Ekoloji 28(107): 743-751 (2019)

Octagonal Arrays and Petri Nets: A Computational Ecology Approach

S. Kuberal [1], T. Kamaraj [2*], T. Kalyani [3]

[1] Department of Applied Mathematics, Pillai College of Engineering, New Panvel-410 206, Maharastra, INDIA

[2] Department of Mathematics, Sathyabama University, Chennai-600 119, INDIA

[3] Department of Mathematics, St. Joseph's Institute of Technology, Chennai-600 119, INDIA

* Corresponding author: kamaraj_mx@yahoo.com

Abstract

Ecosystems are extraordinarily complex assemblages of plants and animals interacting with each other, with the physical environment, and, increasingly, with humans. Ecology is the scientific field that seeks to understand ecosystems. Ecologists are turning to computer models to help them make their models of ecosystem processes concrete and to provide predictions about the future of the ecosystem. Computer models allow rapid testing of ecology ideas by simulation and provide the means to run "what-if" scenarios that would be difficult or impossible otherwise. Petri Net Generating Triangular Arrays were introduced by Kuberal et al. (2015) to generate Triangular Arrays using Petri Net structure. A new model to generate Octagonal Arrays using Petri Net Structure has been defined and it is proved that this model generate Octagonal Array Languages. The catenation of an arrowhead to a b-octagon results in a similar b-octagon. This concept has been used in Octagonal Array Token Petri Net Structure (OATPNS).

Keywords: computational ecology, Octagonal Array Language, Petri nets, array tokens, arrowhead catenations

Kuberal S, Kamaraj T, Kalyani T (2019) Octagonal Arrays and Petri Nets: A Computational Ecology Approach. Ekoloji 28(107): 743-751.

INTRODUCTION

Computational science happens when algorithms, software, data management practices, and advanced research computing are put in interaction with the explicit goal of solving "complex" problems. Typically, problems are considered *complex* when they cannot be solved appropriately with modeling or data-collection only. Computational science is one of the ways to practice computational thinking (Papert 1996), *i.e.* the feedback loop of abstracting a problem to its core mechanisms, expressing a solution in a way that can be automated, and using interactions between simulations and data to refine the original problem or suggest new knowledge. Computational approaches are common place in most area of biology, to the point where one would almost be confident that they represent a viable career path (Bourne 2011). Data usually collected in ecological studies have a high variability, are time-consuming, costly, and demanding to collect. In parallel, many problems lack appropriate formal mathematical formulations. For these reasons, computational approaches hold great possibilities, notably to further ecological synthesis and help decision-making (Petrovskii and Petrovskaya 2012).

Computational ecology is the application of computational thinking to ecological problems. This defines three core characteristics of computational ecology. First, it recognizes ecological systems as complex and adaptive; this places a great emphasis on mathematical tools that can handle, or even require, a certain degree of stochasticity. Second, it understands that data are the final arbiter of any simulation or model (Petrovskii and Petrovskaya 2012); this favors the use of data-driven approaches and analyses (Beaumont 2010). Finally, it accepts that some ecological systems are too complex to be formulated in mathematical or programmatic terms (Pascual 2005); the use of conceptual, or "toy" models, as long as they can be confronted to empirical data, is preferable to "abusing" mathematics by describing the wrong mechanism well (May 2004).

Octagonal arrays and Octagonal patterns are found in the literature on picture processing and scene analysis. Image generation can be done in many ways in formal languages. Several grammars were introduced in

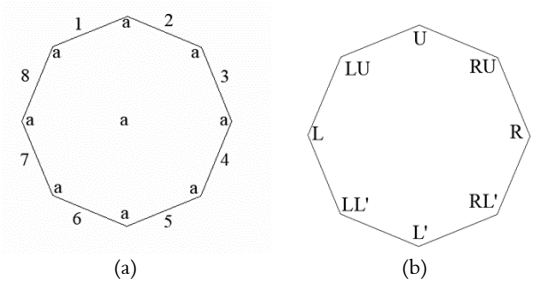
 Received: 10 Sep 2017 / Accepted: 26 Jun 2018

the literature to generate various classes of octagonal picture languages. The classes of local and recognizable octagonal picture languages were introduced by S. Kuberal et al. (2017). Petri net have been used for analyzing systems that are concurrent, asynchronous, distributed, parallel, non-deterministic and/or stochastic. Tokens are used in petri nets to simulate dynamic and concurrent activities of the system. A language can be associated with the execution of a petri net. By defining a labeling function for transitions over an alphabet, the set of all firing sequences, starting from a specific initial marking leading to a finite set of terminal markings, generates a language over the alphabet.

