The Artificial Life of Plants

Przemyslaw Prusinkiewicz, Mark Hammel, Radomír Měch
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4
e-mail: pwp|hammel|mech@cpsc.ucalgary.ca

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

From *Artificial life for graphics, animation, and virtual reality*, volume 7 of SIGGRAPH ’95 Course Notes, pages 1-1–1-38. ACM Press, 1995.

The Artificial Life of Plants

Przemyslaw Prusinkiewicz, Mark Hammel, Radomír Měch
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4
e-mail: pwp|hammel|mech@cpsc.ucalgary.ca

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

August 23, 1995

Abstract

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* [61]. The new developments include:

- a better understanding of theoretical issues pertinent to graphical applications of L-systems,
- extensions to programming techniques based on L-systems, and
- an extension of the range of phenomena expressible using L-systems.

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

1 Introduction

In 1968, Aristid Lindenmayer introduced a formalism for simulating the development of multicellular organisms, subsequently named L-systems [36]. This formalism was closely related to abstract automata and formal languages, and attracted the immediate interest of theoretical computer scientists [67]. The vigorous development of the mathematical theory of L-systems [70, 27, 66] was followed by the application of the theory to the modeling of plants [18, 17, 16, 31, 42, 56, 61]. In the present notes we survey recent results, current work, and open problems pertinent to this latter domain. In this context, we emphasize the phenomenon of *data base amplification* [71], that

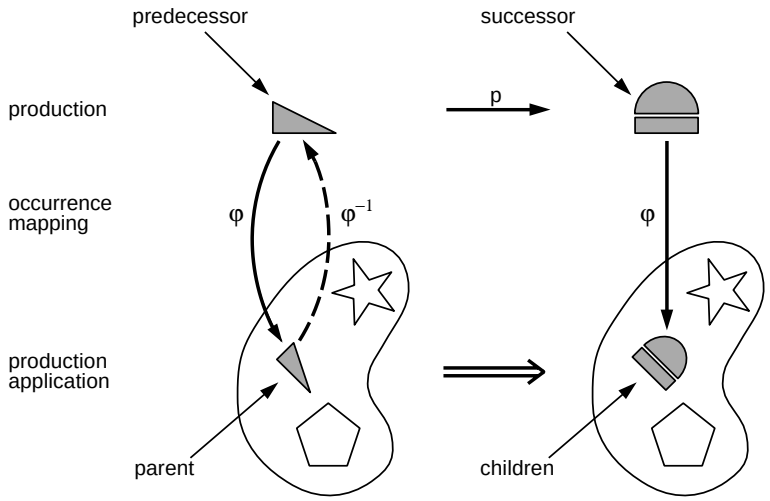

Figure 1: Illustration of the concept of rewriting applied to modules with geometric interpretation. A parent module is replaced by child modules in a sequence of transformations $\varphi^{-1}p\varphi$.

is the possibility of generating complex structures from small data sets, and the related notion of *emergence*. According to Taylor [74, page 31], emergence is a process in which a collection of interacting units acquires qualitatively new properties that cannot be reduced to a simple superposition of individual contributions. Studies of emergence are amongst the central themes of *artificial life* (*Problem 1.1*, see Section 11).

2 The modular structure of plants

L-systems were originally introduced to model the development of simple *multicellular* organisms (for example, algae) in terms of division, growth, and death of individual cells [36, 37]. The range of L-system applications has subsequently been extended to higher plants and complex branching structures, in particular inflorescences [18, 19], described as configurations of modules in space (*Problem 2.1*). In the context of L-systems, the term *module* denotes any discrete constructional unit that is repeated as the plant develops, for example an internode, an apex, a flower, or a branch (c.f. [4, page 284]) (*Problem 2.2*). The goal of modeling at the modular level is to describe the development of a plant as a whole, and in particular the emergence of plant shape, as the integration of the development of individual units (*Problem 2.3*).

3 Plant development as a rewriting process

The essence of development at the modular level can be conveniently captured by a parallel *rewriting system* that replaces individual *parent*, *mother*, or *ancestor* modules by configurations of *child*, *daughter*, or *descendant* modules. All modules belong to a finite *alphabet of module types*, thus the behavior of an arbitrarily large configuration of modules can be specified using a finite set of *rewriting rules* or *productions*. In the simplest case of *context-free* rewriting, a production consists of a single module called the *predecessor* or the *left-hand side*, and a configuration of zero, one,

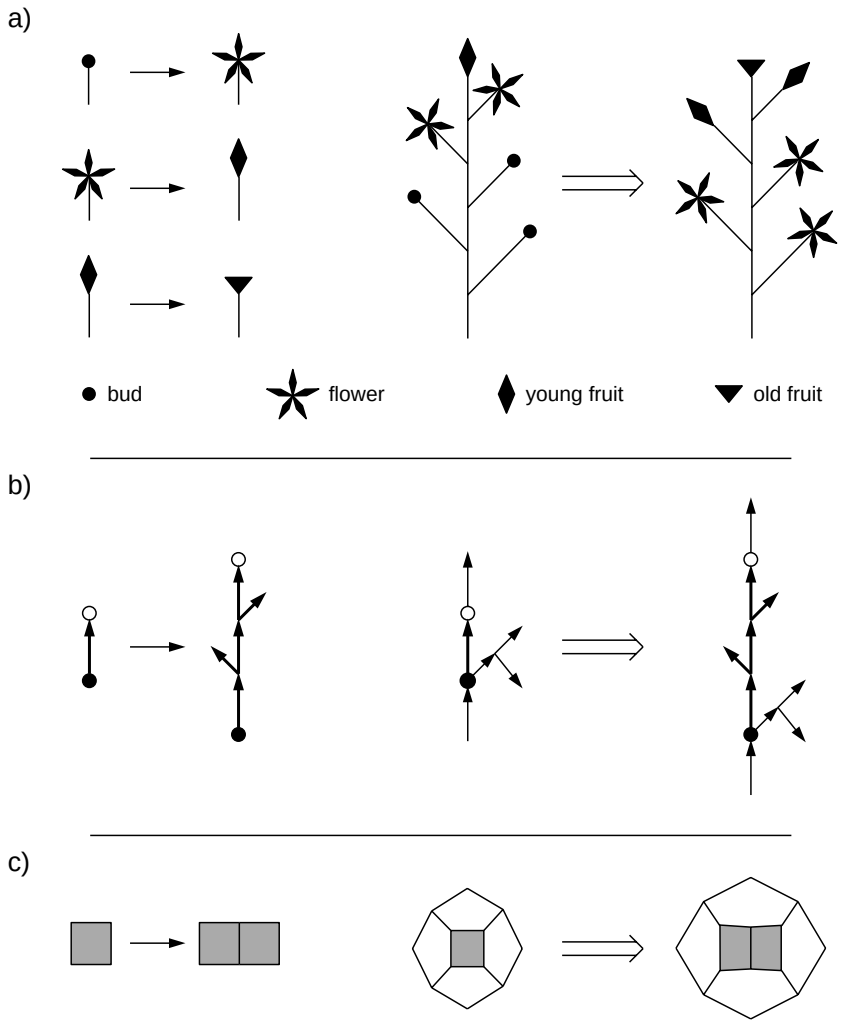

Figure 2: Examples of production specification and application: (a) development of a flower, (b) development of a branch, and (c) cell division.

or more modules called the *successor* or the *right-hand side*. A production p with the predecessor matching a given mother module can be applied by deleting this module from the rewritten structure and inserting the daughter modules specified by the production's successor. This process requires finding an *occurrence map* φ that transforms the predecessor into the mother module, and applying the same transformation to all the modules in the successor in order to produce the appropriate child modules (Figure 1).

Three examples of production application are shown in Figure 2. In case (a), modules located at the extremities of a branching structure are replaced without affecting the remainder of the structure. In case (b), productions that replace internodes divide the branching 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 successor modules, but the shape and size of both the lower and upper part are not changed. Finally, in case (c), the rewritten structures are represented by graphs with cycles. The size and shape of the production successor does not exactly match the size and

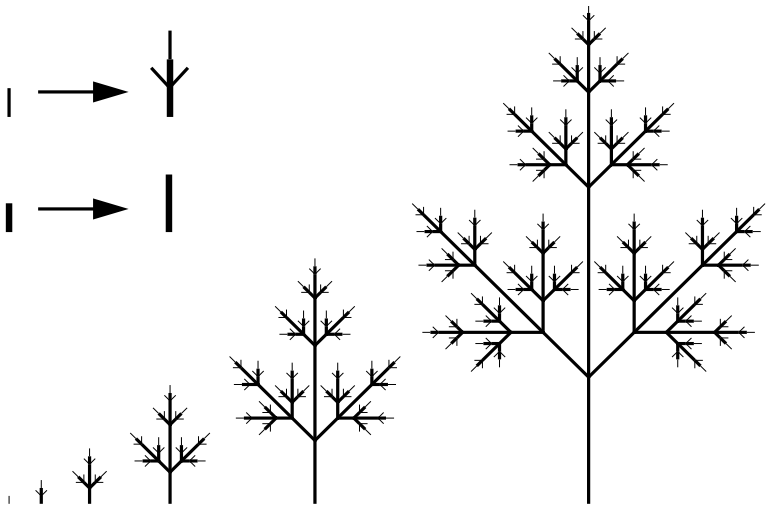

Figure 3: Developmental model of a compound leaf, modeled as a configuration of apices and internodes.

shape of the predecessor, and the geometry of the predecessor and the embedding structure had to be adjusted to accommodate the successor. The last case is 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 [11, 12, 15, 61]. In these notes we focus on the rewriting of branching structures corresponding to cases (a) and (b). (*Problem 3.1*).

