Punjab University Journal of Mathematics
(ISSN 1016-2526)
Vol.52(11)2020 pp.11-18

P Systems for Patterns of Sierpiński Square Snowflake Curve

P.S. Azeezun Nisha
Department of Mathematics,
J.B.A.S. College for Women, Chennai 600018 India

Research Scholar, University of Madras, Chennai, India
Email: hajma29112008@gmail.com

S. Hemalatha
Department of Mathematics,
S.D.N.B. Vaishnav College for Women, Chennai 600044 India
Email: hemrav2008@gmail.com

S. Sriram
Department of Mathematics, School of Arts, Science and Humanities,
SASTRA Deemed University, Tanjore, Tamil Nadu 613 401 India
Email: sriram.discrete@gmail.com

K.G. Subramanian**
Department of Mathematics, Computer Science and Engineering,
Liverpool Hope University, Liverpool L16 9JD U.K.

**Honorary Visiting Professor
Email: kgsmani1948@gmail.com

Received:23 December,2019 / Accepted:18 September,2020 / Published online:12 Novem-ber,2020

Abstract.: Syntactic models for generating the approximating polygon patterns of space-filling curves such as the well-known Peano and Hilbert curves have been studied in the recent past. Here we consider the polygon patterns of another space-filling curve, namely, Sierpiński square snowflake curve. For generating array representations of these patterns, we introduce a variant of a fairly recent and novel computing model, known as P system in the membrane computing field. We then develop a finite set of joining rules which on application, join the primitive patterns in the arrays, thereby yielding the polygon patterns of the Sierpiński square snowflake curve.

AMS (MOS) Subject Classification Codes: 68Q07
Key Words: Space-filling curve, Sierpinski curve, Membrane computing, P system.

1. INTRODUCTION

Space-filling curves [13] have always fascinated researchers with their remarkable property of being continuous and at the same time passing through every point of a unit square in the two-dimensional plane. Space-filling curves also have applications in scientific computing [2]. Peano [11] introduced a space-filling curve in 1890, subsequently called Peano's curve. Hilbert in 1891 constructed another space-filling curve, now referred to as Hilbert's curve [7] and also provided a geometrical generating procedure. Sierpiński [14] in 1912, introduced a space-filling curve and showed that it is the limit of a sequence of polygons. This curve is called in [17] as Sierpiński square snowflake curve. The first two of the polygonal patterns in the sequence of patterns defining this curve is shown in Fig. 1.

In the fast growing field of membrane computing [9, 10], the new model of computing, generally called P system (with P standing for Gh. Păun who is the creator of this system) has served as a very suitable platform for handling problems in diverse directions of study and application. The fundamental model of a cell-like P system [8] rewriting structured strings has a hierarchical arrangement of membranes, one within another. A single membrane called skin membrane contains all other membranes. The regions defined by the membranes can have objects which are structured strings of symbols and evolution rules. The working of such a P system involves rewriting the strings, if any, in all the membrane regions of the P system by applying the evolution rules to the structured string objects in the regions in a manner which involves nondeterminism and maximal parallelism and which allows the objects to evolve. The evolved objects, in the subsequent step, can remain in the same membrane or can be sent to an immediate neighbour region which is determined by a feature called target specification. A computation comes to a halt when no further evolution of objects in all the membranes can take place. Such a computation results in the set of strings generated.

Among the various areas where P systems have played a great role, generation of two-dimensional pictures and patterns [6] based on P systems, called array P systems, has been of great interest [16]. The problem of generating the approximating polygons of the space-filling curves of Peano and Hilbert based on array grammars has been considered in [15]. There has also been an intensive research in the problem of this kind with constructions of P systems for generating patterns of many space-filling curves [3, 4, 5]. Here we deal with the problem of generating the set of polygonal patterns of Sierpiński square snowflake curve and construct a cell-like P system generating arrays which describe these patterns. The P system itself has only two membranes and the evolution rules in the membranes of the P system are the context-free rules rewriting in parallel the array-objects in the membrane regions of the P system, which are arrays encoding the patterns. When the symbols in the generated arrays of the constructed P system are suitably joined by using a finite number of instructions, the polygonal patterns of the Sierpiński square snowflake curve are obtained. A first draft version of this work was presented in the conference on Mathematical Computer Engineering, held in India in 2018 [1].

2. PRELIMINARIES

For concepts related to words and arrays, we refer to [6, 12] and for P systems, to [8, 16]. We state here certain basic concepts that are used in subsequent sections.

