



3D-Array Token Petri Nets Generating Tetrahedral Picture Languages

T. Kalyani¹ , K. Sasikala² , D. G. Thomas³ , Thamburaj Robinson⁴ ,
Atulya K. Nagar⁵ , and Meenakshi Paramasivan⁶ 

¹ Department of Mathematics, St. Joseph's Institute of Technology,
Chennai 119, India

kalphd02@yahoo.com

² Department of Mathematics, St. Joseph's college of Engineering,
Chennai 119, India

sasikalaveerabadran2013@gmail.com

³ Department of Science and Humanities, (Mathematics Division),
Saveetha School of Engineering, SIMATS, Chennai 602105, India

dgthomasmcc@yahoo.com

⁴ Department of Mathematics, Madras Christian College, Chennai 59, India

robinson@mcc.edu.in

⁵ Department of Computer Science, Liverpool Hope University, Hope Park,
Liverpool L169JD, UK

nagara@hope.ac.uk

⁶ Department of Computer Science, University of Trier, Trier, Germany

meena_maths@yahoo.com

Abstract. The study of two-dimensional picture languages has a wide application in image analysis and pattern recognition [5, 7, 13, 17]. There are various models such as grammars, automata, P systems and Petri Nets to generate different picture languages available in the literature [1–4, 6, 8, 10–12, 14, 15, 18]. In this paper we consider Petri Nets generating tetrahedral picture languages. The patterns generated are interesting, new and are applicable in floor design, wall design and tiling. We compare the generative power of these Petri Nets with that of other recent models [9, 16] developed by our group.

Keywords: Petri Net · Tetrahedral tiles · P systems

1 Introduction

The art of tiling plays an important role in human civilization. A two-dimensional pattern generating model called pasting system was introduced in the literature which glues two square tiles together at the edges. Later on two isosceles right angled triangular tiles are pasted together at the glueable edges and a new pasting system called triangular tile pasting system was introduced in [3].

Petri Nets are mathematical models introduced to model dynamic systems [15]. Tokens represented by black dots are used to simulate the dynamic activity

of the system. Array token Petri Nets are models which generate array languages [12]. Array Token Petri nets have applications in the following areas namely character recognition, generation and recognition of picture patterns, tiling pattern and kolam patterns. Arrays are used as token. The transitions are associated with catenation rules. Firing of transitions catenate arrays to grow in bigger size.

The area of membrane computing is a new computability model called P system introduced by Gh. Păun inspired by the functioning of living cells. Ceterchi et al. [4] proposed a theoretical model of P -system called Array Rewriting P -system for generating two-dimensional patterns. Motivated by these studies a three-dimensional pattern generating model called tetrahedral tiling pasting system and tetrahedral tile pasting P system were introduced in [9] by gluing two tetrahedral tiles at the glueable edges. In the literature the studies on membrane computing generating picture languages is very limited. We have used membrane computing to generate 3D Tetrahedral picture languages, in which we can generate both rectangular and non rectangular 3D pictures like stars, triangles, rhombuses, hexagons, octagons and some kolam patterns which are some of the interesting patterns.

In this paper we introduce 3D-Array token Petri Nets generating three-dimensional tetrahedral picture languages (3D-TetATPN) and this model is compared with K -Tabled Tetrahedral Tile Pasting System (K -TTTPS), Tetrahedral Tile Pasting P System (TetTPPS), Regular Tetrahedral Array Languages (RTAL) and Basic Puzzle Tetrahedral Array Languages (BPTAL) for generative powers. The patterns generated by the Petri Nets are useful in floor design, wall design and tiling.

2 Preliminaries

In this section we recall the notion of tetrahedral tiles, K -TTTPS and TetTPPS

Definition 1. [9] *A tetrahedral tile is a polyhedral which has four vertices, four faces and six edges. Each face is an equilateral triangle. f_4 is the base of the tetrahedron(Fig. 1).*

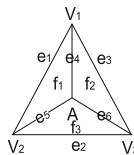
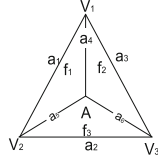
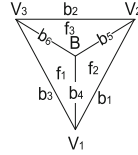


Fig. 1. A Tetrahedron.

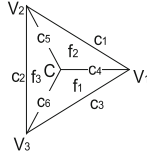
We consider tetrahedral tiles of the following four types, named as



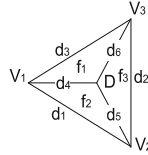
A-tetrahedral tile



B-tetrahedral tile



C-tetrahedral tile



D-tetrahedral tile

f_4 is the base $V_1V_2V_3$ of the tetrahedral tile.

Definition 2. [9] A K -Tabled Tetrahedral Tile Pasting System (K -TTTPS) is a 4-tuple $M = (\Gamma, E, P, t_0)$, where Γ is a finite set of tetrahedral tiles of the forms A and B . E is a set of edge labels of base of tetrahedral tiles A and B . P is a finite set of tables $\{T_1, T_2, \dots, T_K\}$ where T_1, T_2, \dots, T_K ($k \geq 1$) are finite sets of pasting rules. t_0 is the axiom pattern.

A tiling pattern t_{i+1} is generated from a pattern t_i in k stages. In each stage, the rules of the table T_i ($i = 1, 2, \dots, k$) are applied in parallel to the boundary edges of the pattern obtained in the previous stage. When all the rules in P are applied one after the other in succession the pattern t_{i+1} is generated from t_i . i.e. $t_i \Rightarrow t_{i+1}$. We write $t_0 \xrightarrow{*} t_j$ if $t_0 \Rightarrow t_1 \Rightarrow t_2 \Rightarrow \dots \Rightarrow t_j$. The collection of all patterns generated by K -TTTPS derived from the axiom t_0 using the pasting rules of the system M is denoted by $T(M) = \{t_j \in \Gamma^{***} : t_0 \xrightarrow{*} t_j / j \geq 0\}$, where Γ^{***} represents the set of all three-dimensional tetrahedral patterns obtained by gluing tetrahedral tiles of Γ .