Productions may be applied *sequentially*, to one module at a time, or they may be applied *in parallel*, with all modules being rewritten simultaneously in every *derivation step*. Parallel rewriting is more appropriate for the modeling of biological development, since development takes place simultaneously in all parts of an organism. A derivation step then corresponds to the progress of time over some interval. A sequence of structures obtained in consecutive derivation steps from a predefined *initial structure* or *axiom* is called a *developmental sequence*. It can be viewed as the result of a *discrete-time simulation* of development (*Problem 3.2*).

For example, Figure 3 illustrates the development of a stylized compound leaf including two module types, the *apices* (represented by thin lines) and the *internodes* (thick lines). An apex yields a structure that consists of two internodes, two lateral apices, and a replica of the main apex. An internode elongates by a constant scaling factor. In spite of the simplicity of these rules, an intricate branching structure develops from a single apex over a number of derivation steps. The fractal geometry of this structure can be viewed as an emergent property of the rewriting rules.

At first sight, Figure 3 resembles fractal generation using a Koch construction. We will see, however, that there are important differences between these two processes.

Mandelbrot [48, page 39] characterized Koch constructions as follows:

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

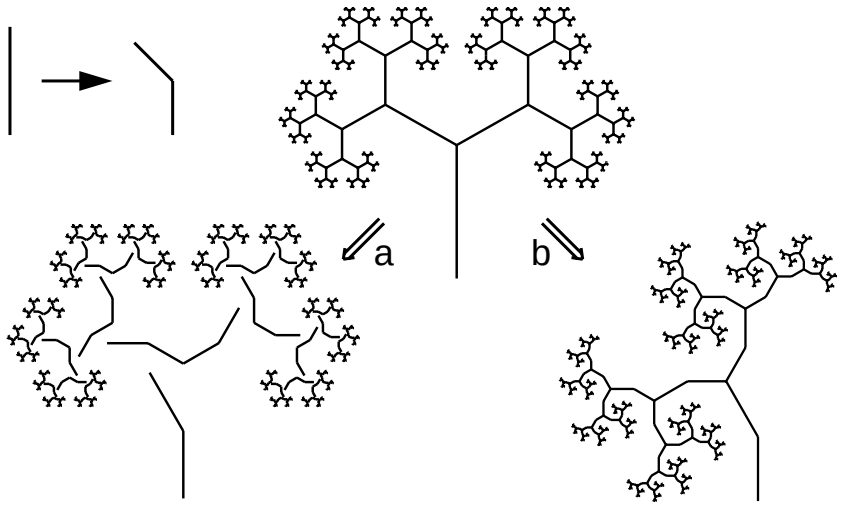

Figure 4: A comparison of the Koch construction (a) with a rewriting system preserving the branching topology of the modeled structures (b). The same production is applied in both cases, but the rules for incorporating the successor into the structure are different.

Mandelbrot introduced many extensions of this basic concept, including generators with lines of unequal length (pages 56–57) and with a branching topology (pages 71–73) (*Problem 3.3*). 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 models of plants, however, the position and orientation of each module should be 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 elongate (as is the case in Figure 3), all the subtended branches are moved upwards in response. Similarly, when the internodes bend, the subtended branches are rotated and displaced (Figure 4b). A Koch construction cannot capture these phenomena, because it operates under the assumption that the parent modules determine all aspects of their descendants (Figure 4a). In contrast, in a developmental model of a branching structure the position and orientation of the descendants are determined by the subtending modules. The difference between these two cases is illustrated in Figure 5.

The information flow taking place during the development of a branching structure can be expressed directly in the geometric domain, using a proper modification of the Koch construction (*Problem 3.4*). A different approach was proposed by Lindenmayer [36, 37] and is essential to the resulting theory of L-systems. The generated structures and the rewriting rules are expressed symbolically using a string notation. The *geometric interpretation* of these strings automatically captures proper positioning of the higher branches on the lower ones.

The basic notions of the theory of L-systems have been presented in many survey papers [39, 40, 41, 43, 44, 45] and books [27, 56, 61, 66, 70]. Consequently, we only describe parametric L-systems, which are a convenient programming tool for expressing models of plant development.

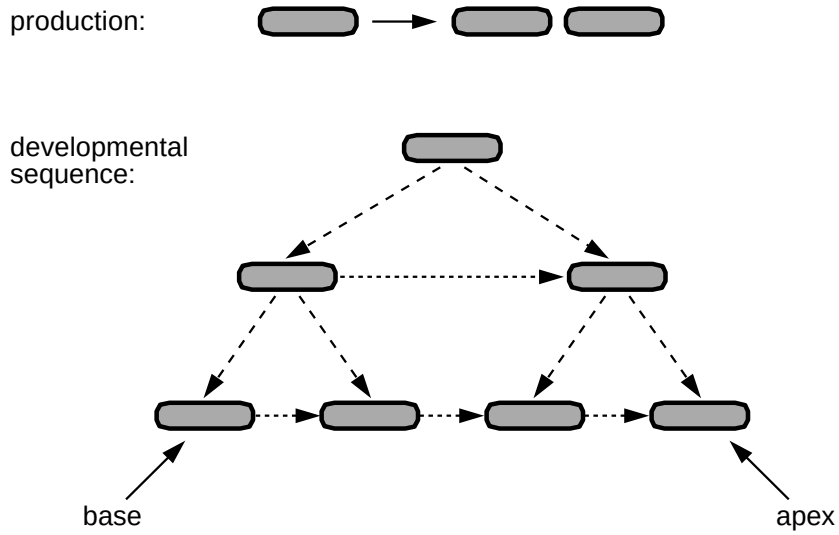

Figure 5: Information flow in a Koch construction and in a developing modular structure. In the Koch construction, information flows only from the parent modules to their descendants (continuous line). In the developmental model, positional information flows along the paths from the root to the apices of the branches (dashed line).

4 Parametric L-systems

Parametric L-systems extend the basic concept of L-systems by assigning numerical attributes to the L-system symbols. This extension was implemented as early as the 1970's in the first simulator based on L-systems, called CELIA (and acronym for CEllular Linear Iterative Array simulator) [3, 26, 27, 38], as a programming rather than theoretical construct. Our presentation closely follows the formalization introduced in [57, 61] (see also [25, 58]) (*Problems 4.1, 4.2*).

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

The real-valued *actual* parameters appearing in the words correspond with *formal* parameters used in the specification of L-system productions. If Σ is a set of formal parameters, then $C(\Sigma)$ denotes a *logical expression* with parameters from Σ, and $E(\Sigma)$ is an *arithmetic expression* with parameters from the same set. Both types of expressions consist of formal parameters and numeric constants, combined using the arithmetic operators $+$, $-$, $*$, $/$; the exponentiation operator $\wedge$, the relational operators $<$, $<=$, $>$, $>=$, $==$; the logical operators !, $\&\&$, $||$ (not, and, or); and parentheses $()$. The operation symbols and the rules for constructing syntactically correct expressions are the same as in the C programming language [32]. 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 OL-system* is defined as 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*.

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

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

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

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

A matching production can be *applied* to the module, creating a string of modules specified by the production successor. The actual parameter values are substituted for the formal parameters according to their position. For example, production (1) 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 the application of this production is a parametric word $B(10)CD(3, 7)$.

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

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

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

It is assumed that a module replaces itself if no matching production is found in the set P. The words obtained in the first few derivation steps are shown in Figure 6.

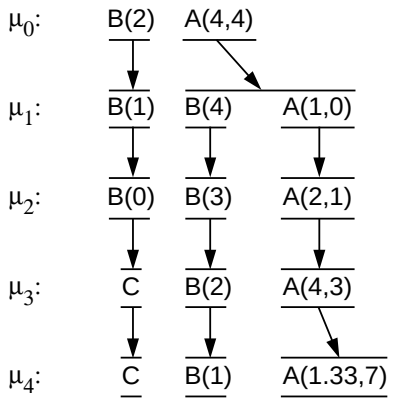

Figure 6: The initial sequence of strings generated by the parametric L-system specified in equation (2)

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

A sample context-sensitive production is given below:

$$A(x) < B(y) > C(z) : x + y + z > 10 \rightarrow E((x + y)/2)F((y + z)/2). \tag{3}$$

The left context is separated from the strict predecessor by the symbol $<$. Similarly, the strict predecessor is separated from the right context by the symbol $>$. Production 3 can be applied to the module $B(5)$ that appears in a parametric word

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

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

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.

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 L-system $G = \langle V, \Sigma, \omega, P\rangle$ is called *deterministic* 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 matching production.

