Int J Syst Assur Eng Manag (June 2018) 9(3):630–638
https://doi.org/10.1007/s13198-014-0304-8

ORIGINAL ARTICLE

A formal study on generative power of a class of array token Petri net structure

T. Kamaraj · D. Lalitha · D. G. Thomas

Received: 9 August 2014 / Published online: 24 October 2014

Abstract Adjunct array token Petri net structure (AATPNS) is a recently introduced rectangular picture generating device which extends Array token Petri net structure. AATPNS generates some of the classical classes of picture languages namely Siromoney matrix languages and kolam array languages, and a recent, pure 2D context-free languages. In this paper we concentrate our attention on determining the generative power of AATPNS by comparing it with some expressive rectangular picture grammar models and also with the classes of languages recognizable by tiling systems. We also study certain closure properties with respect to basic array operations.

Keywords Petri nets · Array tokens · Adjunction · Pure 2D grammars · Picture languages

Mathematics Subject Classification 68Q85 · 42 · 45 · 10 · 15

This work is an extension of a preliminary paper (Kamaraj et al. 2014).

T. Kamaraj (✉) · D. Lalitha
Department of Mathematics, Sathyabama University, Chennai 600119, India
e-mail: kamaraj_mx@yahoo.co.in

D. Lalitha
e-mail: lalkrish_24@yahoo.co.in

D. G. Thomas
Department of Mathematics, Madras Christian College, Chennai 600059, India
e-mail: dgthomasmcc@yahoo.com

1 Introduction

Since seventies, the study of two dimensional languages generated by grammars or recognized by Automata have been found in the theory of formal languages with the motivation of picture processing and pattern recognition tasks (Giammarresi and Restivo 1997; Rosenfeld and Siromoney 1993; Siromoney 1987). With the quest of syntactic techniques on digital picture patterns, many array generating grammar devices have been proposed. Siromoney matrix grammars (SMG; Siromoney et al. 1972), controlled table L array grammars (T0LG/T1LG; Siromoney and Siromoney 1977), kolam array grammars (KAG; Siromoney et al. 1973, 1974) are some of the classical generating devices, which used sequential and parallel application of rewriting rules. Pure 2D context-free grammars (P2DCFG; Subramanian et al. 2009) and parallel contextual array grammars (Subramanian et al. 2008a, b) make use of only terminal symbols as in pure string grammars. Prusa grammars (PG; 2004) tile rewriting grammars (TRG) and regional tile rewriting (RTG) grammars (Crespi-Reghizzi and Pradella 2005; Cherubini et al. 2006; Pradella et al. 2011) are some of the more expressive context-free grammars. Tiling system (TS; Giammarresi and Restivo 1997; Cherubini et al. 2006) is a recognizing device for the class of recognizable picture languages (REC), which involves the projection of languages belonging to the class of local picture languages (LOC) defined by a finite set of tiles.

Recently another picture generating mechanism, array token Petri net structure (ATPNS; Lalitha et al. 2012b) has been evolved from string generating Petri nets (Baker 1972; Hack 1975). Petri net (Peterson 1981) is one of the formal models used for analyzing systems that are concurrent, distributed and parallel. In ATPNS, array tokens are used to simulate the dynamism of the net.

ATPNS model along with a control feature called inhibitor arcs generate the same family of languages as generated by KAG, T0LG and P2DCFG. To increase the generative power of this model, adjunction rules are introduced and adjunct array token Petri net structure (AATPNS; Lalitha et al. 2012a) is defined. This model generates the table 1L languages and strictly included ATPNS family of languages. In Kamaraj et al. (2013) AATPNS is also compared with extended forms of SMGs (Nivat et al. 1989; Subramanian et al. 1989), extended P2DCFGs (Subramanian et al. 2008b) and internal parallel contextual grammars (Subramanian et al. 2008a).

Several approaches have been proposed or attempted to find a Chomsky-like hierarchy for array grammars. To find a hierarchy it is very essential to compare generating capacities of various models. Recently, in Bersani et al. (2011) and Pradella et al. (2011) mutual relationship between various context-free array grammars has been established. In this paper, we compare AATPNS, with some expressive rectangular picture grammar devices and also with REC and LOC.

This paper is organized in the following manner. In Sect. 2, basic definitions of arrays, Petri nets and notions of Petri nets pertaining to arrays have been recalled. In Sect. 3, we recall the definition of adjunct array token Petri nets and provide some illustrative examples. In Sect. 4, some of the closure properties are discussed. In Sect. 5 we compare AATPNS with various array grammars and also with REC and LOC with respect to the generative capacity.

2 Preliminaries

The following notation and definitions are mainly from Giammarresi and Restivo (1997), Bersani et al. (2011) and Lalitha et al. (2012b). In this section, we review some of the definitions of arrays, array grammars, Petri nets and notions of Petri nets pertaining to arrays.

2.1 Arrays and languages

Definition 2.1.1 Let Σ be a finite alphabet. A rectangular arrangement of elements of Σ is called an array or a picture over Σ. The set of all non-empty arrays over Σ is denoted by Σ^{++}.

For $h, k \geq 0$, $\Sigma^{(h,k)}$ denotes the set of pictures of size (h, k) and $\Sigma^{**} = \Sigma^{++} \cup \{\Lambda\}$ where Λ is the empty picture. We denote by $|p|_r$ and $|p|_c$, the number of rows and columns of a picture $p \in \Sigma^{**}$. The size of p is the pair $|p| = (|p|_r, |p|_c)$. A picture (array) language is a subset of Σ^{**}.

