PETRI NETS GENERATING KOLAM PATTERNS

D. Lalitha

Department of Mathematics
Sathyabama University, Chennai-119, India
lalkrish_24@yahoo.co.in

K. Rangarajan

Department of MCA
Bharath University, Chennai-73, India
kr2210@hotmail.com

Abstract

Array Token Petri Nets are known to generate all the nine families generated by Array Grammars [10]. Application of two dimensional array grammars in kolam generation has been discussed [1]. Motivated by this we have used Array Token Petri Nets to generate kolam patterns. Parallel generation and sequential/parallel generation are the two types of generation used. The Kolam patterns which are generated by Regular Grammars are generated by Array Token Petri Nets and Kolam patterns which are generated by Context-free and Context-sensitive grammars are generated by Array Token Petri Nets with inhibitor arcs.

Keywords: Kolam; array token; parallel generation; sequential generation; Petri net; inhibitor arc.

1. Introduction

Kolam is a traditional art of decorating courtyards in South India. Every kolam pattern can be treated as a two dimensional language over the alphabet of tiles. Theoretical models generating two dimensional arrays have been used to generate complicated kolam patterns [1, 2, 3, 4, 5]. Classification of kolam patterns has been done based on the different array grammars which generate them [1]. Several approaches have been used to generate these intricate designs.

On the other hand Petri net is an abstract formal model of information flow [7]. Petri nets have been used for analyzing systems that are concurrent, asynchronous, distributed, parallel, non deterministic and / or stochastic. Tokens are used in Petri nets to simulate dynamic and concurrent activities of the system. A language can be associated with the execution of a Petri net. By defining a labeling function for transitions over an alphabet, the set of all firing sequences, starting from a specific initial marking leading to a finite set of terminal markings, generates a language over the alphabet.

Petri net model to generate arrays of tiles has already [9] been introduced. In this model arrays over a given alphabet of square tiles of same size are used as tokens in the places of the net. Joining of two square tiles can be done in four ways. → (right), ← (left), ↑ (up) and ↓ (down). Catenation rule is used as label of transition. Firing a sequence of transitions starting from a specific initial marking leading to a finite set of terminal markings would join tiles according to the rule and move the array to the final set of places. This model generates the kolam patterns that can be generated by regular grammars. To increase the generative capacity we have introduced the concept of inhibitor arcs. Inhibitor arcs are used to generate the kolams generated by context-free or context-sensitive grammars.

This paper is organized as follows. Section 2 gives the preliminary definitions, defines Kolam Array Token Petri Net Structure (KATPNS) and gives an example. Section 3 defines KATPNS with inhibitor arcs and gives an example. Section 4 compares the models with Regular and Context-free array grammars generating kolams.

2. Kolam Array Token Petri Nets

In this section we define Petri net, give the preliminary notations, define Kolam Array Token Petri Net Structure and also give an example.

Definition 1. A Petri Net structure is a four tuple C = (P,T,I,O) where P = $\{p_1, p_2, \ldots, p_n\}$ is a finite set of places, $n \geq 0$, T = $\{t_1, t_2, \ldots, t_m\}$ is a finite set of transitions $m \geq 0$, $P \cap T = \emptyset$, $I: T \rightarrow P^{\infty}$ is the input function from transitions to bags of places and $O: T \rightarrow P^{\infty}$ is the output function from transitions to bags of places.

Definition 2. A Petri Net marking is an assignment of tokens to the places of a Petri Net. The tokens are used to define the execution of a Petri Net .The number and position of tokens may change during the execution of a Petri Net.

Definition 3. An inhibitor arc from a place p_i to a transition t_j has a small circle in the place of an arrow in regular arcs. This means the transition t_j is enabled only if p_i has no tokens. A transition is enabled only if all its regular inputs have tokens and all its inhibitor inputs have zero tokens.

To generate kolams we use alphabets consisting of square tiles of same size with patterns in it. A certain family of patterns called 'Kambi Kolam' over a square grid, is expressed with regular dotted square tiles. Nagata et al [8] have given a digital representation to such tiles. The set of primitives given below are primitives used in generating 'Kambi Kolam' [8].

