REPRINTED FROM:

FRACTALS IN THE FUNDAMENTAL AND APPLIED SCIENCES

Proceedings of the First IFIP Conference on
Fractals in the Fundamental and Applied Sciences
Lisbon, Portugal, 6-8 June, 1990

Edited by

Heinz-Otto PEITGEN
University of Bremen
Bremen, Germany

José Marques HENRIQUES
Technical University of Lisbon
Lisbon, Portugal

Luís Filipe PENEDO
IBM Portugal
Portugal

1991

NORTH-HOLLAND
AMSTERDAM • NEW YORK • OXFORD • TOKYO

Fractals in the Fundamental and Applied Sciences
H.-O. Peitgen, J.M. Henriques & L.F. Penedo (Editors)
Elsevier Science Publishers B.V. (North-Holland)

SYNTHESIS OF SPACE-FILLING CURVES ON THE SQUARE GRID

Przemyslaw Prusinkiewicz[†], Aristid Lindenmayer[‡], and F. David Fracchia[†]

[†]Department of Computer Science
University of Regina
Regina, Saskatchewan, S4S 0A2
Canada

[‡]Theoretical Biology Group
University of Utrecht
Padualaan 8, 3584 CH Utrecht
The Netherlands

This paper investigates a relationship between a class of polygonal fractal curves and tilings of the plane. The curves are constructed by connecting polygons inscribed into tiles using inserted line segments. Conditions are imposed to ensure that the resulting curve is approximately space-filling, self-avoiding, simple (single-stroke) and self-similar. Such curves are referred to as FASS curves. Their relationship to tilings is used to explain the design of classic space-filling curves, and is applied to synthesize new FASS curves. A program which generates FASS curves on a square grid is described, and examples of curves produced using this program are given. The FASS curves are expressed using the formalism of L-systems with turtle interpretation. Thus, the paper provides a method for synthesizing L-systems that generate classic and new space-filling curves.

1 INTRODUCTION

Although the first space-filling curves were discovered almost a century ago, an aura of mystery still surrounds them. One reason is that a relatively small number of such curves have been reported to date. The classic space-filling curves were given by Peano [16], Hilbert [10] and Sierpiński [22]. Subsequent examples were found by Heighway [6], Gosper [7, 12], Szilard and Quinton [26], and Dekking [4]. Are there more space-filling curves? If so, what do they look like? Can they be found algorithmically?

Before we attempt to answer these questions, we must characterize the class of curves under consideration. Their distinctive features can be described informally as follows:

- *Self-similarity* — the curve can be constructed by the recursive application of a set of rules for connecting components.

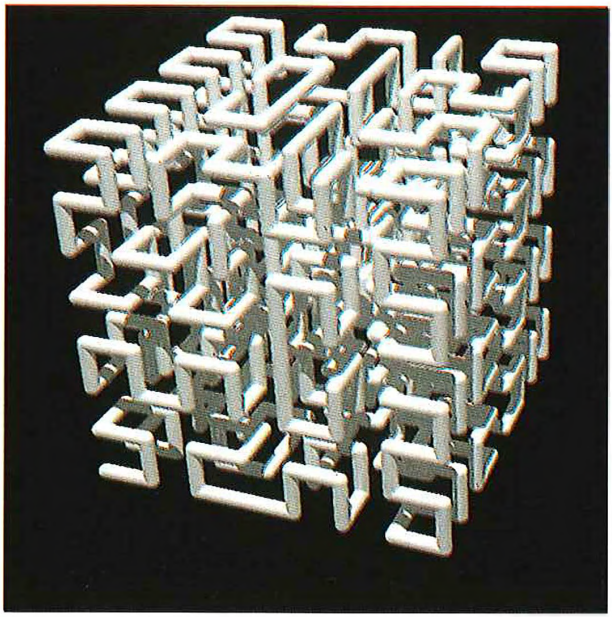

Plate 1: A three-dimensional extension of the Hilbert curve.

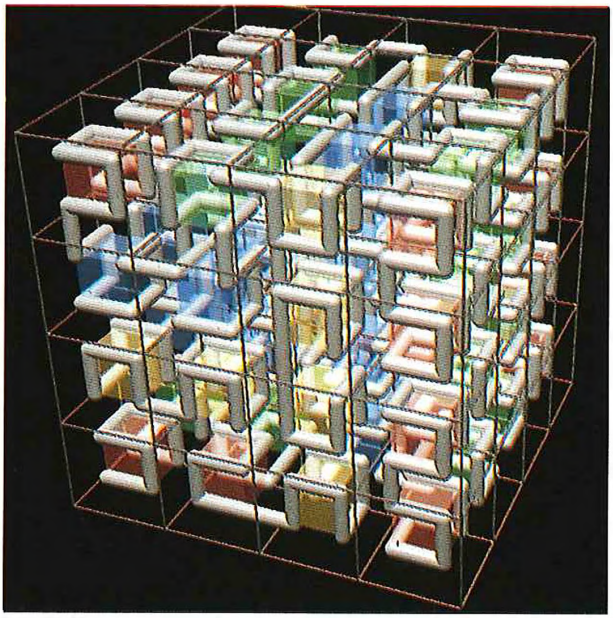

Plate 2: The same curve shown on a grid representing boundaries of the 3D 'tiles'. The semitransparent cubes are the 3D equivalent of frames.

- *Approximate space-filling* — the curve passes within a small distance from all points of a given two-dimensional figure, such as a square, which surrounds the curve. This distance can be arbitrarily reduced by carrying on the recursive construction to an appropriate level.

- *Self-avoidance* — the segments of the curve do not touch nor intersect.

- *Simplicity* — the curve can be drawn by a single stroke of a pen, without lifting the pen nor drawing any segments more than once.

The curves which share these properties are called FASS curves (an acronym for space-filling, self-avoiding, simple and self-similar). In most cases, we additionally assume that the curves consist of straight line segments which run vertically or horizontally along the edges of a square grid.

In this paper we present many FASS curves other than the classic space-filling curves. To construct them, we investigate a relationship between FASS curves and tilings of the plane. On this basis, we develop an algorithm which generates all FASS curves on a square grid, given a type of recursive grid subdivision. The output of this algorithm specifies FASS curves using the formalism of Lindenmayer systems [11] with geometric interpretation [17, 18] based on turtle geometry [1]. In other words, the algorithm outputs L-systems that can be used as input for a fractal-generating program [18].

Relationships between recursive tilings of the plane and space-filling curves were used to explain the design of specific curves in the past [7, 10, 14, 15, 22]. McKenna [14] observed that this relationship can form a basis for an algorithm which will generate a class of space-filling curves automatically. This paper shares the spirit of McKenna's approach. However, the actual technique is quite different. McKenna considered curves whose segments lay *on the edges* of the tiles, while this paper investigates curves having components drawn *inside* the tiles and connected by line segments which do not coincide with the tile edges. This yields dramatically different results. For example, McKenna arrived at one FASS curve corresponding to the recursive grid subdivision into 5×5 components (the E curve), while the method described in this paper produces 43 pairs of so-called uniform curves, and hundreds more non-uniform curves, assuming the same subdivision factor (Section 7).

Our paper begins with an intuitive description of the proposed algorithm for FASS curve generation (Section 2). A detailed presentation follows. In Section 3 the class of FASS curves is defined formally. Numerical parameters are introduced to characterize the properties of self-avoidance and of approximate space-filling. Section 4 investigates the construction of larger curves from smaller components by means of the set-theoretic union operation. It is shown that, under certain conditions, union preserves the properties of self-avoidance and of approximate space-filling of the component curves. Particular consideration is given to the case where the components are associated with an array of square tiles. Section 5 introduces a string notation based on turtle geometry as a formalism for specifying connections between components, and identifies self-similarity with the recursive application of connecting patterns. The scope of the discussion is limited to single-stroke curves. Section 6 relates recursive connecting patterns to L-systems. They are applied to generate images of FASS curves using a graphical interpretation developed in earlier papers [17, 18]. Section 7 combines the results from previous sections in an algorithm which generates FASS curves on a square grid. Examples of the output are

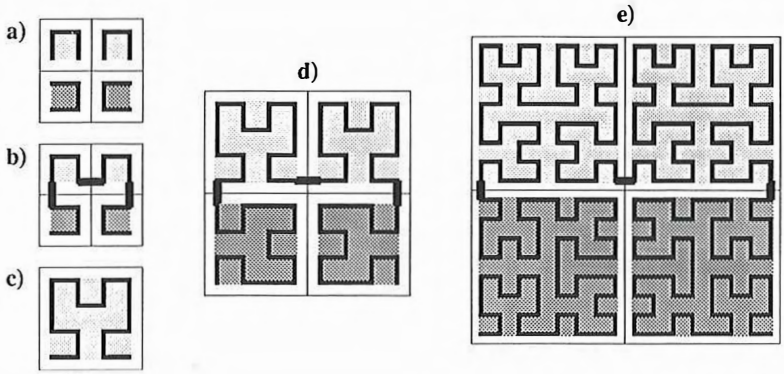

Figure 1: Construction of a FASS curve (the Hilbert curve). The shaded areas indicate frames which bound inscribed polygons.

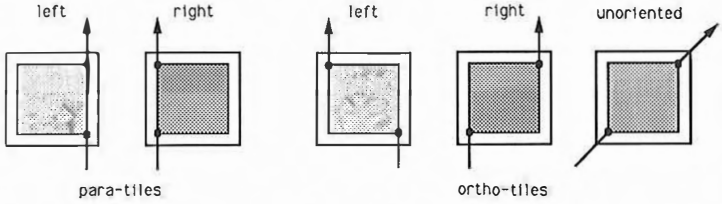

Figure 2: Types of tiles. Shading indicates tile orientation.

given in the form of connecting patterns which yield L-systems producing FASS curves. Sample L-systems and curves are presented. Section 8 discusses possible extensions and lists problems open for further research.