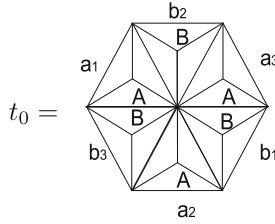
The family of all three-dimensional patterns generated by K -TTTPS is denoted as $\mathcal{L}(K\text{-TTTPS})$.

Example 1. A one-tabled Tetrahedral Tile Pasting System, generating a sequence of three-dimensional patterns whose boundaries are hexagons and stars alternatively is given below:

$$M = (\Gamma, E, P, t_0) \text{ where } \Gamma = \left\{ \begin{array}{c} \triangle \\ \text{A} \end{array} \begin{array}{c} \triangle \\ \text{B} \end{array} \right\}$$

$$E = \{a_1, a_2, a_3, b_1, b_2, b_3\}; P = \{T_1\};$$

$$T_1 = \{(a_1, b_1), (a_2, b_2), (a_3, b_3), (b_1, a_1), (b_2, a_2), (b_3, a_3)\}$$



The first three members of $T(M)$ are shown in Fig. 2.

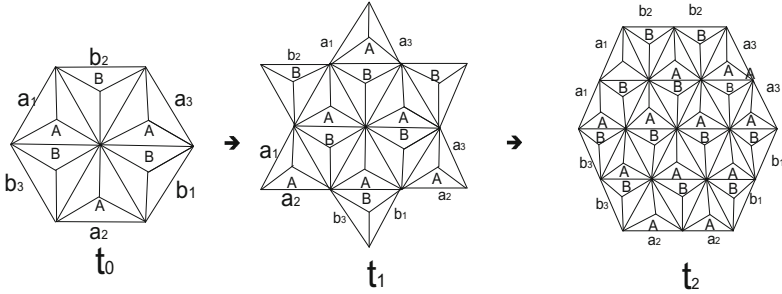


Fig. 2. Hexagon and Star polyhedral.

Definition 3. [9] A Tetrahedral Tile Pasting P system (TetTPPS) $\Pi = (\Gamma, \mu, F_1, \dots, F_m, R_1, R_2, \dots, R_m, i_0)$ where Γ is a finite set of labeled tetrahedral tiles; μ is a membrane structure with m membranes, labeled in an one-to-one way with $1, 2, \dots, m$; F_1, F_2, \dots, F_m are finite sets of three-dimensional picture patterns over tiles of Γ associated with the m regions of μ ; R_1, R_2, \dots, R_m are finite sets of pasting rules of the type $(t_i, (x_i, y_i), tar)$, $1 \leq i \leq n$ associated with the m regions of μ and i_0 is the output membrane which is an elementary membrane.

The computation process in TetTPPS is defined as, to each 3D-Picture pattern present in the region of the system, the pasting rule associated with the respective region should be applied in parallel to the boundary edges of the base of the tetrahedral tile. Then the resultant tetrahedral 3D-pattern is moved (remains) to another region (in the same region) with respect to the target indicator in_i (here) associated with the pasting rule. If the target indicator is out, then the resultant tetrahedral 3D-pattern is sent immediately to the next outer region of the membrane structure.

The computation is successful only if the pasting rules of each region are applied. The computation stops if no further application of pasting rule is applicable. The result of a halting computation consists of the 3D-picture patterns composed only of tetrahedral tiles from Γ placed in the membrane with label i_0 in the halting configuration.

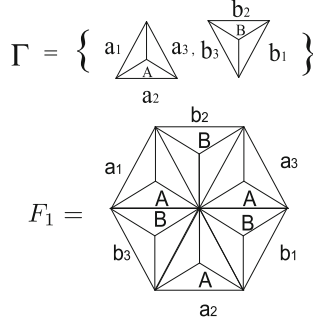
The set of all such tetrahedral 3D-patterns computed or generated by a TetTPPS Π is denoted by $TetPL(\Pi)$. The family of all such languages

$TetPL(\Pi)$ generated by system Π with at most m membranes, is denoted by $TetPL_m(TetTPPS)$.

Example 2. Consider the Tetrahedral Tile Pasting P System, TetTPPS

$$\Pi_1 = (\Gamma, \mu = [1[2[3]3]2]_1, F_1, F_2, F_3, R_1, R_2, R_3, 3),$$

which generates a sequence of tetrahedral 3D-picture patterns whose boundaries are hexagons, μ indicates that the system has three regions one within another i.e. region 1 is the ‘skin’ membrane which contains region 2, which in turn contains region 3, $i_0 = 3$ indicates that region 3 is the output region.



$$\begin{aligned}
 R_1 &= \{(B, (a_1, b_1), here), (A, (b_1, a_1), here), (B, (a_2, b_2), here), \\
 &\quad (A, (b_3, a_3), here), (B, (a_3, b_3), here), (A, (b_2, a_2), in)\} \\
 R_2 &= \{(B, (a_1, b_1), here), (A, (b_1, a_1), here), (B, (a_2, b_2), here), \\
 &\quad (A, (b_3, a_3), here), (B, (a_3, b_3), here), (A, (b_2, a_2), in), (A, (b_2, a_2), out)\} \\
 R_3 &= \emptyset.
 \end{aligned}$$

Beginning with the initial object F_1 in region 1, the pasting rule R_1 is applied, where the rules in R_1 are applied in parallel to the boundary edges of the picture pattern present in region 1. Once the rule $(A, (b_2, a_2), in)$ is applied, the generated 3D-pattern is sent to the inner membrane 2, and in region 2, the rules of R_2 are applied in parallel to the boundary edges of the pattern generated in region 1. If the rule $(A, (b_2, a_2), out)$ is applied, the 3D-pattern generated is sent to region 1, and the process continues. Whereas if the rule $(A, (b_2, a_2), in)$ is applied the 3D-pattern generated is sent to region 3, which is the output region, wherein it is collected in the 3D-picture pattern language formed by $TetTPPS\Pi_1$. $TetTPPS\Pi_1$ is the tetrahedral 3D-Picture language whose boundary is the hexagon.

