Int. J. Math. And Appl., 6(2–B)(2018), 295–304
ISSN: 2347-1557
Available Online: http://ijmaa.in/

International Journal of *Mathematics And its Applications*

Context-free Array-token Petri Nets

D. K. Shirley Gloria[1,*] and G. Albert Asirvatham[2]

1 Department of Mathematics, Dr. Ambedkar Government Arts College, Vyasarpadi, Chennai, Tamilnadu, India.

2 Department of Mathematics, Thiru. A. Govindaswami Government Arts College, Tindivanam, Tamilnadu, India.

Abstract: Any pure two dimensional context - free Language can be generated by Array - Token Petri Nets. Some of the closure properties are obtained.

Keywords: Array-token Petri net (ATPN), context-free grammar (CFG), pure two dimensional context-free grammar (P2DCFG), pure two dimensional context-free language (P2DCFL).

Accepted on: 11.07.2018

1. Introduction

Petri Net has its origin in Carl Adam Petri's dissertation submitted in the year 1962. Tokens are used in these nets to simulate the dynamic and concurrent activities of systems [4]. Petri net is a bipartite graph with place nodes, transition nodes and directed arcs connecting places with transitions. Places are connected only by transitions. Similarly, transitions are connected only by places. ie, an arc can be drawn from place to transition or transition to place. The place from which an arc enters a transition is called the input place of the transition and the place to which an arc enters from a transition is called the output place of the transition. Any number of tokens are given on places. 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. An enabled transition can fire. ie, if there are tokens in every input place of the transition, then a transition fires. In that case tokens are removed from the input place of the transition and added at all of its output places [4].

A coloured Petri Net (CPN) has the net structure of a Petri Net and colours are assosiated with places, transitions and tokens. A transition can fire with respect to each of its colours [1]. A different kind of CPN, called string - token Petri Net was introduced in [1] by labeling the tokens with strings of symbols and the transition with evolution rules [6]. Firing of a transition removes token with a string label from the input places and deposits it in the output places of the transition after performing on the strings, the evolution rule indicated at the transition. In [9], a new two-dimensional 2D grammar based on pure Context-free rules, called pure 2D context-free grammar (P2DCFG), for rectangular picture array generation is introduced. In this 2D model, any column or any row of the rectangular array rewritten without any priority of rewriting columns and rows as in [7] and [8]. Certain closure properties of this 2D model were also obtained.

On the other hand an extension of the string-token Petri net called array-token Petri net is introduced in [2] and [5] by labeling tokens by arrays and is used to generate picture languages. It has been shown in [3] that the class of languages

* *E-mail: shirleygloria1976@gmail.com*

generated by array-token Petri nets intersects certain classes of picture languages generated by 2D matrix grammars. Also in [2], the languages generated by these Array-token Petri nets were compared with the languages generated by 2D grammars. In this paper, the languages generated by P2DCFG are compared with the languages generated by these ATPN's. Some of it's closure properties are discussed. These are illustrated by examples.

2. Preliminaries

Definition 2.1. *A pure 2D context-free grammar(P2DCFG) is a 4-tuple* $G = (\Sigma, P_c, P_r, A_0)$ *where*

- Σ *is a finite set of symbols.*
- $P_c = \{t_{c_i} | 1 \leq i \leq m\}, P_r = \{t_{r_j} | 1 \leq j \leq n\}$*. Each* $t_{c_i}, (1 \leq i \leq m)$*, called a column table, is a set of context-free rules of the form* $a \to \alpha, a \in \Sigma, \alpha \in \Sigma^*$ *such that for any two rules* $a \to \alpha, b \to \beta$ *in* t_{c_i}*, we have* $|\alpha| = |\beta|$ *where* $|\alpha|$ *denotes the length of* α*. Each* t_{r_j}*,* $(1 \leq j \leq n)$*, called a row table, is a set of context free rules of the form* $c \to \gamma^T$*,* $c \in \Sigma$ *and* $\gamma \in \Sigma^*$ *such that for any two rules* $c \to \gamma^T$*,* $d \to \delta^T$ *in* t_{r_j}*, we have* $|\gamma| = |\delta|$*.*
- $A_0 \subseteq \Sigma^{**} - \{\lambda\}$ *is a finite set of axiom arrays.*

Definition 2.2. *Evolution rules R(t) which are used in Array-Token Petri Net (ATPN) are as follows:*

(a). I is the identity rule that keeps the array unaltered (for example A→ A is an identity rule, where A denotes an array).