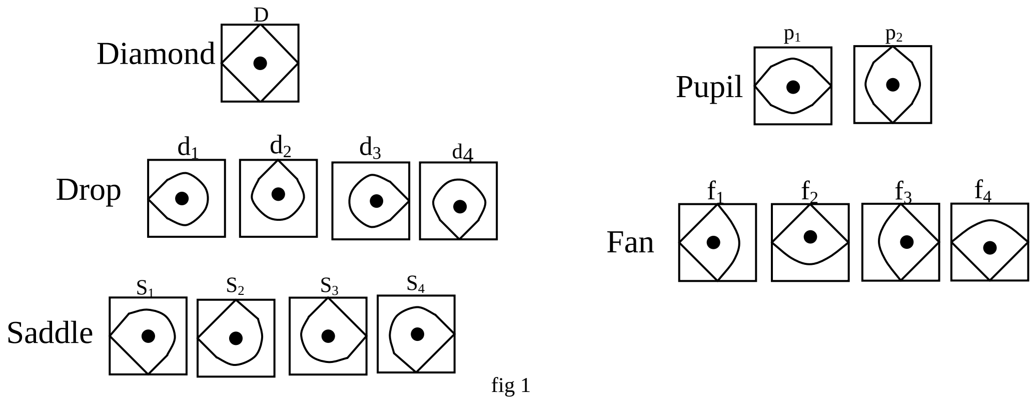


fig 1

We denote a square tile without any pattern by B (blank). Arrays over the alphabet of tiles will form a pattern. For example the array $\begin{matrix} B & d_4 & B \\ d_3 & D & d_1 \\ B & d_2 & B \end{matrix}$ with the alphabet taken from fig 1, will form the kolam in fig 2.

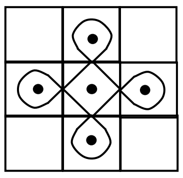


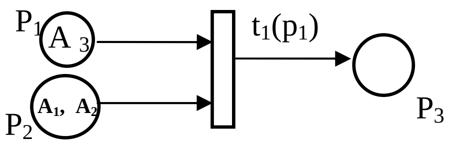
fig 2

In this model arrays over the alphabet of tiles is used as tokens in places. If all the input places of a transition have the same array as tokens then firing the transition moves the array from the input place to its output place. If all the input places of a transition have different arrays or if one input place has different arrays then the transition without label cannot fire. Two types of labels are used in this model.

I. Name of an input place:

If the transition has many input places with different arrays as tokens then the label specifies
an input place which has sufficient number of tokens of the same array. Then firing the transition will remove tokens from all its regular input places and the array which is in the (label specified) place will be moved into the output place. An example of this is given in fig 3.

Position of tokens before firing:



Position of tokens after firing:

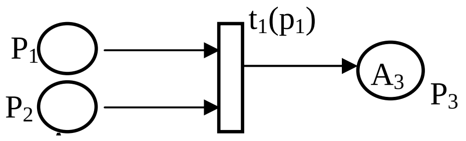


fig 3

II. Catenation rule:

If T is a transition of a net then the label σ(T) is a set of catenation rules. When any catenation rule is used as a label of a transition, all the input places of the transition should have the same array as token. A transition with input places having different arrays and a catenation rule as label cannot fire.

Catenation of tiles or joining of tiles can be done in four directions → (right), ← (left), ↑ (up) and ↓ (down). For A, D Σ^{**} the four catenation rules are given below.

A→D joins D to the right of A so that right side of A and left side of D coincide.

A← D joins D to the left of A so that the left side of A and right side of D coincide.

In the above two cases A and D should have the same number of rows.

A ↑ D joins D above A so that the top of A and bottom of D coincide.

A ↓ D joins D below A so that the bottom of A and top of D coincide.

In the above two cases A and D should have the same number of columns.

In applying catenation rules two types of generations are considered (i) Parallel and (ii) Sequential/ Parallel.

Parallel generation: Rules are applied in a parallel manner in this generation. If A→D is the rule applied then all A in the last column will be joined to D in the right. If a right rule → is used as a label all the tiles in the last column of the array in the input place will have to be catenated with some tile in the alphabet. If a rule is missing for any of the tile, then a blank tile will be joined to the specific tile. For example let the input place of a transition T have the array $\begin{matrix} X & D & X \\ D & D & D \\ X & D & X \end{matrix}$ and σ(T) = { X →X, D→X }. In parallel generation all X on the last column will be catenated to X and D is catenated to X. Hence firing the transition will join X to the right of all tiles of the array. The resultant array reaching the output place would be $\begin{matrix} X & D & X & X \\ D & D & D & X \\ X & D & X & X \end{matrix}$.

