Visual models of plant development

Przemyslaw Prusinkiewicz†, Mark Hammel†,
Jim Hanan‡, and Radomír Měch†

†Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4
e-mail: pwp|hammel|mech@cpsc.ucalgary.ca

‡CSIRO - Cooperative Research Centre for Tropical
Pest Management
Brisbane, Australia
e-mail: jim@ctpm.uq.oz.au

From G. Rozenberg and A. Salomaa, editors,
Handbook of formal languages
Springer-Verlag, 1996. To appear.

Visual models of plant development

Przemyslaw Prusinkiewicz[1], Mark Hammel[1],
Jim Hanan[2], and Radomír Měch[1]

[1] Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4
e-mail: pwp|hammel|mech@cpsc.ucalgary.ca
[2] CSIRO - Cooperative Research Centre for Tropical Pest Management
Brisbane, Australia
e-mail: jim@ctpm.uq.oz.au

Summary. In these notes we survey applications of L-systems to the modeling of plants, with an emphasis on the results obtained since the comprehensive presentation of this area in *The Algorithmic Beauty of Plants* [99]. The new developments include:

- extensions to the L-system formalism that increase its expressive power as needed for practical biological applications,
- introduction of programming constructs that enhance the use of L-systems as a language for describing developmental algorithms and as input for simulation programs, and
- new biological applications of L-systems.

Keywords: L-system, fractal, plant, modeling, simulation, realistic image synthesis, emergence, artificial life.

There is nothing so practical as a good theory.
Immanuel Kant (1724–1804)

1. Introduction

In 1968, Aristid Lindenmayer introduced a formalism for modeling and simulating the development of multicellular organisms [67], subsequently named L-systems. This formalism was closely related to the theory of automata and formal languages, and immediately attracted the interest of computer scientists [114]. The vigorous development of the theory of L-systems [46, 113, 116] was followed by its application to the modeling of plants (for example, [28, 29, 30, 55, 74]). A series of position and survey papers by Lindenmayer addressed methodological aspects of modeling using L-systems and their role in biology [66, 70, 71, 72, 73, 75]. Concurrent advances in computer graphics made it possible to visualize modeled structures in forms ranging from schematic diagrams [30, 49] to realistic three-dimensional renderings of abstract branching structures [91, 120, 121] and real plants [101]. Visualizations have also revealed intriguing relationships between L-systems and

fractals [19, 20, 90, 100, 122]. Graphical applications of L-systems devised until 1990 were comprehensively presented in books [94] and [99].

In the present chapter we focus on recent results pertinent to the modeling, simulation, and visualization of plant development using L-systems. These include, in particular:

- extensions to the L-system formalism that increase its expressive power as needed for practical biological applications,
- introduction of programming constructs that enhance the use of L-systems as a language for describing developmental algorithms and as input for simulation programs, and
- new biological applications of L-systems,

The organization of this chapter mimics the general pattern of theory construction in the natural sciences, where observed facts are distilled into a mathematical abstraction, and the resulting predictions are compared with reality to validate the theory [58]. First, we present modular plant architecture as our domain of interest and show that the essential aspects of development at the modular level can be viewed as rewriting processes (Section 2.). In order to formally describe the architecture of plants, we introduce, after Lindenmayer, the bracketed string notation to express the topology of branching structures, and we extend this notation with symbols and constructs based on turtle geometry to capture the shape (Section 3.). We then present L-systems as a rewriting mechanism that simulates developmental processes by operating on strings, and use biologically motivated examples to illustrate the basic definitions (Section 4.). More involved constructs and extensions of L-systems are introduced to simulate fragmentation and the loss of modules (Section 6.), different aspects of information flow within the modeled structure (Section 7.), and interaction between plants and their environment (Section 8.). Finally, we characterize the role of L-systems in the current practice of biological modeling, and point out selected open problems (Section 9.).

Since the modeling and visualization of plant development is an interdisciplinary area of research, we have attempted to make the results legible to readers in various disciplines. Consequently, we have emphasized the motivations behind the theory, and avoided specialized notions of formal languages, computer graphics, and biology.

2. Developmental models of plant architecture

2.1 The modular structure of plants

Mathematical models in botany correspond to various levels of plant organization (Figure 2.1). In this paper, we focus on the level of entire plants. We regard a plant as a spatial configuration of discrete constructional units or

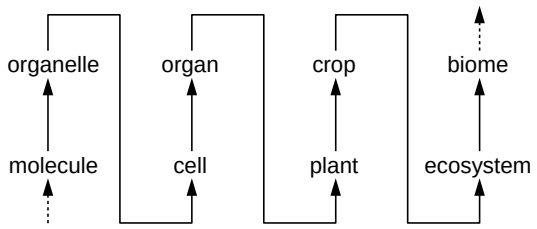


Fig. 2.1. A hierarchy of levels of plant organization. One objective of modeling is to predict and understand phenomena taking place at a given level on the basis of models operating at lower levels. Adapted from [124, 131].

modules, which develop over time. Typically, modules represent repeating basic structural components of a plant, such as flowers, leaves, and internodes, or groupings of these components, such as metamers (single internodes with an associated leaf and lateral bud) and branches (Figure 2.2) [5, 43, 128]). (A different meaning of the term "module" is also found in the literature [4, 38, 110].) The goal is to describe the development of a plant, and in particular the emergence of plant shape, as the integration of the development and functioning of individual modules.

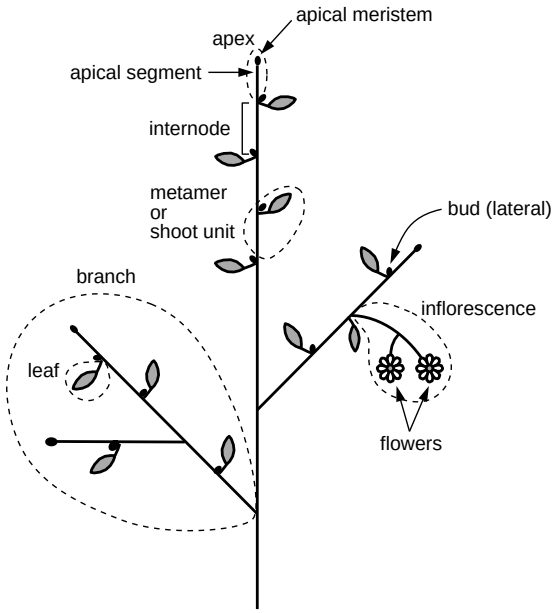


Fig. 2.2. Selected modules and groups of modules (encircled with dashed lines) used to describe plant structure.

2.2 Plant development as a rewriting process

The essence of plant development can be described by a *rewriting system* that repetitively replaces individual *parent*, *mother*, or *ancestor* modules by configurations of *child*, *daughter*, or *descendant* modules.

Assuming that all modules belong to a finite set of *module types*, the behavior of an arbitrarily large configuration of modules can be specified using a finite set of *rewriting rules* or *productions*. A production specifies how to replace a single *predecessor* module by a configuration of zero, one, or more *successor* modules. A simple example of this process is shown in Figure 2.3. An *occurrence map* φ transforms the predecessor of the production to the mother modules; the same map is then applied to the successor in order to determine the child modules [103].

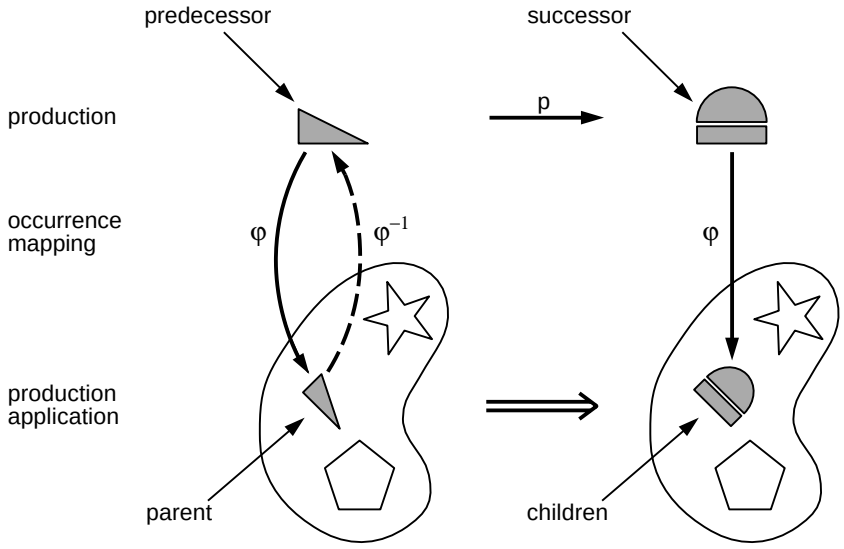


Fig. 2.3. Illustration of the concept of rewriting applied to modules with geometric interpretation. A parent module is replaced by child modules in a sequence of transformations $\varphi^{-1}p\varphi$.

The replacement of modules in a structure may become difficult when there are significant differences in the geometry of a parent and its children. Several possibilities are illustrated in Figure 2.4. In case (a), modules located at the extremities of a branching structure are replaced without affecting the remainder of the structure. The production applications may be implemented using the mechanism just discussed (Figure 2.3). In case (b), the production that replaces internodes divides the rewritten structure into a lower part (below the internode) and an upper part. The position of the upper part is adjusted to accommodate the insertion of the child modules, but the shape

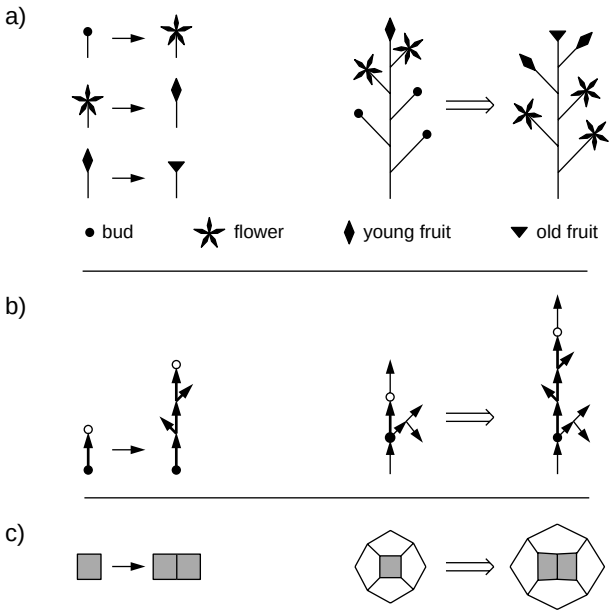


Fig. 2.4. Examples of production specification and application: (a) development of a flower, (b) development of a branch, and (c) cell division.

and size of both the lower and upper part are not changed, and the children remain defined entirely by the production. Finally, in case (c), the rewritten structure is represented by a graph with cycles. The size and shape of the production successor do not exactly match the size and shape of the predecessor, thus the geometry of the successor, the embedding structure, or both must be adjusted to accommodate the successor. The last case is the most complex, since the application of a local rewriting rule may lead to a global change of the structure's geometry. Developmental models of cellular layers operating in this manner have been presented in [16, 17, 26, 99]. In this paper we will limit our interest to branching structures. This limitation offers a useful compromise between breadth and depth in the resulting theory of development.

The differences between cases (a) and (b) are further discussed below. Case (a) is similar to the basic method of fractal generation using Koch construction, described by Mandelbrot as follows [81, page 39]:

> 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.

Mandelbrot introduced many extensions to this basic concept, including generators with lines of unequal length [81, pages 56–57] and with branching topology [81, pages 71–73]. All these variants share one fundamental characteristic, namely that the position, orientation, and scale of the interval being replaced determine the position, orientation, and scale of the replacement (a copy of the generator). In plants, however, the position and orientation of each module is determined by the chain of modules beginning at the base of the structure and extending to the module under consideration. For example, when the internodes of a plant develop (as is the case in Figure 2.4b), the subtended segment is moved upwards in response. Similarly, when the internodes bend, the subtended branches do not become disconnected as implied by the Koch construction (Figure 2.5a), but are rotated and displaced

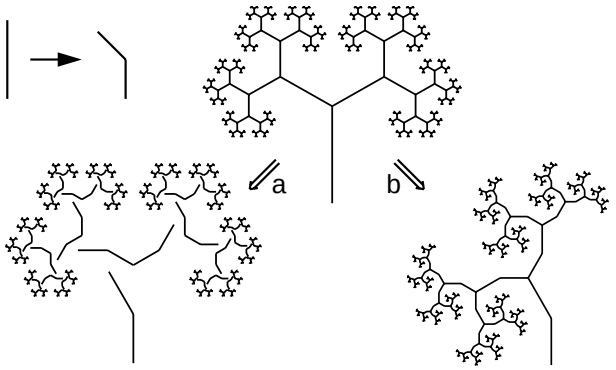


Fig. 2.5. A comparison of the Koch construction (a) with a rewriting system preserving the branching topology of the modeled structure (b). The same production is applied in both cases, but the rules for incorporating the successor into the structure are different.

to maintain the connectivity of the structure (Figure 2.5b). The bracketed string notation introduced by Lindenmayer [67, 68] inherently maintains the branching topology of the modeled structures while their component modules are rewritten. We describe it in detail in the next section.

3. Formal description of branching structures

3.1 Axial trees and bracketed strings

In order to consider plant architecture at an abstract level, we use the notion of an axial tree (Figure 3.1) which complements the graph-theoretic notion of a rooted tree [89] with the botanically motivated notion of branch axis [99, 101]. A *rooted tree* has edges that are labelled and directed, and form paths from a specific node called the *base* to the *terminal nodes*. In the biological

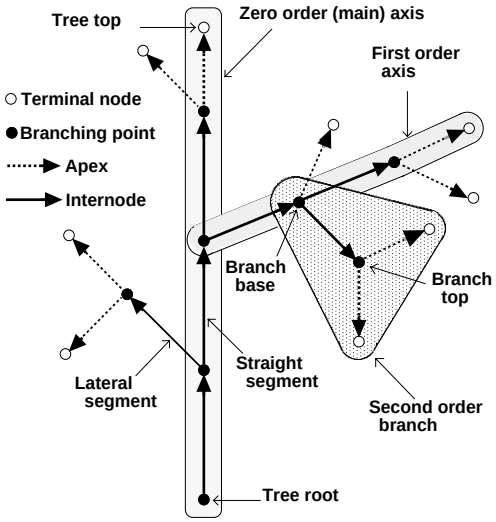


Fig. 3.1. An axial tree. From [101].

context, these edges are referred to as *branch segments*. For each nonterminal node, we distinguish between the *subtending* segment (closer to the tree root) and the *subtended* segments (further away). A segment followed by at least one more segment in some path is called an *internode*. A terminal segment (with no following edges) is called an *apex*.

An *axial tree* is a special type of rooted tree, in which we distinguish at most one subtended *straight* segment at each of the nodes. 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 base of the entire plant has order zero. An axis originating as a lateral segment of a parent axis of order n has order $n+1$. The order of a branch is equal to the order of its lowest-order axis. The terminal node of this axis is called the branch *top*.

3.2 The bracketed string notation

To represent axial trees, Lindenmayer introduced a *bracketed string* notation [67, 68], which we present here according to [98]. An axial tree with edge labels from alphabet V is represented by a word (string of *symbols* or *letters*) w over alphabet $V_E = V \cup \{[,]\}$:

$$w = x_1[\alpha_1]x_2[\alpha_2]\dots x_n[\alpha_n]x_{n+1}. \tag{3.1}$$

It is assumed that the subwords $x_1, x_2, \dots, x_{n+1} \in V^*$ do not contain brackets, and the subwords $\alpha_1, \alpha_2, \dots, \alpha_n \in V_E^*$ are well nested. The word $x_1x_2\dots x_nx_{n+1}$ represents the main (i.e., zero-order) axis of w, with internodes $x_1, x_2, \dots, x_n$ and apex x_{n+1}. The words $\alpha_1, \alpha_2, \dots, \alpha_n$ represent the first-order branches of w. Each branch α_i can be decomposed in a manner similar to w, yielding a first-order axis and, possibly, second-order branches. This decomposition can be carried out recursively. It is known that the decomposition of a well-nested word w is unique, thus all terms introduced above are unambiguous [57].

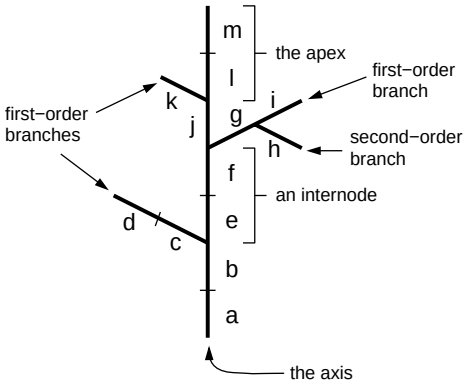


Fig. 3.2. Example of a branching structure. From [98].

For example, the axial tree shown in Figure 3.2 is represented by the string:

$$w = \overbrace{ab}^{x_1}\underbrace{[cd]}_{[\alpha_1]}\overbrace{ef}^{x_2}\underbrace{[g[h]i]}_{[\alpha_2]}\overbrace{j}^{x_3}\underbrace{[k]}_{[\alpha_3]}\overbrace{lm}^{x_4}. \tag{3.2}$$

The subword $abefjlm$ is the main axis of w; $x_1 = ab$, $x_2 = ef$ and $x_3 = j$ are the internodes, and $x_4 = lm$ is the apex. The subwords $\alpha_1 = cd$, $\alpha_2 = g[h]i$ and $\alpha_3 = k$ denote the lateral branches. α_1 and α_3 have only apices, whereas α_2 has an internode g, an apex i, and a (second-order) lateral branch h.

It is often useful to characterize plant modules (and the relationships between them) in more detail than is possible or practical using the labels

alone. This can be accomplished be associating one or more numerical parameters with the bracketed string symbols. A symbol (letter) with associated parameters is called a *parametric letter*, and a string of parametric letters is called a *parametric word*. A method for expressing the geometry of branching structures using parametric words is discussed next.

3.3 The turtle interpretation of bracketed strings

Bracketed strings were introduced to represent branching structure topology (connections between the modules). To express the form manifested by such properties as the orientation of branches and the length of internodes, the strings must be assigned a geometric interpretation. In simple cases, geometric information may be *external* to the string. For example, rules for drawing two-dimensional tree-like structures may state that the branches are issued at a constant angle in alternating directions, to the left and right, and all segments have the same length [49]. The branching angles might also vary as a function of branch order [30]. Unfortunately, such simple rules, operating uniformly on the entire string, are not sufficiently flexible to express the variety of branching forms found in nature. Consequently, more versatile techniques have been developed, based on explicit incorporation of geometric information into the strings. Selected symbols, which may appear with or without associated parameters, are reserved to represent geometric properties of the described structure. The one-to-one correspondence between the string symbols and the modules of the described structure is lost, but details of structure geometry can now be expressed. *Turtle interpretation*, introduced by Szilard and Quinton [122], and extended by Prusinkiewicz [90, 91] and Hanan [41, 42], is representative of this approach. The summary below is based on [54, 91, 99].