(b). $\lambda \to A(l)$ *is the left insertion rule,*

$\lambda \to A(r)$ *is the right insertion rule,*

$\lambda \to A(u)$ *is the up (above) insertion rule,*

$\lambda \to A(d)$ *is the down insertion rule, where* λ *is the empty array.*

(c). $A \to \lambda(l)$ *is the left deletion rule,*

$A \to \lambda(r)$ *is the right deletion rule,*

$A \to \lambda(u)$ *is the deletion above (up) rule,*

$A \to \lambda(d)$ *is the deletion below (down) rule.*

(d). $A \to B$ *is the substitution rule where A and B are arrays*

(array A is replaced by array B).

Definition 2.3. *An Array - Token Petri Net (ATPN) is a 7-tuple* $N = (Q, T, V, A, R(t), F, M_0)$ *where*

- $Q = \{q_1, q_2, ..., q_n\}$ *is a finite set of places.*
- $T = \{t_1, t_2, ..., t_m\}$ *is a finite set of transitions and each* t_i *is the label of evolution rules.*
- V *is a finite non-empty set of arrays.*
- $A \subseteq (QXT) \cup (TXQ)$ *is a set of arcs (flow relation).*
- $R(t)$ *is the set of evolution rules associated with each transition* t *of* T*.*
- F *is a set of final places that is places which are not having output arcs or input places of transition which are not enabled.*
- $M_0 : Q \to$ *(an array over* V*) is the initial marking.*
- $Q \cup T \neq \phi$*;* $Q \cap T = \phi$*.*

Definition 2.4. *In order to simulate the dynamic behaviour of a system, a state or marking in an ATPN is changed according to the following transition (firing) rule:*

(i). A transition t is said to be ***enabled*** *if each input place p of t consists of an array or part of an array, called sub array with a left side expression of the transition rule. Suppose*

$t : \begin{matrix} a & b \\ a & c \end{matrix} \rightarrow \begin{matrix} a & a & b \\ b & b & d \end{matrix}$ *then* $\begin{matrix} a & b \\ a & c \end{matrix}$ *should be there in input place q of t.*

(ii). An enabled transition may fire.

(iii). A firing of an enabled transition t removes an array or sub array say $\begin{matrix} X & X & X \\ a & b & X \\ a & c & X \\ X & X & X \end{matrix}$ *from its input place q of t and adds an array say* $\begin{matrix} X & X & X & X \\ a & a & b & X \\ b & b & d & X \\ X & X & X & X \end{matrix}$ *on the output place of t.*

Definition 2.5. *A language L is an ATPN language if there exists an ATPN* $N = (Q, T, V, A, R(t), F, M_0)$ *such that* $L = \{A/A \in M(Q), M \text{ is a reachable marking of } N, q \in F\}$

Example 2.6. *Consider the pure 2D context-free grammar*

$G_1 = (\Sigma_1, P_{c_1}, P_{r_1}, A_0)$ *where*

$\Sigma_1 = \{a, b, c, \bullet\}$, $P_{c_1} = \{t_{c_1}\}$, $P_{r_1} = \{t_{r_1}\}$,

$t_{c_1} = \{\bullet \rightarrow \bullet\bullet, b \rightarrow bb\}$, $t_{r_1} = \{b \rightarrow \begin{matrix} \bullet \\ b \\ \bullet \end{matrix}, c \rightarrow \begin{matrix} a \\ c \\ a \end{matrix}\}$, $A_0 = \begin{matrix} a & \bullet & a \\ c & b & c \\ a & \bullet & a \end{matrix}$

An ATPN that generates G_1 *is given below (see Figure 1):*

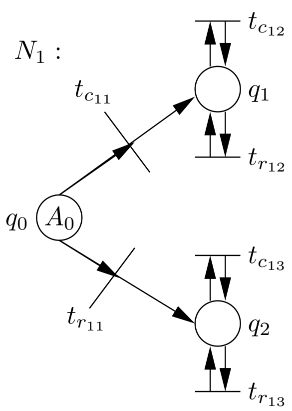


Figure 1.

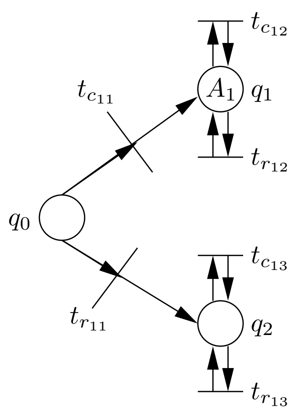


Figure 2.

