Chapter 3

Lindenmayer Systems

This chapter describes the classes of Lindenmayer Systems. Lindenmayer systems (L-systems) were conceived as a mathematical theory of plant development. 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.

3.1 Rewriting systems

The main concept of L-systems is rewriting. The 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* [27]. The classic example of graphical object defined in terms of rewriting rules is the *snowflake curve* in Figure 3.1, proposed in 1905 by von Koch [27]. Mandelbrot restates this construction as follow:

One begins with *two shapes*, an *initiator* and a *generator*. The latter is an oriented broken line make 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.

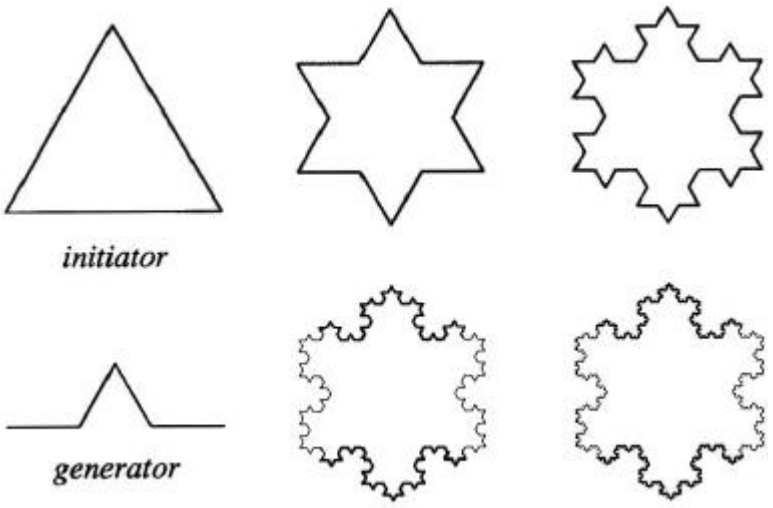


Figure 3.1: Construction of the snowflake curve.

In 1968 a biologist, Aristid Lindenmayer, introduced a new type of string-rewriting mechanism, subsequently termed L-systems [27]. The essential difference between Chomsky grammars and L-systems lies in 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 multi-cellular organisms, where many divisions may occur at the same time. Parallel production application has an essential impact on the formal properties of rewriting systems.

3.2 Deterministic and Context-Free L-systems

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

Consider strings built of two symbols *a* and *b*, which many occur many times in a string. Each symbol 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. Assuming that it consists of a single letter *b*. In the first derivation step (the first step of rewriting, n=1), the axiom *b* is replaced by a using production

$b \rightarrow a$. In the second step (n=2) 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 as illustrated in Figure 3.2.

$n = 5$
$\omega : b$
$p_1 : a \rightarrow ab$
$p_2 : b \rightarrow a$

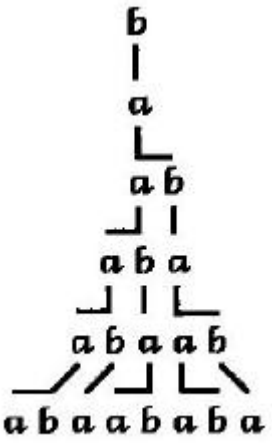


Figure 3.2: Example of a derivation in a DOL-system.

Formal definitions describing DOL-systems and their operation are given below. For more details see [27]. Let V denote an alphabet, V^* the set of all 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 productions, 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 $a \rightarrow \chi$.

Let $\mu = a_1 \ldots a_m$ be an arbitrary word over V. The word $\nu = \chi_1 \ldots \chi_m \in V^*$ is *directly generated* by μ, noted $\mu \rightarrow \nu$, if and only if $a_i \rightarrow \chi_1$ for all $i = 1,\ldots,m$.

A word $\boldsymbol{n}$ is generated by *G* in a derivation of *length n* if there exists a *developmental sequence* of words $m_0, m_1, \ldots, m_n$ such that $m_0 = \boldsymbol{w}$, $m_n = \boldsymbol{n}$ and $m_0 \rightarrow m_1 \rightarrow \ldots \rightarrow m_n$.

