Abstract

A new model to generate triangular arrays using petri net structure has been defined. The catenation of an arrowhead to an equilateral triangle results in a similar equilateral triangle. This concept has been used in Triangular Array Token Petri Net Structure (TATPNS). Comparisons with other triangular array models have been made.

Keywords: Triangular picture languages, array tokens, petri nets, arrowhead catenations.

1. Introduction

Triangular arrays and triangular patterns are found in the literature on picture processing and scene analysis. Image generation can be done in many ways in formal languages.

Petri net model to generate rectangular arrays has been introduced in [4] and petri net model to generate hexagonal picture languages in [3]. Motivated by this concept we have introduced a petri net model to generate triangular picture languages.

* Corresponding author.
E-mail address: kuberal@yahoo.com
In this model triangular 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 arrays 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 Triangular Array Token Petri Net Structure (TATPNS). One application for such a generation are in the field of tiling patterns and generation of kolam patterns.

2. Triangular Array Token Petri Nets

In this section we give preliminary definitions of petri net and give the notations used.

A petri net is one of several mathematical models for the description of distributed systems. A petri net is a directed bipartite graph, in which the nodes represent transitions (i.e., events that may occur, signified by bars) places (i.e., conditions, signified by circles). The directed arcs from places to a transition denote the pre-conditions and the directed arcs from the transition to places denote the post-conditions (signified by arrows). Graphically, places in a petri net 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 an abstract sense relating to a petri net diagram, a transition of a petri net may fire whenever these are sufficient, tokens at the start of all input arcs, when it fires, it consumes these tokens, and places tokens at the end of all output arcs. Transitions can be labeled with elements of an alphabet. So that the firing sequence corresponds to a string over the alphabet. A labeled petri net generates a language. Petri net to generate string languages is also found in [1].

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 \( m \geq 0 \), \( P \cap T = \emptyset \), \( I : T \rightarrow P^\infty \) is input function from transitions to bags of places and \( O : T \rightarrow P^\infty \) is the output function from transitions to bags of places.

Definition 2. A petri net marking is an assignment to tokens to the places of a petri net. The tokens are asked to define the execution of a petri net. The number and position of tokens may change during the execution of a petri net.

In this paper triangular arrays over an alphabet are used as tokens.

Definition 3. An inhibitor arc from a place \( p_i \) to a transition \( t_j \) has a small circle in the place of arrow in regular arcs. This means the transitions \( t_j \) is enabled only if \( p_i \) has no tokens. A transitions is enabled only if all its regular inputs have tokens and all its inhibitor inputs have zero tokens.

3. Basic Notations

Let \( \Sigma \) be a finite non empty set of symbols. \( \Sigma_T^{\infty} \) denotes the arrays made up of element of \( \Sigma \). The size of any triangular array is defined by its parameters. Let \( T \) be a triangular array of size \( n \). Let \( T' \) be left or right or upper cap catenations. Add \( T' \) with \( T \) the size of the picture \( n+2 \).

The petri net model defined have has places and transitions connected by directed arcs. Triangular arrays over an alphabet are taken as tokens to be distributed in places. Variation in firing rules and labels of the transition are listed out below.

Firing rules in TATPNS

We define three different types of enabled transition in TATPNS. The pre and post condition for firing the
transition in all the three cases are given below.

(i) A transition without any label will fire only if all the input places have the same triangular array as a token. Then or firing the transition arrays from all the input places are removed and put in all its output places.

\[ \text{Fig. 1. Position of arrays before firing} \]

\[ \text{Fig. 2. Position of array after firing} \]

(ii) 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.

\[ \text{Fig. 3. Transition with label before firing} \]

\[ \text{Fig. 4. Transition with label after firing} \]

**Definition 4.** If \( C = (P, T, I, O) \) is a petri net structure with triangular array over of \( \Sigma_T^m \) as initial markings, \( M_0 : P \rightarrow \Sigma_T^m \), label of at least one transition being cap catenation rule and a finite set of final places \( F \subseteq P \), then the petri net structure \( C \) is defined as Triangular Array Token Petri Net Structure (TATPNS).