For $1 \leq i \leq |p|_r$, $1 \leq j \leq |p|_c$, the element of p in the ith row and jth column is called a pixel, denoted by p(i, j). The domain of a picture p, denoted by d(p), defined by

$$p = \begin{matrix} p(1,1) & \cdots & p(1,k) \\ \vdots & \ddots & \vdots \\ p(n,1) & \cdots & p(n,k) \end{matrix} \qquad \hat{p} = \begin{matrix} \# & \# & \# & \# & \# \\ \# & p(1,1) & \cdots & p(1,k) & \# \\ \# & \vdots & \ddots & \vdots & \# \\ \# & p(n,1) & \cdots & p(n,k) & \# \\ \# & \# & \# & \# & \# \end{matrix}$$

Fig. 1 A picture p and the corresponding picture $\hat{p}$

$d(p) = [1, |p|_r] \times [1, |p|_c] \subseteq N^2$. Subdomain of d(p) is a set of the form $\{k, k+1,\ldots,k'\} \times \{\ell, \ell+1,\ldots,\ell'\}$ where $1 \leq k \leq k' \leq |p|_r$ and $1 \leq \ell \leq \ell' \leq |p|_c$ also denoted as $(k, \ell; k', \ell')$. The set of subdomains of p is denoted by $D_s(p)$. Let $d_s = (k, \ell; k', \ell') \in D_s(p)$, then the subpicture $pic_s(p, d_s) \in \Sigma^{k'-k+1,\ell'-\ell+1}$ is defined as $pic_s(p, d_s)(i, j) = p(k + i - 1, \ell + j - 1)$ for all $1 \leq i \leq k' - k + 1, 1 \leq j \leq \ell' - \ell + 1$.

A subdomain is c-homogeneous (or homogeneous) when all pixels in the associated picture are identical to $c \in \Sigma$.

Let $u, v \in Z$, the translation of d_s by (u, v) is the subdomain $d_s \oplus (u, v) = (k + u, \ell + v; k' + u, \ell' + v)$.

An homogenous partition of a picture p is any partition $P = \{d_{s_1}, d_{s_2}, \ldots, d_{s_n}\}$ of d(p) into homogeneous subdomains $d_{s_1}, d_{s_2}, \ldots, d_{s_n}$. Then unit partition of p, written U(p), is the homogeneous partition of d(p) defined by single pixels. A homogeneous partition is called strong if adjacent subdomains have different labels and the partition is also unique in this case.

Let $\# \notin \Sigma$ be a boundary symbol. If $p \in \Sigma^{(h,k)}$, $\hat{p} \in \Sigma^{(h+2,k+2)}$ is the picture p surrounded by # as shown in Fig. 1.

A picture p of size (2, 2) is called a tile. Set of all tiles contained in a picture p is denoted by [[p]].

Let Γ and Σ be two finite alphabets and $\pi: \Gamma \to \Sigma$ a function called projection, if $p \in \Gamma^{**}$, the projection of p by π is the picture $p' \in \Sigma^{**}$ such that $p'(i, j) = \pi(p(i, j))$, for all $1 \leq i \leq |p|_r$, $1 \leq j \leq |p|_c$. Row and column catenations are partial operations on arrays denoted by $\bigcirc$ and $\square$. If $p, q \in \Sigma^{(k,*)}$ (resp. $p, q \in \Sigma^{(*,\ell)}$) $p \square q$ (resp. $p \bigcirc q$) is the horizontal (resp. vertical) juxtaposition of p and q. With $(p)^n$ [resp. $(p)_n$] is denoted the horizontal (resp. vertical) juxtaposition of n copies of p. If L_1 and L_2 are two array languages over Σ then the column catenation of L_1, L_2, denoted as $L_1 \square L_2$, is defined by

$$L_1 \square L_2 = \{p \square q / p \in L_1 \text{ and } q \in L_2\}.$$

The row catenation of L_1, L_2 can be defined in the similar notion.

2.2 Pure 2D context-free grammars

P2DCFGs, unlike the SMGs, admit rewriting of any row or column of pictures by equal length strings involving only terminal symbols.

Definition 2.2.1 A P2DCFG is a four-tuple $G = (\Sigma, P_c, P_r, \mathscr{M}_0)$, where Σ is a set of symbols, $P_c = \{t_{c_i} / 1 \le i \le m\}$, $P_r = \{t_{r_j} / 1 \le j \le n\}$.

Each t_{c_i} $(1 \le i \le m)$, 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 any two rules of the form $a \to \alpha$, $b \to \beta$ in t_{c_i}, have $|\alpha| = |\beta|$ where $|\alpha|$ denotes the length of α.

Each t_{r_j} $(1 \le j \le n)$, called a row table, is a set of context free rules of the form $c \to \gamma^T$, $c \in \Sigma$, $\gamma \in \Sigma^*$ such that any two rules of the form $c \to \gamma^T$, $d \to \delta^T$ in t_{r_j}, have $|\gamma| = |\delta|$.

$\mathscr{M}_0 \subseteq \Sigma^{++}$ is a finite set of axiom arrays.

Derivations are defined as follows. For any two arrays M_1, M_2, $M_1 \Rightarrow M_2$ denotes that M_2 is obtained from M_1 by either rewriting a column of M_1 by rules of a column table t_{c_i} in P_c or a row of M_1 by rules of a row table t_{r_j} in P_r. $\overset{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$.

The picture language L(G) generated by G is the set of rectangular picture arrays $\{M / M_0 \overset{*}{\Rightarrow} M \in \Sigma^{**}$, for some $M_0 \in \mathscr{M}_0\}$.

Definition 2.2.2 A P2DCFG with regular control [(R)P2DCFG] is a tuple $G_r = <G, \Gamma, C>$ where G is a P2DCFG, Γ is the control alphabet, the set of labels of the rule tables in $P_c \cup P_r$, $C \subseteq \Gamma^*$ is the regular control associated with the grammar.