An alphabet Γ is a finite set of symbols. A word or a string over an alphabet Γ is a finite sequence of symbols, taken from Γ. For example, $\Gamma = \{0, 1\}$ is an alphabet and 01011 is a word over Γ. The set of all words, which contains λ, the empty word with no symbols, is denoted by Γ^*. A word can be considered as a $1 \times n$ array of symbols and an extension of the notion of a word to two dimensions is the concept of a rectangular $m \times n$, $(m, n \geq 1)$, array of symbols from an alphabet with the symbols placed in m rows and n columns. For example, if $\Gamma = \{x, b\}$ is an alphabet, then

$$\begin{array}{ccccc} x & x & x & x & x \\ b & b & x & b & b \\ b & b & x & b & b \end{array}$$

is an 3×5 array which can be interpreted as the letter T in a digitized form over the symbol x treating b as standing for a blank or empty symbol. The set of all arrays, which contains λ, the empty array with no symbols, is denoted by Γ^{**} and $\Gamma^{++} = \Gamma^{**} - \{\lambda\}$.

The computing model of a P system has different variants. Here we recall one of the basic forms [8] of a string language generating P system of the rewriting variety. Such a P system $\Pi = (V, \Sigma, \mu, A_1, \cdots, A_m, P_1, \cdots, P_m, i_0)$ where V is an alphabet consisting of nonterminals and terminals and $\Sigma \subset V$ is the set of terminals; μ is a membrane structure which has m membranes with each having a distinct label; The objects in the membranes or regions of Π are words over V; Each A_i, $1 \leq i \leq m$, is a set of initial words in region i; P_i, $1 \leq i \leq m$, is a set of context-free rules which are of the form $X \rightarrow \alpha$ where $X \in V - \Sigma$ is a nonterminal and $\alpha \in V^*$ is a string of nonterminals and terminals. Each rule has a target tar attached where $tar \in \{here, out, in\}$ with the interpretation that $here$ indicates that the evolved word remains in the same region, out indicates that the evolved word goes out as a whole, to the immediate outer region (except when the word is in the outermost skin membrane in which case it is lost) and in indicates the evolved word enters again as a whole, the immediate inner region; i_0 is the label of the output region.

In a computation in Π, starting from the words initially present in the regions, words evolve by rewriting using context-free rules. Words, if any, in all the regions evolve by application of a rule at the same time, if there is any rule in the region applicable to the word. But the rewriting is done in a sequential manner at the level of a region which means that only one rule is applied to a word in a region. A computation halts and is successful if no applicable rule is available in the regions. The words collected in the output region at the end of successful computations constitute the language which is generated by the system Π. Note that in this variant of P system of the string variety, the output region is one of the regions that constitute the membrane structure and hence there is no role for string objects, if any, in the environment which is the region outside the outermost skin membrane. Note also that in a P system, by definition the objects (here the strings of symbols), if any, in each region can evolve if there are rules applicable to the objects in the region and hence there is no need for specifying an input region. Also note that an object might get stuck in a region due to the absence of any rule which would allow it to enter an inner or outer region, irrespective of the computation is successful or not. But this does not cause any difficulty as the words in the output region, if any, over terminal symbols are only collected in defining the language generated when the computation halts and is successful.

FIGURE 1. First and Second polygons of Sierpiński curve

An illustration is now given by constructing a P system of the above type with two membranes, one inside another and generating the language
$\{x^n y^n \mid n \geq 1\}$ over the alphabet $\{x, y\}$. Consider the P system

$$\Pi_1 = (\{X, Y, x, y\}, \{x, y\}, [_1 [_2]_2]_1, \{XY\}, \emptyset, P_1, P_2, 2)$$

where $P_1 = \{X \rightarrow xX(in)\}$, $P_2 = \{Y \rightarrow yY(out), X \rightarrow x(here), Y \rightarrow y(here)\}$.

A computation can start with region 1 as only this region has an initial word XY. Applyig the rule $X \rightarrow xX(in)$ in region 1, the word xXY is generated, which goes to membrane 2 as the associated target is in. In region 2, if the rule $Y \rightarrow yY$ with target "out" is applied, the string generated is $xXyY$ which goes back to membrane 1. The procedure can be repeated. An appropriate sequence of application of the rules to halt the computation is to apply in region 2, the rules $X \rightarrow x$ and $Y \rightarrow y$ (one of these followed by the other) and words of the form $x^n y^n$ are generated with the computation halting as no other rule could be applied. Note that if , after applying the rule $X \rightarrow x$ in region 2, the rule $Y \rightarrow y(out)$ is applied then the word generated is sent to region 1 where it gets stuck and hence cannot contribute any word to the language generated.