**Definition 5.** If \( C \) is a TATPNS then the language generated by the Petri net \( C \) is defined as

\[ L(C) = \{ T \in \Sigma_T^m / T \text{ is in } P \text{ for some } P \in F \} \]

with arrays of \( \Sigma_T^m \) in some places as initial marking all possible sequences of transitions are fixed. The set of all arrays in the final places \( F \) is called the language generated by \( C \).
Example 1. $\Sigma = \{x, \bullet\}$, $F = \{P_1\}$

(iii) Let a transition $t$ have $T(\bullet)A$ as a label where $\bullet$ is any one of the three directions $(\uparrow, \downarrow, \rightarrow)$. $T$ is the triangular array in all the input places and $A$ is a predefined arrowhead. Then firing the transition will catenate $A$ with $T$ 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.

Transition with catenation rule before firing

If $T = \begin{array}{c} \text{a} \\ \text{a} \end{array}$, the size of $T$ is 2, firing $t_1$ adds upper cap $A_1$ we get $T_1 = \begin{array}{c} \text{a} \\ \text{a} \\ \text{a} \end{array}$, the size of $T_1$ is 2+2 = 4 where $S = \begin{array}{c} \bullet \\ x \\ x \end{array}$, $A_1 = \begin{bmatrix} x & x \\ x & \bullet \\ x & \bullet \\ x & \bullet \end{bmatrix}$, $A_2 = \begin{bmatrix} x & x \\ x & \bullet \\ x & \bullet \end{bmatrix}$, $A_3 = \begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ x \\ x \\ x \end{bmatrix}$

Firing $t_1$ puts an array in $p_2$ making $t_2$ enabled. Firing $t_2$ puts an array in $p_3$ making $t_3$ enabled. Firing $t_3$ puts an array in $p_1$. The firing sequence $(t_1 t_2 t_3)^n$, $n > 0$, puts a triangular spiral of size $6n-3$ in $P_1$. The language generated by this TATPNS is a set of triangular spirals. When the transitions $t_1$, $t_2$ and $t_3$ fire the array that reaches the output places is shown below.
The language generated by this TATPNS is triangular spirals of size $6n - 3$, $n > 0$.

4. Comparison Results

In this section we recall the definition of Triangular array Grammar (TAG) [8] and define the Triangular Kolam Array Grammar (TKAG) and compare the generative power of TATPNS with $(X : Y)TAL$, $(R : Y)TKAL$ where $X$, $Y \in \{R, CF, CS\}$.

**Definition 6.** A Triangular Array Grammar (TAG) is $G = (N, I, T, P, S, L)$ where $N$, $I$ and $T$ are finite non empty sets of non terminals, intermediates and terminals respectively.

$S \in N$ is the start symbol. For each $A$ in $I$, $I_A$ is an intermediate language which is regular, context-free or context-sensitive string language written in the appropriate cap form. A cap is written in the form $\{ \ldots <v> \ldots \}$ where $<v>$ denoted the vertex and the cap in the clockwise direction.

$L = \{L_A / A \in I\}$, $P = P_1 \cup P_2$ is a finite non empty set of production where $P_1$ consists of initial rules of the following forms.

1) $S \rightarrow T(△)S'$
2) $S \rightarrow T(□)S'$
3) $S \rightarrow T(▲)S'$

where $S' \in N$, $S' \neq S$ and $T$ is a triangular array over $T$. $G$ is regular of the rules of $P_2$ are of the forms:

1) $S_1 \rightarrow A(\bullet)S_2$
2) $S_1 \rightarrow A(\square)S_2$
3) $S_1 \rightarrow A(\uparrow)S_2$
4) $S_1 \rightarrow A(\bigtriangleup)S_2$

Furthermore, if an initial rule of $P_1$ is of the form $(r)$, $r = 1, 2, 3$ then $P_2$ does
not contain any rule of the form \( r \) and \( r+1 \) if \( r = 1 \) and \( (r+1) \) and \( (r+3) \) if \( r = 2, 3 \).

G is CF if the rules of \( P_2 \) are of the form:

\[ S_1 \rightarrow \alpha_1 \alpha_2 \ldots \alpha_{k-1} \alpha_k \] (\( k \geq 1 \)) and \( \alpha_j \) denotes any one of the three cap catenations (\( 1 \leq j \leq k-i \)).

G is CS if the rules of \( P_2 \) are of the form:

\[ \beta \bigotimes S_1 \bigotimes \delta \rightarrow \beta \bigotimes \gamma \bigotimes \delta \text{ or } \beta \bigotimes S_2 \rightarrow \beta \bigotimes \gamma \]

or \( S_1 \bigotimes \delta \rightarrow \gamma \bigotimes \delta \text{ or } S_1 \rightarrow \gamma \) where \( S_1 \in N \) and \( S_1 \neq S \) and

\( \beta, \gamma, \delta \) are of the form \( \alpha_1 \ldots \alpha_k \) with \( \alpha_i \in (N\setminus\{S\}) \cup I \), (\( 1 \leq i \leq k \)).