If σ(t) = { X →X}then firing the transition will join X to the right of all X and a blank tile B to the right of D. The resultant array reaching the output place would be $\begin{matrix} X & D & X & X \\ D & D & D & B \\ X & D & X & X \end{matrix}$. Similarly in all other three rules we apply the rules in a parallel manner and missing rule is automatically joined to B in the specified direction.

Sequential/ Parallel generation

In kolam patterns we find two kinds of *pullis* used. One is rectangular and the other is in the form of diamond or triangular. Hence parallel generation may not be able to generate all kolams. Hence we use Sequential/ Parallel generation.

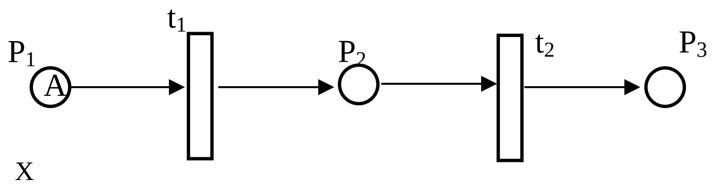
Column Rule: In case all X in the last column (first column) of the array is not required to be joined to X using right rule (left rule). If the topmost(tm) X and bottommost(bm) X is to be joined to a blank tile then we use the notation tmX→B (tmX←B), bmX→B (bmX←B). Except the first X and the last X of the column all the other X are catenated to X in a parallel manner. For example let the input place of a transition T have the array $\begin{matrix} X & X & X \\ X & D & X \\ X & X & X \end{matrix}$.

If σ(T)={X →X, tmX→B, bmX→B }. The resultant array reaching the output place would be $\begin{array}{cccc} X & X & X & B \\ X & D & X & X \\ X & X & X & B \end{array}$.

Row Rule:
In case all X in the first row (last row) of the array is not required to be joined to X using up rule (down rule). If the leftmost(lm) X and rightmost(rm) X is to be joined to a blank tile then we use the notation lmX↑B (lmX↓ B), rmX↑B (rmX↓ B). Except the first X and the last X of the row all the other X are catenated to X in a parallel manner. For example let the input place of a transition T have the array $\begin{array}{ccc} X & X & X \\ X & D & X \\ X & X & X \end{array}$.

If σ(T)={ lmX↑B, rmX↑B, X ↑X, }. The resultant array reaching the output place would be $\begin{array}{ccc} B & X & B \\ X & X & X \\ X & D & X \\ X & X & X \end{array}$.

For example consider the net in the following figure



If A = $\begin{array}{ccc} X & X & X \\ X & D & X \\ X & X & X \end{array}$, σ($t_1$) = {X ←Y} and σ($t_2$) = { lmX↑B, rmX↑B, X ↑X, } then after firing t_1 the array that reaches the place p_2 is $\begin{array}{cccc} Y & X & X & X \\ Y & X & D & X \\ Y & X & X & X \end{array}$**.** All X in the first column is joined to y in the left. Once this array reaches p_2 the transition t_2 is enabled and so when t_2 fires the array that reaches p_3 will be $\begin{array}{cccc} B & B & X & B \\ Y & X & X & X \\ Y & X & D & X \\ Y & X & X & X \end{array}$. The left most of X and the right most of X are joined to a blank tile and the middle X is joined to X. Since no rule is given for Y it is being joined with a blank tile. All these catenations are done in the upward direction. In general all rules are applied in a parallel manner unless the prefix rm, lm, tm and bm is used along with a rule. Hence we call it sequential/parallel generation.

Definition 4. A Kolam Array Token Petri Net Structure (KATPNS) is a four tuple (C, Σ, σ, F) where C = (P, T, I, O) is a Petri net structure, Σ an alphabet of square tiles of same size, σ partial function on the set of transitions which define the labels, M_0:P→Σ^{**} is the initial marking and F ⊆ P is a set of final places.

