Math. Appl. **3** (2014), 45–59
DOI: 10.13164/ma.2014.04

ADJUNCT HEXAGONAL ARRAY TOKEN PETRI NETS AND HEXAGONAL PICTURE LANGUAGES

THANGASAMY KAMARAJ, DORAISWAMY LALITHA,
DURAIRAJ GNANARAJ THOMAS, ROBINSON THAMBURAJ
AND ATULYA K. NAGAR

Abstract. Adjunct Hexagonal Array Token Petri Net Structures (AHPN) are recently introduced hexagonal picture generating devices which extended the Hexagonal Array Token Petri Net Structures . In this paper we consider AHPN model along with a control feature called inhibitor arcs and compare it with some expressive hexagonal picture generating and recognizing models with respect to the generating power.

1. Introduction

Hexagonal arrays and hexagonal patterns are known to occur in studies of picture processing and scene analysis [9–12]. In [10] the results of some experiments using COMPAX to produce isometric views of scenes of rectangular parellepipeds showed the generation of hexagonal arrays. In [12] hexagonal arrays on triangular grid are viewed as two-dimensional representation of three-dimensional blocks, and "perceptual twins" of pictures of given set of blocks. In biomedical image processing, it has been shown that a programmable cellular automaton with hexagonal structure is a worthy device for rapid processing of biomedical images [9]. In a chromosome analysis program [9], the circumscribing polygons associated with each image turn out to be hexagons. Since late seventies, formal models to generate or recognize the hexagonal pictures have been found in the literature [3,5–7,12,14] in the framework of pattern recognition and image analysis. Some of the classical formalisms to generate hexagonal arrays are Hexagonal Kolam Array Grammars (HKAG) [12] and its generalization Hexagonal Array Grammars (HAG) [13]. Sequential and parallel applications of rewriting rules and arrow head catenations are the common features of those models. Hexagonal Tile Rewriting Grammars [16] and Regional Hexagonal Tile Rewriting Grammars [5] are the recent hexagonal tiling based isometric grammar models, which have more generative capacity than HAG. Differently, Pure 2D Hexagonal Context Free Grammars [14, 15] and Hexagonal Prusa Grammars [6] are the simple yet expressive non-isometric formalisms, where parallel application of rewriting rules are attempted in the derivation of hexagonal pictures. On the

MSC (2010): primary 68Q42, 68Q45.
Keywords: Petri nets, hexagonal array tokens, adjunction, hexagonal grammars, hexagonal tiling systems.
We thank Liverpool Hope University for the support.

other hand, Hexagonal Tiling System (HTS) [3] equivalent to hexagonal online tessellation automata is a recognizing device for the class, Hexagonal Recognizable Languages (HREC), which involves the projection of languages belonging to the class, Hexagonal Local Languages (HLOC) defined by a finite set of hexagonal tiles.

Recently another hexagonal picture generating mechanism, Hexagonal Array Token Petri Net Structure (HPN) [7] has been evolved from string generating Petri nets [1, 4]. Petri net is one of the formal models used for analysing systems that are concurrent, distributed, and parallel. In HPN, hexagonal array tokens are used to simulate the dynamism of the net. In [7], the authors also introduced a generalization of this model, Adjunct Hexagonal Array Token Petri Net Structure (AHPN), incorporating adjunction operation, a variation in the position of arrowhead catenations. An AHPN model generates the same family of languages generated by some of the classes of HKAG and HAG.

With the purpose of gaining more generative power, we now consider an AHPN model along with a control feature, called inhibitor arcs like in [8], comparing it with some expressive hexagonal picture generating and recognizing models.

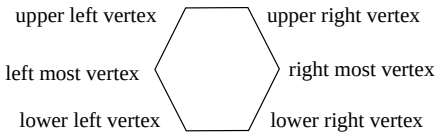
This paper is organized in the following manner. In Section 2, basic definitions of hexagonal arrays, Petri nets, and notions of Petri nets pertaining to hexagonal arrays are recalled. In Section 3, we recall the definition of AHPN in more general form and provide some illustrative examples. In Section 4 we compare AHPN with various hexagonal array grammars and also with HREC and HLOC with respect to the generative capacity.

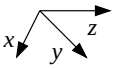
2. Preliminaries

In this section, we review some of the definitions of hexagonal arrays, Petri nets, and notions of Petri nets pertaining to hexagonal arrays. The following notations and definitions are mainly from [3–5, 12].

Let T be a finite alphabet of symbols. A hexagonal array of symbols of T is a hexagonal picture over T.

We consider hexagons of the type



For example, a hexagonal picture over the alphabet $\{a\}$ is $\begin{matrix} & a & & a & & a & \\ a & & a & & a & & a \\ & a & & a & & a & \end{matrix}$. With respect to a triad  of triangular axes x, y, z, the co-ordinates of each element of a hexagonal picture can be fixed [3].

The set of all hexagonal arrays over the alphabet T is denoted by T^{**H} and the set of all non-empty hexagonal arrays over T is denoted by T^{++H}. A non-empty hexagonal picture language L over T is a subset of T^{++H}.

For $p \in T^{++H}$, let $\hat{p}$ be the hexagonal array obtained by surrounding p with a special boundary symbol $\# \notin T$. For example,

$$\text{if} \quad p = \begin{array}{ccccc} & a & & a & \\ a & & a & & a \\ & a & & a & \end{array} \quad \text{then} \quad \hat{p} = \begin{array}{ccccccc} & & \# & & \# & & \# \\ & \# & & a & & a & & \# \\ \# & & a & & a & & a & & \# \\ & \# & & a & & a & & \# \\ & & \# & & \# & & \# \end{array}.$$

Given a picture $p \in T^{++H}$, let $|p|_x$ denote the number of elements in the border of p from the upper left vertex to the leftmost vertex in the direction $\swarrow$ called x direction, $|p|_y$ denotes the number of elements in the border of p from the upper right vertex to the right most vertex in the direction $\searrow$ called y direction and $|p|_z$ denote the number of elements in the border of p from the upper left vertex to the upper right vertex in the direction $\rightarrow$ called z direction. The triplet $(|p|_x, |p|_y, |p|_z)$ is called the size of a hexagonal picture p. The directions are fixed with the origin of reference as the upper left vertex, having co-ordinates (1, 1, 1). Let p_{ijk} denote the symbol in p, called a pixel, with the coordinates (i, j, k) where $1 \leq i \leq |p|_x$, $1 \leq j \leq |p|_y$, $1 \leq k \leq |p|_z$. Let $T^{(\ell,m,n)H}$ be the set of all hexagonal pictures of size (ℓ, m, n). A typical hexagonal array of size (ℓ, m, n) can be denoted by $[p_{ijk}]^{(\ell,m,n)H}$. Given a hexagonal picture p of size (ℓ, m, n), we denote by $B_{g,h,k}(p)$ the set of all hexagonal sub pictures of p of size (g, h, k), where $g \leq \ell$, $h \leq m$ and $k \leq n$. Each member of $B_{2,2,2}(p)$ is called a hexagonal tile. We denote the set of all hexagonal tiles contained in a picture $\hat{p}$ by $[[\hat{p}]]$.