If more than one production in P matches a module, an additional mechanism is needed to select which production should be applied to this module. In *stochastic* L-systems, this decision is based on random factors. In the most extensive case, a production has the format:

$$id : lc < pred > rc : cond \rightarrow succ : \pi$$

where id is the production identifier (label), lc, $pred$, and rc are the left context, the strict predecessor, and the right context, $cond$ is the condition, $succ$ is the successor, and π is an arithmetic expression returning a non-negative number called the *probability factor*. If $\hat{P} \subseteq P$ is the set of productions matching a given module $A(t_1, t_2, \ldots, t_n) \in V \times \Re^*$ in the rewritten string, then the probability $\mathrm{prob}(p_k)$ of applying a particular production $p_k \in \hat{P}$ to this module is equal to:

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

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

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

$$\begin{array}{ll} \omega: & A(1)B(3)A(5) \\ p_1: & A(x) \rightarrow A(x+1) : 2 \\ p_2: & A(x) \rightarrow B(x-1) : 3 \\ p_3: & A(x) : x > 3 \rightarrow C(x) : x \\ p_4: & A(x) < B(y) > A(z) : y < 4 \rightarrow B(x+z)A(y) \end{array}$$

The 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: $\mathrm{prob}(p_1) = 2/(2+3) = 0.4$, and $\mathrm{prob}(p_2) = 3/(2+3) = 0.6$. 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: $\mathrm{prob}(p_1) = 2/(2+3+5) = 0.2$, $\mathrm{prob}(p_2) = 3/(2+3+5) = 0.3$, and $\mathrm{prob}(p_3) = 5/(2+3+5) = 0.5$. The context-sensitive production p_4 replaces a module $B(y)$ with left context $A(x)$ and right context $A(z)$ by the pair of modules $B(x+z)A(y)$. The application of this production is guarded by condition $y < 4$. Taking all these factors into account, the first derivation step may have the form:

$$A(1)B(3)A(5) \Longrightarrow A(2)B(6)A(3)C(5)$$

It was assumed that, as a result of random choice, production p_1 was applied to the module $A(1)$, and production p_3 to the module $A(5)$. Production p_4 was applied to the module $B(3)$, because it occurred with the required left and right context, and the condition $3 < 4$ was true.

5 The turtle interpretation of L-systems

Strings generated by L-systems may be interpreted geometrically in many different ways [57, 61]. Below we outline the *turtle interpretation* of parametric L-systems, introduced by Szilard and Quinton [73], and extended by Prusinkiewicz [51, 52] and Hanan [24, 25]. A tutorial exposition is included in [61], and subsequent results are presented in [25]. The summary below is based on [30, 52] and [61].

After a string has been generated by an L-system, it is scanned sequentially from left to right, and the consecutive symbols are interpreted as commands that maneuver a LOGO-style turtle [1,

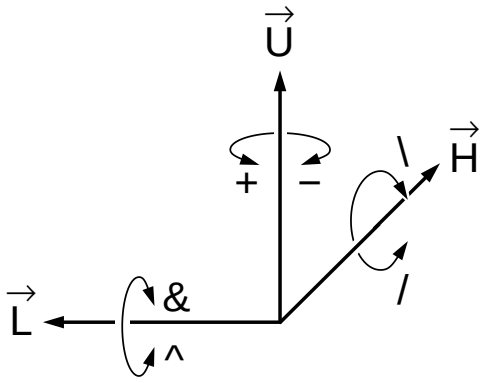

Figure 7: Controlling the turtle in three dimensions

50] 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 various attribute values, such as current *color* and *line width*. The position is defined by a vector $\vec{P}$, and the orientation is defined by three vectors $\vec{H}$, $\vec{L}$, and $\vec{U}$, indicating the turtle's *heading* and the directions to the *left* and *up* (Figure 7). These vectors have unit length, are perpendicular to each other, and satisfy the equation $\vec{H} \times \vec{L} = \vec{U}$. Rotations of the turtle are expressed by the equation:

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

where $\mathbf{R}$ is a 3×3 rotation matrix [14]. Specifically, rotations by angle α about vectors $\vec{U}$, $\vec{L}$ and $\vec{H}$ are represented by the matrices:

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

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

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

Changes in the turtle's 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

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

$f(s)$ 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 7)

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

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

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

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

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

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

$|$ Turn 180° around the $\vec{U}$ 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.

$\sim$ Draw the surface identified by the symbol immediately following the $\sim$ at the turtle's current location and orientation.

$@PS(i, basis)$ Begin definition of bicubic surface i by initializing its 16 control points to $(0, 0, 0)$. The optional parameter $basis$ specifies the type of patch as:

1. Bézier,
2. B-spline,
3. Cardinal spline.

If no basis is given, use Bézier surface as the default.

$@PC(i, r, c)$ Assign the current position of the turtle to the control point of surface i in row r and column c.

$@PD(i, s, t)$ Draw surface i by subdividing it into s quadrangles along the rows and t along the columns.

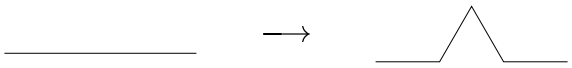

Figure 8: The production $F(s) \rightarrow F(s/3) + (60)F(s/3) - (120)F(s/3) + (60)F(s/3)$

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 colour 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 colour decrement if no parameter is given.

6 Examples of L-systems

This section presents selected examples that illustrate the operation of L-systems with turtle interpretation. The material is based on [30, 59], and [61].

Fractal curves are useful in explaining the operation of L-systems that do not include branches. The following L-system generates the Koch snowflake curve.

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

The axiom $F(1) - (120)F(1) - (120)F(1)$ draws an equilateral triangle, with the edges of unit length. Production p_1 replaces each line segment with the polygonal shape shown in Figure 8. The productions for the $+$ and $-$ symbols 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:

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

The axiom and the first three derivation steps are illustrated in Figure 9.

The next L-system generates the developmental sequence of the compound leaf model presented in Figure 3.

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

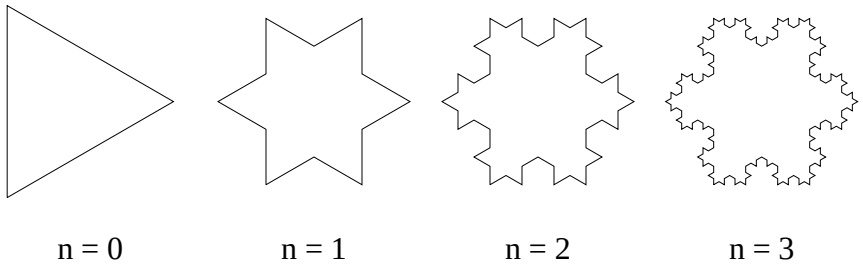

Figure 9: The snowflake curve after n = 0, 1, 2, and 3 derivation steps

The axiom and productions p_1 and p_2 include modules F with two parameters. The first parameter specifies the length of the line representing the module. The second parameter determines whether the module is an apex ($t == 1$) or an internode ($t == 2$). The graphical interpretation of both productions is shown in Figure 3. The branching angle associated with symbols $+$ and $-$ is set to 45° by a global variable outside the L-system. Production p_3 is used to make the lines representing the internodes wider than the lines representing the apices.

The following example illustrates the application of a stochastic L-system to the generation of a three-dimensional tree. The model is based on the analysis of tree growth by Borchert and Slade [7].

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

The generation of the tree begins with a single internode F terminated by apex $A(1)$. The parameter of the apex (k) acts as a counter of derivation steps. Production p_1 describes the creation of two new branches, whereas production p_2 describes the production of a branch segment and a dormant bud B. Probabilities of these events are equal to $\mathrm{prob}(p_1) = \min\{1, (2k+1)/k^2\}$ and $\mathrm{prob}(p_2) = 1 - \mathrm{prob}(p_1)$, respectively, thus the probability of branching (captured by production p_1) gradually decreases as the tree grows older. A detailed justification of these formulas is given in [7, 59]. Figure 10 shows side views of three sample trees after 18 derivation steps. The branching angles, equal to $\phi = 90^\circ, \alpha = 32^\circ$, and $\beta = 20^\circ$, yield a sympodial branching structure (new shoots do not continue the growth direction of the preceding segments). This structure is representative to the Leeuwenberg's model of tree architecture identified by Hallé *et al.* [23], although no attempt to capture a particular tree species was made.

The final example of this section illustrates the inclusion of predefined surfaces into a model. The following L-system generates the stylized flower shown in Figure 11.

$$\begin{array}{ll} \omega: & /(154)B \\ p_1: & B \rightarrow [\&(72)\#F(5)!P] \\ p_2: & P \rightarrow [S/(72)S/(72)S/(72)S/(72)S] \\ p_3: & S \rightarrow [\wedge(103) \sim s][\wedge(72) \sim p][\wedge(34)F(1)\#[-F(1)][+F(1)]] \end{array}$$

Production p_1 creates the stalk $F(5)$ and the symbol P, which gives rise to the *perianth*. Production p_2 describes the perianth as consisting of five sectors S, rotated with respect to each other by 72°.