3. POLYGON PATTERNS OF SIERPIŃSKI'S SNOWFLAKE CURVE

We now consider Sierpiński's original construction of a space filling curve [14, 13] which is considered as a mapping from a closed unit interval onto a square. As there is another form [13] of Sierpiński space-filling curve which fills an isosceles triangle, following [17], we refer to this original space-filling curve as Sierpiński square snowflake curve or in short, Sierpiński curve. For mathematical details relating to this curve, we refer to [13, Page 50].

The first two elements of the sequence of the polygon patterns that approximate the Sierpiński curve are shown in Fig. 1 with the polygons passing through neighbouring subtriangles of the square.

Motivated by the method in [15] for generating the sequences of the polygon patterns of the space-filling curves of Peano and Hilbert, we propose a method of generation of the sequence of polygon patterns, which is infinite, corresponding to the the Sierpiński curve. The difference in the methods is that array grammars are used in [15] for the generation of the polygons while we make use of a P system model. Note that in the array grammar model in [15], an array grammar, called CF Parantheses Kolam array grammar, is given for patterns of Peano curve and this grammar uses two kinds of rules with the first kind involving CF type rules with intermediate symbols (which act as terminal symbols in this step)

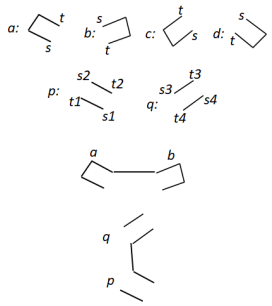

FIGURE 2. (i) Primitives a, b, c, d, p, q of polygons (shown on the top) (ii) Primitives joined according to certain joining rules (shown in the bottom)

followed by suitable array languages corresponding to the intermediate symbols, which are replaced by arrays of these languages. Hence the working of the array grammar, although correct, is more cumbersome compared to the working of the P system model, proposed here for patterns of Sierpiński curve. Besides this advantage, the versatile nature of the P system model is also brought out, as the P system model has been used in different kinds of applications [10]. First we informally describe the proposed method which involves the following steps:
(i) Identify a finite set of primitive patterns in the polygon patterns and encode the polygon patterns by arrays with the symbols of the arrays representing the primitive patterns ;
(ii) Construct joining rules which when applied allow to join the primitives corresponding to the symbols in the arrays resulting in the polygonal patterns of the Sierpiński curve ;
(iii) Construct a P system of the type mentioned in Section 2 but with the objects in the membrane regions of the P system as arrays rectangular in shape and evolution rules as context-free type array rewriting rules with rewriting done in parallel in the sense of replacing at the same time with arrays of the same size each of the symbols in an array in the regions of the P system.

Encoding by arrays the Polygons of Sierpiński Curve

$$\begin{array}{cc} a & b \\ c & d \end{array} \qquad\qquad \begin{array}{cccc} a & b & a & b \\ c & p & q & d \\ a & q & p & b \\ c & d & c & d \end{array}$$

FIGURE 3. Arrays encoding the polygon patterns in Fig. 1

We illustrate the encoding by considering the two polygonal patterns of Sierpiński curve shown in Fig. 1. The arrays encoding these two polygonal patterns are given in Fig. 3.

Joining Rules

We now construct a finite set J of joining rules that allow the primitives corresponding to the symbols in the arrays in Fig. 3 to be joined appropriately resulting in the polygon patterns in Fig. 1.

The set J consists of the following rules:

$$1 : (a,t) \rightarrow^h (b,s),\ 2 : (a,t) \rightarrow^h (q,s_3),\ 3 : (b,t) \rightarrow^v (d,s),\ 4 : (b,t) \rightarrow^v (p,s_2),$$
$$5 : (c,t) \rightarrow^v (a,s),\ 6 : (c,t) \rightarrow^v (p,s_1),\ 7 : (d,t) \rightarrow^h (c,s),\ 8 : (d,t) \rightarrow^h (q,s_4),$$
$$9 : (p,t_1) \rightarrow^h (c,s),\ 10 : (p,t_1) \rightarrow^h (q,s_4),\ 11 : (p,t_2) \rightarrow^h (b,s),$$
$$12 : (p,t_2) \rightarrow^h (q,s_3),\ 13 : (q,t_3) \rightarrow^v (p,s_1),\ 14 : (q,t_3) \rightarrow^v (a,s),$$
$$15 : (q,t_4) \rightarrow^v (p,s_2),\ 16 : (q,t_4) \rightarrow^v (d,s)$$

