Chapter 1

Graphical modeling using L-systems

Lindenmayer systems — or L-systems for short — were conceived as a mathematical theory of plant development [82]. Originally, they did not include enough detail to allow for comprehensive modeling of higher plants. The emphasis was on plant topology, that is, the neighborhood relations between cells or larger plant modules. Their geometric aspects were beyond the scope of the theory. Subsequently, several geometric interpretations of L-systems were proposed with a view to turning them into a versatile tool for plant modeling. Throughout this book, an interpretation based on turtle geometry is used [109]. Basic notions related to L-system theory and their turtle interpretation are presented below.

1.1 Rewriting systems

The central concept of L-systems is that of rewriting. In general, rewriting is a technique for defining complex objects by successively replacing parts of a simple initial object using a set of *rewriting rules* or *productions*. The classic example of a graphical object defined in terms of rewriting rules is the *snowflake curve* (Figure 1.1), proposed in 1905 by von Koch [155]. Mandelbrot [95, page 39] restates this construction as follows:

Koch construction

> One begins with *two shapes*, an *initiator* and a *generator*. The latter is an oriented broken line made up of N equal sides of length r. Thus each stage of the construction begins with a broken line and consists in replacing each straight interval with a copy of the generator, reduced and displaced so as to have the same end points as those of the interval being replaced.

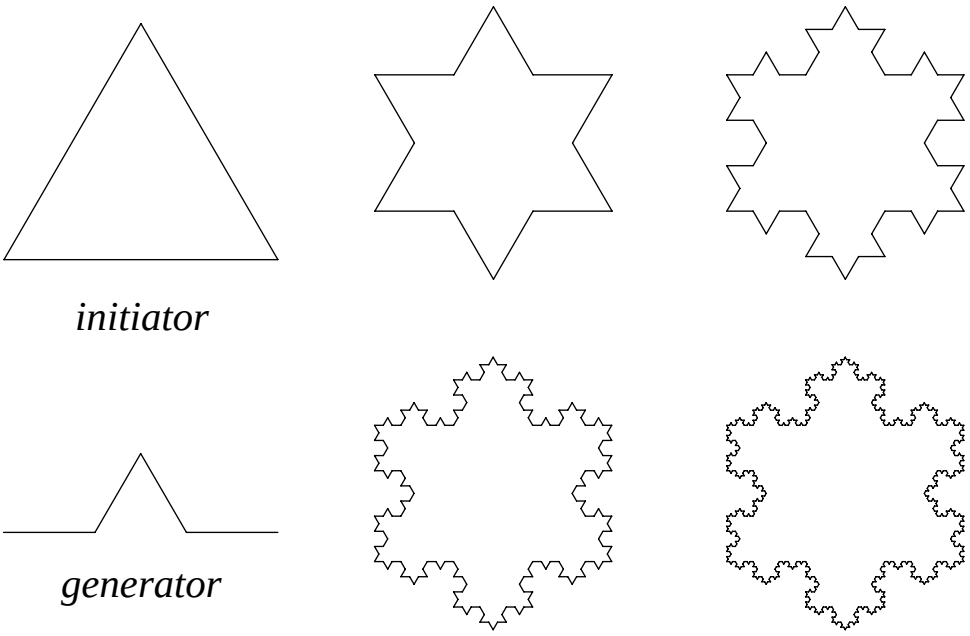


Figure 1.1: Construction of the snowflake curve

While the Koch construction recursively replaces open polygons, rewriting systems that operate on other objects have also been investigated. For example, Wolfram [160, 161] studied patterns generated by rewriting elements of rectangular arrays. A similar array-rewriting mechanism is the cornerstone of Conway's popular *game of life* [49, 50]. An important body of research has been devoted to various graph-rewriting systems [14, 33, 34].

Grammars

The most extensively studied and the best understood rewriting systems operate on character strings. The first formal definition of such a system was given at the beginning of this century by Thue [128], but a wide interest in string rewriting was spawned in the late 1950s by Chomsky's work on formal grammars [13]. He applied the concept of rewriting to describe the syntactic features of natural languages. A few years later Backus and Naur introduced a rewriting-based notation in order to provide a formal definition of the programming language ALGOL-60 [5, 103]. The equivalence of the Backus-Naur form (BNF) and the context-free class of Chomsky grammars was soon recognized [52], and a period of fascination with syntax, grammars and their application to computer science began. At the center of attention were sets of strings — called formal languages — and the methods for generating, recognizing and transforming them.

L-systems

In 1968 a biologist, Aristid Lindenmayer, introduced a new type of string-rewriting mechanism, subsequently termed L-systems [82]. The essential difference between Chomsky grammars and L-systems lies in

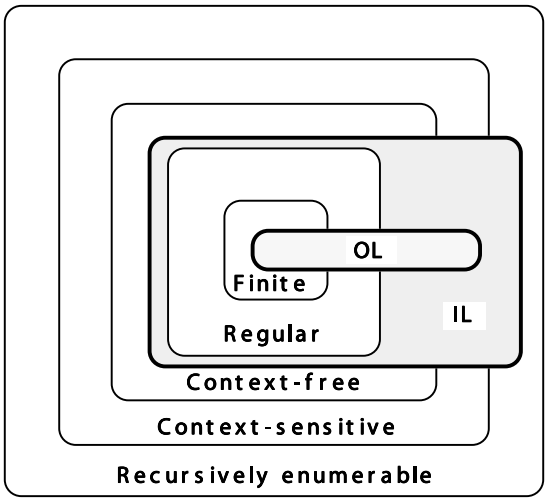


Figure 1.2: Relations between Chomsky classes of languages and language classes generated by L-systems. The symbols OL and IL denote language classes generated by context-free and context-sensitive L-systems, respectively.

the method of applying productions. In Chomsky grammars productions are applied sequentially, whereas in L-systems they are applied in parallel and simultaneously replace all letters in a given word. This difference reflects the biological motivation of L-systems. Productions are intended to capture cell divisions in multicellular organisms, where many divisions may occur at the same time. Parallel production application has an essential impact on the formal properties of rewriting systems. For example, there are languages which can be generated by context-free L-systems (called OL-systems) but not by context-free Chomsky grammars [62, 128] (Figure 1.2).

1.2 DOL-systems

This section presents the simplest class of L-systems, those which are deterministic and context-free, called DOL-systems. The discussion starts with an example that introduces the main idea in intuitive terms.

Example

Consider strings (words) built of two letters a and b, which may occur many times in a string. Each letter is associated with a rewriting rule. The rule $a \rightarrow ab$ means that the letter a is to be replaced by the string ab, and the rule $b \rightarrow a$ means that the letter b is to be replaced by a. The rewriting process starts from a distinguished string called the axiom. Assume that it consists of a single letter b. In the first derivation step (the first step of rewriting) the axiom b is replaced

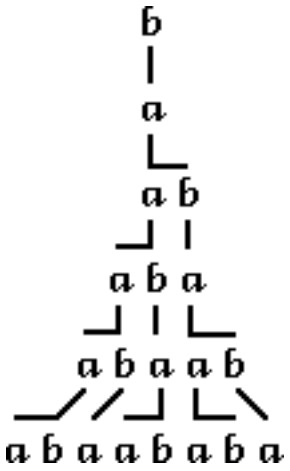


Figure 1.3: Example of a derivation in a DOL-system

by a using production $b \rightarrow a$. In the second step a is replaced by ab using production $a \rightarrow ab$. The word ab consists of two letters, both of which are *simultaneously* replaced in the next derivation step. Thus, a is replaced by ab, b is replaced by a, and the string aba results. In a similar way, the string aba yields $abaab$ which in turn yields $abaababa$, then $abaababaabaab$, and so on (Figure 1.3).

Formal definitions describing DOL-systems and their operation are given below. For more details see [62, 127].

L-system

Let V denote an alphabet, V^* the set of all words over V, and V^+ the set of all nonempty words over V. A *string OL-system* is an ordered triplet $G = \langle V, \omega, P \rangle$ where V is the *alphabet* of the system, $\omega \in V^+$ is a nonempty word called the *axiom* and $P \subset V \times V^*$ is a finite *set of productions*. A production $(a, \chi) \in P$ is written as $a \rightarrow \chi$. The letter a and the word χ are called the *predecessor* and the *successor* of this production, respectively. It is assumed that for any letter $a \in V$, there is at least one word $\chi \in V^*$ such that $a \rightarrow \chi$. If no production is explicitly specified for a given predecessor $a \in V$, the *identity production* $a \rightarrow a$ is assumed to belong to the set of productions P. An OL-system is *deterministic* (noted *DOL-system*) if and only if for each $a \in V$ there is exactly one $\chi \in V^*$ such that $a \rightarrow \chi$.

Derivation

Let $\mu = a_1 \dots a_m$ be an arbitrary word over V. The word $\nu = \chi_1 \dots \chi_m \in V^*$ is *directly derived* from (or *generated* by) μ, noted $\mu \Rightarrow \nu$, if and only if $a_i \rightarrow \chi_i$ for all $i = 1, \dots, m$. A word ν is generated by G in a derivation of *length* n if there exists a *developmental sequence* of words $\mu_0, \mu_1, \dots, \mu_n$ such that $\mu_0 = \omega$, $\mu_n = \nu$ and $\mu_0 \Rightarrow \mu_1 \Rightarrow \dots \Rightarrow \mu_n$.

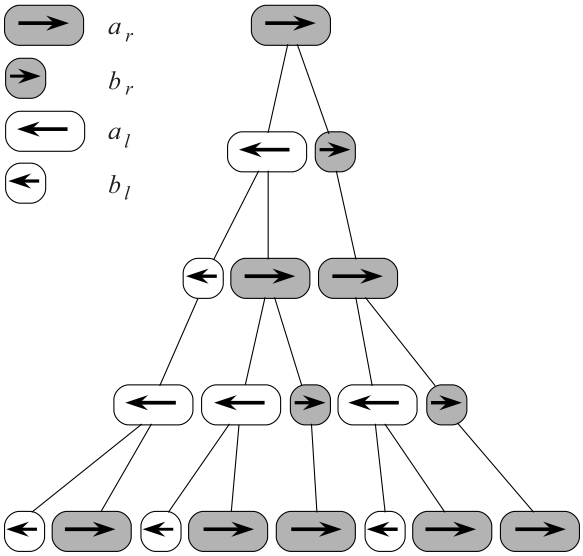


Figure 1.4: Development of a filament (*Anabaena catenula*) simulated using a DOL-system

Anabaena

The following example provides another illustration of the operation of DOL-systems. The formalism is used to simulate the development of a fragment of a multicellular filament such as that found in the blue-green bacteria *Anabaena catenula* and various algae [25, 84, 99]. The symbols a and b represent cytological states of the cells (their size and readiness to divide). The subscripts l and r indicate cell polarity, specifying the positions in which daughter cells of type a and b will be produced. The development is described by the following L-system:

$$\begin{array}{ll} \omega: & a_r \\ p_1: & a_r \rightarrow a_l b_r \\ p_2: & a_l \rightarrow b_l a_r \\ p_3: & b_r \rightarrow a_r \\ p_4: & b_l \rightarrow a_l \end{array} \tag{1.1}$$

Starting from a single cell a_r (the axiom), the following sequence of words is generated:

$$\begin{array}{l} a_r \\ a_l b_r \\ b_l a_r a_r \\ a_l a_l b_r a_l b_r \\ b_l a_r b_l a_r a_r b_l a_r a_r \\ \ldots \end{array}$$

Under a microscope, the filaments appear as a sequence of cylinders of various lengths, with a-type cells longer than b-type cells. The corresponding schematic image of filament development is shown in Figure 1.4. Note that due to the discrete nature of L-systems, the continuous growth of cells between subdivisions is not captured by this model.

1.3 Turtle interpretation of strings

The geometric interpretation of strings applied to generate schematic images of *Anabaena catenula* is a very simple one. Letters of the L-system alphabet are represented graphically as shorter or longer rectangles with rounded corners. The generated structures are one-dimensional chains of rectangles, reflecting the sequence of symbols in the corresponding strings.

Previous methods

In order to model higher plants, a more sophisticated graphical interpretation of L-systems is needed. The first results in this direction were published in 1974 by Frijters and Lindenmayer [46], and Hogeweg and Hesper [64]. In both cases, L-systems were used primarily to determine the branching topology of the modeled plants. The geometric aspects, such as the lengths of line segments and the angle values, were added in a post-processing phase. The results of Hogeweg and Hesper were subsequently extended by Smith [136, 137], who demonstrated the potential of L-systems for realistic image synthesis.

Szilard and Quinton [141] proposed a different approach to L-system interpretation in 1979. They concentrated on image representations with rigorously defined geometry, such as chain coding [43], and showed that strikingly simple DOL-systems could generate the intriguing, convoluted curves known today as *fractals* [95]. These results were subsequently extended in several directions. Siromoney and Subramanian [135] specified L-systems which generate classic space-filling curves. Dekking investigated the limit properties of curves generated by L-systems [32] and concentrated on the problem of determining the fractal (Hausdorff) dimension of the limit set [31]. Prusinkiewicz focused on an interpretation based on a LOGO-style turtle [1] and presented more examples of fractals and plant-like structures modeled using L-systems [109, 111]. Further applications of L-systems with turtle interpretation include realistic modeling of herbaceous plants [117], description of *kolam* patterns (an art form from Southern India) [112, 115, 133, 134], synthesis of musical scores [110] and automatic generation of space-filling curves [116].

Turtle

The basic idea of turtle interpretation is given below. A *state* of the *turtle* is defined as a triplet (x, y, α), where the Cartesian coordinates (x, y) represent the turtle's *position*, and the angle α, called the *heading*, is interpreted as the direction in which the turtle is facing. Given the *step size* d and the *angle increment* δ, the turtle can respond to

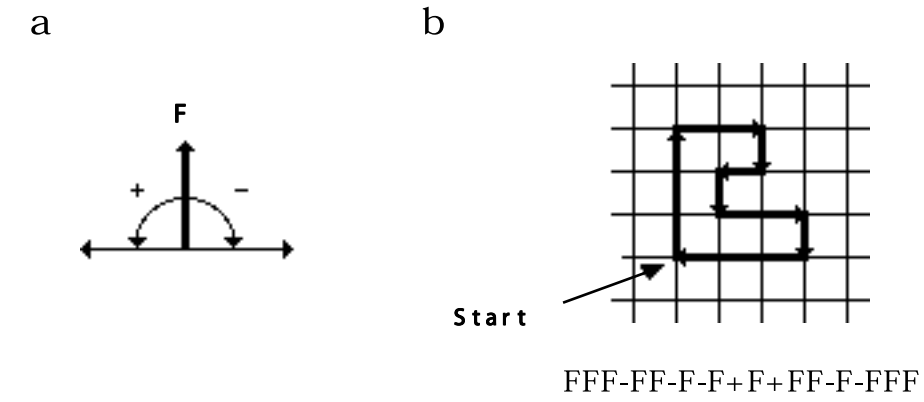


Figure 1.5: (a) Turtle interpretation of string symbols F, $+$, $-$. (b) Interpretation of a string. The angle increment δ is equal to 90°. Initially the turtle faces up.

commands represented by the following symbols (Figure 1.5a):

F	Move forward a step of length d. The state of the turtle changes to (x', y', α), where $x' = x + d\cos\alpha$ and $y' = y + d\sin\alpha$. A line segment between points (x, y) and (x', y') is drawn.
f	Move forward a step of length d without drawing a line.
$+$	Turn left by angle δ. The next state of the turtle is $(x, y, \alpha+\delta)$. The positive orientation of angles is counterclockwise.
$-$	Turn right by angle δ. The next state of the turtle is $(x, y, \alpha-\delta)$.