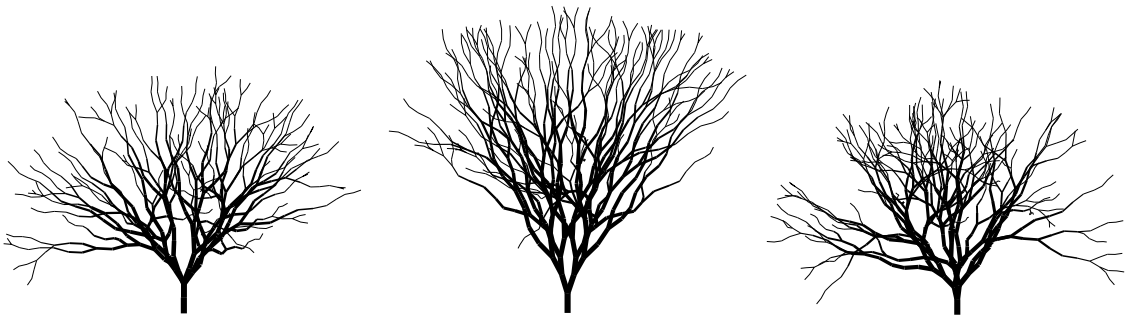

Figure 10: Sample tree structures generated using a stochastic L-system

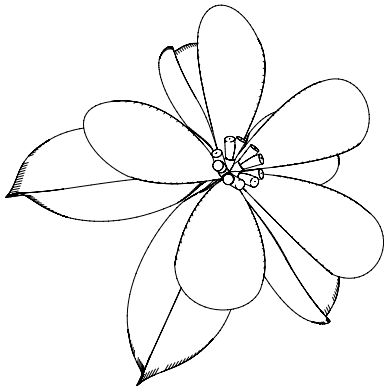

Figure 11: A stylized flower

Figure 12: Simulated development of a bluebell flower

According to production p_3, each sector consists of the *sepal* represented by a predefined surface s, the petal represented by a predefined surface p, and a configuration of line segments representing an *anther*. The exact shape of the sepals and petals is defined outside the L-system. This L-system does not simulate the process of flower development, but uses productions to capture the flower's structure in a compact, hierarchical manner.

7 Life, death, and reproduction

The L-systems considered so far were *propagating*. 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.

The original approach to simulating module death was to use *non-propagating* L-systems, which incorporate *erasing* productions [27]. In the context-free case these productions have the form $A \longrightarrow \varepsilon$, where ε denotes the empty string. Intuitively, the 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 12 (described in detail in [55]), erasing productions have been used to model the fall of petals.

The modeling task becomes more difficult when an entire structure, such as a branch, is shed

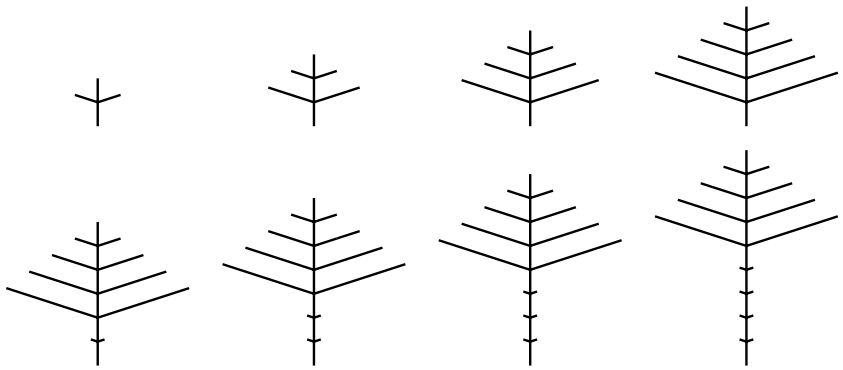

Figure 13: A developmental sequence generated by the L-system specified in Equation 5. The images shown represent derivation steps 2 through 9.

by the plant. A 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 [25] 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:

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

It is interesting to consider the operation of the cut symbol in the context of Figure 5 from Section 3. Information about the occurrence of a cut symbol does not flow from the parent module or its immediate neighbors to their children, but propagates from the cutting point to the extremities of the branches in the same manner positional information does (*Problem 7.1*).

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

$$\begin{array}{lllll} \omega: & A & & & \\ p_1: & A & & \rightarrow & F(1)[-X(3)B][+X(3)B]A \\ p_2: & B & & \rightarrow & F(1)B \\ p_3: & X(d) & : d > 0 & \rightarrow & X(d-1) \\ p_4: & X(d) & : d == 0 & \rightarrow & U\% \\ p_5: & U & & \rightarrow & F(0.3) \end{array} \tag{5}$$

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 13. The initial steps capture the growth of a *basipetal* 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 (*Problem 7.2*).

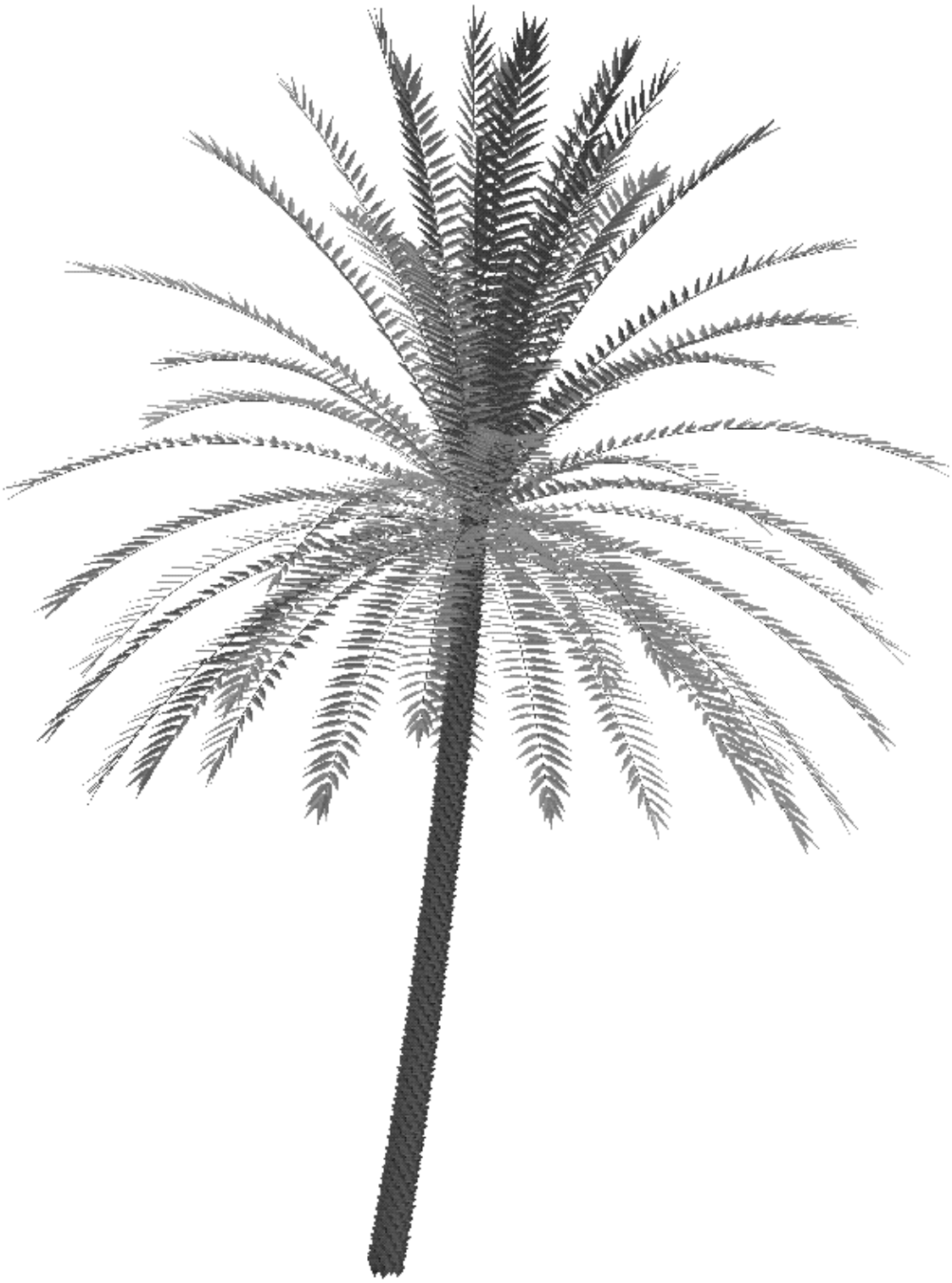

Figure 14: A model of the date palm (*Phoenix dactylifera*). This image was created using an L-system with the general structure specified in Equation 5.

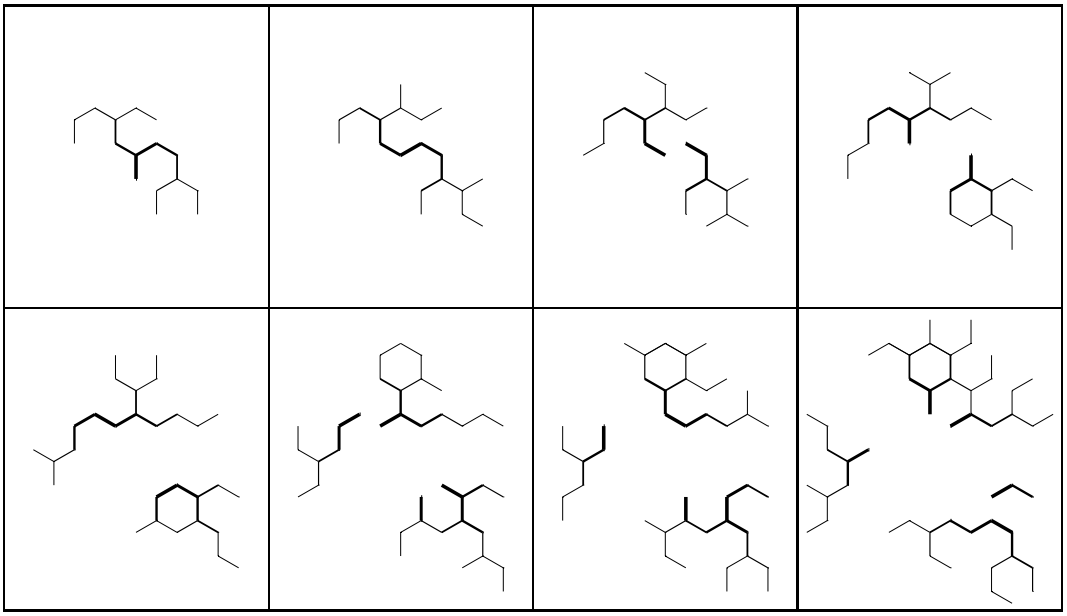

Figure 15: Development of the rhizomatous system of *Alpinia speciosa*. The images show consecutive stages of simulation generated in 6 to 13 derivation steps. Line width indicates the age of the rhizome segments. Each segment dies and disappears seven steps after its creation.