The notions of non-convex hexagonal arrays called arrow heads, and arrow head catenations in six directions are adapted as in [3, 12]. An arrowhead is written in the form, $\{\cdots < y > \ldots\}$ where $< y >$ denotes the vertex, and the arrowhead is written in the clockwise direction. An arrowhead of thickness one is referred to as a unit arrowhead.

A 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 may contain a discrete number of marks called tokens. 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. In the hexagonal array generating Petri Net structure, hexagonal arrays over an alphabet J are used as tokens in some input places.

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 that 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 a number of tokens and all its inhibitor arc (if exists) input places have zero tokens.

3. Adjunct hexagonal array token Petri net structure

In this section, we recall the notions of Adjunct Hexagonal Array Token Petri Net Structure [7] with different adaptations and also in a more generalized form and give some examples.

Adjunction is a generalization of arrowhead catenation. In the upper right (UR) arrowhead catenation $H \textcircled{\nearrow} A$, the arrow head A is joined (catenated) to H after the unit UR arrowhead present in the border of H. But an UR adjunction can join the array A into array H after or before any unit UR arrowhead of H. Let H be a hexagonal array of size $(|H|_x, |H|_y, |H|_z)$ in J^{**} called host array; $A \subset J^{**}$ be an arrowhead language whose members, called adjunct arrow heads, have fixed thickness and varying length which depend on the corresponding size parameters of the host array H. For example, if A is an adjunct UR-arrowhead, $|A|_x$ (thickness) is fixed and the other two parameters $|A|_y, |A|_z$ (length) depend on the corresponding parameters $|H|_y, |H|_z$ of the host array H.

In the host array H, there are $|H|_x$ number of unit UR-arrowheads (LL-arrowheads) present, which we denote by $ur_1, ur_2, ur_3, \ldots, ur_{|H|x}$ $(ll_1, ll_2, \ldots, ll_{|H|x})$. Here, $ur_1(ll_1)$ denotes the border unit arrowhead and $ur_{|H|x}$ $(ll_{|H|x})$ denotes the innermost unit arrowhead in the UR (LL) direction. Any position between ur_i and ur_j, $i < j$, is called after ur_i (aur_i) or before ur_j (bur_j). An UR (LL) adjunct arrow head A can be joined into the host array H in $|H|_x + 1$ positions subject to the condition of arrowhead catenation. An UR(LL) adjunction rule is a tuple $(H, A, bur_i(bll_i)/aur_i(all_i))$, $1 \leq i \leq |H|_x$ joining A into H before ur_i (ll_i) or after ur_i (ll_i).

Similarly, in a host array H, $|H|_y$ a number of unit arrowheads in the LR (UL) direction are found. They are denoted by

$$lr_1, lr_2, \ldots, lr_{|H|y} \qquad (ul_1, ul_2, ul_3, \ldots, ul_{|H|y}).$$

An LR (UL) adjunct arrow head A can be joined into the host array H in $|H|_y + 1$ positions subject to the condition of arrowhead catenation. An LR (UL) adjunction rule is a tuple $(H, A, blr_i(bul_i)/alr_i(aul_i))$, $1 \leq i \leq |H|_y$ joining A into H before lr_i (ul_i) or after lr_i (ul_i).

Again, in a host array H, $|H|_z$ a number of unit arrowheads in the L (R) direction are found. They are denoted by $l_1, l_2, \ldots, l_{|H|z}$ $(r_1, r_2, r_3, \ldots, r_{|H|z})$. An L (R) adjunct arrow head A can be joined into the host array H in $|H|_z + 1$ positions subject to the condition of arrowhead catenation. An $L(R)$ adjunction rule is a tuple $(H, A, bl_i(br_i)/al_i(ar_i))$, $1 \leq i \leq |H|_z$ joining A into H before l_i (r_i) or after l_i (r_i).

Figure 2 (a) shows all the unit arrowheads in the UR, LR and L directions for the hexagon in Figure 1. The thick lines show the unit UR arrowheads. The dotted lines show the unit LR arrowheads. The normal lines show the unit L arrowheads. Figure 2 (b) shows the duals of the arrow heads given in Figure 2 (a).

$$\begin{array}{ccccc} & a & a & a & \\ a & & a & & a \\ & a & & a & \\ a & & a & & a \\ & a & & a & \end{array}$$

Figure 1. Hexagonal array.

Definition 3.1. An Adjunct Hexagonal Array Token Petri Net Structure (AHPN) is a five tuple $P =< J, C, M_0, \rho, F >$ where J is a given alphabet,

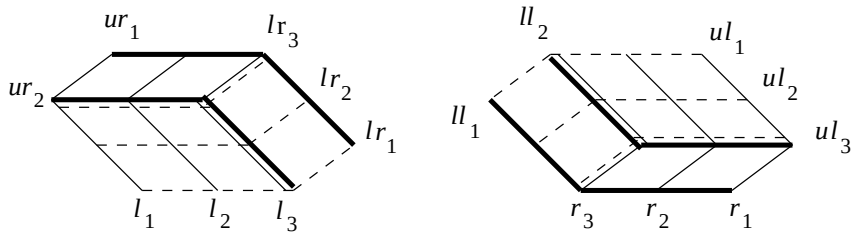


Figure 2. (a) Positions of Adjunctions in the directions of UR, LR and L, (b) Positions of Adjunctions in the dual directions.

$C =< Q, \mathcal{T}, I, O >$ is a Petri net structure [7] with tokens as hexagonal arrays over J and $\mathcal{T}$ may contain transitions with inhibitor arcs, $M_0 : Q \to J^{**}$, is the initial marking of the net, $\rho : \mathcal{T} \to L$, a mapping from the set of transitions to the set of labels where some of the transitions may have arrowhead adjunction rules as the labels and $F \subset Q$, is a finite set of final places. In AHPN, the types of transitions which can be enabled and fired are similar to that of HPN [7] except the type (iii) where labels of transitions are arrowhead adjunction rules instead of arrowhead catenation rules.