Interpretation

Given a string ν, the initial state of the turtle (x_0, y_0, α_0) and fixed parameters d and δ, the *turtle interpretation* of ν is the figure (set of lines) drawn by the turtle in response to the string ν (Figure 1.5b). Specifically, this method can be applied to interpret strings which are generated by L-systems. For example, Figure 1.6 presents four approximations of the *quadratic Koch island* taken from Mandelbrot's book [95, page 51]. These figures were obtained by interpreting strings generated by the following L-system:

$$\begin{aligned} \omega &: \ F - F - F - F \\ p &: \ F \to F - F + F + FF - F - F + F \end{aligned}$$

The images correspond to the strings obtained in derivations of length 0 to 3. The angle increment δ is equal to 90°. The step length d is decreased four times between subsequent images, making the distance

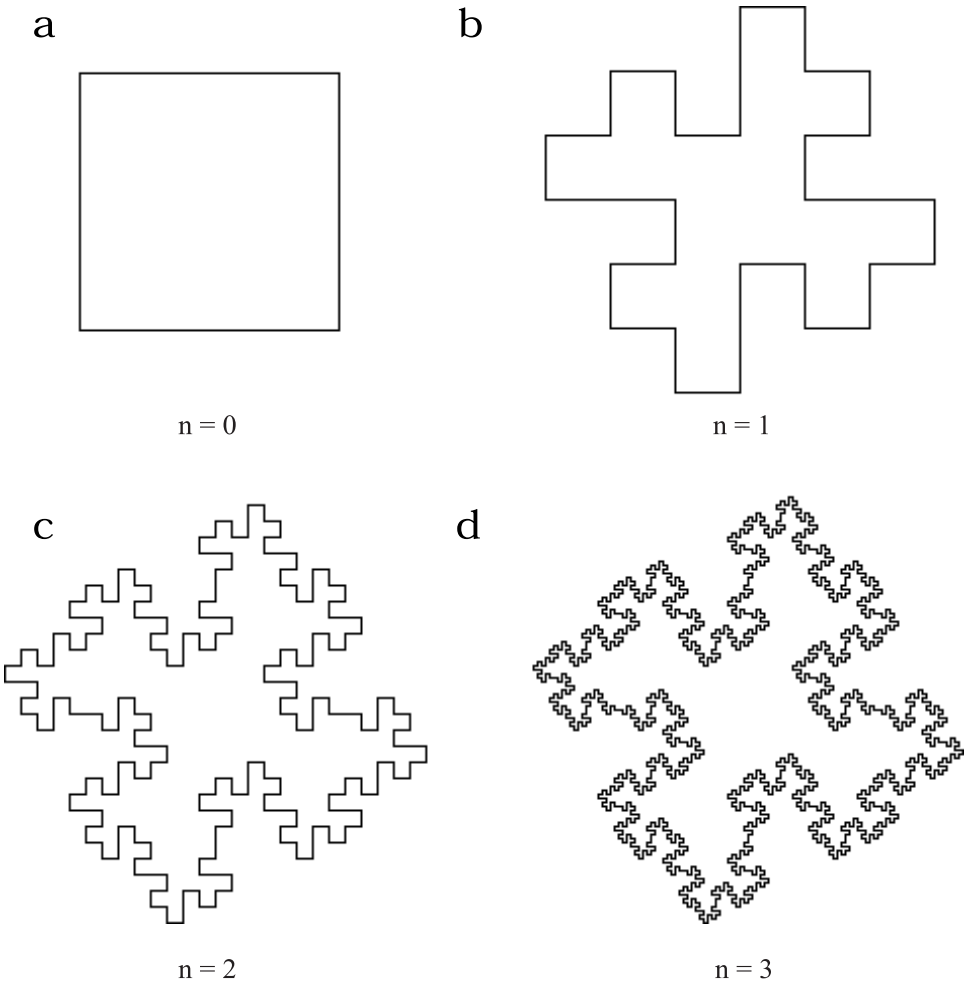


Figure 1.6: Generating a quadratic Koch island

between the endpoints of the successor polygon equal to the length of the predecessor segment.

Koch constructions vs. L-systems

The above example reveals a close relationship between Koch constructions and L-systems. The initiator corresponds to the axiom and the generator corresponds to the production successor. The predecessor F represents a single edge. L-systems specified in this way can be perceived as *codings* for Koch constructions. Figure 1.7 presents further examples of Koch curves generated using L-systems. A slight complication occurs if the curve is not connected; a second production (with the predecessor f) is then required to keep components the proper distance from each other (Figure 1.8). The ease of modifying L-systems makes them suitable for developing new Koch curves. For example, one can start from a particular L-system and observe the results of inserting, deleting or replacing some symbols. A variety of curves obtained this way are shown in Figure 1.9.

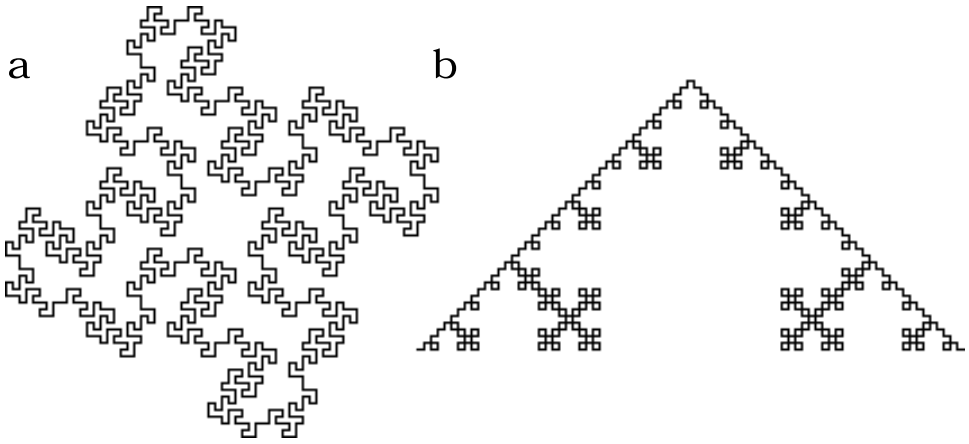


Figure 1.7: Examples of Koch curves generated using L-systems: (a) Quadratic Koch island [95, page 52], (b) A quadratic modification of the snowflake curve [95, page 139]

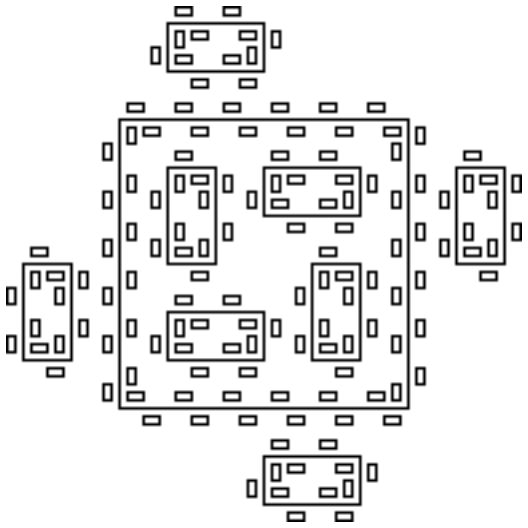
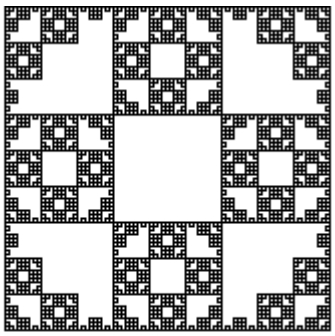
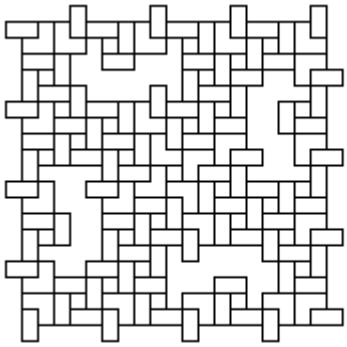


Figure 1.8: Combination of islands and lakes [95, page 121]

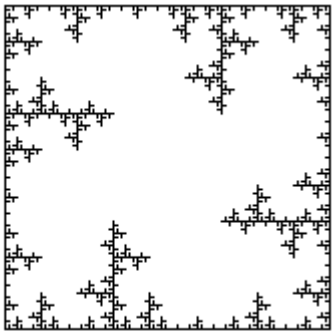
a n = 4, $\delta = 90^\circ$
F-F-F-F
F → FF-F-F-F-F-F+F



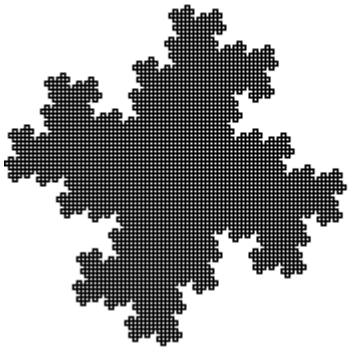
b n = 4, $\delta = 90^\circ$
F-F-F-F
F → FF-F-F-F-FF



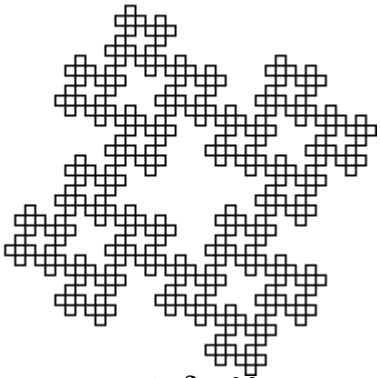
c n = 3, $\delta = 90^\circ$
F-F-F-F
F → FF-F+F-F-FF



d n = 4, $\delta = 90^\circ$
F-F-F-F
F → FF-F--F-F

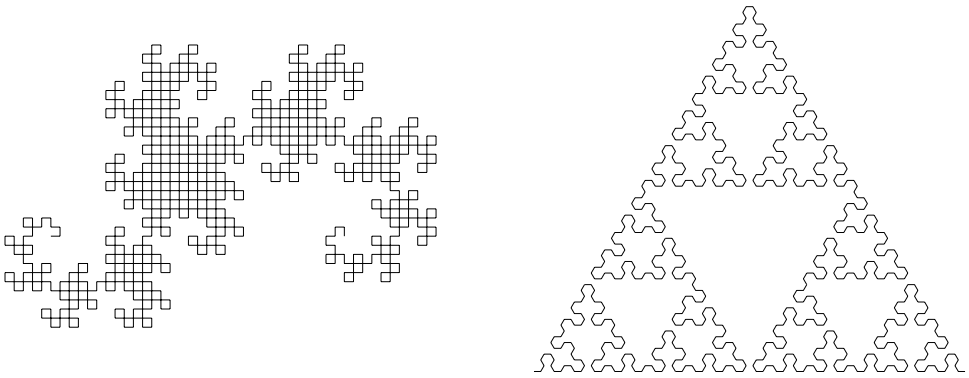


e n = 5, $\delta = 90^\circ$
F-F-F-F
F → F-FF--F-F



f n = 4, $\delta = 90^\circ$
F-F-F-F
F → F-F+F-F-F

Figure 1.9: A sequence of Koch curves obtained by successive modification of the production successor



a n=10, δ=90°
F_l
$F_l \rightarrow F_l+F_r+$
$F_r \rightarrow -F_l-F_r$

b n=6, δ=60°
F_r
$F_l \rightarrow F_r+F_l+F_r$
$F_r \rightarrow F_l-F_r-F_l$

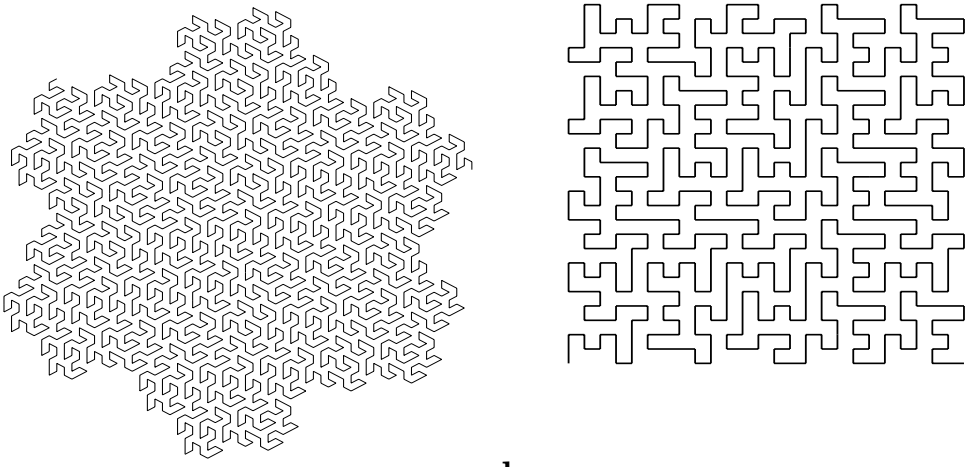
Figure 1.10: Examples of curves generated by edge-rewriting L-systems: (a) the dragon curve [48], (b) the Sierpiński gasket [132]

1.4 Synthesis of DOL-systems

Random modification of productions gives little insight into the relationship between L-systems and the figures they generate. However, we often wish to construct an L-system which captures a given structure or sequence of structures representing a developmental process. This is called the *inference problem* in the theory of L-systems. Although some algorithms for solving it were reported in the literature [79, 88, 89], they are still too limited to be of practical value in the modeling of higher plants. Consequently, the methods introduced below are more intuitive in nature. They exploit two modes of operation for L-systems with turtle interpretation, called *edge rewriting* and *node rewriting* using terminology borrowed from graph grammars [56, 57, 87]. In the case of edge rewriting, productions substitute figures for polygon edges, while in node rewriting, productions operate on polygon vertices. Both approaches rely on capturing the recursive structure of figures and relating it to a tiling of a plane. Although the concepts are illustrated using abstract curves, they apply to branching structures found in plants as well.

1.4.1 Edge rewriting

Edge rewriting can be viewed as an extension of Koch constructions. For example, Figure 1.10a shows the *dragon curve* [21, 48, 95] and the L-system that generated it. Both the F_l and F_r symbols represent edges created by the turtle executing the "move forward" command. The productions substitute F_l or F_r edges by pairs of lines forming



a
n=4, δ=60°
F_l
$F_l \rightarrow F_l+F_r++F_r-F_l--F_lF_l-F_r+$
$F_r \rightarrow -F_l+F_rF_r++F_r+F_l--F_l-F_r$

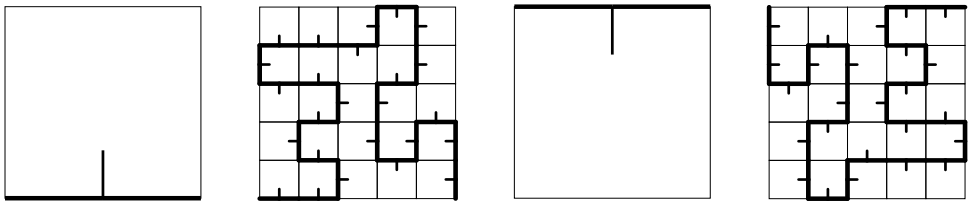
b
n=2, δ=90°
$-F_r$
$F_l \rightarrow F_lF_l-F_r-F_r+F_l+F_l-F_r-F_rF_l+$
$F_r+F_lF_lF_r-F_l+F_r+F_lF_l+$
$F_r-F_lF_r-F_r-F_l+F_l+F_rF_r-$
$F_r \rightarrow +F_lF_l-F_r-F_r+F_l+F_lF_r+F_l-$
$F_rF_r-F_l-F_r+F_lF_rF_r-F_l-$
$F_rF_l+F_l+F_r-F_r-F_l+F_l+F_rF_r$

Figure 1.11: Examples of FASS curves generated by edge-rewriting L-systems: (a) hexagonal Gosper curve [51], (b) quadratic Gosper curve [32] or E-curve [96]

left or right turns. Many interesting curves can be obtained assuming two types of edges, "left" and "right." Figures 1.10b and 1.11 present additional examples.

FASS curve construction

The curves included in Figure 1.11 belong to the class of *FASS* curves (an acronym for space-filling, self-avoiding, simple and self-similar) [116], which can be thought of as finite, self-avoiding approximations of curves that pass through *all* points of a square (space-filling curves [106]). McKenna [96] presented an algorithm for constructing FASS curves using edge replacement. It exploits the relationship between such a curve and a recursive subdivision of a square into tiles. For example, Figure 1.12 shows the tiling that corresponds to the E-curve of Figure 1.11b. The polygon replacing an edge F_l (Figure 1.12a) approximately fills the square on the left side of F_l (b). Similarly, the polygon replacing an edge F_r (c) approximately fills the square on the right side of that edge (d). Consequently, in the next derivation step, each of the 25 tiles associated with the curves (b) or (d) will be covered by their reduced copies (Figure 1.11b). A recursive application of this argument indicates that the whole curve is approximately space-filling. It is also self-avoiding due to the following two properties:



$$
\begin{aligned}
F_l \rightarrow\ & F_l F_l + F_r + F_r - F_l - F_l + F_r + F_r F_l - F_r - F_l F_l F_r + \\
& F_l - F_r - F_l F_l - F_r + F_l F_r + F_r + F_l - F_l - F_r F_r + \\
F_r \rightarrow\ & - F_l F_l + F_r + F_r - F_l - F_l F_r - F_l + F_r F_r + F_l + F_r - \\
& F_l F_r F_r + F_l + F_r F_l - F_l - F_r + F_r + F_l - F_l - F_r F_r
\end{aligned}
$$

Figure 1.12: Construction of the E-curve on the square grid. Left and right edges are distinguished by the direction of ticks.