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 14). Both processes take place at the same rate, thus 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 substances through the trunk can support only a crown of a constant size.

In the case of falling leaves and branches, the parts separated from the main structure die. Separation of modules can also lead to the reproduction of plants. This phenomenon takes place, for example, when plants propagate through *rhizomes*, that is 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 [5]. A simulation carried out using an L-system reimplementation of this model is shown in Figure 15. 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.

The death of segments can be captured using productions of type $F \longrightarrow f$, which replace "old" segments F by invisible links (turtle movements) f. This replacement guarantees that the separated organisms will maintain their relative positions. Although the effect is visually correct, maintaining

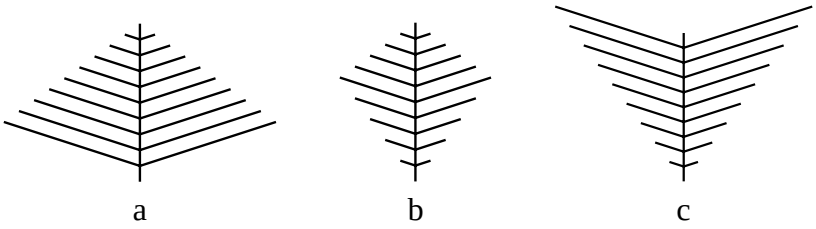

Figure 16: Basitonic (a), mesotonic (b), and acrotonic (c) branching structures differ by the position of the most developed branches on the stem.

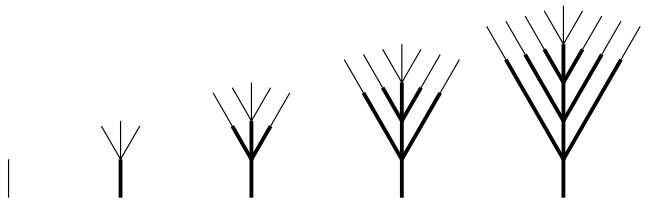

Figure 17: Development of a basitonic branching structure. The thin lines indicate segments created in the current derivation step.

any type of connection between the separated plants is artificial. An alternative solution is to extend the notion of L-systems so that they operate on *sets* of words, instead of individual words. In this case, once a branching structure becomes disconnected due to the death of an internode, each of the resulting structures develops separately (*Problems 7.3, 7.4*).

8 Information flow in growing plants

In this section we consider the propagation of control information through the structure of the developing plant (*endogenous information flow* [53]), which is captured by context-sensitive productions in the framework of L-systems. The conceptual elegance and expressive power of context-sensitive productions are among the most important assets of L-systems in modeling applications (*Problem 8.1*).

When modeling the development of branching structures, one can often divide aspects of a particular phenomenon into those that can be modeled using OL-systems, and those that cannot. For example, *acropetal* flowering sequences (with the flowering zone progressing upwards from the base of the plant) generally can be simulated using OL-systems (even without parameters), since the flowers develop in the same order in which they have been formed by the apices of their supporting branches. In contrast, *basipetal* sequences (with the flowering zone progressing downwards) require additional control mechanisms, and can be best explained in terms of the flow of control *signals* through the growing structures [31, 61, 62]. An analogous distinction can be observed between *basitonic* structures on the one hand, and *mesotonic* and *acrotonic* structures on the other hand (see Figure 16). It is intuitively clear that basitonic structures can be created using OL-systems: the lower branches are created first and, consequently, have more time to develop than the upper branches. For example, the following L-system simulates the development of the simple monopodial structure shown in Figure 17.

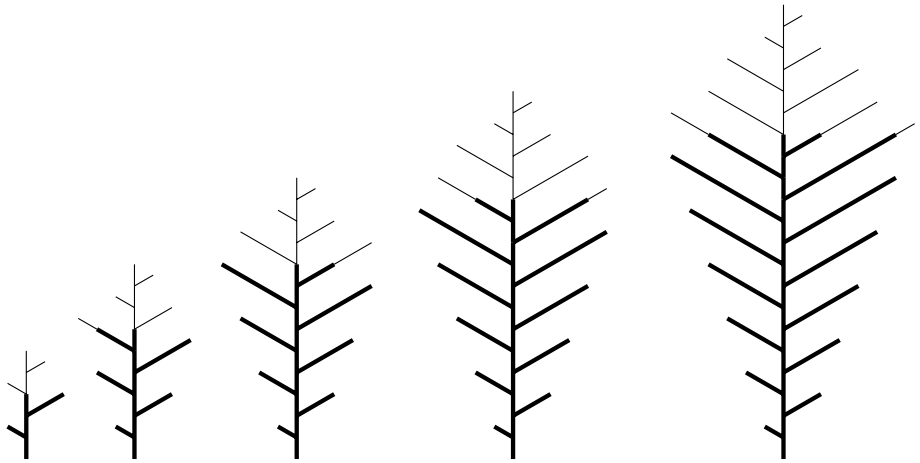

Figure 18: Development of a mesotonic branching structure controlled by an acropetal signal. Wide lines indicate the internodes reached by the signal. The stages shown correspond to derivation lengths 12 – 24 – 36 – 48 – 60

$$
\begin{array}{lcll}
\omega & : & A & \\
p_1 & : & A & \rightarrow \quad F[-B][+B]A \\
p_2 & : & B & \rightarrow \quad FB
\end{array}
$$

In contrast, indeterminate (arbitrarily large) mesotonic and acrotonic structures cannot be generated using simple deterministic OL-systems without parameters [60]. 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 [46, 47], and those postulating control of development by signals [18, 31]. The following L-system, adapted from [61, page 77], simulates the development of the mesotonic structure shown in Figure 18 using an acropetal (upward moving) signal.

$$
\begin{array}{lll}
\#\text{define} & m & 3 \quad /* \text{ plastochron of the main axis } */ \\
\#\text{define} & n & 4 \quad /* \text{ plastochron of the branch } */ \\
\#\text{define} & u & 4 \quad /* \text{ signal propagation rate in the main axis } */ \\
\#\text{define} & v & 2 \quad /* \text{ signal propagation rate in the branch } */
\end{array}
$$

$$
\text{ignore : } \quad + - /
$$

$$
\begin{array}{ll}
\omega : & S(0)F(1,0)A(0) \\
p_1 : & A(i) \quad : i < m-1 \longrightarrow A(i+1) \\
p_2 : & A(i) \quad : i == m-1 \longrightarrow [+(60)F(1,1)B(0)]F(1,0)/(180)A(0) \\
p_3 : & B(i) \quad : i < n-1 \longrightarrow B(i+1) \\
p_4 : & B(i) \quad : i == n-1 \longrightarrow F(1,1)B(0) \\
p_5 : & S(i) \quad : i < u+v \longrightarrow S(i+1) \\
p_6 : & S(i) \quad : i == u+v \longrightarrow \varepsilon \\
p_7 : & S(i) < F(l,o) : (o == 0)\&\&(i == u-1) \longrightarrow \#F(l,o)!S(0) \\
p_8 : & S(i) < F(l,o) : (o == 1)\&\&(i == v-1) \longrightarrow \#F(l,o)!S(0) \\
p_9 : & S(i) < B(j) \longrightarrow \varepsilon
\end{array}
\tag{6}
$$

The above L-system operates under the assumption that the context-sensitive production p_9 takes priority over p_3 or p_4, and that the symbols $+$, $-$, $/$ are ignored during the context matching (*c.f.* [61]). The axiom ω describes the initial structure as an internode F terminated by an apex A. The 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 *plastochrone* 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 plastochrone 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 18 shows that, for the values of plastochrones and signal propagation rates indicated in Equation 6, the lower branches have less time to develop than the higher branches, and a mesotonic structure results.

In the above model, the parameter associated with the signal was used to control its propagation rate. The signal itself acted in a binary way: it was either present or absent in a particular internode, and controlled the development by simply removing apices B from the structure. In many models, however, signals represent quantifiable entities, such as the amount of mineral substances absorbed by roots and carried upwards, or the amount of photosynthate produced by the leaves and transported down the tree. For example, Figure 19 illustrates an extension of the tree model by Borchert and Slade (*c.f.* Section 6) with an endogenously controlled mechanism for shedding branches. The model operates under the assumption that photosynthates are produced by the leaves located on the apical branch segments (shoots) and are used at a constant rate by the internodes. Information about the balance of photosynthates is carried basipetally (from the shoots towards the trunk). A branch that produces less photosynthate than it uses becomes a liability and is shed by the tree. The shedding of branches has an important impact on the structure and visual presentation of older trees (bottom row of Figure 19).