Petri net model to generate rectangular arrays has been introduced in Lalitha et al. (2012) and Petri net model to generate hexagonal arrays has been introduced in Lalitha et al. (2011) and Petri net model to generate triangular arrays has been introduced in Kuberal et al. (2015). Motivated by this concept we have introduced a petri net model to generate octagonal arrays. The resulting model as an octagonal array token petri net structure (OATPNS).

OCTAGONAL ARRAYS AND ARROWHEADS

Let Σ be a finite non empty set of symbols. The set of all octagonal arrays made up of element of Σ is denoted by Σ_0^{**}. The size of any octagonal array is defined by its parameter. For an octagon the parameters are $|O|_{U,}|O|_{RU,}|O|_{R,}|O|_{RL',}|O|_{L',}|O|_{LL',}|O|_{L,}|O|_{LU}$and ; where U stands for upper; RU stands for right upper; R, right; RL', right lower; L', lower; LL', left lower; L, left; LU, left upper.

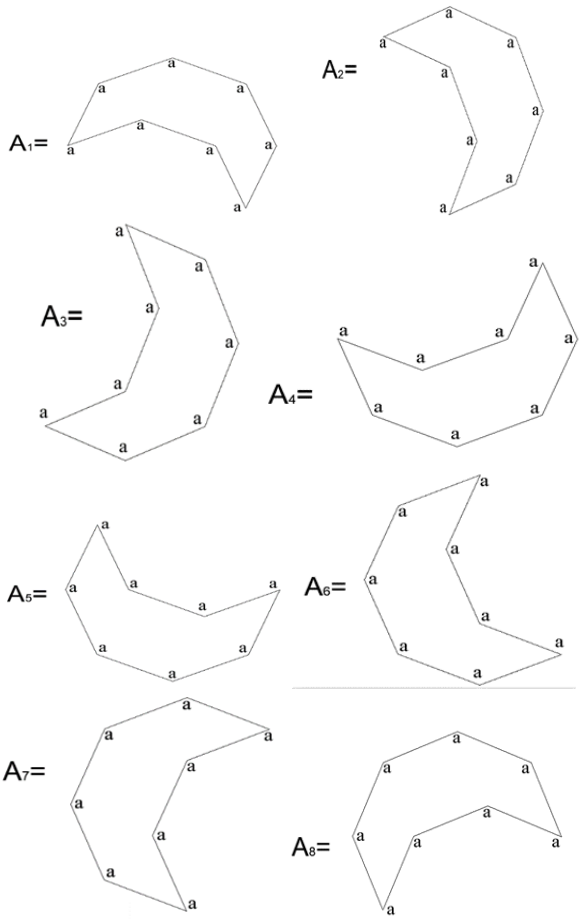


(a) (b)

Fig. 1.

For any octagon there are four types of catenation with eight arrowheads are possible. An arrowhead is written in the form $\{\ldots < v > \cdots\}$ where <v> denotes the vertex and the arrowhead is written in the clockwise direction.

Octagon has eight sides namely 1, 2, …, 8 as shown in **Fig. 1(a)**. Then A_1 will be catenated with sides 1, 2 and 3; A_2 with sides 2,3 and 4; A_3 with sides 3, 4 and 5; A_4 with sides 4,5 and 6; A_5 with sides 5,6 and 7; A_6 with sides 6, 7 and 8; A_7 with sides 7, 8 and 1; A_8 with sides 8, 1 and 2.



OCTAGONAL ARRAY TOKEN PETRI NETS (OATPN)

In this section we give preliminary definitions of petri net and 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 (that is events that may occur, signified by bars) places (that is 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 (Becerril-Ángel et al. 2017).

In an abstract sense relating to a petri net diagram, a transition of a petri net may fire whenever there 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 (Peterson 1981). Hack (1975) and Baker (1972) have published a report on petri net languages.

We now recall the basic definitions of petri net (Lalitha et al. 2012) and the basic notations pertaining to octagonal arrays.

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≥ 0, T = $\{t_1, t_2, \ldots, t_m\}$ is a finite set of transitions m ≥ 0, P ∩ T = φ, I: T → P^∞ is input function from transitions to bags of places and O: T → P^∞ is the output function from transitions to bags of places.