The interpreted string is scanned sequentially from left to right, and its consecutive symbols are interpreted as commands that maneuver a LOGO-style turtle [1, 88] in three dimensions. The turtle is represented by its *state*, which consists of turtle *position* and *orientation* in the Cartesian coordinate system, as well as additional attributes, such as current *color* and *line width*. The position is defined by a vector $\mathbf{P}$, and the orientation is defined by three vectors $\mathbf{H}$, $\mathbf{L}$, and $\mathbf{U}$, indicating the turtle's *heading* and the directions to the *left* and *up* (Figure 3.3). These vectors have unit length, are perpendicular to each other, and satisfy the equation $\mathbf{H} \times \mathbf{L} = \mathbf{U}$. Consequently, rotations of the turtle can be expressed by the equation:

$$\left[\begin{array}{ccc} \mathbf{H'} & \mathbf{L'} & \mathbf{U'} \end{array} \right] = \left[\begin{array}{ccc} \mathbf{H} & \mathbf{L} & \mathbf{U} \end{array} \right] \mathbf{R}, \tag{3.3}$$

where $\mathbf{R}$ is a 3×3 rotation matrix [23]. Specifically, rotations by angle α about vectors $\mathbf{U}$, $\mathbf{L}$ and $\mathbf{H}$ are represented by the matrices:

$$\mathbf{R_U}(\alpha) = \left[\begin{array}{ccc} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{array} \right], \tag{3.4}$$

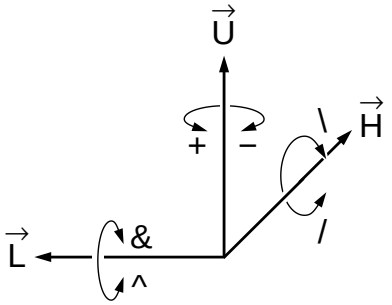


Fig. 3.3. Controlling the turtle in three dimensions. From [101].

$$\mathbf{R_L}(\alpha) = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}, \tag{3.5}$$

$$\mathbf{R_H}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}. \tag{3.6}$$

The turtle is initially located at the origin of a Cartesian coordinate system, with the heading vector **H** pointing in the positive direction of the y axis, and the left vector **L** pointing in the negative direction of the x axis. The turtle's actions and changes to its state are caused by interpretation of specific symbols, each of which may be followed by parameters. If one or more parameters are present, the value of the first parameter affects the turtle's state. If the symbol is not followed by any parameter, default values specified outside the L-system are used. The following list specifies the basic set of symbols interpreted by the turtle.

Symbols that cause the turtle to move and draw.

$F(s), G(s)$ Move forward a step of length s and draw a line segment from the original to the new position of the turtle.

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

$@O(r)$ Draw a sphere of radius r at the current position.

Symbols that control turtle orientation in space (Figure 3.3).

$+(\theta)$ Turn left by angle θ around the **U** axis. The rotation matrix is $\mathbf{R_U}(\theta)$.

$-(\theta)$ Turn right by angle θ around the **U** axis. The rotation matrix is $\mathbf{R_U}(-\theta)$.

$\&(\theta)$ Pitch down by angle θ around the **L** axis. The rotation matrix is $\mathbf{R_L}(\theta)$.

$\wedge(\theta)$ Pitch up by angle θ around the **L** axis. The rotation matrix is $\mathbf{R_L}(-\theta)$.

$/(\theta)$ Roll left by angle θ around the **H** axis. The rotation matrix is $\mathbf{R_H}(\theta)$.

$\backslash(\theta)$ Roll right by angle θ around the **H** axis. The rotation matrix is $\mathbf{R_H}(-\theta)$.

| Turn 180° around the **U** axis. This is equivalent to $+(180)$ or $-(180)$.

Symbols for modeling structures with branches.

[Push the current state of the turtle (position, orientation and drawing attributes) onto a pushdown stack.

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

Symbols for creating and incorporating surfaces.

{ Start saving the subsequent positions of the turtle as the vertices of a polygon to be filled.

} Fill the saved polygon.

$\sim X(s)$ Draw the surface identified by symbol X, scaled by s, at the turtle's current location and orientation. Such a surface is usually defined as a bicubic patch [41, 91].

Symbols that change the drawing attributes.

$\#(w)$ Set line width to w, or increase the value of the current line width by the default width increment if no parameter is given.

$!(w)$ Set line width to w, or decrease the value of the current line width by the default width decrement if no parameter is given.

$;(n)$ Set the index of the color map to n, or increase the value of the current index by the default color increment if no parameter is given.

$,(n)$ Set the index of the color map to n, or decrease the value of the current index by the default color decrement if no parameter is given.

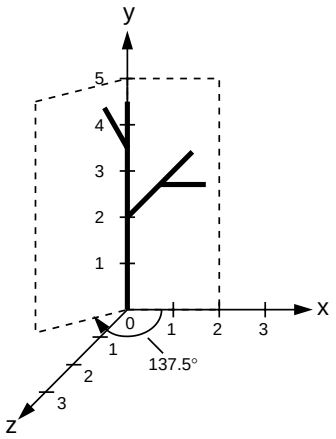
A sample string and its interpretation are shown in Figure 3.4.

The bracketed string notation can be applied, for example, to record the structure of observed plants (*plant mapping*, [14, 82]). Most of its applications are related, however, to the modeling and simulation of plant development using L-systems.

4. Fundamentals of modeling using L-systems

4.1 Parametric D0L-systems

The foundations of the theory of L-systems have been presented in many survey papers [70, 72, 73, 75, 76, 77] and books [46, 94, 99, 113, 116]. Consequently, we focus on parametric L-systems, which, according to our experience, are particularly convenient for modeling applications.



F(2)[−F[−F]F]/(137.5)F(1.5)[−F]F

Fig. 3.4. Example of the turtle interpretation of a string. The default length of lines represented by symbols F without a parameter is 1, and the default magnitude of the angles represented by symbols $+$ and $-$ is $45°$.

Parametric L-systems extend the basic concept of parallel rewriting from strings of symbols to parametric words. This extension was first implemented as a programming rather than a theoretical construct in the original simulator based on L-systems, called CELIA (an acronym for CEllular Linear Iterative Array simulator) [3, 45]. Early models relying on the use of parameters were described by Baker and Herman [3] and Lindenmayer [69] (see also [46, Chapter 18]). More recently, definitions of L-systems operating on symbols with parameters were proposed independently by Shebell [119] (who pointed out their relationship to attribute grammars [60]), Chien and Jürgensen [11], and Prusinkiewicz and Hanan [42, 96, 95, 99]. Our presentation follows this last approach.

Whenever no ambiguity is introduced by the biological connotation of the term "module", we will use it as a synonym for a "letter with associated parameters." We assume that 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.

The real-valued *actual* parameters appearing in words have a counterpart in the *formal* parameters, which may occur 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 (). The expressions can also include calls to standard mathematical functions, such as natural logarithm, sine, floor, and functions returning random variables. The operation symbols and the rules for constructing syntactically correct expressions are the same as in the C programming language [59]. For clarity of presentation, however, we sometimes use Greek letters and symbols with subscripts in print. 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 $\mathcal{E}(\Sigma)$.

A *parametric 0L-system* is an ordered quadruple $G = \langle V, \Sigma, \omega, P \rangle$, 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*.

We use symbols : and $\rightarrow$ to separate the three components of a production: the *predecessor*, the *condition* and the *successor*. Thus, a production in a 0L-system has the format

$$pred : cond \rightarrow succ. \tag{4.1}$$

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

$$A(t) : t > 5 \rightarrow B(t+1)CD(t \wedge 0.5, t-2). \tag{4.2}$$

The digit "0" in the term "0L-system" means that the context of the predecessor is not considered (in contrast to the context-sensitive 1L-systems and 2L-systems, described in Section 7.).

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 (4.2) 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 equal to 9. The result of production application in this case is the 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 $\mu_0 \Longrightarrow \mu_1 \Longrightarrow ... \Longrightarrow \mu_n$.

When no production explicitly listed as a member of the production set P matches a module in the rewritten string, we assume that an appropriate *identity production* belongs to P and replaces this module by itself. Under this assumption, a parametric 0L-system $G = \langle V, \Sigma, \omega, P \rangle$ is called *deterministic* (noted D0L) if and only if for each module $A(t_1, t_2, \ldots, t_n) \in V \times \Re^*$ the production set includes exactly one applicable production.

An example of a parametric D0L-system is given below. The words obtained in the first few derivation steps are shown in Figure 4.1.

$$\begin{array}{lllll} \omega: & B(2)A(4,4) & & & \\ p_1: & A(x,y) & : y < 3 & \rightarrow & A(x*2, x+y) \\ p_2: & A(x,y) & : y >= 3 & \rightarrow & B(x)A(x/y, 0) \\ p_3: & B(x) & : x < 1 & \rightarrow & C \\ p_4: & B(x) & : x >= 1 & \rightarrow & B(x-1) \end{array} \tag{4.3}$$

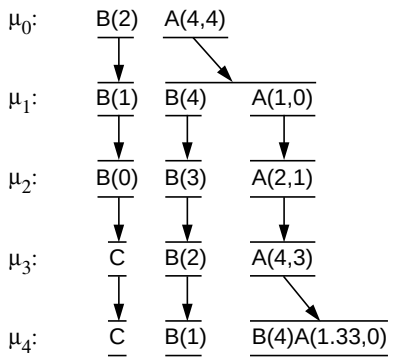


Fig. 4.1. The initial sequence of strings generated by the parametric L-system specified in equation (4.3)

There is no substantial difference between 0L-systems that operate on strings with or without brackets. Due to the interpretation of the brackets as delimiters of branches, however, productions involving brackets are restricted to the following forms:

– $pred : cond \rightarrow succ$, where $pred \in V \times \Sigma^*$, $cond \in \mathcal{C}(\Sigma)$, $succ \in ((V \times \mathcal{E}(\Sigma)^* \cup \{[,]\})^*$, and $succ$ is well nested;
– $[\rightarrow [$, or
– $] \rightarrow]$.

The above restrictions reflect, in particular, a biologically motivated assumption that branches can be initiated only by individual parent modules (*c.f.* [67, 68]).

4.2 Examples of parametric D0L-system models

This section presents selected examples that illustrate the operation of L-systems with turtle interpretation and their application to the modeling of plants. Many other examples are included in [42, 95, 99].

4.2.1 Fractal generation. Fractal curves provide a convenient means for illustrating the basic principle of L-system operation [90, 94, 99, 100]. For example, the following L-system generates the well-known snowflake curve [81, 127].

$$\begin{array}{ll} \omega : & F(1) - (120)F(1) - (120)F(1) \\ p_1 : & F(s) \rightarrow F(s/3) + (60)F(s/3) - (120)F(s/3) + (60)F(s/3) \end{array} \tag{4.4}$$

The axiom $F(1) - (120)F(1) - (120)F(1)$ draws an equilateral triangle, with edges of unit length. Production p_1 replaces each line segment with a polygonal shape, as shown at the top of Figure 4.2. Productions for symbols $+$ and $-$ are not listed, which means that the corresponding modules will be replaced by themselves during the derivation. The same effect could have been obtained by explicit inclusion of productions:

$$\begin{array}{ll} p_2 : & +(a) \rightarrow +(a) \\ p_3 : & -(a) \rightarrow -(a) \end{array} \tag{4.5}$$

The axiom and the figures obtained in the first three derivation steps are shown at the bottom of Figure 4.2.

4.2.2 Simulation of development. The next L-system generates the developmental sequence of the stylized compound leaf model presented in Figure 4.3.

$$\begin{array}{lllll} \omega : & !(1)F(1,1) & & & \\ p_1 : & F(s) & \rightarrow & G(s)[-!(1)F(s)][+!(1)F(s)]G(s)!(1)F(s) \\ p_2 : & G(s) & \rightarrow & G(2 * s) \\ p_3 : & !(w) & \rightarrow & !(3) \end{array} \tag{4.6}$$

The structure is built from two module types, apices F (represented by thin lines) and internodes G (thick lines). In both cases the parameter s determines the length of the line representing the module. An apex yields a structure

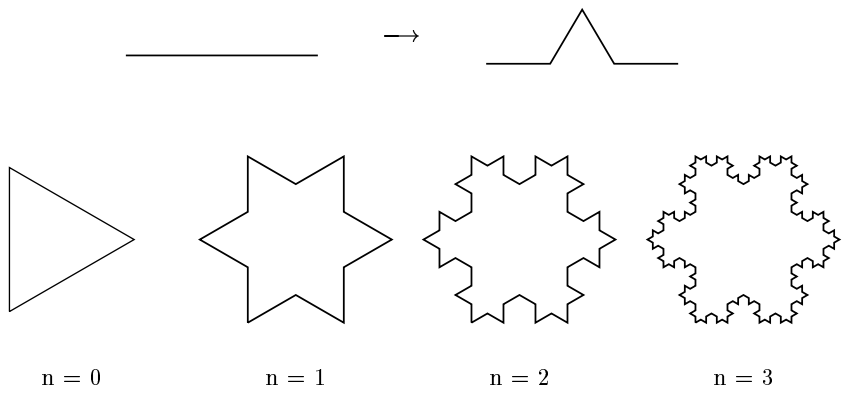


Fig. 4.2. Visual interpretation of the production for the snowflake curve, and the curve after $n = 0$, 1, 2, and 3 derivation steps

that consists of two internodes, two lateral apices, and a replica of the main apex (production p_1). An internode elongates by a constant scaling factor (production p_2). Production p_3 is used to make the lines representing the internodes wider (3 units of width) than the lines representing the apices (1 unit of width). The branching angle associated with symbols + and − is set to 45° by a global variable outside the L-system.

This example shows that parallel rewriting, inherent in L-systems, captures simultaneous changes that take place in different parts of an organism. A derivation step corresponds to the progress of time over some interval. The developmental sequence of structures obtained in consecutive derivation steps can be considered the result of a *discrete-time simulation* of development.

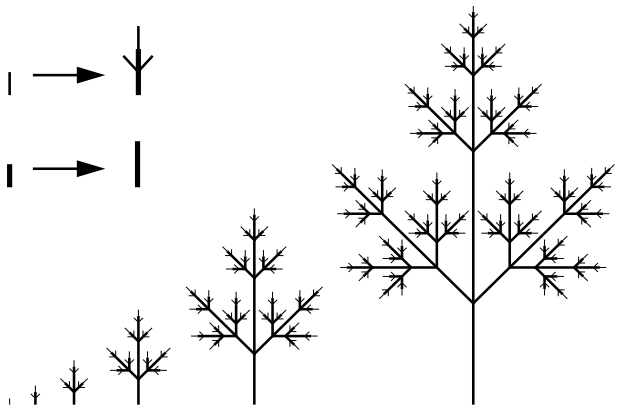


Fig. 4.3. The productions and a developmental sequence illustrating the operation of a compound leaf model

4.2.3 Exploration of parameter space. Parametric L-systems provide a convenient mathematical framework for exploring the range of forms that can be captured by the same structural model with varying attributes (constants in the productions). Such *parameter space explorations* motivated some of the earliest computer simulations of biological structures: the models of sea shells devised by Raup and Michelson [104, 105] and the models of trees proposed by Honda [50] to study factors that determine overall tree shape. (An L-system reproduction of Honda's results is presented in [99, Chapter 2].) Parameter space exploration may reveal an unexpected richness of forms that can be produced by even the simplest models. For example, Figure 4.4 shows nine branching structures selected from a continuum generated by the following parametric D0L-system:

$$\begin{array}{lll} \omega & : & A(100, w_0) \\ p_1 & : & A(s,w) \quad : s >= min \quad \rightarrow \quad !(w)F(s) \\ & & \qquad [+(\alpha_1)/(\varphi_1)A(s * r_1, w * q \wedge e)] \\ & & \qquad [+(\alpha_2)/(\varphi_2)A(s * r_2, w * (1-q) \wedge e)] \end{array} \tag{4.7}$$

The single non-identity production p_1 replaces apex A by an internode F and two new apices A. The angle values α_1, α_2, φ_1, and φ_2 determine the orientation of these apices with respect to the subtending internode. Parameters s and w specify internode length and width. The constants r_1 and r_2 determine the gradual decrease in internode length that occurs while traversing the tree from its base towards the apices. The constants w_0, q, and e control the width of branches. The initial stem width is specified by w_0 in the second parameter of the axiom module A. For $e = 0.5$, the combined area of the descendant branches is equal to the area of the mother branch, as postulated by Leonardo da Vinci [81, page 156] (see also [80, pages 131–135]). The value q specifies the differences in width between descendant branches originating at the same vertex. Finally, the condition prevents formation of branches with length less then the threshold value *min*. The values of constants corresponding to each structure are collected in Table 4.1. The final column headed n indicates the number of derivation steps.

Table 4.1. The values of constants used to generate Figure 4.4

Figure	r_1	r_2	α_1	α_2	φ_1	φ_2	w_0	q	e	min	n
a	.75	.77	35	-35	0	0	30	.50	.40	0.0	10
b	.65	.71	27	-68	0	0	20	.53	.50	1.7	12
c	.50	.85	25	-15	180	0	20	.45	.50	0.5	9
d	.60	.85	25	-15	180	180	20	.45	.50	0.0	10
e	.58	.83	30	15	0	180	20	.40	.50	1.0	11
f	.92	.37	0	60	180	0	2	.50	.00	0.5	15
g	.80	.80	30	-30	137	137	30	.50	.50	0.0	10
h	.95	.75	5	-30	-90	90	40	.60	.45	25.0	12
i	.55	.95	-5	30	137	137	5	.40	.00	5.0	12

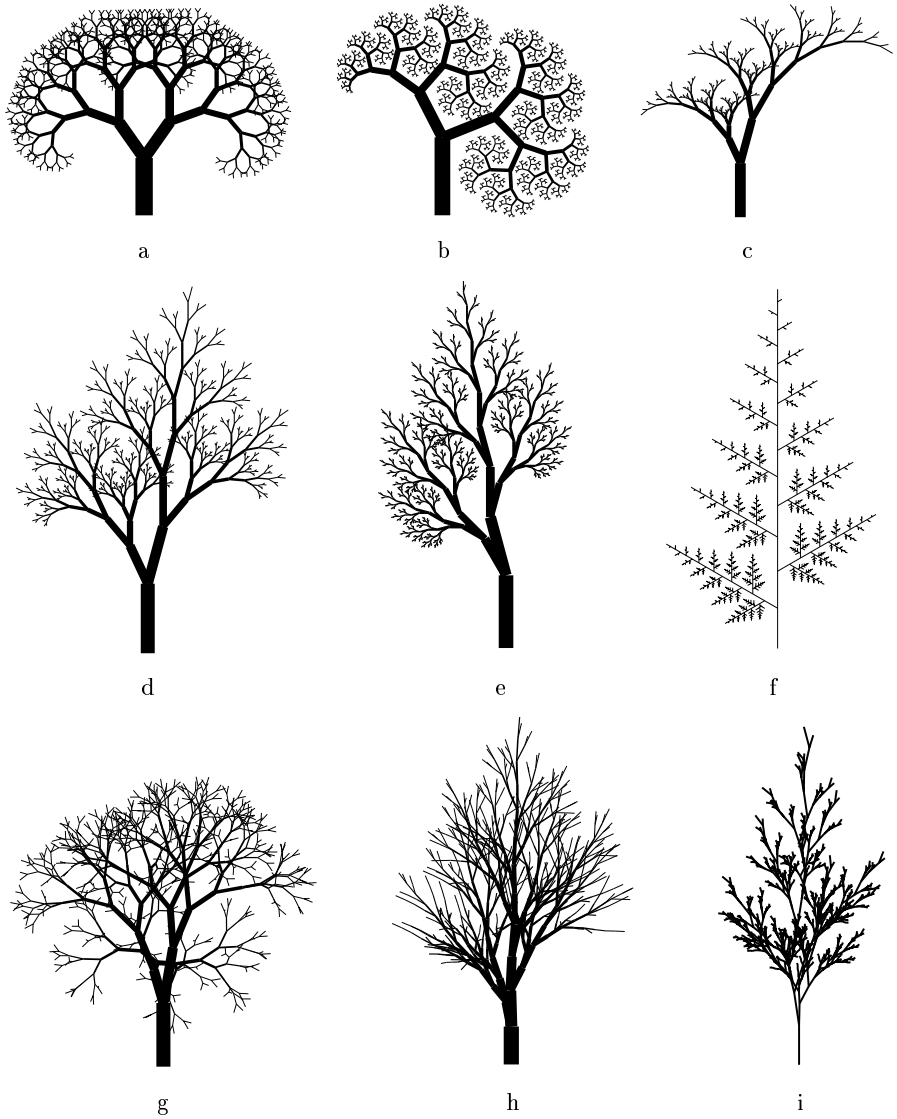


Fig. 4.4. Sample structures generated by a parametric D0L-system with different values of constants

4.2.4 Modeling mesotonic and acrotonic structures. In spite of their apparent diversity, the structures generated by L-system (4.7) share a common developmental pattern: in each derivation step, every apex gives rise to an internode terminated by a pair of new apices. This is a simple instance of *subapical branching*, a common developmental pattern in plants, in which new branches can be initiated only near the apices of the existing axes. As a consequence of this pattern, the lower branches, being created first, have more time to develop than the branches further up, and a *basitonic* structure (more developed near the base than near the top) results (Figure 4.5a). In

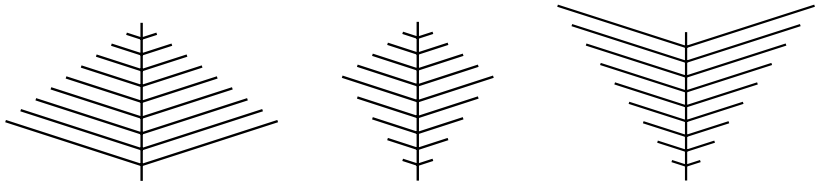


Fig. 4.5. Schematic representation of a basitonic (a), mesotonic (b), and acrotonic (c) branching pattern. From [98].

