International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-12, October 2019

# Splicing Context-Free Matrix Grammars using Parallel Communicating Grammar Systems

**M. Iffath Mubeen, J.D.Emerald**

***Abstract—Extensive research on splicing of strings in DNA computing has established important theoretical results in computational theory. Further, splicing on strings has been extended to arrays in[2]. In this context, we propose, a grammar system, using queries to splice context-free matrix grammars and show that the language generated by this grammar system is incomparable to the language given in [3] and has more generative power than in [2].***

## I. INTRODUCTION

Matrix grammars were introduced by Siromoney et.al [4,5] to generate two- dimensional picture arrays to 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 consists of a finite set of grammars to generate a language. Sequential and parallel are the two types of grammar systems that are defined [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 have their own production rules and start with their own axiom. In this system, all components work in parallel and communication between them is done through query symbols to generate a terminal string.

In this paper, we use parallel communicating grammars which communicate through queries to capture the splicing mechanism on context-free matrix grammars without splicing rules as in [2]. The language generated by this Parallel Communicating grammar system on Context-Free Matrix grammar using Queries (PCQCFML) is incomparable with the splicing array grammar system language in [3] and contains the flat splicing context-free array language in [2].

## II. PRELIMINARIES

**Definition 2.1** A matrix grammar M= (G, G´) is said to be a context-free matrix grammar if G= (V, I, P, S) is a context-free grammar that generates horizontal strings,

**Revised Manuscript Received on October 30, 2019.**
***** Correspondence Author
**M. Iffath Mubeen*,** Assistant Professor, Department of Mathematics, Government Arts College for Men, Chennai, TamilNadu, India.(Email: iffathmalick@hotmail.com)
**J.D.Emerald,** Assistant Professor, Department of Mathematics,Queen Mary's College, Chennai, TamilNadu, India. (E-mail: emeraldsolomon96@gmail.com)



where I = {$S_1$, $S_2$,,…$S_n$} and G´ = {V', T', P', $S_i$} is length equivalent context-free grammar for 1≤i≤n. Here P' has table rules P' ={$t_k$}, 1≤k≤m where $t_m$has rules of the form {$X_i$→ wA} or {$X_i$ →wAw} or {$X_i$→w}. If there exist strings $\alpha_1\alpha_2\ldots\alpha_k$ such that $\alpha_i \in L(G_i')$, then $|\alpha_1| = |\alpha_2| = \ldots = |\alpha_k|$, 1≤i≤k.

Let I = $c_1 \Theta c_2 \Theta \ldots \Theta c_n$ be an image defined over Σ. I ∈M(G) iff there exists $S_1, S_2 \ldots S_n \in L(G)$ such that $c_j \in L(G_j)$, 1≤j≤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 ∈ {CF, CF}.

*Example 2.1*

Consider a context-free matrix grammar

M= (G, G') where G-represents horizontal rules and G'-represents vertical table rules of the grammar

G= ({S,A,B,C},{$S_1$,$S_2$,$S_3$},{S→ ABC, A→ AB, A→$S_1$, B→ $S_2$, C→ $S_3$},S), G´= (V', T', P', $S_i$) where V'= {$S_1$,$S_2$,$S_3$, A,B,C}, T'={r,t,u} and P' ={$t_1$,$t_2$,$t_3$} where

$$t_1 = \begin{bmatrix} S_1 \to rrA \\ S_2 \to ttB \\ S_3 \to uuC \end{bmatrix} \quad t_2 = \begin{bmatrix} A \to rrA \\ B \to ttB \\ C \to uuC \end{bmatrix}, t_3 = \begin{bmatrix} A \to r \\ B \to t \\ C \to u \end{bmatrix}$$

Then S →ABC==>$^*$ $S_1$ $S_2$ $S_3$

$\Downarrow *$

$r \quad t \quad t \quad u$

$r \quad t \quad t \quad u$

$r \quad t \quad t \quad u$

In the above example the set of all matrices generated by M is L (M) = {$S_1S_2^kS_3$/k≥1}=

$$\left\{\begin{bmatrix} r & t & t & u \\ r & t & t & u \\ r & t & t & u \end{bmatrix}\right\}$$

## III. PARALLEL COMMUNICATING GRAMMAR SYSTEM FOR CONTEXT-FREE MATRIX GRAMMAR

**Definition 3.1** A Parallel Communicating grammar system for Context-Free Matrix grammar is a construct G={$N_h$, $N_v$, $N_i$, T, Q, ($S_1$,$P_1^h$,$P_1^v$), …, ($S_n$, $P_n^h$, $P_n^v$)} 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 alphabets, Q = $\{q_1,q_2,..,q_n\}$ denotes a finite set of query symbols. $(S_i,P_i^h,P_i^v)$ denotes the components of context-free matrix grammars over T, $S_i$ denotes the start symbol of the corresponding component for $1\le i\le n$, $P_i^h$ denotes a finite set of context-free horizontal rules, $1\le i\le n$, $P_i^v = \{t_{ik}\}$ denotes a finite set of table of context-free vertical rules that generate vertical strings, $1\le i\le n$, $1\le k\le m$. The query $q_i$ belongs to the component $P_i$.

