Bulletin of Pure and Applied Sciences
Section - E - Mathematics & Statistics
Website : https : //www.bpasjournals.com/

Bull. Pure Appl. Sci. Sect. E Math. Stat.
38E(Special Issue)(2S), 24–29 (2019)
e-ISSN:2320-3226, Print ISSN:0970-6577
DOI: 10.5958/2320-3226.2019.00074.2
©Dr. A.K. Sharma, BPAS PUBLICATIONS,
387-RPS- DDA Flat, Mansarover Park,
Shahdara, Delhi-110032, India. 2019

Splicing regular matrix grammars using parallel communicating grammar systems*

M. Iffath Mubeen[1] and J.D.Emerald[2]

1. Department of Mathematics, Government Arts College for Men,
Affiliated to University of Madras, Chennai, India.
2. Department of Mathematics, Queen Mary's College,
Affiliated to University of Madras, Chennai, India.
1. E-mail: iffathmalick@hotmail.com

Abstract Extensive research has been done on splicing of words. Further, splicing on words has been extended to arrays in Samdanielthompson et al. (Samdanielthompson, G., David, N. Gnanamalar, Nagar, Atulya K. and Subramanian, K.G., Flat splicing array grammar systems generating picture arrays, International Journal of Computer Information System and Industrial Management Applications, (2016) Vol 8, 336–344). In this context, we propose, a grammar system, using queries to splice regular matrix grammars and show that the language generated by this grammar system is incomparable to the language given in Subramanian et al. (Subramanian, K.G., Mary, A. Roslin Sagaya and Dersanambika, K.S., Splicing array grammar systems, Proceedings of the Second International conference on Theoretical Aspects of Computing ICTAC, (2005), 125–135.) and has more generative power than in Samdanielthompson et al. (op. cit.).

Key words Grammar systems, parallel communicating grammar system, query symbols.

2010 Mathematics Subject Classification 03D05, 68Q42, 68Q45 68Q70.

1 Introduction

Matrix grammars were introduced by Siromoney et al. [4] and Rosenfeld and Siromoney [5] as generative models to generate two-dimensional picture arrays which can be used in image processing. Grammar systems which are a model of distributed computation were developed in the field of computer science which can be used in computer networks.

A grammar system contains a finite set of grammars that work under certain rules to generate a language. Sequential and parallel are the two types of grammar systems that are defined in Dassow et al. [1]. A parallel communicating grammar system is a construct Ω, that contains a finite set of non-terminals, terminals, query symbols and n number of axioms and components. All components

* Communicated, edited and typeset in Latex by *Lalit Mohan Upadhyaya* (Editor-in-Chief).
Received March 16, 2019 / Revised August 27, 2019 / Accepted September 19, 2019. Online First Published on June 03, 2020 at https://www.bpasjournals.com/.
Corresponding author M. Iffath Mubeen, E-mail: iffatthmalick@hotmail.com
Refereed Proceedings of the National Conference on Mathematics and Computer Applications held at the Department of Mathematics, Women's Christian College, Chennai, India from January 29, 2019 to January 30, 2019.

have their own production rules and start with their own axiom. In this system, all components work in parallel and communicate between them through query symbols to generate a terminal string. The component that generates the terminal string is known as the master of the system.

In this paper we use parallel communicating grammars which communicate through queries to capture the splicing mechanism on regular matrix grammars without splicing rules as in Samdanielthompson et al. [2]. The language generated by this Parallel Communicating Grammar System on Regular Matrix Grammar using Queries (PCQRML) is incomparable with the splicing array grammar system language in Subramanian et al. [3] and contains the flat splicing array language in Samdanielthompson et al. [2].

2 Preliminaries

An array M of size $m \times n$ over an alphabet Σ is a rectangular array of m rows and n columns of the form

$M = \begin{matrix} x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{matrix}$ where each $x_{ij} \in \Sigma, 1 \leq i \leq m, 1 \leq j \leq n$. The set of all arrays over Σ is denoted by Σ^{**}, which also includes the empty array λ without symbols. Consider two arrays A of size $p \times q$ and B of size $r \times s$. The row catenation of these arrays $A\theta B$ is defined only when $q = s$ and column catenation $A\phi B$ is defined only when $p = r$. That is, in row catenation $A\theta B$, $B$is placed below A while in column catenation $A\phi B$, $B$is placed to the right of A.