Definition 2. A petri net marking is an assignment of 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 petrinet.

Definition 3. An inhibitor arc from a place pi 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 pi has no tokens. A transition is enabled only if all its regular inputs have tokens and all its inhibitor inputs have zero tokens.

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

Firing Rules

We define three different types of enabled transition in OATPNS. 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 octagonal array as a token.

Then on firing the transition arrays from all the input places are removed and put in all its output places.

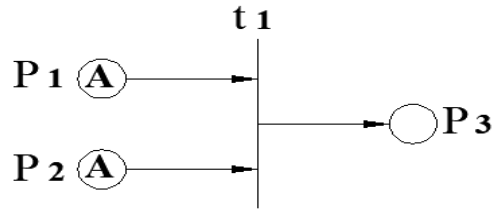


Fig. 2. Position of arrays before firing

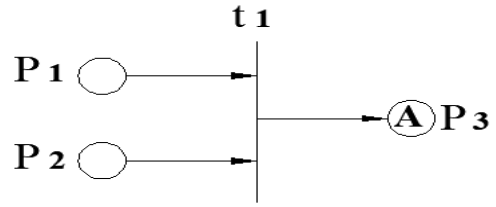


Fig. 3. Position of array after firing

(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.

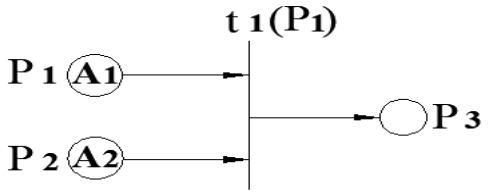


Fig. 4. Transition with label before firing

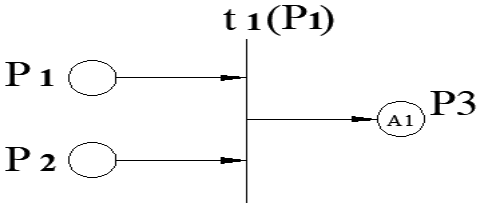


Fig. 5. Transition with label after firing

Definition 4. If C = (P, T, I, O) is a petri net structure with octagonal arrays over of Σ_0^{**} as initial markings, $M_0: P \rightarrow \Sigma_0^{**}$, labels of at least one transition being an arrowhead catenation rule and a finite set of final places $F \subseteq P$, then the petri net structure C is defined as Octagonal Array Token Petri Net Structure (OATPNS).

Definition 5. If C is an OATPNS then the language generated by the Petri net C is defined as L(C) = {O ∈ Σ_0^{**}/ O is in P for some P in F} with arrays of Σ_0^{**} in some places as initial marking all possible sequences of transitions are fixed. The set of all octagonal arrays in the final places F is called the language generated by C.

(iii) Let a transition t have O ⊛ A as a label where is in any one of the eight directions (⍓, ⍌, ⍐, ⍇, ⍈, ⍄, ⍃, ⍗). O is the octagonal array in all the input places and A is a predefined arrowhead. Then firing the transition will catenate A with O in the specified direction and put in all the output places subject to the condition of arrowhead catenation. If the condition for arrowhead catenation is not satisfied then the transition cannot fire.