In particular G is called \((X : R)\) TAG or \((X : CF)\) TAG or \((X : CS)\) TAG for \( X \in \{R, CF, CS\} \) according as all two intermediate language are regular or at least one of them is CF or at least one of them is CS.

**Derivations proceed as follows:**

First, an initial rule of \( P_1 \) is applied and then rules of \( P_2 \) are applied sequentially as in string grammars until all the non-terminals are replaced, resulting in a string of the form \( T A_1 \ldots A_n \), where \( A_i \in I \) (\( 1 \leq i \leq n \)).

In the second phase of derivation \( A_1 \) is replaced by an cap from \( L_{A_1} \) and catenated to the triangular array \( T \) according to the cap catenation symbol between \( T \) and \( A \). This is continued until \( A_n \) is replaced, resulting in a triangular array of terminals. The length of the cap is determined by the condition for cap catenations.

**Definition 7.** For \( X, Y \in \{R, CF, CS\} \), the \((X, Y)\) triangular array language \(((X : Y) \text{TAL})\) generated by the \((X : Y)\) TAG. \( G \) is \( L(G) = \{ T / T S \}_{G}^{\ast} \), \( T \) is a triangular array.

**Example 2.** Let \( G = (\Sigma, N, S, R) \) be a Triangular Tile Rewriting Grammar, where \( \Sigma = \{\bullet, x\}, N = \{S\} \), and \( R \) consists of the rules:

\[
S \rightarrow x
\]

\[
S \rightarrow \left\{ S' \cdot x \cdot S' \cdot S' \cdot x \cdot x \right\}
\]

A picture in \( L(G) \) is,

![Picture](attachment:image.png)

Size 8

It is know that the language \( L(G) \) is generated by the grammar \((R : CF)\) TAG, \( G = \{N, I, \{\bullet, x\}, P_1 \cup P_2, S_1, S_2, T\} \)

\( N = \{S_1, S_2\} \)

\( I = \{A_1, A_2, A_3\} \)

\( T \rightarrow x \)

\( P_1 = \{S_1 \rightarrow T \uparrow S_2\} \)

\( P_2 = \{S_2 \rightarrow A_1 \bigotimes A_2 \bigotimes A \bigotimes S_2, S_2 \rightarrow A \bigotimes A \bigotimes A \} \)
LA1 = \{<x> • <x> • <x>, n ≥ 2\} 
LA2 = \{• <x> • <x> • <x> • <x>, n ≥ 2\} 
LA3 = \{• <x> • <x> • <x> • <x> • <x> • <x>, n ≥ 2\}

**Definition 8.** A Triangular Kolam Array Grammar (TKAG) is a 5-tuple \((V, I, P, S, L)\) where 
\(V = V_1 \cup V_2\), \(V_1\) is a finite set of non-terminals and \(V_2\) is a finite set of intermediates, \(I\) is a finite set of terminates; 
\(P = P_1 \cup P_2\), \(P_1\) is a finite set of non-terminal rules of the form 
\(S \rightarrow S_1 \ a, S \rightarrow S_1 \ b, S \rightarrow S_1 \ c\), where \(S, S_1 \in V_1, a, b, c \in V_2\) and \(P_2\) is a terminal rule of the form \(S \rightarrow T\) where \(S \in V\) and \(T\) is a triangular array over \(I\). \(S\) is the start symbol, \(L\) 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 form. An arrowhead is written in the form \{ ... < v > ... \} where \(< v >\) denotes the vertex and the arrowhead is written in the clockwise direction. A TKAG is called \((R : R)\) TKAG, \((R : CF)\) TKAG, \((R : CS)\) TKAG according as all the members of \(L\) is regular, at least one of \(L\) is CF or at least one of \(L\) is CS.

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

**Proof.** A Triangular Array Grammar (TAG) is \(G = (N, I, T, P, S, L)\) where \(N, I\) and \(T\) are finite non empty sets of non terminals, intermediates and terminals respectively. \(S \in N\) is the start symbol. For each \(A \in I, I_A\) is an intermediate language which is regular, context-free or context-sensitive string language written in the appropriate cap form. Let \(P = P_1 \cup P_2\) be a finite non empty set of production. Let \(S \rightarrow T\) be the initial rule in \(P_1\) and \(L = \{LA / A \in I\}\) be the set of intermediates languages. Define for every \(L_A\) an arrowhead of similar type from \(LA_1\) to \(LA_3\).