2 INTUITIVE DESCRIPTION OF THE METHOD

This section presents the intuition behind the proposed method for the generation of FASS curves. Consider an array of 2×2 square tiles. Each tile includes a smaller square, called a *frame*. The edges of the frame run at some distance from the tile's edges. Each frame bounds an open self-avoiding polygon. The endpoints of this polygon coincide with two vertices of the frame, called *active* vertices (Figure 1.a). Assume that a single-stroke curve running through all tiles can be constructed by connecting the active vertices of the neighboring frames using short horizontal or vertical line segments (Figure 1.b). The key to the construction of convoluted FASS curves lies in the recursive repetition of this *connecting pattern*. Thus, the array of 2×2 connected tiles is considered a *macrotile* which contains an open polygon inscribed into a *macroframe* (Figure 1.c). An array of 2×2 macrotiles is formed, and the polygons inscribed into the macroframes are connected together (Figure 1.d). This construction is carried out recursively, with 2×2 macrotiles at level m yielding one macrotile at level $m + 1$ (Figure 1.e).

In order to make the recursive construction possible, the polygon inscribed into a

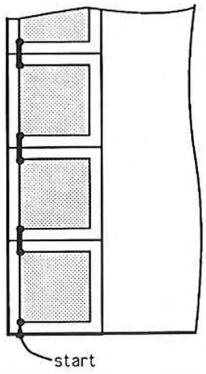

Figure 3: A possible arrangement of uniformly oriented, connected para-tiles.

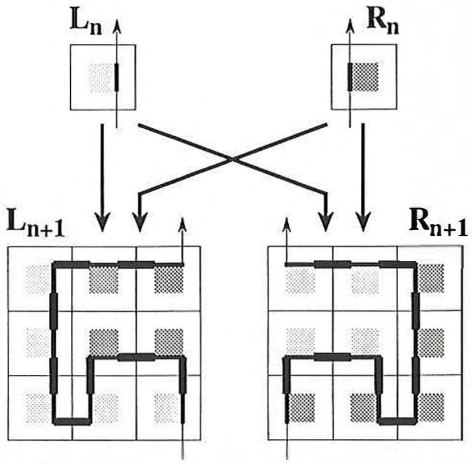

Figure 4: Construction of a space-filling curve using para-tiles involves simultaneous recursion with respect to both tile orientations.

macrotile must start and end at the vertices of its macroframe. Furthermore, the active vertices of the frame and of the macroframe must be at the same relative positions. Using terminology borrowed from organic chemistry, we call these positions *ortho* and *para*, and consequently we distinguish between *ortho-tiles* and *para-tiles* (Figure 2). For a given direction of line drawing, we further distinguish between *left* and *right* tiles. The orientation of a para-tile is determined by the position of the frame with respect to the oriented line connecting its active vertices. In the case of ortho-tiles the distinction between left and right tiles is more arbitrary.

A reexamination of Figure 1 reveals the use of both left and right para-tiles at each recursion level. This corresponds with the observation that, starting from a vertex of a macrotile, the only possible arrangement of uniformly oriented, connected para-tiles is a row (Figure 3) or a column. As neither arrangement covers a square macrotile, tiles of both orientations must be used (Figure 4). In the case of a curve constructed from ortho-tiles, unoriented tiles may be used (Figure 5.a). However, two tile orientations yield a significantly different curve based on the same connecting pattern (Figure 5.b). Curves which combine ortho- and para-tiles in a single design are also possible (Figure 6). In this case, the recursion may involve more than two tile types.

The recursive pattern of tile connections does not depend on the shape of the polygons inscribed in the tiles. Consequently, it is convenient to consider all curves employing the same connecting pattern as belonging to one class. The appearance of FASS curves representing the same class may vary largely as a function of the shape (Figure 7) and size (Figure 8) of the initial polygons. Of special interest are curves in which the initial polygons are reduced to single points. Such curves are *pure* in the sense that they are fully defined by the underlying connecting patterns (for example, see Figure 8.a). The following discussion concentrates on pure curves, since their extension to curves with non-empty initial polygons is obvious.

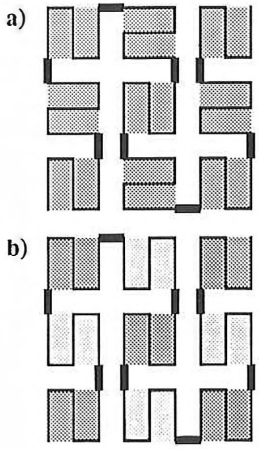

Figure 5: Two related curves constructed using: (a) unoriented ortho-tiles, and (b) left and right ortho-tiles.

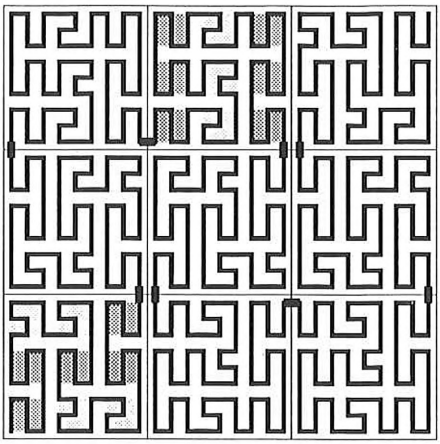

Figure 6: Example of a curve incorporating both para-tiles and ortho-tiles. Some of them are indicated by light-shaded and dark-shaded frames, respectively. The wide segments show connections between macrotiles.

3 BASIC NOTIONS

3.1 Finite approximations of space-filling curves

The key problem related to the characterization of the curves considered in this paper can be described by referring to the Hilbert curve shown in Figure 1.e. Should we consider it as a finite approximation of the "true" infinite curve that would result from an infinite application of recursive construction rules? Or should we consider this finite curve in its own right?

Frequently, the first approach is taken. For example, in the context of Koch constructions, Mandelbrot [12] writes (p. 39):

> Strictly speaking, the triangle, the Star of David, and the finite Koch teragons are of dimension 1. However, both intuitively and from the pragmatic point of view of the simplicity and naturalness of the corrective terms required, it is reasonable to consider an advanced Koch teragon as being closer to a curve of dimension $\log 4/\log 3$ than to a curve of dimension 1.

A finite curve can be considered as an approximate rendering of an infinite curve as long as the interesting properties of both are closely related. However, this is not the case in the domain of space-filling curves, where the properties of the infinite limit curve and its finite approximations are often different. Specifically, the Mazurkiewicz theorem [13, 15] states that there is no continuous, one-to-one mapping of a line segment onto a square. Consequently, no curve can be both self-avoiding and strictly space-filling. On the other hand, a finite approximation of a given space-filling curve *can* be both self-avoiding and

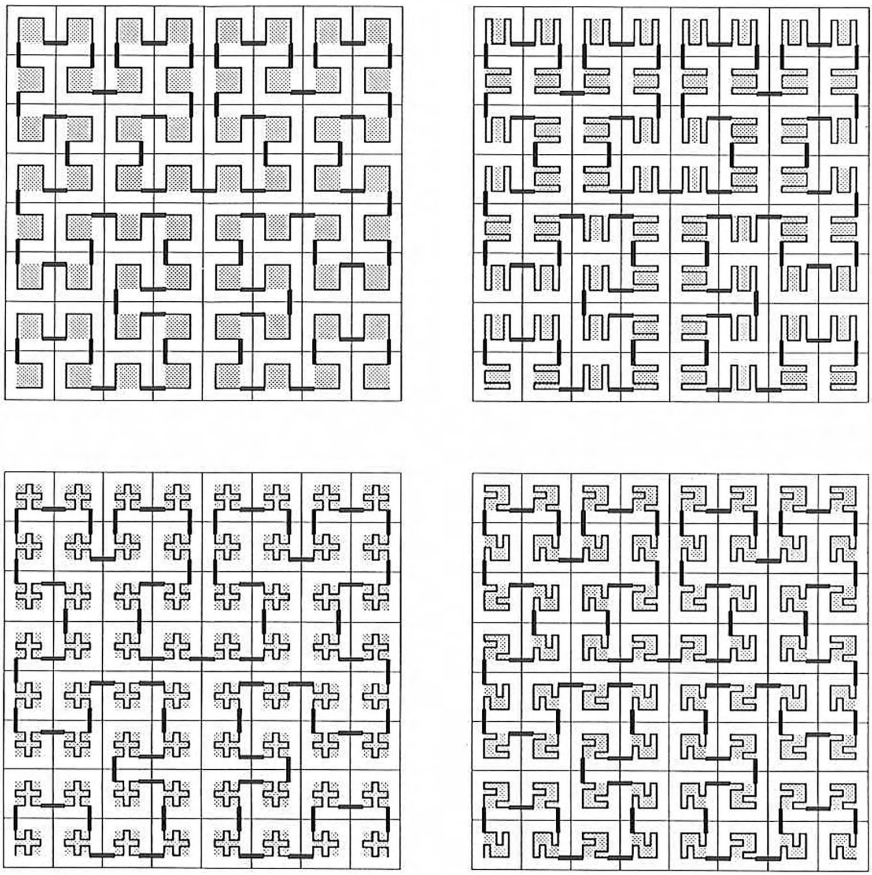

Figure 7: Example of curves that have the same connecting pattern and different initial polygons.

approximately space-filling, as exemplified by Figures 1, 5, 6 and 7. Such finite curves are the focus of this paper.

3.2 Characterization of FASS curves

This section defines terms and parameters used to describe the family of FASS curves. The discussion is restricted to *sets of lines* (finite sets of straight line segments). Specifically, a *polygon* or *single-stroke* curve (Figure 9) is a set of lines connected by the endpoints (*vertices* of the polygon) in a sequence. No two segments may overlap (although they may intersect). If the last segment in the sequence shares a vertex with the first one, the polygon is *closed*, otherwise it is *open*. The key properties of FASS curves, self-avoidance and approximate space-filling, are defined below.

