Application of Array-token Petri nets in generating English Alphabetic Letters

P. Usha[1], Beulah Immanuel [2], R.Sattanathan[3]
[1]Department of Mathematics, D.G.Vaishnav College,
Chennai - 600106.
[2]Department of Mathematics, Women's Christian College,
Chennai - 600006.
[3]Department of Mathematics, D.G.Vaishnav College,
Chennai - 600106.
[1]ushaprab@yahoo.co.in
[2]beulah.immanuel@gmail.com
[3]rsattanathan@gmail.com

Abstract: String-token Petri net which is a variation of coloured Petri net has been introduced in [1] by requiring the tokens to be labeled by strings. Languages in regular and linear families which are two basic classes in the Chomsky hierarchy are generated by these Petri nets [2]. An extension called array - token Petri net, introduced in [5] by labeling tokens by arrays, generates picture languages. Properties related to generative power of array-token Petri net are considered in [3]. In this paper, application of array-token Petri net to generate English alphabetic letters treated as rectangular arrays is examined.

Keywords: Petri net, string-token Petri net, array-token Petri net, picture array language, 2D matrix grammar.

1. Introduction

Petri net introduced by Carl Adam Petri in 1962 has served as a basic model of systems with concurrency and graphically depicts the structure of a distributed system as a directed bipartite graph. As such, a Petri net has place nodes, transition nodes and directed arcs connecting places with transitions but there are no arcs between places and no arcs between transitions. The place from which an arc enters a transition is called the *input place* of the transition; the place to which an arc enters from a transition is called the *output place* of the transition. Places may contain any number of tokens. A distribution of tokens over the places of a net is called a *marking*. Transitions act on input tokens by a process known as *firing*. A transition is *enabled* if it can fire, i.e., there are tokens in every input place of the transition and when a transition fires, tokens are removed from its input places and added at all of the output places of the transition [6].

A coloured Petri net (CPN) has the net structure of a Petri net, and colours are associated with places, transitions and tokens. A transition can fire with respect to each of its colours [4]. A different kind of CPN, called string-token Petri net is

introduced in [1] by labeling the tokens with strings of symbols and the transitions with evolution rules. Firing of a transition removes the token with a string label from the input place and deposits it in the output places of the transition after performing on the string the evolution rule indicated at the transition. This model is examined in [2] for generating regular and linear languages of the Chomsky hierarchy in the study of formal languages.
Syntactic methods of generation and recognition of patterns and pictures have been developed for many years by researchers with different motivations and have been applied in practical problems such as character recognition, two-dimensional mathematical symbols, 3D object recognition and many others. Several two-dimensional grammars which constitute one such area of syntactic methods have been proposed and studied [7]. In particular the 2D matrix grammar introduced by Siromoney et al is a widely investigated class with a number of theoretical as well as application oriented studies having been done [9].
On the other hand, an extension of the string-token Petri net called array- token Petri net is introduced in [5] by labeling tokens by arrays and is used to generate picture languages. It has been shown in [3] that the class of languages generated by array-token Petri nets intersects certain classes of picture languages generated by 2D matrix grammars [3].
In this paper, we consider an application of array-token Petri net by generating English alphabetic letters treated as digitized rectangular arrays.

2. Basic notions

The notions of 2D matrix grammar [7, 9] and Array-token Petri nets [5] are recalled.

Definition 2.1

A 2D matrix grammar is a 2-tuple (G_1, G_2) where $G_1 = (H_1,I_1,P_1,S)$ is a regular, CF or CS grammar; H_1 is a finite set of horizontal non-terminals; $I_1 = \{ S_1,S_2,\ldots, S_k\}$, a finite set of intermediates, $H_1 \cap I_1 = \phi$; P_1 is a finite set of production rules called horizontal production rules; S is the start symbol, $S \varepsilon H_1$; $G_2 = (G_{21}, G_{22},\ldots,G_{2k})$ where $G_{2i} = (V_{2i},T,P_{2i},S_i)$, $1 \leq i \leq k$ are regular grammars; V_{2i} is a finite set of vertical non terminals, $V_{2i} \cap V_{2j} = \phi$, $i \neq j$; T is a finite set of terminals; P_{2i} is a finite set of right-linear production rules of the form $X \rightarrow aY$ or $X \rightarrow a$ where $X,Y \varepsilon V_{2i}, a \varepsilon T$, $S_i \varepsilon V_{2i}$ is the start symbol of G_{2i}.G is a regular, context-free, context sensitive 2D matrix grammar if G_1 is regular, context-free, context sensitive respectively.

