c's$ and the number of columns of $d's$. This property cannot be handled by any $2TRLG$ as the rules in the first phase of a $2TRLG$ are only regular rules. Thus $L$ cannot be generated by any $2TRLG$.*

In [20], analogous to $2TRLG$, a two-dimensional grammar, here called two-dimensional tabled context-free grammar $2TCFG$ (originally called 2D tabled CF matrix grammar) was considered in [20] and this 2D grammar has context-free grammar kind of rules in the first phase while the second phase has tables of right-linear rules as in a $2TRLG$. Derivations are done as in a $2TRLG$. We denote by $L(2TCFG)$, the family of picture array languages generated by $2TCFG$. Also, when there are exactly two tables in the second phase of a $2TCFG$, with one table consisting of all right-linear rules of the form $A \to aB$ and another table consisting of all terminal rules of the form $A \to a$, where $A, B$ are nonterminals and $a$ is a terminal symbol, the $2TRLG$ is indeed a $2RLG$. We now show that the family $L_3(FSRAGS)$ of picture languages generated by flat splicing regular array grammar systems with three components,contains picture languages that cannot be generated by any $2TCFG$.

**Theorem 4** $L_3(FSRAGS) \setminus L(2TCFG) \neq \emptyset$

**Proof 4** *Consider the picture array language $L_2$ whose elements are picture arrays with two rows of the forms*

$$M_1 = \begin{matrix} a & \cdots & a & b & \cdots & b & e & d \\ a & \cdots & a & b & \cdots & b & e & d \end{matrix}$$

*or*

$$M_2 = \begin{matrix} a & \cdots & a & b & \cdots & b & c & \cdots & c & e & d \\ a & \cdots & a & b & \cdots & b & c & \cdots & c & e & d \end{matrix}$$

*with the columns of $a's$ being equal in number to the columns of $b's$ in the former array while this number also equals the number of columns of $c's$ in the latter array. This kind of picture language cannot be generated by any $2TCFG$ as the feature present in the arrays of the form $M_2$, namely, equal number of columns of $a's$, $b's$ and $c's$ is analogous to the context-sensitive string language $\{a^n b^n c^n | n \geq 1\}$ cannot be handled by the context-free kind of rules in the first phase of a $2TCFG$.*

*But the language $L_2$ is generated by the following $FSRAGS$ $G_2$ with three components: Define*

$$G_2 = (V_h, V_v, V_i, T, (S_1, R_1^h, R_1^v),$$

$$(S_2, R_2^h, R_2^v), (S_3, R_3^h, R_3^v), F)$$

*where*

$$T = \{a, b, c, d, e, f\}, V_h = \{S_1, S_2, S_3, X, Y, Z\},$$

$$V_i = \{A, D, B, E, C, F\},$$

$$V_v = \{A, D, B, E, C, F, A', D', B', E', C', F',$$

$$A'', D'', B'', E'', C'', F''\}.$$

$$R_1^h = \{S_1 \to AS_1, S_1 \to AX, X \to D\},$$

$$R_2^h = \{S_2 \to BS_2, S_2 \to BY, Y \to E\},$$

$$R_3^h = \{S_3 \to CS_3, S_3 \to CZ, Z \to F\},$$

$$R_1^v = \{A \to \begin{matrix} a \\ A' \end{matrix}, D \to \begin{matrix} d \\ D' \end{matrix}, A' \to A'', B' \to B'',$$

$$D' \to D'', E' \to E'', A'' \to a, B'' \to b, D'' \to d,$$

$$E'' \to e, C'' \to c, F'' \to f\}$$

$$R_2^v = \{B \to \begin{matrix} b \\ B' \end{matrix}, E \to \begin{matrix} e \\ E' \end{matrix},$$

$$B' \to B'', E' \to E'', B'' \to b, E'' \to e\}$$

$$R_3^v = \{C \to \begin{matrix} c \\ C' \end{matrix}, F \to \begin{matrix} f \\ F' \end{matrix},$$

$$C' \to C'', F' \to F'', C'' \to c, F'' \to f\}$$

*$F$ consists of the following column flat splicing rules:*

$$(\ \begin{matrix} a \\ A' \end{matrix} \mid \begin{matrix} b \\ B' \end{matrix} - \begin{matrix} e \\ E' \end{matrix} \mid \begin{matrix} d \\ D' \end{matrix}\ ), (\ \begin{matrix} b \\ B'' \end{matrix} \mid \begin{matrix} c \\ C'' \end{matrix} - \begin{matrix} f \\ F'' \end{matrix} \mid \begin{matrix} e \\ E'' \end{matrix}\ )$$