3 3D-Array Token Petri Nets

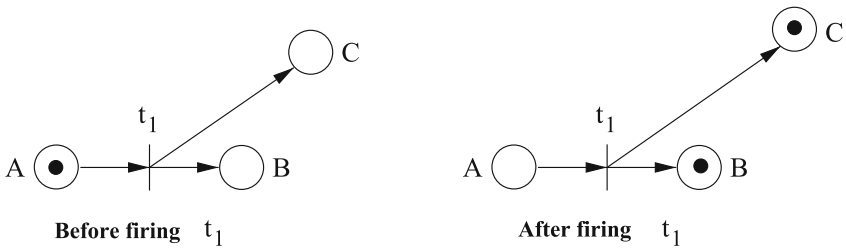
In this section we recall some notions of Petri Nets. For more details we refer to [15]. Here we introduce catenation rules and firing rules for 3D-Array token Petri Nets and Tetrahedral 3D-Array token Petri Nets structure.

Definition 4. [11] A Petri Net structure is a four tuple $C = (P, T, I, O)$ where $P = \{P_1, P_2, \dots, P_n\}$ is a finite set of places, $n \geq 0$, $T = \{t_1, t_2, \dots, t_m\}$ is a finite set of transitions $m \geq 0$, $P \cap T = \emptyset$, $T \rightarrow P^\infty$ is the 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 5. [11] 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 a Petri Net. The number and position of tokens may change during the execution of a Petri Net. The marking can be defined as an n -vector $\mu = (\mu_1, \mu_2, \mu_3, \dots, \mu_n)$ where μ_i is the number of tokens in the P_i , $i = 1, 2, \dots, n$. We can also write $\mu(P_i) = \mu_i$.

Definition 6. [11] A Petri Net C with initial marking μ is called a marked Petri Net. A marked Petri Net $M = (C, \mu)$ can also be written as $M = (P, T, I, O, \mu)$.

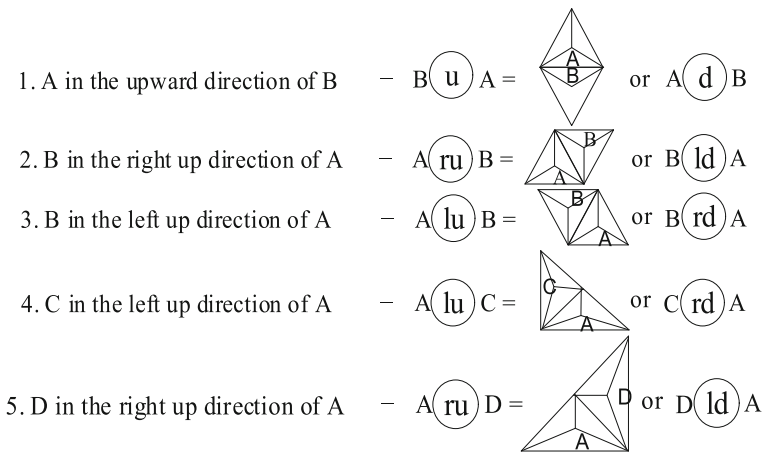
When a transition is fired one token is removed from its input place and one token is placed in each of its output place. For example when t_1 is fired in the following figure one token from place A is removed and one token is placed in both B & C which are the output places of t_1 .

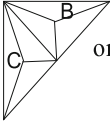


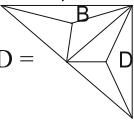
Now we turn our attention to define Tetrahedral 3D Array Token Petri Net.


Catenation Rules

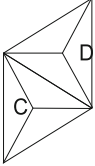
The catenation rules which glue any two tetrahedral tiles at the glueable edges are given below:

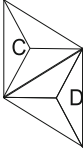


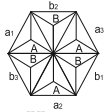
6. C in the left down direction of B - $B \textcircled{\text{ld}} C =$  or $C \textcircled{\text{ru}} B$

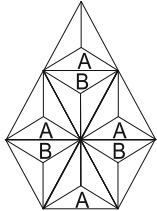
7. D in the right down direction of B - $B \textcircled{\text{rd}} D =$  or $D \textcircled{\text{lu}} B$

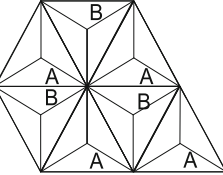
8. D in the right of C - $C \textcircled{\text{r}} D =$  or $D \textcircled{\text{l}} B$

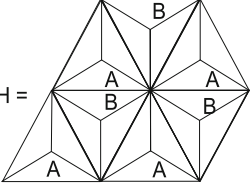
9. D in the left up direction of C - $C \textcircled{\text{lu}} D =$  or $D \textcircled{\text{rd}} C$

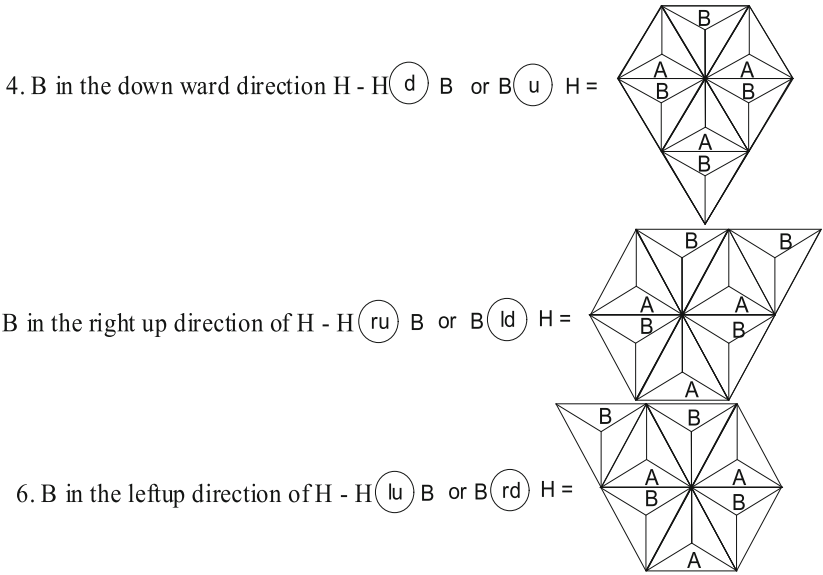
10. C in the right up direction of D - $D \textcircled{\text{ru}} C =$  or $C \textcircled{\text{ld}} D$