The set L(G) of all matrices generated by G consists of all m×n arrays $[a_{ij}]$ such that $1 \leq i \leq m$, $1 \leq j \leq n$ and $S \Rightarrow {}^*_{G1} S_{i1}\ S_{i2} \ldots S_{in} \Rightarrow {}^*_{G2} [a_{ij}]$.
Informally described a derivation in a 2D matrix grammar has two phases. In the first phase a string S of intermediates is generated. In the second phase all symbols of S are rewritten in parallel in the vertical direction by rules of the form $X \rightarrow aY$ or derivation is terminated by rewriting in the vertical direction all symbols by rules of the form $X \rightarrow a$ to yield a rectangular picture array.

We denote the picture language classes of regular, CF, CS, 2D Matrix grammars by RML, CFML, CSML respectively.

Definition 2.2
An Array-token Petri net (ATPN) is a 6-tuple N = (P, T, C, A, R, M_0), where (i) P is a set of places; (ii) T is a set of transitions; (iii) C is set of symbols (colours) and C_{AY} is the set of all rectangular arrays over this colour set C, that are associated with the tokens; (iv) $A \subseteq (P \times T) \cup (T \times P)$ is a set of arcs; (v) R(t) is the set of evolution rules associated with a transition t; (vi) M_0, the initial marking, is a function defined on P such that, for $p \in P$, $M_0(p) \in [C_{AY}]_{MS}$. It is further assumed that there are no isolated places/transitions.

Definition 2.3
An evolution rule over V_{AY}, where V an alphabet, is one of the following: (i) identity , which keeps the array unaltered; (ii) column insertion $\lambda \rightarrow a$ (l/r, according as it is on left or right); (iii) row insertion $\lambda \rightarrow a$ (u/d according as it is up or down); (iv) column deletion $a \rightarrow \lambda$ (l/r, according as it is on left or right); (v) row deletion $a \rightarrow \lambda$ (u/d according as it is up or down); (vi) substitution $a \rightarrow b$ where $a, b \in$ V and λ is the empty word.

The following subnets illustrate how column insertion rules are applied on the left and right of an array A respectively. The rule $\lambda \rightarrow a(l)$, inserts a column of a's to the left of the array A (Fig. 1). The rule $\lambda \rightarrow a(r)$, inserts a column of a's to the right of the array A (Fig. 2).

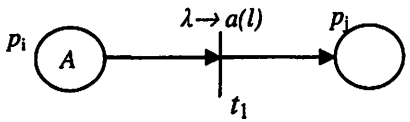

Fig. 1

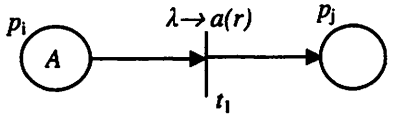

Fig. 2

Similarly, subnets to apply row insertion rules are formed. Replacing insertion by deletion or substitution rules, subnets to apply these rules are formed.

Definition 2.4
A language ATPNL is an L-type ATPN language if there exists a ATPN N= (P, T, C, R, M_0) and a set of final places F such that L = {w /wεM(p)εM is a reachable marking of N, pεF}.
In other words, L-type ATPN language is defined in terms of strings corresponding to some reachable markings in a specified set of final places F, where L is a set of all strings generated in the places for all reachable markings.

We state two results that give a comparison of ATPNL with the 2D matrix languages.

Theorem 2.1[5]
(i) The class of RML intersects the class of ATPNL (ii) The class of strict CFML intersects the class of ATPNL.

Theorem 2.2[3]
There exists an ATPNL that cannot be generated by any CFMG and hence by any RMG.
Proof[3]: The theorem is a consequence of the following observation: (i) The ATPN generated by A_1 (Fig. 3) consisting of 'H' (Fig. 4) shaped arrays of 2m+1 rows and n+2 columns, if t_1 fires n times, t_2 and t_3 fire m times (equal number of times by construction). It is known that 'H' type arrays [9] to be not a CFML and hence not a RML.