The interpretation of a rule in J is that an end labelled by t or t_i, $1 \leq i \leq 4$ in the primitive on the left of the rule is joined by a horizontal or vertical edge (depending on h or v present in the rule) to the end labelled by s or s_i, $1 \leq i \leq 4$ in the primitive on the right of the rule, with the primitives being adjacent to each other in an array. For example, the rule $(a,t) \rightarrow^h (b,s)$ means that the end labelled t in the primitive a is joined by a horizontal edge to the end labelled s in the primitive b. Likewise the rule $(q,t_4) \rightarrow^v (p,s_2)$ means that the end labelled t_4 in the primitive q is joined by a vertical edge to the end labelled s_2 in the primitive p. The symbols s, s_i, $1 \leq i \leq 4$ and t, t_i, $1 \leq i \leq 4$ are indicative of how to join the primitives and are not retained in the generated polygon. For example, the effects of the two rules $(a,t) \rightarrow^h (b,s)$ and $(q,t_4) \rightarrow^v (p,s_2)$ are shown in Fig. 2.

4. A P SYSTEM FOR POLYGON PATTERNS OF SIERPIŃSKI CURVE

We now construct a P system Π with two membranes one within another, array objects and parallel rewriting in order to generate the sequence of polygon patterns of Sierpiński curve.

Let $\Pi = (V, \Sigma, \mu, A_1, A_2, P_1, P_2, 2)$ where $V - \Sigma = \{A, B, C, D, P, Q\}$ is a set of nonterminals, $\Sigma = \{a, b, c, d, p, q\}$ is a set of terminals which represent the primitives shown in Fig. 2 , $\mu = [_1\,[_2\,]_2\,]_1$ is the membrane structure with the membrane labelled 2 inside the membrane labelled 1. The arrays that are axioms in membranes labelled 1 and 2 are respectively given by the elements of A_1 and A_2 where $A_1 = \{ \begin{matrix} A & B \\ C & D \end{matrix} \}$ and $A_2 = \emptyset$. The membrane with label 2 is the output membrane. The rules of the membranes labelled 1 and 2 are given by the elements of P_1 and P_2. The set P_1 contains the context-free rules $r_1, r_2, r_3, r_4, r_5, r_6$ all having target *here* and the rules $r_7, r_8, r_9, r_{10}, r_{11}, r_{12}$ all having target *in*. The rules are as follows:

$$r_1 : A \rightarrow \begin{matrix} A & B \\ C & P \end{matrix}, r_2 : B \rightarrow \begin{matrix} A & B \\ Q & D \end{matrix}, r_3 : C \rightarrow \begin{matrix} A & Q \\ C & D \end{matrix},$$

$$r_4 : D \rightarrow \begin{matrix} P & B \\ C & D \end{matrix}, r_5 : P \rightarrow \begin{matrix} P & B \\ C & P \end{matrix}, r_6 : Q \rightarrow \begin{matrix} A & Q \\ Q & D \end{matrix},$$

$$r_7 : A \to a, r_8 : B \to b, r_9 : C \to c,$$

$$r_{10} : D \to d, r_{11} : P \to p, r_{12} : Q \to q.$$

The set P_2 is empty. Note that the rewriting is done in a region by applying the rules with the same target in the region in parallel to an array in the region to which the rules are applicable. Note also that a set of applicable rules replace each of the symbols in an array α by an array β of the same size $i.e.$ having the same number of rows and the same number of columns. Also, rules in a region with the same target indication are applied at a time. A computation in Π takes place as follows: Since only the region 1 has an axiom array $\begin{matrix} A & B \\ C & D \end{matrix}$, application of the rules r_7 to r_{12} all having the same target *in* yields the array $\begin{matrix} a & b \\ c & d \end{matrix}$ which is sent to the output region 2 due to target indication *in*. Note that this array is the encoding of the first polygon (Fig. 1 (on the left)) in the patterns of Sierpiński curve. If in region 1, the rules r_1 to r_6 with target indication *here* are applied to the axiom array $\begin{matrix} A & B \\ C & D \end{matrix}$, the resulting array $\begin{matrix} A & B & A & B \\ C & P & Q & D \\ A & Q & P & B \\ C & D & C & D \end{matrix}$ which remains in the same region due to target indication *here*. If now the rules r_7 to r_{12} all having the same target *in* are applied, then this yields the array $\begin{matrix} a & b & a & b \\ c & p & q & d \\ a & q & p & b \\ c & d & c & d \end{matrix}$ which is sent to the output region 2 due to target indication *in*. Note that this array is the encoding of the second polygon (Fig. 1 (on the right)) in the patterns of Sierpiński curve. The process can repeat. The arrays collected in the region 2 are the encodings of the polygon patterns of Sierpiński curve.