If $p \in \Sigma^{**}$ and $S \in S'$, p is derived from S in G_r by means of a control word

$w = w_1 w_2 \cdots \in C$, in symbols $S \Rightarrow_w p$, if p is obtained from S by applying the column/row rules in sequence of tables $w_1 w_2 \cdots$ The language L(G) generated by (R)P2DCFG G_r is the set of pictures $\{p / S \Rightarrow_w p \in \Sigma^{++}$ for some $w \in C\}$.

While a P2DCFG allows rewriting any column or any row of a picture, a variant of P2DCFG investigated in Zbynek et al. (2014), allows rewriting of only the left most column or the upper most row. P2DCFG working under this derivation mode is known as (*l*/u) P2DCFG and the corresponding family of picture languages generated by them is denoted as (*l*/u) P2DCFL. It is shown that (Zbynek et al. 2014) (*l*/u) P2DCFL is incomparable with P2DCFL.

2.3 Tiling systems

TS define a ground level class of array languages known as recognizable languages (REC) by means of projection of pictures of a local language belonging to the class LOC. The local language is defined by means of a set of tiles.

Definition 2.3.1 Let Σ be a finite alphabet, θ a finite set of tiles over the alphabet $\Sigma \cup \{\#\}$. The local language defined by θ is the set $L(\theta) = \{p \in \Sigma^{**} / [[\hat{p}\,]] \subseteq \theta\}$. The family of local languages is denoted by LOC.

Definition 2.3.2 A TS is a four-tuple $T = (\Sigma, \Gamma, \theta, \pi)$, where Σ and Γ are two finite alphabets, θ is a finite set of tiles over the alphabet $\Gamma \cup \{\#\}$, and π: $\Gamma \to \Sigma$ is a projection. A picture $p \in \Sigma^{**}$ is recognized by T if there exists $p' \in \Gamma^{**}$ such that $p' \in LOC(\theta)$ and $=\pi(p')$. The class of all languages recognized by some TS is denoted by REC.

2.4 Prusa grammars

PGs are the formalisms that admit parallel application of rules in which non-terminal symbols can be substituted with rectangular pictures.

Definition 2.4.1 PG is a tuple (J, N, R, S), where J is the finite set of terminal symbols, disjoint from the set N of non-terminal symbols; $S \in N$ is the start symbol; and $R \subseteq N \times (N \cup J)^{++}$ is the set of rules.

Let G = (J, N, R, S) be a PG. We define a picture language L(G, A) over J for every $A \in N$. The definition is given by the following recursive descriptions:

(1) if $A \to w$ is in R, and $w \in J^{++}$, then $w \in L(G, A)$;
(2) let $A \to w$ be a production in R, $w = (N \cup J)^{(m,n)}$, for some m, $n \ge 1$, and $p_{i,j}$, with $1 \le i \le m$, $1 \le j \le n$, be pictures such that:
   (a) if $w(i, j) \in J$, then $p_{i,j} = w(i, j)$;
   (b) if $w(i, j) \in N$, then $p_{i,j} \in L(G, w(i, j))$;
   (c) if $P_k = p_{k,1} \square p_{k,2} \square \cdots \square p_{k,n}$, for any $1 \le i \le m$, $1 \le j \le n$, $|p_{i,j}|_c = |p_{i+1,j}|_c$, and $P = P_1 \bigcirc P_2 \bigcirc \cdots \bigcirc P_m$; then $P \in L(G, A)$.

The set L(G, A) contains exactly the pictures that can be obtained by applying a finite sequence of rules (i) and (ii). The language L(G) generated by grammar G is L(G, S).

2.5 Regional TRGs

TRGs (Crespi-Reghizzi and Pradella 2005) perform an isometric derivation process for which homogeneous sub pictures are replaced with isometric pictures of the local language defined by the right part of the rules.

Definition 2.5.1 A tile rewriting grammar (TRG) is a tuple (J, N, S, R), where J is the terminal alphabet, N is a set of non-terminal symbols, $S \in N$ is the starting symbol, R is a set of rules. Let $A \in N$. There are two kinds of rules in R:

(1) *fixed size* $A \to t$, where $t \in J$;
(2) *variable size* $A \to \omega$, ω is a set of tiles over $N \cup \{\#\}$.

At each step of the derivation, an A-homogeneous sub picture is replaced with an isometric picture of the local

Fig. 2 Transition of type 1

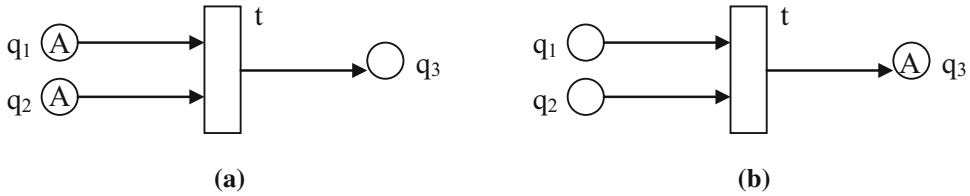
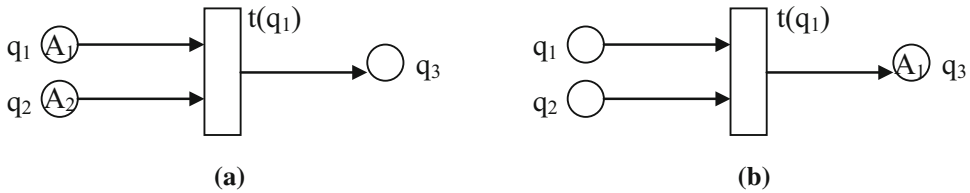


Fig. 3 Transition of type 2



language defined by the right part α of a rule $A \rightarrow \alpha$, where α admits a strong homogeneous partition. The process terminates when all non terminals have been eliminated from the current picture.

Regional tile grammars (RTG; Pradella et al. 2011) are the TRGs with specified set of tiling called regional.