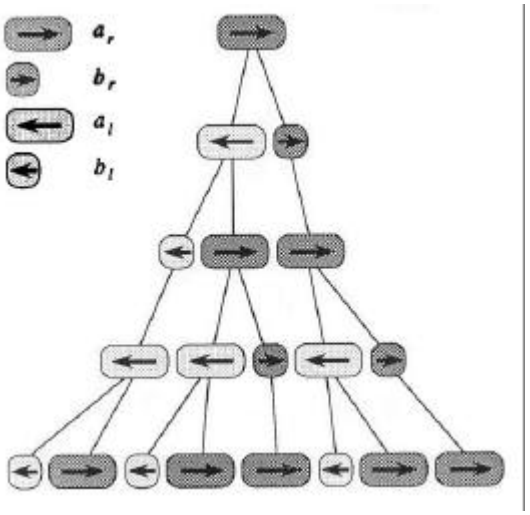


Figure 3.3: Development of a filament (*Anabaena catenula*) simulated using a DOL-system.

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 multi-cellular filament such as that found in the blue-green bacteria *Anabaena catenula* and various algae [27]. 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:

$n = 4$
$\boldsymbol{w} : a_r$
$p_1 : a_r \rightarrow a_l b_r$
$p_2 : a_l \rightarrow b_l a_r$
$p_3 : b_r \rightarrow a_r$
$p_4 : b_l \rightarrow a_l$

Starting from a single cell $\boldsymbol{a_r}$ (the axiom), the following sequence of words is generated:

a_r

$a_l\ b_r$

$b_l\ a_r\ a_r$

$a_l\ a_l\ b_r\ a_l\ b_r$

$b_l\ a_r\ b_l\ a_r\ a_r\ b_l\ a_r\ a_r$

. . .

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 3.3. Note that due to the discrete nature of L-systems, the continuous growth of cells between subdivisions is not captured by this model.

3.3 Turtle interpretation of strings

In order to model the higher plants, a more sophisticated graphical interpretation of L-systems in needed. The first results in this direction were published in 1974 by Frijters and Lindenmayer, and Hogeweg and Hesper. 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, who demonstrated the potential of L-systems for realistic image synthesis.

The basic idea of turtle interpretation is given below. A *state* of the *turtle* is defined as a triplet $(x,y,\boldsymbol{a})$, where the Cartesian coordinates (x,y) represent the turtle's *position*, and the angle $\boldsymbol{a}$, call the *heading*, is interpreted as the direction in which the turtle is facing. Given the *step size d* and the *angle increment* $\boldsymbol{d}$, the turtle can respond to commands represented by the following symbols in Table 3.1.

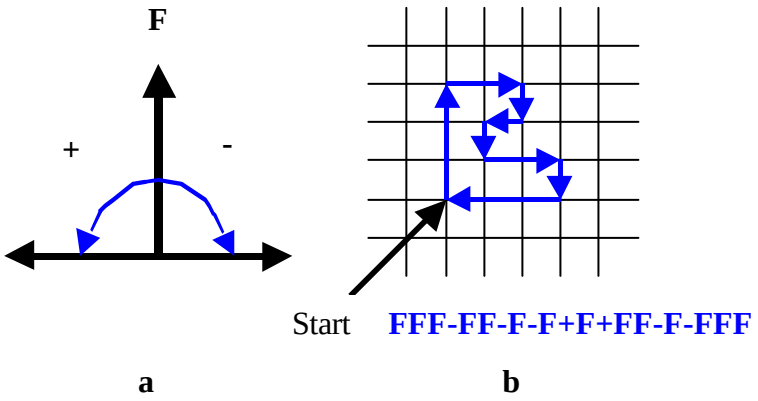


Figure 3.4: Turtle Interpretation of String.

Table 3.1: The two-dimensional turtle interpretation.

Symbols	Meaning
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 position orientation of angles is counter-clockwise.
-	Turn right by angle δ. The next state of the turtle is $(x,y,\alpha-\delta)$. The position orientation of angles is clockwise.

Given a string ω, the initial state of the turtle (x_0,y_0,α_0) and fixed parameters d and δ, the *turtle interpretation* of ω is the figure (set of lines) drawn by the turtle in response to the string ω in Figure 3.4. Specifically, this method can be applied to interpret strings which are generated by L-systems. For example, Figure 3.5 presents four approximations of the *quadratic Koch island*. These figures were obtained by interpreting strings generated by the following L-system:

ω: *F-F-F-F*

p : $F \rightarrow$ *F-F+F+FF-F-F+F*

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

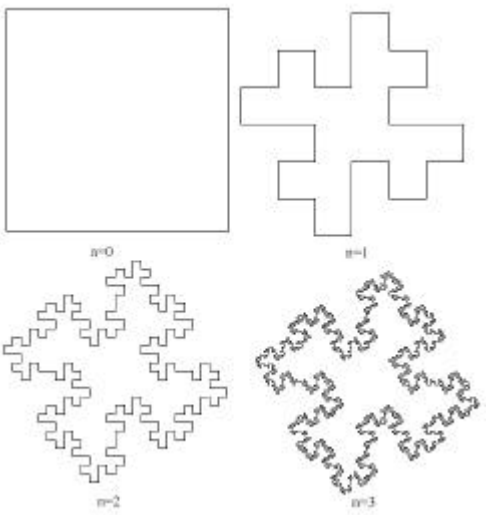


Figure 3.5: Generating a quadratic Koch island.