Definition 3.2. If P is an AHPN, then the hexagonal picture language generated by P is defined as $L(P) = \{X \in J^{**H}/X$ is in the place q for some q in $F\}$. Starting with hexagonal arrays (tokens) over a given alphabet as initial marking, all possible sequences of transitions are fired. Set of all arrays created in final places F is called the language generated by the AHPN. We denote by AHPL the family of hexagonal picture languages generated by Adjunct Hexagonal Array Token Petri Net Structures

Example 3.3. Consider the AHPN, $P_1 =< J, C, M_0, \rho, F >$ where $J = \{0, 1\}$, $C = (Q, \mathcal{T}, I, O)$, $Q = \{q_1, q_2, q_3\}$, $\mathcal{T} = \{t_1, t_2, t_3\}$, $I(t_1) = \{q_1\}$, $I(t_2) = \{q_2\}$, $I(t_3) = \{q_3\}$, $O(t_1) = \{q_2\}$, $O(t_2) = \{q_3\}$, $O(t_3) = \{q_1\}$, M_0 is the initial marking: the array S is in q_1 and there is no array in q_2 and q_3, $\rho(t_1) = (H, A_1, aur_1)$, $\rho(t_2) = (H, A_2, alr_1)$, $\rho(t_3) = (H, A_3, al_1)$ and $F = \{q_1\}$. The arrays used in the net are defined as follows:

$$S = \begin{matrix} & 1 & 1 \\ 1 & 1 & 1 \\ & 1 & 1 \end{matrix}, \quad A_1 = 1(0)^{|H|_z - 2} < 0 > (0)^{|H|_y - 2}1,$$

$A_2 = 1(0)^{|H|_x - 2} < 0 > (0)^{|H|_z - 2}1$ and $A_3 = 1(0)^{|H|_y - 2} < 0 > (0)^{|H|_x - 2}1$. The Petri net graph is given in Figure 3.

Initially t_1 is the only enabled transition. Firing of t_1 adjoins a UR arrowhead $\begin{matrix} 1 & 0 \\ & 1 \end{matrix}$ (taken from the language A_1) after the unit upper right arrowhead ur_1 (in other words: just inside the border UR arrowhead) of array S and puts the derived array in the place q_2, making t_2 enabled. Firing t_2 adjoins a LR arrowhead $\begin{matrix} & 1 \\ 0 & \\ 1 & 0 \end{matrix}$ (taken from the language A_2) after the unit LR arrowhead lr_1 (in other

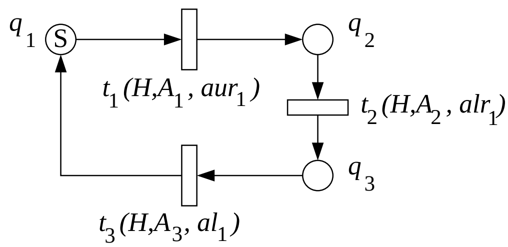


Figure 3. Petri net to generate equal sized hexagonal pictures of a's.

words: just inside the border LR arrowhead) of the array H in q_2 and puts the derived array in q_3 making t_3 enabled. Firing t_3 adjoins a L arrowhead $\begin{array}{ccc} & & 1 \\ & 0 & \\ 0 & & \\ & 0 & \\ & & 1 \end{array}$ (taken from the language A_3) after the unit L arrowhead l_1 (in other words: just inside the border L arrowhead) of the array H in q_3 and puts the derived array in q_1.

When the sequence of transitions $t_1t_2t_3$ fires, the hexagonal array that reaches the output place q_1 is shown as

$$\begin{array}{ccccc} & 1 & & 1 & \\ 1 & & 1 & & 1 \\ & 1 & & 1 & \end{array} \overset{t_1}{\Longrightarrow} \begin{array}{cccccc} & & 1 & & 1 & \\ & 1 & & 0 & & 1 \\ 1 & & 1 & & 1 & \\ & 1 & & 1 & & \end{array} \overset{t_2}{\Longrightarrow} \begin{array}{ccccccc} & & 1 & & 1 & & \\ & 1 & & 0 & & 1 & \\ 1 & & 1 & & 0 & & 1 \\ & 1 & & 0 & & 1 & \\ & & 1 & & 1 & & \end{array} \overset{t_3}{\Longrightarrow} \begin{array}{ccccccccc} & & 1 & & 1 & & 1 & & \\ & 1 & & 0 & & 0 & & 1 & \\ 1 & & 0 & & 1 & & 0 & & 1 \\ & 1 & & 0 & & 0 & & 1 & \\ & & 1 & & 1 & & 1 & & \end{array}.$$

Firing the sequence $(t_1t_2t_3)^2$ generates the output array as

$$\begin{array}{ccccccccccccc} & & & 1 & & 1 & & 1 & & 1 & & & \\ & & 1 & & 0 & & 0 & & 0 & & 1 & & \\ & 1 & & 0 & & 0 & & 0 & & 0 & & 1 & \\ 1 & & 0 & & 0 & & 1 & & 0 & & 0 & & 1 \\ & 1 & & 0 & & 0 & & 0 & & 0 & & 1 & \\ & & 1 & & 0 & & 0 & & 0 & & 1 & & \\ & & & 1 & & 1 & & 1 & & 1 & & & \end{array}.$$

Firing sequence $(t_1t_2t_3)^{n-1}$ gives the n^{th} array of the language. The language L_1 generated by Petri net P_1 is the set of all equal sized hexagonal pictures over $\{0, 1\}$ with 1's in the border and at the center and other elements are 0's.

It should be noted that the sizes of the arrowheads taken from the languages A_1, A_2, A_3 are not fixed, but vary depending on the sizes of the hexagons H in the input place of the transition, so that the condition for catenation is satisfied.

Example 3.4. If in Example 3.3, $J = \{a\}$, $S = \begin{matrix} & a & a & \\ a & & a & & a \\ & a & & a & \end{matrix}$,

$$A_1 = (a)^{|H|_z - 1} < a > (a)^{|H|_y - 1}, A_2 = (a)^{|H|_x - 1} < a > (a)^{|H|_z - 1}$$

and $A_3 = (a)^{|H|_y - 1} < a > (a)^{|H|_x - 1}$, then the language L_2 of hexagonal pictures over $\{a\}$ with equal sides is generated.