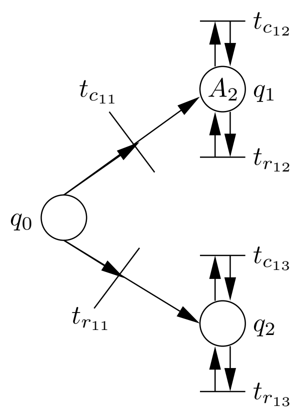
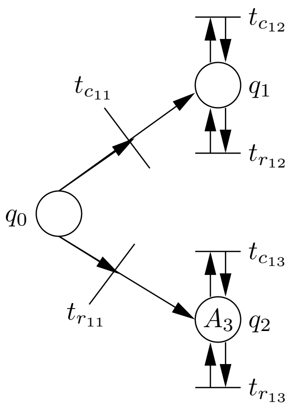
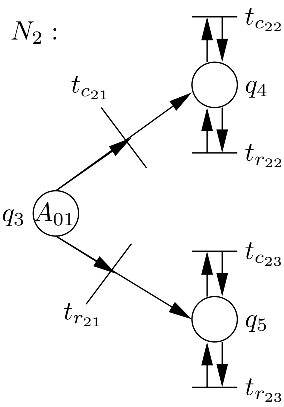


Figure 3.

$t_{c_{11}}$ *stands for* 1^{st} *transition with* t_{c_1} *as evolution rule,* $t_{c_{12}}$ *stands for* 2^{nd} *transition with* t_{c_1} *as evolution rule and so on.*





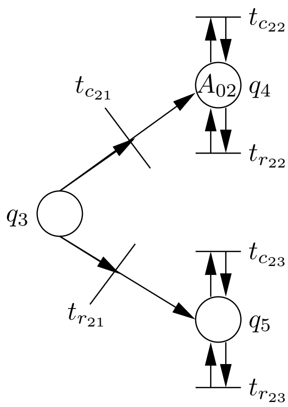
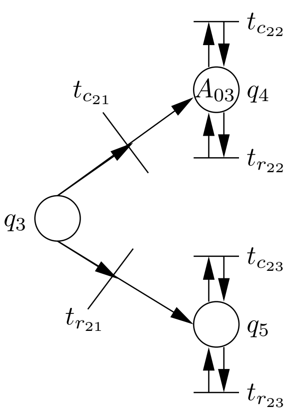


Figure 4. **Figure 5.** **Figure 6.**

In Figure 2, when $t_{c_{11}}$ fires, we would get A_1 on q_1 where $A_1 = \begin{array}{cccc} a & \bullet & \bullet & a \\ c & b & b & c \\ a & \bullet & \bullet & a \end{array}$. *In Figure 3, when $t_{r_{12}}$ fires, we would get A_2 on q_1 where* $A_2 = \begin{array}{cccc} a & \bullet & \bullet & a \\ a & \bullet & \bullet & a \\ c & b & b & c \\ a & \bullet & \bullet & a \\ a & \bullet & \bullet & a \end{array}$. *In Figure 4, when $t_{r_{11}}$ fires, we would get A_3 on q_2 where* $A_3 = \begin{array}{ccc} a & \bullet & a \\ a & \bullet & a \\ c & b & c \\ a & \bullet & a \\ a & \bullet & a \end{array}$ *and so on.*



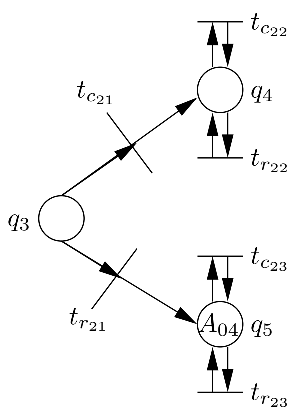


Figure 7. **Figure 8.**

Example 2.7. *Consider the P2DCFG* $G_2 = (\Sigma_2, P_{c_2}, P_{r_2}, A_{01})$, *where* $\Sigma_2 = \{a, b, c, \bullet\}$, $P_{c_2} = \{t_{c_2}\}$, $P_{r_2} = \{t_{r_2}\}$, $t_{c_2} = \{b \to a\ b\ a, c \to \bullet\ c\ \bullet\}$, $t_{r_2} = \left\{ a \to \begin{array}{c} a \\ \bullet \end{array}, b \to \begin{array}{c} b \\ c \end{array} \right\}$,