*Derivations in the three components take place in parallel with the application of the horizontal rules $S_1 \to AS_1, S_1 \to AX, X \to D$ in the first component yielding the word $A^n D$ for some $n \geq 1$, while, likewise, in the second and third components, the horizontal rules yield respectively the words $B^n E$ and $C^n F$. At a time all three words derived involve the same $n$. If the vertical rules $A \to \begin{matrix} a \\ A' \end{matrix}, D \to \begin{matrix} d \\ D' \end{matrix}$ in the first component are applied, an array of the form $\begin{matrix} a & \cdots & a & d \\ A' & \cdots & A' & D' \end{matrix}$ is generated in the first component while, likewise, application of similar rules in the second and third components genrate the arrays respectively of the forms $\begin{matrix} b & \cdots & b & e \\ B' & \cdots & B' & E' \end{matrix}$ and $\begin{matrix} c & \cdots & c & f \\ C' & \cdots & C' & F' \end{matrix}$. If at this stage, the column flat splicing rule $(\ \begin{matrix} a \\ A' \end{matrix} \mid \begin{matrix} b \\ B' \end{matrix} - \begin{matrix} e \\ E' \end{matrix} \mid \begin{matrix} d \\ D' \end{matrix}\ )$ is applied, then the array obtained is of the form $\begin{matrix} a & \cdots & a & b & \cdots & b & e & d \\ A' & \cdots & A' & B' & \cdots & B' & E' & D' \end{matrix}$ If now the vertical rules such as $A' \to A''$ that change the primed symbols*

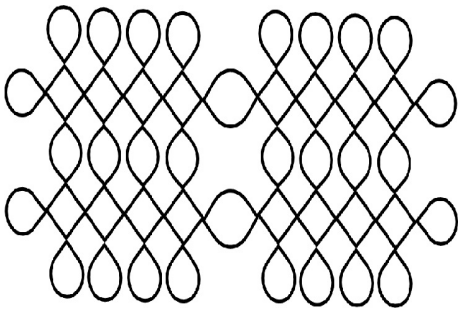

**Figure. 6**: A floor design

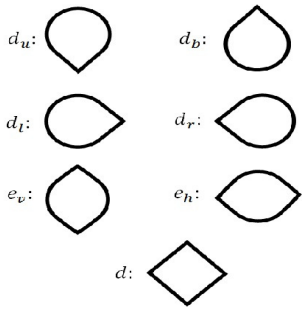


**Figure. 7**: Primitive Patterns

*into double primed symbols are applied followed by the application of terminal rules, then picture arrays of the form*

$$\begin{array}{ccccccccc} a & \cdots & a & b & \cdots & b & e & d \\ a & & a & b & & b & e & d \end{array}$$

*are generated which are collected in the picture language of* $G_2$. *On the otherhand, prior to the application of the terminal rules, if the column flat splicing rule* $( \begin{smallmatrix} b \\ B'' \end{smallmatrix} \mid \begin{smallmatrix} c \\ C'' \end{smallmatrix} - \begin{smallmatrix} f \\ F'' \end{smallmatrix} \mid \begin{smallmatrix} e \\ E'' \end{smallmatrix} )$ *is applied and the double primed symbols are changed into terminals by the application of terminal rules, then arrays of the form*

$$\begin{array}{ccccccccccc} a & \cdots & a & b & \cdots & b & c & \cdots & c & f & e & d \\ a & & a & b & & b & c & & c & f & e & d \end{array}$$

*are generated and are also collected in the picture language of* $G_2$.

## IV. Application to Generation of Floor Designs

Generation of certain picture patterns, such as "floor designs" (see, for example, Fig. 6), using array generating grammar models has been done [19]. The idea is to consider the picture pattern as an array over certain terminal symbols and generate the array with an array grammar. Then substitute for each symbol some relevant primitive pattern of the picture to be generated, yielding the pattern. Here we explain how to construct a $FSRAGS$ with two components to generate the collection of floor designs, one member of which is shown in Fig. 6. The primitive patterns involved in this floor design are shown in Fig. 7. The picture array encoding the floor design is given in Fig. 8. The picture array language consisting of picture arrays such as the one given in Fig. 8, along with picture arrays as the one given

$$\begin{array}{ccccccccccc} b & d_u & d_u & d_u & d_u & b & d_u & d_u & d_u & d_u & b \\ d_l & d & d & d & d & e_h & d & d & d & d & d_r \\ b & e_u & e_u & e_u & e_u & b & e_u & e_u & e_u & e_u & b \\ d_l & d & d & d & d & e_h & d & d & d & d & d_r \\ b & d_b & d_b & d_b & d_b & b & d_b & d_b & d_b & d_b & b \end{array}$$