Definition 2.5.2 A homogeneous partition is regional (HR) iff distinct (not necessarily adjacent) subdomains have distinct labels. A picture p is regional if it admits a HR partition. A language is regional if all its pictures are so. A regional tile grammar (RTG) is a TRG (see Definition 2.5.1), in which every variable size rule $A \rightarrow \omega$ is such that $L(\omega) \in LOC$, is a regional language.

2.6 Petri nets

Definition 2.6.1 Petri Net is one of the mathematical modeling tools for the description of distributed systems involving concurrency and synchronization. It is a weighted directed bipartite graph consisting of two kinds of nodes called places (represented by circles) and transitions (represented by bars). Places represent conditions and transition represents events. The places from which a directed arc runs to a transition are called input places of the transition and the places to which directed arcs run from a transition are called output places. Places in Petri nets may contain a discrete number of marks called tokens. Any distribution of tokens over the places will represent a configuration of the net called a marking. In the abstract sense, a transition of a Petri net may fire if it is enabled; when there are sufficient tokens in all of its input places.

Definition 2.6.2 A Petri net structure is a four tuple $C = (Q, T, I, O)$ where $Q = \{q_1, q_2,\ldots,q_n\}$ is a finite set of places, $n \geq 1$, $T = \{t_1, t_2,\ldots,t_m\}$ is a finite set of transitions $m \geq 1$, $Q \cap T = \phi$, $I: T \rightarrow Q^{\infty}$ is the input function from transitions to bags of places and $O: T \rightarrow Q^{\infty}$ is the output function from transitions to bags of places.

Definition 2.6.3 An inhibitor arc from a place q_l to a transition t_k has a small circle in the place of an arrow in regular arcs. This means the transition t_k is enabled only if q_l has no tokens in it. In other words a transition is enabled only if all its regular arc input places have required number of tokens and all its inhibitor arc (if exists) input places have zero tokens.

2.7 Array token Petri nets

In the array generating Petri Net structure, arrays over an alphabet J are used as tokens in some input places.

Definition 2.7.1 Row (resp. column) catenation rules in the form of $A \bigcirc B$ (resp. $A \square B$) can be associated with a transition t as a label, where A is a $m \times n$ array in the input place and B is an array language whose number of columns (resp. rows) depends on the number of columns (resp. rows) of A. Three types of transitions can be enabled and fired

(i) When all the input places of transition t (without label) having the same array as tokens.

- Each input place should have at least the required number of tokens (arrays).
- Firing t removes arrays from all its input places and moves the array to all its output places.

The graph in Fig. 2a represents position of arrays in the net before the transition t fires and Fig. 2b represents position of arrays in the net after the transition t fires.

(ii) When all the input places of transition t have the different arrays as tokens.

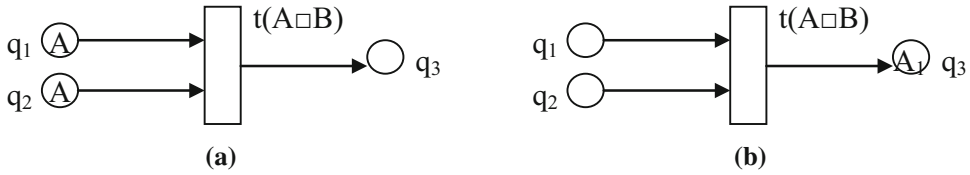


Fig. 4 Transition of type 3

- The label of t designates one of its input places which has sufficient number of same arrays as tokens.
- Firing t removes arrays from all its input places and moves the array from the designated input place to all its output places.
  The graph in Fig. 3a shows the transition t with label before firing and Fig. 3b shows the transition t with label after firing.

(iii) When all the input places of transition t (with row or column catenation rule as label) have the same array as tokens.

- Each input place should have at least the required number of tokens (arrays).
- Firing t removes arrays from all its input places and creates the catenated array as per the catenation rule, in all its output places.
  The graph in Fig. 4a shows the transition t with catenation rule before firing and Fig. 4b shows the transition t with catenation rule after firing.

In all the three types, firing of a transition t is enabled only if all the input places corresponding to inhibitor arcs (if exist) does not have any tokens in it.

Definition 2.7.2 An ATPNS is a five tuple $N = (J, C, M_0, \rho, F)$ where J is a given alphabet, $C = \langle Q, T, I, O\rangle$ is a Petri net structure with tokens as arrays over J, M_0: $Q \rightarrow J^{**}$, is the initial marking of the net, ρ: $T \rightarrow L$, a mapping from the set of transitions to set of labels of transitions and $F \subset Q$, is a finite set of final places.

3 Adjunct ATPNS

In this section, we recall the notions of AATPNS (Lalitha et al. 2012a) in generalized form and give some examples.

Definition 3.1 Adjunction is a generalization of catenation. In the row catenation $A \bigcirc B$, the array B is joined to A after the last row. But row adjunction can join the array B into array A after any row of A. Similarly column adjunction can join the array B into array A after any column of A. Let A be an $m \times n$ array in J^{**} called host array; $B \subset J^{**}$ be an array language whose members, called adjunct arrays have fixed number of rows. A row adjunct rule (RAR) joins an adjunct array B into a host array A in two ways: by post rule denoted by (A, B, ar_j), array B is juxtaposed into array A after jth row and by pre rule denoted by (A, B, br_j), array B is juxtaposed into array A before jth row. The number of columns of B is same as the number of columns of A. In the similar notion column adjunct rule (CAR) can also be defined in two ways: post rule (A, B, ac_j) and pre rule (A, B, bc_j) joining B into A, after jth column of A and before jth column of A, respectively. It is obvious that a row catenation rule $A \bigcirc B$ in ATPNS is a post RAR rule (A, B, ar_m) and column catenation rule $A \square B$ is a post CAR rule (A, B, ac_n). Transitions of a Petri net structure can also be labeled with RAR or CARs.