$A_{01} = \begin{array}{ccc} a & b & a \\ \bullet & c & \bullet \end{array}$. *Here, $t_{r_{23}}$ stands for 3^{rd} transition with t_{r_2} as evolution rule, $t_{c_{22}}$ stands for 2^{nd} transition with t_{c_2} as evolution rule and so on (see Figure 5). In Figure 6, when $t_{c_{21}}$ fires, we would get A_{02} on q_4 where* $A_{02} = \begin{array}{ccccc} a & a & b & a & a \\ \bullet & \bullet & c & \bullet & \bullet \end{array}$. *In Figure 7, when $t_{c_{22}}$ fires, we would get A_{03} on q_4 where* $A_{03} = \begin{array}{ccccccc} a & a & a & b & a & a & a \\ \bullet & \bullet & \bullet & c & \bullet & \bullet & \bullet \end{array}$. *In Figure 8, when $t_{r_{21}}$ fires, we would* *get* A_{04} *on* q_5 *where* $A_{04} = \begin{matrix} a & b & a \\ \bullet & c & \bullet \\ \bullet & c & \bullet \end{matrix}$ *and so on.*

Theorem 2.8. *If L is a P2DCFL, then there exists an ATPN 'N' such that L is generated by ATPN.*

Proof. Let $G = (\Sigma, P_c, P_r, A_0)$ be a P2DCFG that generates L.

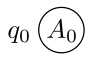


Figure 9.

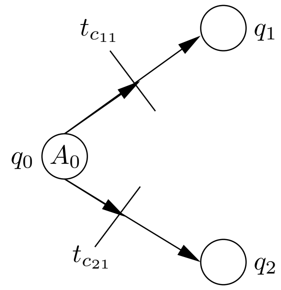


Figure 10.

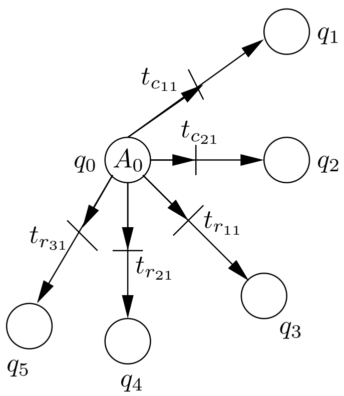


Figure 11.

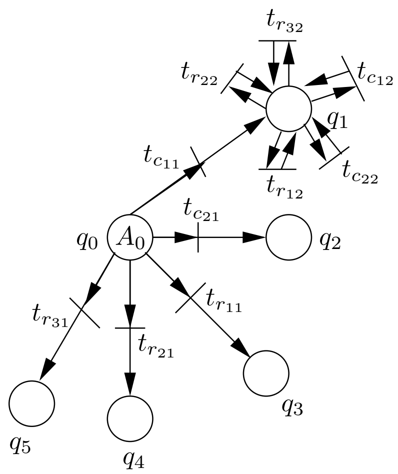


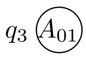
Figure 12.

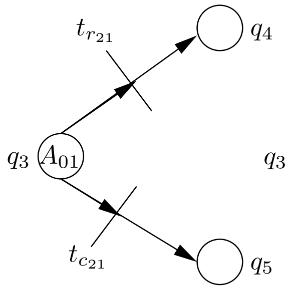
We construct an ATPN that generates L as follows:

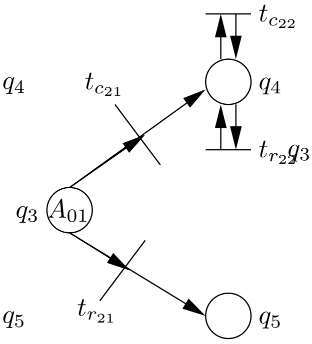
We start with a place q_0 with A_0 as the initial array token (see Figure 9). For every t_{c_i} , $1 \leq i \leq m$, and t_{r_j} , $1 \leq j \leq n$, construct transitions with $t_{c_{i_1}}$, $t_{r_{j_1}}$, $t_{c_{i_2}}$, $t_{r_{j_2}}$, and so on as evolution rules. Let $m = 2$, $n = 3$. ie, Assume that t_{c_1}, t_{c_2}, t_{r_1}, t_{r_2}, t_{r_3} are given in P_c and P_r. Now construct $t_{c_{11}}$, $t_{c_{12}}$, $\ldots$, $t_{r_{11}}$, $t_{r_{12}}$, $\ldots$, $t_{c_{21}}$, $t_{c_{22}}$, $\ldots$, $t_{r_{21}}$, $t_{r_{22}}$, $\ldots$, $t_{r_{31}}$, $t_{r_{32}}$, $\ldots$ as evolution rules. From q_0, attach transitions $t_{c_{11}}$, $t_{c_{21}}$, $t_{r_{11}}$, $t_{r_{21}}$, $t_{r_{31}}$. From these transitions attach places q_1, q_2, q_3, q_4, q_5 (see Figures 10 and 11). We attach transitions to q_1 that gives loop structure. Let $t_{c_{12}}, t_{c_{22}}, t_{r_{12}}, t_{r_{22}}, t_{r_{32}}$ be the transitions that gives loop structure. So, attach these transitions to q_1 (see Figure 12). Likewise on q_2, q_3, q_4, q_5, we attach transitions with loop structure. Hence, the resulting ATPN 'N' will generate given L. □