*Rewriting*

The rewriting of the components $(S_i, P_i^h, P_i^v)$ is done in two phases. In the first phase each component grammar starts from its own start symbol $S_i$ and using its own horizontal rules generates a intermediate word. These intermediates 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, using the table of vertical rules each component rewrites as in a two dimensional matrix grammar. Here the intermediate word generated in the first phase act as the start symbol. All the component grammars work in parallel with their respective vertical table rules. When a query symbol $q_j$ appear 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 $j^{th}$ component goes back to non returning mode i.e. it continues to work with the vertical string obtained. This leads to 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 the context-free rules or together terminate with all the rules used of the form $A\rightarrow r$.

The set of all such spliced context-free matrices Z for the language generated by this parallel communicating grammar system using queries is known as Parallel Communicating on Context-Free Matrix Language using queries (PCQCFML)

*Example 3.1*

Consider a parallel communicating context-free matrix grammar G with two components given by

G =({$S_1$, $S_2$, A,B,C,X,Y, Z}, {A, B,C, X,Y}, {A, B, C, X, Y}, {r, t, u,v,w},Q, $(S_1,P_1^h,P_1^v)$, $(S_2, P_2^h, P_2^v)$)

where $Q=\{q_1,q_2\}$is a finite set of query symbols

$P_1^h$: {$S_1\rightarrow$ ABC, A$\rightarrow$ AB, A$\rightarrow$AB$q_2$, $S_1\rightarrow$ A$q_2$C, $S_1\rightarrow$ $q_2$AC, $S_1\rightarrow$A$q_2$}

and $P_1^v$:

$$t_{11}=\begin{bmatrix} A\rightarrow rrA \\ B\rightarrow ttB \\ C\rightarrow uuC \end{bmatrix}, t_{12}=\begin{bmatrix} A\rightarrow q_2A \\ B\rightarrow q_2B \\ C\rightarrow q_2C \end{bmatrix}, t_{13}=\begin{bmatrix} A\rightarrow Aq_2 \\ B\rightarrow Bq_2 \\ C\rightarrow Cq_2 \end{bmatrix}, t_{14}=\begin{bmatrix} A\rightarrow rq_2A \\ B\rightarrow tq_2B \\ C\rightarrow uq_2C \end{bmatrix}, t_{15}=\begin{bmatrix} A\rightarrow r \\ B\rightarrow t \\ C\rightarrow u \end{bmatrix}$$

$P_2^h$ :{$S_2\rightarrow$ XYZ, Y$\rightarrow$YZ, Z$\rightarrow$Y, $S_2\rightarrow q_1$Z, $S_2\rightarrow$X$q_1$Z, Z$\rightarrow$Y$q_1$}

and $P_2^v$:

$$t_{21}=\begin{bmatrix} X\rightarrow vvX \\ Y\rightarrow wwY \end{bmatrix}, t_{22}=\begin{bmatrix} X\rightarrow q_1X \\ Y\rightarrow q_1Y \end{bmatrix}, t_{23}=\begin{bmatrix} X\rightarrow Xq_1 \\ Y\rightarrow Yq_1 \end{bmatrix}, t_{24}=\begin{bmatrix} X\rightarrow vq_1X \\ Y\rightarrow wq_1Y \end{bmatrix}, t_{25}=\begin{bmatrix} X\rightarrow v \\ Y\rightarrow w \end{bmatrix}$$

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

$(S_1,S_2)\Rightarrow$(ABC,XYZ) $\Rightarrow$(ABBC, XYY$q_1$) $\Rightarrow$(ABBC, XYYABBC)$\Rightarrow^*$

$$\left\{\begin{bmatrix} r & t & t & u \\ r & t & t & u \\ r & t & t & u \end{bmatrix}, \begin{bmatrix} v & w & w & r & t & t & u \\ v & w & w & r & t & t & u \\ v & w & w & r & t & t & u \end{bmatrix}\right\}$$

**Figure. 1**

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

$(S_1, S2)\Rightarrow$(ABC, XYZ) $\Rightarrow$(ABC, XYY) $\Rightarrow$(A$\phi$B$\phi$C, X$\phi$Y$\phi$Y)