Now let us consider the hexagonal polyhedral $H =$ , which is made up of gluing *A*-tetrahedral and *B* tetrahedral tiles. This *H* can be catenated to *A* and *B* - tetrahedral tiles in the following manner.

1. *A* in the upward direction of *H* - $H \textcircled{\text{u}} A$ or $A \textcircled{\text{d}} H =$ 

2. *A* in the right down direction of *H* - $H \textcircled{\text{rd}} A$ or $A \textcircled{\text{lu}} H =$ 

3. *A* in the left down direction of *H* - $H \textcircled{\text{ld}} A$ or $A \textcircled{\text{ru}} H =$ 

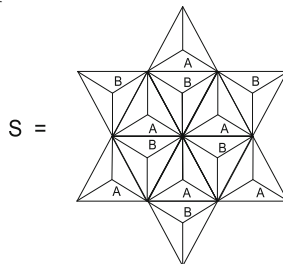


Firing Rules

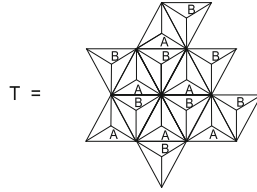
The transitions of the Petri Net are associated with the catenation rules of the form $P \overset{\text{cat}}{\circlearrowleft} Q$ where $P, Q \in \{A, B, C, D, H\}$ and $\overset{\text{cat}}{\circlearrowleft}$ is any one of the above catenation rules. When transition fires, the array in the input place gets catenated according to the catenation rule and the resultant array is placed in the output place. The transitions will be enabled as per the following conditions.

- (i) All the input places will have the same array as token.
- (ii) If there is no label for the transition then the same array will be moved to all the output places.
- (iii) If there is a label i.e a catenation rule for the transition then the array in the input place gets catenated according to the catenation rule and the resultant array is moved to all the output places.

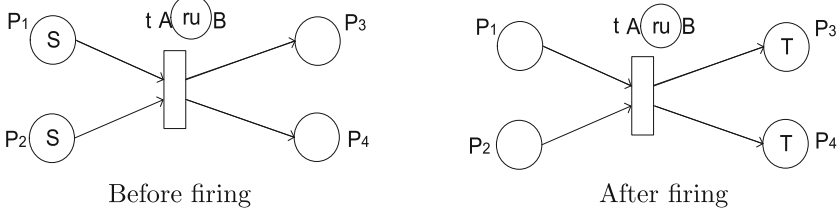
Example 3. If the input place of transition has the tetrahedral polyhedral



as token and the transition is attached to the rule $A \overset{\text{ru}}{\circlearrowleft} B$ then after the firing the output places of the transition will have the tetrahedral picture



The tetrahedral tile - B is catenated in parallel manner to all the A -tetrahedral tiles in the right up direction. The 3D-array token Petri Net diagram is given below for the above transition.



Definition 7. A 3D Tetrahedral Tile Array Token Petri Net (3D-TetATPN) is a six tuple $N = (\Sigma, C, \mu, S, \sigma, F)$ where Σ is an alphabet of tetrahedral tiles or extended tetrahedral tiles (3D-picture made up of tetrahedral tiles), C is a Petri Net structure, μ is an initial marking of 3D-pictures made up of tetrahedral tiles or extended tetrahedral tiles kept in some places of the net, S is a set of catenation rules, σ is a partial mapping which attaches rules to the various transitions of the Petri Net of the form $\sigma(t_i) = P \overset{\text{cat}}{\circlearrowright} Q$, F is a subset of the set of places of the Petri Net where the final 3D-tetrahedral picture is stored after all the firing of the various possible transitions of the Petri Net.

Definition 8. The language generated by 3D-TetATPN is the set of all 3D-tetrahedral pictures stored in the final places of the Petri Net structure and is denoted by $\mathcal{L}(N)$.

Example 4. Consider the 3D-TetATPN $N_1 = (\Sigma, C, \mu, S, \sigma, F)$ where $\Sigma = \{R, A, B\}$ where $R = \begin{matrix} \blacktriangle \\ \blacktriangle \\ \blacktriangle \\ \blacktriangle \end{matrix}$, $C = (P, T, I, O)$ where $P = \{P_1, P_2, P_3, P_4, P_5\}$, $T = \{t_1, t_2, t_3, t_4\}$. The initial marking μ is the rhombus polyhedral R in the place of P_1 . $S = \{H \overset{\text{ru}}{\circlearrowright} B, H \overset{\text{rd}}{\circlearrowright} A, B \overset{\text{rd}}{\circlearrowright} A, D \overset{\text{ru}}{\circlearrowright} C, C \overset{\text{ru}}{\circlearrowright} H, C \overset{\text{rd}}{\circlearrowright} H, H \overset{\text{rd}}{\circlearrowright} H\}$ σ the mapping from the set of transitions to the set of rules is shown in Fig. 3 and $F = \{P_5\}$.

Starting with R , on firing the sequence $t_1 t_2 t_3 t_4$, the rhombus polyhedral is generated. The first two members of the language are shown in the following Fig. 4.