- the generating polygon is self-avoiding, and
- no matter what the relative orientation of the polygons lying on two adjacent tiles, their union is a self-avoiding curve.

The first property is obvious, while the second can be verified by considering all possible relative positions of a pair of adjacent tiles.

Using a computer program to search the space of generating polygons, McKenna found that the E-curve is the simplest FASS curve obtained by edge replacement in a square grid. Other curves require generators with more edges (Figure 1.13). The relationship between edge rewriting and tiling of the plane extends to branching structures, providing a method for constructing and analyzing L-systems which operate according to the edge-rewriting paradigm (see Section 1.10.3).

1.4.2 Node rewriting

Subfigures

The idea of node rewriting is to substitute new polygons for nodes of the predecessor curve. In order to make this possible, turtle interpretation is extended by symbols which represent arbitrary subfigures. As shown in Figure 1.14, each subfigure A from a set of subfigures $\mathcal{A}$ is represented by:

- 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* $\vec{q}_A$.

During turtle interpretation of a string ν, a symbol $A \in \mathcal{A}$ incorporates the corresponding subfigure into a picture. To this end, A is translated

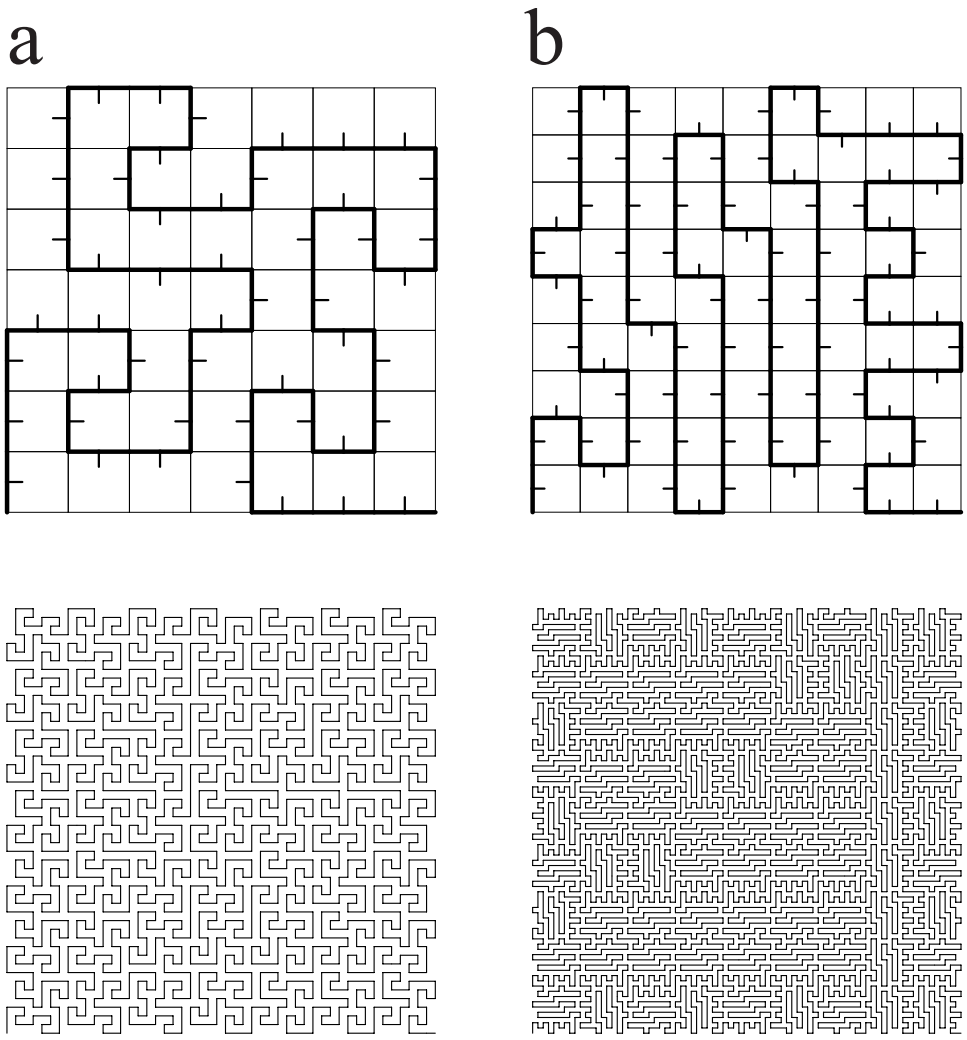


Figure 1.13: Examples of FASS curves generated on the square grid using edge replacement: (a) a SquaRecurve (grid size 7×7), (b) an E-tour (grid size 9×9). Both curves are from [96].

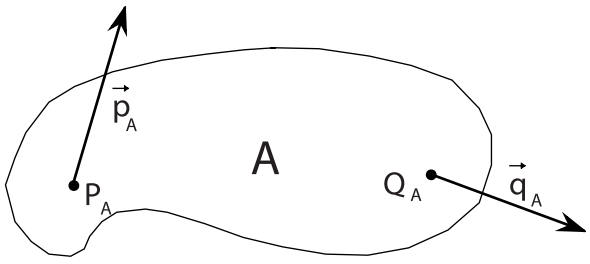


Figure 1.14: Description of a subfigure A

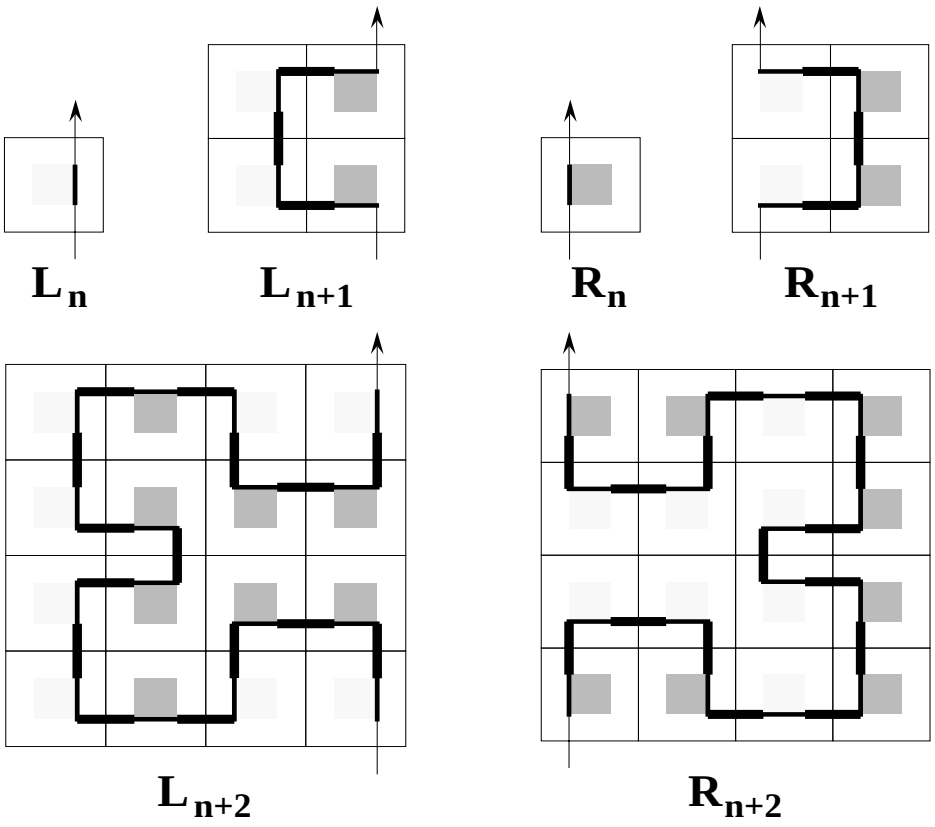


Figure 1.15: Recursive construction of the Hilbert curve [63] in terms of node replacement

and rotated in order to align its entry point P_A and direction $\vec{p}_A$ with the current position and orientation of the turtle. Having placed A, the turtle is assigned the resulting exit position Q_A and direction $\vec{q}_A$.

Recursive formulas

For example, assuming that the contact points and directions of subfigures L_n and R_n are as in Figure 1.15, the figures L_{n+1} and R_{n+1} are captured by the following formulas:

$$\begin{aligned} L_{n+1} &= +R_nF - L_nFL_n - FR_n+ \\ R_{n+1} &= -L_nF + R_nFR_n + FL_n- \end{aligned}$$

Suppose that curves L_0 and R_0 are given. One way of evaluating the string L_n (or R_n) for $n > 0$ is to generate successive strings recursively, in the order of decreasing value of index n. For example, the computation of L_2 would proceed as follows:

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

Thus, the generation of string L_n can be considered as a string-rewriting mechanism, where the symbols on the left side of the recursive formulas are substituted by corresponding strings on the right side. The substitution proceeds in a parallel way with $F, +$ and $-$ 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. Consequently, string L_n can be obtained in a derivation of length n using the following L-system:

$$\begin{array}{ll} \omega: & L \\ p_1: & L \rightarrow +RF - LFL - FR+ \\ p_2: & R \rightarrow -LF + RFR + FL- \end{array}$$

Pure curves

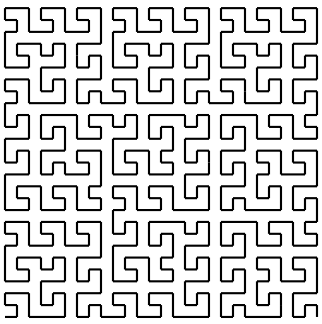
In order to complete the curve definition, it is necessary to specify the subfigures represented by symbols L and R. In the special case of *pure curves* [116], these subfigures are reduced to single points. Thus, one can assume that symbols L and R are erased (replaced by the empty string) at the end of the derivation. Alternatively, they can be left in the string and ignored by the turtle during string interpretation. This second approach is consistent with previous definitions of turtle interpretation [109, 112]. A general discussion of the relationship between recurrent formulas and L-systems is presented in [61, 62].

FASS curve construction

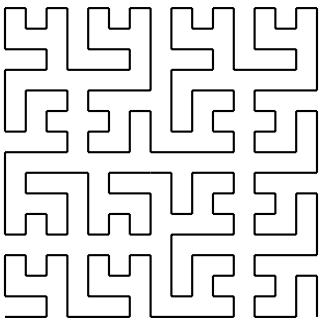
Construction of the L-system generating the Hilbert curve can be extended to other FASS curves [116]. Consider an array of $m \times m$ square tiles, each including a smaller square, called a *frame*. The edges of the frame run at some distance from the tile's edges. Each frame bounds an open self-avoiding polygon. The endpoints of this polygon coincide with the two contact vertices of the frame. Suppose that a single-stroke line running through all tiles can be constructed by connecting the contact vertices of neighboring frames using short horizontal or vertical line segments. A FASS curve can be constructed by the recursive repetition of this connecting pattern. To this end, the array of $m \times m$ connected tiles is considered a *macrotile* which contains an open polygon inscribed into a *macroframe*. An array of $m \times m$ macrotiles is formed, and the polygons inscribed into the macroframes are connected together. This construction is carried out recursively, with $m \times m$ macrotiles at level n yielding one macrotile at level $n + 1$.

Tile arrangements suitable for the generation of FASS curves can be found algorithmically, by searching the space of all possible arrangements on a grid of a given size. Examples of curves synthesized this way are given in Figures 1.16 and 1.17.

As in the case of edge rewriting, the relationship between node rewriting and tilings of the plane extends to branching structures. It offers a method for synthesizing L-systems that generate objects with a given recursive structure, and links methods for plant generation based on L-systems with those using iterated function systems [7] (see Chapter 8).



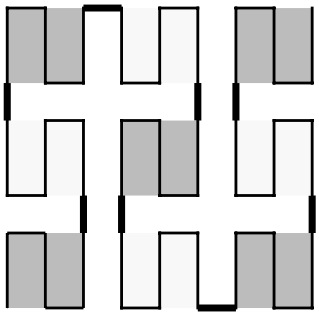
a
n=3,δ=90°
-L
L→LF+RFR+FL-F-LFLFL-FRFR+
R→-LFLF+RFRFR+F+RF-LFL-FR



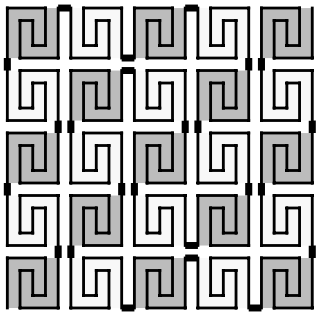
b
n=2, δ=90°
-L
L→LFLF+RFR+FLFL-FRF-LFL-
FR+F+RF-LFL-FRFRFR+
R→-LFLFLF+RFR+FL-F-LF+RFR+
FLF+RFRF-LFL-FRFR

Figure 1.16: Sample FASS curves constructed using tiles with contact points positioned along a tile edge: (a) 3×3 tiles form a macrotile, (b) 4×4 tiles form a macrotile

a



b



a n=2, δ=90°
L
L→LFRFL-F-RFLFR+F+LFRFL
R→RFLFR+F+LFRFL-F-RFLFR

b n=2, δ=45°
L
L→L+F+R-F-L+F+R-F-L-F-R+F+L-F-R-F-L+F+R-F-L-F-R-F-
L+F+R+F+L+F+R-F-L+F+R+F+L-R-F+F+L+F+R-F-L+F+R-F-L
R→R-F-L+F+R-F-L+F+R+F+L-F-R+F+L+F+R-F-L+F+R+F+L+F+
R-F-L-F-R-F-L+F+R-F-L-F-R+F+L-F-R-F-L+F+R-F-L+F+R

Figure 1.17: Sample FASS curves constructed using tiles with contact points positioned diagonally: (a) 3×3 tiles form a macrotile (Peano curve [106]), (b) 5×5 tiles form a macrotile

1.4.3 Relationship between edge and node rewriting

The classes of curves that can be generated using the edge-rewriting and node-rewriting techniques are not disjoint. For example, reconsider the L-system that generates the dragon curve using edge replacement:

$$\begin{array}{ll} \omega: & F_l \\ p_1: & F_l \rightarrow F_l + F_r + \\ p_2: & F_r \rightarrow -F_l - F_r \end{array}$$

Assume temporarily that a production predecessor can contain more than one letter; thus an entire subword can be replaced by the successor of a single production (a formalization of this concept is termed *pseudo-L-systems* [109]). The dragon-generating L-system can be rewritten as:

$$\begin{array}{ll} \omega: & Fl \\ p_1: & Fl \rightarrow Fl + rF + \\ p_2: & rF \rightarrow -Fl - rF \end{array}$$

where the symbols l and r are not interpreted by the turtle. Production p_1 replaces the letter l by the string $l + rF -$ while the leading letter F is left intact. In a similar way, production p_2 replaces the letter r by the string $-Fl - r$ and leaves the trailing F intact. Thus, the L-system can be transformed into node-rewriting form as follows:

$$\begin{array}{ll} \omega: & Fl \\ p_1: & l \rightarrow l + rF + \\ p_2: & r \rightarrow -Fl - r \end{array}$$

In practice, the choice between edge rewriting and node rewriting is often a matter of convenience. Neither approach offers an automatic, general method for constructing L-systems that capture given structures. However, the distinction between edge and node rewriting makes it easier to understand the intricacies of L-system operation, and in this sense assists in the modeling task. Specifically, the problem of filling a region by a self-avoiding curve is biologically relevant, since some plant structures, such as leaves, may tend to fill a plane without overlapping [38, 66, 67, 94].