Let $\rho(P, Q)$ denote the Euclidean distance between points P and Q. The *distance*

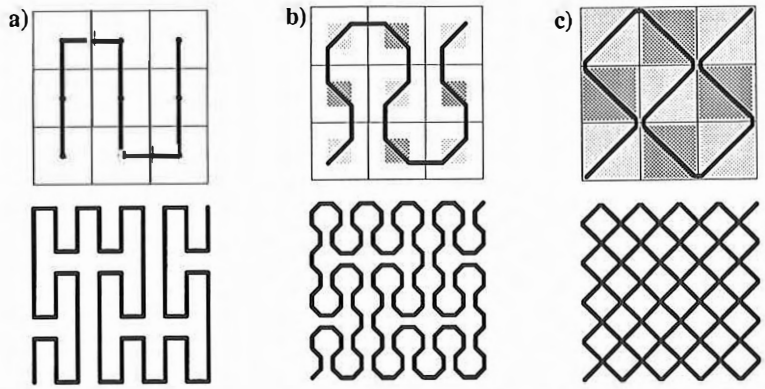

Figure 8: The appearance of a space-filling curve depends on the size of the initial polygons.

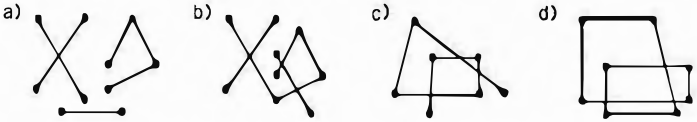

Figure 9: Examples of sets of lines: (a) arbitrary, (b) connected, (c) an open polygon, (d) a closed polygon.

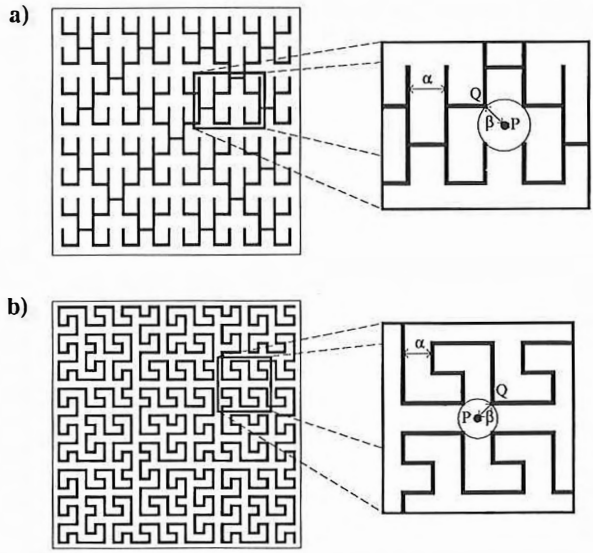

Figure 10: Two self-avoiding, approximately space filling curves. Curve (a) is not a single-stroke curve, while (b) is a single-stroke curve. Parameters α and β denote the distance of self-avoidance and the accuracy of space filling, respectively.

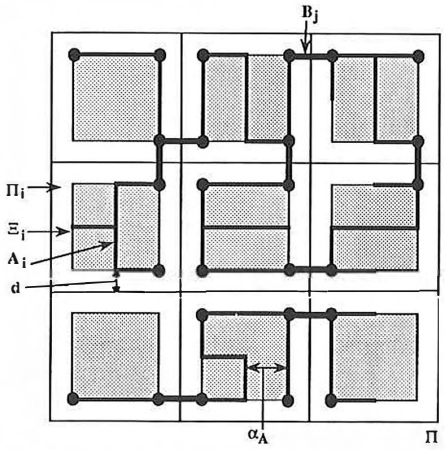

Figure 11: Illustration of Theorem 1. See the proof for the explanation of symbols.

between lines a, b is defined as:

$$\text{dist}(a, b) = \min\{\rho(P, Q) : P \in a \text{ and } Q \in b\}$$

A set C of lines $\{a_0, \ldots, a_{n-1}\}$ is *self-avoiding at distance* $\alpha > 0$ if the minimum distance between lines a_i which do not share a vertex is equal to α (Figure 10).

Let C be a curve and Π be a region of the plane containing C. We will say that C *fills* Π *with accuracy* $\beta > 0$, if for any point $P \in \Pi$ there exists a point $Q \in C$ such that the distance $\rho(P, Q)$ is less then or equal to β (Figure 10).

4 CURVE COMPOSITION

In this section we investigate curve composition as a method for creating larger FASS curves from smaller components. We are primarily concerned with the preservation of self-avoidance under composition. Other properties of FASS curves will be preserved automatically by the construction.

Lemma 1. Let A and B be two sets of lines, self-avoiding at distances α_A and α_B. Assume that the intersection between any pair of lines $a \in A$ and $b \in B$ is either the empty set or a shared vertex, and that the minimum distance between those lines which do not share a vertex is equal to $d_{A,B} > 0$. The set of lines $C = A \cup B$ is self-avoiding at distance $\alpha_C = \min\{\alpha_A, \alpha_B, d_{A,B}\}$.

Proof. The parameter α_C is determined by the distance between the two closest segments of C which do not share a vertex. Three possibilities may occur: (a) both segments are included in A, (b) both segments are included in B, or (c) one segment is included in A and another in B. Consequently, $\alpha_C = \min\{\alpha_A, \alpha_B, d_{A,B}\}$. □

Consider an array consisting of $N = n \times n$ squares Π_i (Figure 11). They are assumed to partition the entire square $\Pi = \bigcup_{i=1}^{N} \Pi_i$ without gaps or overlaps, and thus they can be referred to as *tiles* [8]. Each tile contains a smaller square Ξ_i called a *frame*, with edges running at a distance d from the tile's edges. Let each frame Ξ_i bound a set of

lines A_i, self-avoiding at distance α_A. Some vertices of A_i may coincide with vertices of its bounding frame Ξ_i; such vertices are called *active*. If a line does not end at an active vertex, the distance between the line and this vertex is assumed to be at least α_A. A horizontal or vertical line that connects adjacent active vertices of two neighboring frames is called an *admissible connecting segment*. We obtain the following result.

Theorem 1. For any set of J admissible connecting segments B_j, where $j = 1, 2, \ldots, J$, the set of lines

$$C = \bigcup_{i=1}^{N} \Lambda_i \cup \bigcup_{j=1}^{J} B_j$$

is self-avoiding at distance $\alpha_C \geq \min\{\alpha_A, 2d\}$.

Proof. The composed set C can be viewed as the union of two sets of lines, $\tilde{A} = \bigcup_{i=1}^{N} A_i$, and $\tilde{B} = \bigcup_{j=1}^{J} B_j$. From the repetitive application of Lemma 1 to sets $\tilde{A}_l = \bigcup_{i=1}^{l} A_i$ and A_{l+1}, where $l = 1, 2, \ldots, N-1$, it follows that the set $\tilde{A} = \tilde{A}_N$ is self-avoiding at distance $\alpha_1 \geq \min\{\alpha_A, 2d\}$. The definition of admissible connecting segments implies that any two of them either share a vertex or are at distance $\alpha_2 \geq \min\{\alpha_A, 2d\}$ from each other. Thus, the set $\tilde{B}$ is self-avoiding at distance α_2. Finally, two cases are possible when considering the position of a segment $A_i \subset \tilde{A}$ with respect to a segment $B_j \subset \tilde{B}$. If B_j is not connected to the frame Ξ_i which encloses A_i, then B_j is at least at distance $2d$ from Λ_i. Otherwise, B_j is connected to A_i at an active vertex P. Hence, B_j is at least at distance α_A from any segment of A_i which does not pass through P. Consequently, the minimum distance between two segments $a \in \tilde{A}$ and $b \in \tilde{B}$ which do not share a vertex satisfies the inequality $d_{\tilde{A},\tilde{B}} \geq \min\{2d, \alpha_A\}$. By referring to Lemma 1 one more time we obtain that $C = \tilde{A} \cup \tilde{B}$ is a self-avoiding curve at distance $\alpha_C = \min\{\alpha_1, \alpha_2, d_{\tilde{A},\tilde{B}}\} \geq \min\{\alpha_A, 2d\}$. □

A square array of $n \times n$ tiles Π_i, with sets A_i connected using segments B_j, can be considered a first-level *macrotile* Π_k^1, with the set $A_k^1 = C$ inscribed into a *macroframe* Ξ_k^1. Subsequently, an array of $n \times n$ first-level macrotiles can be formed, and the respective sets of lines connected again. This construction can be carried out recursively, with $n \times n$ m-level line sets A_k^m joined into a set A_l^{m+1} inside a tile Π_l^{m+1}.

Theorem 2. For any recursion level $m \geq 0$, the set of lines A_l^m is self-avoiding at distance $\alpha_A^m \geq \min\{\alpha_A, 2d\}$.

Proof: by induction on m. For $m = 0$ the hypothesis is obvious: $\alpha_A^0 = \alpha_A \geq \min\{\alpha_A, 2d\}$. Assume the hypothesis true for some $m \geq 0$. The distance d between the boundaries of a macrotile Π_l^m and its macroframe Ξ_l^m does not depend on the recursion level m (Figure 12). Hence, from Theorem 1 we obtain:

$$\alpha_A^{m+1} \geq \min\{\alpha_A^m, 2d\} \geq \min\{\min\{\alpha_A, 2d\}, 2d\} = \min\{\alpha_A, 2d\}. \quad \square$$

Theorem 1 places few constraints on the way the connecting segments B_j are placed. For example, Figure 11 shows that the resulting set C need not be connected; at the same time some components may be connected by more than one path. Similarly, Theorem 2 does not impose any relationship between connections at level m and $m+1$. Consequently, it is possible to construct irregular FASS curves, such as the curve shown in Figure 17. However, we are primarily interested in curves which are connected, single-stroke, and can by formed by repeating the same connecting pattern at any recursion level. We also want these curves to be approximately space-filling, but this results automatically from the assumption that they pass through all tiles. Construction of curves fulfilling all these conditions is discussed in the next section.