Definition 3.2 An AATPNS is a five tuple $N = (J, C, M_0, \rho, F)$ where J is a given alphabet, $C = (Q, T, I, O)$ is a Petri net structure with tokens as arrays over J, M_0: $Q \rightarrow J^{**}$, is the initial marking of the net, ρ: $T \rightarrow L$, a mapping from the set of transitions to set of labels where catenation rules of the labels are either RAR or CAR and $F \subset Q$, is a finite set of final places.

In AATPNS, the types of transitions which can be enabled and fired are similar to that of Definition 2.7.1 except the type (iii) where labels of transitions may be RAR or CAR rules instead of row or column catenation rules.

When all the input places of a transition t (with RAR or CAR rule as label) have the same array as tokens.

- Each input place should have at least the required number of tokens (arrays).
- Firing t removes arrays from all its input places and creates the array through adjunction as per the RAR or CAR rule, in all its output places.

In all the three types, firing of a transition t is enabled only if all the input places corresponding to inhibitor arcs (if exists) does not have any tokens in it.

Definition 3.3 If P is an AATPNS then the language generated by P is defined as $L(P) = \{X \in J^{**} / X$ is in the place q for some q in F$\}$. Starting with arrays (tokens) over a given alphabet as initial marking, all possible

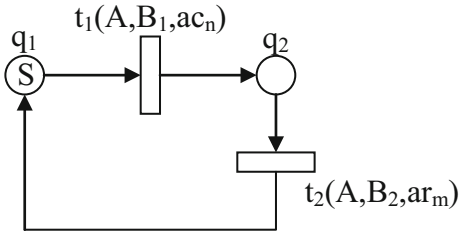


Fig. 5 Petri net for generating the picture language L_1

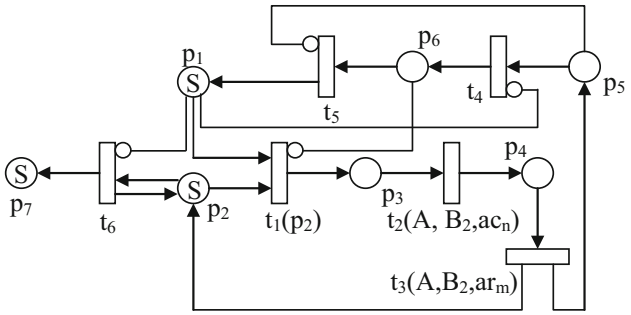


Fig. 6 Petri net for generating the picture language L_3

sequences of transitions are fired. Set of all arrays created in final places F is called the language generated by AATPNS.

Example 3.1 Consider the AATPNS $P_1 = (J, C, M_0, \rho, F)$ where $J = \{0, 1\}$, $C = (Q, T, I, O)$, $Q = \{q_1, q_2\}$, $T = \{t_1, t_2\}$, $I(t_1) = \{q_1\}$, $I(t_2) = \{q_2\}$, $O(t_1) = \{q_2\}$, $O(t_2) = \{q_1\}$, M_0 is the initial marking: the array S is in q_1 and there is no array in q_2, $\rho(t_1) = (A, B_1, ac_n)$ and $\rho(t_2) = (A, B_2, ar_m)$ and $F = \{q_1\}$. The arrays used in the net are defined as follows: $S = \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$ and $B_1 = (0)_m$ and $B_2 = (0)^{n-1}\,1$. The Petri net graph is given in Fig. 5.

Initially t_1 is the only enabled transition. Firing of t_1 adjoins a column of 0's after the last column of array S and puts the derived array in q_2, making t_2 enabled. Firing t_2 adjoins a row of 0's ending with 1 after the last row of the array in q_2 and puts the derived array in q_1. When the transitions t_1, t_2 fire the array that reaches the output place q_1 is shown as $\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \overset{t_1}{\Rightarrow} \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{matrix} \overset{t_2}{\Rightarrow} \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$. Firing the sequence $(t_1t_2)^2$ generates the output array as $\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$. The language L_1 generated by Petri net is set of square pictures over {0, 1} with 1's in the main diagonal and other elements are 0's.

Example 3.2 If in Example 3.1, $J = \{a\}$, $S = a$, $B_1 = (a)_m$, $B_2 = (a)^n$, then the language L_2 of square pictures of a's is generated.

Example 3.3 Consider the AATPNS $P_3 = (J, C, M_0, \rho, F)$, where $J = \{a\}$, the Petri net structure is $C = (Q, T, I, O)$ with $Q = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$, $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$, $I(t_1) = \{p_1, p_2\}$, $I(t_2) = \{p_3\}$, $I(t_3) = \{p_4\}$, $I(t_4) = \{p_1, p_5\}$, $I(t_5) = \{p_5, p_6\}$, $I(t_6) = \{p_1, p_2\}$, $O(t_1) = \{p_3\}$, $O(t_2) = \{p_4\}$, $O(t_3) = \{p_2, p_5\}$, $O(t_4) = \{p_6\}$, $O(t_5) = \{p_1\}$, $O(t_6) = \{p_7, p_2\}$.

ρ: T → L is defined as follows: $\rho(t_1) = p_2$, $\rho(t_2) = (A, B_1, ac_n)$, $\rho(t_3) = (A, B_2, ar_m)$, $\rho(t_4) = \lambda$, $\rho(t_5) = \lambda$, $\rho(t_6) = \lambda$, $\rho(t_7) = \lambda$, $F = \{p_7\}$. The Petri net graph is given in Fig. 6. The arrays used are $S = \begin{matrix} a & a \\ a & a \end{matrix}$, $B_1 = (a\ a)_m$, $B_2 = \begin{pmatrix} a \\ a \end{pmatrix}^n$.