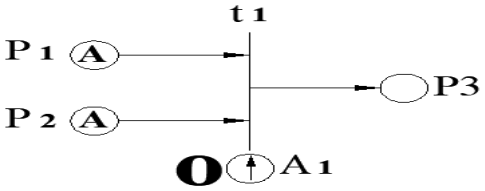


Fig. 6. Transition with upper arrowhead catenation rule before firing

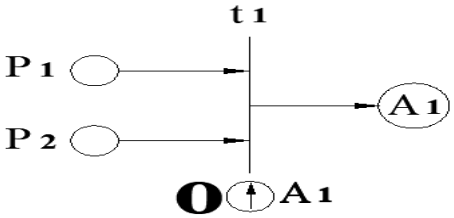


Fig. 7. Transition with upper arrowhead catenation rule after firing

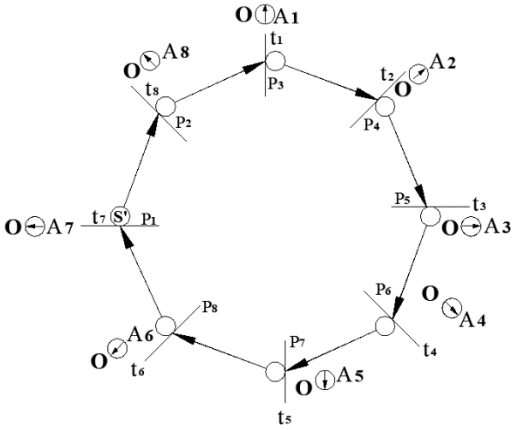
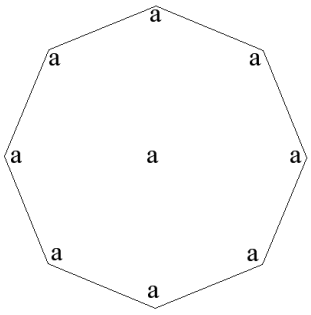


Fig. 8.

If A =



the number of columns of A is 2, firing t adds upper arrowhead catenation A_1, we get B

B=

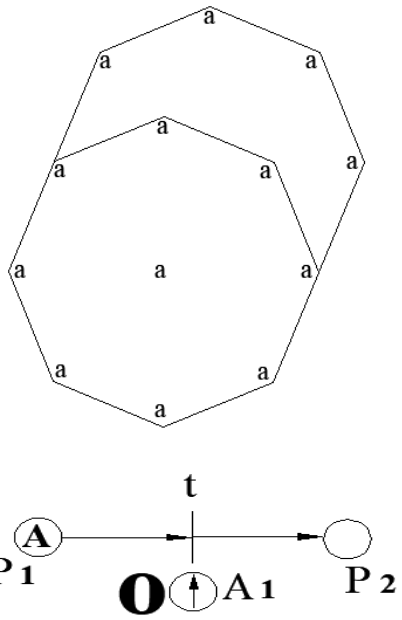


Fig. 9. Position of token before firing

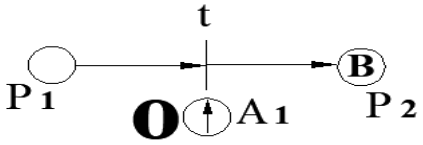
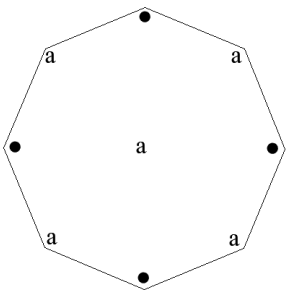
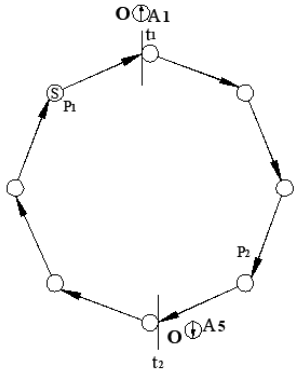


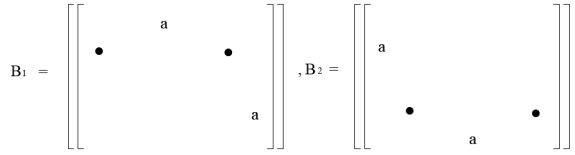
Fig. 10. Position of token after firing

Example: 1 ∑={a,●}, F={P_1}

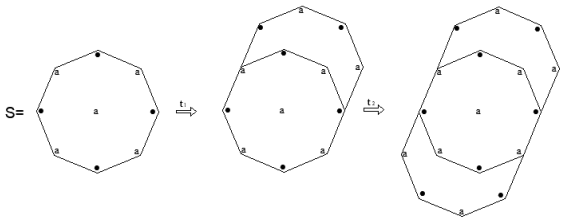
S=







Firing t_1 puts an array in P_2 making t_2 enabled. Firing t_2 puts an array in P_1. The firing sequence $(t_1t_2)^k$, $k \geq 0$. When the transitions t_1, t_2 fire the array that reaches the output place is show below.



The language generated by this OATPNS is Octagonal Rangoli.

Comparison Results

In this section we recall the definition of Octagonal Array Grammar (OAG) (Kamaraj and Thomas 2012) and Octagonal Array Language (OAL) and compare the generative power of OATPNS with (X: Y) OAL, where X, Y ∈{R, CF, CS}.

Definition 6:

An octagonal array grammar G = (N, I, T, P, S, L) where N, I and T are finite sets of non-terminals, intermediates and terminals respectively. P is a finite set of productions, P = $P_1 \cup P_2$ and S ∈ N is the start symbol. For each A in I, L_A is an intermediate language which is a regular CF and CS string language 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 L = {L_A / A∈ I}.