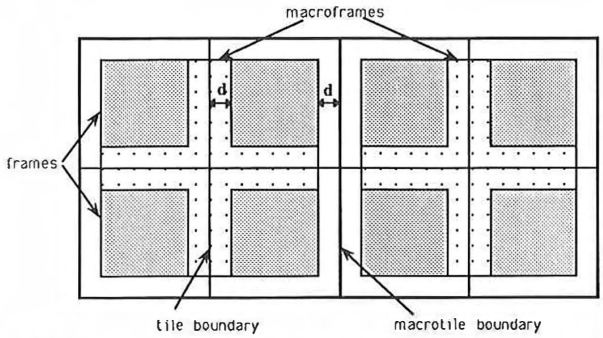

Figure 12: Distances between the tiles and the frames do not depend on the recursion level.

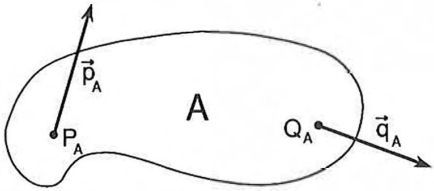

Figure 13: Description of a subfigure A.

5 CONNECTING PATTERNS AND SELF-SIMILARITY

The notion of a connecting pattern formalizes the manner in which components of a figure are joined together without specifying what these components are. Since the components of a single-stroke pattern can be put in a sequence, we describe these patterns using a string notation. In addition to its descriptive value, this notation can be directly used to generate FASS curves by applying a string-rewriting mechanism to produce a string of symbols and interpreting it graphically to create the final image. To this end, we employ an approach based on turtle geometry [1], which extends the turtle interpretation of L-systems [17, 18] by including symbols that represent entire subfigures. Their relative positions are specified by means of distinguished contact points associated with each subfigure, using a concept first introduced in the Picture Definition Language [21].

Consider a set of subfigures $\mathcal{A}$. Associated with each subfigure $A \in \mathcal{A}$ are (Figure 13):

- two *contact points*, called the *entry point* P_A and the *exit point* Q_A, and
- two *direction vectors*, called the *entry vector* $\vec{p_A}$ and the *exit vector* $\vec{q_A}$.

A *connecting pattern* is defined by a string x over the alphabet $\mathcal{V} = \{+, -, \mathcal{F}\} \cup \mathcal{A}$. This string is interpreted as a sequence of commands which control a LOGO-style turtle [1]. Given two parameters, the *step size* s and the *angle increment* δ, the turtle responds to symbols of $\mathcal{V}$ as follows:

F move forward a step of length s. A line segment between the old and the new position of the turtle is drawn.

A place subfigure $A \in \mathcal{A}$. To this end, A is translated and rotated so that its entry point P_A and direction $\vec{p_A}$ are aligned with the current position and orientation of the turtle. Having placed A, the turtle assumes the resulting exit position Q_A and direction $\vec{q_A}$.

\+ turn left by angle δ.

\- turn right by angle δ.

Unless otherwise indicated, the angle increment δ is equal to 90°.

Example 1. Assuming that the contact points and directions of frames L_m and R_m are as given at the top of Figure 4, the connecting patterns shown at the bottom of that figure are represented by the following formulas:

$$L_{m+1} = L_m F + R_m F R_m + F L_m - F - L_m F L_m F L_m - F R_m F R_m$$

$$R_{m+1} = R_m F - L_m F L_m - F R_m + F + R_m F R_m F R_m + F L_m F L_m$$

Example 2. Assuming that the contact points and directions of frames L_m and R_m are as in the previous example, the recursive pattern of the Hilbert curve (Figure 1) is captured by the following formulas:

$$L_{m+1} = +R_m F - L_m F L_m - F R_m +$$

$$R_{m+1} = -L_m F + R_m F R_m + F L_m -$$

In accordance with Figure 1.a, the initial polygons L_0 and R_0 are:

$$L_0 = +F - F - F +$$

$$R_0 = -F + F + F -$$

The above set of four equations provides a complete specification of the Hilbert curve at any recursion level m. The effects of modifying the initial polygon while maintaining the connecting pattern are shown in Figure 7.

The connecting patterns of Examples 1 and 2 are captured by finite sets of recursive formulas. Consequently, the same patterns are employed to connect components at all recursion levels. In this sense, the components are similar to the entire curve; the curve is *self-similar*. Since the curves under consideration are finite, we deal with self-similarity limited to a finite range of recursion levels, starting with the initial polygons and ending with the entire curve.

6 RECURSIVE FORMULAS AND L-SYSTEMS

In this section we show that strings describing FASS curves at any recursion level can be conveniently computed from the recursive formulas using the formalism of L-systems [11].

Consider the generation of a string that describes a FASS curve at level m. One way of carrying out the computation would be to start from L_m and recursively generate

the successive strings in order of decreasing values of index m. For example, refer to Example 2. The computation of L_2 would proceed as follows:

$$\begin{aligned} L_2 &= +R_1F - L_1FL_1 - FR_1 + \\ &= +(-L_0F + R_0FR_0 + FL_0-)F - (+R_0F - L_0FL_0 - FR_0+) \\ &\quad F(+R_0F - L_0FL_0 - FR_0+) - F(-L_0F + R_0FR_0 + FL_0-) + \\ &= +(-(+F - F - F+)F + (-F + F + F-)F \\ &\quad (-F + F + F-) + F(+F - F - F+)-)F - \ldots \end{aligned}$$

This example shows that the generation of the string L_m (or R_m) can be considered as a string-rewriting mechanism which substitutes L_i and R_i by expressions specified on the right side of the corresponding pattern definition formulas. The substitution proceeds in a parallel way, which means that all symbols in a string are replaced simultaneously (formally, $F, +$ and $-$ can be viewed as replacing themselves). Since all indices in any given string have the same value, they can be dropped, provided that a global count of derivation steps is kept and the proper set of rules is applied during each step. Such a rewriting process corresponds with the definition of a context-free table L-system or TOL-system [9, 20]. The rewriting rules, referred to as *productions*, are grouped into sets called *tables*. All productions used in a particular derivation step belong to the same table. A complete TOL-system also specifies the initial symbol or string of symbols called the *axiom*, and a *control function* which determines which table should be used at each derivation step.

The table mechanism provides a means of switching from productions describing the recursive connection pattern (by convention, table 1) to productions describing the initial polygons (table 2). In the applications under consideration the control function selects table 1 for all derivation step except the last one, when table 2 is used.

Example 3. The TOL-system corresponding to the recursive equations of Example 2 is given below:

```
Axiom:  L
Table 1:                Table 2:
L → +RF-LFL-FR+         L → +F-F-F+
R → -LF+RFR+FL-         R → -F+F+F-
```

An important special case of *pure* curves occurs when the initial polygons are reduced to single points, making the entry point and the exit point of the frames coincide (Figure 8.a). Technically, one can assume that symbols L and R are erased (replaced by the empty string) in the last derivation step. Alternatively, these symbols can be left in the string and ignored by the turtle during string interpretation. This second approach is consistent with previous papers describing turtle interpretation of L-systems [17, 18]. Since the symbols L and R do not need to be removed, table 2 becomes unnecessary, and all productions belong to the same set. Consequently, pure space-filling curves can be generated by DOL-systems (deterministic, context-free L-systems without tables [11]).

Example 4. The following DOL-system generates pure curves with the connection pattern described in Example 1:

```
Axiom:  L (or R)
Productions:
L → LF+RFR+FL-F-LFLFL-FRFR
R → RF-LFL-FR+F+RFRFR+FLFL
```

The relationship between recursive equations and L-systems has been presented above in the context of FASS curves. A general theory of recursive systems in connection with L-systems is given in [9].

7 ALGORITHMIC CONSTRUCTION OF FASS CURVES

The key problem in the construction of FASS curves is that of finding suitable connecting patterns. This problem can be viewed as a *marked tile puzzle.* The marks represent the positions of the contact points. An array of $n \times n$ square tiles is formed by putting the tiles into a sequence, so that the entry point of the next tile is connected to the exit point of the previous tile by an admissible segment. The entry and the exit points of the whole sequence must be near the corners of the macrotile (in the para or ortho position), allowing for the use of the macrotile as a building block in the next step of the recursive construction of the curve. Once a sequence of tiles satisfying these conditions has been found, the specification of the connecting pattern is completed and the path of a turtle traversing the tiles is examined. The connecting segments are inserted, and changes in turtle orientation are recorded. In order to make the turtle enter and exit the macrotile at directions consistent with the entry and exit directions of the corresponding tile, additional changes in turtle direction at the beginning and at the end of the path may be necessary. These changes are manifested as symbols + or − at the beginning or at the end of the string representing the connecting pattern.

The problem of solving the marked tile puzzle can be approached "by hand" or using a computer program. A program which generates all "left" connecting patterns made of para-tiles in a grid of dimensions $n \times n$ is described below (the number n will be further referred to as the *macrotile subdivision factor*).

The program attempts to construct a path of n^2 connected para-tiles, starting at the bottom left corner and ending in the bottom right corner of a square macrotile. A two-dimensional array of $n \times n$ locations indicates positions occupied by the tiles at a given stage of path construction. A one-dimensional array lists the consecutive tiles that form this path. Associated with each tile is the information about the tile's location within the macrotile, tile orientation (left or right), the coordinates of the exit contact point and direction, and the type of connection to the next tile. The possible connecting configurations for pairs of para-tiles are shown in Figure 14. The algorithm operates iteratively. In the first step, the initial tile is placed in the bottom left corner of the macrotile. In every subsequent step, an attempt is made to extend the path constructed so far by appending a new tile. The added tile has to form one of the admissible connecting configurations with the preceding tile in the path, and must not be placed in an already occupied location within the macrotile. If a suitable tile is found, the path is extended and the next iteration step is performed. Otherwise, the existing path cannot be extended and the algorithm backtracks by removing the last tile. Thus, the algorithm proceeds by adding or removing a tile in every iteration step.

