Petri Net Generating Hexagonal Arrays

D. Lalitha[1], K. Rangarajan[2], and D.G. Thomas[3]

[1] Department of Mathematics, Sathyabama University
Chennai - 600 119, India
lalkrish_24@yahoo.co.in

[2] Department of Mathematics, Barath University, Chennai - 600 073, India
kr2210@hotmail.com

[3] Department of Mathematics, Madras Christian College
Tambaram, Chennai - 600 059, India
dgthomasmcc@yahoo.com

Abstract. A new model to generate hexagonal arrays using Petri net structure has been defined. The catenation of an arrowhead to a b-hexagon results in a similar b-hexagon. This concept has been used in Hexagonal Array Token Petri Net Structure (HATPNS). A variation in the position of catenation has been introduced in Adjunct Hexagonal Array Token Petri Net Structure (AHATPNS). Comparisons with other hexagonal array models have been made.

Keywords: Hexagonal picture languages, Petri nets, arrowhead catenations, adjunct rules, array tokens.

1 Introduction

Hexagonal arrays and hexagonal patterns are found in the literature on picture processing and scene analysis. Image generation can be done in many ways in formal languages. Several grammars were introduced in the literature to generate various classes of hexagonal picture languages. The class of Hexagonal Kolam Array Languages (HKAL) was introduced by Siromoneys [7]. The class of Hexagonal Array Languages (HAL) was introduced by Subramanian [8]. The class of local and recognizable hexagonal picture languages were introduced by Dersanambika et al. [1].

On the other hand, a Petri net is an abstract formal model of information flow [3]. Petri nets have been used for analysing 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 rectangular arrays has been introduced in [5]. Motivated by this concept we have introduced a Petri net model to generate

J.K. Aggarwal et al. (Eds.): IWCIA 2011, LNCS 6636, pp. 235–247, 2011.

hexagonal picture languages. In this model hexagonal arrays over a given alphabet are used as tokens in the places of the net. Labeling of transitions are defined as arrowhead catenation rules. Firing a sequence of transitions starting from a specific initial marking leading to a finite set of terminal markings would catenate the arrowheads to the initial array and move the array to the final set of places. The collection of such arrays is defined as the language generated by the Petri net structure. We call the resulting model as Hexagonal Array Token Petri Net Structure (HATPNS). The generative capacity of HATPNS is compared with HKAG [7] and HAG [8]. Various positions of adjunction have been defined to join the arrowhead into the hexagonal array of the input place. With these adjunction rules as labels the firing sequence would move the input array to the final place after joining the various arrowheads. This model is called Adjunct Hexagonal Array Token Petri Net Structure (AHATPNS). The generative capacity of this model is compared with controlled table 0L/1L hexagonal array grammars, extended 2D hexagonal context-free grammar and HATPNS.

One application for such a generation is in biomedical image processing [6]. A programmable cellular automaton is used for processing biomedical images. The cellular register is arranged in a hexagonal pattern produced by alternatively switching the shift register stages selected from line to line. Other applications are in the field of tiling patterns and generation of kolam patterns.

The paper is organized as follows: Section 2 recalls the notions of hexagonal arrays and arrowheads. Section 3 introduces a new model HATPNS. Section 4 compares the generative powers of HATPNS with other existing models. Section 5 introduces another model AHATPNS and compares it with HATPNS.

2 Hexagonal Arrays and Arrowheads

Let Σ be a finite non empty set of symbols. The set of all hexagonal arrays made up of elements of Σ is denoted by Σ^{**H}. The size of any hexagonal array is defined by its parameters. For a hexagon the parameters are $|H|_{LU}$, $|H|_{RU}$, $|H|_{R}$, $|H|_{RL}$, $|H|_{LL}$ and $|H|_{L}$ where LU stands for left upper; RU, right upper; R, right; RL, right lower; LL, left lower and L, lower. A hexagon of Σ^{**H} is shown in Fig. 1(a) where $\Sigma = \{a\}$ and the sides of the hexagon are shown in Fig. 1(b).

For any hexagon three types of catenation with six arrowheads are possible.