Example 3.5. Consider the Adjunct Array token Petri net structure $P_3 =< J, C, M_0, \rho, F >$, where $J = \{a\}$, the Petri net structure is $C = (Q, \mathcal{T}, I, O)$ with $Q = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$, $\mathcal{T} = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$, $I(t_1) = \{p_1, p_2\}$, $I(t_2) = \{p_3\}$, $I(t_3) = \{p_4\}$, $I(t_4) = \{p_5\}$, $I(t_5) = \{p_6\}$, $I(t_6) = \{p_7\}$, $I(t_7) = \{p_2\}$, $O(t_1) = \{p_3\}$, $O(t_2) = \{p_4\}$, $O(t_3) = \{p_5\}$, $O(t_4) = \{p_2, p_6\}$, $O(t_5) = \{p_7\}$, $O(t_6) = \{p_1, p_1\}$. Since the weight of the arc from t_6 to p_1 is two, $O(t_6)$ is containing p_1 two times. This implies that firing t_6 will deposit two copies of the hexagonal array in p_1.

p_1 is an inhibitor input for both t_5 and t_7, p_6 is an inhibitor input for t_6 and p_7 is an inhibitor input for t_1. M_0, the initial marking is the array S in p_1, p_2, p_8.

$\rho : \mathcal{T} \to L$ is defined as follows: $\rho(t_1) = p_2$, $\rho(t_2) = (H, A_1, aur_1)$, $\rho(t_3) = (H, A_2, alr_1)$, $\rho(t_4) = (H, A_3, al_1)$, $\rho(t_5) = \lambda$, $\rho(t_6) = \lambda$, $\rho(t_7) = \lambda$, $F = \{p_8\}$. The Petri net graph is given in Figure 4.

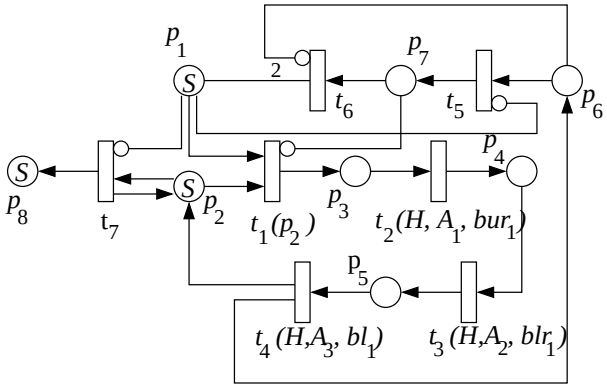


Figure 4. Petri net with inhibitor arcs to generate hexagons of a's of size $(2^n, 2^n, 2^n)$.

The arrays used are $S = \begin{matrix} & a & a & \\ a & & a & & a \\ & a & & a & \end{matrix}$, $A_1 = \begin{pmatrix} a \\ a \end{pmatrix}^{|H|_z - 1} \left\langle \begin{matrix} a \\ a \end{matrix} \right\rangle \begin{pmatrix} a \\ a \end{pmatrix}^{|H|_y - 1}$, $A_2 = \begin{pmatrix} a \\ a \end{pmatrix}^{|H|_x - 1} \left\langle \begin{matrix} a \\ a \end{matrix} \right\rangle \begin{pmatrix} a \\ a \end{pmatrix}^{|H|_z - 1}$ and $A_3 = \begin{pmatrix} a \\ a \end{pmatrix}^{|H|_y - 1} \left\langle \begin{matrix} a \\ a \end{matrix} \right\rangle \begin{pmatrix} a \\ a \end{pmatrix}^{|H|_x - 1}$.

To start with, only t_1 is enabled. Firing the sequence of transitions $t_1 t_2 t_3 t_4$ results in a hexagonal picture of a's of size $4 \times 4 \times 4$ in p_2 and p_6. At this stage both t_5 and t_7 are enabled. Firing the sequence $t_1 t_2 t_3 t_4 t_7$ puts a hexagonal picture of size $4 \times 4 \times 4$ in p_8. Firing t_5 pushes the array to p_7, emptying p_6. In this position

t_6 is enabled. Firing t_6 puts two copies of same array in p_1. Since at this stage there are two tokens in p_1, the sequence $t_1t_2t_3t_4$ has to fire two times to empty p_1. The firing of the sequence $t_5t_6(t_1t_2t_3t_4)^2t_7$ puts a hexagonal picture of a's of size 8×8 in p_8. The inhibitor input p_1 makes sure that a square of size 6×6 does not reach p_8. This AHPN generates the language L_3 of hexagonal pictures of a's of size $(2^n, 2^n, 2^n)$, $n \geq 1$.

Example 3.6. The AHPN $P_4 =< J, C, M_0, \rho, F >$ with $J = \{1, 2, 3\}$, $F = \{p, p_1\}$ given in Figure 6, where $S = \begin{array}{ccccc} & & 1 & 1 & \\ & 1 & 3 & 1 & \\ 1 & & 2 & & 1 \\ & 1 & & 1 & \end{array}$, $A\begin{pmatrix}1\\1\end{pmatrix}\left\langle \begin{matrix}2\\3\end{matrix} \right\rangle \begin{pmatrix}1\\1\end{pmatrix}$, generates the language L_4 of hexagonal arrays of size $(2n+1, 2, 2)$, $n \geq 1$, with interior elements along the x-direction forming the pattern 2^n3^n and elements in the border are 1's. A typical picture in this language is given by Figure 5.

$$\begin{array}{ccccccc} & & & 1 & 1 & & \\ & & 1 & & 3 & & 1 \\ & 1 & & 3 & & 1 & \\ 1 & & 2 & & 1 & & \\ & 1 & & 2 & & 1 & \\ & & 1 & & 1 & & \end{array}$$

Figure 5. An hexagonal array in the language L_4.

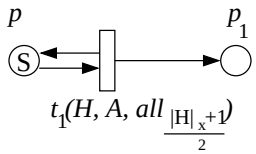


Figure 6. Petri net to generate the language L_4.

4. Comparative results

We now recall the definitions of some of the expressive parallel rewriting hexagonal grammar devices and compare them with AHPN for the hierarchy.

Definition 4.1. A pure 2D hexagonal context-free grammar ($HCFG_{pure}$) [15] is a construct $G = (T, P_{ur}, P_{ul}, P_{lr}, P_{ll}, P_l, P_r, \mathcal{M}_0)$, where T is a finite set of 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 context-free rules of the form $a \to \alpha$, $a \in T$, $\alpha \in T^*$ such that any two rules of the form $a \to \alpha$, $b \to \beta$ in t_{ur_i}, we have $|\alpha| = |\beta|$ where $|\alpha|$ denotes the length of α, each of the other five components P_{ul}, P_{lr}, P_{ll}, P_l and P_r is similarly defined, $\mathcal{M}_0 \subseteq T^{++H}$ is a finite set of hexagonal arrays or arrow heads, named axioms. Derivations are defined as follows: For any two hexagonal arrays H_1, H_2,

we write $H_1 \Rightarrow H_2$ if H_2 is obtained from H_1 by rewriting all the symbols in an unit arrowhead of H_1 by rules of a relevant table in $P_{ur} \cup P_{ul} \cup P_{lr} \cup P_{ll} \cup P_l \cup P_r$. $\overset{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$. The hexagonal picture language $L(G)$ generated by G is the set $\{H/H_0 \overset{*}{\Rightarrow} H \in T^{**H}$, for some $H_0 \in \mathcal{M}_0\}$. We denote by $HCFL_{pure}$ the family of all hexagonal picture languages generated by pure 2D hexagonal context-free grammars. To augment the expressive power of $HCFG_{pure}$, we associate a control language to the sequence of table rules to be used like in [15].