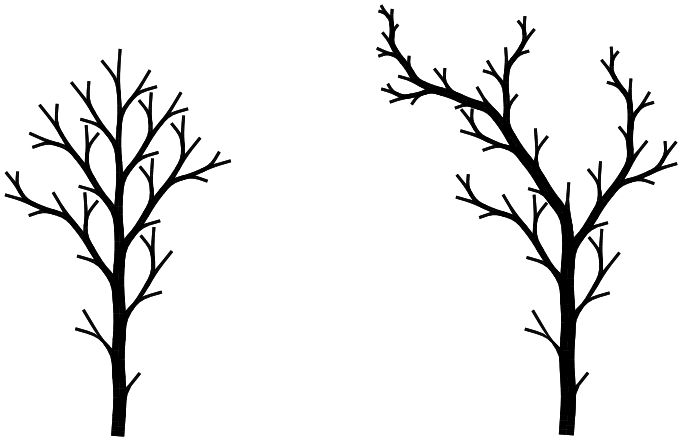
nature, however, one also finds *mesotonic* and *acrotonic* structures, in which the most developed branches are located near the middle or the top of the mother branch (Figures 4.5 b and c). As observed by Frijters and Lindenmayer [31], and formalized by Prusinkiewicz and Kari [98], arbitrarily large mesotonic and acrotonic structures cannot be generated by non-parametric deterministic 0L-systems with subapical branching. In contrast, *parametric* D0L-systems can generate such structures, as demonstrated by the following model, proposed by Lück, Lück, and Bakkali [79].

$$\begin{array}{lll} \omega & : & -(\alpha_1/2)/(180)F(1,0,v_0) \\ p_1 & : & F(s,k,v) \quad : v > 0 \rightarrow G(s)[+(\alpha_2)F(s*r_2,0,k)] \\ & & \qquad /(180) + (\alpha_1)F(s*r_1,k+1,v-1) \end{array} \tag{4.8}$$

Like L-system (4.6), L-system (4.8) operates on two types of segments: apices F and internodes G. Their length is controlled by parameter s. The segments created in consecutive derivation steps are shortened by factors r_1 and r_2 with respect to their parents, as in L-system (4.7). The remaining two parameters of the apices specify:

- the current number k of segments in the axis to which the apex belongs, and
- the growth potential or vigor v of the apex, defined as the number of straight segments that can be appended to the current axis (if the derivation is sufficiently long).

Except for the apex of the main axis, which has an initial vigor v_0 determined by the axiom ω, the vigor of the apices is determined by production p_1. Specifically, a new lateral apex is assigned an initial vigor value v equal to the ordering number k of its supporting segment in the mother axis. Consequently, lateral branches positioned further away from the base of their mother axis have relatively larger initial values of v, and may develop more extensively than those close to the base. Two examples of the resulting structures are shown in Figure 4.6. In case (a), the upper branches are still developing, and



a b

Fig. 4.6. A juvenile mesotonic structure (a) and a mature acrotonic structure (b) generated using L-system (4.8)

the resulting structure is mesotonic. In case (b), all branches have already reached their full growth potential, and the resulting structure is acrotonic. The values of the constants corresponding to each structure are collected in Table 4.2. For better appearance, additional rules, not shown in L-system (4.8), were used to define the width of segments and to capture their curving near branching points.

Table 4.2. The values of constants used to generate Figure 4.6

Figure	r_1	r_2	α_1	α_2	v_0	n
a	.98	.80	8	35	11	10
b	.98	.80	8	35	6	21

5. Random factors in development

5.1 The role of randomness in the description of development

In the previous section, developmental processes were regarded as deterministic phenomena and captured by L-systems that always produce the same developmental sequence. However, random factors often intervene in the construction of plant models. Two cases can be distinguished:

- Details of the mechanisms that control the development and resulting architecture are not known, and only statistical data are available for model construction. For example, such data may express the distribution of internode lengths and branching angles, probability of branching, or correlation between bud position on a branch and its developmental fate. Statistical descriptions of this nature are widely reported in the botanical literature and provide essential input for some modeling methods (for instance, see [7, 53, 18, 106, 130]).
- Physiological mechanisms responsible for the control of developmental processes are known or have been postulated, but it is convenient to capture their average outcome rather than the details of operation in models constructed at a high level of abstraction. For example, several plausible mechanisms for preventing the overcrowding of branches in a tree have been described in the literature [8, 51], yet a statistical model that captures the density of branches without simulating the controlling processes is also useful (Sections 5.3 and 8.3.2).

Several stochastic extensions of L-systems have been proposed in the literature [21, 56, 87, 133] and applied to express developmental plant models [87, 91, 94]. The definition given below extends previous concepts to parametric L-systems.

5.2 Stochastic 0L-systems

A *parametric stochastic 0L-system* is an ordered quintuplet:

$$G_\pi = \langle V, \Sigma, \omega, P, \pi \rangle. \tag{5.1}$$

The alphabet V, set of formal parameters Σ, axiom ω and set of productions P are defined as for parametric D0L-systems (Section 4.1). Function $\pi : P \to \mathcal{E}(\Sigma)$, assigns an arithmetic expression returning a nonnegative number called the *probability factor* to each production in P. A production in a stochastic L-system is written as

$$pred : cond \to succ : prob, \tag{5.2}$$

where *pred*, *cond* and *succ* play the same role as in D0L-systems (Equation 4.1), and *prob* is an expression returning the probability factor.

If $\hat{P} \subseteq P$ is the set of productions matching a given module $A(t_1, t_2, \ldots, t_n)$ in the rewritten string, then the probability $\text{prob}(p_k)$ of applying a particular production $p_k \in \hat{P}$ to this module is equal to:

$$\text{prob}(p_k) = \frac{\pi(p_k)}{\sum_{p_i \in \hat{P}} \pi(p_i)} \tag{5.3}$$

In general, this probability is not a constant associated with a production, but may depend on the parameter values in the rewritten module.

We will call the derivation $\mu \Longrightarrow \nu$ a *stochastic derivation* in G_π if for each occurrence of a module a in the word μ the probability of applying production p_k with the predecessor a is given by Equation (5.3).

According to this definition, different productions with the same predecessor may be applied to different occurrences of the same module in one derivation step.

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

$$\begin{array}{ll} \omega: & A(1)B(3)A(5) \\ p_1: & A(x) \rightarrow A(x+1) : 2 \\ p_2: & A(x) \rightarrow B(x-1) : 3 \\ p_3: & A(x) : x > 3 \rightarrow C(x) : x \\ p_4: & B(x) \rightarrow B(2 * x)A(x) : 1 \end{array} \tag{5.4}$$

Productions p_1, p_2, and p_3 replace module $A(x)$ by $A(x+1)$, $B(x-1)$, or $C(x)$. If the value of parameter x is less then or equal to 3, only the first two productions match $A(x)$. The probabilities of applying each production are: $\text{prob}(p_1) = 2/(2+3) = 0.4$, and $\text{prob}(p_2) = 3/(2+3) = 0.6$. If parameter x is greater then 3, production p_3 also matches the module $A(x)$, and the probability of applying each production depends on the value of x. For example, if x is equal to 5, these probabilities are: $\text{prob}(p_1) = 2/(2+3+5) = 0.2$, $\text{prob}(p_2) = 3/(2+3+5) = 0.3$, and $\text{prob}(p_3) = 5/(2+3+5) = 0.5$. Production p_4 replaces a module $B(x)$ by the pair of modules $B(2*x)A(x)$. Taking all these factors into account, the first derivation step in L-system (5.4) may have the form:

$$A(1)B(3)A(5) \Longrightarrow A(2)B(6)A(3)C(5) \tag{5.5}$$

where production p_1 was applied to the module $A(1)$, and production p_3 to the module $A(5)$ as a result of random choice.

5.3 A stochastic tree model

Many simple models of branching structures, for example those discussed in Sections 4.2.3 and 4.2.2, produce an exponentially increasing number of branch segments. Using morphometric data of young cottonwood *(Populus deltoides)* and observations of the tropical tree *Tabebuia rosea*, Borchert and Slade [9] showed that in reality this exponential increase is not sustained

beyond the early stages of tree development. As soon as a tree surpasses a certain, relatively small size, the rate of branching decreases. Below we present a stochastic model of a hypothetical tree, based on the analysis by Borchert and Slade. The material in this section includes an edited version of [97].

The model is constructed to meet the following botanically justifiable postulates:

- The development begins in season $k = 1$ with the formation of a single nonbranching shoot (branch segment bearing leaves).
- In each subsequent growth season, new shoots grow from the buds situated near the distal ends of last year's segments. There is a constant, $b_{max} > 1$, that determines the maximum bifurcation ratio (*i.e.*, the maximum number of shoots originating at the mother branch).
- All branch segments have approximately the same length l, independent of their position and the age of the tree, and reach out forming a hemispherical crown.
- Leaves are produced on the terminal (current year) branch segments, thus forming a hemispherical layer of leaves near the perimeter of the crown. There is a constant, σ_{min}, that determines the minimum area of leaves that must be exposed to light coming from the outside in order to create a viable shoot.

According to these postulates, the radius of a tree crown after $k \geq 1$ growth seasons is limited by $R_k = lk$. A hemispherical crown of this radius has surface area $S_k = 2\pi R_k^2 = 2\pi l^2 k^2$, and this value determines the upper bound on the crown area exposed to direct light. The number N_{k+1} of branch segments added to the tree in year $k+1$ is limited, on the one hand, by the number of last year's segments N_k multiplied by the maximum bifurcation ratio b_{max}, and on the other hand, by the maximum number of shoots $v_{k+1} = S_{k+1}/\sigma_{min}$ that may be produced without excessively obscuring each other. Thus,

$$N_{k+1} = \min\{b_{max}N_k, v_{k+1}\} = \min\{b_{max}N_k, \frac{2\pi l^2}{\sigma_{\min}}(k+1)^2\}. \tag{5.6}$$

Let us assume that the minimum leaf area exposed to light per shoot is small compared to the crown area, $\sigma_{min} \ll 2\pi l^2$. In a young tree (during the first few growth seasons), the maximum number of new shoots does not suffice to cover the available crown surface ($b_{max}N_k < v_{k+1}$), and the number of new shoots will increase exponentially with the age of the tree: $N_{k+1} = b_{max}N_k = b_{max}^k$. Since the crown area is proportional only to the square of the age of the tree, at some age t the potential number of new shoots will exceed the number that can be sufficiently exposed to direct light: $b_{max}N_t \geq v_t$. From then on, branching will be limited by the crown area, with the average bifurcation ratio b_k at age $k \geq t$ equal to:

$$b_k = \frac{N_{k+1}}{N_k} = \frac{2\pi l^2(k+1)^2/\sigma_{min}}{2\pi l^2 k^2/\sigma_{min}} = 1 + \frac{2k+1}{k^2}. \tag{5.7}$$

Different branching patterns may satisfy this general formula. For example, if each segment from the previous year gives rise to either one or two new shoots, the fraction of segments supporting two shoots will be equal to:

$$\frac{N_{k+1} - N_k}{N_k} = \frac{2k+1}{k^2}. \tag{5.8}$$

The stochastic L-system below has been constructed to satisfy this equation.

$$\begin{array}{lll} \omega & : & FA(1) \\ p_1 & : & A(k) \rightarrow /(\varphi)[+(\alpha)FA(k+1)] - (\beta)FA(k+1) : \\ & & \qquad \min\{1, (2k+1)/k^2\} \\ p_2 & : & A(k) \rightarrow /(\varphi)B - (\beta)FA(k+1) : \\ & & \qquad \max\{0, 1-(2k+1)/k^2\} \end{array} \tag{5.9}$$

Generation of the tree begins with a single internode F terminated by apex $A(1)$. The parameter of the apex acts as a counter of derivation steps. Production p_1 describes the creation of two new branches, while production p_2 describes the production of a branch segment and a dormant bud B. Probabilities of these events are equal to $p = \min\{1, (2k+1)/k^2\}$, and $q = 1 - p$, respectively. This corresponds to the assumption that the departure from exponential bifurcation occurs in step $k = 3$, and in subsequent steps the probability of bifurcation is determined by Equation (5.8). Figure 5.1 shows

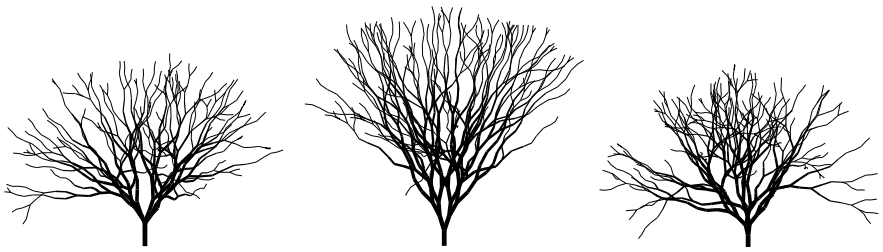


Fig. 5.1. Sample tree structures generated using the stochastic L-system (5.9). From [97].

side views of three sample trees after 18 derivation steps. The branching angles, equal to $\varphi = 90°, \alpha = 32°$, and $\beta = 20°$, yield a sympodial branching structure (new shoots do not continue the growth direction of the preceding segments). This structure is a representative of Leeuwenberg's model of tree architecture identified by Hallé *et al.* [39], although no attempt to capture a particular tree species was made.

6. Life, death, and reproduction

The L-systems considered so far were of the *propagating* type. Each module, once created, continued to exist indefinitely or gave rise to one or more children, but never disappeared without a trace. The natural processes of plant development, however, often involve the programmed death of selected modules and their removal from the resulting structures. We consider these phenomena in the present section.

6.1 Non-propagating L-systems

The original approach to simulating module death was to use *non-propagating* L-systems, which incorporate *erasing* productions [46]. In the context-free case these productions have the form $A \rightarrow \varepsilon$, where ε denotes the empty string. Intuitively, module A is replaced by "nothing" and is removed from the structure. Erasing productions can faithfully simulate the disappearance of individual modules placed at the extremities of the branching structure (that is, not followed by other modules). For example, in the developmental sequence shown in Figure 6.1 (described in detail in [93]), erasing productions have been used to model the fall of petals.

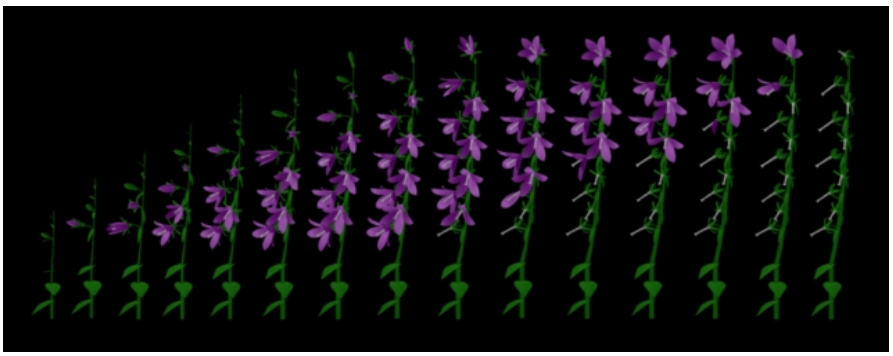


Fig. 6.1. Simulated development of a bluebell flower (*Campanula rapunculoides*). From [93].

6.2 L-systems with a cut symbol

The modeling task becomes more difficult when an entire structure, such as a branch, is shed by a plant. The plant can control this process by *abscission*, that is, the development of a pithy layer of cells that weakens the stem of a branch at its base. Obviously, abscission is represented more faithfully by cutting a branch off than by simultaneously erasing all of its modules. In

order to simulate shedding, Hanan [42] extended the formalism of L-systems with the "cut symbol" %, which causes the removal of the remainder of the branch that follows it. For example, in the absence of other productions, the derivation step given below takes place:

$$a[b\%[cd]e[\%f]]g[h[\%i]j]k \Longrightarrow a[b]g[h[]j]k \tag{6.1}$$

A simple example of an L-system incorporating the cut symbol is given below:

$$\begin{array}{lllll} \omega: & A & & & \\ p_1: & A & & \rightarrow & F(1)[-X(3)B][+X(3)B]A \\ p_2: & B & & \rightarrow & F(1)B \\ p_3: & X(d) & : d > 0 & \rightarrow & X(d-1) \\ p_4: & X(d) & : d == 0 & \rightarrow & U\% \\ p_5: & U & & \rightarrow & F(0.3) \end{array} \tag{6.2}$$

According to production p_1, in each derivation step the apex of the main axis A produces an internode F of unit length and a pair of lateral apices B. Each apex B extends a branch by forming a succession of internodes F (production p_2). After three steps from branch initiation (controlled by production p_3), production p_4 inserts the cut symbol % and an auxiliary symbol U at the base of the branch. In the next step, the cut symbol removes the branch, while symbol U inserts a marker $F(0.3)$ indicating a "scar" left by the removed branch. The resulting developmental sequence is shown in Figure 6.2. The

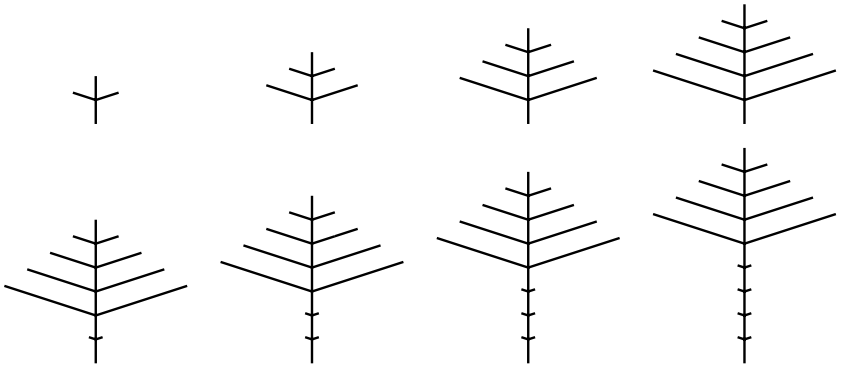


Fig. 6.2. A developmental sequence generated by the L-system specified in Equation 6.2. The images shown represent derivation steps 2 through 9.

initial steps capture the growth of a *basitonic* structure (developed most extensively at the base). Beginning at derivation step 6, the oldest branches are shed, creating an impression of a tree crown of constant shape and size moving upwards. The crown is in a state of dynamic equilibrium: the addition of new branches and internodes at the apices is compensated by the loss of branches further down.