For a hexagon H an arrowhead of type $A_1 (A_2)$ can be catenated in the direction $\downarrow$ $(\uparrow)$ if and only if $|H|_{LL} = |A_1|_{LU}$, $|H|_{RL} = |A_1|_{RU}$ $(|H|_{LU} = |A_2|_{LL}$, $|H|_{RU} = |A_2|_{RL})$. Similarly the other two catenations are possible subject to the corresponding conditions [2, 7]. The arrowheads for the hexagon in Fig. 1(a) are shown in Fig. 2. An arrowhead is written in the form $\{\ldots \langle v \rangle \ldots\}$ where $\langle v \rangle$ dentoes the vertex and the arrowhead is written in the clockwise direction.

Hexagon has 6 sides namely 1, 2, 3, 4, 5, 6 as shown in Fig. 1(a). Then A_1 will be catenated with sides 4, 5; A_2 with sides 1, 2; A_3 with sides 2, 3; A_4 with sides 5, 6; A_5 with sides 6, 1 and A_6 with sides 3, 4.

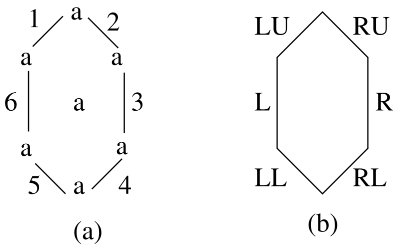


Fig. 1.

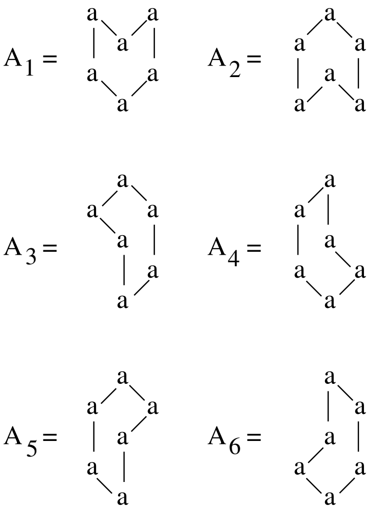


Fig. 2. Arrowheads for the hexagon in Fig. 1

3 Hexagonal Array Token Petri Net Structure

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 $m \geq 0$, $P \cap T = \phi$. $I : T \rightarrow P^{\infty}$ is the input function from transition to the bags of places and $O : T \rightarrow P^{\infty}$ is the output function from the transition to the 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 the Petri net. The number and position of tokens may change during the execution of the Petri net.*

Definition 3. *If the tokens are arrays over a given alphabet Σ then the Petri net is an array token Petri net [5].*

3.1 Firing Rules

A transition without any label will fire only if all the input places have the same hexagonal array as a token. Then on firing the transition arrays from all the input places are removed and put in all its output places. If all the input places have different arrays then the transition without label cannot fire. If the input places have different arrays then the label of the transition has to specify an input place. When the transition fires the arrays in the input places are removed and the array in the place specified in the label is put in all the output places.

3.2 Arrowhead Catenation Rules as Labels

Let a transition t have $H \circledast A$ as a label where $\circledast$ is any one of the six directions $(\textcircled{\nearrow}, \textcircled{\rightarrow}, \textcircled{\searrow}, \textcircled{\swarrow}, \textcircled{\leftarrow}, \textcircled{\nwarrow})$, H is the hexagonal array in all the input places and A a predefined arrowhead. Then firing the transition will catenate A with H in the specified direction and put in all the output places subject to the condition of catenation. If the condition for catenation is not satisfied then the transition cannot fire.

Example 1.

$$\text{Let } H = \begin{array}{ccc} & a & \\ a & & a \\ & a & \\ a & & a \\ & a & \end{array}, \quad A_6 = \begin{array}{cc} & b \\ & b \\ b & \end{array}, \quad H_1 = \begin{array}{cccc} & a & & \\ a & & a & \\ & a & & b \\ a & & a & \\ & a & & b \\ & & b & \end{array}$$

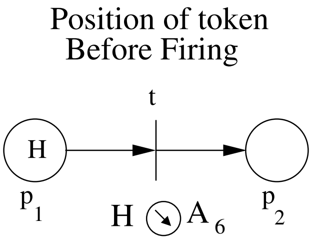


Fig. 3.