To avoid an infinite loop which would result from removing a tile and placing the same tile over again, the connecting configurations are ordered. The new tile must form a configuration with a larger ordering number than the configuration it replaces.

The program produces output when a path consisting of all n^2 tiles is formed and the exit point of the last tile is placed in the bottom right corner of the macrotile. The constructed sequence of tiles is then scanned and the corresponding string, representing

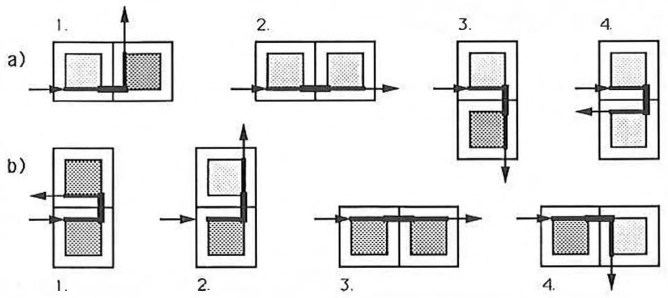

Figure 14: All possible relative configurations of two consecutive para-tiles. (a) The first tile is a left tile. (b) The first tile is a right tile.

the connecting pattern using turtle notation, is generated. Subsequently, the last tile is removed and the search for the next connecting pattern is resumed. The algorithm terminates when the backtracking procedure removes the first tile in the path and no alternative tile can be placed at this position.

In summary, the algorithm proceeds by searching a tree of possible paths using a depth-first strategy. The explicit tree construction is avoided by the lexicographic ordering of paths. This limits the space required by the algorithm to $O(n^2)$.

The connecting patterns produced using the above algorithm for macrotile subdivision factor ranging from 2 to 5 are collected in Table 1. These patterns describe left macrotiles. If two patterns differ only in the order in which their components are scanned (as in the top row of Figure 16), only one representative pattern is given. The other pattern and the symmetric patterns describing right macrotiles can be found using the following Lemmas.

Lemma 2. Let x^I denote the inverse of string x, that is the string resulting from writing the symbols of x from the last to the first. For any connecting pattern expressed by a string x over the alphabet $\mathcal{V}$, the string $y = x^I$ represents a symmetric pattern. If

```
Grid subdivision = 2
2.1 +RF-LFL-FR+

Grid subdivision = 3
3.1 LF+RFR+FL-F-LFLFL-FRFR+

Grid subdivision = 4
4.1  LF+RFR+FL-FRF-LFL-FR+F+RF-LFL-FRF-LF+RFR+FL (*)
4.2  LFLF+RFRFR+F+RF-LFL-FRF-LFLFLFL-FRFRFR+
4.3  LFLF+RFR+FLFL-FRF-LFL-FR+F+RF-LFL-FRFRFR+
4.4  +RF-LFL-FR+F+RFRFR+FLFL-F-LFLFLFL-FRFRFR+
4.5  +RFRFRF-LFL-FRFRFR+F+RFRFRF-LFL-FRFRFR+ (*)
```

Table 1: Connecting patterns suitable for the construction of FASS curves using para-tiles. Asterisks indicate palindromes.

```
Grid subdivision = 5
5.1  LF+RFR+FL-FRFRF-LFLFLFLFL-FRF-LFL-FR+FLF+RFR+FL-FRFR+F+RF-LFL-FR+
5.2  LF+RFR+FL-FRFRF-LFLFLFLFL-FRFRF-LFL-FRFR+FLF+RFR+FL-FRFR+FLFL
5.3  LF+RFR+FL-FRFRF-LFLFLFLFL-F-LFLFLF+RFR+FLFLFL-F-LFLF+RFR+FLFL
5.4  LF+RFR+FL-FRFRF-LFLFLFLFL-F-LFLFLF+RFR+FL-FRFR+F+RFRF-LFL-FRFR+
5.5  LF+RFR+FL-FRFRF-LFLFL-F-LF+RFR+FLF+RFRF-LFL-FRFRF-LFLF+RFR+FLFL
5.6  LF+RFR+FL-FRFRF-LFLFL-F-LF+RFR+FL-FRFR+F+RFRFRFRF-LFL-FRFRFRFR+
5.7  LF+RFR+FL-FRFRF-LFL-FRFR+FLF+RFR+FL-F-LFLFL-FRFRF-LFLF+RFR+FLFL
5.8  LF+RFR+FL-FRFRF-LFL-FRFR+FL-FRFR+F+RFRFRFR+FL-F-LFLFL-FRFRFRFR+
5.9  LF+RFR+FL-FRFRF-LFL-FRFR+F+RFRF-LFLFL-F-LF+RFR+FL-F-LFLF+RFR+FLFL
5.10 LF+RFR+FL-F-LF+RFR+FL-F-LFLFLFLFL-FRF-LFL-FR+F+RFRFRFR+F+RF-LFL-FR+
5.11 LF+RFR+FL-F-LF+RFR+FL-F-LFLFLFLFL-FRFRF-LFL-FRFR+F+RFRFRFR+FLFL
5.12 LF+RFR+FL-F-LF+RFR+FL-F-LFLFLFLFL-F-LFLF+RFRFRFR+F+RFRF-LFL-FRFR+
5.13 LF+RFR+FL-F-LF+RFR+FL-F-LFLFLFLFL-F-LFLF+RFR+FLFL-F-LFLF+RFR+FLFL
5.14 LF+RFR+FL-F-LF+RFR+FL-F-LFLFL-FRFRFRFR+F+RFRFRFRF-LFL-FRFRFRFR+
5.15 LF+RFR+FL-F-LF+RFR+FL-F-LFLFL-FRFR+F+RFRF-LFL-FRFRF-LFLF+RFR+FLFL
5.16 LF+RFR+FL-F-LF+RFR+FL-F-LFLFL-FR+F+RF-LFL-FRF-LFLF+RFRFR+F+RF-LFL-FR+
5.17 LF+RFR+FL-F-LF+RFR+FL-F-LFLFL-FR+F+RF-LFL-FRFRF-LFL-FR+F+RFRFR+FLFL
5.18 LF+RFR+FL-F-LFLF+RFR+FLFL-F-LFLFLFLFL-F-LF+RFR+FL-F-LFLF+RFR+FLFL
5.19 LF+RFR+FL-F-LFLFLF+RFR+FLFLFL-F-LFLFLFLFL-FRFRF-LFLF+RFR+FLFL
5.20 LF+RFR+FL-F-LFLFL-FRFR+F+RFRFRFR+FLFLFL-F-LFLFLFLFL-FRFRFRFR+
5.21 LFLFLF+RFRFRFR+FLF+RFR+FL-F-LFLFL-FRFRF-LFLFLFLFL-FRFRFRFR+
5.22 LFLFLF+RFRFRFR+F+RFRF-LFLFL-F-LF+RFR+FL-F-LFLFLFLFL-FRFRFRFR+
5.23 LFLFLF+RFRFR+F+RF-LFLFL-F-LF+RFR+FL-F-LFLFL-FR+F+RF-LFL-FRFRFRFR+
5.24 LFLFLF+RFR+FLFLFL-FRFRF-LFLFL-F-LF+RFR+FLF+RFRF-LFL-FRFRFRFR+
5.25 LFLFLF+RFR+FLFLFL-FRFRF-LFL-FRFR+FLF+RFR+FL-F-LFLFL-FRFRFRFR+
5.26 LFLFLF+RFR+FLFLFL-F-LF+RFR+FL-F-LFLFL-FRFR+F+RFRF-LFL-FRFRFRFR+
5.27 LFLFLF+RFR+FLFLFL-F-LFLFLF+RFR+FLFLFL-F-LFLFLFLFL-FRFRFRFR+
5.28 LFLFLF+RFR+FL-F-LF+RFR+FLF+RFRF-LFL-FRFRF-LFLFLFLFL-FRFRFRFR+
5.29 +RF-LFL-FR+FLF+RFR+FL-FRFR+F+RF-LFL-FRF-LFLFLFLFL-F-LF+RFR+FL-FRFR+
5.30 +RF-LFL-FR+FLF+RFR+FL-FR+FLFL-FRF-LFL-FR+F+RF-LFLFL-F-LF+RFR+FL-FRFR+
5.31 +RF-LFL-FR+FLF+RFR+FL-F-LF+RFR+FLF+RF-LFL-FRF-LFLFLFLFL-FRFRFRFR+
5.32 +RFRF-LF+RFR+FL-F-LFLFLFLFL-FRFRF-LFL-FRFR+F+RFRF-LF+RFR+FLFLFL
5.33 +RFRF-LF+RFR+FL-F-LFLFLFLFL-F-LFLF+RFRF-LF+RFR+FLF+RFRF-LFL-FRFR+
5.34 +RFRF-LF+RFR+FL-F-LFLFLFLFL-F-LFLF+RFR+FLFL-F-LFLFLF+RFR+FLFLFL
5.35 +RFRF-LF+RFR+FL-F-LFLFL-FRFRF-LF+RFR+FLF+RFRFRFRF-LFL-FRFRFRFR+
5.36 +RFRF-LF+RFR+FL-F-LFLFL-FRFR+F+RFRF-LFL-FRFRF-LFLFLF+RFR+FLFLFL
5.37 +RFRF-LF+RFR+FL-F-LFLFL-FR+F+RF-LFL-FRFRF-LFL-FR+F+RF-LF+RFR+FLFLFL
5.38 +RFRF-LFLF+RFR+FLFL-F-LFLFLFLFL-F-LF+RFR+FL-F-LFLFLF+RFR+FLFLFL
5.39 +RFRF-LFLFLF+RFR+FLFLFL-F-LFLFLFLFL-FRFRF-LFLFLF+RFR+FLFLFL
5.40 +RFRF-LFLFL-F-LF+RFR+FLF+RFRFRFR+FLFLFL-F-LFLFLFLFL-FRFRFRFR+
5.41 +RFRF-LFL-FRFR+FLF+RFR+FL-F-LF+RFR+FLFLFL-F-LFLFLFLFL-FRFRFRFR+
5.42 +RFRFRFRF-LFLFL-F-LF+RFRFRFR+FLF+RFR+FL-F-LF+RFRF-LFL-FRFRFRFR+
5.43 +RFRFRFRF-LFLFL-F-LF+RFR+FL-F-LF+RFR+FLF+RFRFRFRF-LFL-FRFRFRFR+

Grid subdivision = 6
There are 897 different patterns, 24 of which are palindromic.
```

Table 1 (continued): Connecting patterns suitable for the construction of FASS curves using para-tiles.