Fig. 6.3. A model of the date palm (*Phoenix dactylifera*). This image was created using an L-system with the general structure specified in Equation 6.2.

The state of dynamic equilibrium can be easily observed in the development of palms, where new leaves are created at the apex of the trunk while old leaves are shed at the base of the crown (Figure 6.3). Since both processes take place at the same rate, an adult palm carries an approximately constant number of leaves. This phenomenon has an interesting physiological explanation: palms are unable to gradually increase the diameter of their trunk over time, thus the flow of water and nutrients through the trunk can support only a crown of constant size.

6.3 Fragmentation

In the case of falling leaves and branches, the parts separated from the main structure die. Separation of modules can also lead to the asexual reproduction of plants. This phenomenon takes place, for example, when plants propagate through *rhizomes*, or stems that grow horizontally below the ground and bear buds which produce vertical shoots. The rhizome segments (internodes) have a finite life span, and rot progressively from the oldest end, thus dividing the original plant into independent organisms.

A model of the propagation of rhizomes in *Alpinia specioza*, a plant of the ginger family, was proposed by Bell, Roberts, and Smith [7]. A simulation carried out using an L-system reimplementation of this model is shown in Figure 6.4. All rhizome segments are assumed to have the same length. Each year (one derivation step in the simulation), an apex produces one or two daughter segments. The decision mechanism is expressed using stochastic productions. The segments persist for seven years from their creation, then die off, thereby dividing the plant.

In the model shown in Figure 6.4, the death of segments has been captured using productions of type $F \rightarrow f$, which replace "old" segments F by invisible links (turtle movements) f (*c.f.* Section 3.3). This replacement guarantees that the separated organisms will maintain their relative positions. Although the effect is visually correct, maintaining any type of connection between the separated plants is artificial and of limited practical use. For instance, it is inappropriate for expressing such phenomena as the parting of free-floating water plants after separation [107]. An alternative solution may be to modify the notion of L-systems so that they operate on *sets* of words, each representing an individual plant. A separation of structures could be then expressed as a division of a word into two or more independent subwords. The notion of *L-systems with fragmentation*, defined by Rozenberg, Ruohonen, and Salomaa for strings without brackets [112, 115] (see also [113, pages 78–82]) may offer a starting point for a future extension applicable to branching structures as well.

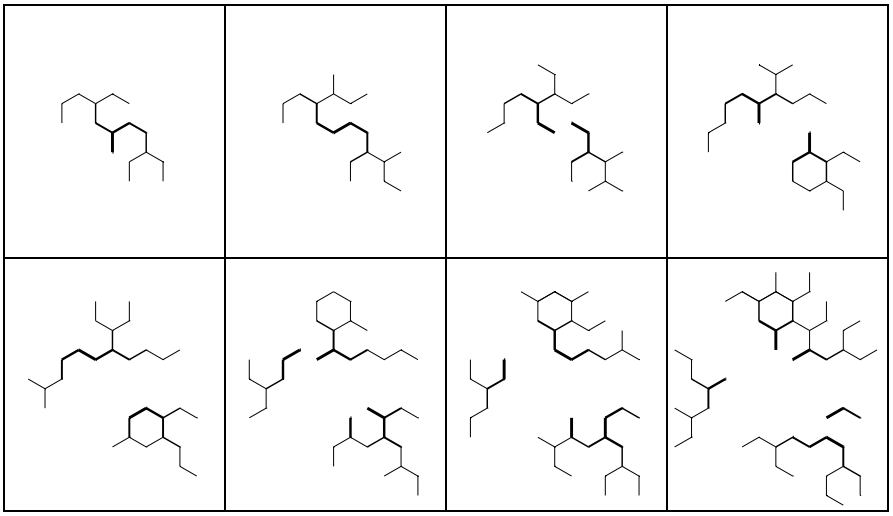


Fig. 6.4. Development of the rhizomatous system of *Alpinia speciosa*. The images show consecutive stages of simulation generated in 6 to 13 derivation steps. Line width indicates the age of the rhizome segments. Each segment dies and disappears seven steps after its creation.

7. Development controlled by endogenous mechanisms

7.1 Information flow in growing plants

Communication between modules plays a crucial role in the control of developmental processes in plants. Lindenmayer distinguished two forms of communication: *lineage* (also called *cellular descent*), which represents information transfer from a parent module to its children, and *interaction*, which represents information transfer between coexisting modules [67, 73]. In the latter case, the information exchange may be *endogenous* (between adjacent modules of the structure, as defined by its topology), or *exogenous* (through the space embedding the structure) [92]. The flow of water, hormones, or nutrients through the vascular system of a plant are examples of endogenous information transfer, whereas the shading of lower branches by upper ones is a form of exogenous transfer. Referring specifically to the initiation of branches, Bell [6] further clarified these distinctions as follows:

- In *blind* patterns, based on lineage, branch initiation is controlled by the parent module independent of the remainder of the structure and the environment in which this structure develops;
- In *self-regulatory* patterns, based on endogenous interaction, branch initiation is controlled potentially by the whole developing structure, using communication via the existing components of this structure;

– In *sighted* patterns, based on exogenous interaction, the initiation of a new branch is influenced by factors detected by its parent in the immediate geometric neighborhood, such as proximity of other organisms or parts of the same organism.

Productions in 0L-systems are *context-free* (applicable irrespective of the context in which the predecessor appears), and therefore can only express developmental mechanisms controlled by lineage. However, by making production application depend on the predecessor's context, endogenous interaction can be captured as well. The conceptual elegance and expressive power of the resulting *context-sensitive* L-systems are among the most important assets of L-systems in plant modeling applications.

7.2 Context-sensitive L-systems

Various context-sensitive extensions of L-systems have been proposed and thoroughly studied in the past (for example, see [46, 78, 116]). Below we focus on *bracketed* context-sensitive L-systems, which operate on strings of modules describing branching structures [42, 95, 99]. In the non-stochastic case, their productions have the format:

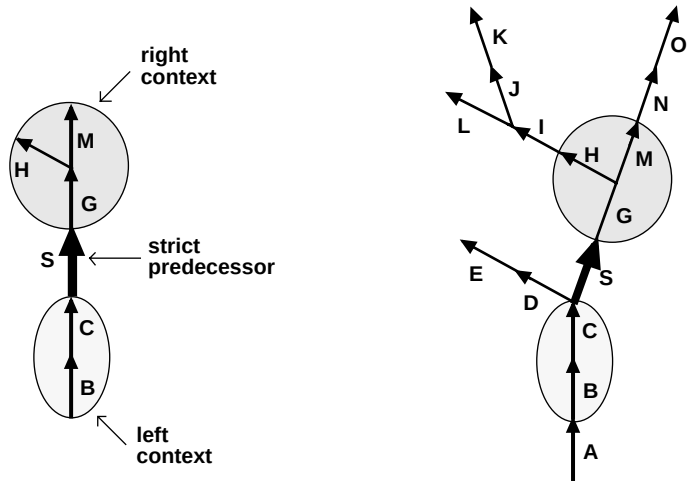
$$lc < pred > rc : cond \rightarrow succ, \tag{7.1}$$

where symbols $<$ and $>$ separate the three components of the predecessor: a string of modules without brackets lc called the *left context*, a module *pred* called the *strict predecessor*, and a well-nested bracketed string of modules rc called the *right context*. The remaining components of the production are the condition *cond* and the successor *succ*, defined as for (bracketed) D0L-systems.

The key new mechanism introduced in context-sensitive L-systems is the process of matching the predecessor of a production to a given module in the rewritten string. We will describe this process in terms of axial trees (*c.f.* Section 3.), assuming initially that modules (letters) have no associated parameters. As shown in Figure 7.1a, the strict predecessor of a production is a segment situated between a path specified by the left context, and an axial tree specified by the right context. When no parameters are present, a production p *matches* a given occurrence of segment S in an axial tree T if the following conditions are met:

– the strict predecessor *pred* is the symbol S,
– the left context lc represents a path in T terminating at the beginning of S, and
– the right context rc represents a subtree of T originating at the end of S.

A matching production can be applied by replacing S with the axial tree specified as the production successor (Figure 7.1b).



a b

Fig. 7.1. The predecessor of a context-sensitive tree production (a) matches edge S in a tree T (b). From [101]

When a bracketed string representation is used to express the productions and rewritten structure, the process of matching predecessors to modules in bracketed context-sensitive L-systems cannot be reduced to simple string matching, 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 7.1 indicates that a production with the predecessor $BC < S > G[H]M$ can be applied to symbol S in the string

$$ABC[DE][SG[HI[JK]L]MNO], \tag{7.2}$$

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

In the parametric case, a production that matches an occurrence of the module S in a rewritten structure must satisfy additional conditions, similar to those defined for parametric 0L-systems:

- the number of formal parameters in each module in the predecessor (*i.e.*, in the strict predecessor and the contexts) is the same as the number of actual parameters in the corresponding modules of the tree T, and
- the condition evaluates to *true* if the actual parameter values are substituted for the formal parameters in the production.

For example, the parametric context-sensitive production:

$$A(x) < B(y) > C[D(z)]F : x{+}y{+}z > 10 \rightarrow U((x{+}y)/2)[V((y{+}z)/2)] \tag{7.3}$$

can be applied to the module $B(5)$ appearing in a parametric word

$$\cdots A(4)B(5)C[D(6)E]F\cdots \tag{7.4}$$

because the letters and the numbers of parameters in all modules involved in comparisons 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 the branching structure $U(4.5)[V(5.5)]$. Naturally, the remaining modules of the rewritten string may also be replaced (by other productions) in the same derivation step.

Productions in 2L-systems use context on both sides of the strict predecessor. 1L-systems are a special case of 2L-systems in which context appears only on one side of the productions. (Consistent with this convention, no context is considered in 0L-systems.) Using biological terminology, the left context expresses *acropetal* information flow (from the base of a branching structure up towards the apices) whereas the right context expresses *basipetal* flow (from the apices down towards the root). For example, the following 1L-system simulates propagation of an acropetal signal in a branching structure that does not grow:

$$\begin{array}{ll} \multicolumn{2}{l}{\text{ignore} : +-} \\ \omega & : \quad F_b[+F_a]F_a[-F_a]F_a[+F_a]F_a \\ p_1 & : \quad F_b < F_a \rightarrow F_b \end{array} \tag{7.5}$$

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 7.2a.

The propagation of a basipetal signal can be simulated in a similar way (Figure 7.2b), using the following L-system:

$$\begin{array}{ll} \multicolumn{2}{l}{\text{ignore} : +-} \\ \omega & : \quad F_a[+F_a]F_a[-F_a]F_a[+F_a]F_b \\ p_1 & : \quad F_a > F_b \rightarrow F_b \end{array} \tag{7.6}$$

The asymmetry between acropetal signal propagation (where the signal enters the lateral branches) and basipetal propagation (where it does not enter these branches), apparent in Figure 7.2, is a consequence of segment orientation in rooted trees. According to this orientation, there exists a directed path from the root to any of the apices, but there is no directed path from one apex to another (Section 3.1).

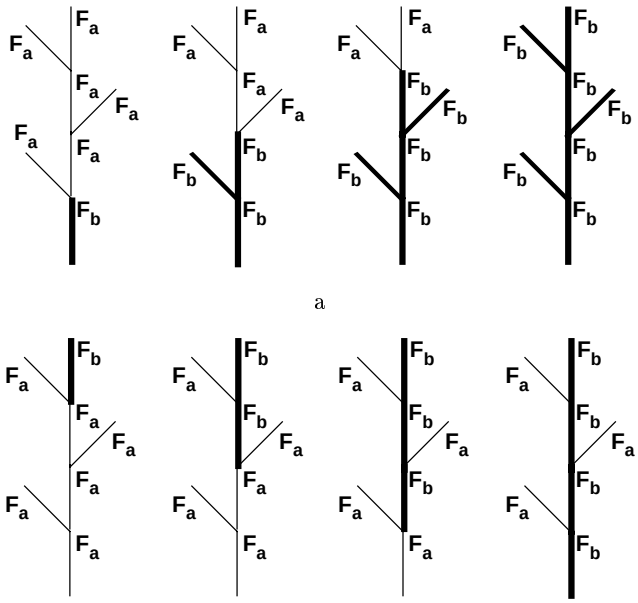


Fig. 7.2. Signal propagation in a branching structure: (a) acropetal, (b) basipetal. From [101].

7.3 Examples

Developmental processes based on endogenous information flow can be analyzed, modeled, and categorized in terms of the following features.

- *Direction of information flow.* As described above, the two basic cases are acropetal and basipetal flow. Information may also be exchanged between neighboring modules in a symmetric fashion (without a preferred direction), for example when the substances carrying the information are transported by diffusion. One model may include several signals, propagating in the same or different directions simultaneously or one after another.
- *The processes taking place at branching points.* A branching point can be likened to a communications hub, where the information coming from several sources is combined, processed, and redistributed in particular directions. Different variants of these processes may occur.
- *Granularity of information.* The information exchanged between the modules may represent discrete *signals* (for example, the presence of a hormone triggering the transformation of a bud to a flower), or quantifiable *values* (for example, the concentration of photosynthates produced by leaves).

Several possibilities are illustrated by the models given below. We begin with a simple model in which a directional (acrotonic) signal is uniformly distributed at each branching point towards all supported segments.

7.3.1 Development of a mesotonic structure. As outlined in Section 4.2.4, arbitrarily large mesotonic and acrotonic structures cannot be generated using deterministic 0L-systems without parameters [98]. The proposed mechanisms for modeling these structures can be divided into two categories: those using parameters to characterize the growth potential or vigor of individual apices, such as L-system (4.8), and those postulating control of development by signals [30, 55]. The following L-system simulates the development of the mesotonic structure shown in Figure 7.3 using an acropetal (upward moving) signal.

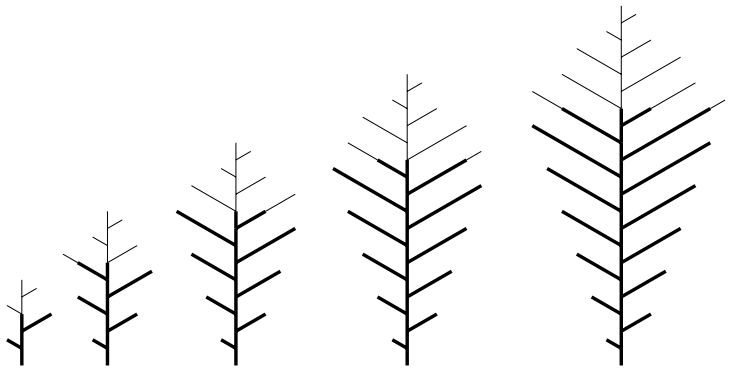


Fig. 7.3. Development of a mesotonic branching structure controlled by an acropetal signal. Wide lines indicate the internodes reached by the signal. The stages shown correspond to derivation lengths 12, 24, 36, 48, and 60.

#define m 3 /* plastochron of the main axis */
#define n 4 /* plastochron of the branch */
#define u 4 /* signal propagation rate in the main axis */
#define v 2 /* signal propagation rate in the branch */

ignore : $+ -/$

$$
\begin{array}{lll}
\omega : & S(0)F(1,0)A(0) & \\
p_1 : & A(i) \quad : i < m-1 \rightarrow A(i+1) & (7.7) \\
p_2 : & A(i) \quad : i == m-1 \rightarrow [+(60)F(1,1)B(0)]F(1,0)/(180)A(0) & \\
p_3 : & B(i) \quad : i < n-1 \rightarrow B(i+1) & \\
p_4 : & B(i) \quad : i == n-1 \rightarrow F(1,1)B(0) & \\
p_5 : & S(i) \quad : i < u+v \rightarrow S(i+1) & \\
p_6 : & S(i) \quad : i == u+v \rightarrow \varepsilon & \\
p_7 : & S(i) < F(l,o) : (o == 0)\&\&(i == u-1) \rightarrow \#F(l,o)!S(0) & \\
p_8 : & S(i) < F(l,o) : (o == 1)\&\&(i == v-1) \rightarrow \#F(l,o)!S(0) & \\
p_9 : & S(i) < B(j) \rightarrow \varepsilon &
\end{array}
$$

L-system (7.7) operates under the assumption that the context-sensitive production p_9 takes priority over p_3 or p_4. The axiom ω describes the initial structure as an internode F terminated by an apex A. A signal S is placed at the base of this structure. According to productions p_1 and p_2, the apex A periodically produces a lateral branch and adds an internode to the main axis. The period (called the *plastochron* of the main axis) is controlled by the constant m. Productions p_3 and p_4 describe the development of the lateral branches, where new segments F are added with plastochron n. Productions p_5 to p_8 describe the propagation of the signal through the structure. The signal propagation rate is u in the main axis, and v in the branches. Production p_9 removes the apex B when the signal reaches it, thus terminating the development of the corresponding lateral branch. Figure 7.3 shows that, for the values of plastochrons and signal propagation rates specified be the #define statements, the lower branches have less time to grow than the higher branches, and a mesotonic structure develops as a result.

A similar mechanism, based on the pursuit of apices by acropetal signals, has been proposed to model basipetal flowering sequences [55, 74, 99]. These sequences are characterized by the appearance of the first flower near the top of a plant, and a subsequent downward propagation of the flowering zone.

7.3.2 Attack of a plant by an insect. More complex information flow is considered in the next example. A hypothetical insect explores a growing branching structure and feeds on its apices. The insect always moves along the branches (*i.e.,* it does not jump or drop from one branch to another) and therefore can be treated as an endogenous signal. The insect's behavior at a branching point depends on its direction of motion and the state of the branching point, as explained in Figure 7.4. In a nutshell, the insect attempts to traverse the entire developing structure using the depth-first strategy. A

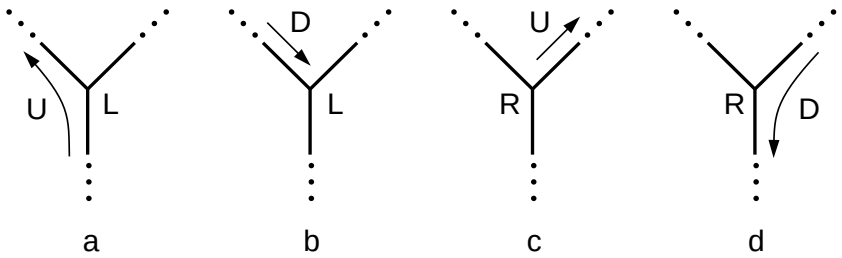


Fig. 7.4. Insect's behavior at a branching point. An upward-moving insect U that approaches a branching point marked L is directed to the left daughter branch (a). A downward moving insect D that approaches a branching point marked L changes this marking to R, returns to state U, and enters the right branch (b and c). A downward moving insect D approaching a branching point marked R continues its downward motion (d).

context-sensitive L-system that integrates plant growth with the behavior of the insect is given below.

```
#define l_L   3     /* length of the left branch */
#define l_R   5     /* length of the right branch */
#define d     5     /* plastochron */
#define w     40    /* delay */
```

$$
\begin{array}{lllll}
\omega & : & W(w)FA(l_L, d) & & \\
p_1 : & F & < \; A(n,m) & : m > 0 & \rightarrow \; A(n, m-1) \\
p_2 : & F & < \; A(n,m) & : n > 0 \;\&\&\; m == 0 & \rightarrow \; FA(n-1, d) \\
p_3 : & F & < \; A(n,m) & : n == 0 \;\&\&\; m == 0 & \\
 & & & \rightarrow \; L[+FA(l_L, d)][-FA(l_R, d)] & \\
p_4 : & W(t) & : \; t > 0 & \rightarrow \; W(t-1) & \\
p_5 : & W(t) & : \; t == 0 & \rightarrow \; U & \\
p_6 : & U & < \; F & \rightarrow \; FU & \\
p_7 : & U & & \rightarrow \; \varepsilon & \\
p_8 : & UL & < \; + & \rightarrow \; +U & \\
p_9 : & U & < \; A(n,m) & \rightarrow \; D & \\
p_{10} : & F & > \; D & \rightarrow \; DF & \\
p_{11} : & D & & \rightarrow \; \varepsilon & \\
p_{12} : & L & > \; [+D] & \rightarrow \; UR & \\
p_{13} : & UR & < \; - & \rightarrow \; -U & \\
p_{14} : & R & > \; [\,][-D] & \rightarrow \; D &
\end{array}
\tag{7.8}
$$

Productions p_1 to p_3 describe the development of a simple branching structure. Starting with a single axis specified by axiom ω, the apex A appends a sequence of branch segments F to the current axis (productions p_1 and p_2), then initiates a pair of new lateral apices (production p_3) that recursively repeat the same pattern. Parameter m is used to count the derivation steps between the creation of consecutive segments F. Parameter n determines the remaining number of segments to be produced before the next branching occurs. The total number of segments in an axis is defined by constants l_L (for the main axis and the branches issued to the left) and l_R (for the branches issued to the right). A newly created branching point is marked by symbol L (production p_3).