Example 2.9. *Consider the P2DCFG as in Example 7. Since* A_{01} *is the axiom array, we construct a place* q_3 *with* A_{01} *as initial token array (see Figure 13). Now from* q_3*, attach transitions for* t_{c_2} *and* t_{r_2}*. We call them* $t_{c_{21}}$ *and* $t_{r_{21}}$ *because* $t_{c_{21}}$ *stands for first transition with* t_{c_2} *as evolution rule and* $t_{r_{21}}$ *stands for first transition with* t_{r_2} *as evolution rule. Connect transitions* $t_{c_{21}}$ *and* $t_{r_{21}}$ *to the places* q_4 *and* q_5 *(see Figure 14). We can see that* t_{c_2} *and* t_{r_2} *will give a loop. From* q_4 *attach*

t_{c22} and t_{r22} that gives loop structure. Similarly, from q_5 attach t_{c23} and t_{r23} (see Figures 15 and 16). Now the resulting net will be $N = (\{q_3, q_4, q_5\}, \{t_{c21}, t_{c22}, t_{c23}, t_{r21}, t_{r22}, t_{r23}\}, A, \{t_{c_2}, t_{r_2}\}, F, A_{01})$. We can see that N generates P2DCFL L.







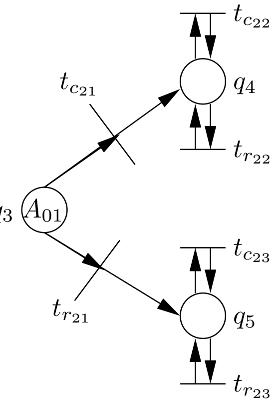


Figure 13. **Figure 14.** **Figure 15.** **Figure 16.**

Theorem 2.10. *The class of ATPN's that generates P2DCFL is closed under union.*

Proof. Let N_1 be an ATPN that generates a P2DCFL L_1 and N_2 be an ATPN that generates a P2DCFL L_2. Now, we can construct an ATPN that generates $L_1 \cup L_2$ as follows:

Let q_s be the place with an array λ (any empty array of desirable size). Delete the axiom arrays of N_1 and N_2. Attach a transition from q_s with evolution rule $\lambda \to A_0$ that connects to N_1 where A_0 is the initial array of N_1. Similarly from q_s , attach a transition with evolution rule $\lambda \to A_1$ that connects to N_2 where A_1 is the initial array of N_2. Now, N will generate $L_1 \cup L_2$ (see Figure 17).

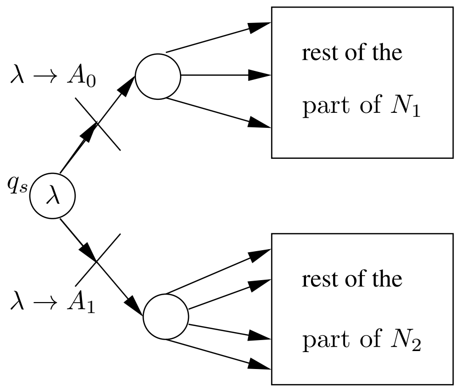


Figure 17.

□

Example 2.11. *Consider ATPNs as in Examples 6 and 7. Start with the place q_s consisting of an empty array λ. Delete A_0 and A_{01} from ATPNs in Examples 6 and 7. Attach q_s to the transition with the evolution rule $\lambda \to A_0$. Connect this transition to initial place of N_1. Similarly attach q_s to the transition with the evolution rule $\lambda \to A_{01}$ and connect this transition to the initial place of N_2. Rest of the operations in N_1 and N_2 are same. Now, we obtain a net that generates $L_1 \cup L_2$ (see Figure 18).*

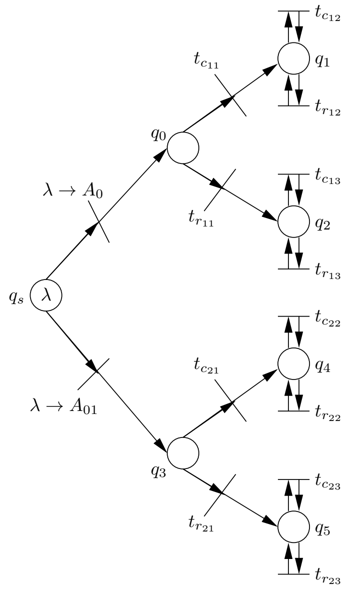


Figure 18.