P_1 consists of an initial rule of one of the following form

(1) $S \rightarrow O \circledast S'$ (2) $S \rightarrow O \circledast S'$

(3) $S \rightarrow O \circledast S'$ (4) $S \rightarrow O \circledast S'$

(5) $S \rightarrow O \circledast S'$ (6) $S \rightarrow O \circledast S'$

(7) $S \rightarrow O \circledast S'$ (8) $S \rightarrow O \circledast S'$

where S′∈ N, S′≠S and O is an octagonal array over T.

G is regular if the rules of P_2 are of the forms

(1) $S_1 \rightarrow A \circledast S_2$ (2) $S_1 \rightarrow A \circledast S_2$

(3) $S_1 \rightarrow A \circledast S_2$ (4) $S_1 \rightarrow A \circledast S_2$

(5) $S_1 \rightarrow A \circledast S_2$ (6) $S_1 \rightarrow A \circledast S_2$

(7) $S_1 \rightarrow A \circledast S_2$ (8) $S_1 \rightarrow A \circledast S_2$

where $S_1, S_2 \in N$, $S_1, S_2 \neq S$ and $A \in I$.

Furthermore, if an initial rule is P_1 is of the form (r), r = 1, 2, …. then P_2 does not contain any rule of the form (r+1) if r is odd, (r−1) if r is even. Also P_1 and P_2 do not contain rules of both the form (r) and (r+1), r = 1, 3, 5, 7.

G is CF if the rules are of the form $S_1 \rightarrow \alpha_1 \circledast \ldots \circledast_{k-1} \alpha_k$ ($k \geq 1$) where $S_1 \in N$; $S_1 \neq S$ and $\alpha_i \in (N - \{S\}) \cup I$ ($1 \leq i \leq k$) and $\circledast_j$ denotes any one of the six arrowhead catenations ($1 \leq j \leq k-1$).

G is CS if the rules of P_2 are of the form $\beta \circledast S_1 \circledast \delta \rightarrow \beta\gamma\delta$ or $\beta \circledast S_1 \rightarrow \beta\gamma$ or $S_1 \circledast \delta \rightarrow \gamma\delta$ or $S_1 \rightarrow \gamma$, where $S_1 \in N$, $S_1 \neq S$ and β, γ, δ are of the form $\alpha_1 \circledast \ldots \circledast \alpha_k$ with $\alpha_i \in (N - \{S\}) \cup I$, $1 \leq i \leq k$.

In particular, G is called (X : R) OAG or (X : CF) OAG or (X : CS) OAG, for X ∈ {R, CF, CS} according as all the intermediate languages 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 the 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

$O \circledast A_1 \circledast \ldots \circledast A_n$, where $A_i \in I$ ($1 \leq i \leq n$).

In the second phase of derivations, A_1 replaced by an arrowhead from L_{A_1} and catenated to an octagonal array O according to the arrowhead catenation symbol between O and A_1. This is continued until A_n is replaced, resulting in an octagonal array of terminals. The length of the arrowhead is determined by the condition for arrowhead catenation.

Definition 7:

For X, Y ∈ {R, CF, CS}, the (X, Y) octagonal array language ((X:Y) OAL) generated by the (X:Y)OAG G is L(G) = { O/ $S \overset{*}{\Rightarrow}_G O$, O is an Octagonal array}

Theorem 1. Every (R:X) OAL, for X ∈ {R, CF, CS} can be generated by OATPNS.

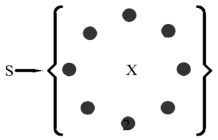
Proof: Let G be the corresponding (R:X) OAG. Let $S \rightarrow O \circledast S'$ be the initial rule in P_1 and $L = \{L_A / A \in I\}$ be the set of intermediates languages. Define for every L_A an arrowhead of similar type from A_1 toA_8. The parameters of the arrowhead will depend on the parameters of an octagon in the input place. So, firing the transition with catenation rule as labels will catenate the arrowhead to an octagon. Let $O \circledast A_1 \ldots A_n$ be the string of arrowheads A_i and O, which derives the first array of the language.