After a delay of w steps introduced by production p_4, production p_5 places an insect U at the base of the branching structure. This insect moves upwards, one branch segment per derivation step (productions p_6 and p_7), until it encounters the branching point marker L. The insect is then directed to the left daughter branch (production p_8). After crossing a number of segments and, possibly, further branching points, the insect eventually reaches an apex A. As specified by production p_9, this apex is then removed from the structure, thus stopping further growth of its axis, and the state of the insect is changed from U (moving upwards) to D (moving downwards). The downward

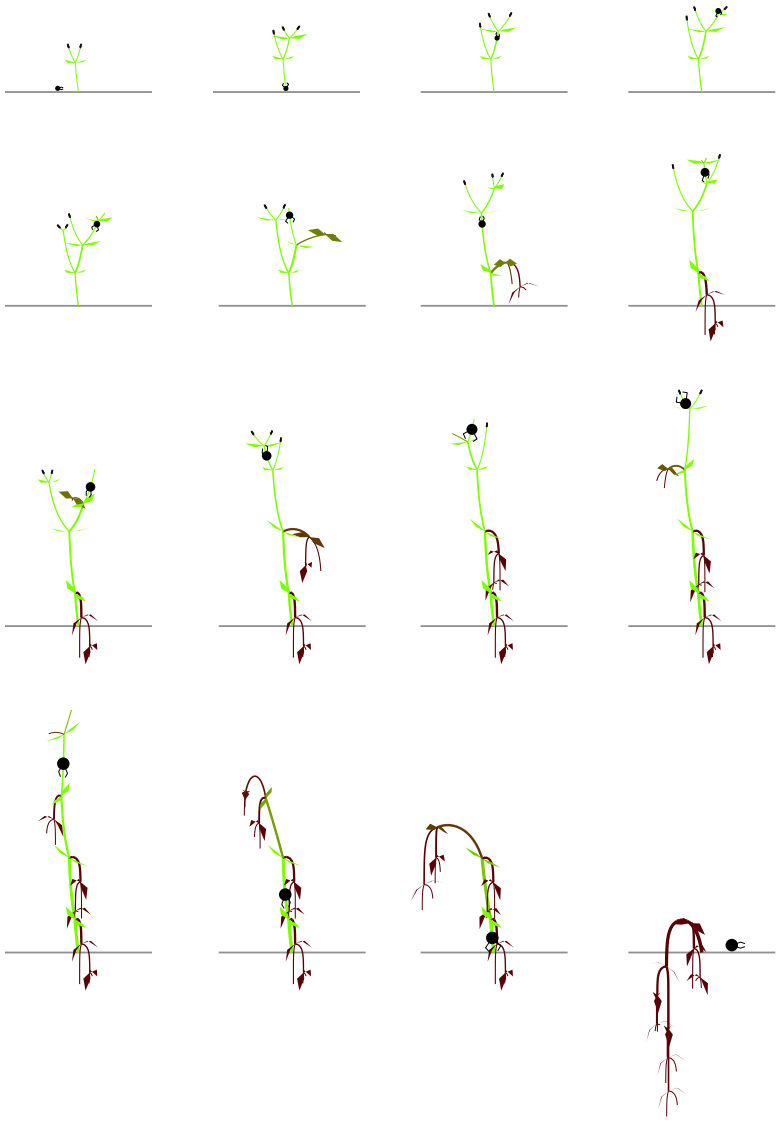


Fig. 7.5. Simulation of the development of a plant attacked by an insect

movement is simulated by productions p_{10} and p_{11}. Returning to a branching point marked L, the insect changes this marking to R to indicate that the left branch has been already explored, reverts its own state to U, and enters the right branch (productions p_{12} and p_{13}). Coming back from that branch, the insect continues its downward movement (production p_{14}) until it reaches another branching point marked L and enters an unexplored right branch, or until it completes the traversal of the entire structure at its base.

A sequence of images obtained using a straightforward extension of L-system (7.8) is shown in Figure 7.5. In this case, the insect feeds on the apices of a three-dimensional structure, and a branch that no longer carries any apices wilts.

Similar models can be constructed assuming different traversing and feeding strategies for one or many insects (which may interact with each other). Prospective applications of such models include simulation studies of insects used for weed control and of the impact of insects on crop plants [109, 110].

7.3.3 Development controlled by resource allocation. In the previous examples, discrete information was transferred between the modules of a developing structure. A signal (or insect) was either present or absent at any particular point, and affected the structure in an "all-or-nothing" manner, by removing the apices at the ends of branches. In nature, however, developmental processes are often controlled in a more modulated way, by the quantity of substances (*resources*) exchanged between the modules. For example, the growth of plants depends on the amount of water and minerals absorbed by the roots and carried acropetally, and by the amount of photosynthates produced by the leaves and transported basipetally. An early developmental model of branching structures making use of quantitative information flow was proposed by Borchert and Honda [8]. Below we restate the essence of this model using the formalism of L-systems, then we extend it to simulate interactions between the shoot and the roots in a growing plant.

Borchert and Honda postulated that the development of a branching structure is controlled by a flow or *flux* of substances, which propagate from the base of the structure towards the apices and supply them with materials needed for growth. When the flux reaching an apex exceeds a predefined threshold value, the apex bifurcates and initiates a lateral branch; otherwise it remains inactive. At branching points the flux is distributed according to the types of the supported internodes (straight or lateral) and the numbers of apices in the corresponding branches. These numbers are accumulated by messages that originate at the apices and propagate towards the base of the plant. Thus, development is controlled by a cycle of alternating acropetal and basipetal information flow.

An L-system that implements these mechanisms is given below.

$$
\begin{array}{lll}
\#\text{define } \alpha_1 & 10 & /* \text{ branching angle - straight segment } */ \\
\#\text{define } \alpha_2 & 32 & /* \text{ branching angle - lateral segment } */ \\
\#\text{define } \sigma_0 & 17 & /* \text{ initial flux } */ \\
\#\text{define } \eta & 0.89 & /* \text{ controls input flux changes } */ \\
\#\text{define } \lambda & 0.7 & /* \text{ flux distribution factor } */ \\
\#\text{define } v_{th} & 5.0 & /* \text{ threshold flux for branching } */
\end{array}
$$

ignore: $+ -/$

$$
\begin{array}{ll}
\omega : & N(1)I(0,2,0,1)A \\
p_1 : & N(k) < I(b,m,v,c) : b == 0 \;\&\&\; m == 2 \\
 & \quad \rightarrow I(b,1,\sigma_0 * 2 \wedge (k-1) * (\eta \wedge k), c) \\
p_2 : & N(k) > I(b,m,v,c) : b == 0 \;\&\&\; m == 2 \rightarrow N(k+1) \\
p_3 : & I(b,m,v,c) < A : m == 1 \;\&\&\; v > v_{th} \\
 & \quad \rightarrow /(180)[-(\alpha_2)I(2,2,v*(1-\lambda),1)A] \\
 & \quad\quad +(\alpha_1)I(1,2,v*\lambda,1)A \\
p_4 : & I(b,m,v,c) > A : m == 1 \;\&\&\; v <= v_{th} \rightarrow I(b,2,v,c) \\
p_5 : & I(b_l,m_l,v_l,c_l) < I(b,m,v,c) : m_l == 1 \;\&\&\; b == 1 \\
 & \quad \rightarrow I(b,m_l,v_l - v_l * (1-\lambda) * ((c_l - c)/c), c) \\
p_6 : & I(b_l,m_l,v_l,c_l) < I(b,m,v,c) : m_l == 1 \;\&\&\; b == 2 \\
 & \quad \rightarrow I(b,m_l,v_l * (1-\lambda) * (c/(c_l - c)), c) \\
p_7 : & I(b,m,v,c) > [I(b_2,m_2,v_2,c_2)]I(b_1,m_1,v_1,c_1) : \\
 & \quad\quad m == 0 \;\&\&\; m_1 == 2 \;\&\&\; m_2 == 2 \\
 & \quad \rightarrow I(b,2,v,c_1 + c_2) \\
p_8 : & I(b,m,v,c) : m == 1 \rightarrow I(b,0,v,c) \\
p_9 : & I(b_l,m_l,v_l,c_l) < I(b,m,v,c) : m_l == 2 \;\&\&\; m == 2 \\
 & \quad \rightarrow I(b,0,v,c)
\end{array}
\tag{7.9}
$$

This L-system operates on three types of modules: apices A, internodes I, and an auxiliary module N. The internodes are visualized as lines of unit length. Each internode has four parameters:

- **segment type** b, where 0 denotes base of the tree, 1 – a straight segment, and 2 – a lateral segment;
- **message type** m, where 0 denotes no message currently carried by the internode, 1 – an acropetal message (flux), and 2 – a basipetal message (apex count);
- **flux value** v, and
- **apex count** c.

All internodes are visualized as lines of unit length.

At the beginning of a developmental cycle, indicated by the presence of a basipetal message ($m = 2$) in the basal internode ($b = 0$), production p_1 calculates an input flux value. The expression used for this purpose, $v = \sigma_0 2^{(k-1)\eta^k}$, was introduced by Borchert and Honda to simulate a sigmoid

increase of flux penetrating the base of a plant over time. The progress of time is captured by production p_2, which increments parameter k of the module N, representing the current cycle number.

Productions p_3 and p_4 simulate acropetal flux propagation and distribute it between the straight segment and the lateral segment. If both the straight and lateral branch support the same number of apices, the straight segment will obtain a predefined fraction λ of the flux v_l reaching the branching point; the lateral segment will obtain the remainder, $(1-\lambda)v_l$. If a lateral branch supports c apices and its sister straight branch supports c_s apices, the flux reaching the lateral branch is further multiplied by the ratio c/c_s. The number c_s is not directly available to the lateral branch, but it can be calculated as the difference between the number of apices supported by this branch and its mother, $c_s = c_l - c$. In total, the flux directed towards the lateral branch is equal to $v_l(1-\lambda)(c/(c_l-c)$ (production p_3). The remaining flux reaches the straight segment. The parameter c denotes, in this case, the number of apices supported by the straight segment, and the resulting expression is $v_l - v_l(1-\lambda)((c_l-c)/c)$ (production p_4).

Productions p_5 and p_6 control the addition of new segments to the structure. According to production p_5, if the internode preceding an apex A reaches a sufficient flux $v > v_{th}$, the apex will create two new internodes I terminated by apices A. The new segments are assigned an initial message type $m = 2$, which triggers the basipetal signal propagation needed to update the count of apices supported by each segment. Alternatively, if the flux reaching an apex is not sufficient for bifurcation ($v \leq v_{th}$), the supporting internode itself starts the propagation of the basipetal signal (production p_6).

Production p_7 adds the number of apices supported by the daughter branches (c_1 and c_2), and propagates the result to the mother internode.

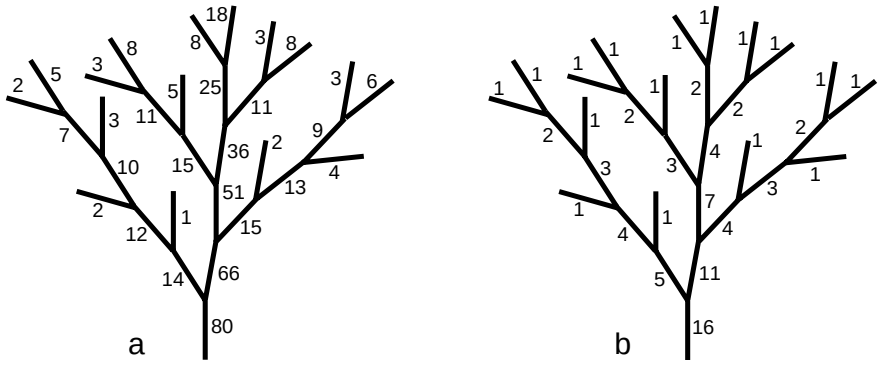


Fig. 7.6. The structure generated by L-system (7.9) at completion of the fifth developmental cycle. The numbers indicate the flow values v rounded to the nearest integer (a), and the numbers of apices c in the branches supported by each internode (b).

Both input numbers must be available ($m_1 = 2$ and $m_2 = 2$) before basipetal message propagation takes place.

The remaining productions reset the message value m to zero, after the flux values have been transferred acropetally (p_8) or the apex count has been passed basipetally (p_9).

The initial state of the model is determined by the axiom ω. The value of the parameter to module N sets the current cycle number to 1. The initial structure consists of a single internode I terminated by an apex A. The message type indicates the presence of a basipetal message ($m = 2$) which triggers the application of productions p_1 and p_2, initiating the first full developmental cycle. The state of the structure after 35 derivation steps (completion of the fifth developmental cycle) is shown in Figure 7.6.

A remarkable feature of Borchert and Honda's model is its ability to simulate the response of a plant to its environment. Specifically, after a branch has been pruned, the model redirects the fluxes to the remaining branches and accelerates their growth to compensate for the loss. A sequence of structures that illustrates this phenomenon is shown in Figure 7.7. In accordance with [8], the L-system used in this case extends L-system (7.9) with parameters and productions needed to capture the effect of aging. Consequently, a branch that was unable to grow for a given number of developmental cycles dies: it loses the ability to develop further and stops taking any fluxes.

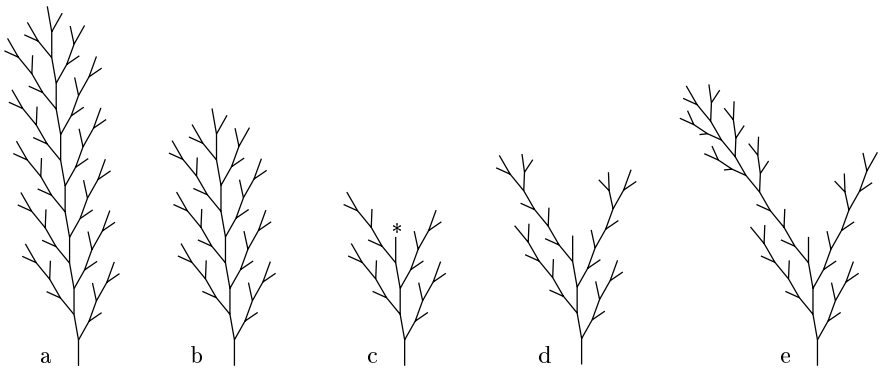


Fig. 7.7. Development of a branching structure simulated using an L-system implementation of the model by Borchert and Honda. (a) Development not affected by pruning; (b, c) the structure immediately before and after pruning; (d, e) the subsequent development of the pruned structure. Based on [8].

Similar behavior is shown in Figure 7.8. In this case, two structures representing the shoot and the root of a plant are generated simultaneously. The flux penetrating the root at the beginning of a developmental cycle is assumed to be proportional to the number of apices in the shoot; reciprocally, the flux

penetrating the shoot is proportional to the number of apices in the root. These assumptions form a crude approximation of plant physiology, whereby

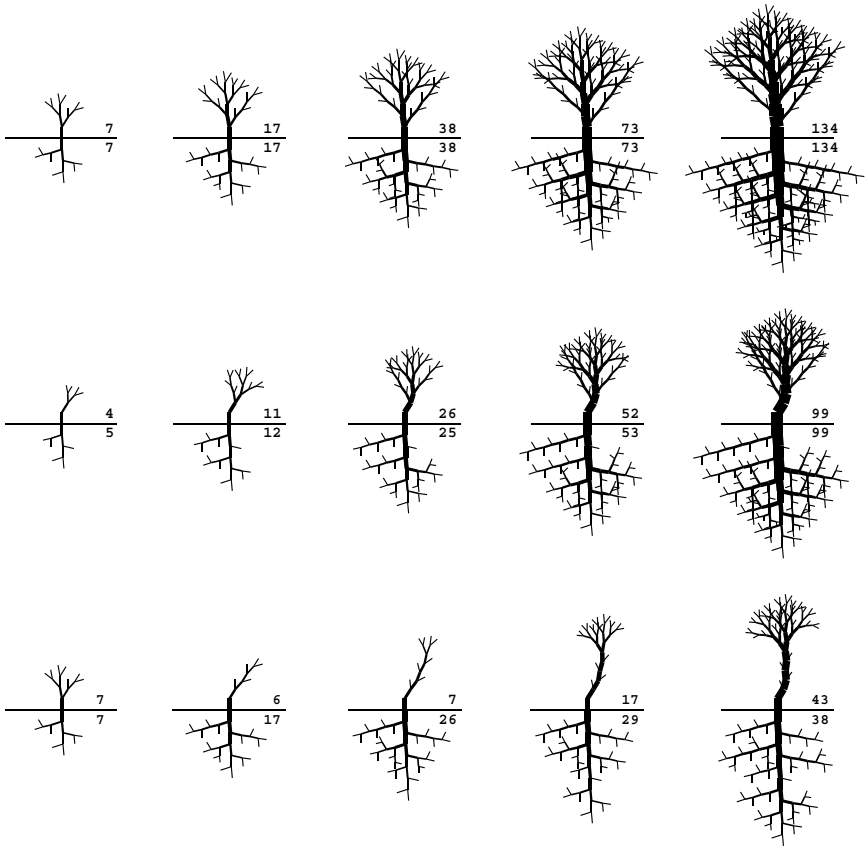


Fig. 7.8. Application of the Borchert and Honda's model to the simulation of a complete plant, showing development unaffected by pruning (top row), affected by pruning during the third cycle of development (middle row), and affected by pruning during the fifth cycle of development (bottom row). The numbers of live apices in the shoot and root are indicated above and below the ground level. The numbers at the base of the figure indicate the number of completed developmental cycles.

the photosynthates produced by the shoot fuel the development of the root, and water and mineral compounds gathered by the root are required for the development of the shoot. The model also captures an increase of internode width over time, and a gradual assumption of the position of a straight segment by its sister lateral segment, after the straight segment has been lost. The developmental sequence shown in the top row of Figure 7.8 is unaffected

by pruning. The shoot and the root develop in concert. The next two rows illustrate development affected by a loss of branches. The removal of a shoot branch slows down the development of the root; on the other hand, the large size of the root, compared to the remaining shoot, fuels a fast re-growth of the shoot. Eventually, the plant is able to redress the balance between the size of the shoot and the root. Damage occurring at an early stage of plant development (middle row) had a less pronounced effect than the loss of a branch at a later stage (bottom row).

The last two examples leave open the question of incorporating environmental factors such as pruning into L-system models. We consider this question in the next section.