3.4 Modeling in three dimensions

Turtle interpretation of L-systems can be extended to three dimensions following the ideas of Abelson and diSeassa [27]. The key concept is to represent the current *orientation* of the turtle in space by three vectors

$$\vec{X}, \vec{Y}, \vec{Z},$$

indicating the direction in X-axis, Y-axis, Z-axis, respectively. These vectors have orthorgonal unit vector that satisfy the equation

$$\vec{X} \times \vec{Y} = \vec{Z}$$

Rotations of the turtle are expressed by the equation

$$[X' \; Y' \; Z'] = [X \; Y \; Z] \; R$$

where R is a 3x3 rotation matrix. The rotations by angle *è* about vectors X, Y, Z are represented by the following matrices:

$$R_X(q) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & Cos(q) & -Sin(q) \\ 0 & Sin(q) & Cos(q) \end{bmatrix}$$

$$R_Y(q) = \begin{bmatrix} Cos(q) & 0 & -Sin(q) \\ 0 & 1 & 0 \\ Sin(q) & 0 & Cos(q) \end{bmatrix}$$

$$R_z(q) = \begin{bmatrix} Cos(q) & Sin(q) & 0 \\ -Sin(q) & Cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The following symbols control turtle orientation in three-dimensional space as Figure 3.6:

Table 3.2: The symbol of three-dimensional turtle interpretation.

Symbols	Meaning
+	Roll counter-clockwise to positive Z-axis by angle δ_z, using rotation matrix $R_z(\delta_z)$.
-	Roll clockwise to positive Z-axis by angle δ_z, using rotation matrix $R_z(-\delta_z)$.
&	Roll counter-clockwise to positive Y-axis by angle δ_y, using rotation matrix $R_y(\delta_y)$.
^	Roll clockwise to positive Y-axis by angle δ_y, using rotation matrix $R_y(-\delta_y)$.
\	Roll counter-clockwise to positive X-axis by angle δ_x, using rotation matrix $R_x(\delta_x)$.
/	Roll clockwise to positive X-axis by angle δ_x, using rotation matrix $R_x(-\delta_x)$.
|	Turn around, using rotation matrix $R_y(180)$.

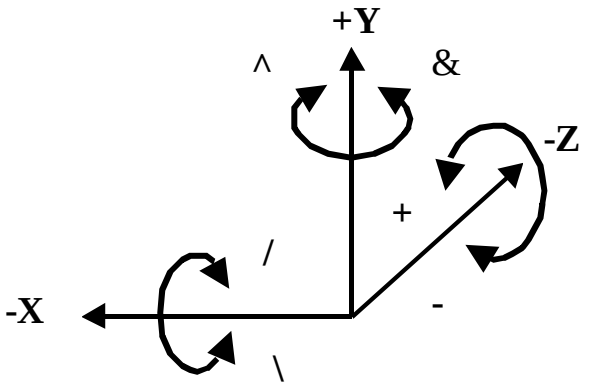


Figure 3.6: Controlling the turtle in three dimensions.

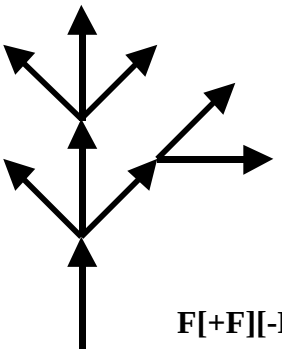
3.5 Bracketed OL-systems

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 tree can be represented using *strings with brackets* [27]. 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 [27] are described below.

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

Table 3.3: The bracketed symbols.

Symbols	Meaning
[	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.



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

Figure 3.7: Bracketed string representation of an axial tree.

An example of an axial tree and its string representation are shown in Figure 3.7. Derivations in bracketed OL-systems proceed as in OL-systems with out brackets. The brackets replace themselves. Examples of two-dimensional branching structures generated by bracketed OL-systems are shown in Figure 3.8.