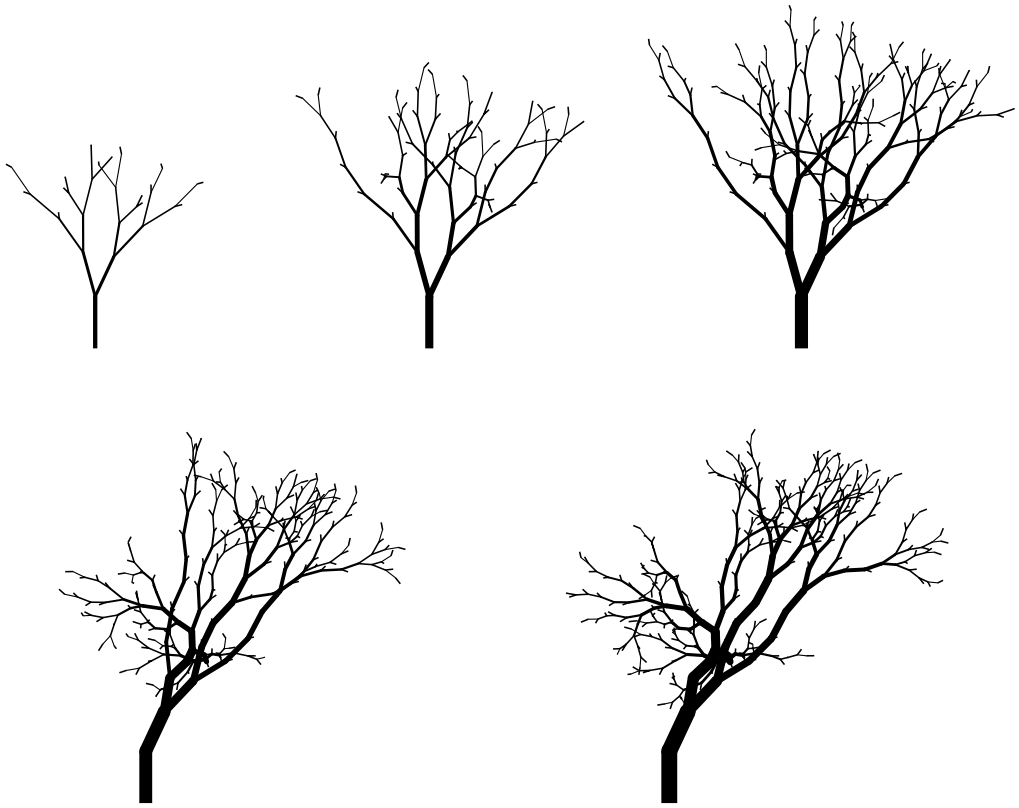

Figure 19: A modification of the tree model by Borchert and Slade [7]. The branches that produce less photosynthate than they use are shed by the tree. For clarity, leaves are not shown.

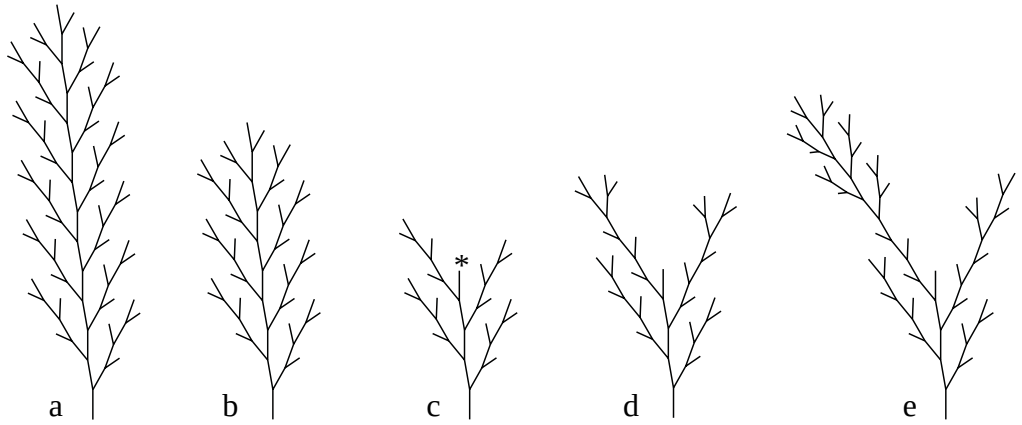

Figure 20: Development of a branching structure simulated using an L-system implementation of the model by Borchert and Honda. (a) Development not affected by pruning; (b, c) the structure immediately before and after pruning; (d, e) the subsequent development of the pruned structure.

In the above model, the shedding had only a limited effect on the development of the remaining parts of the tree. In nature, trees may compensate for the loss of a branch by the more vigorous growth of other branches. A model that captures this phenomenon has been proposed by Borchert and Honda [6], and is illustrated in Figure 20. In this case, basipetal signals originating at each node carry information about the size of branches. This information is used to allocate "fluxes" that propagate acropetally and determine the vigor of the apices. Pruning of a branch redirects the fluxes to the remaining branches and accelerates their growth.

Context-sensitive L-systems can also be used to simulate phenomena other than endogenously propagating signals. For example, Figure 21 shows a simple model of a plant attacked by an insect. The insect feeds on the apical buds; a branch that no longer carries any buds wilts. The insect is assumed to move only along the branches, thus its motions can be captured using context-sensitive L-systems. In the example under consideration, the insect systematically visits all buds, using the depth-first strategy for traversing a tree structure. Extensions of this model, including different traversing and feeding strategies, numbers of insects, etc., can be introduced.

9 Plants and the environment

The turtle interpretation of L-systems described in Section 5 creates the geometric representation of the model in a postprocessing step, with no effect on the operation of the L-system itself. Referring to Figure 5, the flow of information regarding position and orientation of the modules is postponed until all component modules are already determined. In this section we present an *environmentally-sensitive extension* of L-systems, which makes information about position and orientation of the modules available at each derivation step. Consequently, it is possible to model the influence of predefined environmental factors, such as the presence of obstacles, on a growing plant. Our presentation is based on the paper [59] and includes its edited sections.

In environmentally-sensitive L-systems, the generated string is interpreted after each derivation

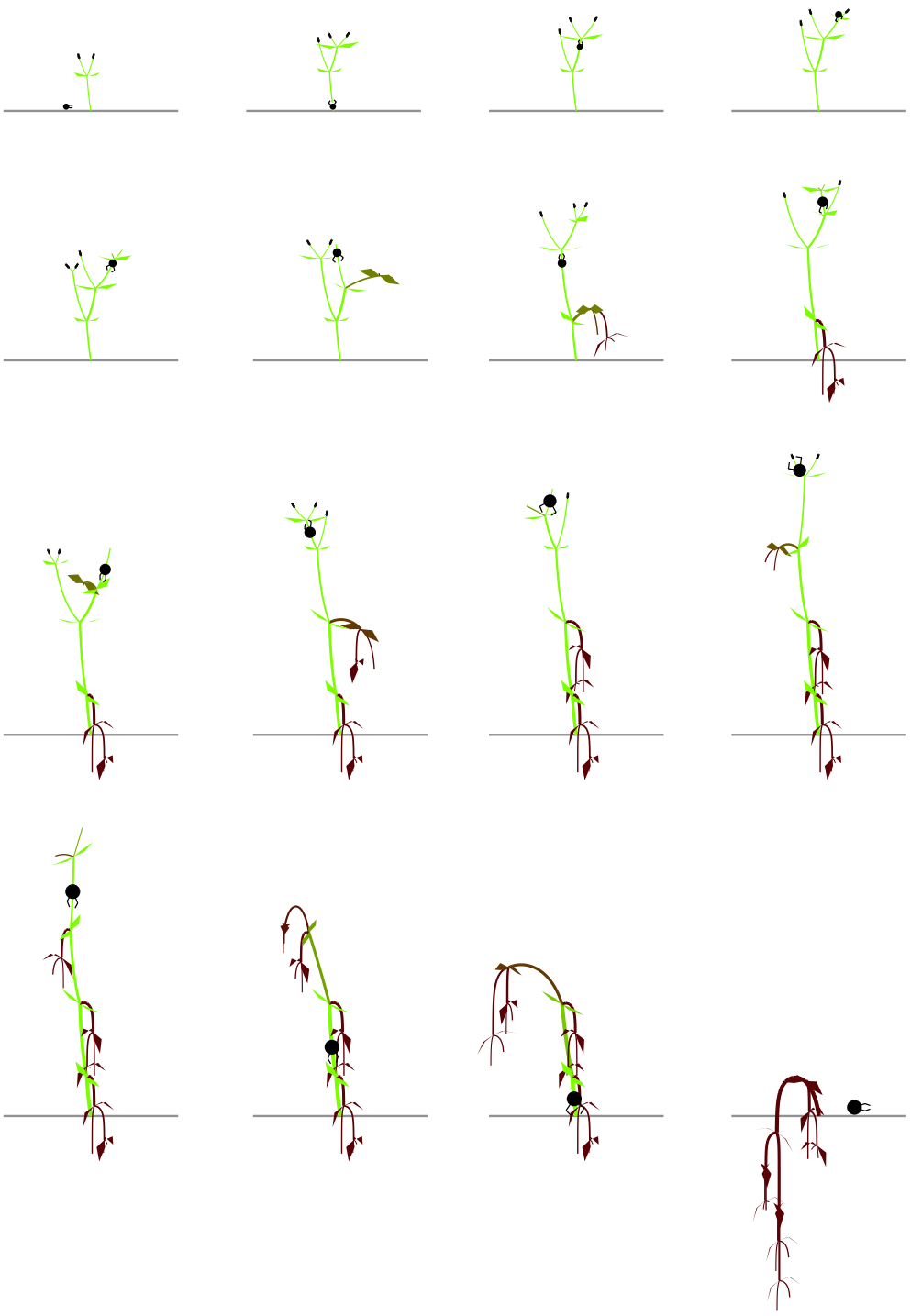

Figure 21: Simulation of the development of a plant attacked by an insect

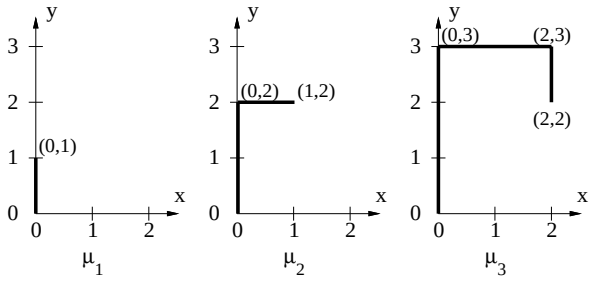

Figure 22: Assignment of values to query modules

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.