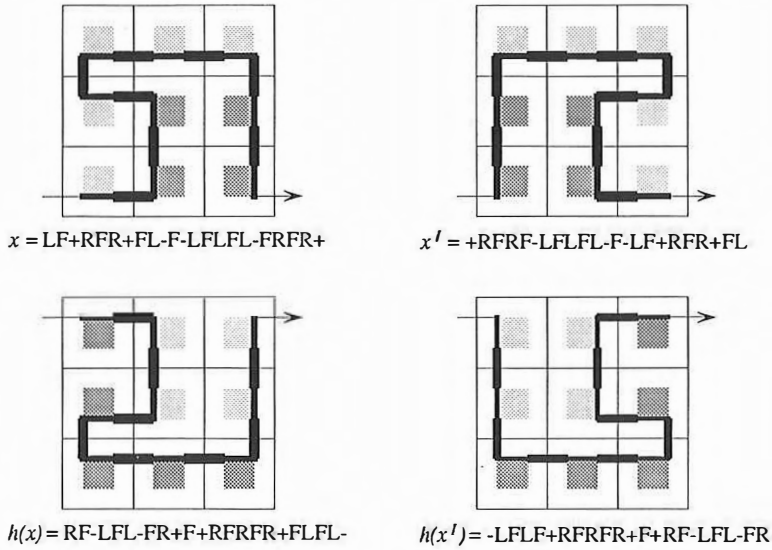

Figure 15: Symmetric connecting patterns.

the pattern x represents a tiling, the position of contact points and direction vectors *with respect to the original macrotile* is preserved. Thus, the inversion operation preserves the orientation of tiles.

Lemma 3. Consider the alphabet $\mathcal{V} = \{F, +, -, L, R\}$, where symbols L and R denote an arbitrary pair of symmetric polygons, and let $h : \mathcal{V} \rightarrow \mathcal{V}$ be a homomorphism such that $h(F) = F$, $h(+) = -$, $h(-) = +$, $h(L) = R$ and $h(R) = L$. For any connecting pattern expressed by a string x over the alphabet $\mathcal{V}$, the string $y = h(x)$ represents a symmetric pattern. The reflection operation maps left tiles into symmetric right tiles and vice-versa.

Both Lemmas are illustrated in Figure 15. Simple proofs (by induction on the string length) are omitted. Note that the pattern obtained by inverting the string x may be identical to the original pattern. This situation occurs when the word x is a palindrome.

A pure FASS curve can be generated using any pair of patterns that correspond to the same macrotile division factor and have opposite orientations (left – right). For example, the curve shown in Figure 16.b uses patterns $x_{4.1}$ and $h(x_{4.2})$, where $x_{4.1}$ and $x_{4.2}$ stand for the patterns numbered 4.1 and 4.2 in Table 1. The corresponding L-system is given below:

```
L → LF+RFR+FL-FRF-LFL-FR+F+RF-LFL-FRF-LF+RFR+FL
R → RFRF-LFLFL-F-LF+RFR+FLF+RFRFRFR+FLFLFL-
```

It is worth noticing that, as long as the macrotile division factor is constant, a variety of connecting patterns can be mixed within a single FASS curve. For example, Figure 17 presents an example of a curve that combines *all* connecting patterns of para-tiles with grid subdivision $n = 4$, listed in Table 1. Technically, this curve was generated using a stochastic L-system [5, 27] with sixteen productions. Eight productions with the predecessor L had successors $x_{4.1}$, $x_{4.2}$, $x^I_{4.2}$, $x_{4.3}$, $x^I_{4.3}$, $x_{4.4}$, $x^I_{4.4}$ and $x_{4.5}$. Eight productions

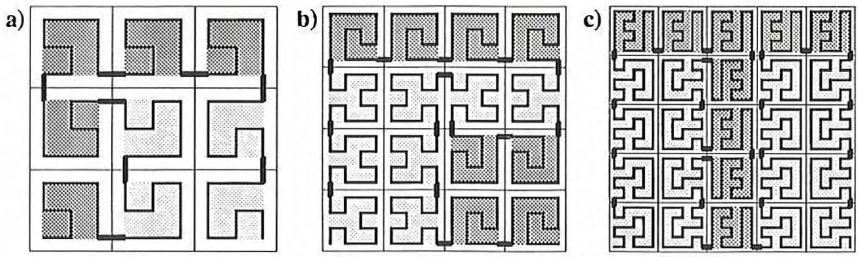

Figure 16: Examples of algorithmically-generated FASS curves using grid subdivisions: (a) $n = 3$, (b) $n = 4$, and (c) $n = 5$.

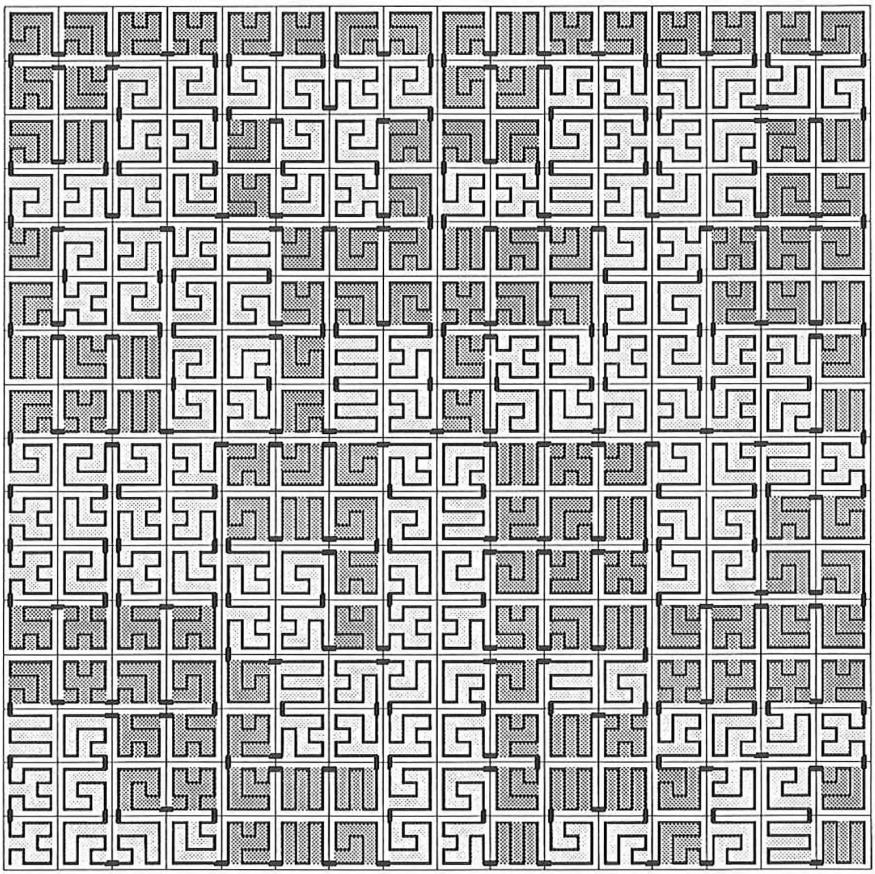

Figure 17: A FASS curve generated using a stochastic L-system, using para-tiles with grid subdivision $n = 4$. Note the irregular distribution of left and right tiles, indicated by the shading of frames.

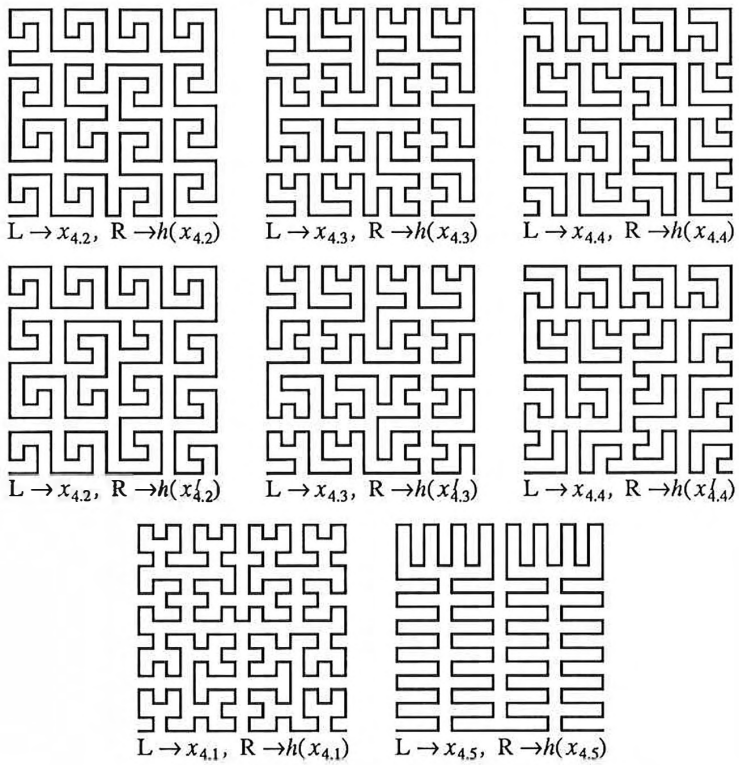

Figure 18: All uniform FASS curves constructed using para-tiles in a square grid with subdivision factor $n = 4$. Productions are specified in reference to Table 1, for example $R \rightarrow h(x^I_{4.3})$ denotes a production with the predecessor R and successor obtained as the homomorphic image of the inverted string 4.3 in Table 1.

with the predecessor R were obtained by replacing the successors of the "left" productions using the homomorphism h. During the derivation, the productions were selected at random, with equal probabilities, for each letter L or R in the string.