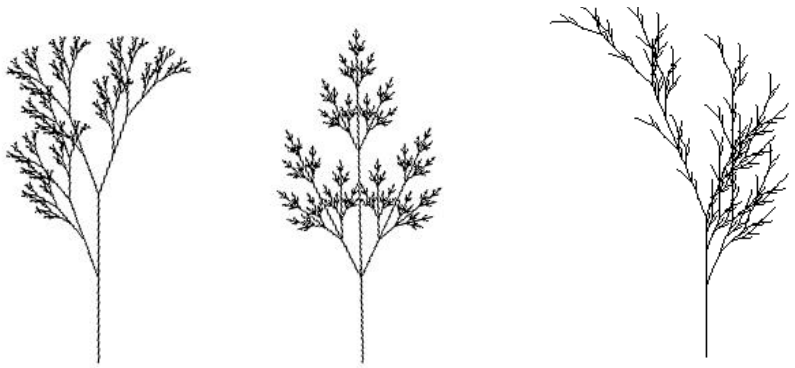
Figure 3.8 illustrates some examples of a three-dimensional bush-like structure generated by a bracketed L-system [27]. Production p_1 creates three new branches from an apex of the old branch. A branch consists of an edge *F* forming the initial internode, a leaf *L* and an apex *A* (which will subsequently create three new branches). Productions p_2 and p_3 specify internode growth. In subsequent derivation steps, the internode gets longer and acquires new leaves. This violates a biological rule of *subapical growth*, 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 }. The symbols ! and ’ are used to decrement the diameter of segments and increment the current index to the color table, respectively.

Another example of a three-dimensional plat is shown in Figure 3.9. The L-systems can be described and analyzed in a similar manner.

(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 3.8: Examples of plant-like structures generated by bracketed OL-systems.

n=7,δ=22.7
ω=A
p1 : A → [&FL!A]/////'[&FL!A]///////'[&FL!A]
p2 : F → S/////F
p3 : S → FL
p4 : L → ['''^^{-f+f+f-|-f+f+f}]

Figure 3.9: A three-dimensional bush-like structure.

3.6 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.

A *stochastic OL-system* is an ordered quadruplet G_p = <V,ω,*P*,p>. The alphabet *V*, the axiom w and the set of productions *P* are defined as in an OL-system.

Function $\pi : P \rightarrow (0,1]$, 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.

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

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

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

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 3.9. Note that these structures look like different specimens of the same plant species.

3.7 Edge and Node rewriting

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-systems 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 [27], the 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 [27]. In the case of edge rewriting, productions substitute figures for polygon edges, while 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.

Edge rewriting identifies edges as specific types of edges, which the turtle does not interpret, but the different types of edges affect rewriting rules.

Example:

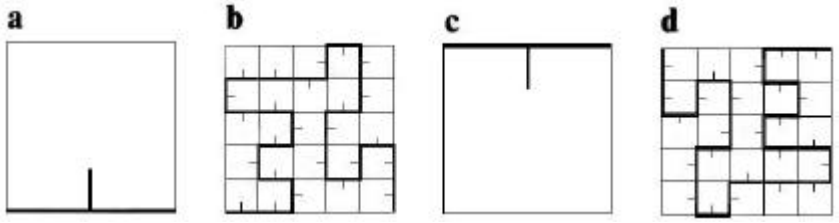


Figure 3.10: Construction of the E-curve on the square grid.
a) F_l **b)** $F_l \rightarrow F_lF_l + F_r + F_r - F_l - F_l + F_r + F_rF_l - F_r - F_lF_lF_r + F_l - F_r - F_lF_l - F_r + F_lF_r + F_r + F_l - F_l - F_rF_r +$
c) F_r **d)** $F_r \rightarrow - F_lF_l + F_r + F_r - F_l - F_lF_r - F_l + F_rF_r + F_l + F_r - F_lF_rF_r + F_l + F_rF_l - F_l - F_r + F_r + F_l - F_l - F_rF_r$

Node rewriting substitutes new polygons for nodes of the predecessor curve.
Example:

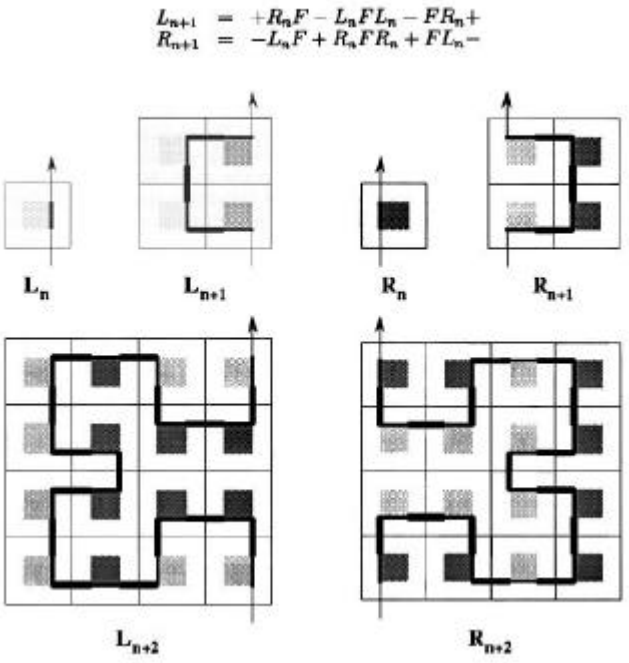


Figure 3.11: Recursive construction of the Hilbert curve in term of replacement.