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

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

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

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 $\vec{H}$ is aligned with the y axis, vector $\vec{L}$ 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 22.

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. A more realistic example, presenting a simple model of tree response to pruning, is given below. As described, for example, by Hallé *et al.* [23, Chapter 4] and Bell [4, 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.

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

The user defined function

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

defines 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 $\vec{H}$), 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 23. 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 23, follows a similar pattern.

The above L-system simulates the response of a tree 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 6.

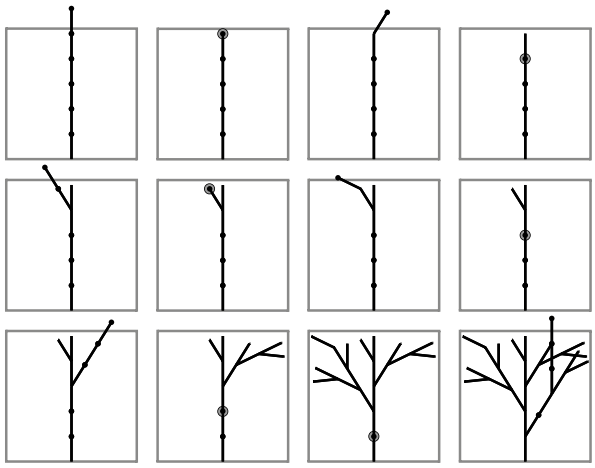

Figure 23: A simple model of a tree's response to pruning. Top row: derivation steps 6,7,8, and 10; middle row: steps 12, 13, 14, and 17; bottom row: steps 20, 40, 75, and 94. Small black circles indicate dormant buds, the larger circles indicate the position of signal S.

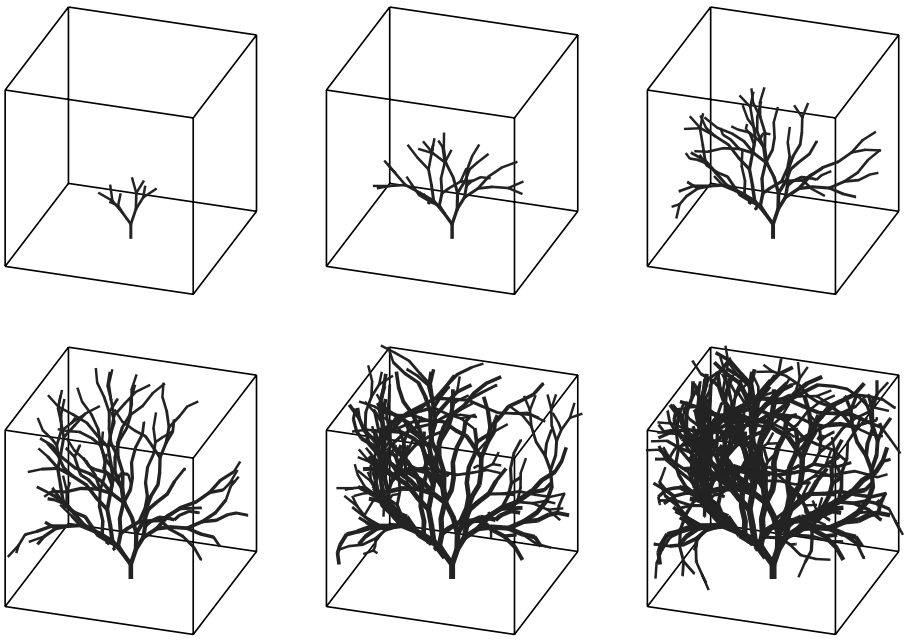

Figure 24: Simulation of tree response to pruning. The structures shown have been generated in 3, 6, 9, 13, 21, and 27 steps. Leaves have been omitted for clarity.

Figure 25: A model of the topiary garden at Levens Hall, England

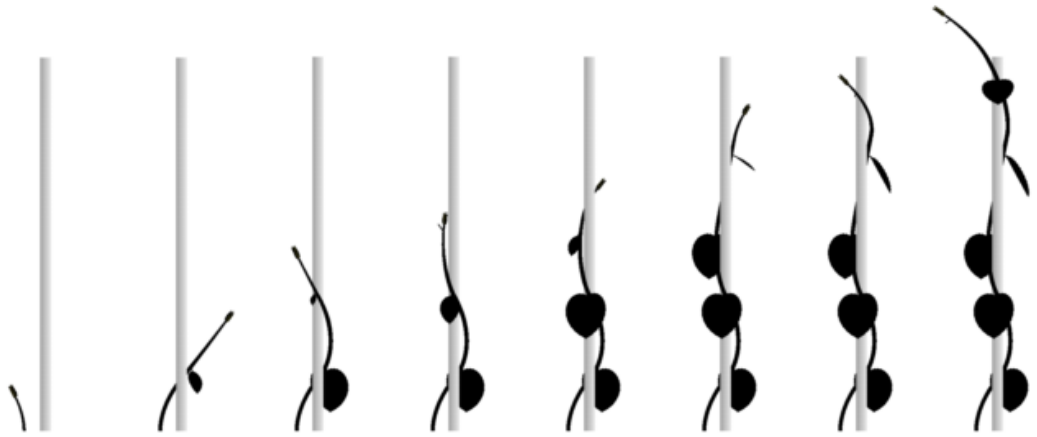

Figure 26: A simple model of a climbing plant

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

According to axiom ω, the development begins with a single internode F supporting apex $A(1)$ 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 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).

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* [4, 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 [4, page 298].

The operation of this model is illustrated in Figure 24. 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 $c = -5$.

By changing the clipping function, one can shape plant models to a variety of artificial forms. For example, Figure 25 presents a synthetic image inspired by the Levens Hall garden in England, considered the most famous topiary garden in the world [9, pages 52–57].

Pruning is only one of a range of phenomena that can be modeled using environmentally-sensitive L-systems. A different example is given in Figure 26. In this case, a climbing plant detects the presence of a supporting pole and winds around it. In contrast to the earlier models of plants growing around obstacles [2, 21, 22], the L-system model captures the *nutation*, or the spiralling movement of the free stem tip "searching" for support. Once a collision with the pole has been detected, a signal is sent basipetally (down the stem), freezing further motions of the stem below the contact point.

10 Conclusions

L-systems were introduced almost thirty years ago and have been extensively studied, yet they continue to represent a fascinating area for further research. This situation is due to several factors. Computer graphics has made it possible to visualize the structures generated by L-systems, thus turning them from a theoretical concept to a programming language for synthesizing fractals and realistic plant images. The modeling power of L-systems, made apparent by the synthetic images, has attracted a growing number of biologists interested in modular architecture and plant development. Biological applications frequently require the inclusion of environmental factors into the models, which fuels the work on environmentally-sensitive extensions to L-systems. Furthermore, the interest of biologists in modeling actual plant species is complemented by the fundamental studies of emergence in the field of artificial life. These varied interests and applications place L-systems in the center of interdisciplinary studies bridging theoretical computer science, computer graphics, biology, and artificial life. Even the material in these notes raises many nontrivial questions. Some of them are listed in the next section.

11 Problems

1.1. The importance of simulation to the studies of emergent phenomena led Darley to the following characterization [10, Page 412]:

> Emergent phenomena are those for which the amount of computation necessary for prediction from an optimal set of rules, classification and analysis, even de-

> rived from an idealised perfect understanding, can never improve upon the amount of computation necessary to simulate the system directly from our knowledge of the rules of its interactions.

Based on this characterization, propose a formal definition of emergence expressed in terms of algorithmic complexity. Illustrate your definition by examples of emergent and non-emergent phenomena. Compare your definition with Darley's own elaboration of his concept.

2.1. Hallé, Oldeman, and Tomlinson introduced the term *architectural model* to denote a program that determines successive stages of the development of a tree [23]. Compare this notion with the developmental models of plants expressed using L-systems.

2.2. Room, Maillette and Hanan classified different types of construction units that can be used to describe the modular growth of plants [64]. Analyze the relationship between these units and the notion of a module in L-systems.