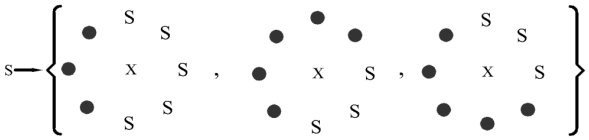
In OAG the derivation is as follows: Starting with S the non-terminal rules are applied without any restriction as in string grammar, till all the non-terminals are replaced then A_1 is replaced by an arrowhead catenation from LA_1 an catenated to the octagonal array O according to the arrowhead catenation symbol between O and A_1. This is continued until A_n is replaced, resulting in an octagonal array of terminals. Construction of OATPNS for the case when $S \rightarrow O_1A_1$, 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 O_1 be the start place P_1 as token. Have a transition t_1 with upper arrowhead catenation rule O_1A_1as a label. Let P_1 be the input place of t_1. The length of the arrowhead is determined by the condition for arrowhead 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 upper arrowhead catenation rule A_2, O_1A_2as 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 t1 during the course of the execution of the net. Let P_3 be the output place for the transition t_2. Similarly the other cases can be dealt. Finally, P_8 be the input place of t_7, have a transition $t_8$with the left upper arrowhead catenation rule A_8 that is O_1A_8as a label. Let P_1 be the output place for the transition t_8. First time the sequence $t_1t_2\ldots t_8$ is executed, the octagonal array $O_2$is put in P_1. Let $F=\{P_1\}$ be the final set of places. The firing sequence $(t_1t_2\ldots t_8)^n$, $n>0$in P_1. Thus $\{O_n/n>0\}$ of the o array is octagonal array in the language generated.

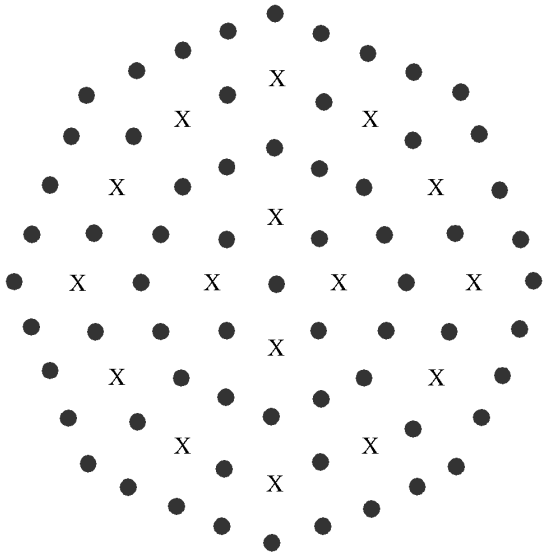
Example: 1

Let $G=\{\Sigma, N, S, R\}$ be an octagonal tile rewriting grammar, where $\{\cdot, x\}$, $N=\{S\}$ and R consists of the rules:





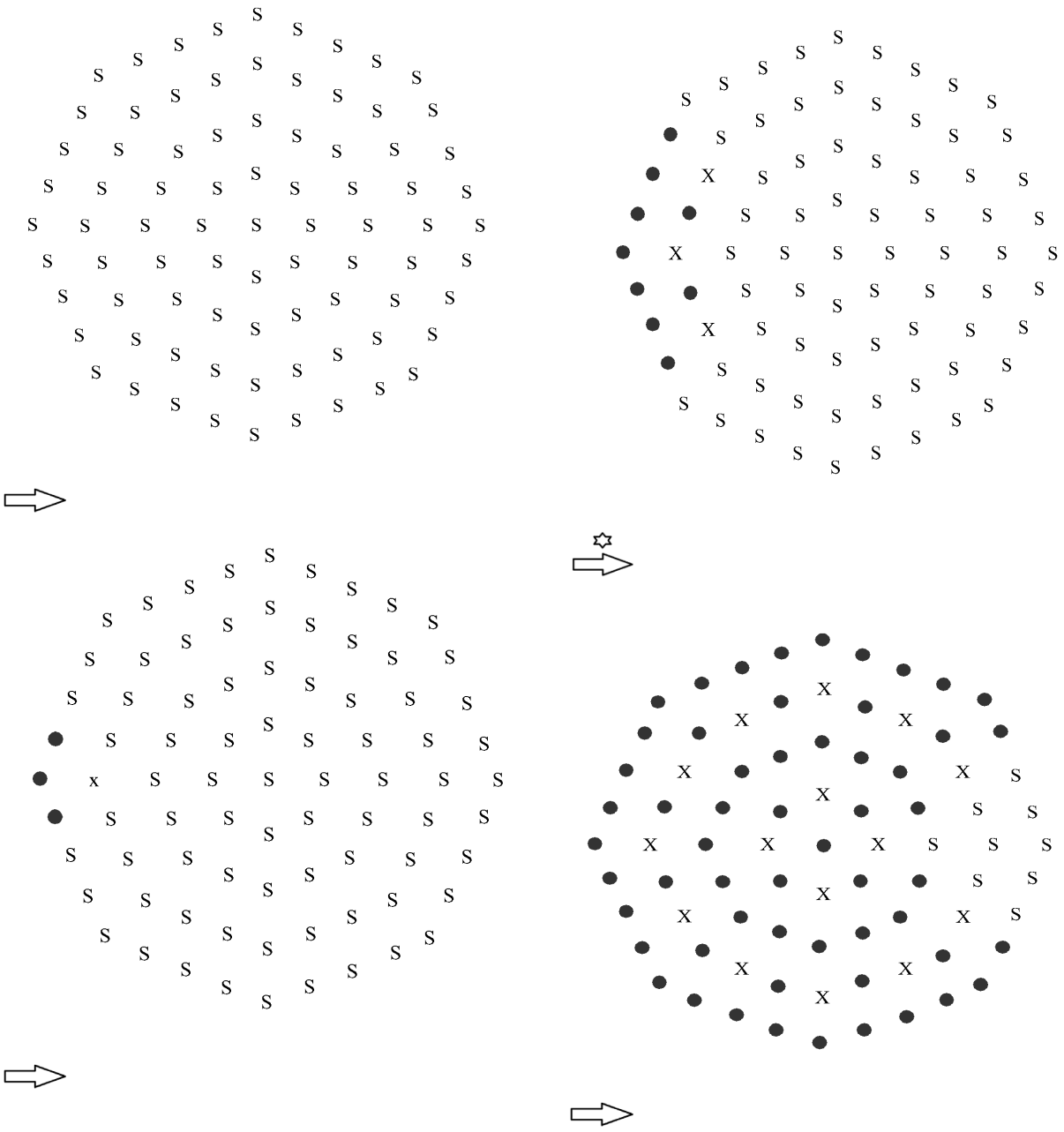
A picture in L(G) is,

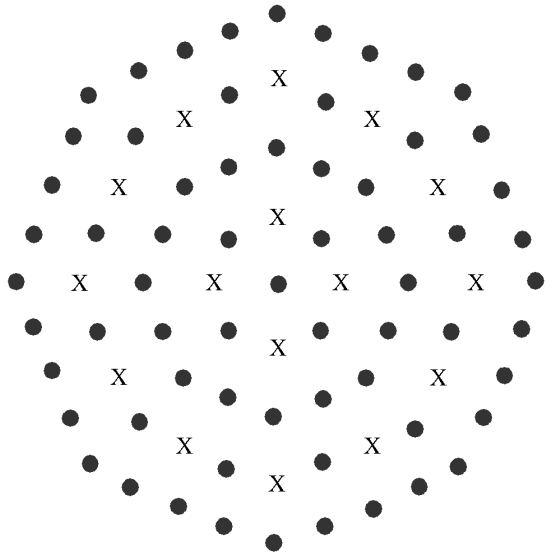


Size 5

A picture L(G) is obtained by repeated application of the variable size rule and fixed size rule.

For example, we give the sample derivation to yield the picture.





Theorem 2. For X ∈ {CF, CS}, Y = {R, CF, CS} the family (X:Y) OAL cannot be generated by OATPNS.

Proof. In (CF:Y) OAG the rules in P_2would have a sequence of catenation on O 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 beno control on the number of times C_1 and C_2 get executed. Hence OATPNS cannot generate a (CF:Y) OAL. Since (CF:Y) OAL does not contains (CS:Y) OAL, TATPNS cannot generate (CS:Y) OAL.

CONCLUSION

Ecological computation, due to the unique combination of ecology and big data, the course is suitable for: ecologists involved in scientific research, graduates in mathematics or computing. OATPNS generates octagonal arrays. There are three models for generating octagonal arrays have been defined and compared with some of the already existing models. These models are able to generate certain families of OAL. If some sort of control is defined on the sequence of firing the other families of OAL can also be generated.

REFERENCES

Baker HG (1972) Petri net languages. Computation Structures Group Memo 124, Project MAC. MIT, Cambridge.