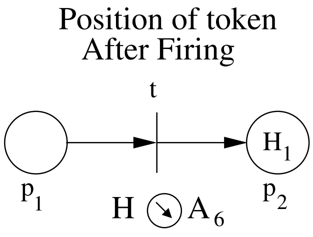


Fig. 4.

Definition 4. *If $C = (P, T, I, O)$ is a Petri net structure with hexagonal arrays of Σ^{**H} as initial markings $M_0 : P \rightarrow \Sigma^{**H}$, labels of transitions being arrowhead catenation rules and a finite set of final places $F \subseteq P$, then the Petri net structure is defined as Hexagonal Array Token Petri Net Structure (HATPNS).*

Definition 5. *If C is a HATPNS then the language generated by the Petri net C is defined as*

$$L(C) = \{H \in \Sigma^{**H} / H \text{ is in } p \text{ for some } p \in F\}$$

With hexagonal 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. Let $\Sigma = \{X, \bullet\}$;

$$B_1 = \begin{pmatrix} \bullet \\ \bullet \\ \bullet \end{pmatrix}^{|H|_{LU}\ -1} \left\langle \begin{matrix} X \\ X \\ X \end{matrix} \right\rangle \begin{matrix} X & X & X \\ \bullet & \bullet & X \\ \bullet & \bullet & X \end{matrix}\,;$$

$$B_2 = \begin{matrix} X & X & X \\ X & \bullet & \bullet \\ X & \bullet & \bullet \end{matrix} \left\langle \begin{matrix} X \\ X \\ X \end{matrix} \right\rangle \begin{pmatrix} \bullet \\ \bullet \\ \bullet \end{pmatrix}^{|H|_{R}\ -2} \begin{matrix} X \\ X \\ X \end{matrix}\,;$$

$$B_3 = \begin{matrix} X & X & X \\ X & \bullet & \bullet \\ X & \bullet & \bullet \end{matrix} \left\langle \begin{matrix} X \\ X \\ X \end{matrix} \right\rangle \begin{pmatrix} X \\ \bullet \\ \bullet \end{pmatrix}^{|H|_{R}\ -2} \begin{matrix} X \\ X \\ X \end{matrix} \text{ and}$$

$F = \{p_5\}$.

$$\text{Consider } S = \begin{array}{ccccccc} & & & X & & & \\ & & X & & X & & \\ & X & & \bullet & & X & \\ X & & \bullet & & \bullet & & X \\ & X & & \bullet & & X & \\ X & & X & & X & & \bullet \\ & \bullet & & X & & \bullet & \\ X & & \bullet & & \bullet & & \bullet \\ & \bullet & & X & & \bullet & \\ X & & \bullet & & \bullet & & X \\ & X & & X & & X & \\ & & X & & X & & \\ & & & X & & & \end{array}$$

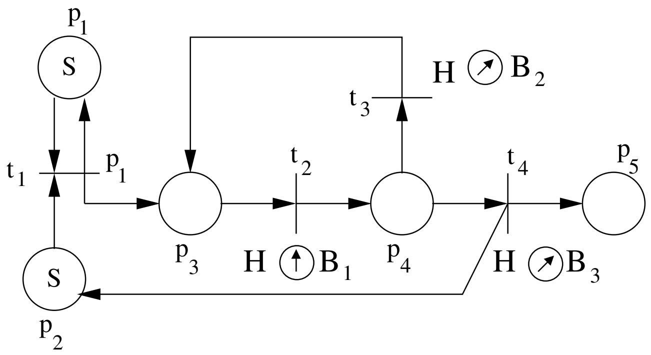
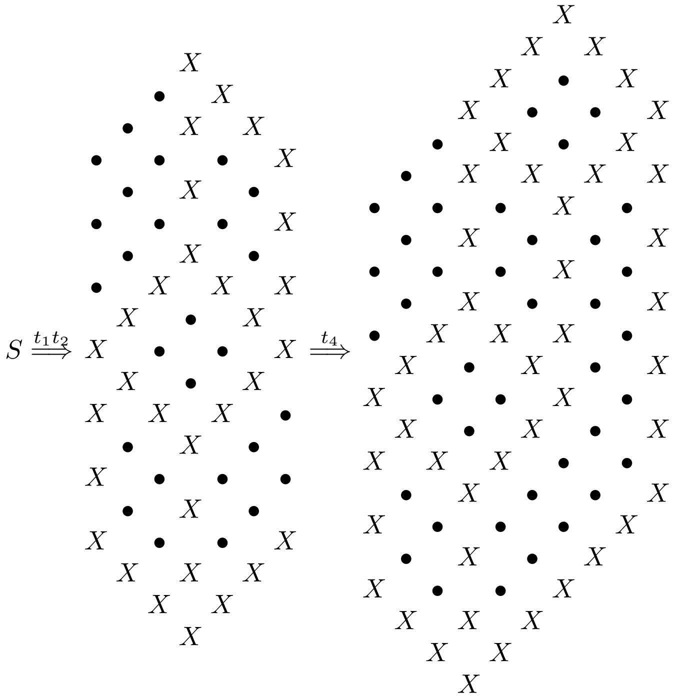