**Figure. 8**: A picture array generated by $G_F$ using array flat splicing

$$\begin{array}{cccccc} b & d_u & d_u & d_u & d_u & b \\ d_l & d & d & d & d & e_h \\ b & e_u & e_u & e_u & e_u & b \\ d_l & d & d & d & d & e_h \\ b & d_b & d_b & d_b & d_b & b \end{array}$$

**Figure. 9**: A picture array generated by $G_F$

in Fig. 9, can be generated by a $FSRAGS$ $G_F$ with two components. We do not give the formal details of $G_F$ but informally describe the idea.

The idea is to allow the first component generate the left half of the picture array (with the rightmost column made of only $e_h$) while the second component generates the second half and array flat splicing rules are created to concatenate the "right half" to the right of the "left half", yielding the picture array of the form in Fig. 8. When the relevant primitive patterns are substituted for the corresponding symbols, this yields the floor design patterns as in Fig. 6.

## V. Conclusion and Discussion

We have considered here a grammar system that involves regular rules as defined in the two-phase $2RLG$ and array flat splicing rules introduced in [23]. One could also consider in the flat splicing array grammar system, CF horizontal rules as in the two-phase two-dimensional grammar called context-free matrix grammar considered in [19], and examine the generative power of the resulting system, which might help describe more complex floor designs. In fact it will be of interest to explore the theoretical model proposed here for other possible applications in terms of experimental studies.

## Acknowledgments

The authors are grateful to the reviewers for their very useful comments.

## References

[1] Adleman, L.M.: Molecular Computation of Solutions to Combinatorial Problems, Science New Series, 266(5187) (1994) 1021-1024.

[2] Berstel, J., Boasson, L., Fagnot, I. : Splicing systems and the Chomsky hierarchy. Theoret. Comput. Sci. 436, 2 -22 (2012).

[3] Csuhaj-Varjú, E., Dassow, J., Kelemen, J., Păun, Gh.: Grammar systems: A grammatical approach to distri-

bution and cooperation. Gordon and Breach Science Publishers, 1994.

[4] Dassow, J., Mitrana, V.: Splicing grammar systems. Computers and Artificial Intell. 15, 109-122 (1996).

[5] Esik, Z., Martin-Vide, C., Mitrana, V. (Eds.): Recent Advances in Formal Languages and Applications, Studies in Computatioinal Intelligence, Springer-Verlag, 2006.

[6] Freund, R., Kari, L., Păun, Gh.,: DNA Computing Based on Splicing: The Existence of Universal Computers. Theory Comput. Syst. 32(1): 69-112 (1999)

[7] Giammarresi, D., Restivo, A.: Two-dimensional languages. In: [7], Vol. 3, pp. 215-267 (1997).

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

[9] Head, T., Circular suggestions for DNA Computing, In: Pattern Formation in Biology, Vision and Dynamics(Ed. Carbone, A. et al.), World Scientific, (2000) 325-335.

[10] Head, T., Păun, Gh., Pixton, D. : Language theory and molecular genetics: generative mechanisms suggested by DNA recombination, In: [7], Vol. 2, pp. 295-358, 1997.

[11] Head, T., Pixton, D.: Splicing and Regularity. Recent Advances in Formal Languages and Applications 2006: 119-147

[12] Martin-Vide, C., Mitrana, V., Păun, Gh. (Eds.): Formal Languages and Applications. Springer-Verlag, 2004

[13] Păun, Gh.: DNA computing based on splicing: universality results. Theor. Comput. Sci. 231(2): 275-296 (2000)

[14] Păun, Gh., Rozenberg, G., Salomaa, A. : DNA Computing - New Computing Paradigms. Texts in Theoretical Computer Science. An EATCS Series, Springer 1998.

[15] A. Rosenfeld, Picture Languages. Academic Press, Reading, MA, 1979.

[16] A. Rosenfeld, R. Siromoney, Picture languages – a survey, Languages of Design, 1 (1993) 229-245.

[17] Rozenberg, G., Salomaa, A. (Eds.): Handbook of Formal Languages Vol. 1–3, Springer, Berlin, (1997).

[18] Subramanian, K.G., Samdanielthompson, G., David, N.G., Nagar, A.K. : Generating Picture Arrays Based on Grammar Systems with Flat Splicing Operation,In: V. Snel et al. (eds.), Innovations in Bio-Inspired Computing and Applications, Advances in Intelligent Systems and Computing 424,Springer International Publishing Switzerland 2016, 251-261.