1.5 Modeling in three dimensions

Turtle interpretation of L-systems can be extended to three dimensions following the ideas of Abelson and diSessa [1]. The key concept is to represent the current *orientation* of the turtle in space by three vectors $\vec{H}, \vec{L}, \vec{U}$, indicating the turtle's *heading*, the direction to the *left*, and the direction *up*. These vectors have unit length, are perpendicular to each

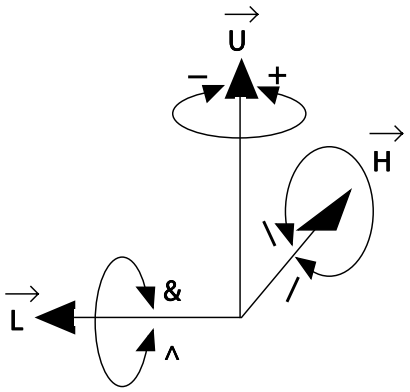


Figure 1.18: Controlling the turtle in three dimensions

other, and satisfy the equation $\vec{H} \times \vec{L} = \vec{U}$. Rotations of the turtle are then expressed by the equation

$$\left[\begin{array}{ccc} \vec{H}' & \vec{L}' & \vec{U}' \end{array} \right] = \left[\begin{array}{ccc} \vec{H} & \vec{L} & \vec{U} \end{array} \right] \mathbf{R},$$

where $\mathbf{R}$ is a 3×3 rotation matrix [40]. Specifically, rotations by angle α about vectors $\vec{U}, \vec{L}$ and $\vec{H}$ are represented by the matrices:

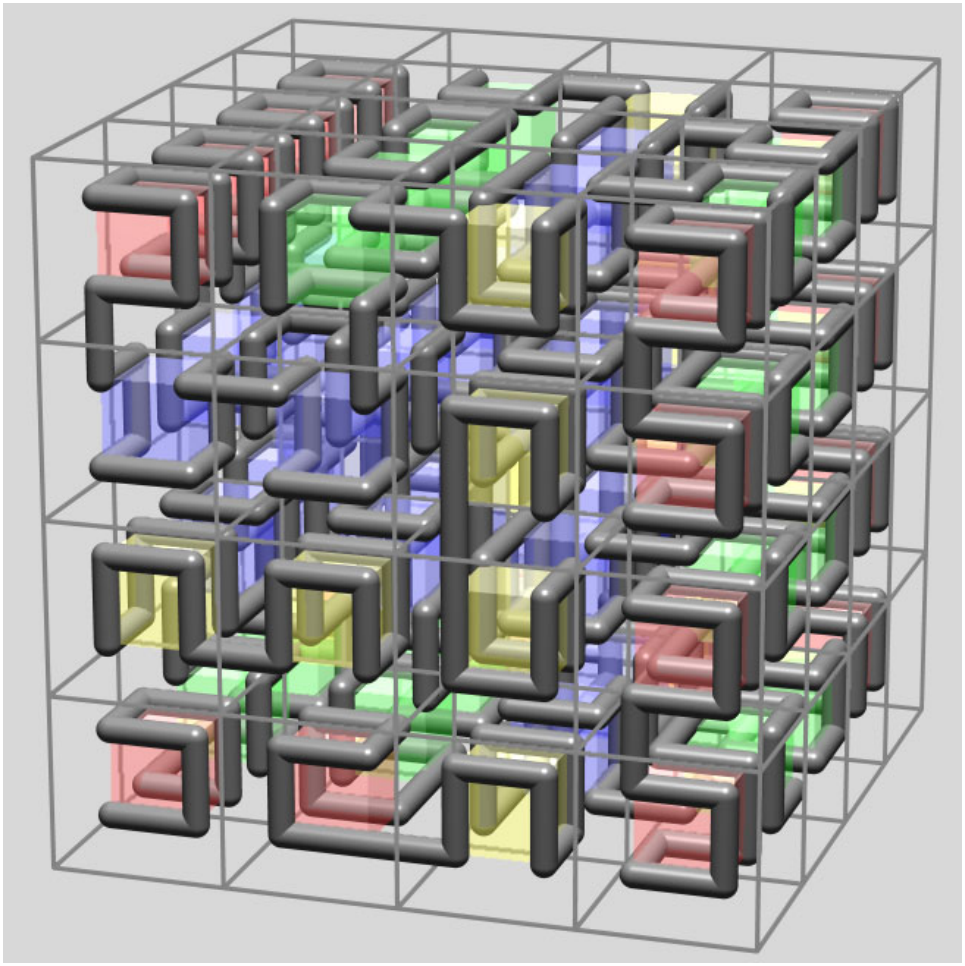
$$\mathbf{R_U}(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R_L}(\alpha) = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

$$\mathbf{R_H}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

The following symbols control turtle orientation in space (Figure 1.18):

$+$	Turn left by angle δ, using rotation matrix $\mathbf{R_U}(\delta)$.
$-$	Turn right by angle δ, using rotation matrix $\mathbf{R_U}(-\delta)$.
&	Pitch down by angle δ, using rotation matrix $\mathbf{R_L}(\delta)$.
$\wedge$	Pitch up by angle δ, using rotation matrix $\mathbf{R_L}(-\delta)$.
\	Roll left by angle δ, using rotation matrix $\mathbf{R_H}(\delta)$.
/	Roll right by angle δ, using rotation matrix $\mathbf{R_H}(-\delta)$.
|	Turn around, using rotation matrix $\mathbf{R_U}(180^\circ)$.



n=2, δ=90°
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//

Figure 1.19: A three-dimensional extension of the Hilbert curve [139]. Colors represent three-dimensional "frames" associated with symbols A (red), B (blue), C (green) and D (yellow).

As an example of a three-dimensional object created using an L-system, consider the extension of the Hilbert curve shown in Figure 1.19. The L-system was constructed with the node-replacement technique discussed in the previous section, using cubes and "macrocubes" instead of tiles and macrotiles.

1.6 Branching structures

According to the rules presented so far, the turtle interprets a character string as a sequence of line segments. Depending on the segment lengths and the angles between them, the resulting line is self-intersecting or not, can be more or less convoluted, and may have some segments drawn many times and others made invisible, but it always remains just a single line. However, the plant kingdom is dominated by branching structures; thus a mathematical description of tree-like shapes and methods for generating them are needed for modeling purposes. An axial tree [89, 117] complements the graph-theoretic notion of a rooted tree [108] with the botanically motivated notion of branch axis.

1.6.1 Axial trees

A *rooted tree* has edges that are labeled and directed. The edge sequences form paths from a distinguished node, called the *root* or *base*, to the *terminal nodes*. In the biological context, these edges are referred to as *branch segments*. A segment followed by at least one more segment in some path is called an *internode*. A terminal segment (with no succeeding edges) is called an *apex*.

An *axial tree* is a special type of rooted tree (Figure 1.20). At each of its nodes, at most one outgoing *straight* segment is distinguished. All remaining edges are called *lateral* or *side* segments. A sequence of segments is called an *axis* if:

- the first segment in the sequence originates at the root of the tree or as a lateral segment at some node,
- each subsequent segment is a straight segment, and
- the last segment is not followed by any straight segment in the tree.

Together with all its descendants, an axis constitutes a *branch*. A branch is itself an axial (sub)tree.

Axes and branches are ordered. The axis originating at the root of the entire plant has order zero. An axis originating as a lateral segment of an n-order parent axis has order $n+1$. The order of a branch is equal to the order of its lowest-order or *main* axis.

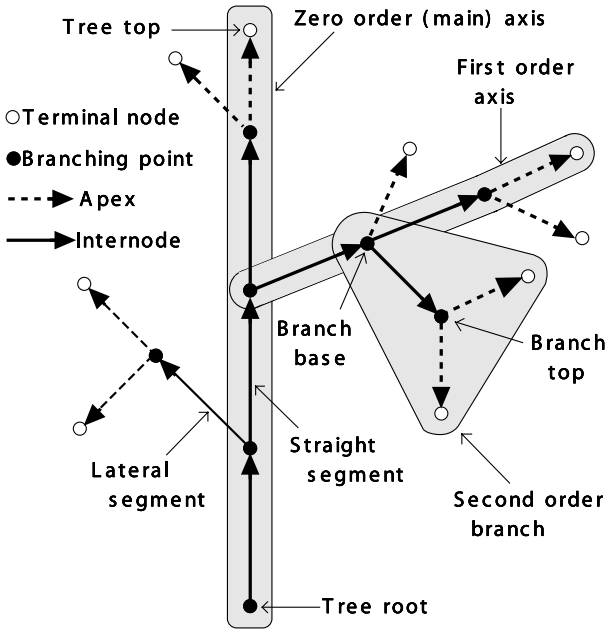


Figure 1.20: An axial tree

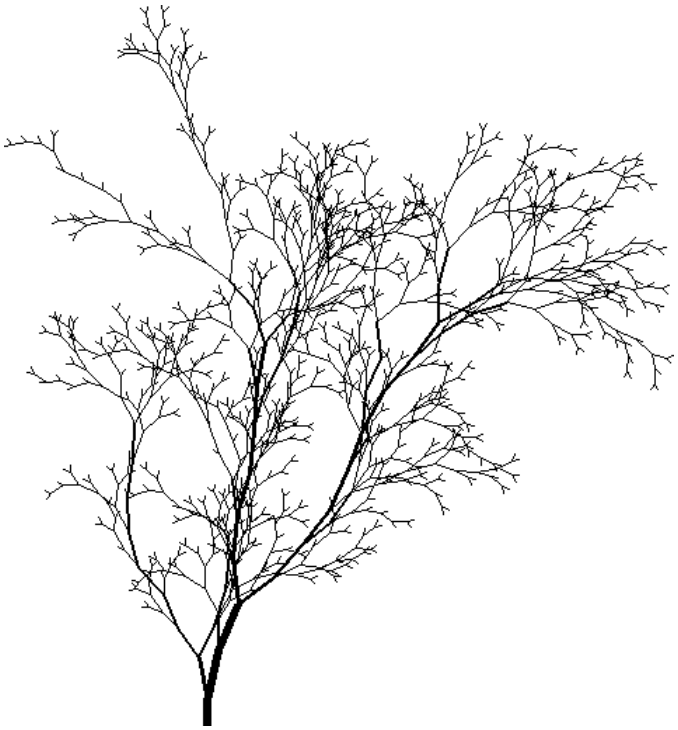


Figure 1.21: Sample tree generated using a method based on Horton–Strahler analysis of branching patterns

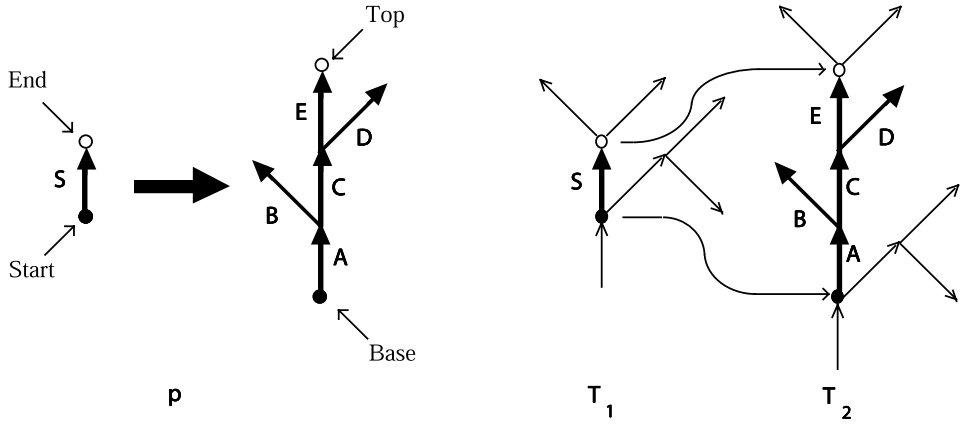


Figure 1.22: A tree production p and its application to the edge S in a tree T_1

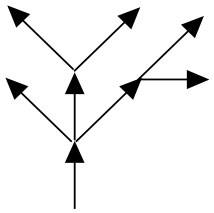
Axial trees are purely topological objects. The geometric connotation of such terms as straight segment, lateral segment and axis should be viewed at this point as an intuitive link between the graph-theoretic formalism and real plant structures.

The proposed scheme for ordering branches in axial trees was introduced originally by Gravelius [53]. MacDonald [94, pages 110–121] surveys this and other methods applicable to biological and geographical data such as stream networks. Of special interest are methods proposed by Horton [70, 71] and Strahler, which served as a basis for synthesizing botanical trees [37, 152] (Figure 1.21).

1.6.2 Tree OL-systems

In order to model development of branching structures, a rewriting mechanism can be used that operates directly on axial trees. A rewriting rule, or *tree production*, replaces a predecessor edge by a successor axial tree in such a way that the starting node of the predecessor is identified with the successor's base and the ending node is identified with the successor's top (Figure 1.22).

A *tree OL-system* G is specified by three components: a set of edge labels V, an initial tree ω with labels from V, and a set of tree productions P. Given the L-system G, an axial tree T_2 is directly derived from a tree T_1, noted $T_1 \Rightarrow T_2$, if T_2 is obtained from T_1 by simultaneously replacing each edge in T_1 by its successor according to the production set P. A tree T is generated by G in a derivation of length n if there exists a sequence of trees $T_0, T_1, \ldots, T_n$ such that $T_0 = \omega, T_n = T$ and $T_0 \Rightarrow T_1 \Rightarrow \ldots \Rightarrow T_n$.



F[+F][-F[-F]F]F[+F][-F]

Figure 1.23: Bracketed string representation of an axial tree

1.6.3 Bracketed OL-systems

The definition of tree L-systems does not specify the data structure for representing axial trees. One possibility is to use a list representation with a tree topology. Alternatively, axial trees can be represented using *strings with brackets* [82]. A similar distinction can be observed in Koch constructions, which can be implemented either by rewriting edges and polygons or their string representations. An extension of turtle interpretation to strings with brackets and the operation of bracketed L-systems [109, 111] are described below.

Two new symbols are introduced to delimit a branch. They are interpreted by the turtle as follows:

Stack operations

[Push the current state of the turtle onto a pushdown stack. The information saved on the stack contains the turtle's position and orientation, and possibly other attributes such as the color and width of lines being drawn.

] Pop a state from the stack and make it the current state of the turtle. No line is drawn, although in general the position of the turtle changes.

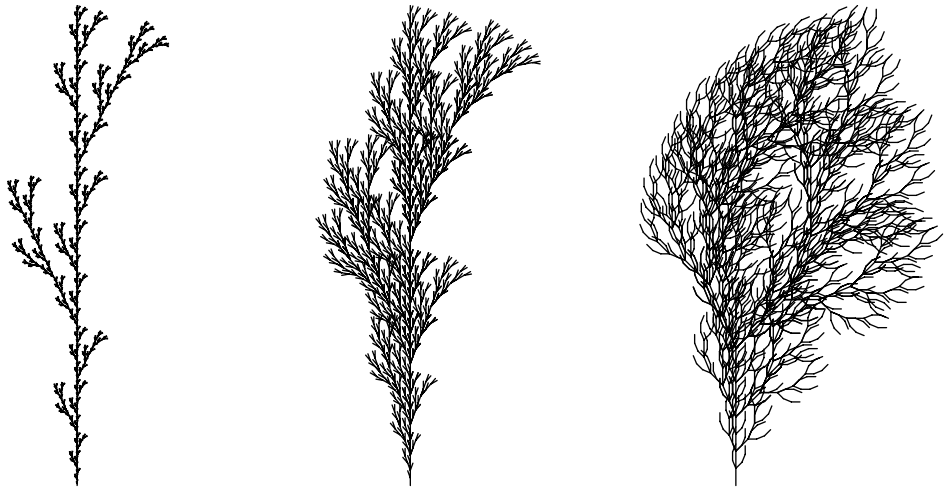
An example of an axial tree and its string representation are shown in Figure 1.23.

2D structures

Derivations in bracketed OL-systems proceed as in OL-systems without brackets. The brackets replace themselves. Examples of two-dimensional branching structures generated by bracketed OL-systems are shown in Figure 1.24.