Definition 4.2. A pure 2D hexagonal context-free grammar with regular control ($(R)HCFG_{pure}$) is a tuple $G_r = (G, \Gamma, C)$, where

(1) G is a $HCFG_{pure}$,
(2) Γ is the control alphabet, the set of labels of the rule tables in $P_{ur} \cup P_{ul} \cup P_{lr} \cup P_{ll} \cup P_l \cup P_r$,
(3) $C \subseteq \Gamma^*$ is the regular control associated with the G_r.

If $H \in T^{**H}$ and $H_0 \in \mathcal{M}_0$, H is derived from H_0 in G_r by means of a control word $w = w_1 w_2 \cdots \in C$, in symbols $C \Rightarrow_w H$, if H is obtained from H_0 by applying the table rules as in the sequence of tables $w_1 w_2 \ldots$. The language $L(G)$ generated by $(R)HCFG_{pure}$ G_r is the set of pictures $\{H/H_0 \Rightarrow_w H \in T^{++H}$ for some $w \in C\}$. We denote by $(R)HCFL_{pure}$ the family of hexagonal picture languages generated by pure 2D hexagonal context-free grammars with regular control.

Example 4.3. The language L_2 of hexagonal pictures over $\{a\}$ with equal sides in Example 3.4 can be generated by $(R)HCFG_{pure} < G, \{ul, ll, r\}, ((ul)(ll)r)^* >$ where $G =< \{a\}, \{\lambda\}, \{\lambda\}, \{\lambda\}, \{ul\}, \{ll\}, \{r\}, \mathcal{M}_0 >$, and $\mathcal{M}_0 = \left\{ \begin{array}{ccc} & a & a \\ a & & a \\ & a & a \end{array} \right\}$, $ul = \{a \ \nwarrow \ a\}$, $ll = \{a \ \swarrow \ a\}$ and $r = \{a \ \rightarrow \ a\}$. The derivation of hexagonal picture of size (3, 3, 3) from the axiom hexagonal array is given as follows

$$\begin{array}{ccccccccccc} & & a & a & & & & a & a & & \\ & a & & & a & & a & & a & & \\ a & & a & & & a & & a & & a & \\ & a & & a & & \Rightarrow & a & & a & & \\ a & & a & & & & & a & & a & \\ & a & & a & & & a & & a & & \\ & & a & a & & & & a & a & & \end{array} \Rightarrow \begin{array}{ccccc} & a & a & & \\ a & & a & & a \\ & a & & a & \\ a & & a & & a \\ & a & & a & \end{array} \Rightarrow \begin{array}{ccccc} & a & a & & \\ a & & a & & a \\ & a & & a & \\ a & & a & & a \\ & a & & a & \end{array}.$$

To increase the generating power of $(R)HCFG_{pure}$ further, we refine the definition of this class of grammars like that of pure 2D rectangular picture grammars considered in [14] but adapted as in [2].

Definition 4.4. An extended pure 2D hexagonal context-free grammar with regular control $(R)HCFG_{expure}$ is a pure 2D hexagonal context free grammar with regular control, with the alphabet $T = T_f \cup T_c$ where T_f is the alphabet of final symbols defining the hexagonal pictures and T_c is a set of control symbols which are involved only in the process of derivation and they do not appear in the final picture. We denote by $(R)HCFL_{expure}$ the family of hexagonal picture languages generated by Extended pure 2D hexagonal context-free grammars with regular control.

Example 4.5. The language L_4 in Example 3.6 can be generated by a $(R)HCFG_{expure} < G, \{ur_1, ur_2\}, C >$ where $G =< T_f \cup T_c$, $\{ur_1, ur_2\}$, $\{\lambda\}$, $\{\lambda\}$, $\{\lambda\}$, $\{\lambda\}$, $M_0 >$ and $T_f = \{1, 2, 3\}$, $T_c = \{x, y\}$, $M_0 = \begin{Bmatrix} & 1 & 1 & \\ & z & y & 1 \\ z & x & z & \\ & 1 & z & \end{Bmatrix}$, $ur_1 = \{z \nearrow 1z, x \nearrow 2x\}$, $ur_2 = \{z \nearrow 1z, y \nearrow 3y\}$. The regular control language, $C = ((ur_1)(ur_2))^* (ur_1')(ur_2')$ where $ur_1' = \{z \nearrow 1, x \nearrow 2\}$ and $ur_2' = \{z \nearrow 1, y \nearrow 3\}$. The sample derivation of a hexagonal picture of size (5, 2, 2) in this language is shown below.

$$\begin{array}{ccccccccc}
\begin{array}{ccc} & 1 & 1 \\ z & y & 1 \\ z & x & z \\ & 1 & z \end{array}
& \Rightarrow &
\begin{array}{ccc} & 1 & 1 \\ z & y & 1 \\ z & x & z \\ 1 & 2 & z \\ & 1 & 1 \end{array}
& \Rightarrow &
\begin{array}{ccc} & 1 & 1 \\ z & y & 1 \\ 1 & 3 & z \\ z & x & 1 \\ 1 & 2 & z \\ & 1 & 1 \end{array}
& \Rightarrow &
\begin{array}{ccc} & 1 & 1 \\ z & y & 1 \\ 1 & 3 & z \\ z & x & 1 \\ 1 & 2 & z \\ & 1 & 1 \end{array}
\end{array}$$

$$\Rightarrow
\begin{array}{ccc} & 1 & 1 \\ z & y & 1 \\ 1 & 3 & z \\ 1 & 2 & 1 \\ 1 & 2 & 1 \\ & 1 & 1 \end{array}
\Rightarrow
\begin{array}{ccc} & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 2 & 1 \\ 1 & 2 & 1 \\ & 1 & 1 \end{array}.$$

Proposition 4.6. *The family $(R)HCFL_{pure}$ properly includes the family $HCFL_{pure}$ languages.*

Proof. Every $HCFG_{pure}$ G can be considered $(R)HCFG_{pure}$ with regular control language $C = \Gamma^*$, where Γ is the control alphabet. The language L_2 of hexagonal pictures over $\{a\}$ with equal sides is shown to be in $(R)HCFL_{pure}$, in Example 4.3. But L_2 cannot be generated by any $HCFG_{pure}$ as application of the ul, ll, r table rules without any control leads to the generation of hexagonal pictures over $\{a\}$ with unequal sides. □

Theorem 4.7. *The family $(R)HCFL_{pure}$ is properly contained in AHPL.*

Proof. Let L be the hexagonal picture language generated by the pure 2D hexagonal context-free grammar $G = (T, P_{ur}, P_{ul}, P_{lr}, P_{ll}, P_l, P_r, \mathcal{M}_0)$, with a regular control language over the set of labels, say $(\ell_1, \ell_2, \ldots, \ell_m)$.