Among pure FASS curves one can distinguish those with particularly *uniform* designs. This uniformity occurs when the right connecting pattern is the reflection of the left pattern, or the reflection of the inversion of the left pattern (as in the bottom row of Figure 15). For example, Figure 18 shows all uniform FASS curves constructed in a square grid with subdivision factor $n = 4$. The set consists of three pairs of patterns and two patterns without a match. The curves which form a pair differ by the presence or absence of the inversion in the definition of the right connecting pattern. The remaining two curves correspond to palindromic connecting patterns. Figure 19 shows selected uniform FASS curves constructed in a square grid with subdivision factors equal to 5 and 6.

So far, the discussion concentrated on the curves constructed from para-tiles. Obvi-

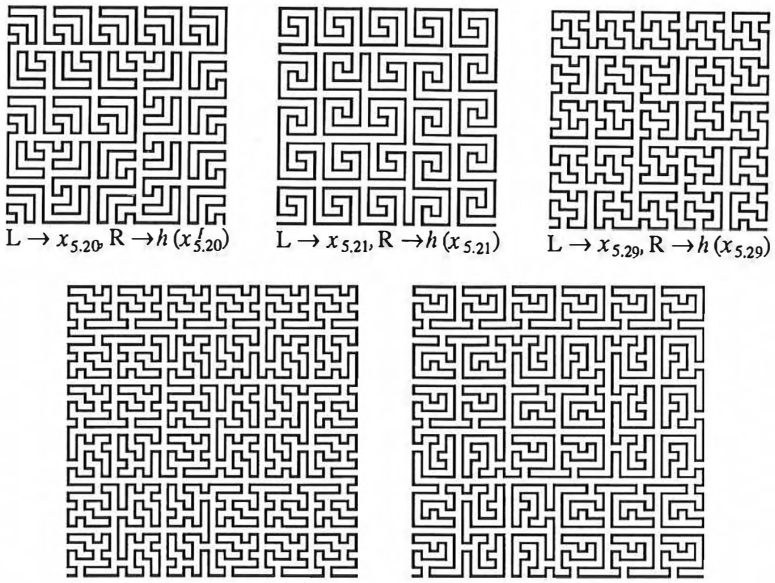

Figure 19: Examples of uniform FASS curves constructed using para-tiles for grid subdivision factors $n = 5$ and $n = 6$.

ously, a similar method can be applied to find connecting patterns using ortho-tiles, and generate the corresponding curves. The connecting patterns for $n \leq 6$ are collected in Table 2. Selected curves are shown in Figure 20.

8 CONCLUSIONS

In this paper we have investigated a relationship between self-avoiding, space-filling curves and tilings of the plane. Particular attention was given to curves which are single-stroke and self-similar. We referred to this class as FASS curves. The notion of a connecting pattern was introduced to characterize the recursive design of the curves. An algorithmic method was given for constructing all connecting patterns suitable for the

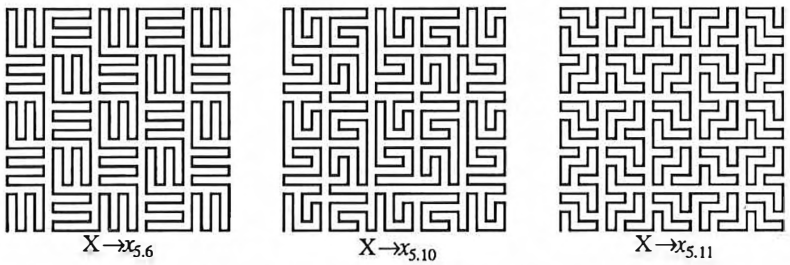

Figure 20: Examples of uniform FASS curves constructed using ortho-tiles.

```
Grid subdivision = 3
3.1  X+F+X-F-X-F-X-F-X+F+X+F+X+F+X-F-X

Grid subdivision = 5
5.1  X+F+X-F-X+F+X-F-X-F-X+F+X-F-X-F-X+F+X-F-X-F-X-F-X+F+
           X+F+X+F+X-F-X+F+X+F+X-F-X+F+X+F+X-F-X+F+X-F-X
5.2  X+F+X-F-X+F+X-F-X-F-X+F+X-F-X-F-X-F-X+F+X+F+X+F+X-F-
           X-F-X-F-X+F+X+F+X+F+X-F-X+F+X+F+X-F-X+F+X-F-X
5.3  X+F+X-F-X+F+X-F-X-F-X+F+X-F-X-F-X-F-X+F+X+F+X-F-X+F+
           X+F+X+F+X-F-X-F-X-F-X+F+X+F+X+F+X-F-X+F+X-F-X
5.4  X+F+X-F-X+F+X-F-X-F-X+F+X-F-X-F-X-F-X+F+X+F+X-F-X+F+
           X+F+X-F-X+F+X+F+X+F+X-F-X-F-X-F-X+F+X+F+X-F-X
5.5  X+F+X-F-X+F+X-F-X-F-X-F-X+F+X+F+X-F-X-F-X-F-X+F+X+F+
           X+F+X-F-X+F+X+F+X-F-X+F+X+F+X-F-X-F-X-F-X+F+X
5.6  X+F+X-F-X+F+X-F-X-F-X-F-X+F+X-F-X+F+X+F+X+F+X-F-X+F+
           X-F-X-F-X-F-X+F+X-F-X+F+X+F+X+F+X-F-X+F+X-F-X
5.7  X+F+X-F-X+F+X-F-X-F-X-F-X+F+X-F-X+F+X+F+X+F+X-F-X-F-
           X-F-X+F+X+F+X+F+X-F-X+F+X+F+X-F-X-F-X-F-X+F+X
5.8  X+F+X-F-X+F+X-F-X-F-X-F-X+F+X-F-X+F+X+F+X-F-X+F+X+F+
           X+F+X-F-X-F-X+F+X-F-X-F-X-F-X+F+X+F+X+F+X-F-X
5.9  X+F+X-F-X+F+X-F-X-F-X-F-X+F+X-F-X+F+X+F+X-F-X+F+X+F+
           X+F+X-F-X-F-X-F-X+F+X+F+X+F+X-F-X-F-X-F-X+F+X
5.10 X+F+X-F-X+F+X-F-X-F-X-F-X+F+X-F-X+F+X+F+X-F-X+F+X+F+
           X-F-X+F+X+F+X+F+X-F-X-F-X-F-X+F+X-F-X-F-X+F+X
5.11 X+F+X-F-X-F-X-F-X+F+X+F+X-F-X+F+X+F+X+F+X-F-X-F-X+F+
           X+F+X-F-X-F-X-F-X+F+X-F-X-F-X+F+X+F+X+F+X-F-X

NOTE: There are no patterns at even grid subdivisions.
      Angle increment is 45 degrees.
```

Table 2: Connecting patterns suitable for the construction of FASS curves using orthotiles.

generation of FASS curves in a square grid recursively subdivided into $n \times n$ subgrids. The results were given in the form of L-system. The patterns obtained for $n \leq 5$ were listed and illustrated using selected curves.

We perceive the results presented in this paper as an introduction to the area "fractals and tilings". A wealth of problems remain open. Here we list some of the obvious ones.

- The relationship between fractals and tilings is not limited to square tilings. Other tile shapes can be used as well. For example, the design of the Sierpiński curve can be explained in terms of a triangular tiling (Figure 21.a), and the design of the Gosper curve can be explained using hexagonal tiles (Figure 21.b). Programs for generating connecting patterns using these tiles can be devised and applied to explore the respective families of FASS curves. Whether tilings combining several tile types in a single design could also be used as a basis for FASS curve construction remains an open problem.

- The results described in this paper were obtained under the assumption that the

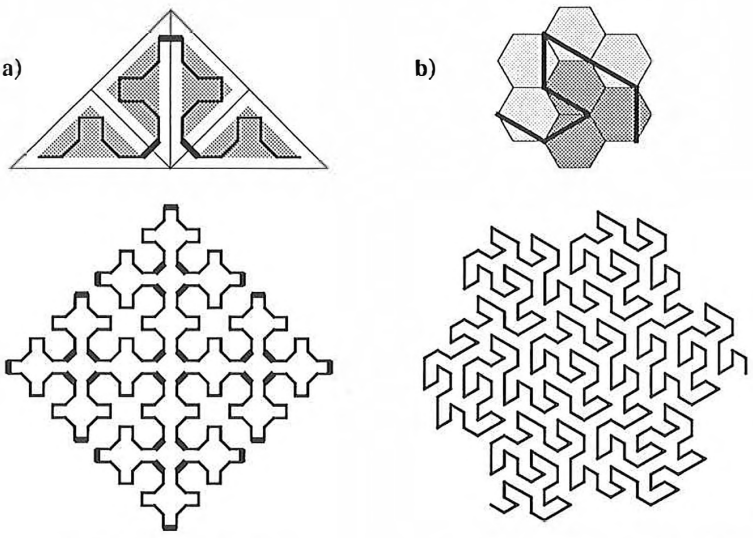

Figure 21: FASS curves defined over non-square grids. (a) The Sierpiński curve (angle increment $\delta = 45°$). (b) The Gosper curve ($\delta = 60°$).

frames are entirely included in the tiles. However, this restriction can be relaxed, yielding a separate family of intricate curves. For example, some of the South Indian folk art designs known as *kolam patterns* [19, 24, 25] can be explained in terms of tilings with frames (and the related polygons) "sticking out" of the tiles (Figure 22).