Bush-like structure

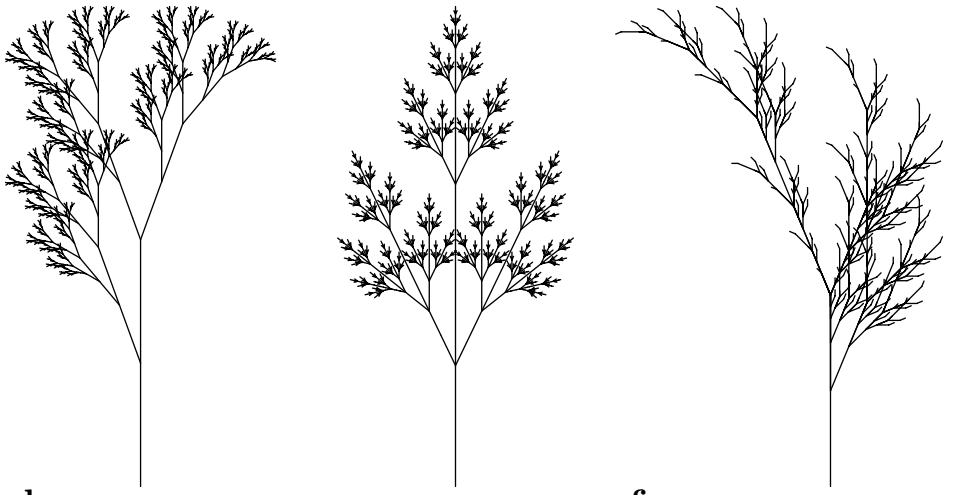
Figure 1.25 is an example of a three-dimensional bush-like structure generated by a bracketed L-system. Production p_1 creates three new branches from an apex of the old branch. A branch consists of an edge F forming the initial internode, a leaf L and an apex A (which will subsequently create three new branches). Productions p_2 and p_3



a
n=5, δ=25.7°
F
F →F[+F]F[-F]F

b
n=5, δ=20°
F
F →F[+F]F[-F][F]

c
n=4, δ=22.5°
F
F →FF-[-F+F+F]+
[+F-F-F]



d
n=7, δ=20°
X
X →F[+X]F[-X]+X
F →FF

e
n=7, δ=25.7°
X
X→F[+X][-X]FX
F→FF

f
n=5, δ=22.5°
X
X→F-[[X]+X]+F[+FX]-X
F→FF

Figure 1.24: Examples of plant-like structures generated by bracketed OL-systems. L-systems (a), (b) and (c) are edge-rewriting, while (d), (e) and (f) are node-rewriting.

n=7, δ=22.5°

ω : A
p_1 : A $\rightarrow$ [&FL!A]/////'[&FL!A]///////'[&FL!A]
p_2 : F $\rightarrow$ S ///// F
p_3 : S $\rightarrow$ F L
p_4 : L $\rightarrow$ [’’’$\wedge\wedge${-f+f+f-|-f+f+f}]

Figure 1.25: A three-dimensional bush-like structure

specify internode growth. In subsequent derivation steps the internode gets longer and acquires new leaves. This violates a biological rule of *subapical growth* (discussed in detail in Chapter 3), but produces an acceptable visual effect in a still picture. Production p_4 specifies the leaf as a filled polygon with six edges. Its boundary is formed from the edges f enclosed between the braces { and } (see Chapter 5 for further discussion). The symbols ! and ′ are used to decrement the diameter of segments and increment the current index to the color table, respectively.

Plant with flowers

Another example of a three-dimensional plant is shown in Figure 1.26. The L-system can be described and analyzed in a way similar to the previous one.

n=5, δ=18°

ω : **plant**
p_1 : **plant** → **internode** + [**plant** + **flower**] − − //
[− − **leaf**] **internode** [+ + **leaf**] −
[**plant flower**] + + **plant flower**
p_2 : **internode** → F **seg** [// & & **leaf**] [// ∧ ∧ **leaf**] F **seg**
p_3 : **seg** → **seg** F **seg**
p_4 : **leaf** → [' { +f−ff−f+ | +f−ff−f }]
p_5 : **flower** → [& & & **pedicel** ‘ / **wedge** //// **wedge** ////
wedge //// **wedge** //// **wedge**]
p_6 : **pedicel** → FF
p_7 : **wedge** → [‘ ∧ F] [{ & & & & −f+f | −f+f }]

Figure 1.26: A plant generated by an L-system

1.7 Stochastic L-systems

All plants generated by the same deterministic L-system are identical. An attempt to combine them in the same picture would produce a striking, artificial regularity. In order to prevent this effect, it is necessary to introduce specimen-to-specimen variations that will preserve the general aspects of a plant but will modify its details.

Variation can be achieved by randomizing the turtle interpretation, the L-system, or both. Randomization of the interpretation alone has a limited effect. While the geometric aspects of a plant — such as the stem lengths and branching angles — are modified, the underlying topology remains unchanged. In contrast, stochastic application of productions may affect both the topology and the geometry of the plant. The following definition is similar to that of Yokomori [162] and Eichhorst and Savitch [35].

L-system

A *stochastic OL-system* is an ordered quadruplet $G_\pi = \langle V, \omega, P, \pi \rangle$. The alphabet V, the axiom ω and the set of productions P are defined as in an OL-system (page 4). Function $\pi : P \to (0, 1]$, called the *probability distribution*, maps the set of productions into the set of *production probabilities*. It is assumed that for any letter $a \in V$, the sum of probabilities of all productions with the predecessor a is equal to 1.

Derivation

We will call the derivation $\mu \Rightarrow \nu$ a *stochastic derivation* in G_π if for each *occurrence* of the letter a in the word μ the probability of applying production p with the predecessor a is equal to $\pi(p)$. Thus, different productions with the same predecessor can be applied to various occurrences of the same letter in one derivation step.

Example

A simple example of a stochastic L-system is given below.

$$\begin{aligned} \omega &: F \\ p_1 &: F \xrightarrow{.33} F[+F]F[-F]F \\ p_2 &: F \xrightarrow{.33} F[+F]F \\ p_3 &: F \xrightarrow{.34} F[-F]F \end{aligned}$$

The production probabilities are listed above the derivation symbol $\rightarrow$. Each production can be selected with approximately the same probability of 1/3. Examples of branching structures generated by this L-system with derivations of length 5 are shown in Figure 1.27. Note that these structures look like different specimens of the same (albeit fictitious) plant species.

Flower field

A more complex example is shown in Figure 1.28. The field consists of four rows and four columns of plants. All plants are generated by a stochastic modification of the L-system used to generate Figure 1.26.

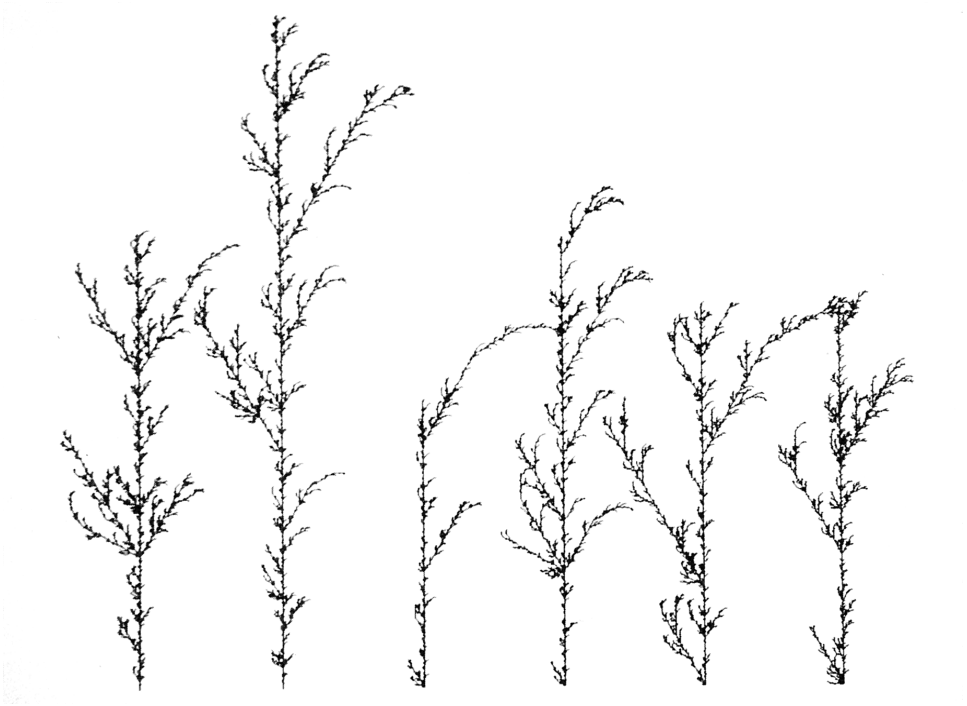


Figure 1.27: Stochastic branching structures

Figure 1.28: Flower field

The essence of this modification is to replace the original production p_3 by the following three productions:

$$\begin{aligned} p_3' &: \quad \mathbf{seg} \xrightarrow{.33} \mathbf{seg}\ [\ //\ \&\ \&\ \mathbf{leaf}\ ]\ [//\ \wedge\wedge\ \mathbf{leaf}\ ]\ F\ \mathbf{seg} \\ p_3'' &: \quad \mathbf{seg} \xrightarrow{.33} \mathbf{seg}\ F\ \mathbf{seg} \\ p_3''' &: \quad \mathbf{seg} \xrightarrow{.34} \mathbf{seg} \end{aligned}$$

Thus, in any step of the derivation, the stem segment **seg** may either grow and produce new leaves (production p_3'), grow without producing new leaves (production p_3''), or not grow at all (production p_3'''). All three events occur with approximately the same probability. The resulting field appears to consist of various specimens of the same plant species. If the same L-system was used again (with different seed values for the random number generator), a variation of this image would be obtained.

1.8 Context-sensitive L-systems

Context in string L-systems

Productions in OL-systems are context-free; i.e. applicable regardless of the context in which the predecessor appears. However, production application may also depend on the predecessor's context. This effect is useful in simulating interactions between plant parts, due for example to the flow of nutrients or hormones. Various context-sensitive extensions of L-systems have been proposed and studied thoroughly in the past [62, 90, 128]. *2L-systems* use productions of the form $a_l < a > a_r \rightarrow \chi$, where the letter a (called the *strict predecessor*) can produce word χ if and only if a is preceded by letter a_l and followed by a_r. Thus, letters a_l and a_r form the left and the right *context* of a in this production. Productions in *1L-systems* have one-sided context only; consequently, they are either of the form $a_l < a \rightarrow \chi$ or $a > a_r \rightarrow \chi$. OL-systems, 1L-systems and 2L-systems belong to a wider class of *IL-systems*, also called *(k,l)-systems*. In a (k,l)-system, the left context is a word of length k and the right context is a word of length l.

In order to keep specifications of L-systems short, the usual notion of IL-systems has been modified here by allowing productions with different context lengths to coexist within a single system. Furthermore, context-sensitive productions are assumed to have precedence over context-free productions with the same strict predecessor. Consequently, if a context-free and a context-sensitive production both apply to a given letter, the context-sensitive one should be selected. If no production applies, this letter is replaced by itself as previously assumed for OL-systems.

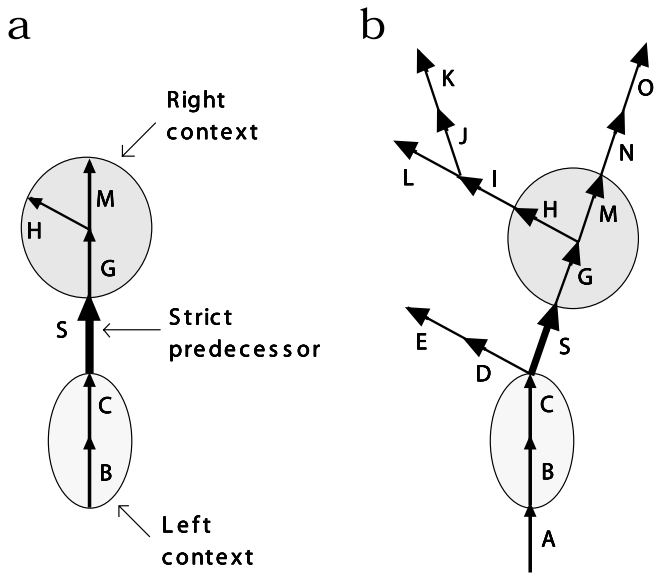


Figure 1.29: The predecessor of a context-sensitive tree production (a) matches edge S in a tree T (b)

Signal propagation

The following sample 1L-system makes use of context to simulate signal propagation throughout a string of symbols:

$$\begin{array}{lll} \omega & : & baaaaaaaa \\ p_1 & : & b < a \quad \rightarrow b \\ p_2 & : & \quad b \quad \rightarrow a \end{array}$$

The first few words generated by this L-system are given below:

$$\begin{array}{l} baaaaaaaa \\ abaaaaaaa \\ aabaaaaaa \\ aaabaaaaa \\ aaaabaaaa \\ \ldots \end{array}$$

The letter b moves from the left side to the right side of the string.

Context in tree L-systems

A context-sensitive extension of tree L-systems requires neighbor edges of the replaced edge to be tested for context matching. A predecessor of a context-sensitive production p consists of three components: a *path* l forming the left context, an *edge* S called the strict predecessor, and an *axial tree* r constituting the right context (Figure 1.29). The asymmetry between the left context and the right context reflects the fact that there is only one path from the root of a tree to a given edge, while there can be many paths from this edge to various terminal nodes. Production p matches a given occurrence of the edge S in a tree T if l is a path in T terminating at the starting node of S, and r is a subtree

of T originating at the ending node of S. The production can then be applied by replacing S with the axial tree specified as the production successor.

Context in bracketed L-systems

The introduction of context to bracketed L-systems is more difficult than in L-systems without brackets, because the bracketed string representation of axial trees does not preserve segment neighborhood. Consequently, the context matching procedure may need to skip over symbols representing branches or branch portions. For example, Figure 1.29 indicates that a production with the predecessor $BC < S > G[H]M$ can be applied to symbol S in the string

$$ABC[DE][SG[HI[JK]L]MNO],$$

which involves skipping over symbols $[DE]$ in the search for left context, and $I[JK]L$ in the search for right context.

Within the formalism of bracketed L-systems, the left context can be used to simulate control signals that propagate *acropetally*, *i.e.*, from the root or basal leaves towards the apices of the modeled plant, while the right context represents signals that propagate *basipetally*, *i.e.*, from the apices towards the root. For example, the following 1L-system simulates propagation of an acropetal signal in a branching structure that does not grow:

$$\begin{array}{ll} \#\text{ignore} : +- & \\ & \\ \omega & : \quad F_b[+F_a]F_a[-F_a]F_a[+F_a]F_a \\ p_1 & : \quad F_b < F_a \rightarrow F_b \end{array}$$

Symbol F_b represents a segment already reached by the signal, while F_a represents a segment that has not yet been reached. The #ignore statement indicates that the geometric symbols $+$ and $-$ should be considered as non-existent while context matching. Images representing consecutive stages of signal propagation (corresponding to consecutive words generated by the L-system under consideration) are shown in Figure 1.30a.

The propagation of a basipetal signal can be simulated in a similar way (Figure 1.30b):

$$\begin{array}{ll} \#\text{ignore} : +- & \\ & \\ \omega & : \quad F_a[+F_a]F_a[-F_a]F_a[+F_a]F_b \\ p_1 & : \quad F_a > F_b \rightarrow F_b \end{array}$$

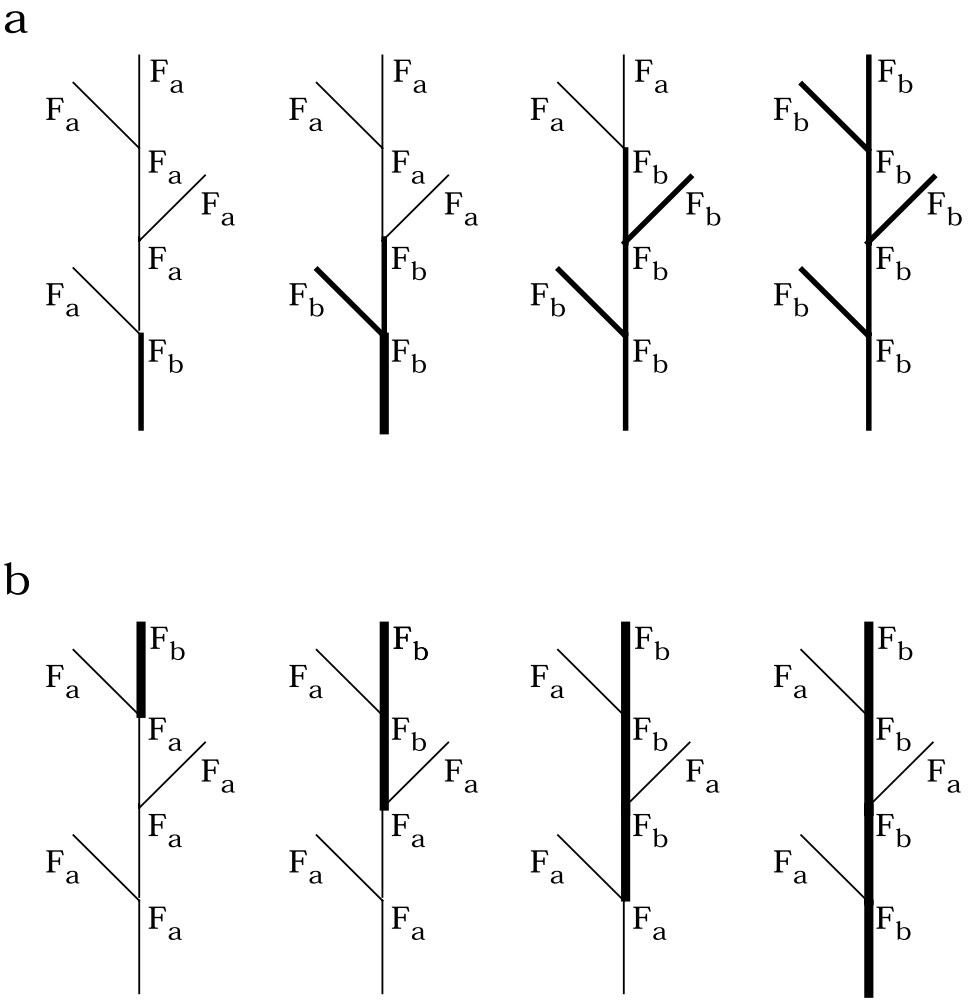


Figure 1.30: Signal propagation in a branching structure: (a) acropetal, (b) basipetal

L-systems of Hogeweg, Hesper and Smith

The operation of context-sensitive L-systems is examined further using examples obtained by Hogeweg and Hesper [64]. In 1974, they published the results of an exhaustive study of 3,584 patterns generated by a class of bracketed 2L-systems defined over the alphabet {0,1}. Some of these patterns had plant-like shapes. Subsequently, Smith significantly improved the quality of the generated images using state-of-the-art computer imagery techniques [136, 137]. Sample structures generated by L-systems similar to those proposed by Hogeweg and Hesper are shown in Figure 1.31. The differences are related to the geometric interpretation of the resulting strings. According to the original interpretation, consecutive branches are issued alternately to the left and right, whereas turtle interpretation requires explicit specification of branching angles within the L-system.