Theorem 2.12. *The class of ATPN's that generate P2DCFL is closed under column catenation.*

Proof. Let N_1 be the ATPN generated by P2DCFL L_1 and N_2 be the ATPN generated by P2DCFL L_2. Let the symbol $\bigcirc\!\!\!\!\mid$ denote column catenation. Here, L_1 and L_2 must have same number of rows. Then $L_1 \bigcirc\!\!\!\!\mid L_2$ can be generated by ATPN N as follows (see Figure 19):

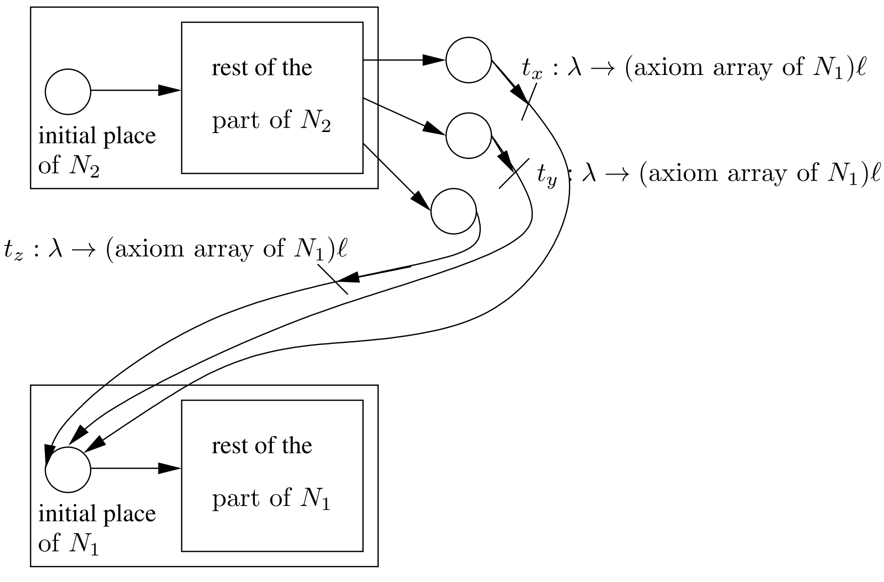


Figure 19.

Consider the ATPN N_2. From the final places of ATPN N_2, attach transitions with the evolution rule $\lambda \to$ (axiom array of N_1)ℓ and delete the axiom array of N_1 from N_1. Connect final places of N_2 with the initial place of N_1 through the transitions with the evolution rule $\lambda \to$ (axiom array of N_1) ℓ. Here, $\lambda \to$ (axiom array of N_1) ℓ means left insertion as in Definition 2. Rest of the operations are done as in N_1 and N_2 (see Figure 19). Similarly, $L_2 \bigcirc\!\!\!\!\mid L_1$ can be done by taking N_1 first and then N_2. □

Example 2.13. *Consider ATPN's as in Examples 6 and 7. Here, number of rows should be the same. Since the number* *of rows should be the same, we consider* A_{01} *to be* $A_{01} = \begin{array}{ccc} a & b & a \\ \bullet & c & \bullet \\ \bullet & c & \bullet \end{array}$. *Rest of the operations of* N_2 *are the same. Connect final places of* N_2 *to the transition with the evolution rule* $\lambda \to$ *(axiom array of* N_1*)*ℓ*. i.e.,* $\lambda \to A_0\ell$*. Delete the axiom array of* N_1 *from its initial place. Now connect the transitions with the evolution rule* $\lambda \to A_0\ell$ *to the initial place of* N_1 *(see Figure 20). Now, after completing all possible firings, the resulting array will be of the form* $L_1 \obar L_2$*.*

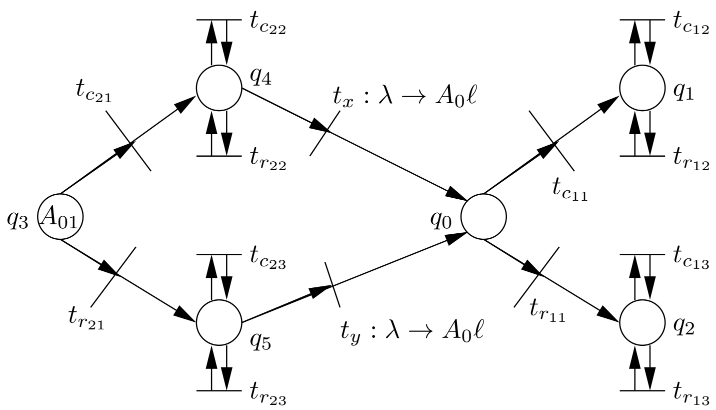


Figure 20.