[19] Siromoney, G., Siromoney, R., Krithivasan, K. : Abstract families of matrices and picture languages. Comput. Graph. Imag. Process. 1, 284-307 (1972).

[20] Siromoney, R., Subramanian, K.G., Rangarajan, K. : Parallel/Sequential rectangular arrays with tables. Int. J. Comput. Math., 6A, 143-158 (1977)

[21] Subramanian, K.G., Ali, R.M., Geethalakshmi, M., Nagar, A.K. : Pure 2D picture grammars and languages. Discrete Appl. Math. 157 (2009) 3401-3411.

[22] Subramanian, K.G., Roslin Sagaya Mary, A., Dersanambika, K.S. : Splicing Array Grammar Systems, Proc. International Colloquium Theoretical Aspects of Computing 2005, Lecture Notes in Computer Science 3722, Springer 125-135 (2005).

[23] Subramanian, K.G., Nagar, A.K., Pan, L. : A Picture Array Generating Model Based on Flat Splicing Operation, Proc. of the 10th International Conference on Bio-inspired Computing:Theories and Applications (BICTA), (2015).

[24] Subramanian, K.G., Rangarajan, K., Mukund. M. (Eds.): Formal Models, Languages and Applications. Series in Machine Perception and Artificial Intell. Vol. 66, World Scientific Publishing, 2006.

# Author Biographies

**G. Samdaielthompson** was born on March 13, 1990. He is pursuing his PhD programme from January 2015 in the Department of Mathematics, Madras Christian College, India with Prof. N. Gnanamalar David as his supervisor. He received his M.Phil degree in the year 2014. His research interests include DNA computing, picture languages and graph languages.

**N. Gnanamalar David** was born on December 22, 1958. He has been on the faculty of the Department of Mathematics, Madras Christian College, India since 1980 and is currently the Head of this department. He was the Dean of Sciences, Madras Christian College during the years 2012-2014. He received his Ph.D. degree from the University of Madras in 2002. He has published a number of research papers in reputed journals. His research areas include graph grammars, picture languages, DNA computing and astronomy. His academic visits include University of Bremen, Germany and other institutions in France, Netherlands and Greece.

**Atulya Nagar** was born on January 25, 1967. He is the Dean, Faculty of science, Liverpool Hope University, UK and also holds the Foundation Chair as Professor of Mathematical Sciences in this university. He received his Doctorate (DPhil) in 1996 from the University of York. Prior to joining Liverpool Hope, he was with the Department of Mathematical Sciences, and later at the Department of Systems Engineering, at Brunel University, London. Earlier he has lectured at the Tata Institute of Fundamental Research (TIFR), Bangalore, BITS Pilani and the Indian Institute of Technology (IITs) in India.

His research is multi-disciplinary with expertise in Nonlinear Mathematics, Natural Computing, Bio-Mathematics and Computational Biology, Operations Research, and Control Systems Engineering. He has edited volumes on Intelligent Systems, and Applied Mathematics. He is the Editor-in-Chief of the International Journal of Artificial Intelligence and Soft Computing (IJAISC) and serves on editorial boards for a number of prestigious journals such as the Journal of Universal Computer science. He has more than 200 publications in prestigious publishing outlets and journals such as the Journal of Applied Mathematics and Stochastic Analysis, the International Journal of Advances in Engineering Sciences and Applied Mathematics, the International Journal of Foundations of Computer Science, the IEEE Transactions on Systems, Man, and Cybernetics, Discrete Applied Mathematics, Fundamenta Informaticae, IET Control Theory and Applications and others.

**K.G. Subramanian** was born on July 6, 1948. He received his PhD in 1980 from the University of Madras, India. He was on the faculty of the Department of Mathematics, Madras Christian College, Chennai, India from 1970 to 2006. He has served as supervisor for a number of PhD scholars of University of Madras. His main area of research falls under the broad topic of Mathematical aspects of Computer Science and his research interests include Combinatorics on words, applications of the theory of formal languages to picture generation, learning and cryptography and biologically motivated computing models. He has visited many institutions in France, Spain, Germany, Vietnam, China, Malaysia, Japan, UK on different periods for collaborative research. He has many publications in reputed journals. He was a visiting Professor at the School of Mathematical Sciences (2007-2010) and the School of Computer Sciences (2011-2015) at the Universiti Sains Malaysia, Malaysia. Currently, he has a visiting professor position at Liverpool Hope University, UK.