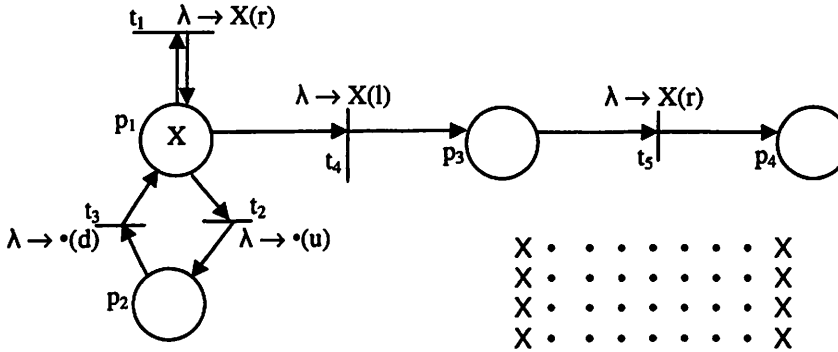

Fig. 3: An ATPN A_1

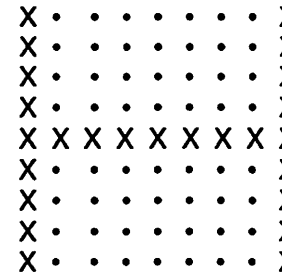

Fig. 4: 'H' shaped array

3. Generation of English Alphabetic Letters by Array-token Petri Nets

In this paper, we consider the case where 'F' consists of only one final place p_f. Some L-type languages generated by ATPNs are character type languages as seen in the following examples. We can get the alphabets in the following examples, only by firing the transitions in the order $t_1, t_2, t_3, t_4, \ldots$
The digitized versions of letters of the English alphabet embedded in a rectangular array background of dots (treated as blanks) are considered and array-token Petri-nets generating these are given.

Generation of Letters B, E, F
The ATPN A_2 (Fig.5) generates the picture language consisting of 'B' (Fig.6) shaped array.

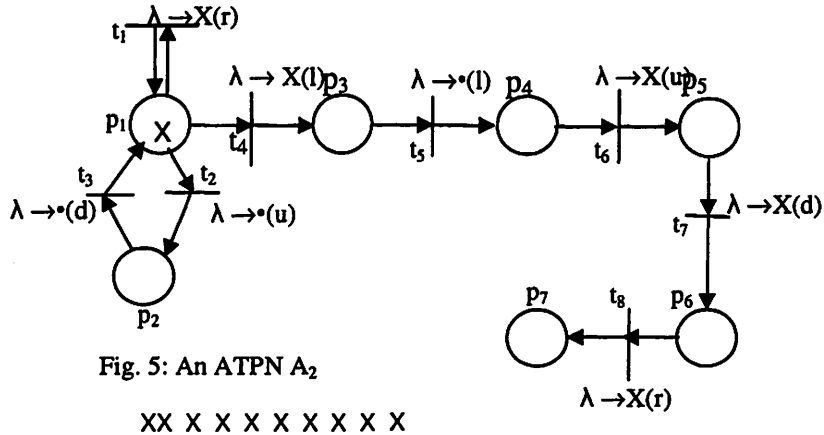

Fig. 5: An ATPN A_2

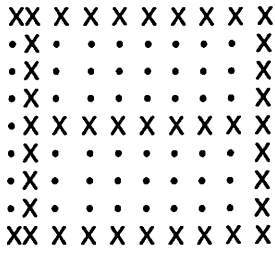

Fig. 6: 'B' shaped array

The ATPN A_2 generates, 'B' shaped array only when t_2 and t_3 fire equal number of times and if t_1 fires n times, t_2 and t_3 fire m times each, the generated 'B' shaped array (Fig.6) consists of 2m+3 rows and n+3 columns. If we (i) delete from ATPN A_2, rule t_8 ($\lambda \rightarrow X(r)$), we can generate letter 'E' (ii) delete t_8 and t_7 ($\lambda \rightarrow X(d)$), we can generate 'F' (iii) replace X by • in p_1, we can generate 'D'.