Theorem 2.14. *The class of ATPN's that generate P2DCFL is closed under row catenation.*

Proof. Let N_1 be the ATPN generated by P2DCFL L_1 and N_2 be the ATPN generated by P2DCFL L_2. Let the symbol $\ominus$ denote row catenation. Here, L_1 and L_2 must have same number of columns. Then $L_1 \ominus L_2$ can be generated by ATPN N as follows (see Figure 21): Consider the ATPN N_2. From the final places of ATPN N_2, attach transitions with evolution rule $\lambda \to$ (axiom array of N_1)u and delete the axiom array of N_1 from N_1. Connect final places of N_2 with the initial places of N_1 through the transitions with evolution rule $\lambda \to$ (axiom array of N_1)u. Here, $\lambda \to$ (axiom array of N_1)u means above (up) insertion as in Definition 2. Rest of the operations are done as in N_1 and N_2 (see Figure 21).

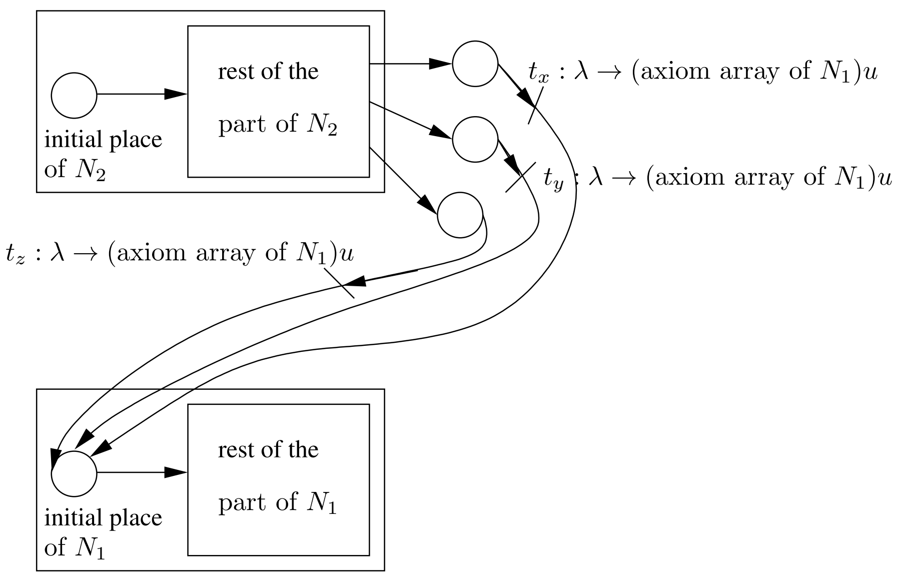


Figure 21.

□

Example 2.15. *Consider ATPNs as in Examples 6 and 7. Here the number of column should be the same. Delete axiom array of* N_1 *from* N_1*. Consider first ATPN* N_2*. From the final places of* N_2 *attach transition with evolution rule* $\lambda \to A_0u$*. Connect final places of* N_2 *with the initial place of* N_1 *through the transitions with evolution rule* $\lambda \to A_0u$*. Now,* $\lambda \to A_0u$ *means up insertion as in Definition 2. Rest of the operations are done as in* N_1 *and* N_2 *(see Figure 22).*

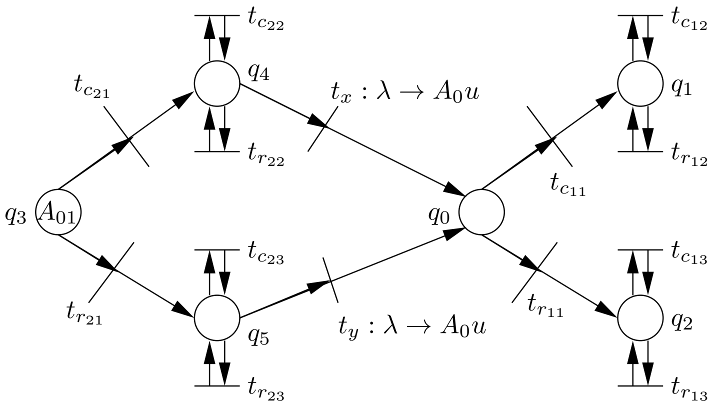


Figure 22.