Becerril-Ángel, M. L., Castillo-Pérez, J. J., Montiel-Jarquín, Á. J., Villatoro-Martínez, A., García- Cano, E (2017) Aspects of the submission of articles to scientific journals. European Journal of General Medicine, 14(2), 47-50. doi: 10.29333/ejgm/81882

Bhuvaneswari K, Kalyani T, Gnanaraj Thomas K, Nagar AK, Robinson T (2014) Iso-array rewriting P systems with context-free rules, International Journal of Application for Mathematics, 3(1): 1-16.

Crespi Reghizyi S, Pradella M (2005) Tile Rewriting Grammars and Picture Languages, Theoretical Comp. Science, 340: 257–272.

Crespi Reghizzi S, Pradella M (n.d.) Tile Rewriting Grammars. DEI Politecnico di Milano and CNR IEIITMI, Piazza Leonardo da Vinci, 32, I-20133 Milano, Italy.

Giammarresi D, Restivo A (1992) Recogniable Picture Languages, International Journal Pattern Recognition and Artificial Intelligence 6(2-3): 241– 256.

Giammarresi D, Restivo A (1997) Two-dimensional Languages, In Arto Salomaa and Grzegorz Rozenberg, editors Handbook of Formal Languages, 3(Beyond Words): 215–267.

Hack M (1975) Petri net languages. Computation Structures Group Memo 124, Project MAC.MIT.

Kamaraj T, Thomas DG (2012) Regional hexagonal tile rewriting grammars, In R.P. Barneva etal (eds.), IWCIA 2012, LNCS, Vol. 7655, Springer, Heidelberg: 181-195.

Kuberal S, Kalyani T, Kamaraj T, Bhuvaneswari K (2017) Octagonal Tile Rewriting Grammars and Picture Languages. International Journal of Computer & Mathematical Sciences, 6(9).

Kuberal S, Kalyani T, Kamaraj T, Bhuvaneswari K (2018a) Regional Octagonal Tile Rewriting Grammars. Journal of Advanced Research in Dynamical and Control Systems, (06-special issue): 1838-1849.

Kuberal S, Kalyani T, Thomas DG (2016) Triangular Tile Rewriting Grammars and Triangular Picture Languages, Global Journal of Pure and Applied Mathematics, 12(3): 1965–1978.

Kuberal S, Kalyani T, Thomas DG, Kamaraj T (2015) Petri net generating Triangular arrays. Procedia Computer science, Elsevier, 57: 642-649.

Kuberal S, Kamaraj T, Kalyani T (2018) Regional Triangular tile rewriting grammars. International Journal of Pure and Applied Mathematics, 119(15): 1325-1337. Retrieved from http://www.acadpubl.eu/hub/

Lalitha D, Rangarajan K (2010) Column and row catenation petri net systems, in Proceedings of Fifth IEEE International Conference on Bio-Inspired Computing: Theories and Applications: 1382-1387.

Lalitha D, Rangarajan K, Thomas DG (2011) Petri net generating hexagonal arrays, J.K. Aggarwal et al. (eds.) IWCIA 2011, LNCS 6636, p. 235-247. Springer-Verlag, Berlin, Heidelberg.

Lalitha D, Rangarajan K, Thomas DG (2012) Rectangular arrays and petri nets, R.P. Baineva et al. (eds.) IWCIA 2012, LNCS 7655, pp. 166-180. Springer-Verlag, Berlin, Heidelberg.

Middleton L, Sivaswamy J (2005) Hexagonal image processing: A practical approach, Springer; Heidelberg.

Peterson JL (1981) Petri Net Theory and Modeling of Systems, Prentice Hall, Inc., Englewood Cliffs.

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

Siromoney G, Siromoney R, Kamala K (1974) Array grammars and kolam, Computer Graphics and Image Processing, 3(1): 63-82.

Subramanian KG (1979) Hexagonal array grammar, Computer Graphics and Image Processing, 10: 388-394.

Subramanian KG, Geethalakshmi M, Atulya K. Nagar LSK (2009) Hexagonal picture languages, in Proceedings of the 5th Asian Mathematical Conference, Malaysia: 510-514.

Sweety F, Kalyani T, Thomas DG (2007) Hexagonal Tile Rewriting Grammars, 7th National Conference on Emerging Trends in Automata (ETA'07): 101–115.

Thomas DG, Sweety F, Kalyani T (2008) 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: 945–952.

Wang PSP (1989) Array grammars, Patterns and Recognizers. World Scientific, Series in Computer Science, 18.