If a UR table, say t_{ur_i}, rewrites the arrowhead ur_i then every symbol in ur_i has to be rewritten using the rules of the table in parallel. But application of an UR-adjunction rule $(H, A, bur_i/aur_i)$ will also have the same result as that of table rule t_{ur_i}. In other words, application of a UR table is equivalent to a UR-adjunction. Hence, for every UR table t_{ur_i}, a corresponding UR-adjunction rule $(H, A, aur_i/bur_i)$ can be defined. Similarly, for every X table t_{x_j}, a corresponding X-adjunction rule $(A, B, ax_j/bx_j)$ can be defined, where $X \in \{LR, L, LL, UL, R\}$ and $x \in \{lr, l, ll, ul, r\}$. If it is assumed that the derivation $M_0 \overset{w}{\Rightarrow} M$ yields a hexagonal array M of the language, where w is a regular control word $\ell_1 \ell_2 \ldots \ell_m$,

then the adjunct hexagonal array token Petri net structure P can be constructed as follows.

Let p_0 be a place with a hexagonal 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 of the sequence $t_1 t_2 \dots t_m$ will have the same effect as applying the rules $\ell_1, \ell_2, \dots, \ell_m$ in that order once. The firing of the the sequence $(t_1 t_2 \dots t_m)^n$ generates the same array which is obtained by applying the set of tables in the control word $(\ell_1 \ell_2 \dots \ell_m)^n$. Thus, the Petri net P constructed will generate the same language L as generated by the RP2DHCFG, G. In other words, The family $(R)HCFL_{pure}$ is included in AHPL.

For the strict inclusion, we consider the language L_4 in Example 3.6. Suppose there exists an $(R)HCFG_{pure} < G, \{ur_1, ur_2\}, ((ur_1)(ur_1))^* >$ where $G =< \{1,2,3\}, \{ur_1, ur_2\}, \{\lambda\}, \{\lambda\}, \{\lambda\}, \{\lambda\}, \{\lambda\}, \mathcal{M}_0 >$ and $M_0 = \begin{array}{ccccc} & & 1 & & 1 \\ & 1 & & 3 & & 1 \\ 1 & & 2 & & 1 & \\ & 1 & & 1 & & \end{array}$, $ur_1 = \{1 \nearrow 11, 2 \nearrow 22\}$, $ur_2 = \{1 \nearrow 11, 3 \nearrow 33\}$. Then there is a possibility of applying the table rule ur_1 to the unit arrowhead $\begin{array}{ccc} 1 & & 1 \\ & 1 & \end{array}$ to get $\begin{array}{ccccc} & & 1 & & 1 \\ 1 & & 1 & & 1 & \\ & & & 1 & & \end{array}$ as a part of the picture. But such a picture is not in L_4. □

Now by Proposition 4.6 and Theorem 4.7, we can state the following result immediately.

Corollary 4.8. *The family AHPL properly contains the family $HCFL_{pure}$.*

Theorem 4.9. (i) *$(R)HCFL_{expure}$ and AHPL are not disjoint.*
(ii) *There exists a language in AHPL which cannot be generated by any $(R)HCFG_{expure}$.*

Proof. The AHPL language L_4 considered in Example 3.4 is shown to be in $(R)HCFL_{expure}$ (see Example 4.5). Hence, $(R)HCFL_{expure}$ and AHPL are not disjoint.

The AHPL language L_3, in Example 3.5 cannot be generated by any $(R)HCFG_{expure}$ G, since a hexagonal picture of size $(2^n, 2^n, 2^n)$ cannot be derived from that of size $(2^{n-1}, 2^{n-1}, 2^{n-1})$ because there should exists a procedure to apply the rules exactly $2(n-1)$ times on UR, LR, L arrowheads of the picture. But the number of applications of rules to be applied cannot be a function of the dimension of the hexagonal picture. Hence, the second part of the theorem is proved. □

Definition 4.10. A Hexagonal Prusa Grammar (HG_{prusa}) [6] is a tuple, $G =< N, T, P, S >$, where N is the set of non terminals, T is set of terminals, P is set

of productions, and $S \in N$ is the start symbol. The hexagonal picture language $L(G, C)$ over T for every $C \in N$ is defined by the following recursive rules.

(1) Terminal rule: If $C \to X$ is in P, and $X \in (T^{++H} \cup T^{+})$, then $X \in L(G, C)$.
(2) Mixed rule: Let $C \to X$ be a production in P with $X \in \cup(N \cup T)^{(\ell', m', n')H}$, $1 \leq \ell' \leq \ell$, $1 \leq m' \leq m$ and $1 \leq n' \leq n$ and Q_{ijk} $(1 \leq i \leq \ell, 1 \leq j \leq m, 1 \leq k \leq n)$ be the pictures such that
   (i) if $X_{ijk} \in T$ then $Q_{ijk} = X_{ijk}$,
   (ii) if $X_{ijk} \in N$ then $Q_{ijk} \in L[G, X]$.

If $Q = [Q_{ijk}]^{(\ell', m', n')H}$ is defined through string catenation (or) arrow head catenation, then $Q \in L[G, C]$.

Here T^{+} denotes set of all non empty strings in any of the 3 directions parallel to the triangular axes. The set $L[G, C]$ contains all and only pictures that can be obtained by applying a finite sequence of rules (i) and (ii). The hexagonal language $L[G]$ generated by the grammar G is defined to be the language $L[G, S]$. HL_{prusa} is the class of all languages generated by these grammars.

Informally, the rules may be either terminal array rules or mixed array rules involving terminals and non terminals. In both types of rules, the right-hand side array may be a hexagon or arrowhead or string in one of the three possible triangular axis directions.