Definition 5. The language generated by the KATPNS is L(C) = {A Σ^{**}/ A is in p for some p in F}. With arrays of Σ^{**} in some places as initial marking all possible sequences of transitions are fired. The set of all arrays collected in the final places F is called the language generated by C.

Example 1. The KATPNS (C, Σ, σ, F) where the graph of C is given in fig 6, Σ ={X, T, R, D, L, C_1, C_2,C_3, C_4}, σ the partial function is given in fig 5, S = X is the initial marking in $p_1$and p_3 and F = {p_2}. The generation here is parallel. The tiles are given in fig 4. The arrays reaching p_2 are collected as the language generated.

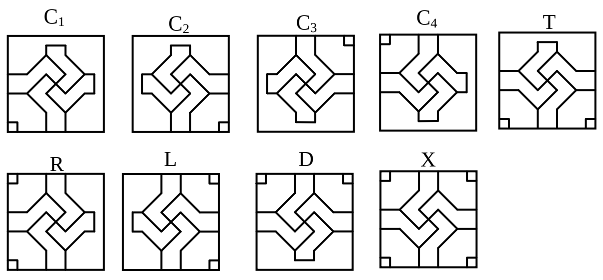


fig 4

$\sigma(t_2) = \{ X\uparrow X\}$,
$\sigma(t_3) = \{ X\rightarrow X\}$,
$\sigma(t_4) = \{ X\uparrow T\}$,
$\sigma(t_5) = \{ X\downarrow D\}$
$\sigma(t_6) = \{ T \rightarrow C_1, X\rightarrow R, D\rightarrow C_4\}$,
$\sigma(t_7) = \{ T\leftarrow C_2, X\leftarrow L, D\leftarrow C_3\}$,

fig 5

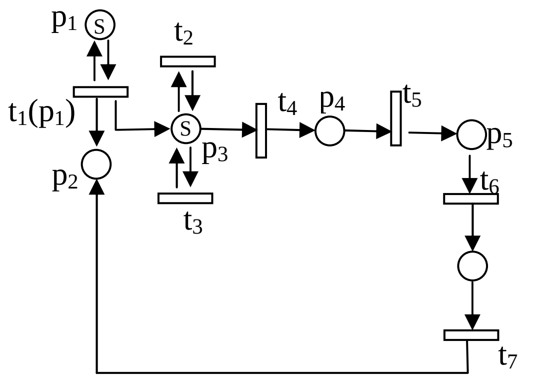


fig 6

The firing sequence t_4 t_5 t_6 t_7 generates the kolam referred to as '"Asanapalakai" is given in fig 7.

$$\begin{matrix} C_2 & T & C_1 \\ L & X & R \\ C_3 & D & C_4 \end{matrix}$$

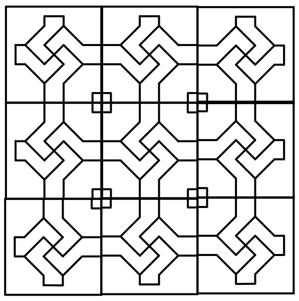


fig 7

Transitions t_2 and t_3 can be fired any number of times before we fire t_4. Firing t_2 and t_3 generates an array made up of the tile X. Then when the sequence t_4 t_5 t_6 t_7 is fired we get all the boundary tiles. This kolam pattern can be generated by a Regular Matrix Grammar [1] and has used only parallel generation.

3. KATPNS with inhibitor arcs

In this section we define KATPNS with inhibitor arcs and give example. When parallel generation is used we can generate only rectangular pictures. To generate triangular or diamond shaped kolams we have to use in Sequential/ Parallel generation. To generate Kolams generated by Context-free or Context-sensitive grammar we have to control the sequence of firing. This control is achieved by introducing inhibitor arcs in the Petri net.

Definition 6. If the Petri net C of a KATPNS has at least one inhibitor arc then we define the KATPNS as Kolam Array Token Petri Net Structure with inhibitor arcs. With arrays of Σ^{**} in some places as initial marking all possible sequences of transitions are fired. The set of all arrays collected in the final places F is called the language generated by C.