The curves are generated by either edge or node rewriting method. 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.

3.8 Parametric L-systems

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.

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* $\hat{A}$. A module with letter $A \in V$ and parameters $a1,a2,\ldots,an \in \hat{A}$ is denoted by $A(a1,a2,\ldots,an)$. Every module belongs to the set $M = V \times \hat{A}^*$, where $\hat{A}^*$ 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 $M^* = (V \times \hat{A}^*)^*$ and $M^+ = (V \times \hat{A}^*)^+$, respectively.

The real-valued *actual* parameters appearing in the words correspond with *formal* parameters used in the specification of L-systems 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 ^, the relational operators <, >, =; the logical operator !,&,| (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 $C(\Sigma)$ and $\mathcal{E}(\Sigma)$.

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

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

The symbols: and → 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 $B(t+1)CD(t\text{^}0.5,t-2)$ is written as

$$A(t) : t > 5 \rightarrow B(t+1)CD(t\text{^}0.5,t-2). \qquad (3.1)$$

A production *matches* a modules 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 (3.1) 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 $a \rightarrow \chi$. Given a parametric word $m = a1a2\ldots am$, we say that the word $\eta = \chi1\chi2\ldots\chi m$ is *directly derived* from (or *generated* by) m and write $m \rightarrow \eta$ if and only if $a_i \rightarrow \chi_i$ for all $i = 1,2,..,m$. A parametric word η is generated by G in a *derivation of length n* if there exists a sequence of words $m_0,m_1,\ldots,m_n$ such that $m_0 = w$, $m_n = \eta$ and $m_0 \rightarrow m_1 \rightarrow \ldots \rightarrow m_n$.

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

$$
\begin{array}{lll}
w : B(2)A(4,4) & & \\
p_1 : A(x,y) & : y <= 3 & \rightarrow A(x * 2,\ x + y) \\
p_2 : A(x,y) & : y > 3 & \rightarrow B(x)A(x/y,\ 0) \\
p_3 : B(x) & : x < 1 & \rightarrow C \\
p_4 : B(x) & : x >= 1 & \rightarrow B(x - 1)
\end{array}
\qquad (3.2)
$$

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 show in Figure 3.10

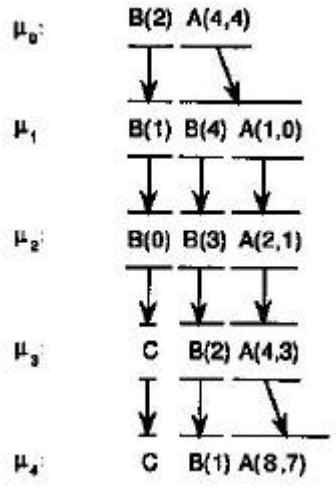


Figure 3.12: The initial sequence of strings generated by the parametric L-system specified in prototype (3.2).

3.9 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 symbols are 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.

Table 3.4 The symbol of parametric words.

Parametric words	Meaning
$F(a)$	Move forward a step of length $a > 0$. The position of the turtle changes to (x', y', z'), where $X' = x + aX_x$, $Y' = y + aX_y$, $Z' = z + aX_z$. A line segment is drawn between points (x, y, z) and (x', y', z').
$f(a)$	Move forward a step of length $\boldsymbol{a}$ without drawing a line.
$+(a)$	Rotate about Z-axis by an angle of a degrees. If $\boldsymbol{a}$ is positive, the turtle is turned to the left and if $\boldsymbol{a}$ is negative, the turn is to the right.
$\&(a)$	Rotate about Y-axis by angle of $\boldsymbol{a}$ degrees. If $\boldsymbol{a}$ is positive, the turtle is pitched down and if $\boldsymbol{a}$ is negative, the turtle is pitched up.
$/(a)$	Rotate about X-axis by an angle of $\boldsymbol{a}$ degrees. If $\boldsymbol{a}$ is positive, the turtle is rolled to the right and if $\boldsymbol{a}$ is negative, it is rolled to the left.

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