For example if $A = \begin{bmatrix} a & a & a \\ a & a & a \end{bmatrix}$, $B = \begin{bmatrix} b & b & b & b \\ b & b & b & b \\ b & b & b & b \end{bmatrix}$, $C = \begin{bmatrix} c & c & c \\ c & c & c \\ c & c & c \end{bmatrix}$

Then $A\theta C = \begin{bmatrix} a & a & a \\ a & a & a \\ c & c & c \\ c & c & c \\ c & c & c \end{bmatrix}$ and $B\phi C = \begin{bmatrix} b & b & b & b & c & c & c \\ b & b & b & b & c & c & c \\ b & b & b & b & c & c & c \end{bmatrix}$.

Also $A\theta\lambda = \lambda\theta A = A$ and $A\phi\lambda = \lambda\phi A = A$.

Definition 2.1. A matrix grammar $M = (G, G')$ is said to be a regular matrix grammar if $G = (V, I, P, S)$ is a regular grammar, where $I = \{S_1, S_2, \ldots, S_n\}$ and $G' = \left\{G_1^{'}, G_2^{'}, \ldots, G_k^{'}\right\}$ where each $G_i^{'} = \{V_i, T_i, P_i, S_i\}$ are length equivalent regular grammars. If there exist strings $\alpha_1\alpha_2 \ldots \alpha_k$ such that $\alpha_i \in L\left(G_i^{'}\right)$, then $|\alpha_1| = |\alpha_2| = \ldots = |\alpha_k|, 1 \leq i \leq k$. P_i denotes the table of regular vertical rules.

Let $I = c_1\theta c_2\theta \ldots \theta c_n$ be an image defined over Σ. $I \in M(G)$ iff there exist $S_1, S_2, \ldots, S_n \in L(G)$such that $c_j \in L(G_j), 1 \leq j \leq n$. The string $S_1S_2 \ldots S_n$ is said to be an intermediate string deriving $I$with respect to M. Note that there can be more than one intermediate string deriving I. The family of languages generated by $(X : Y)MG$ is denoted as $(X : Y)ML$ where $X, Y \in \{R, R\}$.

Example 2.2. Consider a regular matrix grammar $M = (G, G')$ where G represents the horizontal rules and G'represents the vertical table rules of the grammar. Then

$$G = (\{S, A, B\}, \{S_1, S_2, S_3\}, \{S \to AC, C \to BC, A \to S_1, B \to S_2, C \to S_3\}, S), G' = G_1^{'} \cup G_2^{'} \cup G_3^{'}$$

where $G_i^{'} = (V_i, T_i, P_i, S_i), V_i = (S_i, A, B, C), T_i = (a, b, c)$,

$$P_i = \{\{t_1 = \{S_1 \to aA, A \to a\}, t_2 = \{S_2 \to bB, B \to b\}, t_3 = \{S_3 \to cC, C \to c\}\}\},$$

for $i = 1, 2, 3$. Then

$$\begin{array}{c} S \to AB \Rightarrow^* S_1S_2S_2S_3 \\ \Downarrow_* \\ \begin{matrix} a & b & b & c \\ a & b & b & c \\ a & b & b & c \end{matrix} \end{array}$$

In the above example the set of all matrices generated by M is

$$L\left(M\right)=\left\{\left[S_1S_2^mS_3\right]/m\geq 0\right\}=\left\{\begin{bmatrix} a & b & b & c\\ a & b & b & c\\ a & b & b & c\end{bmatrix}\right\}.$$

Definition 2.3. A parallel communicating grammar system is an $n+3$ tuple of degree $n\geq 1$.

$\Omega=\left(N,Q,T,\left(S_1,P_1\right),\ldots,\left(S_n,P_n\right)\right)$where, N is a finite set of non terminals, T is a finite set of terminals, $Q=\{q_1,q_2,\ldots,q_n\}$ is a finite set of query symbols, P_i is a finite set of rewriting rules over $N\cup Q\cup T$ and $S_i\in N$, for $1\leq i\leq n$.

3 Parallel Communicating grammar system for Regular Matrix Grammar

Definition 3.1. A Parallel Communicating grammar system for Regular Matrix grammar is a construct $G=\left\{N_h,N_v,N_i,T,Q,\left(S_1,P_1^h,P_1^v\right),\ldots,\left(S_n,P_n^h,P_n^v\right)\right\}$ where N_h denotes set of horizontal non-terminal alphabet symbols, N_v denotes set of vertical non-terminal alphabet symbols, N_i denotes set of intermediate non-terminal alphabet symbols, T denotes the terminal alphabet**,** $Q=\{q_1,q_2,\ldots,q_n\}$ denotes a finite set of query symbols. $\left(S_i,P_i^h,P_i^v\right)$ denotes the components of regular matrix grammars over T, S_i denotes the start symbol of the corresponding component, P_i^h denotes a finite set of regular horizontal rules, $P_i^v=\{t_1,t_2,\ldots,t_n\}$ denotes a finite set of table of regular vertical rules that generate vertical strings for $1\leq i\leq n$. The query q_i belongs to the component P_i.