Fig. 5. *HATPNS*

When t_1 fires the token from place p_1 moves to p_3. Derivation of the first array of the language is shown below:


$S \overset{t_1 t_2}{\Longrightarrow}$
$\overset{t_4}{\Longrightarrow}$

The firing sequence $t_1(t_2t_3)^{n-1}t_2t_4$ gives the n^{th} array of the language.

It should be noted that the sizes of the arrowheads B_1, B_2, B_3 are not fixed, but vary depending on the sizes of the hexagons in the input place of the transition, so that the condition for catenation is satisfied.

4 Comparison Results

In this section we recall the definition of hexagonal array languages [8], and hexagonal kolam array languages [7] and compare the generative power of HATPNS with $(X : Y)HAL$, $(R : Y)HKAL$ where $X, Y \in \{R, CF, CS\}$.

Definition 6. *A Hexagonal Array Grammar (HAG) is $G = (N, I, T, P, S, \mathcal{L})$ where N, I, T are finite sets of non terminals, intermediates and terminals respectively; $P = P_1 \cup P_2$ is a finite set of productions and $S \in N$ is the start symbol. For each A in I, L_A is an intermediate language which is regular, CF or CS string language written in the appropriate arrowhead form. An arrowhead is written in the form $\{\ldots\langle v\rangle\ldots\}$ where $\langle v\rangle$ denotes the vertex and the arrowhead is written in the clockwise direction $\mathcal{L} = \{L_A/A \in I\}$. G is called $(X : R)HAG$, $(X : CG)HAG$ or $(X : CS)HAG$ depending on the rules of P_2. X is R, CF or CS according as all intermediate languages are regular, atleast one of them is CF or atleast one of them is CS. The language generated by G is $(X : Y)HAL$.*

Definition 7. *A Hexagonal Kolam Array Grammar (HKAG) is G is a 5-tuple $(V, I, P, S, \mathcal{L})$ where $V = V_1 \cup V_2$, V_1 is a finite sets of non terminals and V_2 is a finite set of intermediates; I is a finite set of terminals; $P = P_1 \cup P_2$, P_1 is a finite set of non terminal rules of the form $S \rightarrow S_1 \textcircled{$\nearrow$} a$, $S \rightarrow S_1 \textcircled{$\nwarrow$} b$, $S \rightarrow S_1 \textcircled{$\downarrow$} c$ where $S, S_1 \in V_1$, $a, b, c \in V_2$ and P_2 is a terminal rule of the form $S \rightarrow H$ where $S \in V_1$ and H is a hexagonal array over I; S is the start symbol; $\mathcal{L}$ is a set of intermediate languages corresponding to each one of the intermediate in V_2. These intermediate languages are regular, CF or CS string languages written in the appropriate arrowhead from. An arrowhead is written in the form $\{\ldots\langle v\rangle\ldots\}$ where $\langle v\rangle$ denotes the vertex and the arrowhead is written in the clockwise direction. A HKAG is called $(R : R)HKAG, (R : CF)HKAG, (R : CS)HKAG$ according as all the members of $\mathcal{L}$ is regular, atleast one of $\mathcal{L}$ is CF or atleast one of $\mathcal{L}$ is CS.*