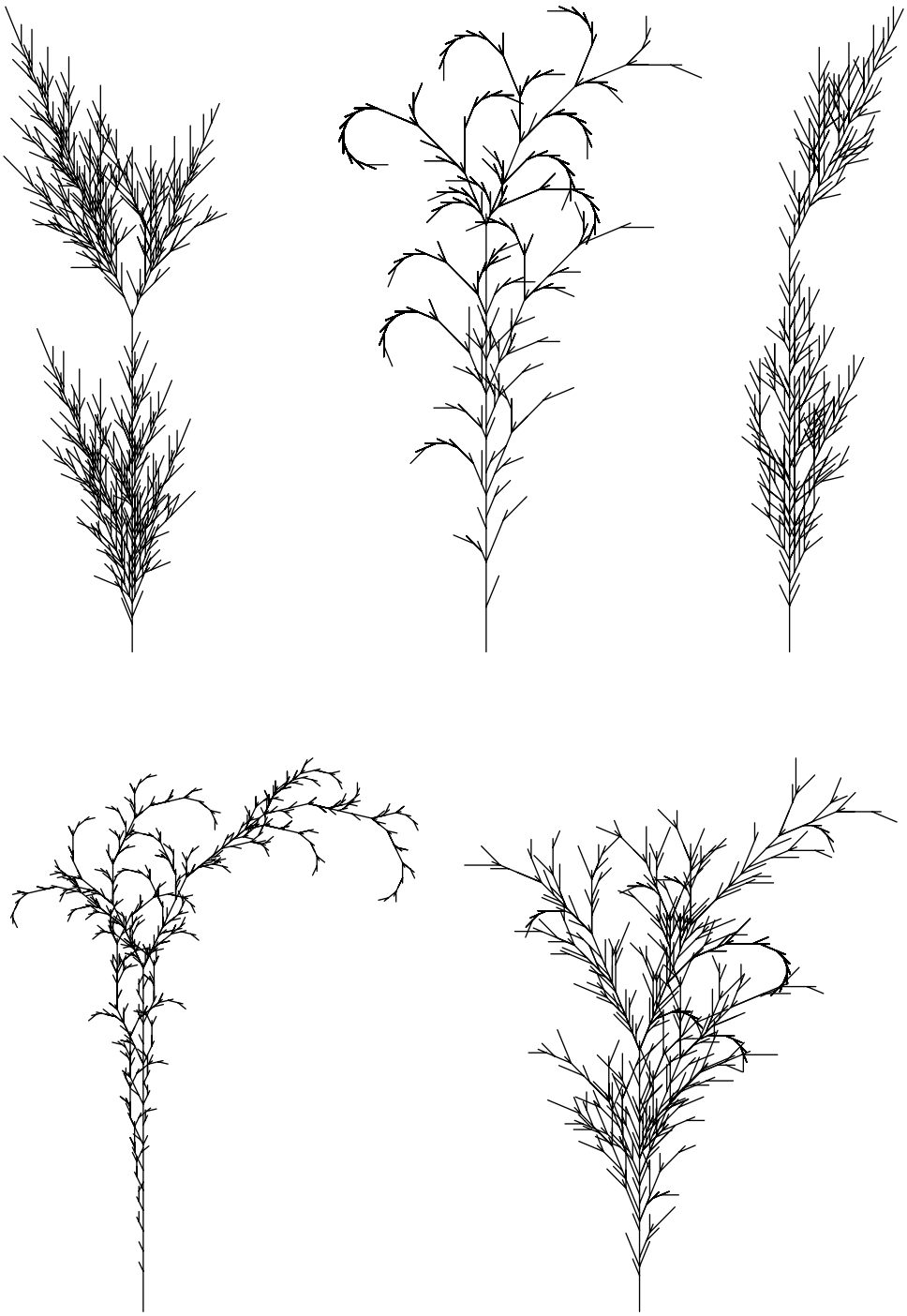


Figure 1.31: Examples of branching structures generated using L-systems based on the results of Hogeweg and Hesper [64]

a n=30,δ=22.5°
#ignore: +-F
F1F1F1
0 < 0 > 0 → 0
0 < 0 > 1 → 1[+F1F1]
0 < 1 > 0 → 1
0 < 1 > 1 → 1
1 < 0 > 0 → 0
1 < 0 > 1 → 1F1
1 < 1 > 0 → 0
1 < 1 > 1 → 0
* < + > * → -
* < - > * → +

b n=30,δ=22.5°
#ignore: +-F
F1F1F1
0 < 0 > 0 → 1
0 < 0 > 1 → 1[-F1F1]
0 < 1 > 0 → 1
0 < 1 > 1 → 1
1 < 0 > 0 → 0
1 < 0 > 1 → 1F1
1 < 1 > 0 → 1
1 < 1 > 1 → 0
* < + > * → -
* < - > * → +

c n=26, δ=25.75°
#ignore: +-F
F1F1F1
0 < 0 > 0 → 0
0 < 0 > 1 → 1
0 < 1 > 0 → 0
0 < 1 > 1 → 1[+F1F1]
1 < 0 > 0 → 0
1 < 0 > 1 → 1F1
1 < 1 > 0 → 0
1 < 1 > 1 → 0
* < - > * → +
* < + > * → -

d n=24, δ=25.75°
#ignore: +-F
F0F1F1
0 < 0 > 0 → 1
0 < 0 > 1 → 0
0 < 1 > 0 → 0
0 < 1 > 1 → 1F1
1 < 0 > 0 → 1
1 < 0 > 1 → 1[+F1F1]
1 < 1 > 0 → 1
1 < 1 > 1 → 0
* < + > * → -
* < - > * → +

e n=26, δ=22.5°
#ignore: +-F
F1F1F1
0 < 0 > 0 → 0
0 < 0 > 1 → 1[-F1F1]
0 < 1 > 0 → 1
0 < 1 > 1 → 1
1 < 0 > 0 → 0
1 < 0 > 1 → 1F1
1 < 1 > 0 → 1
1 < 1 > 1 → 0
* < + > * → -
* < - > * → +

Figure 1.31 (continued): L-systems of Hogeweg and Hesper

1.9 Growth functions

Exponential growth

During the synthesis of a plant model it is often convenient to distinguish productions that specify the branching pattern from those that describe elongation of plant segments. This separation can be observed in some of the L-systems considered so far. For example, in L-systems (d), (e) and (f) from Figure 1.24 the first productions capture the branching patterns, while the remaining productions, equal in all cases to $F \rightarrow FF$, describe elongation of segments represented by sequences of symbols F. The number of letters F in a string χ_n originating from a single letter F is doubled in each derivation step, thus the elongation is exponential, with $\text{length}(\chi_n) = 2^n$.

Basic properties

A function that describes the number of symbols in a word in terms of its derivation length is called a *growth function*. The theory of L-systems contains an extensive body of results on growth functions [62, 127]. The central observation is that the growth functions of DOL-systems are independent of the letter ordering in the productions and derived words. Consequently, the relation between the number of letter occurrences in a pair of words μ and ν, such that $\mu \Rightarrow \nu$, can be conveniently expressed using matrix notation.

Let $G = \langle V, \omega, P \rangle$ be a DOL-system and assume that letters of the alphabet V have been ordered, $V = \{a_1, a_2, \ldots, a_m\}$. Construct a square matrix $Q_{m \times m}$, where entry q_{ij} is equal to the number of occurrences of letter a_j in the successor of the production with predecessor a_i. Let $\underline{a}_i^k$ denote the number of occurrences of letter a_i in the word x generated by G in a derivation of length k. The definition of direct derivation in a DOL-system implies that

$$\left[\begin{array}{cccc} \underline{a}_1^k & \underline{a}_2^k & \cdots & \underline{a}_m^k \end{array} \right] \left[\begin{array}{cccc} q_{11} & q_{12} & \cdots & q_{1m} \\ q_{21} & q_{22} & \cdots & q_{2m} \\ \vdots & & & \\ q_{m1} & q_{m2} & \cdots & q_{mm} \end{array} \right] = \left[\begin{array}{cccc} \underline{a}_1^{k+1} & \underline{a}_2^{k+1} & \cdots & \underline{a}_m^{k+1} \end{array} \right].$$

This matrix notation is useful in the analysis of growth functions. For example, consider the following L-system:

$$\begin{array}{ll} \omega: & a \\ p_1: & a \rightarrow ab \\ p_2: & b \rightarrow a \end{array} \tag{1.2}$$

The relationship between the number of occurrences of letters a and b in two consecutively derived words is

$$\left[\begin{array}{cc} \underline{a}^k & \underline{b}^k \end{array} \right] \left[\begin{array}{cc} 1 & 1 \\ 1 & 0 \end{array} \right] = \left[\begin{array}{cc} \underline{a}^{k+1} & \underline{b}^{k+1} \end{array} \right]$$

or

$$\underline{a}^{k+1} = \underline{a}^k + \underline{b}^k = \underline{a}^k + \underline{a}^{k-1}$$

for $k = 1, 2, 3, \ldots$. From the axiom it follows that $\underline{a}^0 = 1$ and $\underline{a}^1 = \underline{b}^0 = 0$. Thus, the number of letters a in the strings generated by the L-system specified in equation (1.2) grows according to the Fibonacci series: $1, 1, 2, 3, 5, 8, \ldots$. This growth function was implemented by productions p_2 and p_3 in the L-system generating the bush in Figure 1.25 (page 26) to describe the elongation of its internodes.

Polynomial growth

Polynomial growth functions of arbitrary degree can be obtained using L-systems of the following form:

$$\begin{array}{ll} \omega: & a_0 \\ p_1: & a_0 \rightarrow a_0 a_1 \\ p_2: & a_1 \rightarrow a_1 a_2 \\ p_3: & a_2 \rightarrow a_2 a_3 \\ p_4: & a_3 \rightarrow a_3 a_4 \\ & \vdots \end{array}$$

The matrix Q is given below:

$$Q = \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots \\ 0 & 1 & 1 & 0 & \cdots \\ 0 & 0 & 1 & 1 & \cdots \\ 0 & 0 & 0 & 1 & \cdots \\ & & \vdots & & \end{bmatrix}$$

Thus, for any $i, k \geq 1$, the number $\underline{a}_i^k$ of occurrences of symbol a_i in the string generated in a derivation of length k satisfies the equality

$$\underline{a}_i^k + \underline{a}_{i+1}^k = \underline{a}_{i+1}^{k+1}.$$

Taking into consideration the axiom, the distribution of letters a_i as a function of the derivation length is captured by the following table (only non-zero terms are shown):

k	$\underline{a}_0^k$	$\underline{a}_1^k$	$\underline{a}_2^k$	$\underline{a}_3^k$	$\underline{a}_4^k$	$\underline{a}_5^k$	$\underline{a}_6^k$	$\underline{a}_7^k$
0	1							
1	1	1						
2	1	2	1					
3	1	3	3	1				
4	1	4	6	4	1			
5	1	5	10	10	5	1		
6	1	6	15	20	15	6	1	
7	1	7	21	35	35	21	7	1
				$\vdots$				

This table represents the Pascal triangle, thus for any $k \geq i \geq 1$ its terms satisfy the following equality:

$$\underline{a}_i^k = \left(\begin{array}{c} k \\ i \end{array} \right) = \frac{k(k-1)\cdots(k-i+1)}{1 \cdot 2 \cdots i}$$

Consequently, the number of occurrences of letter a_i as a function of the derivation length k is expressed by a polynomial of degree i. By identifying letter a_i with the turtle symbol F, it is possible to model internode elongation expressed by polynomials of arbitrary degree $i \geq 0$. This observation was generalized by Szilard [140], who developed an algorithm for constructing a DOL-system with growth functions specified by any positive, nondecreasing polynomials with integer coefficients [62, page 276].

Characterization

The examples of growth functions considered so far include exponential and polynomial functions. Rozenberg and Salomaa [127, pages 30–38] show that, in general, the growth function $f_G(n)$ of any DOL-system $G = \langle V, \omega, P \rangle$ is a combination of polynomial and exponential functions:

$$f_G(n) = \sum_{i=1}^{s} P_i(n)\rho_i^n \quad \text{for} \quad n \geq n_0, \tag{1.3}$$

where $P_i(n)$ denotes a polynomial with integer coefficients, ρ_i is a nonnegative integer, and n_0 is the total number of letters in the alphabet V. Unfortunately, many growth processes observed in nature cannot be described by equation (1.3). Two approaches are then possible within the framework of the theory of L-systems.

Sigmoidal growth

The first is to extend the size n_0 of the alphabet V, so that the growth process of interest will be captured by the initial derivation steps, $\omega = \mu_0 \Rightarrow \mu_1 \Rightarrow \cdots \Rightarrow \mu_{n_0}$, before equation (1.3) starts to apply. For example, the L-system

$$\begin{array}{lcll} \omega & : & a_0 & \\ p_i & : & a_i \rightarrow a_{i+1}b_0 & \text{for} \quad i < k \\ p_{k+j} & : & b_j \rightarrow b_{j+1}F & \text{for} \quad j < l \end{array} \tag{1.4}$$

over the alphabet $V = \{a_0, a_1, ..., a_k\} \cup \{b_0, b_1, ..., b_l\} \cup \{F\}$ can be used to approximate a sigmoidal elongation of a segment represented by a sequence of symbols F (Figure 1.32). The term *sigmoidal* refers to a function with a plot in the shape of the letter S. Such functions are commonly found in biological processes [143], with the initial part of the curve representing the growth of a young organism, and the latter part corresponding to the organism close to its final size.

Square-root growth

The second approach to the synthesis of growth functions outside the class captured by equation (1.3) is to use context-sensitive L-systems. For example, the following 2L-system has a growth function given by $f_G(n) = \lfloor \sqrt{n} \rfloor + 4$, where $\lfloor x \rfloor$ is the floor function.