Example 4.11. The language L_1 given in Example 3.3 can be generated by a HG_{prusa}, $G =< N, T, P, S >$ where $N = \{S, H, A, B, C, A', B', C'\}$, $T = \{0, 1\}$ and

$$P = \left\{ S \to \begin{matrix} & 1 & \\ C & H & 1 \\ & 1 & \end{matrix}, \; A \to \begin{matrix} 1 & A \\ & 1 \end{matrix} \Big/ 1, \; B \to \begin{matrix} & 1 \\ 1 & B \end{matrix} \Big/ 1, \; C \to \begin{matrix} C & 1 \\ & 1 \end{matrix} \Big/ 1, \right.$$
$$\left. H \to \begin{matrix} & 0 & \\ C' & H & 0 \\ & 0 & \end{matrix} \Big/ 1, \; A' \to \begin{matrix} 0 & A' \\ & 0 \end{matrix} \Big/ 0, B \to \begin{matrix} & 1 \\ 0 & B' \end{matrix} \Big/ 0, \; C \to \begin{matrix} C' & 0 \\ & 0 \end{matrix} \Big/ 0 \right\}.$$

The derivation of the first two pictures in this language is as follows. Since $1 \in L[G, A] \cap L[G, C] \cap L[G, B] \cap L[G, H]$,

$$\begin{matrix} & 1 & 1 & \\ 1 & 1 & 1 & 1 \\ & 1 & 1 & \end{matrix} \in L[G, S] \quad \text{by the mixed rule} \quad S \to \begin{matrix} & 1 & \\ C & H & 1 \\ & 1 & \end{matrix} \begin{matrix} A \\ \\ B \end{matrix}.$$

Since $\begin{matrix} & 0 & \\ 0 & 1 & 0 \\ & 0 & \end{matrix} \in L[G, H]$, $\begin{matrix} 1 & 1 \\ & 1 \end{matrix} \in L[G, A]$, $\begin{matrix} & 1 \\ 1 & 1 \end{matrix} \in L[G, B]$ and $\begin{matrix} & 1 \\ 1 & \\ & 1 \end{matrix} \in L[G, C]$, we have $\begin{matrix} & 1 & 1 & 1 & \\ 1 & 0 & 0 & 1 & \\ & 1 & 0 & 1 & 0 \\ & 1 & 0 & 0 & 1 \\ & & 1 & 1 & 1 \end{matrix} \in L[G, S]$.

Theorem 4.12. *The families AHPL and HL_{prusa} are incomparable but not disjoint.*

Proof. The AHPL language L_1 considered in Example 3.3 is shown to be generated by a HG_{prusa} (see Example 4.11). Hence, $L_1 \in AHPL \cap HL_{prusa}$. For incomparability, we consider the AHPL language L_3 in Example 3.5. It cannot be generated by any hexagonal Prusa grammar as the rules of the grammar cannot derive UR, LR, L arrowheads of thickness 2^{n-1}, $n > 1$, exactly, to get the hexagonal pictures of size $(2^n, 2^n, 2^n)$. On the other hand, Consider the language L_5, which consists of palindromic left arrowheads of thickness one over $T = \{a, b\}$.

A typical element of L_5 is $\begin{matrix} & & & a & \\ & & a & & \\ & b & & & \\ a & & & & \\ & b & & & \\ & & a & & \\ & & & a & \end{matrix}$. L_5 can be generated by the HG_{prusa}, $G =< N, T, P, S >$ with $N = \{S\}$, $T = \{a, b\}$,

$$P = \left\{ S \to \begin{matrix} & a & \\ S & & \\ & a & \end{matrix}, \quad S \to \begin{matrix} & b & \\ S & & \\ & b & \end{matrix}, \quad S \to \begin{matrix} a & & a \\ & a & \end{matrix} \Big/ \begin{matrix} a & & a \\ & b & \end{matrix} \Big/ \begin{matrix} b & & b \\ & a & \end{matrix} \Big/ \begin{matrix} b & & b \\ & b & \end{matrix} \right\}.$$

This language cannot be derived by any AHPN as from the definition of AHPN, the effect of applying an arrowhead adjunction rule is either increasing the thickness or maintaining the same thickness but not on the length of the arrowhead. □

Definition 4.13. The projection by mapping $\pi : \Gamma \to T$, (where Γ and T are two alphabets), of hexagonal picture p is the picture $p' \in T^{**H}$ such that $p'(i, j, k) = \pi(p(i, j, k))$ for all $1 \le i \le \ell$, $1 \le j \le m$ and $1 \le k \le n$, where (ℓ, m, n) is the size of the hexagonal picture.

Definition 4.14. Let Γ be a finite alphabet. A hexagonal picture language $L \subseteq \Gamma^{**H}$ is called local if there exists a finite set θ of hexagonal tiles over $\Gamma \cup \{\#\}$ such that $L = \{p \in \Gamma^{**H} / B_{2,2,2}(\hat{p}) \subseteq \theta\}$.

The family of local hexagonal picture languages will be denoted by HLOC.

Definition 4.15. A hexagonal picture language $L \subseteq T^{**H}$ is called recognizable if there exists a local hexagonal picture language L' over an alphabet Γ and a mapping $\pi : \Gamma \to T$ such that $L = \pi(L')$.

The family of recognizable hexagonal picture languages will be denoted by HREC [3].

Definition 4.16. A Hexagonal Tiling System (HTS) [3] is a 4-tuple $< T, \Gamma, \pi, \theta >$ where T and Γ are two finite set of symbols, $\pi : \Gamma \to T$ is a projection and θ is a set of hexagonal tiles over the alphabet $\Gamma \cup \{\#\}$. A hexagonal picture language $L \subseteq \Gamma^{**H}$ is hexagonal tiling recognizable if there exists a hexagonal tiling system $< T, \Gamma, \pi, \theta >$ such that $L = \pi(L(\theta))$.

The family of hexagonal tiling recognizable languages is denoted by HL_{hts}. It is easy to see that HREC is exactly HL_{hts}.