Theorem 1. *Every $(R : X)HAL$, for $X \in \{R, CF, CS\}$ can be generated by HATPNS.*

Proof. Let G be the corresponding $(R : X)HAG$. Let $S \rightarrow H \circledast S'$ be the initial rule in P_1 and $\mathcal{L} = \{L_A/A \in I\}$ be the set of intermediate languages. Define for every L_A an arrowhead of similar type from A_1 to A_6. The parameters of the arrowhead will depend on the parameters of the hexagon in the input place. So firing the transition with catenation rules as labels will catenate the arrowhead to the hexagon. Let $H \circledast A_1 \circledast \cdots \circledast A_k \circledast \cdots \circledast A_l \circledast \cdots \circledast A_n$ be the string of arrowheads A_i and H, which derives the first array of the language. Let $A_k \circledast \cdots \circledast A_l$ be the substring which applied m times gives the m^{th} array of the language. The steps for constructing the HATPNS to generate the $(R : X)HAL$ would depend on the values of k, l, n. For $k = 1$ we have (i) $k = l = n$, (ii) $k = l < n$, (iii) $k < l = n$ and (iv) $k < l < n$. Similarly for $k > 1$, we have (v) $k = l = n$, (vi) $k = l < n$, (vii) $k < l = n$ and (viii) $k < l < n$. We give the steps for construction of HATPNS when $k < l < n$, $(k = 1)$. The other cases can be dealt similarly.

Step 1. Let p, p' be two places with H as a token, T is a transition with input places p, p' and output places p, p_1. The label of T is p. Let $i = 1$.

Step 2. Let t_i be a transition with input place p_i and label $H \circledast A_i$. If $i < l$, then the output place is p_{i+1} and goto next step. If $i = l$, then the output place is p_k and goto step 4.

Step 3. Let $i = i + 1$ and repeat step 2.

Step 4. Let t_{l+1} be a transition with input place p_k and label $H \circledast A_{l+1}$. If $l + 1 = n$, then the output places are p', P and goto step 7. If $l + 1 < n$, then the output place is p_{l+1} and put $i = l + 2$.

Step 5. Let t_i be a transition with input place p_{i-1} and label $H \circledast A_i$. If $i < n$, then the output place is p_i and goto next step. If $i = n$, then the output places are p', P and goto step 7.

Step 6. Let $i = i + 1$ and repeat step 5.
Step 7. $F = \{P\}$.
Step 8. END. □

Corollary 1. *Every language in $(R, Y)HKAL$ with $Y \in \{R, CF, CS\}$ can be generated by a HATPNS.*

Proof. It is known that $(R, Y)HKAL \subsetneq (R, Y)HAL$ [8]. By Theorem 1, any $(R, Y)HAL$ can be generated by a HATPNS. Thus every language in (R, Y) $HKAL$ can be generated by a HATPNS. □

Theorem 2. *For $X \in \{CF, CS\}$, $Y \in \{R, CF, CS\}$ the family $(X : Y)HAL$ cannot be generated by HATPNS.*

Proof. In $(CF : Y)HAG$ the rules in P_2 would have a sequence of catenation on H a certain number of times and follow it by another sequence of catenations the same number of times. If the Petri net structure has a subnet C_1 for the first sequence of catenations and another subnet C_2 for the second sequence of catenations then there would be no control on the number of times C_1 and C_2 get executed. Hence HATPNS cannot generate a $(CF : Y)HAL$. Since $(CF : Y)HAL \subsetneq (CS : Y)HAL$, HATPNS cannot generate a $(CS : Y)HAL$. □

5 Adjunct Hexagonal Array Token Petri Net Structure

In this section we introduce a variation of the previous model to generate hexagonal pictures known as Adjunct Hexagonal Array Token Petri Net Structure (AHATPNS) and compare its generative power with HATPNS, controlled table 0L/1L hexagonal array grammars [7] and E2DHCFG [9].