To start with only t_1 is enabled. Firing of sequence of transitions $t_1t_2t_3$ results in a square of a's of size 4×4 in p_2 and p_5. At this stage both t_6 and t_4 are enabled. Firing the sequence $t_1t_2t_3t_6$ puts a square of size 4×4 in p_7. Firing t_4 pushes the array to p_6, emptying p_5. In this position t_5 is enabled. Firing t_5 puts two copies of same array in p_1. Since at this stage there are two tokens in p_1, the sequence $t_1t_2t_3$ has to fire two times to empty p_1. The firing of sequence $t_4t_5\,(t_1t_2t_3)^2\,t_6$ puts a square of a's of size 8×8 in p_7. The inhibitor input p_1 make sure that a square of size 6×6 does not reach p_7. This AATPNS generates the language L_3 of squares of a's of size $(2^n, 2^n)$, $n \geq 1$.

Example 3.4 The AATPNS $P_4 = (J, C, M_0, \rho, F)$ with $J = \{a, b\}$, $F = \{p\}$ given in Fig. 7, where $S \in \left\{ \begin{matrix} a & b & b & a & b & b & b & b & b & a & a & b & a & a & a & a \\ b & b & b & b & b & b & b & b & a & a & a & a & a & a & a & a \end{matrix} \right\}$, $B_1 = (aa)_m$, $B_2 = a^n$, $B_3 = (bb)_m$, $B_4 = b^n$, generates the language L_4 of pictures of symmetrical squares, where each square is composed by nested "L" shaped strings of the same character over the alphabet {a, b}.

A typical picture in this language is given in Fig. 8.

4 Closure properties

In this section we discuss the closure property with respect to various standard array operations. The formal definitions of these operations are given in Siromoney et al. (1973). We omit some of the proofs as they are straight forward.

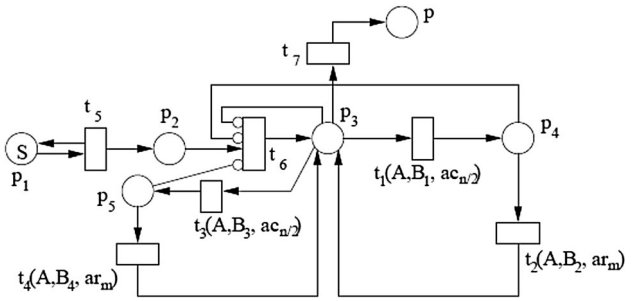


Fig. 7 Petri net for generating the picture language L_4

$$\begin{matrix} a & b & b & a & a & b & b & a \\ b & b & b & a & a & b & b & b \\ b & b & b & a & a & b & b & b \\ a & a & a & a & a & a & a & a \end{matrix}$$

Fig. 8 A typical picture of L_4

Theorem 4.1 *The family of languages generated by AATPNS is closed under the following picture operations.*

(i) *transpose*,
(ii) *rotation through* 90°,
(iii) *rotation through* 180°,
(iv) *reflection about the rightmost vertical*,
(v) *reflection about the base*.

Theorem 4.2 *If L_1 and L_2 are two array languages generated by AATPNSs then $L_1 \square L_2$ and $L_1 \bigcirc L_2$ need not be generated by another AATPNS.*

Proof If we consider both L_1 and L_2 are same as the language L_4 in Example 3.4, then it is easy to see that $L_1 \square L_2$ and $L_1 \bigcirc L_2$ will contain arrays which do not have any pattern. Hence it is impossible to construct a Petri net with finite number of transitions to generate all the arrays.

Theorem 4.3 *The family of languages generated by AATPNS is closed under union and intersection.*

Proof Let P_1 be an AATPNS with start array S_1 at a place p_1 and let P_2 be another AATPNS with start array S_2 at a place p_2. In the net to generate the union, have two extra places p, q with S_1 and S_2, respectively as tokens. Let the extra transitions be t_p and t_q in the net. Let the label of t_p be p and the label of t_q be q. Let the input places of both t_p and t_q be p, q. Output place of t_p is p_1 and output place of t_q is p_2. Rest of the net would be the combination of C_1 and C_2. Once t_p fires, the Petri net P_1 would become operational or if t_q fires, P_2 would become operational. Hence the union of the languages generated by P_1 and P_2 can also be generated by an AATPNS.

If we construct another Petri net with new transition, which fires if and only if the transitions in P_1 and P_2 which have same labeling rule and same input array fires, then the intersection of the languages generated by P_1 and P_2 can also be generated by an AATPNS.

5 Comparative results

In the following results we use the notation $\mathcal{L}(X)$ to denote the family of all languages generated by the device X.

Theorem 5.1 *(R)P2DCFL ⊂ L (AATPNS).*

Proof Let the picture array language be generated by the P2DCFG $G = (\Sigma, P_c, P_r, \mathscr{M}_0)$, where Σ is a set of symbols, $P_c = \{t_{c_i}/1 \le i \le m\}$ is a set of column tables, $P_r = \{t_{r_j}/1 \le j \le n\}$ is a set of row tables, $\mathscr{M}_0 \subseteq \Sigma^{++}$ is a finite set of axiom arrays, with a regular control language over the set of labels, say $(\ell_1, \ell_2, \ldots, \ell_m)$. Application of column table is equivalent to a column adjunction. Hence for every column table t_{c_i}, a corresponding CAR(A, B, ac_i/bc_i) can be defined. Similarly for every row table t_{r_j}, a corresponding RAR(A, B, ar_j/br_j) can be defined. If it is assumed that the derivation $M_0 \overset{w}{\Rightarrow} M$ yields an array M of the language, where w is a control word $\ell_1, \ell_2, \ldots, \ell_m$ then the AATPNS P can be constructed as follows.

Let p_0 be a place with array M_0 as token. Let t_1 be a transition with the adjunction rule corresponding to ℓ_1 as a label; p_0 being the input place and p_1 as its output place. Have a transition t_2 with the adjunction rule corresponding to ℓ_2 as a label; p_1 being the input place and p_2 as its output place and so on. Have a transition t_m with the adjunction rule corresponding to the table ℓ_m as label; p_{m-1} being the input place and p_0 as its output place. Let $F = \{p_0\}$.