Generation of Letters L, C, O, U

The ATPN A_3 (Fig. 7) generates the picture language consisting of 'L' (Fig. 8) shaped array.

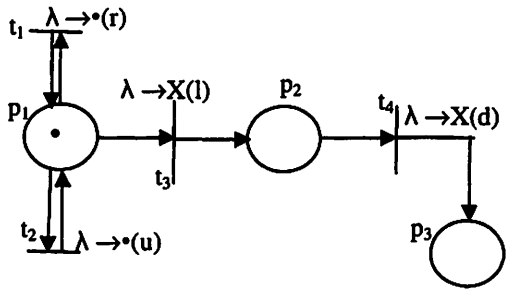

Fig. 7: An ATPN A_3

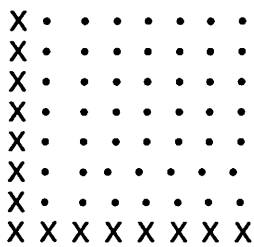

Fig. 8: 'L' shaped array

In ATPN A_3, if t_1 and t_2 fire m and n times respectively, then the generated 'L' shaped array (Fig.8) consists of m+1 rows and n+1 columns, and if we (i) inserting the rule $\lambda \rightarrow X(u)$ as t_5, we obtain 'C' (ii) inserting $\lambda \rightarrow X(u)$ as t_5 and $\lambda \rightarrow X(r)$ as t_6, we obtain 'O'. (iii) inserting $\lambda \rightarrow X(r)$ as t_5, we obtain 'U'.

Generation of Letters T, I

The ATPN A_4 (Fig.9) generates the picture language consisting of 'T' (Fig.10) shaped array.

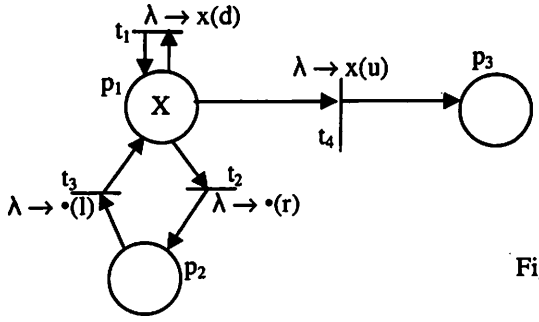

Fig. 9: An ATPN A_4

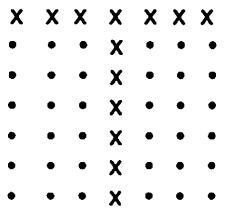

Fig. 10: 'T' shaped array

The ATPN A_4 generates, 'T' shaped array, only when t_2 and t_3 fire equal number of times. If t_1 fires m times, t_2 and t_3 fires n times(equal number of times by construction), then 'T' shaped array in Fig. 10 consists of m+1 rows and 2n+1 columns, and if we inserting as t_5, $\lambda \rightarrow X(d)$, we get 'I'.

Generation of Letter J

The ATPN A_5 (Fig.11) generates the picture language consisting of 'J' (Fig.12) shaped array.

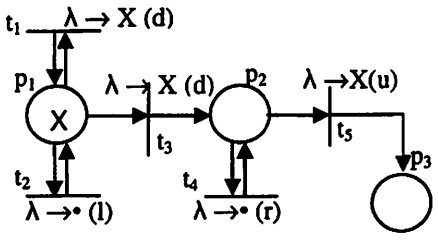

Fig. 11: An ATPN A_5

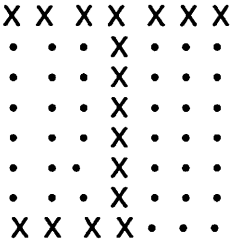

Fig. 12: 'J' shaped array

The ATPN A_5 generates, 'J' shaped array, only when t_2 and t_4 fire equal number of times. If t_1 fires m times, t_2 and t_4 fire n times (equal number of times by

construction), then 'J' shaped array in Fig. 12 consists of m+2 rows and 2n+1 columns.

Generation of Letter P

The ATPN A_6 (Fig. 13) generates the picture language consisting of 'P' (Fig. 14) shaped array.