For any hexagon H, $|H|_{LU}$ arrowheads of thickness one with same parameters can be found in the direction $\nearrow$ (or its dual $\swarrow$). They are denoted by $lu_1, lu_2, \ldots, lu_{|H|_{LU}}$. An A_3 type arrowhead (A_4 type) can be joined at any of the $|H|_{LU} + 1$ positions subject to the conditions of an arrowhead catenation. The adjunction rule is a tuple $(H \textcircled{$\nearrow$} A_3 / H \textcircled{$\swarrow$} A_4, blu_i/alu_j)$, $1 \leq i, j \leq |H|_{LU}$, joining $A_3(A_4)$ into H either before lu_i or after lu_j.

$|H|_L$ arrowheads of thickness one with same parameters can be found in the direction $\downarrow$ (or its dual $\uparrow$). They are denoted by l_1, l_2, $\ldots$, $l_{|H|_L}$. An A_1 type arrowhead (A_2 type) can be joined at any of the $|H|_L + 1$ positions subject to the conditions of an arrowhead catenation. The adjunction rule is a tuple $(H \textcircled{$\downarrow$} A_1 / H \textcircled{$\uparrow$} A_2, bl_i/al_j)$, $1 \leq i, j \leq |H|_L$, joining $A_1(A_2)$ into H either before lu_i or after lu_j.

$|H|_{LL}$ arrowheads of thickness one with same parameters can be found in the direction $\nwarrow$ (or its dual $\searrow$). They are denoted by $ll_1, ll_2, \ldots ll_{|H|_{LL}}$. An $A_5(A_6)$ type arrowhead can be joined at any one of $|H|_{LL} + 1$ positions subject to the conditions of arrowhead catenation. The adjunction rule is a tuple $(H \textcircled{$\nwarrow$} A_5 / H \textcircled{$\searrow$} A_6, bll_i/all_j)$, $1 \leq i, j \leq |H|_{LL}$, joining $A_5(A_6)$ into H either before ll_i or after ll_j. Refer to Figs. 6 and 7 for positions in a hexagon.

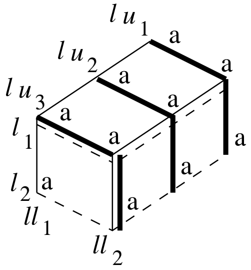


Fig. 6. Positions of adjunction of a hexagon in the directions ↗, ↖ and ↓

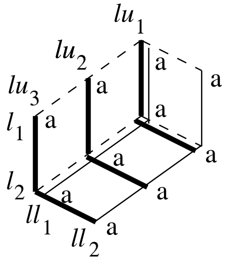


Fig. 7. The corresponding dual positions ↙, ↘, ↑

Note. In Figs. 6 and 7, l_i is represented by dotted lines, ll_i is represented by normal lines, lu_i is represented by thick lines.

5.1 Adjunction Rules as Labels

Let a transition t have ($H \circledast A$, position) a tuple as a label, where H is the hexagonal array in all its input places, A a predefined arrowhead, $\circledast$ is one of the six directions and position being any one of the positions shown in Figs. 6 and 7. Then firing the transition will join A into H in the position given as the second argument of the tuple and the resulting hexagon is put in the output places. The conditions required for catenation should be satisfied, otherwise the transition cannot fire. An example explaining the adjunction rule is given below.

Example 3.

$$\text{Let } H = \begin{array}{ccc} & a & \\ a & & a \\ & a & \\ a & & a \\ & a & \end{array}, \quad A_1 = \begin{array}{ccc} \mathbf{x} & & \mathbf{x} \\ & \mathbf{x} & \\ \mathbf{x} & & \mathbf{x} \\ & \mathbf{x} & \end{array}, \quad H_1 = \begin{array}{ccc} & a & \\ a & & a \\ & a & \\ \mathbf{x} & & \mathbf{x} \\ & \mathbf{x} & \\ \mathbf{x} & & \mathbf{x} \\ & \mathbf{x} & \\ a & & a \\ & a & \end{array}$$

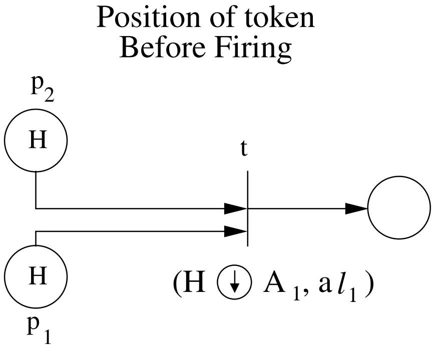


Fig. 8.