8. Development controlled by exogenous mechanisms

8.1 Plants and their environment

The environment is a key factor affecting life cycles of plants and plant communities. Consequently, the role of the environment in plant development is an important area of study for both theoretical and practical reasons (such as the maximization of crop yield and landscape design). In general, descriptions of plant interactions with the environment may take the following forms:

- Plant is affected by global properties of the environment, such as day length, temperature, or air pollution;
- Plant is affected by local properties of the environment, such as support for climbing plants, mechanical obstacles, soil composition, and access to light during colonizing growth;
- Plant interacts with the environment in a feedback loop, which includes bi-directional information flow to and from the environment. Examples include competition for light between branches of a tree (where the upper branches change the amount of light available to the lower branches) and interaction of roots with the soil (taking into account the impact of roots on the transport of nutrients and water in the soil).

Although some phenomena belong quite naturally to one of these groups, the classification of others may depend on the level of abstraction. For example, an approximate model may consider temperature as a global property of the environment, a more detailed one may express temperature locally as a function of distance from the ground, and a yet more detailed model may take into account the changes of temperature determined by the distribution of radiative energy between plant parts. Thus, the above classification is useful primarily from the modeling perspective, since different techniques are required to capture phenomena in each class.

Within the L-system theory, plants were originally regarded as closed cybernetic systems, capable of controlling their development without communicating with the environment. This assumption made it possible to characterize some developmental processes in the form of a mathematical (deductive) theory, with clearly stated assumptions, theorems, and proofs. Unfortunately, the abstraction from environmental factors reduced the scope of this theory, because the environment has a significant impact on many developmental processes. In the first step towards the inclusion of environmental factors, Rozenberg defined *table L-systems*, which allow for changing the production set from one derivation step to another [111] (see also [46, 113]). Table L-systems were applied, for example, to capture the switch from the production of leaves to the production of flowers by an apex of a flowering plant, due to a change in day length [27, 29, 30]. Parametric L-systems make it possible to introduce a variant of this technique, where the environment affects selected numerical values used in productions. In a case study illustrating this approach, weather data containing daily minimum and maximum temperatures control a developmental model of bean [40].

Table L-systems and their extensions can only capture the impact of global environmental characteristics on plant development. The generated strings are not interpreted geometrically until the derivation is completed, thus no information regarding position and orientation of individual modules is available during the rewriting process. Below we describe the *environmentally-sensitive* extension of L-systems, which makes this information available in each derivation step. Therefore, it is possible to model the influence of local environmental factors on a growing plant. Our presentation closely follows the paper [97].

8.2 Environmentally-sensitive L-systems

In environmentally-sensitive L-systems, the generated string is interpreted after each derivation step, and turtle attributes found during the interpretation are returned as parameters to reserved *query modules* in the string. Each derivation step is performed as in parametric L-systems, except that the parameters associated with the query modules remain undefined. During interpretation, these modules are assigned values that depend on the turtle's position and orientation in space. Syntactically, the query modules have the form $?X(x, y, z)$, where $X = P, H, U$, or L. Depending on the actual symbol X, the values of parameters x, y, and z represent a position or an orientation vector. In the two-dimensional case, the coordinate z may be omitted.

The operation of the query module is illustrated by a simple environmentally-sensitive L-system, given below.

$$\begin{array}{lllll} \omega: & A & & & \\ p_1: & A & \rightarrow & F(1)?P(x,y) - A & \\ p_2: & F(k) & \rightarrow & F(k+1) & \end{array} \tag{8.1}$$

The following strings are produced during the first three derivation steps.

$$\begin{array}{ll}
\mu'_0: & A \\
\mu_0: & A \\
\mu'_1: & F(1)?P(\star,\star) - A \\
\mu_1: & F(1)?P(0,1) - A \\
\mu'_2: & F(2)?P(\star,\star) - F(1)?P(\star,\star) - A \\
\mu_2: & F(2)?P(0,2) - F(1)?P(1,2) - A \\
\mu'_3: & F(3)?P(\star,\star) - F(2)?P(\star,\star) - F(1)?P(\star,\star) - A \\
\mu_3: & F(3)?P(0,3) - F(2)?P(2,3) - F(1)?P(2,2) - A
\end{array} \tag{8.2}$$

Strings μ'_0, μ'_1, μ'_2, and μ'_3 represent the axiom and the results of production application before the interpretation steps. Symbol $\star$ indicates an undefined parameter value in a query module. Strings μ_1, μ_2, and μ_3 represent the corresponding strings after interpretation. It has been assumed that the turtle is initially placed at the origin of the coordinate system, vector **H** is aligned with the y axis, vector **L** points in the negative direction of the x axis, and the angle of rotation associated with module "$-$" is equal to 90°. Parameters of the query modules have values representing the positions of the turtle shown in Figure 8.1.

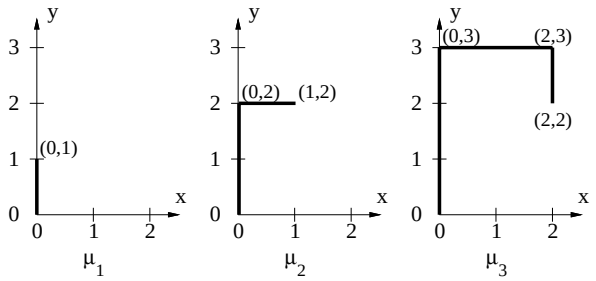


Fig. 8.1. Assignment of values to query modules. From [97].

The above example illustrates the notion of derivation in an environmentally-sensitive L-system, but it is otherwise contrived, since the information returned by the query modules is not further used. An example of an abstract developmental process influenced by the environment is given below.

$$\begin{array}{lll}
\omega: & A & \\
p_1: & A \rightarrow & [+B][-B]F?P(x,y)A \\
p_2: & B \rightarrow & F?P(x,y)@OB \\
p_3: & ?P(x,y) : 4x^2 + (y-10)^2 > 10^2 & \\
 & \rightarrow & [+(2y)F][-(2y)F]\%
\end{array} \tag{8.3}$$

Module F represents a line of unit length, and modules $+$ and $-$ without parameters represent left and right turns of 60°. The development begins

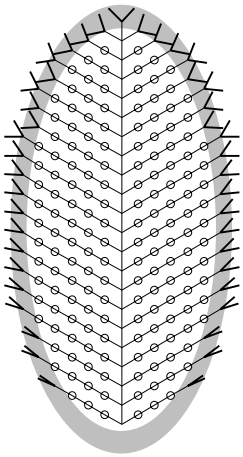


Fig. 8.2. A branching structure pruned to an ellipse. From [97].

with module A, which creates a sequence of opposite branches $[+B][-B]$ separated by internodes (branch segments) F (production p_1). The branches elongate by addition of segments F, delimited by markers $@O$ (production p_2). Both the main apex A and the lateral apices B create query modules $?P(x, y)$, which return the corresponding turtle positions. If a query module is placed beyond the ellipse $4x^2 + (y - 10)^2 = 10^2$, production p_3 creates a pair of "tentacles," represented by the substring $[+(2y)F][-(2y)F]$. The angle $4y$ between these tentacles depends on the vertical position y of the query module. Production p_3 also inserts cutting symbol %, which terminates branch growth by removing its apex. In summary, L-system 8.3 produces a branching structure confined to an ellipse, with tentacles placed at the boundary of the structure, and the angle between the tentacles depending on the turtle's position in space, as shown in Figure 8.2.

8.3 Examples

8.3.1 A deterministic model of plant response to pruning. As described, for example, by Hallé *et al.* [39, Chapter 4] and Bell [5, page 298], during the normal development of a tree many buds do not produce new branches and remain dormant. These buds may be subsequently activated by the removal of leading buds from the branch system (*traumatic reiteration*), which results in an environmentally-adjusted tree architecture. The following L-system represents the extreme case of this process, where buds are activated only as a result of pruning.

$$
\begin{array}{ll}
\omega: & FA?P(x,y) \\
p_1: & A > ?P(x,y): \ !\text{prune}(x,y) \rightarrow @OF/(180)A \\
p_2: & A > ?P(x,y): \ \ \text{prune}(x,y) \rightarrow T\% \\
p_3: & F > T \rightarrow S \\
p_4: & F > S \rightarrow SF \\
p_5: & S \rightarrow \varepsilon \\
p_6: & @O > S \rightarrow [+FA?P(x,y)]
\end{array}
\tag{8.4}
$$

The user-defined function

$$\text{prune}(x,y) = (x < -L/2) \| (x > L/2) \| (y < 0) \| (y > L), \tag{8.5}$$

specifies a square *clipping box* of dimensions $L \times L$ that bounds the growing structure. According to axiom ω, the development begins with an internode F supporting apex A and query module $?P(x,y)$. The initial development of the structure is described by production p_1. In each step, the apex A creates a dormant bud $@O$ and an internode F. The module $/(180)$ rotates the turtle around its own axis (the heading vector $\mathbf{H}$), thus laying a foundation for an alternating branching pattern. The query module $?P(x,y)$, placed by the axiom, is the right context for production p_1 and returns the current position of apex A. When a branch extends beyond the clipping box, production p_2 removes apex A, cuts off the query module $?P(x,y)$ using the symbol %, and generates the pruning signal T. In the presence of this signal, production p_3 removes the last internode of the branch that extends beyond the clipping box and creates bud-activating signal S. Productions p_4 and p_5 propagate this signal basipetally (downwards), until it reaches a dormant bud $@O$. Production p_6 induces this bud to initiate a lateral branch consisting of internode F and apex A followed by query module $?P(x,y)$. According to production p_1, this branch develops in the same manner as the main axis. When its apex extends beyond the clipping box, it is removed by production p_2, and signal S is generated again. This process may continue until all dormant buds have been activated.

Selected phases of the described developmental sequence are illustrated in Figure 8.3. In derivation step 6 the apex of the main axis grows out of the clipping box. In step 7 this apex and the last internode are removed from the structure, and the bud-activating signal S is generated. As a result of bud activation, a lateral branch is created in step 8. As it also extends beyond the bounding box, it is removed in step 9 (not shown). Signal S is generated again, and in step 10 it reaches a dormant bud. The subsequent development of the lateral branches, shown in the middle and bottom rows of Figure 8.3, follows a similar pattern.

8.3.2 A stochastic model of tree response to pruning. L-system (8.4) simulates plant response to pruning using a schematic branching structure. Below we incorporate a similar mechanism into the more realistic stochastic tree model by Borchert and Slade, discussed in Section 5.3.

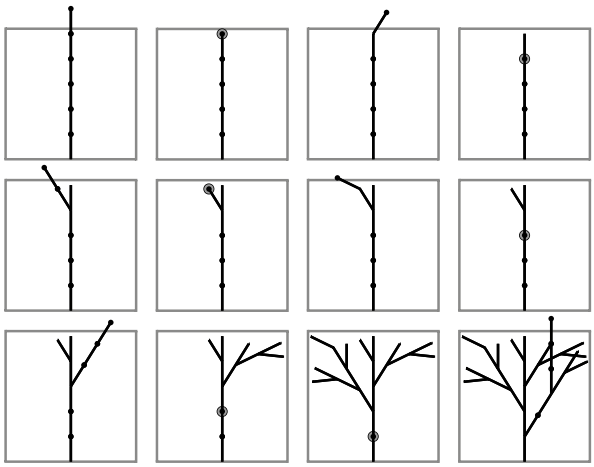


Fig. 8.3. A simple model of a tree's response to pruning. Top row: derivation steps 6,7,8, and 10; middle row: steps 12, 13, 14, and 17; bottom row: steps 20, 40, 75, and 94. Small black circles indicate dormant buds, the larger circles indicate the position of signal S. From [97].

$$
\begin{array}{ll}
\omega: & FA(1)?P(x,y,z) \\
p_1: & A(k) > ?P(x,y,z) : \ !\text{prune}(x,y,z) \rightarrow \\
 & \quad /(\phi)[+(\alpha)FA(k+1)?P(x,y,z)] - (\beta)FA(k+1) : \\
 & \qquad\qquad \min\{1,(2k+1)/k^2\} \\
p_2: & A(k) > ?P(x,y,z) : \ !\text{prune}(x,y,z) \rightarrow \\
 & \quad /(\phi)B(k+1,k+1) - (\beta)FA(k+1) : \\
 & \qquad\qquad \max\{0,1-(2k+1)/k^2\} \\
p_3: & A(k) > ?P(x,y,z) : \ \ \text{prune}(x,y,z) \rightarrow T\% \\
p_4: & F > T \rightarrow S \\
p_5: & F > S \rightarrow SF \\
p_6: & S \rightarrow \epsilon \\
p_7: & B(m,n) > S \rightarrow [+(\alpha)FA(am+bn+c)?P(x,y,z)] \\
p_8: & B(m,n) > F \rightarrow B(m+1,n)
\end{array}
\tag{8.6}
$$

According to axiom ω, the development begins with a single internode F supporting apex A and query module $?P(x,y,z)$. Productions p_1 and p_2 describe the spontaneous growth of the tree within the volume characterized by a user-defined clipping function $\text{prune}(x,y,z)$. Productions p_3 to p_7 specify the mechanism of the tree's response to pruning. Specifically, production p_3 removes the apex A after it has crossed the clipping surface, cuts off the query module $?P(x,y,z)$, and creates pruning signal T. Next, p_4 removes the last internode of the pruned branch and initiates bud-activating signal S, which is propagated basipetally by productions p_5 and p_6. When S reaches a dormant bud B, production p_7 transforms it into a branch consisting of an internode F, apex A, and query module ?P(x,y,z).

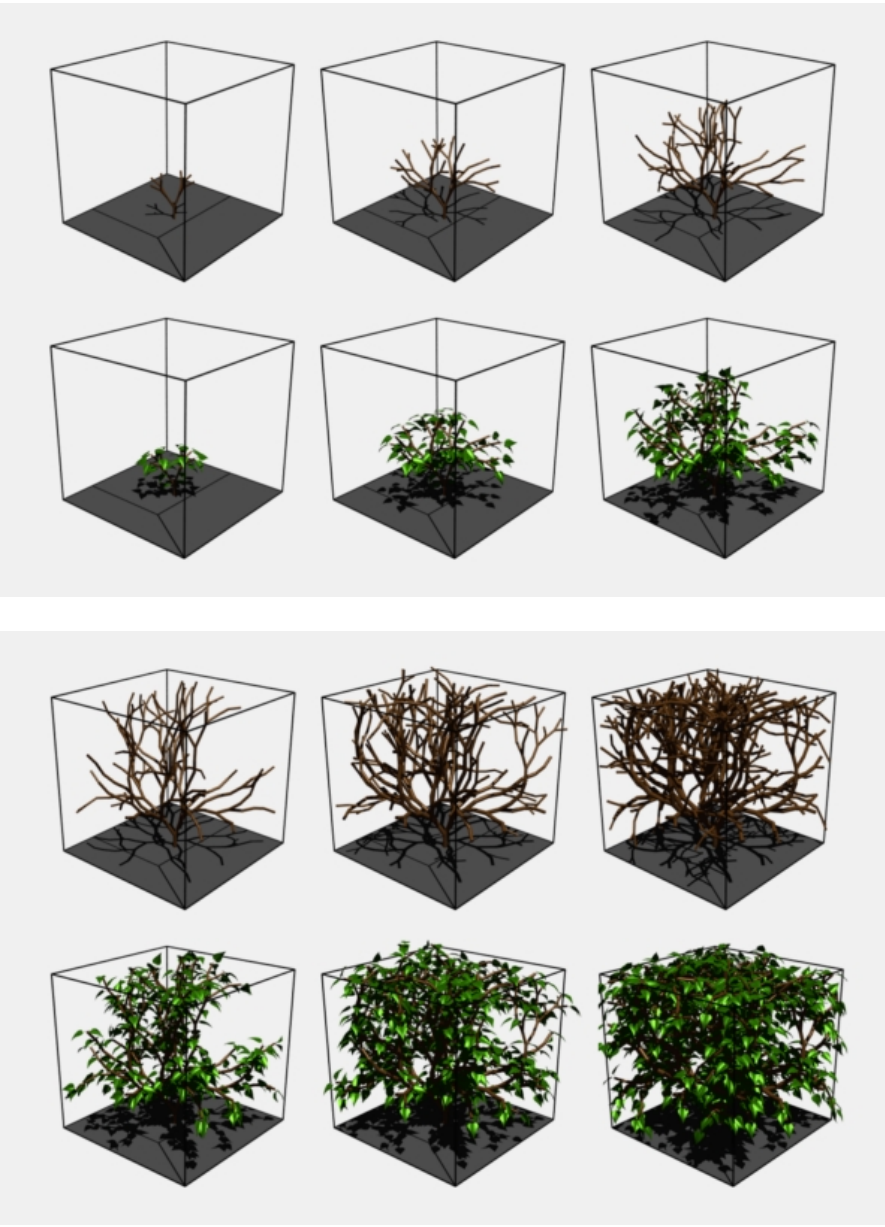


Fig. 8.4. Simulation of tree response to pruning. The structures shown have been generated in 3, 6, 9, 13, 21, and 27 steps. From [97].

The parameter value assigned by production p_7 to apex A is derived as follows. According to production p_2, both parameters associated with a newly created bud B are set to the age of the tree at the time of bud creation (expressed as the the number of derivation steps). Production p_8 updates the value of the first parameter (m), so that it always indicates the actual age of the tree. The second parameter (n) remains unchanged. The initial *biological age* [5, page 315] of the activated apex A in production p_7 is a linear combination of parameters m and n, calculated using the expression $am + bn + c$. Since rule p_1 is more likely to be applied for young apices (for small values of parameter k), by manipulating constants a, b, and c it is possible to control the bifurcation frequency of branches created as a result of traumatic reiteration. This is an important feature of the model, because in nature the reiterated branches tend to be more juvenile and vigorous than the remainder of the tree [5, page 298].

The operation of this model is illustrated in Figure 8.4. The clipping form is a cube with an edge length 12 times longer than the internode length. The constant values used in production p_7 are $a = 0$, $b = 1$, and $c = -5$.

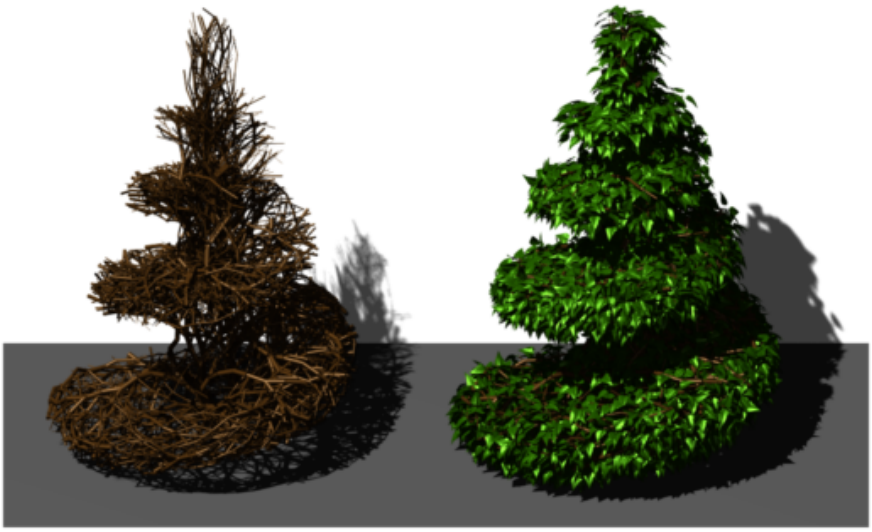


Fig. 8.5. Trees pruned to a spiral shape. From [97].

By changing the clipping function, one can shape plant models to a variety of artificial forms. For example, the trees shown in Figure 8.5 were pruned to a spiral shape. Figure 8.6 combines trees pruned to a variety of shapes into a synthetic image of a topiary garden, inspired by the Levens Hall garden in England [13, pages 52–57]. For other models of topiary trees see [97].