3.1 Rewriting in the components $\left(S_i,P_i^h,P_i^v\right)$

The rewriting in the components $\left(S_i,P_i^h,P_i^v\right)$ is done in two phases. In the first phase each component grammar starts from its own start symbol and generates a word called the intermediate word using its own horizontal rules. The intermediates generated here act as terminals of this phase. Here all the component grammars work in parallel. When a component i generates a query symbol q_j, then the current intermediate word of the j^{th} component is communicated to the i^{th} component, replacing the query symbol q_j, thus splicing horizontally the j^{th} component intermediate word to the right side /in between/left side of the i^{th} component intermediates. Once communicated, the j^{th}component goes back to its non returning mode. This way the horizontal splicing of intermediates is done. A component x_i is spliced only when all occurrences in it of query symbols refer to, intermediates without occurrences of query symbols.
In the second phase, each component rewrites as in a two dimensional matrix grammar using the tables of vertical rules starting from its own intermediate word generated in the first phase. All the component grammars work in parallel with the vertical table rules of each component. When a query symbol q_j appears in the i^{th} component of the vertical table rule then the vertical string in the j^{th} component is communicated to the i^{th} component replacing the query q_j and the j^{th}component goes back to non returning mode, i.e., it continues to work with the vertical string obtained. This yields row splicing of the i^{th} component with the j^{th} component at the top/in between/at the bottom of the i^{th} component vertical string. Note that for row splicing the number of columns of both the component strings should be equal. Here the component grammars together continue rewriting in the vertical direction with all rules used either in the form $A\to aA$ or together terminate with all the rules used of the form $A\to a$. The set of all such spliced regular matrices Z for the language generated by this parallel communicating grammar system using queries is known as Parallel Communicating on Regular Matrix Language using queries (PCQRML).

Example 3.2. Consider a parallel communicating regular matrix grammar G with two components given by
$G=\left(\{S_1,S_2,A,B,C,X,Y\},\{A,C,D\},\{A,C,X,Y\},\{a,c,d\},Q,\left(S_1,P_1^h,P_1^v\right),\left(S_2,P_2^h,P_2^v\right)\right)$ where, $Q=\{q_1,q_2\}$ is a finite set of query symbols.

$$P_1^h=\{S\to AB,B\to CB,B\to A,B\to q_2C,S_1\to Aq_2,S_1\to q_2A,A\to Aq_2\},$$

$$P_1^v : \{t_1 = \{A \to aA, A \to a, A \to q_2A, A \to Aq_2\}, t_2 = \{C \to cC, C \to Cq_2, C \to c, C \to q_2C\}\},$$
$$P_2^h = \{S_2 \to XY, Y \to XY, Y \to X, S_2 \to q_1Y, S_2 \to Yq_1, Y \to q_1X\},$$
$$P_2^v : \{t_1 = \{X \to dD, D \to d\}, t_2 = \{D \to q_1D, D \to Dq_1\}\}.$$

Column splicing of the above grammar through query, generates the picture matrix language

$$(S_1, S_2) \Rightarrow (AB, XY) \Rightarrow (Aq_2C, XX) \Rightarrow (AXXC, XX) \Rightarrow^* \left(\begin{bmatrix} a & d & d & c \\ a & d & d & c \\ a & d & d & c \end{bmatrix}, \begin{bmatrix} d & d \\ d & d \\ d & d \end{bmatrix} \right)$$

Row splicing of the above grammar through query in vertical table rules generates the picture matrix language

$$(S_1, S_2) \Rightarrow (AB, XY) \Rightarrow (ACB, XXY) \Rightarrow (ACA, XXX)$$

$$(ACA, XXX) \Rightarrow (A\phi C\phi A, X\phi X\phi X) \Rightarrow (aA\phi cC\phi aA, dD\phi dD\phi dD) \Rightarrow (aAq_2\phi cCq_2\phi aAq_2, dd\phi dd\phi dd) \Rightarrow$$

$$(aAdd\phi cCdd\phi aAdd, dd\phi dd\phi dd) \Rightarrow \left((aadd)^t\phi(ccdd)^t\phi(aadd)^t, (dd)^t\phi(dd)^t\phi(dd)^t\right)$$
$$\Rightarrow \left(\begin{bmatrix} a & c & a \\ a & c & a \\ d & d & d \\ d & d & d \end{bmatrix}, \begin{bmatrix} d & d & d \\ d & d & d \end{bmatrix} \right).$$

Theorem 3.3. *The language generated by the two component Splicing Array Grammar Systems* $L_2(SAGS)$ *and the matrix language generated by the two component Parallel Communicating Grammar System on Regular Matrix Language using queries* $L_2(PCQRML)$ *are disjoint, i.e.,* $L_2(SAGS) \cap L_2(PCQRML) = \varphi$.