- A three-dimensional extension of the results described in this paper can be applied to synthesize curves which fill three-dimensional space. In this case, planar tilings are replaced by three-dimensional crystallographic patterns. If the construction uses a square grid, the connecting pattern joins $n \times n \times n$ cubes in a macrocube. The concept of a marked tile puzzle can be then extended to a "marked cube puzzle". For example, Plates 1 and 2 show a three-dimensional extension of the Hilbert curve [10], which has been constructed by solving the marked cubes puzzle by hand (using a set of appropriately marked wooden toy blocks). The L-system is given below:

```
Axiom:  A
A → B-F+CFC+F-D&F^D-F+&&CFC+F+B//
B → A&F^CFB^F^D^^-F-D^|F^B|FC^F^A//
C → |D^|F^B-F+C^F^A&&FA&F^C+F+B^F^D//
D → |CFB-F+B|FA&F^A&&FB-F+B|FC//
```

The symbols ^ and & pitch the turtle up and down, / and \ roll it around its own axis, and | turns it around by 180° [18].

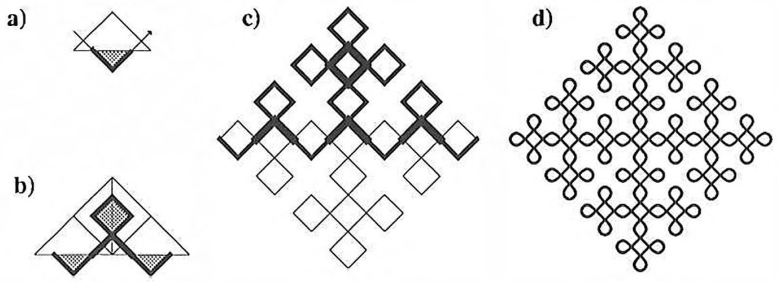

Figure 22: (a) The tile used in the design of the *Anklets of Krishna* kolam, (b) four connected tiles, (c) the entire pattern (consisting of two triangular parts) after three derivation steps, (d) the pattern with rounded corners after four derivation steps.

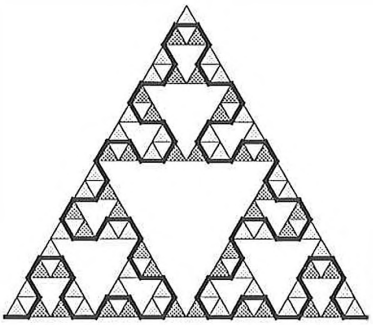

Figure 23: The Sierpiński gasket.

- The relationship between tilings and space-filling curves can be extended to polygonal fractals which do not fill the plane. In this case, the tiles do not cover the entire macrotile. An example (the Sierpiński gasket [23]) is given in Figure 23.

- The curves considered in this paper consisted of straight line segments. However, FASS curves may also be smooth. As a matter of fact, any polygon can be transformed into a smooth curve by rounding off the corners, for example using splines [2, 18]. Thus, the *snake* kolam (Figure 24) is a smooth version of the Sierpiński space-filling curve (Figure 21.a). Intuitively, the snake kolam is a self-avoiding curve. However, the definition of self-avoidance used throughout this paper does not apply, since the curve does not have any inherent vertices which would divide it into segments. How to generalize the notion of self-avoidance to smooth curves is an open problem.

- In the scope of this paper, FASS curves have been explored by an exhaustive search

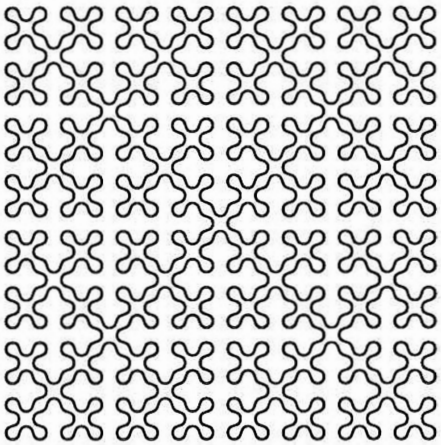

Figure 24: An example of a smooth space-filling curve (the *snake* kolam).

of the space of connecting patterns. This technique is practical only for small values of the macrotile subdivision factor n. For example, the program described in this paper needed 15 minutes of CPU time on a MIPS R2000 processor to generate all 897 connecting patterns made of para-tiles on a grid with subdivision factor $n = 6$. For $n = 7$, the program did not complete its operation after 50 hours of CPU time. Thus, many questions remain open. What is the asymptotic time complexity of the algorithm used to solve the marked tile puzzle? What is the complexity of the problem itself? Can a formula be found to determine or estimate the total number of pure FASS curves for a given grid subdivision factor n *without* constructing these curves?

- There is a close relationship between some space-filling curves and number systems [3, 16]. For example, the Peano curve was originally described by referring to a ternary number system, rather than in geometric terms [16]. Can a similar relationship be established for *any* FASS curve?

ACKNOWLEDGEMENTS

William Gosper brought to our attention the paper by McKenna [14]. Norma Fuller helped us in constructing the 3D Hilbert curve. The ray-tracer used to generate the color plates was written by Craig Kolb. Richard Voss suggested many improvements to the first version of this paper and originated the idea of using stochastic L-systems to FASS curve generation. The research reported in this paper has been supported by an operating grant, equipment grants and a scholarship from the Natural Sciences and Engineering Research Council of Canada, and by an equipment donation from Apple Computer, Inc. Facilities of the Department of Computer Science, University of Regina, were also essential. All support is gratefully acknowledged.

REFERENCES

[1] H. Abelson and A. A. diSessa. *Turtle geometry.* M.I.T. Press, Cambridge, 1982.

[2] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An introduction to splines for use in computer graphics and geometric modeling.* Morgan Kaufmann Publ. Inc., Los Altos, California, 1987.

[3] A. J. Cole. A note on Peano polygons and Gray codes. *Intern. J. Computer Math.*, 18:3–13, 1985.

[4] F. M. Dekking. Recurrent sets. *Advances in Mathematics*, 44:78–104, 1982.

[5] P. Eichhorst and W. J. Savitch. Growth functions of stochastic Lindenmayer systems. *Information and Control*, 45:217–228, 1980.

[6] M. Gardner. An array of problems that can be solved with elementary mathematical techniques. *Scientific American*, 216, 1967. 3:124–129 (March), 4:116–123 (April).

[7] M. Gardner. Mathematical games – in which "monster" curves force redefinition of the word "curve". *Scientific American*, 235(6):124–134, December 1976.

[8] B. Grünbaum and G. C. Shephard. *Tilings and patterns.* W. H. Freeman and Company, New York, 1987.

[9] G. T. Herman and G. Rozenberg. *Developmental systems and languages.* North-Holland, Amsterdam, 1975.

[10] D. Hilbert. Ueber die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38:459–460, 1891.

[11] A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 18:280–315, 1968.

[12] B. B. Mandelbrot. *The fractal geometry of nature.* W. H. Freeman and Company, San Francisco, 1982.

[13] S. Mazurkiewicz. O punktach wielokrotnych krzywych wypelniajacych obszar plaski. *Prace Matematyczno-Fizyczne*, 26:116, 1915.

[14] D. M. McKenna. SquaRecurves, E-tours, Eddies, and Frenzies: basic families of Peano curves on the square grid. Draft paper – for inclusion in the Proceedings of the Eugene Strens Memorial Conference on Recreational Mathematics and its History, 1989.

[15] A. S. Parchomienko. *Co to jest linia.* Państwowe Wydawnictwo Naukowe, Warszawa, 1961.

[16] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36:157–160, 1890. Translated in G. Peano, *Selected works of Giuseppe Peano*, H. C. Kennedy, editor, pages 143–149, University of Toronto Press, Toronto, 1973.

[17] P. Prusinkiewicz. Graphical applications of L-systems. *Proceedings of Graphics Interface '86 – Vision Interface '86*, pages 247–253, 1986.

[18] P. Prusinkiewicz and J. S. Hanan. *Lindenmayer systems, fractals, and plants.* Springer-Verlag, New York, 1989. Lecture Notes in Biomathematics 79.

[19] P. Prusinkiewicz, K. Krithivasan, and M. G. Vijayanarayana. Application of L-systems to algorithmic generation of South Indian folk art patterns and karnatic music. In R. Narasimhan, editor, *A Perspective in Theoretical Computer Science – Commemorative Volume for Gift Siromoney*, pages 229–247. World Scientific Publishing Co., Singapore, 1989.

[20] G. Rozenberg. T0L systems and languages. *Information and Control*, 23:357–381, 1973.

[21] A. C. Shaw. A formal picture description scheme as a basis for a picture processing system. *Information and Control*, 14:9–52, 1969.

[22] W. Sierpiński. Sur une nouvelle courbe continue qui remplit toute une aire plane. *Bull. Intern. Acad. Sci. Cracovie,* Série A, pages 462–478, 1912. Reprinted in W. Sierpiński, *Oeuvres choisies*, S. Hartman et al., editors, pages 52–66, PWN – Éditions Scientifiques de Pologne, Warsaw, 1975.

[23] W. Sierpiński. Sur une courbe dont tout point est un point de ramification. *Comptes Rendus hebdomadaires des séances de l'Académie des Sciences*, 160:302–305, 1915. Reprinted in W. Sierpiński, *Oeuvres choisies*, S. Hartman et al., editors, pages 99–106, PWN – Éditions Scientifiques de Pologne, Warsaw, 1975.

[24] G. Siromoney and R. Siromoney. Rosenfeld's cycle grammars and kolam. In H. Ehrig, M. Nagl, A. Rosenfeld, and G. Rozenberg, editors, *Graph-grammars and their application to computer science*, pages 565–579. Springer-Verlag, 1987. Lecture Notes in Comp. Sci. 291.

[25] G. Siromoney, R. Siromoney, and K. Krithivasan. Array grammars and kolam. *Computer Graphics and Image Processing*, pages 63–82, 1974.

[26] A. L. Szilard and R. E. Quinton. An interpretation for D0L-systems by computer graphics. *The Science Terrapin*, 4:8–13, 1979.

[27] T. Yokomori. Stochastic characterizations of EOL languages. *Information and Control*, 45:26–33, 1980.