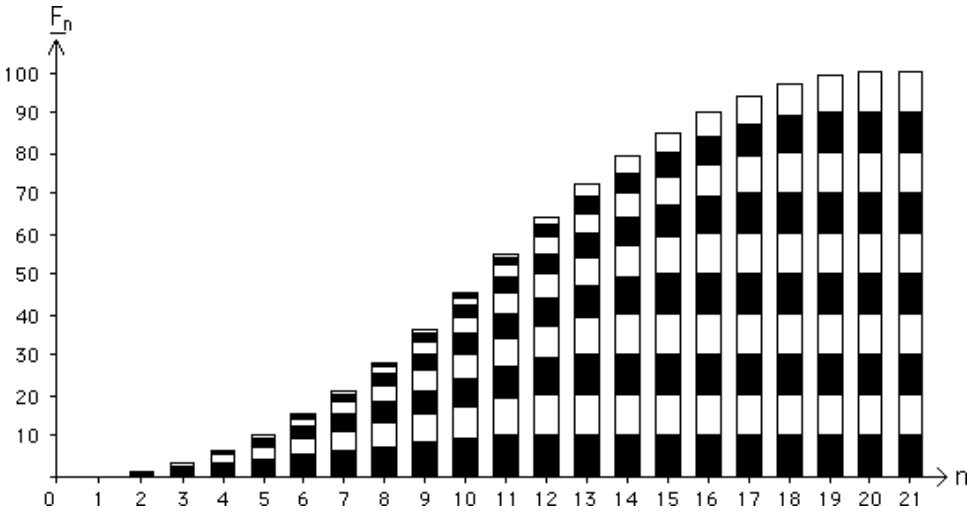


Figure 1.32: A sigmoidal growth function implemented using the L-system in equation (1.4), for $k = l = 20$

$$
\begin{array}{lrcl}
\omega & : & XF_uF_aX & \\
p_1 & : & F_u < F_a > F_a & \rightarrow \quad F_u \\
p_2 & : & F_u < F_a > X & \rightarrow \quad F_dF_a \\
p_3 & : & F_a < F_a > F_d & \rightarrow \quad F_d \\
p_4 & : & X < F_a > F_d & \rightarrow \quad F_u \\
p_5 & : & F_u & \rightarrow \quad F_a \\
p_6 & : & F_d & \rightarrow \quad F_a
\end{array}
\tag{1.5}
$$

The operation of this L-system is illustrated in Figure 1.33. Productions p_1 and p_3, together with p_5 and p_6, propagate symbols F_u and F_d up and down the string of symbols μ. Productions p_2 and p_4 change the propagation direction, after symbol X marking a string end has been reached by F_u or F_d, respectively. In addition, p_2 extends the string with a symbol F_a. Thus, the number of derivation steps increases by two between consecutive applications of production p_2. As a result, string extension occurs at derivation steps n expressed by the square of the string length, which yields the growth function $\lfloor\sqrt{n}\rfloor + 4$.

Limitations

In practice it is often difficult, if not impossible, to find L-systems with the required growth functions. Vitányi [153] illustrates this by referring to sigmoidal curves:

> If we want to obtain sigmoidal growth curves with the original L-systems then not even the introduction of cell interaction can help us out. In the first place, we end up constructing quite unlikely flows of messages through the organism, which are more suitable to electronic computers, and in fact give the organism universal computing power. Secondly, and this is more fundamental, we can not obtain

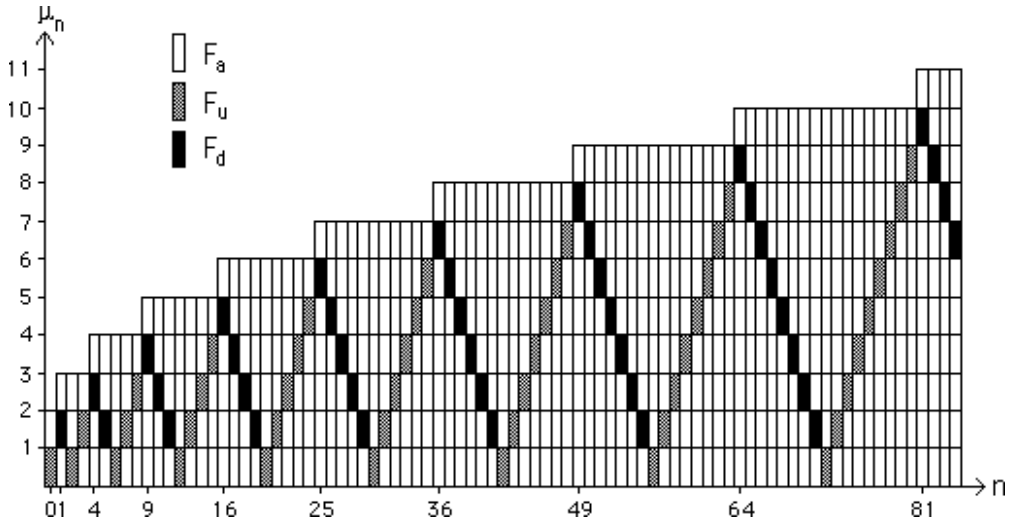


Figure 1.33: Square-root growth implemented using the L-system specified in equation (1.5)

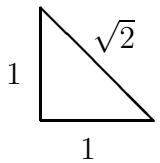
> growth which, always increasing the size of the organism, tends towards stability in the limit. The slowest increasing growth we can obtain by allowing cell interaction is logarithmic and thus can not at all account for the asymptotic behavior of sigmoidal growth functions.

In the next section we present an extension of L-systems that makes it possible to avoid this problem by allowing for explicit inclusion of growth functions into L-system specifications.

1.10 Parametric L-systems

Motivation

Although L-systems with turtle interpretation make it possible to generate a variety of interesting objects, from abstract fractals to plant-like branching structures, their modeling power is quite limited. A major problem can be traced to the reduction of all lines to integer multiples of the unit segment. As a result, even such a simple figure as an isosceles right-angled triangle cannot be traced exactly, since the ratio of its hypotenuse length to the length of a side is expressed by the irrational number $\sqrt{2}$. Rational approximation of line length provides only a limited solution, because the unit step must be the smallest common denominator of all line lengths in the modeled structure. Consequently, the representation of a simple plant module, such as an internode, may require a large number of symbols. The same argument applies to angles. Problems become even more pronounced while simulating changes to the modeled structure over time, since some growth functions cannot be expressed conveniently using L-systems. Generally, it is difficult



to capture continuous phenomena, since the obvious technique of discretizing continuous values may require a large number of quantization levels, yielding L-systems with hundreds of symbols and productions. Consequently, model specification becomes difficult, and the mathematical beauty of L-systems is lost.

In order to solve similar problems, Lindenmayer proposed that numerical parameters be associated with L-system symbols [83]. He illustrated this idea by referring to the continuous development of branching structures and diffusion of chemical compounds in a nonbranching filament of *Anabaena catenula*. Both problems were revisited in later papers [25, 77]. A definition of parametric L-systems was formulated by Prusinkiewicz and Hanan [113] and is presented below.

1.10.1 Parametric OL-systems

Parametric words

Parametric L-systems operate on *parametric words*, which are strings of *modules* consisting of *letters* with associated *parameters*. The letters belong to an *alphabet* V, and the parameters belong to the set of *real numbers* $\Re$. A module with letter $A \in V$ and parameters $a_1, a_2, ..., a_n \in \Re$ is denoted by $A(a_1, a_2, ..., a_n)$. Every module belongs to the set $M = V \times \Re^*$, where $\Re^*$ is the set of all finite sequences of parameters. The set of all strings of modules and the set of all nonempty strings are denoted by $M^* = (V \times \Re^*)^*$ and $M^+ = (V \times \Re^*)^+$, respectively.

Expressions

The real-valued *actual* parameters appearing in the words correspond with *formal* parameters used in the specification of L-system productions. If Σ is a set of formal parameters, then $C(\Sigma)$ denotes a *logical expression* with parameters from Σ, and $E(\Sigma)$ is an *arithmetic expression* with parameters from the same set. Both types of expressions consist of formal parameters and numeric constants, combined using the arithmetic operators $+$, $-$, $*$, $/$; the exponentiation operator $\wedge$, the relational operators $<$, $>$, $=$; the logical operators !, &, | (not, and, or); and parentheses (). Standard rules for constructing syntactically correct expressions and for operator precedence are observed. Relational and logical expressions evaluate to zero for false and one for true. A logical statement specified as the empty string is assumed to have value one. The sets of all correctly constructed logical and arithmetic expressions with parameters from Σ are noted $\mathcal{C}(\Sigma)$ and $\mathcal{E}(\Sigma)$.

Parametric OL-system

A *parametric OL-system* is defined as an ordered quadruplet $G = \langle V, \Sigma, \omega, P \rangle$, where

- V is the *alphabet* of the system,
- Σ is the *set of formal parameters*,
- $\omega \in (V \times \Re^*)^+$ is a nonempty parametric word called the *axiom*,
- $P \subset (V \times \Sigma^*) \times \mathcal{C}(\Sigma) \times (V \times \mathcal{E}(\Sigma))^*$ is a finite *set of productions*.

The symbols : and $\rightarrow$ are used to separate the three components of a production: the *predecessor*, the *condition* and the *successor*. For example, a production with predecessor $A(t)$, condition $t > 5$ and successor $B(t+1)CD(t \wedge 0.5, t-2)$ is written as

$$A(t) : t > 5 \rightarrow B(t+1)CD(t \wedge 0.5, t-2). \tag{1.6}$$

Derivation

A production *matches* a module in a parametric word if the following conditions are met:

- the letter in the module and the letter in the production predecessor are the same,
- the number of actual parameters in the module is equal to the number of formal parameters in the production predecessor, and
- the condition evaluates to *true* if the actual parameter values are substituted for the formal parameters in the production.

A matching production can be *applied* to the module, creating a string of modules specified by the production successor. The actual parameter values are substituted for the formal parameters according to their position. For example, production (1.6) above matches a module $A(9)$, since the letter A in the module is the same as in the production predecessor, there is one actual parameter in the module $A(9)$ and one formal parameter in the predecessor $A(t)$, and the logical expression $t > 5$ is true for $t = 9$. The result of the application of this production is a parametric word $B(10)CD(3, 7)$.

If a module a produces a parametric word χ as the result of a production application in an L-system G, we write $a \mapsto \chi$. Given a parametric word $\mu = a_1 a_2 ... a_m$, we say that the word $\nu = \chi_1 \chi_2 ... \chi_m$ is *directly derived* from (or *generated* by) μ and write $\mu \Longrightarrow \nu$ if and only if $a_i \mapsto \chi_i$ for all $i = 1, 2, ..., m$. A parametric word ν is generated by G in a *derivation of length* n if there exists a sequence of words $\mu_0, \mu_1, ..., \mu_n$ such that $\mu_0 = \omega$, $\mu_n = \nu$ and $\mu_0 \Longrightarrow \mu_1 \Longrightarrow ... \Longrightarrow \mu_n$.

Example

An example of a parametric L-system is given below.

$$\begin{array}{lllll} \omega: & B(2)A(4,4) & & & \\ p_1: & A(x,y) & : y <= 3 & \rightarrow & A(x*2, x+y) \\ p_2: & A(x,y) & : y > 3 & \rightarrow & B(x)A(x/y, 0) \\ p_3: & B(x) & : x < 1 & \rightarrow & C \\ p_4: & B(x) & : x >= 1 & \rightarrow & B(x-1) \end{array} \tag{1.7}$$

As in the case of non-parametric L-systems, it is assumed that a module replaces itself if no matching production is found in the set P. The words obtained in the first few derivation steps are shown in Figure 1.34.

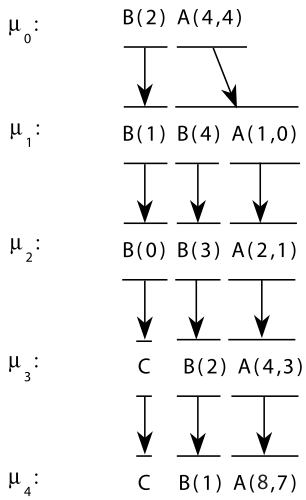


Figure 1.34: The initial sequence of strings generated by the parametric L-system specified in equation (1.7)

1.10.2 Parametric 2L-systems

Productions in parametric OL-systems are context-free, *i.e.,* applicable regardless of the context in which the predecessor appears. A context-sensitive extension is necessary to model information exchange between neighboring modules. In the parametric case, each component of the production predecessor (the left context, the strict predecessor and the right context) is a parametric word with letters from the alphabet V and formal parameters from the set Σ. Any formal parameters may appear in the condition and the production successor.

Example

A sample context-sensitive production is given below:

$$A(x) < B(y) > C(z) : x + y + z > 10 \rightarrow E((x + y)/2)F((y + z)/2)$$

It can be applied to the module $B(5)$ that appears in a parametric word

$$\cdots A(4)B(5)C(6)\cdots \tag{1.8}$$

since the sequence of letters A, B, C in the production and in parametric word (1.8) are the same, the numbers of formal parameters and actual parameters coincide, and the condition $4 + 5 + 6 > 10$ is true. As a result of the production application, the module $B(5)$ will be replaced by a pair of modules $E(4.5)F(5.5)$. Naturally, the modules $A(4)$ and $C(6)$ will be replaced by other productions in the same derivation step.

Anabaena with heterocysts

Parametric 2L-systems provide a convenient tool for expressing developmental models that involve diffusion of substances throughout an organism. A good example is provided by an extended model of the pattern of cells observed in *Anabaena catenula* and other blue-green bacteria [99]. This model was proposed by de Koster and Lindenmayer [25].

```
#define CH 900  /* high concentration */
#define CT 0.4  /* concentration threshold */
#define ST 3.9  /* segment size threshold */
#include H      /* heterocyst shape specification */
#ignore f ~ H

ω  :  -(90)F(0,0,CH)F(4,1,CH)F(0,0,CH)
p1 :  F(s,t,c) :  t=1 & s>=6 →
                            F(s/3*2,2,c)f(1)F(s/3,1,c)
p2 :  F(s,t,c) :  t=2 & s>=6 →
                            F(s/3,2,c)f(1)F(s/3*2,1,c)
p3 :  F(h,i,k) < F(s,t,c) > F(o,p,r) :  s>ST|c>CT →
                            F(s+.1,t,c+0.25*(k+r-3*c))
p4 :  F(h,i,k) < F(s,t,c) > F(o,p,r) :  !(s>ST|c>CT) →
                            F(0,0,CH) ~ H(1)
p5 :  H(s) :  s<3 → H(s*1.1)
```

L-system 1.1: *Anabaena catenula*

Generally, the bacteria under consideration form a nonbranching filament consisting of two classes of cells: *vegetative cells* and *heterocysts*. Usually, the vegetative cells divide and produce two daughter vegetative cells. This mechanism is captured by the L-system specified in equation (1.1) and Figure 1.4 (page 5). However, in some cases the vegetative cells differentiate into heterocysts. Their distribution forms a well-defined pattern, characterized by a relatively constant number of vegetative cells separating consecutive heterocysts. How does the organism maintain constant spacing of heterocysts while growing? The model explains this phenomenon using a biologically well-motivated hypothesis that heterocyst distribution is regulated by nitrogen compounds produced by the heterocysts, transported from cell to cell across the filament, and decayed in the vegetative cells. If the compound's concentration in a young vegetative cell falls below a specific level, this cell differentiates into a heterocyst (L-system 1.1).

The #define statements assign values to numerical constants used in the L-system. The #include statement specifies the shape of a heterocyst (a disk) by referring to a library of predefined shapes (see Section 5.1). Cells are represented by modules $F(s,t,c)$, where s stands for cell length, t is cell type (0 - heterocyst, 1 and 2 - vegetative types[1]),

[1]The model of *Anabaena* introduced in Section 1.2 distinguished between four types of cells: a_r, b_r, a_l and b_l. Cells b do not divide and can be considered as young forms of the corresponding cells a. Thus, the vegetative type 1 considered here embraces cells a_r and b_r, while type 2 embraces a_l and b_l. The formal relationship between the four-cell and two-cell models is further discussed in Chapter 6.