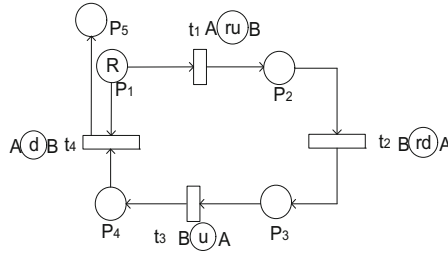


Fig. 3. The 3D - Array token Petri Net generating rhombus polyhedral.

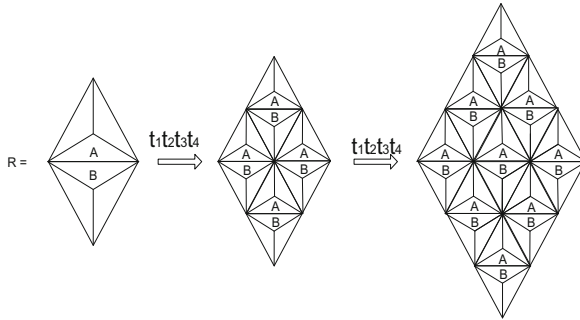


Fig. 4. Rhombus polyhedral.

Example 5. Consider the 3D-TetATPN $N_2 = (\Sigma, C, \mu, S, \sigma, F)$ where, $\Sigma = \{H, A, B, C, D\}$, $C = (P, T, I, O)$, $P = \{P_1, P_2, P_3, \dots P_8\}$, $T = \{t_1, t_2, t_3, \dots t_7\}$. The initial marking μ is the hexagonal polyhedral H in the place of P_1 .

$$S = \{A \textcircled{ru} B, B \textcircled{rd} A, B \textcircled{u} A, A \textcircled{d} B\}.$$

σ the mapping from the set of transitions to the set of rules is shown in Fig. 5 and $F = \{P_8\}$

Starting with H on firing the sequence $t_1t_2t_3t_4$ the tetrahedral tiles B, A, D and C are catenated to H according to the catenation rules respectively and the resultant 3D-array is sent out to place P_5 . On firing the sequence t_5t_6 Hexagonal polyhedrals are catenated to C -tetrahedral tile in parallel in the right up and right down directions and then firing t_7 hexagonal polyhedrals are catenated to hexagonal polyhedrals in the right down direction in parallel and finally the resultant sequence of hexagonal polyhedral language is sent to the final place P_8 .

The first member of the language generated is shown in Fig. 6. In every generation the hexagonal polyhedral tile catenated is increased by one. In the first member two hexagonal polyhedral are catenated twice.

Example 6. Consider the 3D-TetATPN $N_3 = (\Sigma, C, \mu, S, \sigma, F)$ where, $\Sigma = \{H, A, B\}$, $C = (P, T, I, O)$, $P = \{P_1, P_2, P_3, \dots, P_{12}\}$, $T = \{t_1, t_2, t_3, \dots, t_{12}\}$. The initial marking μ is the hexagonal polyhedral H in the place of P_1 .

$$S = \left\{ \begin{array}{l} H \textcircled{u} A, H \textcircled{ru} B, H \textcircled{rd} A, H \textcircled{d} B, H \textcircled{ld} A, H \textcircled{lu} B, A \textcircled{ru} B, A \textcircled{lu} B, \\ A \textcircled{d} B, B \textcircled{u} A, B \textcircled{rd} A, B \textcircled{ld} A \end{array} \right\}$$

σ the mapping from the set of transitions to the set of rules is shown in Fig. 7 and $F = \{P_{13}\}$.

Starting with H on firing the sequence $t_1 t_2 t_3 t_4 t_5 t_6$ the tetrahedral tiles A and B are catenated to H according to the catenation rules respectively and the resultant 3D-array is sent to place P_7 on firing the sequence $t_7 t_8 t_9 t_{10} t_{11} t_{12}$ the tetrahedral tiles A and B are catenated according to the catenation rules and the resultant star polyhedral is generated and it is sent to place P_{13} which is the final place or the sequence of transitions $t_7 t_8 t_9 t_{10} t_{11} t_{12}$ can be repeated any number of times before reaching the final destination P_{13} .

The first two members generated by N_3 are shown in Fig. 8.

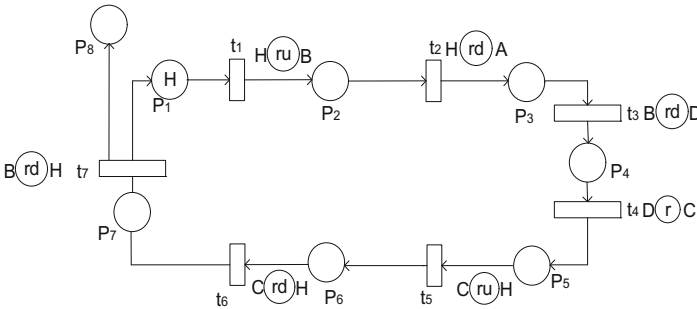


Fig. 5. 3D-Array token Petri Net generating increasing sequence of hexagonal polyhedrals.

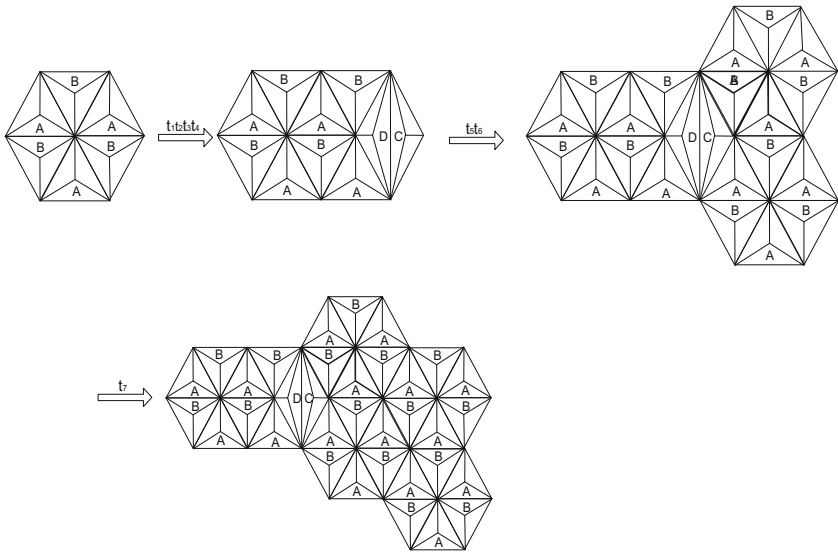


Fig. 6. First member of the language generated by N_2 .