Theorem 2.16. *The class of ATPN's that generate P2DCFL is closed under transposition. ie, If P2DCFL L is generated by ATPN* N_3*, then there exists ATPN* N_4 *such that* N_4 *generates* L^T*.*

Proof. Let N_3 be the ATPN that generates L. L^T is the transposition of L which can be generated by ATPN N_4 as follows. Column tables of L is taken as row tables of L^T and row tables in L is taken as column tables of L^T. Suppose $A \to \alpha$ is in a column table of L, then take $A \to \alpha^T, \alpha \in \Sigma^{**}$ as corresponding row table of L^T. Likewise, for a rule $B \to \beta^T$ in a row table of L, the rule $B \to \beta$ is taken as the corresponding column table of L^T. The resulting N_4 will generate L^T (see Figures 23 and 24).

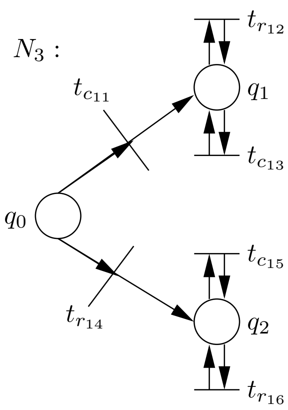


Figure 23.

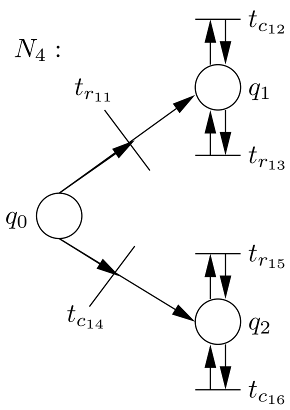


Figure 24.

□

Example 2.17. *Consider ATPN as in Example 6. Now* $\bullet \to \bullet\bullet$*,* $b \to bb$ *are in column table of L, that is in* t_{c_1}*, we take* $\bullet \to \begin{matrix}\bullet\\ \bullet\end{matrix}$ *and* $b \to \begin{matrix}b\\ b\end{matrix}$ *as corresponding row table of* L^T*. We call it as* t'_{r_1} *where* $t'_{r_1} = \left\{ \bullet \to \begin{matrix}\bullet\\ \bullet\end{matrix}, b \to \begin{matrix}b\\ b\end{matrix} \right\}$*. In a row table of*

L, that is in t_{r_1}*, we have* $b \to \begin{matrix} \bullet \\ b \\ \bullet \end{matrix}$, $c \to \begin{matrix} a \\ c \\ a \end{matrix}$ *convert it as* $b \to \bullet b \bullet$, $c \to aca$ *in corresponding column table of* L^T*. We call it as* t'_{c_1} *where* $t'_{c_1} = \{b \to \bullet b \bullet, c \to aca\}$ *(see Figure 25).*

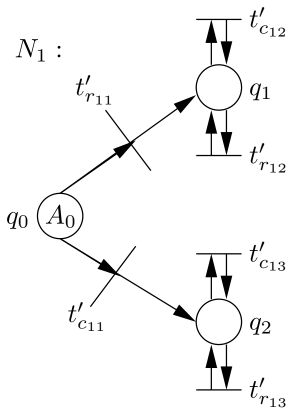


Figure 25.

3. Conclusion

Thus, it can be concluded that every Pure 2D Context-Free Language can be generated by Array Token Petri Net. Also, the class of Array Token Petri Nets that generates Pure 2D Context-Free Languages is closed with respect to union, column catenation, row concatenation and transposition.

References

[1] Beulah Immanuel, K. Rangarajan and K.G. Subramanian, *String-token petri nets*, Proceedings of european conference on artificial intelligence: one day workshop on symbolic networks at valencia, Spain, (2004).

[2] Beulah Immanuel and P. Usha, *Array token petri nets and 2D grammars*, International Conference on Mathematical Computer Engineering-ICMCE 2013, (2013), 722-727.

[3] Beulah Immanuel, K.G. Subramanian and P. Usha, *Array token petri nets and character generation*, Proceedings of National Conference on Computational Mathematics and Soft Computing, (2009).

[4] James Peterson, *Petrinet theory and modeling of systems*, Prentice Hall, USA, (1997).

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

[6] D. K. Shirely Gloria, Beulah Immanuel and K. Rangarajan, *Parallel context-free string-token petrinets*, International Journal of Pure and Applied Mathematics, 59(2010), 275.

[7] G. Siromoney, R. Siromoney and K. Krithivasan, *Abstract families of matrices and picture languages*, Compt. Graph. Image Process, 1(1972), 234-307.

[8] R. Siromoney, K.G. Subramanian and K. Rangarajan, *Parallel sequential rectangular arrays with tables*, Int. J. Comput. Math., 6A(1977), 143-158.

[9] K. G. Subramanian, Roshian M. Ali, M. Geethalakshmi and Atulya K. Nagar, *Pure 2D picture grammars and languages*, Discrete Applied Mathematics, 157(2009), 3401-3411.