Example 4.17. The language L_2 of hexagonal pictures over $\{a\}$ with equal sides in Example 3.4 is in HREC, since there exist a hexagonal tiling system

(T, Γ, θ, π) where $T = \{a\}$, $\Gamma = \{0, 1\}$ and

$$\theta = \left[\left[\begin{array}{ccccccccccccccccccc}
 & & & & & \# & \# & \# & \# & \# & \# & & & & & & & & \\
 & & & & \# & & 1 & & 0 & & 0 & & 0 & & 0 & & \# & & \\
 & & & \# & & 0 & & 0 & & 1 & & 0 & & 0 & & \# & & & \\
 & & \# & & 0 & & 1 & & 0 & & 0 & & 1 & & 0 & & \# & & \\
 & \# & & 0 & & 0 & & 0 & & 0 & & 0 & & 1 & & 0 & & \# & \\
\# & & 0 & & 0 & & 1 & & 0 & & 0 & & 0 & & 0 & & 1 & & \# \\
 & \# & & 0 & & 0 & & 0 & & 0 & & 0 & & 1 & & 0 & & \# & \\
 & & \# & & 0 & & 1 & & 0 & & 0 & & 1 & & 0 & & \# & & \\
 & & & \# & & 0 & & 0 & & 1 & & 0 & & 0 & & \# & & & \\
 & & & & \# & & 1 & & 0 & & 0 & & 0 & & \# & & & & \\
 & & & & & \# & & \# & & \# & & \# & & \# & & & & &
\end{array}\right]\right],$$

$\pi(1) = \pi(0) = a$.

Theorem 4.18. (i) *AHPL and HREC are not disjoint.*
(ii) *There exists a language in AHPL which is not in HLOC.*
(iii) *There exists a language in AHPL which is not in HREC.*

Proof. The AHPL language L_2 considered in Example 3.4 is shown to be in HREC in Example 4.17 but not in HLOC [3]. This proves the first two parts of the theorem. For the third part: the language L_4 in Example 3.6 cannot be recognized by any HTS as the interior elements along the x-direction forming the pattern $2^n 3^n$ can only be recognized by a context free device. □

5. Conclusion

In this paper we have considered a variant class of Hexagonal Array Token Petri Net Structure with arrowhead adjunction rules as labels of transitions, along with a control feature called inhibitor arcs. We compared this model with some of the expressive hexgonal grammar models: $(R)HCFG_{pure}$, $(R)HFCG_{expure}$, HG_{prusa}, and also with HREC and HLOC. We have shown that AHPN have a generative capacity higher than $(R)HCFG_{pure}$ but incomparable and non-disjoint with other models. The non-empty intersection of the hexagonal picture languages generated by this model with other models clearly suggests that this model can generate a wide variety of digitized hexagonal pictures and patterns. The application of this model in picture processing tasks and pattern recognition should be investigated further.

References

[1] H. G. Baker, *Petri Net Languages, Computation Structures Group Memo 68*, Project MAC, MIT, Cambridge, Massachusetts, 1972.

[2] M. M. Bersani, A. Frigeri and A. Cherubini, *On some classes of 2D languages and their relations*, in: J. K. Aggarwal et al. (eds.), IWCIA 2011, Lecture Notes in Compuer Sceince **6636**, Springer, Heidelberg, 2011, 222–234.

[3] K. S. Dersanambika, K. Krithivasan, C. Martin-Vide and K. G. Subramanian, *Local and recognizable hexagonal picture languages*, Int. J. Pattern Recogn. **19** (2005), 853–871.

[4] M. Hack, *Petri Net languages, Computation Structures Group Memo 124*, Project MAC, MIT, 1975.

[5] T. Kamaraj and D. G. Thomas, *Regional hexagonal tile rewriting grammars*, in: R. P. Barneva et al. (eds.), IWCIA 2012, Lecture Notes in Computer Science **7655**, Springer, Heidelberg, 2012, 181–195.
[6] T. Kamaraj and D. G. Thomas, *Hexagonal Prusa grammar model for context-free hexagonal picture languages*, in: G. S. S. Krishnan et al. (eds.), ICC3 2013, AISC **246**, Springer, India, 2014, 305–311.
[7] D. Lalitha, K. Rangarajan and D. G. Thomas, *Petri net generating hexagonal arrays*, in: J. K. Aggarwal et al. (eds.), IWCIA 2011, Lecture Notes in Computer Science **6636**, Springer, Heidelberg, 2011, 235–247.
[8] D. Lalitha, K. Rangarajan and D. G. Thomas, *Rectangular arrays and Petri nets*, in: R. P. Barneva et al. (eds.), IWCIA 2012, Lecture Notes in Computer Science **7655**, Springer, Heidelberg, 2012, 166–180.
[9] K. Preston Jr., *Applications of cellular automata in biomedical image processing*, in: E. Haga (ed.), Computer Techniques in Biomedicine and Medicine, Auerbach, Philadelphia, 1973.
[10] V. S. N. Reddy and R. Narasimhan, *Some experiments in scene analysis and scene regeneration using COMPAX*, Computer Graphics and Image Processing **1** (1972), 386–393.
[11] A. Rosenfeld and J. L. Pfaltz, *Distance functions on digital pictures*, Pattern Recogn. **1** (1968), 33–61.
[12] G. Siromoney and R. Siromomey, *Hexagonal arrays and rectangular blocks*, Computer Graphics and Image Processing **5** (1976), 353–381.
[13] K. G. Subramanian, *Hexagonal array grammars*, Computer Graphics and Image Processing **10** (1979), 388–394.
[14] K. G. Subramanian, M. Geethalakshmi, A. K. Nagar, S. K. Lee, *Two-dimensional picture grammar models*, in: Proceedings of the 2nd European Modelling Symposium, EMS 2008, IEEE, 2008, 263–267.
[15] K. G. Subramanian, Rosihan M. Ali, M. Geethalakshmi, A. K. Nagar, *Pure 2D picture grammars and languages*, Discrete Appl. Math. **157** (2009), 3401–3411.
[16] D. G. Thomas, F. Sweety and T. Kalyani, *Results on hexagonal tile rewriting grammars*, in: G. Bebis et al. (Eds.), International Symposium on Visual Computing, Part II, Lecture Notes in Computer Science **5359**, Springer-Verlag, Berlin, Heidelberg, 2008, 945–952.

Thangasamy Kamaraj, Department of Mathematics, Sathyabama University, Chennai – 600 119, India
e-mail: kamaraj_mx@yahoo.co.in

Doraiswamy Lalitha, Department of Mathematics, Sathyabama University, Chennai - 600 119, India
e-mail: lalkrish_24@yahoo.co.in

Durairaj Gnanaraj Thomas, Department of Mathematics, Madras Christian College, Tambaram, Chennai – 600 059, India
e-mail: dgthomasmcc@yahoo.com

Robinson Thamburaj, Department of Mathematics, Madras Christian College, Tambaram, Chennai – 600 059, India
e-mail: robin.mcc@gmail.com

Atulya K. Nagar, Department of Mathematics and Computer Science, Liverpool Hope University, Hope Park, Liverpool L16 9JD, UK
e-mail: nagara@hope.ac.uk