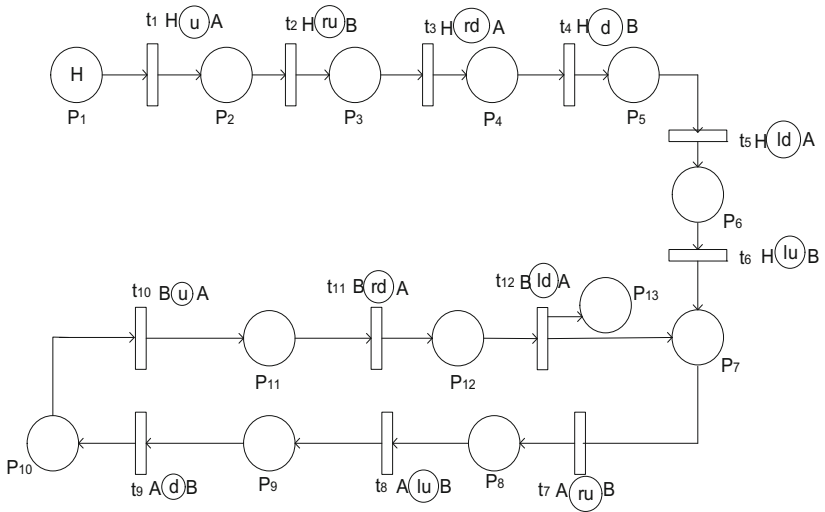


Fig. 7. 3D-array token Petri Net generating sequence of star polyhedrals.

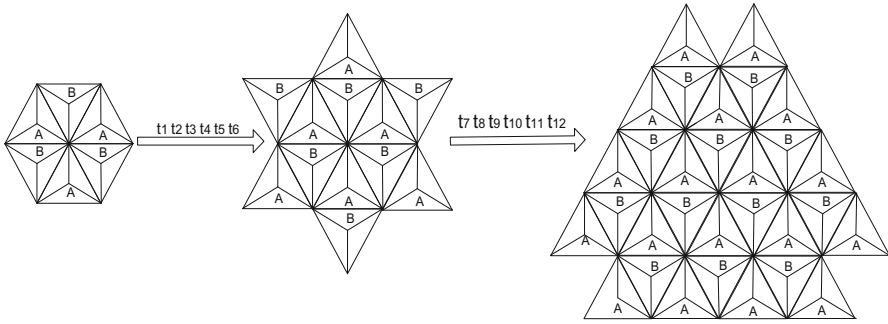


Fig. 8. Star polyhedrals.

4 Comparative Results

In this section, we compare 3D-TetATPN with K -TTTPS, TetTPPS, RTAL [16] and BPTAL [16].

Theorem 1. *The families of languages generated by 3D-TetATPN and K -TTTPS are incomparable but not disjoint.*

Proof. The families of languages generated by K -TTTPS and 3D-TetATPN are by parallel mechanism. The constraint in K -TTTPS is that the pasting rules of the tables are applied in parallel to the pattern obtained in the previous stage. In 3D-TetATPN the catenation rules generate the family of languages where extended tetrahedral tiles are also used.

The language of star and hexagonal polyhedrals in Example 1 cannot be generated by 3D-TetATPN, since the catenation rules are applied in parallel wherever applicable.

The language of increasing sequence of hexagonal polyhedrals given in Example 5 cannot be generated by K -TTTPS, since extended tetrahedral tile, namely hexagonal polyhedral, is used in the catenation rules.

The language of rhombus given in Example 4 can be generated by both systems. A 3-TTTPS generating the family of rhombuses is given below:

Consider a three tabled Tetrahedral Tile Pasting System generating a

$$M = (\Gamma, E, P, t_0) \text{ where } \Gamma = \left\{ \begin{matrix} a_1 & & a_3, b_2 \\ & \triangle & \\ & & b_1 \end{matrix} \right\}$$

$$E = \{a_1, a_2, a_3, b_1, b_2, b_3\}, P = \{T_1, T_2, T_3\}, T_1 = \{(a_3, b_3), (b_1, a_1)\},$$

$$T_2 = \{(b_2, a_2), (b_1, a_1)\} T_3 = \{(a_2, b_2)\}.$$

$$t_0 = \begin{matrix} & \triangle & \\ & & \\ \triangle & & \triangle \\ & & \\ & \triangle & \end{matrix}$$

□

Theorem 2. *The families of languages generated by 3D-TetATPN and TetTPPS are incomparable but not disjoint.*

Proof. In 3D-TetATPN the catenation rules are applied in parallel to generate the language concerned and extended tetrahedral tiles are also used. In TetTPPS the pasting rules are applied in parallel and the target indications permits the array generated to transit within the regions.

The language of stars and hexagonal polyhedrals given in Example 2 generated by TetTPPS cannot be generated by 3D-TetATPN as the catenation rules are applied in parallel wherever applicable.

The language of increasing sequence of hexagonal polyhedrals given in Example 5 cannot be generated by TetTPPS, since extended tetrahedral tiles, namely hexagonal polyhedral, is used in the catenation rules.