$\Rightarrow$(rrA$\phi$ttB $\phi$ uuC, vq$_1$X$\phi$wq$_1$Y$\phi$wq$_1$Y)$\Rightarrow$(rrr $\phi$ ttt $\phi$ uuu, vrrrX $\phi$ wtttY $\phi$ wuuuY) $\Rightarrow$$((rrr)^t$ $\phi$ $(ttt)^t$ $\phi$ $(uuu)^t$, $(vrrrv)^t$ $\phi$ $(wtttw)^t$ $\phi$ $(wuuuw)^t$) $\Rightarrow$

$$\left\{\begin{bmatrix} r & t & u \\ r & t & u \\ r & t & u \\ r & t & u \end{bmatrix}, \begin{bmatrix} v & w & w \\ r & t & u \\ r & t & u \\ r & t & u \\ v & w & w \end{bmatrix}\right\}$$

**Figure. 2**

**Theorem 3.1** The language generated by the two component Splicing Array Grammar Systems $L_2$(SAGS) and the matrix language generated by two component Parallel Communicating grammar sytem on Context-Free Matrix Language using queries $L_2$(PCQCFML) are disjoint i.e. $L_2$(SAGS) $\cap$ $L_2$(PCQCFML) = $\varphi$.

**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 PCQCFML 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 PCQCFML. Hence $L_2$(SAGS) ∩ $L_2$(PCQCFML) = φ.

**Theorem 3.2** $L_2$(AFSCFAGS) ⊂ $L_2$(PCQCFML)

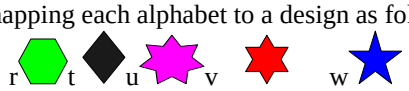
**Proof:** The matrix language L that is being generated by AFSCFAGS [2] is such that the array of one component is inserted into the array of another component. But the matrix language of PCQCFMLis such that the horizontal splicing between two components places the $j^{th}$ component intermediate word 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 $j^{th}$ component vertical table terminals at the top/bottom/ in between the $i^{th}$ component vertical string. Therefore we see that in PCQCFMG 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 splicing is done only between rows/columns of any two component arrays in AFSCFAGS, the languages generated by PCQCFML cannot be generated by AFSCFAGS. Hence $L_2$(AFSRAGS) ⊂ $L_2$(PCQCFML).

## IV.APPLICATION TO PICTURES& RESULTS

The matrix generated using queries to splice column in Example 3.1 generates thematrix

$$\left\{\begin{bmatrix} v & w & w & r & t & t & u \\ v & w & w & r & t & t & u \\ v & w & w & r & t & t & u \end{bmatrix}\right\}$$

which in turn can generate the following designby mapping each alphabet to a design as follows



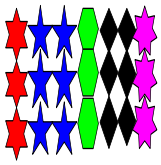



**Figure. 3**

The row splicing of the grammar in Example 3.1 generates the matrix and its design as follows

$$\left\{\begin{bmatrix} v & w & w \\ r & t & u \\ r & t & u \\ r & t & u \\ v & w & w \end{bmatrix}\right\}$$

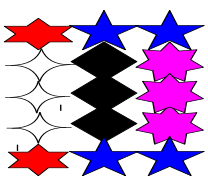
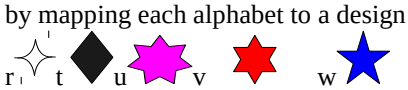


**Figure. 4**

by mapping each alphabet to a design as follows




## V. CONCLUSION

In this paper we use parallel communicatinggrammars which communicate through queries to splicecontext-free matrix grammars without using any splicing rule and whose components work in parallel. The spliced matrices such obtained can be used to generate pictures. The set of all such spliced matrices Z for the language generated by this grammar is known as Parallel Communicating on Context-Free Matrix Language using queries(PCQCFML) which helps in describing complex floor and tile designs.

## REFERENCES

1 Jurgen Dassow, Gheorghe Paun and Grzegorz Rozenberg, Grammar Systems, Handbook of Formal Languages, Springer, Berlin, Heidelberg (1997) 155-213.
2 G.Samdanielthompson, N.G. David,K.G.Subramanian, Flat Splicing Context-Free Array Grammar System for Generating Picture Arrays, International Journal of Artificial Intelligence and Soft Computing, Vol. 5(4), (2016) 294-310.
3 K.G. Subramanian, A. Roslin Sagaya Mary and K.S. Dersanambika Splicing Array Grammar Systems Proceedings of the Second international conference on Theoretical Aspects of Computing ICTAC (2005) 125-135.
4 G. Siromoney, R. Siromoney and K. Krithivasan, Abstract families of matrices and picture languages, Computer Graphics and Image Processing, 1, (1972) 234-307.
5 A. Rosenfeld, R. Siromoney, Picture languages- a survey, Languages of Design, 1, (1993) 229-245.