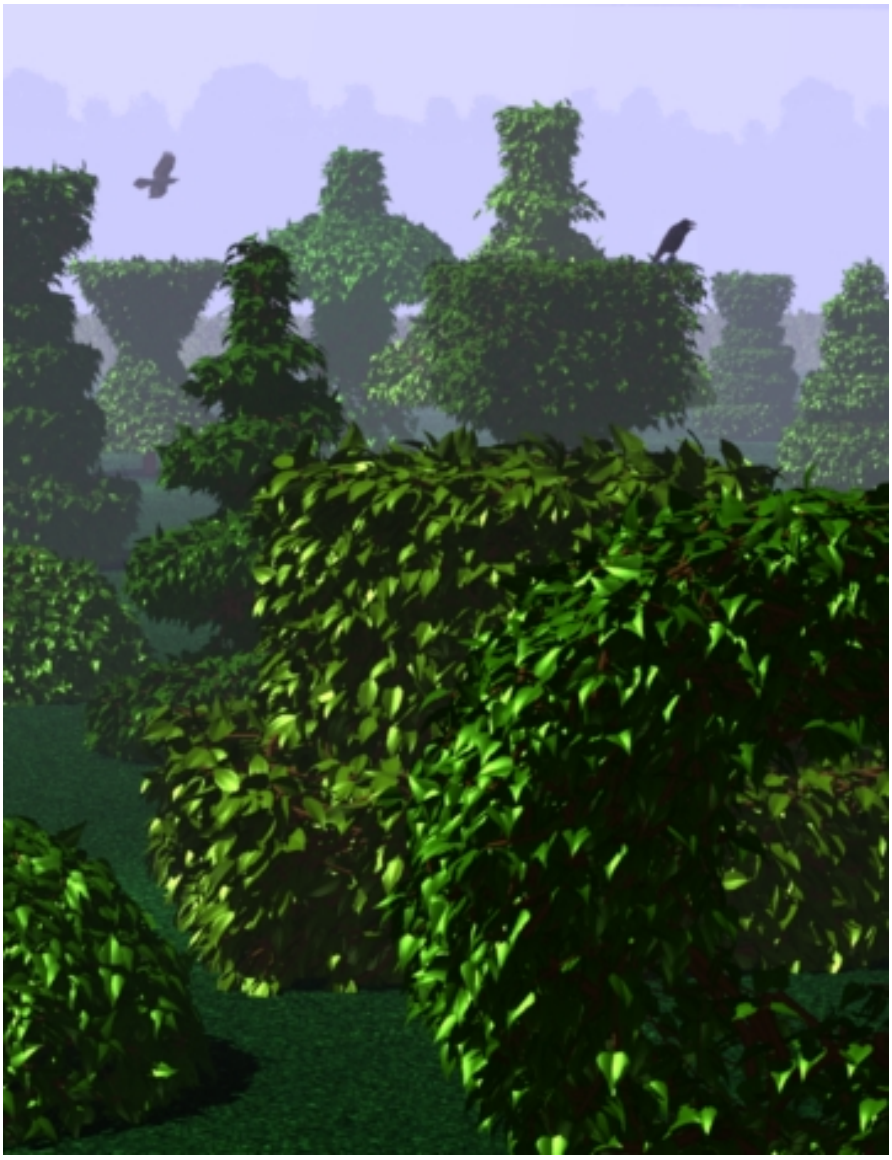


Fig. 8.6. A model of the topiary garden at Levens Hall, England

8.3.3 Plant climbing. Another example of environmental influences on plant development is presented in Figure 8.7. Here, a hypothetical climbing (twining) plant detects the presence of a supporting pole and winds around it. In contrast to the earlier models of plants growing around obstacles [2, 33, 34], the L-system model captures the *nutation*, or spiraling movement, of the free stem tip searching for support [44].

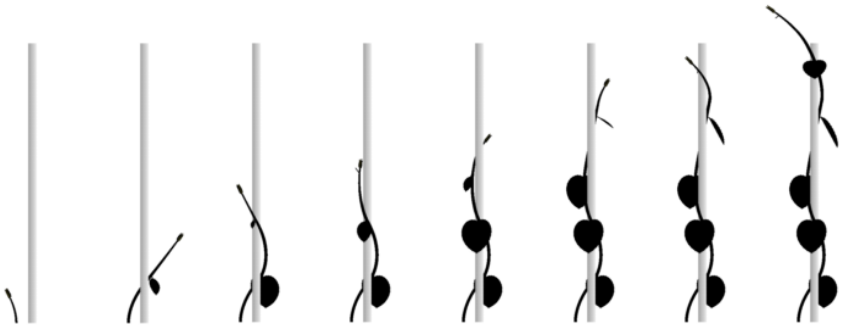


Fig. 8.7. A simple model of a climbing plant

8.3.4 Directional cues in development. In the previous examples, the development of plant was influenced by the *position* of query modules associated with plant apices in space. The formalism of environmentally-sensitive L-systems makes it also possible to query the *orientation* of the turtle at specific points, as illustrated by the simple L-system below.

$$\begin{array}{ll} \omega: & X \\ p_1: & X \rightarrow [A?H(x,y,z)] + (10)X \\ p_2: & A > ?H(x,y,z) \rightarrow F(1 + 0.5 * y) \end{array} \tag{8.7}$$

Beginning with the axiom X, production p_1 generates a sequence of apices A that spread radially from the origin of the coordinate system. Each apex is followed by a query module $?H$ returning the orientation of the turtle's heading vector (Section 8.2). Production p_2 uses the vertical component of this vector, represented by parameter y, to determine the length of the branch F created by its apex A. The vector $\mathbf{H}$ is normalized (*c.f.* Section 3.3), thus the branch length returned by the expression $1 + 0.5 * y$ in production p_1 decreases from 1.5 for branches pointing up to 0.5 for branches pointing down. A resulting structure, generated in 37 derivation steps, is shown in Figure 8.8.

As the structure generated in this example is very simple, the orientation of each branch could also have been determined directly from the form of productions, without using query modules. In the case of three-dimensional

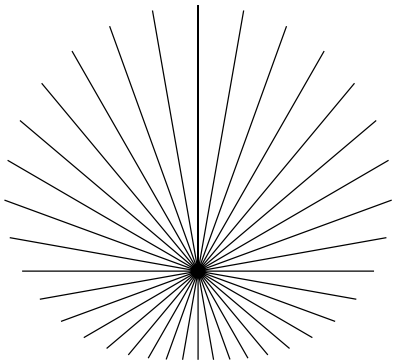


Fig. 8.8. A structure with the length of branches determined by their orientation in space

branching structures, however, determining the turtle's orientation without queries would be much more difficult.

An example of a three-dimensional model making use of directional information is presented next. It extends L-system (4.7), generating simple branching structures discussed in Section 4.2.3, by modifying the length of internodes according to their orientation. This extension is justified by the description of tree architectures presented by Ward [129, Chapter VI] (see also [132, Chapter 4]), who pointed out that considerable differences in tree form may result from "throwing the energy of growth" either towards the inward or outward growing branches.

$$\begin{array}{lll} \omega & : & A(100, w_0)?H(0,0,0) \\ p_1 & : & A(s,w) \;>\; ?H(x,y,z) \;\rightarrow\; !(w)F(s*(a-b*y)) \\ & & [+(\alpha_1)/(\varphi_1)A(s*r_1, w*q \wedge e)?H(x_1,y_1,z_1)] \\ & & [+(\alpha_2)/(\varphi_2)A(s*r_2, w*(1-q) \wedge e)?H(x_2,y_2,z_2)] \end{array} \tag{8.8}$$

According to production p_1, the default length s of an internode F produced by an apex A is multiplied by the expression $a - b * y$, which modifies the internode length according to the orientation of the apex. As a result, for $b > 0$, the internodes growing upwards are shorter than those growing horizontally or downwards. Figure 8.9 illustrates the impact of this modification on the appearance of the final structure using values $a = 1.5$ and $b = 0.7$ (left) or 1.0 (right). The remaining constants are specified in Table 8.1. The turtle

Table 8.1. The values of constants used to generate Figure 8.9

r_1	r_2	α_1	α_2	φ_1	φ_2	w_0	q	e	n
.60	.85	25	-10	90	-90	20	.50	.30	14

Fig. 8.9. Two branching structures generated by L-system (8.8). The internodes pointing up are shortened with respect to the horizontal and drooping ones to a lesser (left) and larger (right) extent.

orientation is biased downwards (*i.e.,* the turtle's heading vector is slightly turned downwards at each node of the branching structure) to simulate the weeping habit of the modeled trees. Details of this technique are described under the general name of *tropisms* in [94, 99, 101].

9. Conclusions

Plants can be modeled using the frameworks offered by different branches of science. For instance, biomechanical models emphasize physical entities, such as force, mass, and stress (*c.f.* [25, 84, 86]), genetic models are inherently rooted in organic chemistry, and many architectural models are based on statistical analysis of observational data [7, 53, 18, 106]. In this context, L-system models, which emphasize the role of information flow in growing structures, represent an approach rooted in computer science [35, 36].

L-system models integrate local processes, taking place at the level of individual modules, into developmental patterns and structures of entire plants. Consequently, they address the central problem of morphogenesis: the description and understanding of mechanisms through which living organisms acquire their form. This aspect of modeling motivated the original biological applications of L-systems investigated by Lindenmayer and his collaborators, is the main thread of the examples included in this chapter, and plays an important role in current biological research using L-systems. The organisms that have been studied by various researchers range from algae and fungi (see [15, 32, 64, 65, 83, 117, 118, 125] for recent results) to herbaceous plants and trees.

In addition to theoretical studies, L-systems are being introduced to applied plant science. As pointed out by Thornely and Johnson [124, page viii], "it is only a matter of time before morphological features are included more explicitly in plant and crop models." Integration of L-system models with crop models has been addressed by Guzy [37] and Hanan [40]. A faster acceptance of L-system models is impeded by the difficulties in acquiring the large amounts of architectural data needed to construct models precise enough for practical applications [110]. Nevertheless, several digital instruments facilitating the measurement of plants exist [85, 108], and the process of model construction according to measurements is being worked out [62, 102].

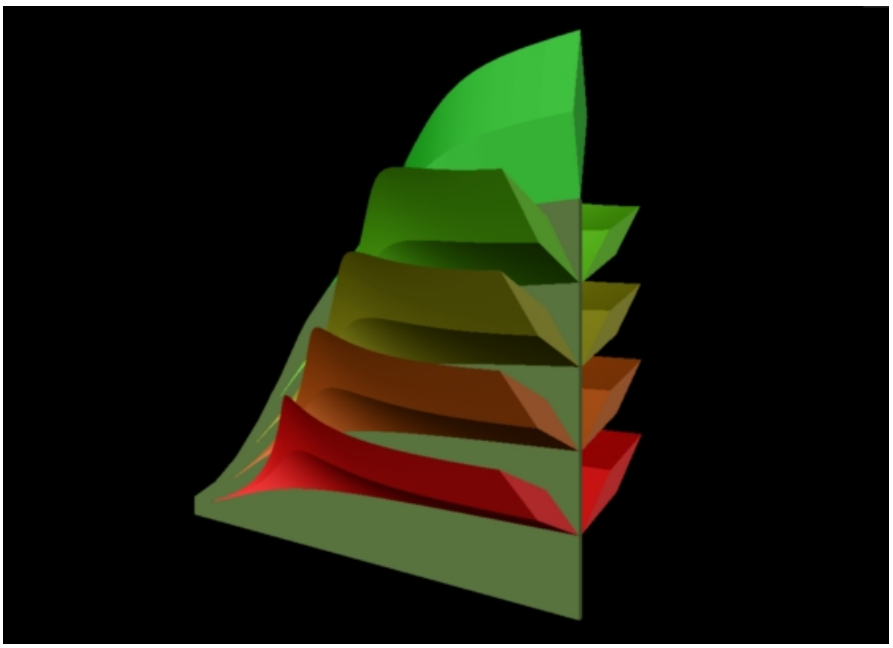


Fig. 9.1. Continuous-time development of a planar structure visualized as an object in three-dimensional space-time. Intersections of the object shown with planes parallel to the front surface represent developmental stages of a pinnate (green ash) leaf. As the plane is swept from the back to the front, consecutive pairs of leaflets separate one after another from the leaf axis and grow until the mature leaf form is reached. Based on data in [102].

At present, the primary use of L-systems in biological applications is as a foundation for simulation languages and software. This role can be compared to the manner in which Chomsky grammars provide a theoretical foundation for sequential programming languages. The strong link between L-systems and simulation was indicated in Lindenmayer's original paper [67], and led to several simulation programs dating back to CELIA [3]. In many implementations the syntax of the modeling language is similar to that used in

this chapter. Alternatives include the specification of models in programming languages such as Simula [47], C [96], and C++ [37], implementation of L-systems in *Mathematica* [52], and design of a special-purpose object-oriented language [10]. Other comprehensive implementations of L-system-based modeling programs have been reported in [61, 63]. At a conceptual level, Hogeweg explored L-systems as a paradigm for discrete-event simulation [47, 48], and Prusinkiewicz *et al.* transformed it into a combined discrete-continuous paradigm by including differential equations as a part of L-system models (*differential L-systems* [93]). An example of a simulation carried out in continuous time using a differential L-system is presented in Figure 9.1.

Applications of L-systems have inspired many more extensions to the basic formalism, such as table mechanism, fragmentation, and the introduction of environmental sensitivity. One extension requiring further research is the bi-directional flow of information between a plant and its environment. There are many phenomena that rely on such a flow. For example, the development of roots is affected by the availability of water and nutrients in the soil, but roots also affect this distribution by absorbing the needed substances from the soil [12]. Similarly, the local availability of light affects the development of tree crowns, but the crown also affects this distribution as the upper branches cast shadow on the lower ones [123]. Figure 9.2 illustrates work in progress on an L-system tree model that can be placed in an environment simulating light propagation from the sun towards the leaves. Branches in the shade grow more slowly and eventually die off. Related results concerning the effect of apex temperature have been described by Fournier [24].

In principle, the mathematical formulation of L-systems should make it possible to address biologically relevant questions in the form of a deductive theory of plant development. The results of this theory could be potentially more general than simulations, which are inherently limited to case studies (*c.f.* [126]). Unfortunately, construction of such a theory still seems quite remote. One reason is the lack of a precise mathematical description of plant form. This is not of crucial importance in simulations, where the results are evaluated visually, but impedes the formulation of theorems and proofs. Another difficulty is the discrepancy between studies on the theory of L-systems and the needs of biological modeling. Most theoretical results are pertinent to non-parametric 0L-systems operating on non-branching strings without geometric interpretation (for examples, see [113]). In contrast, (as illustrated in this chapter) L-system models of biological phenomena often involve parameters, endogenous and exogenous interactions, and geometric features of the modeled structures. We hope that the further development of L-system theory will bridge this gap. For a recent study see [98].

L-systems provide a powerful framework for expressing, simulating, visualizing, and formally reasoning about biological mechanisms that control plant development. Ultimately, however, the place of L-systems in biology will be determined by the soundness of the data and hypotheses that consti-

Fig. 9.2. A tree model sensitive to the local light environment

tute the foundation for specific models, and their predictive value [22]. We believe that in the near future we will witness the formulation of models that provide solutions to mainstream questions of plant development with both conceptual and practical value.

10. Acknowledgements

This chapter incorporates edited versions of publications written with several co-authors. In this context, we wish to acknowledge contributions by the late Professor Aristid Lindenmayer, and by Lila Kari and Mark James. Dale Brisinda implemented the L-system model of acrotonic structures discussed in Section 4.2.4. Many interesting ideas resulted from discussions with Peter Room; in particular, the idea of using L-systems to simulate the interactions between plants and insects belongs to him. We also acknowledge insights from discussions with Bruno Andrieu, Johannes Battjes, Campbell Davidson, Art Diggle, Michael Guzy, Hermann and Jaqueline Lück, Bruno Moulia, Bill Remphrey, and Grzegorz Rozenberg. Lynn Mercer and Chris Prusinkiewicz provided many editorial comments. The reported research has been sponsored by grants and graduate scholarships from the Natural Sciences and Engineering Research Council of Canada.

References

1. H. Abelson and A. A. diSessa. *Turtle geometry*. M.I.T. Press, Cambridge, 1982.
2. J. Arvo and D. Kirk. Modeling plants with environment-sensitive automata. In *Proceedings of Ausgraph'88*, pages 27 – 33, 1988.
3. R. Baker and G. T. Herman. Simulation of organisms using a developmental model, parts I and II. *Int. J. of Bio-Medical Computing*, 3:201–215 and 251–267, 1972.
4. P. W. Barlow. Meristems, metamers and modules and the development of shoot and root systems. *Botanical Journal of the Linnean Society*, 100:255–279, 1989.
5. A. Bell. *Plant form: An illustrated guide to flowering plants*. Oxford University Press, Oxford, 1991.
6. A. D. Bell. The simulation of branching patterns in modular organisms. *Philos. Trans. Royal Society London, Ser. B*, 313:143–169, 1986.
7. A. D. Bell, D. Roberts, and A. Smith. Branching patterns: the simulation of plant architecture. *Journal of Theoretical Biology*, 81:351–375, 1979.
8. R. Borchert and H. Honda. Control of development in the bifurcating branch system of *Tabebuia rosea*: A computer simulation. *Botanical Gazette*, 145(2):184–195, 1984.
9. R. Borchert and N. Slade. Bifurcation ratios and the adaptive geometry of trees. *Botanical Gazette*, 142(3):394–401, 1981.
10. I. A. Borovikov. L-systems with inheritance: an object-oriented extension of L-systems. *ACM SIGPLAN Notices*, 30(5):43–60, 1995.

11. T. W. Chien and H. Jürgensen. Parameterized L systems for modelling: Potential and limitations. In G. Rozenberg and A. Salomaa, editors, *Lindenmayer systems: Impacts on theoretical computer science, computer graphics, and developmental biology*, pages 213–229. Springer-Verlag, Berlin, 1992.
12. V. Clausnitzer and J.W. Hopmans. Simultaneous modeling of transient three-dimensional root growth and soil water flow. *Plant and Soil*, 164:299–314, 1994.
13. P. Coats. *Great gardens of the Western world*. G. P. Putnam's Sons, New York, 1963.
14. G. A. Constable. Mapping the production and survival of fruit on field grown cotton. *Agronomy Journal*, 83(2):374–378, 1991.
15. J. D. Corbit and D. J. Garbary. Computer simulation of the morphology and development of several species of seaweed using Lindenmayer systems. *Computers and Graphics*, 17(1):85–88, 1993.
16. M. J. M. de Boer. *Analysis and computer generation of division patterns in cell layers using developmental algorithms*. PhD thesis, University of Utrecht, 1989.
17. M. J. M. de Boer, F. D. Fracchia, and P. Prusinkiewicz. A model for cellular development in morphogenetic fields. In G. Rozenberg and A. Salomaa, editors, *Lindenmayer systems: Impacts on theoretical computer science, computer graphics, and developmental biology*, pages 351–370. Springer-Verlag, Berlin, 1992.
18. P. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech. Plant models faithful to botanical structure and development. Proceedings of SIGGRAPH '88 (Atlanta, Georgia, August 1–5, 1988), in *Computer Graphics* 22, 4 (August 1988), pages 151–158, ACM SIGGRAPH, New York, 1988.
19. F. M. Dekking. Recurrent sets. *Advances in Mathematics*, 44(1):78–104, 1982.
20. F. M. Dekking. Recurrent sets: A fractal formalism. Report 82-32, Delft University of Technology, 1982.
21. P. Eichhorst and W. J. Savitch. Growth functions of stochastic Lindenmayer systems. *Information and Control*, 45:217–228, 1980.
22. J. B. Fisher. How predictive are computer simulations of tree architecture. *International Journal of Plant Sciences*, 153 (Suppl.):137–146, 1992.
23. J. D. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer graphics: Principles and practice*. Addison-Wesley, Reading, 1990.
24. C. Fournier. Introduction des réponses écophysiologiques à la temérature dans un modèle de plante à la base de L-Systèmes. Master's thesis, Institut National Agronomique Paris-Grignon, 1995.
25. M. Fournier, H. Bailleres, and B. Chanson. Tree biomechanics: growth, cumulative prestress, and reorientations. *Biomimetics*, 2(3):229–251, 1994.
26. F. D. Fracchia, P. Prusinkiewicz, and M. J. M. de Boer. Animation of the development of multicellular structures. In N. Magnenat-Thalmann and D. Thalmann, editors, *Computer Animation '90*, pages 3–18, Tokyo, 1990. Springer-Verlag.
27. D. Frijters. An automata-theoretical model of the vegetative and flowering development of *Hieracium murorum* L. *Biological Cybernetics*, 24:1–13, 1976.
28. D. Frijters. Mechanisms of developmental integration of *Aster novae-angliae* L. and *Hieracium murorum* L. *Annals of Botany*, 42:561–575, 1978.
29. D. Frijters. Principles of simulation of inflorescence development. *Annals of Botany*, 42:549–560, 1978.
30. D. Frijters and A. Lindenmayer. A model for the growth and flowering of *Aster novae-angliae* on the basis of table (1,0)L-systems. In G. Rozenberg and A. Salomaa, editors, *L Systems*, Lecture Notes in Computer Science 15, pages 24–52. Springer-Verlag, Berlin, 1974.