The language of rhombuses given in Example 4 is generated by both systems. TetTPPS generating the language of rhombuses is given below. Consider TetTPPS $\pi_2 = (\Gamma, \mu = [_1[2[3]3]2]_1, F_1, F_2, F_3, R_1, R_2, R_3, 3)$. μ indicates that the system has three regions one within the another i.e region 1 is the skin membrane which contains region 2, which in turn contains region 3, $i_0 = 3$ indicates that region 3 is the output region.

$$\Gamma = \left\{ \begin{array}{c} a_1 \quad \begin{array}{c} \triangle \\ \text{A} \\ \triangle \\ a_2 \end{array} \quad a_3 \cdot \begin{array}{c} b_2 \\ \triangle \\ \text{B} \\ \triangle \\ b_1 \end{array} \end{array} \right\}, F_1 = \begin{array}{c} \diamond \\ \text{a} \\ \diamond \end{array}, F_2 = F_3 = \emptyset$$

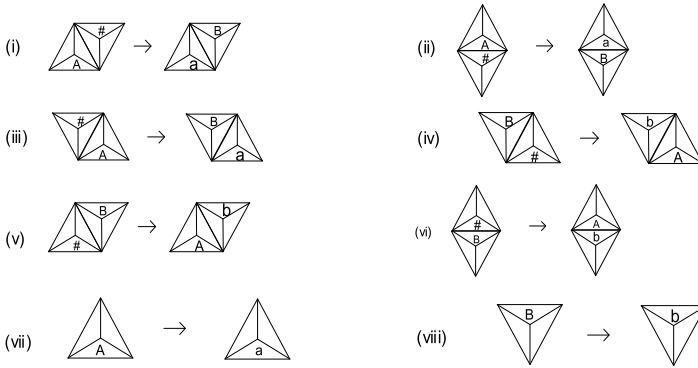
$$R_1 = \{(A, (b_2, a_2), here), (B, (a_3, b_3), here), (A, (b_1, a_1), in)\}$$

$$R_2 = \{(B, (a_2, b_2), in), (B, (a_2, b_2), out)\}, R_3 = \emptyset.$$


Beginning with the initial object F_1 in region 1, the pasting rule R_1 is applied, where the rules in region 1 are applied in parallel to the boundary edges of the pattern present in region 1. Once the rule $(B, (a_3, b_3), here)$ is applied, the tetrahedral tile B is catenated to A and then, when the rule $(A, (b_1, a_1), in)$ is applied, the picture pattern generated is sent to the inner region 2. In region 2, when the rule $(B, (a_2, b_2), in)$ is applied the generated picture pattern is sent to region 3, which is the output region, where there is no rule exits and the language of rhombus is collected in region 3. Whereas if the rule $(B, (a_2, b_2), out)$ is applied, the generated picture pattern is sent to region 1 and the process continues. \square

Theorem 3. $\mathcal{L}(3D - TetATPN) - RTAL \neq \emptyset$.

Proof. We consider a tetrahedral language whose boundary is an equilateral triangle. This language cannot be generated by any Regular Tetrahedral Array Grammar (RTAG) [16]. Since the rules in RTAG are of the following forms:



Similar rules can be given for the other two tetrahedral tiles C and D , where A and B are non terminal symbols and a and b are terminal symbols. Starting with a tetrahedral tile A , RTAG can generate at most three connected tiles. So

it cannot generate an equilateral triangle of the form  but this language can be generated by the following 3D-TetATPN.

Consider a 3D-TetATPN $N_4 = (\Sigma, C, \mu, S, \sigma, F)$, where $\Sigma = \{A, B\}$, $C = (P, T, I, O)$ where $P = \{P_1, P_2, P_3, P_4\}$, $T = \{t_1, t_2, t_3\}$. The initial marking μ is the tetrahedral tile A in place P_1 .

$$S = \{A \text{ (ru) } B, B \text{ (u) } A, B \text{ (rd) } A\}$$

σ the mapping from the set of transitions to the set of rules is shown in Fig. 9, $F = \{P_4\}$ and the language generated by N_4 is shown in Fig. 10. □

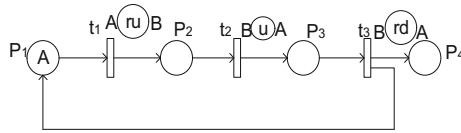


Fig. 9. 3D-TetATPN of the language of equilateral triangle tetrahedral.

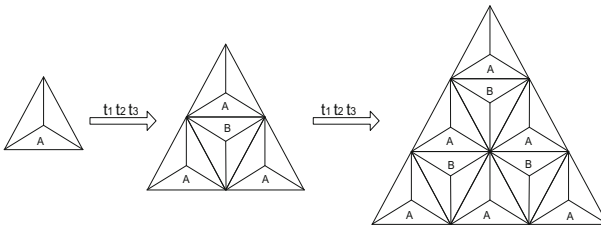
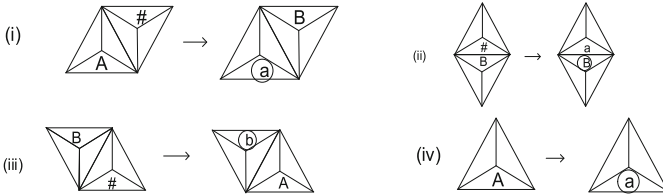


Fig. 10. Language of equilateral triangle tetrahedral.

Theorem 4. $\mathcal{L}(3DTetATPN)$ and $BPTAL$ are incomparable but not disjoint.

Proof. Consider a tetrahedral language whose boundary is an equilateral triangle of size 2. This language is generated by both systems Basic Puzzle Tetrahedral Array Grammar (BPTAG) [16] as well as by 3D-TetATPN.

Consider a BPTAG, $G = (\{ \triangle_{\#}^A, \triangle_{\#}^B \}, \{ \triangle_a^A, \triangle_a^B \}, P, S)$ where P consists of the following rules:



The language generated by G is an equilateral triangle tetrahedral of size 2 which is shown in Fig. 11.