The firing sequence $t_1 t_2 \cdots t_m$ will have the same effect as applying the rules $\ell_1, \ell_2, \ldots, \ell_m$ in that order once. The firing the sequence $(t_1 t_2 \cdots t_m)^n$ generates the same array which is obtained by applying the set of tables in the control word $(\ell_1, \ell_2, \ldots, \ell_m)^n$. Thus the Petri net P constructed will generate the language generated by the (R)P2DCFG G. In other words, (R)P2DCFL $\subseteq \mathcal{L}$ (AATPNS).

For the strict inclusion, we consider the language L_3 in Example 3.3 which cannot be generated by any (R)P2DCFG (Bersani et al. 2011, Proposition 5.5).

Since P2DCFL ⊂ (R)P2DCFL (Subramanian, Theorem 7), we can state the following:

Corollary 5.1 *P2DCFL ⊂ L (AATPNS).*

Theorem 5.2 *(R)(l/u)P2DCFL ⊂ L (AATPNS).*

Proof Let L be the picture language generated by a (l/u) P2DCFG $G = (\Sigma, P_c, P_r, \mathscr{M}_0)$, where Σ is a set of symbols, $P_c = \{t_{c_i}/1 \le i \le m\}$ is a set of column tables, $P_r = \{t_{r_j}/1 \le j \le n\}$ is a set of row tables, $\mathscr{M}_0 \subseteq \Sigma^{++}$ is a

finite set of axiom arrays, with a regular control language over the set of labels, say (ℓ_1, ℓ_2,…,ℓ_m). Application of a column table under (*l*/u) mode of derivation is equivalent to a column adjunction before or after the first column of the host array. Hence for every column table t_{c_j}, a corresponding CAR(A, B_j, ac_1/bc_1) can be defined. Similarly for every row table t_{r_k}, a corresponding RAR(A, B_k, ar_1/br_1) can be defined. If it is assumed that the derivation $M_0 \overset{w}{\Rightarrow} M$ yields an array M of the language, where w is a control word ℓ_1, ℓ_2,…,ℓ_m then the AATPNS P can be constructed as in Theorem 5.2.

Thus the Petri net P constructed will generate the language generated by the (R)(*l*/u)P2DCFG G. In other words, (R)(*l*/u)P2DCFL $\subseteq$ $\mathcal{L}$ (AATPNS).

For the strict inclusion, we again consider the language L_3 in Example 3.3 which is not present in (R)P2DCFL. The families (R)P2DCFL and (R)(*l*/u)P2DCFL coincide if we restrict to only a unary alphabet. Since there is a single symbol and the column rules and row rules can use only one symbol, rewriting any column (row) is equivalent to rewriting the leftmost column (uppermost row). Since L_3 is over unary alphabet {a}, it is not in (R) (*l*/u) P2DCFL also.

Since (*l*/u)P2DCFL $\subset$ (R)(*l*/u)P2DCFL (Zbynek et al. 2014, Theorem 4), we can state the following:

Corollary 5.2 (*l*/u)*P2DCFL* $\subset$ *L* (*AATPNS*).

Theorem 5.3 $\mathcal{L}$ (*AATPNS*) *is incomparable with* $\mathcal{L}$ (*RTG*) *and* $\mathcal{L}$ (*PG*) *but not disjoint*.

Proof The language L_2 of squares over {a}, in Example 3.2, can be generated by a PG, G =(N, T, P, S) where N = {S, H, V}, T = {a} and $P = \left\{ S \rightarrow \begin{matrix} a & H \\ V & S \end{matrix}, H \rightarrow aH/a, V \rightarrow \begin{matrix} a \\ V \end{matrix}/a, S \rightarrow a \right\}$.

Since $\mathcal{L}$ (PG) $\subset$ $\mathcal{L}$ (RTG; Pradella et al. 2011, Proposition 8), this language is also in $\mathcal{L}$ (RTG).

The incomparability with $\mathcal{L}$ (RTG) can be seen as follows: the AATPNS language L_4 of Example 3.4, which consists of pictures of symmetrical squares, where each square is composed by nested "L" shaped strings of the same character. This language cannot be generated by any regional tile grammar and hence by PG. In order to define two square regions, a production of the grammar should define them in the first derivation: since the partition is strong, the two sub-pictures generated from two different non-terminals say *A* and *B* are regionally defined, i.e., the derivations of the one can not affect the derivation of other. So there does not exist a method to make them symmetric.

Now, the language L_{+b} of pictures consists of a horizontal and a vertical string of b's (not in border) in the background of a's can be generated by a PG (Prusa 2004, Example 8) and hence by a RTG. A typical member of L_{+b} is given in Fig. 9.

Fig. 9 A typical picture in L_{+b}

a a a b a
a a a b a
b b b b b
a a a b a
a a a b a

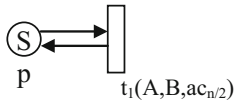


Fig. 10 Petri net for generating the picture language L_6

This language cannot be generated by any AATPNS, as the number of transitions in the net cannot depend on the size of the array. In an m × n array of a's a column of b's can be adjuncted in n − 1 ways and a row of b's can be adjuncted in m − 1 ways. To insert both a column of b's and a row of b's the net required (m − 1)(n − 1) transitions with corresponding adjunction rules. Hence it is not feasible to generate these arrays using AATPNS.

Theorem 5.4 $\mathcal{L}$ (*AATPNS*) *is incomparable but not disjoint with LOC and REC*.