Example 2. The KATPNS (C, Σ, σ, F) where the graph of C is given in fig 8, $\Sigma = \{ D, S_1, S_2, S_3, S_4, d_2, d_4\}$, σ the partial function is given in fig 9, the initial marking S in $p_1$and p_3 and F = $\{p_2\}$. The generation here is sequential/parallel. The tiles are given in fig 1.

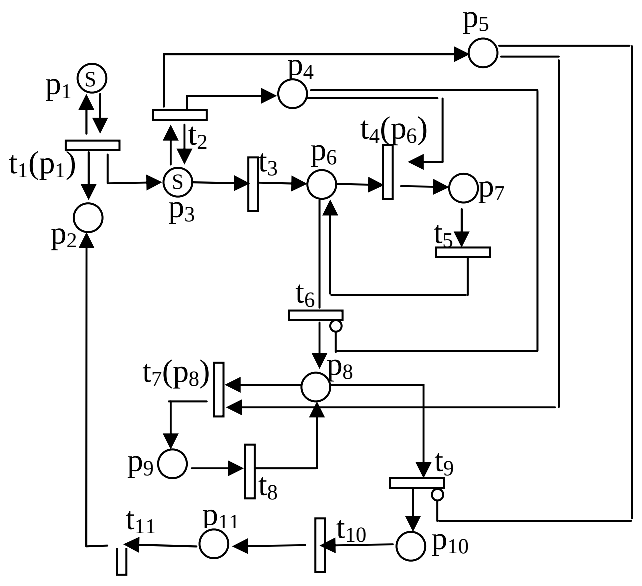


fig 8

$$S = \begin{matrix} D \\ D \end{matrix}$$

$\sigma(t_2) = \{ D\uparrow \begin{smallmatrix} D \\ D \end{smallmatrix} \}$,

$\sigma(t_5) = \{ \text{tm } D \rightarrow S_1, \text{ bm } D \rightarrow S_2, D \rightarrow D \}$,

$\sigma(t_6) = \sigma(t_5)$

$\sigma(t_8) = \{ \text{tm } D \leftarrow S_4, \text{ bm } D \leftarrow S_3, D \leftarrow D \}$,

$\sigma(t_9) = \sigma(t_9)$

$\sigma(t_{10}) = \{ D\uparrow d_4 \}$

$\sigma(t_{11}) = \{ D\downarrow d_2 \}$

fig 9

The firing sequence $t_2\ t_3t_4t_5t_6t_7t_8t_9t_{10}t_{11}$ generates the Kolam pattern in fig 10. To start with t_2 and t_3 are both enabled. If t_2 is fired then the inhibitor input p_4 of t_6 makes sure that the sequence t_4t_5 is also fired the same number of times. Similarly the inhibitor input p_5 of t_9 makes sure that the sequence t_7t_8 is also fired the same number of times. The Kolam patterns generated are of size $(2n + 4) \times (2n +3)/ n \geq 0$, where n is the number of times t_2 is fired. This kolam is context-sensitive [5].

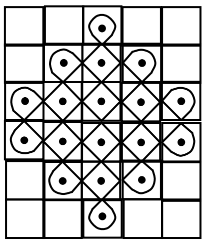


Fig 10: Kambi kolam

4. Comparative Results

In this section we recall the definitions of Array rewriting Grammar [6], compare KATPNS and KATPNS with inhibitor arcs with some of the Kolam array grammars.

Definition 6: $G = (V, I, P, S)$ is an array rewriting grammar(AG), where $V = V_1 \cup V_2$, V_1 a finite set of nonterminals, V_2 a finite set of intermediates, I a finite set of terminals, $P = P_1 \cup P_2 \cup P_3$, P_1 is the finite set of nonterminal rules, P_2 is the finite set of intermediate rules, P_3 is 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), u and v in $(V_1 \cup V_2)^+$ or u and v in $(V_1 \cup V_2)_+$.

P_1 is context-sensitive if there is a (u,v) in P_1 such that $u = u_1S_1v_1$ and $v = u_1\alpha\, v_1$ where $S_1 \in V_1$, u_1, α, v_1 are all in $(V_1 \cup V_2)^+$ or all in $(V_1 \cup V_2)_+$. P_1 is context-free if every (u,v) in P_1 is such that $u \in V_1$ and v in $(V_1 \cup V_2)^+$ or $(V_1 \cup V_2)_+$. P_1 is regular if $u \in V_1$ and v of the form $U \oplus V$, U inV_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 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. P_2 is called CS, CF or R as the intermediate matrix languages generated are CS, CF or R.