31. D. Frijters and A. Lindenmayer. Developmental descriptions of branching patterns with paracladial relationships. In A. Lindenmayer and G. Rozenberg, editors, *Automata, languages, development*, pages 57–73. North-Holland, Amsterdam, 1976.
32. D. J. Garbary and J. D. Corbit. Lindenmayer-systems as models of red algal morphology and development. *Progress in Phycological Research*, 8:143–177, 1992.
33. N. Greene. Organic architecture. SIGGRAPH Video Review 38, segment 16, ACM SIGGRAPH, New York, 1988.
34. N. Greene. Voxel space automata: Modeling with stochastic growth processes in voxel space. Proceedings of SIGGRAPH '89 (Boston, Mass., July 31–August 4, 1989), in *Computer Graphics* 23, 4 (August 1989), pages 175–184, ACM SIGGRAPH, New York, 1989.
35. J. Gruska and H. Jürgensen. Informatics: a fundamental science and methodology for the sciences (emerging from Computer Science and maturing). Manuscript, Department of Informatics, Slovak Academy of Sciences, Bratislava, and Department of Computer Science, University of Western Ontario, London, Ontario, 1990.
36. J. Gruska and H. Jürgensen. Maturing of informatics. In D. Bjørner and V. Kotov, editors, *Images of Programming*, pages I–55 – I–69. North Holland, Amsterdam, 1991.
37. M. R. Guzy. A morphological-mechanistic plant model formalized in an object-oriented parametric L-system. Manuscript, USDA-ARS Salinity Laboratory, Riverside, 1995.
38. F. Hallé. Modular growth in seed plants. *Philos. Trans. Royal Society London, Ser. B*, 313:77–87, 1986.
39. F. Hallé, R. A. A. Oldeman, and P. B. Tomlinson. *Tropical trees and forests: An architectural analysis.* Springer-Verlag, Berlin, 1978.
40. J. Hanan. Virtual plants — Integrating architectural and physiological plant models. In P. Binning, H. Bridgman, and B. Williams, editors, *Proceedings of ModSim 95*, volume 1, pages 44–50, Perth, 1995. The Modelling and Simulation Society of Australia.
41. J. S. Hanan. PLANTWORKS: A software system for realistic plant modelling. Master's thesis, University of Regina, 1988.
42. J. S. Hanan. *Parametric L-systems and their application to the modelling and visualization of plants.* PhD thesis, University of Regina, June 1992.
43. J. L. Harper and A. D. Bell. The population dynamics of growth forms in organisms with modular construction. In R. M. Anderson, B. D. Turner, and L. R. Taylor, editors, *Population dynamics*, pages 29–52. Blackwell, Oxford, 1979.
44. J. W. Hart. *Plant tropisms and other growth movements.* Unwin Hyman, London, 1990.
45. G. T. Herman and W. H. Liu. The daughter of CELIA, the French flag, and the firing squad. *Simulation*, 21:33–41, 1973.
46. G. T. Herman and G. Rozenberg. *Developmental systems and languages.* North-Holland, Amsterdam, 1975.
47. P. Hogeweg. Simulating the growth of cellular forms. *Simulation*, pages 90–96, September 1978.
48. P. Hogeweg. Locally synchronized developmental systems: Conceptual advantages of discrete event formalism. *International Journal of General Systems*, 6:57–73, 1980.
49. P. Hogeweg and B. Hesper. A model study on biomorphological description. *Pattern Recognition*, 6:165–179, 1974.

50. H. Honda. Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body. *Journal of Theoretical Biology*, 31:331–338, 1971.
51. H. Honda, P. B. Tomlinson, and J. B. Fisher. Computer simulation of branch interaction and regulation by unequal flow rates in botanical trees. *American Journal of Botany*, 68:569–585, 1981.
52. C. Jacob. Modeling growth with L-systems and *Mathematica*. *Mathematica in Education and Research*, 4(3):12–19, 1995.
53. M. Jaeger and P. de Reffye. Basic concepts of computer simulation of plant growth. *Journal of Biosciences*, 17(3):275–291, 1992.
54. M. James, J. Hanan, and P. Prusinkiewicz. CPFG version 2.0 user's manual. Manuscript, Department of Computer Science, The University of Calgary, 1993, 50 pages.
55. J. M. Janssen and A. Lindenmayer. Models for the control of branch positions and flowering sequences of capitula in *Mycelis muralis* (L.) Dumont (Compositae). *New Phytologist*, 105:191–220, 1987.
56. H. Jürgensen. Probabilistic L systems. In A. Lindenmayer and G. Rozenberg, editors, *Automata, languages, development*, pages 211–225. North-Holland, 1976.
57. H. Jürgensen, H. Shyr, and G. Thierrin. Monoids with disjunctive identity and their codes. *Acta Mathematica Hungarica*, 47(3–4):299–312, 1986.
58. J. Kemeny. *A philosopher looks at science*. Van Nostrand, Princeton, 1959.
59. B. W. Kernighan and D. M. Ritchie. *The C programming language. Second edition.* Prentice Hall, Englewood Cliffs, 1988.
60. D. E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2(2):191–220, 1968.
61. W. Kurth. *Growth grammar interpreter GROGRA 2.4: A software tool for the 3-dimensional interpretation of stochastic, sensitive growth grammars in the context of plant modeling. Introduction and reference manual.* Forschungszentrum Waldökosysteme der Universität Göttingen, Göttingen, 1994.
62. W. Kurth. Morphological models of plant growth: Possibilities and ecological relevance. *Ecological Modelling*, 75/76:299–308, 1994.
63. W. Kurth. Stochastic sensitive growth grammars: A basis for morphological models of tree growth. *Naturalia Monspeliensia*, 1996. In press.
64. W. Kurth and D. Lanwert. Biometrische Grundlagen für ein dynamisches Architekturmodell der Fichte (*Picea abies* (L.) Karst.). *Allgemeine Forst und Jagdzeitung*, 166:177–184, 9/10 1995.
65. C. M. Liddell and D. Hansen. Visualizing complex biological interactions in the soil ecosystem. *The Journal of Visualization and Computer Animation*, 4:3–12, 1993.
66. A. Lindenmayer. Developmental systems and languages in their biological context. In G. T. Herman and G. Rozenberg, *Developmental systems and languages.* North-Holland, Amsterdam, 1975, pp. 1 – 40.
67. A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 18:280–315, 1968.
68. A. Lindenmayer. Developmental systems without cellular interaction, their languages and grammars. *Journal of Theoretical Biology*, 30:455–484, 1971.
69. A. Lindenmayer. Adding continuous components to L-systems. In G. Rozenberg and A. Salomaa, editors, *L Systems*, Lecture Notes in Computer Science 15, pages 53–68. Springer-Verlag, Berlin, 1974.
70. A. Lindenmayer. Developmental algorithms for multicellular organisms: A survey of L-systems. *Journal of Theoretical Biology*, 54:3–22, 1975.

71. A. Lindenmayer. Theories and observations of developmental biology. In R. E. Butts and J. Hintikka, editors, *Foundational problems in special sciences*, pages 103–118. D. Reidel Publ. Co, Dordrecht-Holland, 1977.
72. A. Lindenmayer. Algorithms for plant morphogenesis. In R. Sattler, editor, *Theoretical plant morphology*, pages 37–81. Leiden University Press, The Hague, 1978.
73. A. Lindenmayer. Developmental algorithms: Lineage versus interactive control mechanisms. In S. Subtelny and P. B. Green, editors, *Developmental order: Its origin and regulation*, pages 219–245. Alan R. Liss, New York, 1982.
74. A. Lindenmayer. Positional and temporal control mechanisms in inflorescence development. In P. W. Barlow and D. J. Carr, editors, *Positional controls in plant development*. University Press, Cambridge, 1984.
75. A. Lindenmayer. Models for multicellular development: Characterization, inference and complexity of L-systems. In A. Kelemenová and J. Kelemen, editors, *Trends, techniques and problems in theoretical computer science*, Lecture Notes in Computer Science 281, pages 138–168. Springer-Verlag, Berlin, 1987.
76. A. Lindenmayer and H. Jürgensen. Grammars of development: Discrete-state models for growth, differentiation and gene expression in modular organisms. In G. Rozenberg and A. Salomaa, editors, *Lindenmayer systems: Impacts on theoretical computer science, computer graphics, and developmental biology*, pages 3–21. Springer-Verlag, Berlin, 1992.
77. A. Lindenmayer and P. Prusinkiewicz. Developmental models of multicellular organisms: A computer graphics perspective. In C. G. Langton, editor, *Artificial Life*, pages 221–249. Addison-Wesley, Redwood City, 1988.
78. A. Lindenmayer and G. Rozenberg, editors. *Automata, languages, development*. North-Holland, Amsterdam, 1976.
79. J. Lück, H. B. Lück, and M. Bakkali. A comprehensive model for acrotonic, mesotonic, and basitonic branching in plants. *Acta Biotheoretica*, 38:257–288, 1990.
80. N. Macdonald. *Trees and networks in biological models*. J. Wiley & Sons, New York, 1983.
81. B. B. Mandelbrot. *The fractal geometry of nature*. W. H. Freeman, San Francisco, 1982.
82. C. K. McClelland. On the regularity of blooming in the cotton plant. *Science*, XLIV:578–581, 1916.
83. R. A. Morelli, R. E. Walde, E. Akstin, and C. W. Schneider. L-system representation of speciation in the red algal genus Dipterosiphonia (Ceramiales, Rhodomelaceae). *The Journal of Theoretic Biology*, 149:453–465, 1991.
84. B. Moulia. The biomechanics of leaf rolling. *Biomimetics*, 2(3):267–281, 1994.
85. B. Moulia and H. Sinoquet. Three-dimensional digitizing systems for plant canopy geometrical structure: a review. In C. Varlet-Grancher, R. Bonhomme, and H. Sinoquet, editors, *Crop structure and light microclimate: Characterization and applications*, pages 183–193. INRA, Paris, 1993.
86. K. J. Niklas. *Plant biomechanics: an engineering approach to plant form and function*. The University of Chicago Press, Chicago, 1992.
87. T. Nishida. K0L-systems simulating almost but not exactly the same development — the case of Japanese cypress. *Memoirs Fac. Sci., Kyoto University, Ser. Bio*, 8:97–122, 1980.
88. S. Papert. *Mindstorms: Children, computers and powerful ideas*. Basic Books, New York, 1980.
89. F. P. Preparata and R. T. Yeh. *Introduction to discrete structures*. Addison-Wesley, Reading, Massachusetts, 1973.

90. P. Prusinkiewicz. Graphical applications of L-systems. In *Proceedings of Graphics Interface '86 — Vision Interface '86*, pages 247–253, 1986.
91. P. Prusinkiewicz. Applications of L-systems to computer imagery. In H. Ehrig, M. Nagl, A. Rosenfeld, and G. Rozenberg, editors, *Graph grammars and their application to computer science; Third International Workshop*, pages 534–548. Springer-Verlag, Berlin, 1987. Lecture Notes in Computer Science 291.
92. P. Prusinkiewicz. Visual models of morphogenesis. *Artificial Life*, 1(1/2):61–74, 1994.
93. P. Prusinkiewicz, M. Hammel, and E. Mjolsness. Animation of plant development. Proceedings of SIGGRAPH 93 (Anaheim, California, August 1–6, 1993). In *Computer Graphics* Proceedings, Annual Conference Series, 1993. ACM SIGGRAPH, New York, 1993, pp. 369–378.
94. P. Prusinkiewicz and J. Hanan. *Lindenmayer systems, fractals, and plants*, volume 79 of *Lecture Notes in Biomathematics*. Springer-Verlag, Berlin, 1989.
95. P. Prusinkiewicz and J. Hanan. Visualization of botanical structures and processes using parametric L-systems. In D. Thalmann, editor, *Scientific visualization and graphics simulation*, pages 183–201. J. Wiley & Sons, Chichester, 1990.
96. P. Prusinkiewicz and J. Hanan. L-systems: From formalism to programming languages. In G. Rozenberg and A. Salomaa, editors, *Lindenmayer systems: Impacts on theoretical computer science, computer graphics, and developmental biology*, pages 193–211. Springer-Verlag, Berlin, 1992.
97. P. Prusinkiewicz, M. James, and R. Měch. Synthetic topiary. Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994), pages 351–358, ACM SIGGRAPH, New York, 1994.
98. P. Prusinkiewicz and L. Kari. Subapical bracketed L-systems. To appear in the Proceedings of the Fifth International Workshop on Graph Grammars and their Application to Computer Science, Williamsburg, 1994.
99. P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*. Springer-Verlag, New York, 1990. With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer.
100. P. Prusinkiewicz, A. Lindenmayer, and F. D. Fracchia. Synthesis of space-filling curves on the square grid. In H.-O. Peitgen, J. M. Henriques, and L. F. Penedo, editors, *Fractals in the fundamental and applied sciences*, pages 341–366. North-Holland, Amsterdam, 1991.
101. P. Prusinkiewicz, A. Lindenmayer, and J. Hanan. Developmental models of herbaceous plants for computer imagery purposes. Proceedings of SIGGRAPH '88 (Atlanta, Georgia, August 1–5, 1988), in *Computer Graphics* 22, 4 (August 1988), pages 141–150, ACM SIGGRAPH, New York, 1988.
102. P. Prusinkiewicz, W. Remphrey, C. Davidson, and M. Hammel. Modeling the architecture of expanding *Fraxinus pennsylvanica* shoots using L-systems. *Canadian Journal of Botany*, 72:701–714, 1994.
103. P. Prusinkiewicz and G. Sandness. Koch curves as attractors and repellers. *IEEE Computer Graphics and Applications*, 8(6):26–40, November 1988.
104. D. M. Raup. Geometric analysis of shell coiling: general problems. *Journal of Paleontology*, 40:1178–1190, 1966.
105. D. M. Raup and A. Michelson. Theoretical morphology of the coiled shell. *Science*, 147:1294–1295, 1965.
106. W. R. Remphrey, B. R. Neal, and T. A. Steeves. The morphology and growth of *Arctostaphylos uva-ursi* (bearberry), parts I and II. *Canadian Journal of Botany*, 61(9):2430–2458, 1983.
107. P. M. Room. 'Falling apart' as a lifestyle: the rhizome architecture and population growth of *Salvinia molesta*. *Journal of Ecology*, 71:349–365, 1983.

108. P. M. Room and J. S. Hanan. Virtual cotton: a new tool for research, management and training. To appear in the Proceedings of the World Cotton Research Conference, Brisbane, Australia, February 14–17, 1994.
109. P. M. Room, J. S. Hanan, and P. Prusinkiewicz. Virtual plants: new perspectives for ecologists, pathologists, and agricultural scientists. *Trends in Plant Science*, 1(1):33–38, 1996.
110. P. M. Room, L. Maillette, and J. Hanan. Module and metamer dynamics and virtual plants. *Advances in Ecological Research*, 25:105–157, 1994.
111. G. Rozenberg. T0L systems and languages. *Information and Control*, 23:357–381, 1973.
112. G. Rozenberg, K. Ruohonen, and A. Salomaa. Developmental systems with fragmentation. *International Journal of Computer Mathematics*, 5:177–191, 1976.
113. G. Rozenberg and A. Salomaa. *The mathematical theory of L systems.* Academic Press, New York, 1980.
114. G. Rozenberg and A. Salomaa. When L was young. In G. Rozenberg and A. Salomaa, editors, *The book of L*, pages 383–392. Springer-Verlag, Berlin, 1986.
115. K. Ruohonen. Developmental systems with interaction and fragmentation. *Information and Control*, 28:91–112, 1975.
116. A. Salomaa. *Formal languages.* Academic Press, New York, 1973.
117. C. W. Schneider and R. E. Walde. L-system computer simulations of branching divergence in some dorsiventral members of the tribe Polysiphonieae (Rhodomelaceae, Rhodophyta). *Phycologia*, 31(6):581–590, 1992.
118. C. W. Schneider, R. E. Walde, and R. A Morelli. L-systems computer models generating distichous from spiral organization in the Dasyaceae (Ceramiales, Rhodophyta). To appear in the *European Journal of Phycology.*
119. M. F. Shebell. Modeling branching plants using attribute L-systems. Master's thesis, Worcester Polytechnic Institute, 1986.
120. A. R. Smith. Plants, fractals, and formal languages. Proceedings of SIGGRAPH '84 (Minneapolis, Minnesota, July 22–27, 1984) in *Computer Graphics*, 18, 3 (July 1984), pages 1–10, ACM SIGGRAPH, New York, 1984.
121. A. R. Smith. About the cover: Reconfigurable machines. *Computer*, 11(7):3–4, 1978.
122. A. L. Szilard and R. E. Quinton. An interpretation for DOL systems by computer graphics. *The Science Terrapin*, 4:8–13, 1979.
123. A. Takenaka. A simulation model of tree architecture development based on growth response to local light environment. *Journal of Plant Research*, 107:321–330, 1994.
124. J. H. M. Thornley and I. R. Johnson. *Plant and crop modeling: A mathematical approach to plant and crop physiology.* Oxford University Press, New York, 1990.
125. A. Tunbridge and H. Jones. An L-systems approach to the modelling of fungal growth. *The Journal of Visualization and Computer Animation*, 6:91–107, 1995.
126. A. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London B*, 237:37–72, 1952.
127. H. von Koch. Une méthode géométrique élémentaire pour l'étude de certaines questions de la théorie des courbes planes. *Acta Mathematica*, 30:145–174, 1905.
128. D. M. Waller and D. A. Steingraeber. Branching and modular growth: Theoretical models and empirical patterns. In J. B. C. Jackson and L. W. Buss, editors, *Population biology and evolution of clonal organisms*, pages 225–257. Yale University Press, New Haven, 1985.
129. H. M. Ward. *Trees. Volume V: Form and habit.* Cambridge University Press, Cambridge, 1909.

130. J. Weber and J. Penn. Creation and rendering of realistic trees. Proceedings of SIGGRAPH '95 (Los Angeles, California, August 6–11, 1995), pages 119–128, ACM SIGGRAPH, New York, 1995.
131. F. D. Whisler, B. Acock, D. N. Baker, R. E. Fye, H. F. Hodges, J. R. Lambert, H. E. Lemmon, J. M. McKinion, and V. R. Reddy. Crop simulation models in agronomic systems. *Advances in Agronomy*, 40:141–208, 1986.
132. B. F. Wilson. *The growing tree*. The University of Massachusetts Press, Amherst, 1984.
133. T. Yokomori. Stochastic characterizations of EOL languages. *Information and Control*, 45:26–33, 1980.

Index