Note that the polygon patterns are obtained by applying the joining rules mentioned earlier to the symbols in the arrays generated by the P system Π with the symbols standing for the primitives of the polygon patterns of Sierpiński curve.

5. Conclusion

We have considered here approximating polygon patterns of Sierpiński curve in the two-dimensional (2D) plane. In [2] extension of Sierpiński curve from the 2D to the three dimensional (3D) case is considered. It will be of interest to examine the possibility of costructing P systems for generation of the corresponding 3D approximating patterns of the Sierpiński curve.

Acknowledgements

The authors are grateful to the reviewers for their time spent and thank them for their helpful and meticulous comments which enabled us to revise the paper presenting the work more clearly.

REFERENCES

1. P.S. Azeezun Nisha, S. Hemalatha, A. K. Nagar, K.G. Subramanian, *Patterns of Sierpiński Square Snowflake Curve and Their Generation with P Systems*, Paper presented in the Int. Conf. on Mathematical Computer Engineering, Vellore Institute of Technology, Chennai, India 2018.
2. M. Bader, *Space-filling curves*, Springer-Verlag Berlin, Heidelbrg, 2013.
3. R. Ceterchi, K.G. Subramanian, I. Venkat, *P Systems with Parallel Rewriting for Chain Code Picture Languages*. Evolving Computability - 11th Conference on Computability in Europe, CiE 2015, Bucharest, Romania, 2015, Proceedings, Lecture Notes in Computer Science, **9136**, Springer, (2015), 145–155 .
4. R. Ceterchi, A.K. Nagar, K.G. Subramanian, *Chain Code P System Generating a Variant of the Peano Space-filling Curve*, In: Hinze T., Rozenberg G., Salomaa A., Zandron C. (eds) Membrane Computing. CMC 2018, Lecture Notes in Computer Science, vol 11399, Springer, Cham (2019).
5. R. Ceterchi, A.K. Nagar, K. G. Subramanian, *Approximating polygons for space-filling curves generated with P systems*, C. Graciani et al. (Eds.): Pérez-Jiménez Festschrift, Lecture Notes in Computer Science, **11270**, (2018), 57–65 .
6. D. Giammarresi, A. Restivo, *Two-dimensional languages In: Rozenberg G., Salomaa, A. (eds.) Handbook of Formal Languages*, **3**, pp. 215–267. Springer, Heidelberg (1997).
7. D. Hilbert, *Über die stetige Abbildung einer Linie auf ein Flächenstück*, Math. Annln., **38**, (1891), 459–460.
8. Gh. Păun, *Computing with membranes*, J. Comp. System Sci., **61**, (2000), 108- 143.
9. Gh. Păun, *Membrane Computing: An Introduction*, Springer-Verlag Berlin, Heidelbrg, 2000.
10. Gh. Păun, Rozenberg, G., Salomaa, *A.: The Oxford Handbook of Membrane Computing*, Oxford University Press, New York, USA, 2010.
11. G. Peano, *Sur une courbe qui remplit toute une aire plane*, Math. Annln., **36** (1890) 157–160.
12. G. Rozenberg, A. Salomaa, (eds.) *Handbook of Formal Languages (*3), Springer-Verlag, Berlin, 1997.
13. H. Sagan, *Space-filling curves* Springer-Verlag, New York, 1994.
14. W. Sierpiński, *Sur une nouvelle courbe continnue qui remplit toute une aire plane*, Bull. Acad. Sci. de (Sci. math et nat.,) Série A), (1912), 462–478.
15. R. Siromoney, K.G. Subramanian, *Space-filling curves and Infinite graphs*, Lecture notes in Comp. Sci. **153**, (1983), 380–391.
16. K.G. Subramanian, S. Sriram, B. Song, L. Pan, *An Overview of 2D Picture Array Generating Models Based on Membrane Computing*, Reversibility and Universality, (2018), 333–356 .
17. D. Wells, *The Penguin Dictionary of Curious and Interesting Geometry*, London: Penguin, p. 229 (1991).