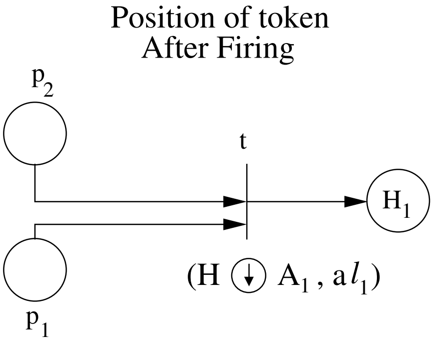


Fig. 9.

Definition 8. *If $C = (P, T, I, O)$ is a Petri net structure with hexagonal arrays of Σ^{**H} as initial markings $M_0 : P \to \Sigma^{**H}$, labels of transitions being adjunct rules and a finite set of final places $F \subseteq P$, then the Petri net structure is defined as Adjunct Hexagonal Array Token Petri Net Structure (AHATPNS).*

Definition 9. *If C is a AHATPNS then the language generated by the Petri net C is defined as*

$$L(C) = \{H \in \Sigma^{**H} / H \text{ is in } p \text{ for some } p \in F\}$$

With hexagonal 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 4. Let $C_1 = \begin{pmatrix} y \\ x \end{pmatrix}^3 \left\langle \begin{matrix} y \\ x \end{matrix} \right\rangle \begin{pmatrix} y \\ x \end{pmatrix}^3$ and $F = \{p_2\}$.

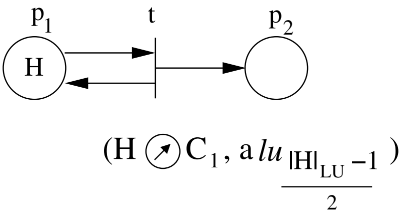


Fig. 10. *AHATPNS*

Firing t once yields

$$H = \begin{array}{c} b \\ a \quad b \\ a \quad a \quad b \\ \;\; a \quad a \quad b \\ a \quad a \quad a \\ \;\; a \quad a \quad b \\ a \quad a \quad a \\ \;\; a \quad a \quad b \\ a \quad a \quad a \\ \;\; a \quad a \quad b \\ a \quad a \\ a \end{array} \overset{t}{\Longrightarrow} \begin{array}{c} b \\ y \quad b \\ x \quad y \quad b \\ a \quad x \quad y \quad b \\ a \quad a \quad x \quad y \\ a \quad a \quad x \quad b \\ a \quad a \quad a \quad y \\ a \quad a \quad x \quad b \\ a \quad a \quad a \quad y \\ a \quad a \quad x \quad b \\ a \quad a \quad a \quad y \\ a \quad a \quad x \\ a \quad a \\ a \end{array}$$

The firing sequence t^2 joins the arrowhead $\begin{pmatrix} y \\ y \\ x \\ x \end{pmatrix}^3 \left\langle \begin{matrix} y \\ y \\ x \\ x \end{matrix} \right\rangle \begin{pmatrix} y \\ y \\ x \\ x \end{pmatrix}^3$ into H.

Theorem 3. *HATPNS $\subsetneq$ AHATPNS.*

Proof. Every arrowhead catenation rule is a special case of adjunction rule. The catenation rule $H\textcircled{\nearrow}A_3$ is equivalent to $(H\textcircled{\nearrow}A_3, blu_1)$. Similarly all catenation rules have an equivalent adjunction rule. Thus every HATPNL can be generated by some AHATPNS. The picture language generated by Example 4 cannot be generated by any HATPNS. Thus we have proper inclusion. □

We are now ready to compare AHATPNS with other models. For this, we recall the following definitions.

Definition 10. *A controlled table 0L hexagonal array model is a 4-tuple $(V, \mathcal{P}, H_0, C)$ where V is a finite alphabet; $\mathcal{P}$ is a finite set of tables $\{P_1, P_2, \ldots, P_k\}$ each table consisting of right up, left up, down (or the duals) rules of the form $a \to bc$, $a, b, c \in V$; H_0 is the axiom; C is a control language which may be regular, context-free or context-sensitive.*

By changing the rules in the tables to be context dependent we get hexagonal 1L array models with regular, CF or CS control.