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

An Array Grammar is called (CS:CS)AG if the nonterminal rules are CS and at least one intermediate language is CS. An Array Grammar is called (CS:CF)AG if the nonterminal rules are CS and none of the intermediate language is CS. An Array Grammar is called (CS:R)AG if the nonterminal rules are CS and all the intermediate languages are regular. Similarly all the other six families (CF:CS)AG, (CF:CF)AG, (CF:R)AG, (R:CS)AG, (R:CF)AG and (R:R)AG are defined. (X:Y)AL refers to the language generated by the (X:Y)AG, where X, Y $\in$ { R, CF,CS}.

Theorem 4.1. Any Regular Matrix kolam can be generated by KATPNS.

Proof. If the pattern is generated by a regular matrix grammar then it can be generated by (R:R)AG. Any regular array grammar can be generated by a Array Token Petri Net Structure (ATPNS) without inhibitor arcs [10]. The sequence of firing in a KATPNS without inhibitor arcs is in the form $(t_1^n\ t_j^m)^k$. Thus we do not require inhibitor arc to generate this kolam pattern.

Theorem 4.2. Any (CF:R) array kolam can be generated by KATPNS with inhibitor arcs.

Proof. The non terminal rules of (CF:R) array grammar are in the form $(A)^n(B)^n$ or $\begin{matrix}(A)_n\\(B)_n\end{matrix}$. Have a subnet to generate $(A)^n$ and another to generate $(B)^n$. Use inhibitor arcs to make sure both the subnets are executed the same number of times. In example 2, the number of times the sequences t_2, t_4t_5 and t_7t_8 are fired is being controlled by the introduction of the two inhibitor arcs. We can restrict the firing of a transition by introducing an inhibitor arc. Thus (CF:R) array kolam can be generated by KATPNS with inhibitor arcs.

5. Conclusion

Interesting kolam patterns are described by Array grammars. Since Array Token Petri Net Structure is able to generate all these families we have tried to generate kolam patterns with Petri net. KATPNS is able to generate all regular kolams and with the introduction of inhibitor arcs the (CF:R) kolam patterns are also generated.

References

[1] G. Siromoney, R. Siromoney and Kamala Krithivasan, Array Grammars and Kolam, Computer Graphics and Image Processing 3, 1974, pp63 - 82.

[2] G. Siromoney, R. Siromoney and Kamala Krithivasan, Abstract families of Matrices and Picture Languages , Computer Graphics and Image Processing 1, 1972, pp 284 – 307.

[3] K.G.Subramanian, A Note on Regular Kolam Array Grammars Generating Right Triangles, Pattern Recognition Vol. 11, 1979, pp 343 – 345.

[4] K.G. Subramanianan, Rosihan M. Ali, M. Geethalakshmi, Atulya K. Nagar, Pure 2D Picture Grammars And Languages, Discrete Applied Mathematics,157(2009) 3401_3411.

[5] G. Siromoney, R. Siromoney and T. Robinson, “Kambi kolam and cycle grammars”, A Perspective in theoretical computer science, 16, 1989, pp 267-300.

[6] G. Siromoney, R. Siromoney and Kamala Krithivasan, Picture Languages with Array Rewriting Rules, Information and Control 22, pp 447 - 470, 1973.

[7] J .L Peterson, Petri Net Theory an Modeling systems, Prentice Hall, Inc., Englewood Cliffs, N J , 1981.

[8] Shojiro Nagata , Robinson Thamburaj, “ Digitalization Of Kolam Patterns And Tactile Kolam Tools”, Formal Models, Languages And Applications, World Scientific, pp 354-363.

[9] D. Lalitha and K. Rangarajan, “Characterisation Of Pasting System Using Array Token Petri Nets”. International Journal of Pure and Applied Mathematics, Vol 70, No. 3, 2011.

[10] D. Lalitha, K. Rangarajan and D.G. Thomas, “Rectangular Arrays and Petri Nets”, Proceedings of National Conference on Theoretical Computer Science and Discrete Mathematics, Kalasalingam University, Jan 2012.