Proof. The array language generated by SAGS [3] is such that based on domino rules the row/column splicing between any two array components is done by cutting rows/columns of one or both array components and pasting the resulting array components together. This eliminates some of the rows or columns or both from the array language generated by SAGS. But we see that in PCQRML splicing is done at the top, at the bottom , to the right side, to the left side and in between rows/ columns of two component matrices without eliminating any row/column of these matrices. Thus we see that the array language generated by SAGS is different from the matrix language generated by PCQRML. Hence $L_2(SAGS) \cap L_2(PCQRML) = \varphi$. □

Theorem 3.4. $L_2(FSRAGS) \subset L_2(PCQRML)$.

Proof. The matrix language L that is being generated by FSRAGS [2] is such that the array of one component is inserted into the array of another component. But the matrix language of PCQRML is such that the horizontal splicing between two component places the j^{th} component intermediate word with the i^{th} component intermediate to the right side /left side/in between the i^{th} component intermediates replacing the query $q_j$and similarly the row splicing of the i^{th} component with the j^{th}component places the j^{th} component vertical table terminals at the top/bottom/ in between the i^{th} component vertical string. Therefore we see that in PCQRMG splicing is done at the top, at the bottom, to the right side, to the left side and in between rows/ columns of two component matrices generating all types of matrix languages. As FSRAGS splicing is restricted only to the row/columns, PCQRML contains languages that cannot be generated by FSRAGS. Hence $L_2(FSRAGS) \subset L_2(PCQRML)$. □

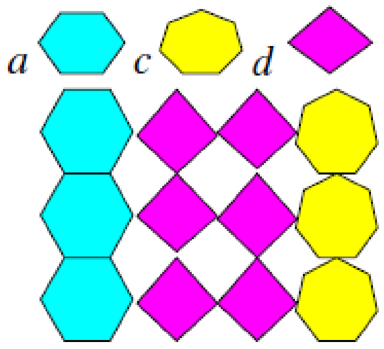

Fig. 1:

4 Application to pictures

The matrix generated using queries to splice column in Example 3.2 generates the matrix $\begin{bmatrix} a & d & d & c \\ a & d & d & c \\ a & d & d & c \end{bmatrix}$ which in turn can generate the following design by mapping each alphabet to a design as shown in the Fig.1.

The row splicing of the grammar in Example 3.2 generates the matrix $\begin{bmatrix} a & c & a \\ a & c & a \\ d & d & d \\ d & d & d \end{bmatrix}$ which in turn can generate the following design by mapping each alphabet to a design as depicted in the Fig.2.

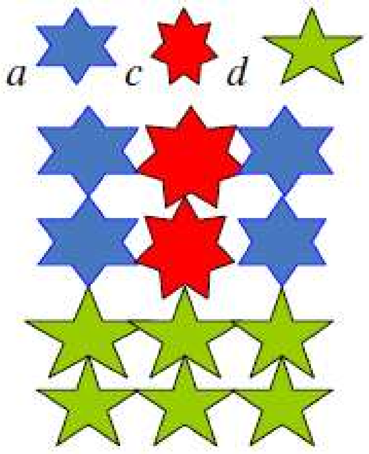

Fig. 2:

5 Conclusion

In this paper we use parallel communicating grammars which communicate through queries to capture the splicing mechanism on regular matrix grammars without splicing rules and whose components work in parallel and generate pictures. The set of all such spliced matrices Z for the language generated by this grammar is known as Parallel Communicating on Regular Matrix Language using queries

(PCQRML) which help in describing complex floor and tile designs.

References

[1] Dassow, Jurgen, Paun, Gheorghe and Rozenberg, Grzegorz (1997). Grammar Systems, Handbook of Formal Languages, Springer, Berlin, Heidelberg, 155–213.

[2] Samdanielthompson, G., David, N. Gnanamalar, Nagar, Atulya K. and Subramanian, K.G. (2016). Flat splicing array grammar systems generating picture arrays, *International Journal of Computer Information System and Industrial Management Applications*, Vol 8, 336–344.

[3] Subramanian, K.G., Mary, A. Roslin Sagaya and Dersanambika, K.S. (2005). Splicing array grammar systems, *Proceedings of the Second International conference on Theoretical Aspects of Computing ICTAC*, 125–135.

[4] Siromoney, G., Siromoney, R. and Krithivasan, K. (1972). Abstract families of matrices and picture languages, *Computer Graphics and Image Processing*, 1, 234–307.

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