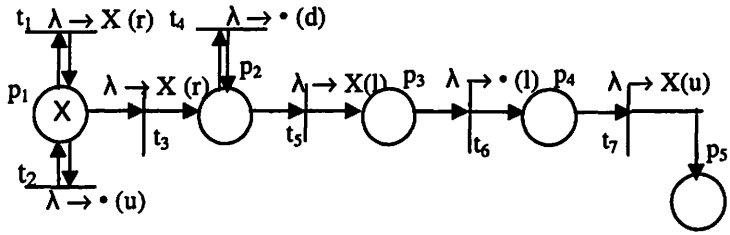

Fig. 13: An ATPN A_6

XX X X X X X X X
•X • • • • • • X
•X • • • • • • X
•X • • • • • • X
•X X X X X X X X
•X • • • • • • •
•X • • • • • • •
•X • • • • • • •
•X • • • • • • •
•X • • • • • • •

Fig. 14: 'P' shaped array

The ATPN A_6 generates, 'P' shaped array, only when the number of times t_2 fires is less than that of t_4 firing. If t_1 fires n times, t_2 and t_4 fire m and r times (m < r) respectively, then 'P' shaped array in Fig. 14 consists of m+r+2 rows and n+3 columns.

Generation of Letter G

The ATPN A_7 (Fig.15) generates the picture language consisting of 'G' (Fig.16) shaped array.

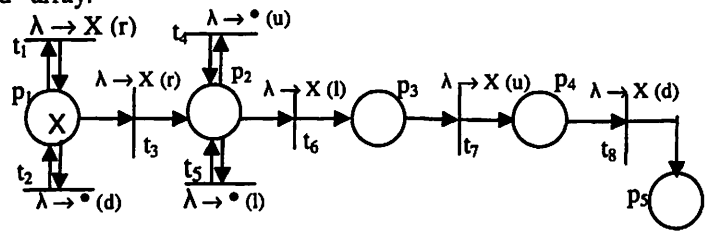

Fig. 15: An ATPN A_7

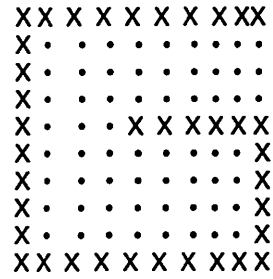

Fig. 16: 'G' shaped array

In ATPN A_7, if t_1 fires n times, t_2 and t_4 fire m and r times respectively, and t_5 fires t times, then 'G' shaped array in Fig. 16 consists m+r+3 rows and n+t+2 columns.

4. Conclusion

In this paper, we have generated some of the alphabetic letters in the form of digitized rectangular arrays using array - token Petri nets. The generation of other letters such as A, K, W, Z would require further effort to define rules to generate diagonals of x's. This question is to be addressed in future work.

5. References

[1] B. Immanuel, K. Rangarajan, K. G. Subramanian, *"String-token Petri nets"*, Proceedings of the European Conference on Artificial Intelligence, One-day Workshop on Symbolic Networks, at Valencia, Spain, 2004.

[2] B. Immanuel, K.G. Subramanian, A. Roslin Sagaya Mary, *"Petri nets with String-labeled Tokens"*, Proceedings of the 2nd International Conference on Cybernetics and Information Technologies Systems and Applications, Orlando, Florida, USA, 2005.

[3] B. Immanuel, K.G. Subramanian, P. Usha, *"Array token Petri nets and Character Generation"*, Proceedings of National Conference on Computational Mathematics and Soft Computing, Women's Christian College, 2009.

[4] K. Jensen, *"Coloured Petri nets"*, Lecture Notes in Computer Science, 254 (248-299),1987.

[5] S. Kannamma, K. Rangarajan, D.G. Thomas, N.G. David, *"Array token Petri nets, Computing and Mathematical Modeling"*, Narosa Publishing House, New Delhi, India, (299-306), 2006.

[6] J. I. Peterson, *"Petri net Theory and The Modeling of systems"*, Prentice Hall, Englewood Cliffs, N.J., 1981.

[7] A. Rosenfeld and R. Siromoney, *"Picture languages – a survey, languages of design"*, 1, (229-245), 1993.

[8] A. Salomaa, *"Formal languages"*, Academic Press, 1973.

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