In TAG the derivation is as follows: Starting with \(S\) the non terminals rules are applied without any restriction; as in string grammar, till all the non terminals are replaced then \(A_1\) is replaced by an cap catenation from \(LA_1\) and catenated to the triangular array \(T\) according to the cap catenation symbol between \(T\) and \(A_1\). This is continued until \(A_n\) is replaced, resulting in a triangular array of terminals. Construction of TATPNS for the case when \(S \rightarrow T(A_1)\), where \(A_1\) is the intermediate. For the other case the construction is similar. Define the array \(A_1\) corresponding to the intermediate language \(L_A\). Let \(T_1\) be the start place \(p_1\) as token. Have a transition \(t_1\) with upper cap catenation rule \(T_1(A_1)\) as a label. Let \(p_1\) be the input place of \(t_1\). The length of the cap is determined by the condition for cap catenations.

\(A_1\) is the array that reaches the input place \(p_1\) of the transition \(t_1\) during the course of the execution of the net. Let \(p_2\) be the output place for the transition \(t_1\). The array \(A_2\) is defined similar to the intermediate language generated by \(L_A\). Have a transition \(t_2\) with the right cap catenation rule \(T_2(A_2)\) as a label. Let \(p_2\) be the input place of \(t_2\). \(A_2\) is the array that reaches the input place \(p_2\) of the transition \(t_1\) during the course of the execution of the net. Let \(p_3\) be the output place for the transition \(t_2\). Similarly \(p_3\) be the input place of \(t_2\), have a transition \(t_3\) with the left cap catenation rule \(T_3(A_3)\) as a label. Let \(p_1\) be the output place for the transition \(t_3\). First time the sequence \(t_1t_2t_3\) is executed, the triangular array \(T_2\) is put in \(p_1\). Let \(F = \{p_1\}\) be the final set of places. The firing sequence \((t_1t_2t_3)^n, n>0\) in \(p_1\). Thus \(\{T_n/n>0\}\) of the triangular array is the language generated.

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

**Proof.** In \((CF : Y)\) TAG the rules in \(P_2\) would have a sequence of catenation on \(T\) 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 sequences 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 TATPNS cannot generate a \((CF : Y)\) TAL. Since \((CF : Y)\) TAL \(\subset (CS : Y)\) TAL, TATPNS cannot generate a \((CS : Y)\) TAL.
Conclusion

Triangular Array Token Petri Net Structure generates triangular arrays. Three models for generating triangular arrays have been defined and compared with some of the already existing models. These models are able to generate certain families of TAL. If some sort of control is defined on the sequence of firing, the other families of TAL can also be generated.

References

1. Peterson JL, *Petri Net theory and modeling of systems*, Prentice-Hall; Englewood Cliffs; 1981.
2. Lalitha D, Rangarajan K, Column and row catenation petri net systems, in Proceedings of Fifth IEEE International Conference on Bio-Inspired Computing: Theories and Applications; 2010, p. 1382-1387.
3. Lalitha D, Rangarajan K, Thomas DG, Petri net generating hexagonal arrays, J.K. Aggarwal et al. (eds.) IWClA 2011, LNCS 6636, p. 235-247, 2011, Springer-Verlag, Berlin, Heidelberg.
4. Lalitha D, Rangarajan K, Thomas DG, Rectangular arrays and petri nets, R.P. Baineva et al. (eds.) IWClA 2012, LNCS 7655, pp. 166-180, 2012, Springer-Verlag, Berlin, Heidelberg.
5. Middleton L, Sivaswamy J, *Hexagonal image processing : A practical approach*, Springer; Heidelberg; 2005.
6. Subramanian KG, Hexagonal array grammar, *Computer Graphics and Image Processing*, 1979; 10; 388-394.
7. Subramanian KG, Geethalakshmi M, Anulya K. Nagar, L.S.K, Hexagonal picture languages, in Proceedings of the 5th Asian Mathematical Conference, Malaysia; 2009, p. 510-514.
8. Kuberal S, Kalyani T, Thomas DG, Triangular tile rewriting grammars and triangular picture languages, in International Conference on Mathematical Computer Engineering (ICMCE 2013), VIT University, Chennai Campus, 2013.