Proof The AATPNS language L_1 of square pictures over {0, 1} with 1's in the main diagonal and other elements are 0's in Example 3.1, is in LOC (Giammarresi and Restivo 1997, Example 7.1) and hence in REC, as LOC $\subset$ REC (Giammarresi and Restivo 1997). So, $\mathcal{L}$ (AATPNS) $\cap$ LOC $\cap$ REC $\neq$ ϕ.

The incomparability of $\mathcal{L}$ (AATPNS) with REC and hence with LOC is shown as follows: the language L_6 = $\{a^n b^n / n \geq 1\}$ is not in REC (Prusa 2004, Theorem 11) but can be generated by a AATPNS, P_5 = (J, C, M_0, C, F) with J = {a, b}, F = {p} given in Fig. 10.

where S = ab, B = ab.

The language L_{diag} of pictures containing arbitrary number of diagonals of 1 (no single 1 are admitted at corners) that are separated by at least one diagonal of 0's is in LOC. But L_{diag} cannot be generated by any AATPNS. The only 2 × 2 array in the language is $\begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix}$. But there are four 3 × 3 arrays with the property belonging to the language. To generate four 3 × 3 arrays from the start array we need eight transitions with different array languages involved in the labels of the transitions. Hence it is impossible to construct a net with finite number of transitions.

6 Conclusion

In this paper we have considered a variant class of array token Petri nets with column and row adjunction rules as

labels of transitions. We compared this model with some of the expressive grammar models: (R)P2DCFG, (R)(*l*/u)P2DFCG, RTG, PG and also with REC and LOC. We have shown that AATPNS have higher generative capacity than (R)P2DCFG and (R)(*l*/u)P2DCFG but incomparable with other models with non-empty intersection. The non-empty intersection clearly suggests that this model can generate a wide variety of digitized pictures and patterns. It will be of interest to allow the removal of a column or a row from the picture array, by the firing of transition in the Petri nets and examine the effect in the derivation of picture. The application of this model in picture processing tasks and Pattern recognition should be investigated further.

References

Baker HG (1972) Petri net languages. In: Computation structures group memo 68, Project MAC. MIT, Cambridge

Bersani MM, Frigeri A, Cherubini A (2011) On some classes of 2D languages and their relations. In: Aggarwal JK et al (eds) IWCIA 2011, LNCS, vol 6636, pp 222–234

Cherubini A, Crespi-Reghizzi S, Pradella M, Peitro PS (2006) Picture languages: tiling systems versus tile rewriting grammars. Theor Comput Sci 356:90–103

Crespi-Reghizzi S, Pradella M (2005) Tile rewriting grammars and picture languages. Theor Comput Sci 340:257–272

Giammarresi D, Restivo A (1997) Two-dimensional languages. In: Rozenberg G Salomaa A (eds) Handbook of formal languages, vol 3. Springer, Heidelberg, pp 215–267

Hack M (1975) Petri net languages. In: Computation structures group memo 124, Project MAC. MIT, Cambridge

Kamaraj T, Lalitha D, Thomas DG (2013) A Comparative study on adjunct array token Petri nets with some classes of array grammars. Appl Math Sci Hikari Publ 7(135):6705–6713

Kamaraj T, Lalitha D, Thomas DG (2014) A study on expressiveness of a class of array token Petri nets, In: M. Pant et al (eds.), SOCPROS 2013, Adv Intell Syst Comput Springer India 259:457–469

Lalitha D, Rangarajan K, Thomas DG (2012a) Adjunct array images using Petri nets. Indian J Math Math Sci 8(1):11–19

Lalitha D, Rangarajan K, Thomas DG (2012) Rectangular arrays and Petri nets. In: Barneva RP et al (eds) IWCIA 2012, LNCS, vol 7655, Springer, Heidelberg, pp 166–180

Nivat M, Saoudi A, Dare R (1989) Parallel generation of finite images. Int J Pattern Recognit Artif Intell 3(1989):279–294

Peterson JL (1981) Petri net theory and modeling of systems. Prentice Hall, Inc., Englewood Cliffs

Pradella M, Cherubini A, Crespi-Reghizzi S (2011) A unifying approach to picture grammars. Inf Comput 209:1246–1267

Prusa D (2004) Two-dimensional languages. PhD Thesis, Charles University, Faculty of Mathematics and Physics, Czech Republic

Rosenfeld A, Siromoney R (1993) Picture languages—a survey. Lang Des 1(3):229–245

Siromoney R (1987) Advances in array languages. In: Proceedings of the 3rd international workshop on graph grammars and their application to computer science, LNCS, vol 291. Springer, Heidelberg, pp 549–563

Siromoney R, Siromoney G (1977) Extended controlled table L-arrays. Inf Control 35:119–138

Siromoney G, Siromoney R, Krithivasan K (1972) Abstract families of matrices and picture languages. Comput Graph Image Process 1:284–307

Siromoney G, Siromoney R, Kamala K (1973) Picture languages with array rewriting rules. Inf Control 22:447–470

Siromoney G, Siromoney R, Kamala K (1974) Array grammars and kolam. Comput Graph Image Process 3(1):63–82

Subramanian KG et al (1989) Siromoney array grammars and applications. Int J Pattern Recognit Artif Intell 3(1989):333–351

Subramanian KG, Van DL, Helen Chandra P, Quyen ND (2008a) Array grammars with contextual operations. Fundam Inform 83(2008):411–428

Subramanian KG et al (2008b) Two-dimensional picture grammar models. In: Proceedings of the 2nd European modelling symposium, EMS2008. IEEE, Los Alamitos, pp 263–267

Subramanian KG, Rosihan M, Geethalakshmi M, Nagar AK (2009) Pure 2D picture grammars and languages. Discret Appl Math 157(16):3401–3411

Zbynek K et al (2014) A variant of pure two-dimensional context-free grammars generating picture languages. In: Barneva RP et al (eds) IWCIA 2014, LNCS, vol 8466. Springer, Heidelberg, pp 123–133