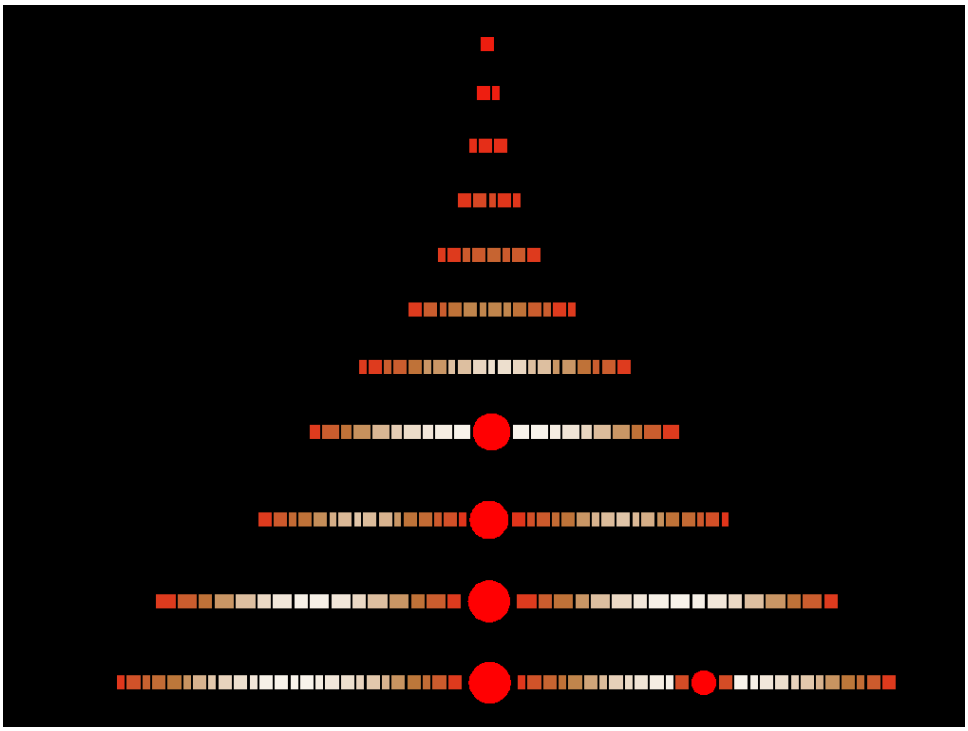


Figure 1.35: Development of *Anabaena catenula* with heterocysts, simulated using parametric L-system 1.1

and c represents the concentration of nitrogen compounds. Productions p_1 and p_2 describe division of the vegetative cells. They each create two daughter cells of unequal length. The difference between cells of type 1 and 2 lies in the ordering of the long and short daughter cells. Production p_3 captures the processes of transportation and decay of the nitrogen compounds. Their concentration c is related to the concentration in the neighboring cells k and r, and changes in each derivation step according to the formula

$$c' = c + 0.25(k + r - 3 * c).$$

Production p_4 describes differentiation of a vegetative cell into a heterocyst. The condition specifies that this process occurs when the cell length does not exceed the threshold value $ST = 3.9$ (which means that the cell is young enough), and the concentration of the nitrogen compounds falls below the threshold value $CT = 0.4$. Production p_5 describes the subsequent growth of the heterocyst.

Snapshots of the developmental sequence of *Anabaena* are given in Figure 1.35. The vegetative cells are shown as rectangles, colored according to the concentration of the nitrogen compounds (white means low concentration). The heterocysts are represented as red disks. The values of parameters CH, CT and ST were selected to provide the

correct distribution of the heterocysts, and correspond closely to the values reported in [25]. The mathematical model made it possible to estimate these parameters, although they are not directly observable.

1.10.3 Turtle interpretation of parametric words

If one or more parameters are associated with a symbol interpreted by the turtle, the value of the first parameter controls the turtle's state. If the symbol is not followed by any parameter, default values specified outside the L-system are used as in the non-parametric case. The basic set of symbols affected by the introduction of parameters is listed below.

$F(a)$ Move forward a step of length $a > 0$. The position of the turtle changes to (x', y', z'), where $x' = x + a\vec{H}_x$, $y' = y + a\vec{H}_y$ and $z' = z + a\vec{H}_z$. A line segment is drawn between points (x, y, z) and (x', y', z').

$f(a)$ Move forward a step of length a without drawing a line.

$+(a)$ Rotate around $\vec{U}$ by an angle of a degrees. If a is positive, the turtle is turned to the left and if a is negative, the turn is to the right.

$\&(a)$ Rotate around $\vec{L}$ by an angle of a degrees. If a is positive, the turtle is pitched down and if a is negative, the turtle is pitched up.

$/(a)$ Rotate around $\vec{H}$ by an angle of a degrees. If a is positive, the turtle is rolled to the right and if a is negative, it is rolled to the left.

It should be noted that symbols $+$, $\&$, $\wedge$, and $/$ are used both as letters of the alphabet V and as operators in logical and arithmetic expressions. Their meaning depends on the context.

Row of trees

The following examples illustrate the operation of parametric L-systems with turtle interpretation. The first L-system is a coding of a Koch construction generating a variant of the snowflake curve (Figure 1.1 on page 2). The initiator (production predecessor) is the hypotenuse AB of a right triangle ABC (Figure 1.36). The first and the fourth edge of the generator subdivide AB into segments AD and DB, while the remaining two edges traverse the altitude CD in opposite directions. From elementary geometry it follows that the lengths of these segments satisfy the equations

$$q = c - p \qquad \text{and} \qquad h = \sqrt{pq}.$$

The edges of the generator can be associated with four triangles that are similar to ABC and tile it without gaps. According to the relationship between curve construction by edge rewriting and planar tilings

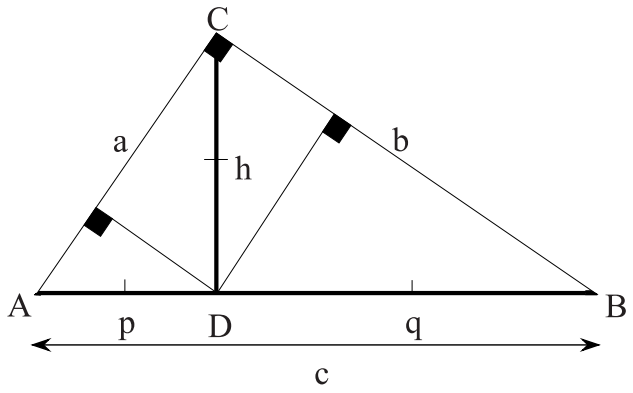


Figure 1.36: Construction of the generator for a "row of trees." The edges are associated with triangles indicated by ticks.

(Section 1.4.1), the generated curve will approximately fill the triangle ABC. The corresponding L-system is given below:

$$\begin{array}{ll} \#\text{define} & c \;\; 1 \\ \#\text{define} & p \;\; 0.3 \\ \#\text{define} & q \;\; c - p \\ \#\text{define} & h \;\; (p * q) \wedge 0.5 \end{array}$$

$$\begin{array}{ll} \omega \; : & F(1) \\ p_1 : & F(x) \rightarrow F(x * p) + F(x * h) - -F(x * h) + F(x * q) \end{array}$$

The resulting curve is shown in Figure 1.37a. In order to better visualize its structure, the angle increment has been set to 86° instead of 90°. The curve fills different areas with unequal density. This results from the fact that all edges, whether long or short, are replaced by the generator in every derivation step. A modified curve that fills the underlying triangle in a more uniform way is shown in Figure 1.37b. It was obtained by delaying the rewriting of shorter segments with respect to the longer ones, as specified by the following L-system.

$$\begin{array}{lll} \omega \; : & F(1,0) & \\ p_1 : & F(x,t) : t = 0 \;\; \rightarrow & F(x * p, 2) + F(x * h, 1) - \\ & & -F(x * h, 1) + F(x * q, 0) \\ p_2 : & F(x,t) : t > 0 \;\; \rightarrow & F(x, t-1) \end{array}$$

Branching structure

The next example makes use of node rewriting (Section 1.4.2). The construction recursively subdivides a rectangular tile $ABCD$ into two tiles, $AEFD$ and $BCFE$, similar to $ABCD$ (Figure 1.38). The lengths of the edges form the proportion

$$\frac{a}{b} = \frac{b}{\frac{1}{2}a},$$

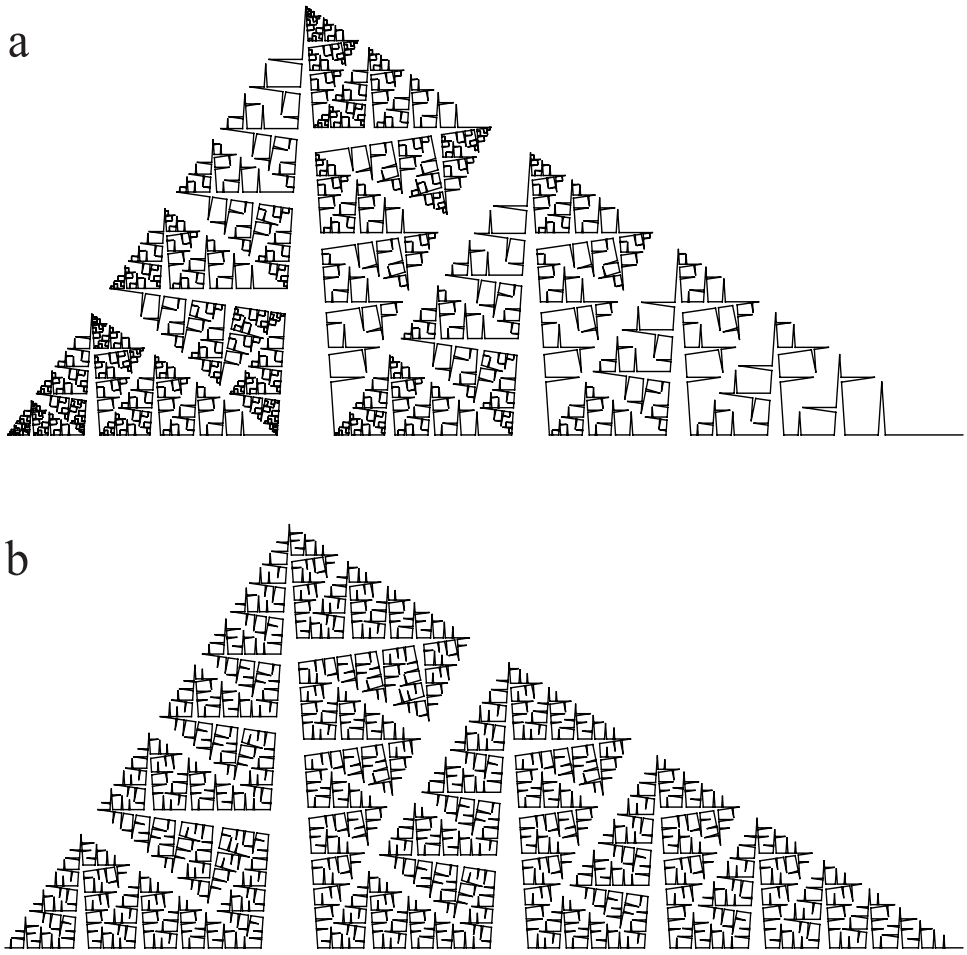


Figure 1.37: Two curves suggesting a "row of trees." Curve (b) is from [95, page 57].

which implies that $b = a/\sqrt{2}$. Each tile is associated with a single-point frame lying in the tile center. The tiles are connected by a branching line specified by the following L-system:

$$\begin{array}{lll} \#\text{define} & R & 1.456 \\ \omega & : & A(1) \\ p_1 & : & A(s) \rightarrow F(s)[+A(s/R)][-A(s/R)] \end{array} \tag{1.9}$$

The ratio of branch sizes R slightly exceeds the theoretical value of $\sqrt{2}$. As a result, the branching structure shown in Figure 1.39 is self-avoiding. The angle increment was set arbitrarily to $\delta = 85°$.

The L-system in equation (1.9) operates by appending segments of decreasing length to the structures obtained in previous derivation steps. Once a segment has been incorporated, its length does not change. A structure with identical proportions can be obtained by

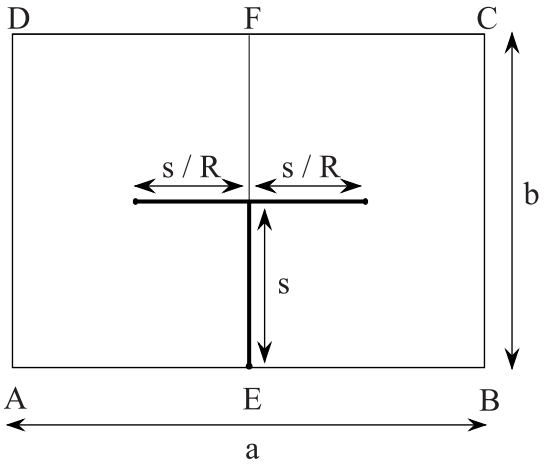


Figure 1.38: Tiling associated with a space-filling branching pattern

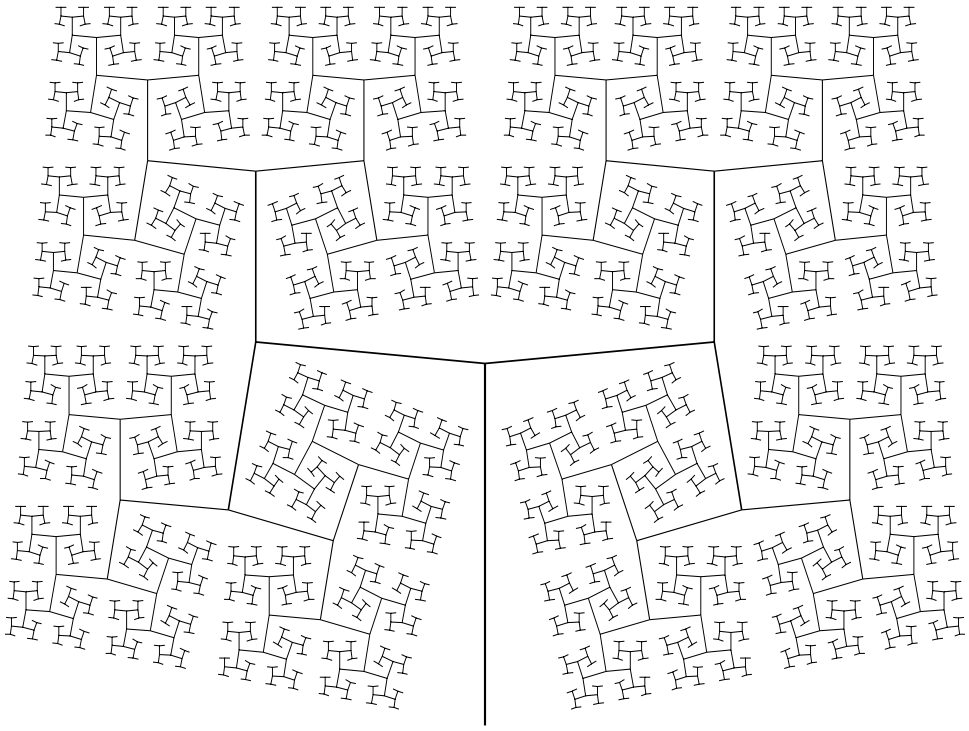


Figure 1.39: A branching pattern generated by the L-system specified in equation (1.9)

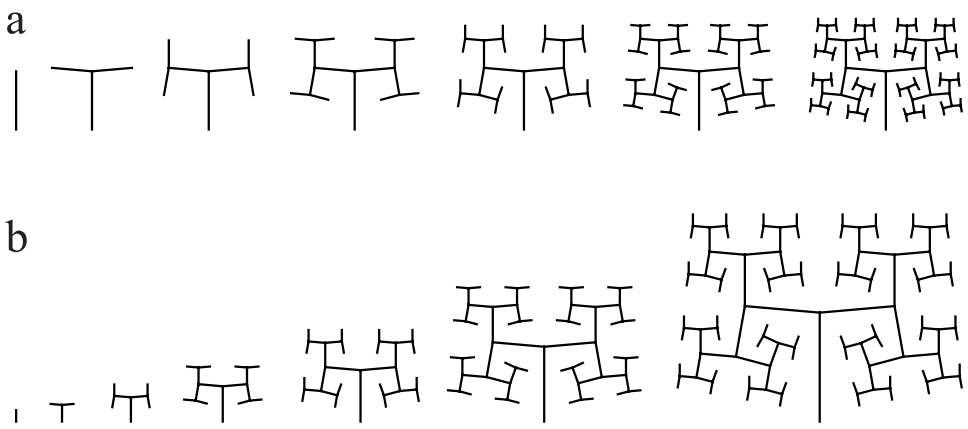


Figure 1.40: Initial sequences of figures generated by the L-systems specified in equations (1.9) and (1.10)

appending segments of constant length and increasing the lengths of previously created segments by constant R in each derivation step. The corresponding L-system is given below.

$$\begin{array}{lll} \omega: & A & \\ p_1: & A & \rightarrow F(1)[+A][-A] \\ p_2: & F(s) & \rightarrow F(s * R) \end{array} \tag{1.10}$$

The initial sequence of structures obtained by both L-systems are compared in Figure 1.40. Sequence (a) emphasizes the fractal character of the resulting structure. Sequence (b) suggests the growth of a tree. The next two chapters show that this is not a mere coincidence, and the L-system specified in equation (1.10) is a simple, but in principle correct, developmental model of a *sympodial* branching pattern found in many herbaceous plants and trees.