Definition 11. *An extended 2D hexagonal context-free grammar (E2DHCFG) is $G = (V, \Sigma, P_{ur}, P_{ul}, P_d, P_{ll}, P_{lr}, P_u, H_0)$ where V is a finite set of symbols; $\Sigma \subseteq V$ is the set of terminal symbols; $P_{ur} = \{t_{ur}(i)/1 \leq i \leq m\}$; Each $t_{ur}(i)$ $(1 \leq i \leq m)$ called a UR table, is a set of CF rules of the form $A \to \alpha$, $A \in V \backslash \Sigma$, $\alpha \in V^*$ such that for any two rules $A \to \alpha$, $B \to \beta$ we have $|\alpha| = |\beta|$ where $|\alpha|$*

denotes the length of α*; Each of the other five components* $P_{ul}, P_d, P_{ll}, P_{lr}, P_u$ *is similarly defined;* $H_0 \subseteq \Sigma^{h**} \backslash \{\lambda\}$ *is a finite set of axiom arrays that are hexagonal arrays.*

Proposition 1. *The family AHATPNL is incomparable with the controlled table 0L/1L hexagonal array languages but not disjoint.*

Proof. The set of all p hexagons with alternate sides equal and of order 1 belong to T1LHAL with regular control [7]. This family can be generated by a AHATPNS. The set of all regular hexagons over a single letter with side 2^n can be generated by T0LHAG with CS control. This family cannot be generated by AHATPNS because a control on a firing sequence cannot be imposed in this model. Example 4 does not belong to T0LHAL since the development of the array is not along the edges. Thus the two families are incomparable but not disjoint. □

Proposition 2. *E2DHCFL and AHATPNL are not mutually disjoint.*

Proof. Example 4 gives a picture language which belongs to both E2DHCFL [9] and AHATPNL. □

6 Conclusion

Two models for generating hexagonal arrays have been defined and compared with some of the already existing models. These models are able to generate certain families of HAL. If some sort of control is defined on the sequence of firing, the other families of HAL can also be generated. It is worth examining the possibilities of defining a control over the firing sequences and obtain further results. It would also be interesting to define a general framework with arbitrary angles dividing equally a circle. Comparisons of controlled table 0L/1L hexagonal array models and E2DHCFG with HATPNS are kept for our future work.

References

1. Dersanambika, K.S., Krithivasan, K., Martin-Vide, C., Subramanian, K.G.: Local and recognizable hexagonal picture languages. International Journal of Pattern Recognition and Artificial Intelligence 19(7), 553–571 (2005)
2. Dersanambika, K.S., Krithivasan, K., Agarwal, H.K., Gupta, J.: Hexagonal contextual array P systems. Formal Models, Languages and Application. Series in Machine Perception Artificial Intelligence 66, 79–96 (2006)
3. Peterson, J.L.: Petri net theory and modeling of systems. Prentice-Hall, Englewood Cliffs (1981)
4. Lalitha, D., Rangarajan, K.: Adjunct array token Petri nets. In: Proceedings of International Conference on Operations Research Applications in Engineering and Management (ICOREM), pp. 431–445. Anna University, Tiruchirapalli (2009)
5. Lalitha, D., Rangarajan, K.: Column and row catenation Petri net systems. In: Proceeding of Fifth IEEE International Conference on Bio-Inspired Computing: Theories and Applications, pp. 1382–1387 (2010)

6. Middleton, L., Sivaswamy, J.: Hexagonal Image Processing: A Practical Approach. Springer, Heidelberg (2005)
7. Siromoney, G., Siromoney, R.: Hexagonal arrays and rectangular blocks. Computer Graphics and Image Processing 5, 353–381 (1976)
8. Subramanian, K.G.: Hexagonal array grammar. Computer Graphics and Image Processing 10, 388–394 (1979)
9. Subramanian, K.G., Geethalakshmi, M., Atulya, K., Nagar, L.S.K.: Hexagonal picture languages. In: Proceedings of the 5^{th} Asian Mathematical Conference, Malaysia, pp. 510–514 (2009)