This language can be generated by the following 3D-TetATPN:

Consider a 3DTetATPN $N_5 = (\Sigma, C, \mu, S, \sigma, F)$, where $\Sigma = \{A, B\}$, $C = (P, T, I, O)$ where $P = \{P_1, P_2, P_3, P_4\}$, $T = \{t_1, t_2, t_3\}$. The initial marking μ is the tetrahedral tile A in place P_1 . $S = \{ A \text{ ru } B, B \text{ u } A, B \text{ rd } A \}$, σ the

mapping from the set of transitions to the set of rules is shown in Fig. 12 and $F = \{P_4\}$

Equilateral triangle tetrahedral of size more than 2 cannot be generated by BPTAG, whereas it can be generated by 3D-TetATPN (as in Theorem 3). On the other hand the sequence of overlapping equilateral triangle tetrahedral can be generated by the above BPTAG, whereas it cannot be generated by any 3D-TetATPN as the catenation rules are applied in parallel wherever possible. □

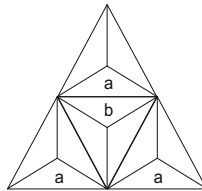


Fig. 11. Equilateral triangle tetrahedral picture of size 2.

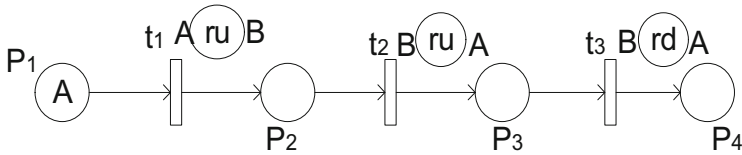


Fig. 12. 3D-TetATPN generating equilateral triangle tetrahedral of size 2.

5 Conclusion

This model is found to be useful in generating interesting patterns and is compared with other recent models in terms of their generative powers. P systems are by definition distributed parallel computing devices and they can solve computationally hard problems in a feasible time. There are NP hard problems in picture languages also. We will analyze whether these problems can be studied using membrane computing. Also we propose to work on a more powerful model of 3D-TetATPN using a concept called ‘inhibitor arc’ to generate further useful patterns and closure properties of 3D-TetATPN. This is our future work.

References

1. Alhazov, A., Fernau, H., Freund, R., Ivanov, S., Siromoney, R., Subramanian, K.G.: Contextual Array Grammars with matrix control, regular control languages and tissue P systems control. *Theor. Comput. Sci.* **682**, 5–21 (2017)
2. Immanuel, B., Usha, P.: Array-token Petri nets and 2d grammars. *Int. J. Pure Appl. Math.* **101**(5), 651–659 (2015)
3. Bhuvanawari, K., Kalyani, T., Lalitha, D.: Triangular tile pasting P system and array generating Petri nets. *Int. J. Pure Appl. Math.* **107**(1), 111–128 (2016)
4. Ceterchi, R., Mutyam, M., Păun, G., Subramanian, K.G.: Array - rewriting P Systems. *Nat. Comput.* **2**, 229–249 (2003)
5. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) *Handbook of Formal Languages*, pp. 215–267. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-642-59126-6_4
6. Siromoney, G., Siromoney, R., Krithivasan, K.: Picture language with array rewrite rules. *Inf. Control* **22**(5), 447–470 (1973)
7. Fernau, H., Paramasivan, M., Schmid, M.L., Thomas, D.G.: Simple picture processing based on finite automata and regular grammars. *J. Comput. Syst. Sci.* **95**, 232–258 (2018)
8. Venkat, I., Robinson, T., Subramanian, K.G., de Wilde, P.: Generation of kolam-designs based on contextual array P systems. In: Chapman, P., Stapleton, G., Moktefi, A., Perez-Kriz, S., Bellucci, F. (eds.) *Diagrams 2018. LNCS (LNAI)*, vol. 10871, pp. 79–86. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91376-6_11
9. Kalyani, T., Raman, T.T., Thomas, D.G.: Tetrahedral tile pasting P system for 3D patterns. *Math. Eng. Sci. Aerosp.* **11**(1), 255–263 (2020)
10. Kamaraj, T., Lalitha, D., Thomas, D.G.: A formal study on generative power of a class of array token Petrinet structure. *Int. J. Syst. Assur. Eng. Manag.* **9**(3), 630–638 (2018)

11. Lalitha, D., Rangarajan, K.: Characterisation of pasting system using array token Petri nets. *Int. J. Pure Appl. Math.* **70**(3), 275–284 (2011)
12. Lalitha, D., Rangarajan, K., Thomas, D.G.: Rectangular arrays and Petri nets. In: Barneva, R.P., Brimkov, V.E., Aggarwal, J.K. (eds.) *IWCIA 2012*. LNCS, vol. 7655, pp. 166–180. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34732-0_13
13. Anselmo, M., Giammarresi, D., Madonia, M.: A common framework to recognize two-dimensional languages. *Fundam. Inform.* **171**(1–4), 1–17 (2020)
14. Păun, G.: *Computing with Membranes: An Introduction*. Springer, Berlin (2002). <https://doi.org/10.1007/978-3-642-56196-2>
15. Peterson, J.L.: *Petri Net Theory and Modeling of Systems*. Prentice Hall Inc., Englewood Cliffs (1981)
16. Raman, T.T., Kalyani, T., Thomas, D.G.: Tetrahedral array grammar system. *Math. Eng. Sc. Aerosp.* **11**(1), 237–254 (2020)
17. Rosenfeld, A.: *Picture Languages (Formal Models for Picture Recognition)*. Academic Press, New York (1979)
18. Subramanian, K.G., Sriram, S., Song, B., Pan, L.: An overview of 2D picture array generating models based on membrane computing. In: Adamatzky, A. (ed.) *Reversibility and Universality*. ECC, vol. 30, pp. 333–356. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73216-9_16