2.3. Plant morphogenesis has been studied from different perspectives and at various levels of abstraction. The resulting models include:

- *genetic* [49] and *mechanistic* [20] models operating at the molecular level,
- *reaction-diffusion* models operating at the chemical level [34],
- *L-system* models operating at the level of modules,
- models of organ distribution and overall plant shape based on *ecological* arguments [29].

Using the indicated references as the starting point for investigation, explain how models operating at different levels relate to each other. Can a counterpart of the *correspondence principle* ("when theories correspond, their predictions must correspond") introduced by Niels Bohr for physics be applied to describe these relations?

3.1. Propose a possibly general yet precise definition of a production, applicable to a wide range of rewriting systems (for example, Chomsky grammars, graph grammars [13], shape grammars [72], and Koch constructions).

3.2. Biologists introduce a distinction between the *calendar age* of a plant, reflecting the objective progress of time, and *physiological age*, reflecting the state of plant development. Bell explains this distinction as follows [4, page 315],

> A tree seedling in an open habitat may be one year old and rapidly approaching the juvenile state, whilst another individual of the same species growing in a closed habitat may be many years old, and held at the seedling state until light conditions improve.

Analyze the relationship between calendar age and physiological age in the context of simulating plant development.

3.3. Formulate a definition of a Koch construction, encompassing the different variants described by Mandelbrot [48]. Compare your definition with those proposed previously [54, 63].

3.4. Propose a modification of the Koch construction, capturing the derivation shown in Figure 4b.

4.1. Chien and Jürgensen [8] introduced a parametrized extension of L-systems called VD0L-systems. Compare this concept with the parametric L-systems described in Section 4.

4.2. Compare parametric L-systems and attribute grammars [33, 35].

4.3. Using the formal definition of a Koch construction obtained as a solution to Problem 3.3, investigate the relationship between the classes of figures and developmental sequences generated by Koch constructions and L-systems with turtle interpretation. Explain why many fractals can be generated using either method [51, 56, 61].

7.1. Propose a formal definition of derivation in L-systems with the cut symbol $\%$.

7.2. In order to capture the phenomenon of dynamic equilibrium in a structure, Herman and Walker [28] (see also [66, pages 70–78]) introduced the notion of *adult languages*. By definition, a word w belongs to the adult language $L_A(G)$ of an L-system G iff w is generated by G and it derives itself: $w \Longrightarrow w$. Extend this definition to characterize biologically important situations where:

1. a part of the growing structure (such as the schematic tree crown shown in Figure 13) does not change over time, although other components of the structure may change;
2. the entire structure or some part of it (such as the crown of the palm in Figure 14) undergoes a cyclic sequence of changes.

7.3. Propose a formal definition of L-systems suitable for modeling organisms that break up into separate structures. Assume that the organisms have branching topology. Extend turtle interpretation to properly handle the geometry of the resulting sets of structures.

7.4. Rozenberg, Ruohonen, and Salomaa [65] (see also [68, 69]) introduced *L-systems with fragmentation*, which can serve as a model of reproductive processes in plants. Compare L-systems with fragmentation with your solution to the previous problem.

8.1. Compare the notions of context-sensitivity, self-modifying code, and self-replication.

10.1. The following exercise concludes the first textbook on L-systems [27, page 341]:

> Go out to a nearby field. Pick a flower. Simulate its development.

Solve this exercise for several plants. Identify simple and difficult components of the solutions. On this basis, propose areas for further research on the application of L-systems to plant modeling.

12 Acknowledgements

These notes incorporate edited versions of publications written with several co-authors. In this context, we acknowledge contributions by the late Professor Aristid Lindenmayer, and by Mark James. Many interesting ideas resulted from discussions with Dr. Peter Room; in particular, the idea of using L-systems for the simulation of interactions between plants and insects belongs to him. Lynn Mercer provided many useful comments, which we tried to incorporate into the final version of these course notes. The underlying 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] James Arvo and David 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] A. Bell. *Plant form: An illustrated guide to flowering plants*. Oxford University Press, Oxford, 1991.

[5] A. D. Bell, D. Roberts, and A. Smith. Branching patterns: the simulation of plant architecture. *Journal of Theoretical Biology*, 81:351–375, 1979.

[6] 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.

[7] R. Borchert and N. Slade. Bifurcation ratios and the adaptive geometry of trees. *Botanical Gazette*, 142(3):394–401, 1981.

[8] 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.

[9] P. Coats. *Great gardens of the Western world*. G. P. Putnam's Sons, New York, 1963.

[10] V. Darley. Emergent phenomena and complexity. In R. A. Brooks and P. Maes, editors, *Artificial Life IV. Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 411–416. MIT Press, Cambridge, 1994.

[11] M. J. M. de Boer. *Analysis and computer generation of division patterns in cell layers using developmental algorithms*. PhD thesis, University of Utrecht, 1989.

[12] 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.

[13] H. Ehrig, M. Korff, and M. Löwe. Tutorial introduction to the algebraic approach of graph grammars based on double and single pushouts. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Graph grammars and their application to computer science; Fourth International Workshop*, Lecture Notes in Computer Science 532, pages 24–37. Springer-Verlag, Berlin, 1990.

[14] J. D. Foley and A. Van Dam. *Fundamentals of interactive computer graphics*. Addison-Wesley, Reading, Massachusetts, 1982.

[15] 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.

[16] D. Frijters. Mechanisms of developmental integration of *Aster novae-angliae* L. and *Hieracium murorum* L. *Annals of Botany*, 42:561–575, 1978.

[17] D. Frijters. Principles of simulation of inflorescence development. *Annals of Botany*, 42:549–560, 1978.

[18] 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.

[19] 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.

[20] B. C. Goodwin. Generative explanations of plant form. In D. S. Ingram and A. Hudson, editors, *Shape and form in plant and fungi*, pages 3–16. Acedemic Press, London, 1994.

[21] N. Greene. Organic architecture. SIGGRAPH Video Review 38, segment 16, ACM SIGGRAPH, New York, 1988.

[22] 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.

[23] F. Hallé, R. A. A. Oldeman, and P. B. Tomlinson. *Tropical trees and forests: An architectural analysis.* Springer-Verlag, Berlin, 1978.

[24] J. S. Hanan. PLANTWORKS: A software system for realistic plant modelling. Master's thesis, University of Regina, 1988.

[25] J. S. Hanan. *Parametric L-systems and their application to the modelling and visualization of plants*. PhD thesis, University of Regina, June 1992.

[26] G. T. Herman and W. H. Liu. The daughter of CELIA, the French flag, and the firing squad. *Simulation*, 21:33–41, 1973.

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

[28] G. T. Herman and A. Walker. Context-free languages in biological systems. *International Journal of Computer Mathematics*, 4:369–391, 1975.

[29] H. S. Horn. *The adaptive geometry of trees*. Princeton University Press, Princeton, 1971.

[30] 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.

[31] 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.

[32] B. W. Kernighan and D. M. Ritchie. *The C programming language. Second edition.* Prentice Hall, Englewood Cliffs, 1988.

[33] D. E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2(2):191–220, 1968.

[34] A. J. Koch and H. Meinhardt. Biological pattern formation: from basic mechanisms to complex structures. Manuscript, Max-Planck-Institut für Entwicklungsbiologie, Tübingen, 1993.

[35] P. M. Lewis II, D. J. Rosenkrantz, and R. E. Stearns. *Compiler design theory*. Addison-Wesley, Reading, 1978.

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

[37] A. Lindenmayer. Developmental systems without cellular interaction, their languages and grammars. *Journal of Theoretical Biology*, 30:455–484, 1971.

[38] 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.

[39] A. Lindenmayer. Developmental algorithms for multicellular organisms: A survey of L-systems. *Journal of Theoretical Biology*, 54:3–22, 1975.

[40] A. Lindenmayer. Algorithms for plant morphogenesis. In R. Sattler, editor, *Theoretical plant morphology*, pages 37–81. Leiden University Press, The Hague, 1978.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] H. B. Lück and J. Lück. Approche algorithmique des structures ramifiées acrotone et basitone des végétaux. In H. Vérine, editor, *La biologie théorique à Solignac*, pages 111–148. Polytechnica, Paris, 1994.

[47] 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.

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

[49] E. M. Meyerowitz. The genetics of flower development. *Scientific American*, pages 56–65, November 1994.

[50] S. Papert. *Mindstorms: Children, computers and powerful ideas*. Basic Books, New York, 1980.

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

[52] 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.

[53] P. Prusinkiewicz. Visual models of morphogenesis. *Artificial Life*, 1:61–74, 1994.

[54] P. Prusinkiewicz and M. Hammel. Language-restricted iterated function systems, Koch constructions, and L-systems. In J. C. Hart, editor, *New directions for fractal modeling in computer graphics*, pages 4.1–4.14. ACM SIGGRAPH, 1994. Course Notes 13.

[55] P. Prusinkiewicz, M. Hammel, and E. Mjolsness. Animation of plant development. *Computer Graphics (SIGGRAPH '93 Conference Proceedings)*, pages 351–360, August 1993.

[56] P. Prusinkiewicz and J. Hanan. *Lindenmayer systems, fractals, and plants*, volume 79 of *Lecture Notes in Biomathematics*. Springer-Verlag, Berlin, 1989.

[57] 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.

[58] 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.

[59] 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.

[60] P. Prusinkiewicz and L. Kari. Sub-apical L-systems. Technical Report 95/552/4, University of Calgary, March 1995.

[61] 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.

[62] 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.

[63] P. Prusinkiewicz and G. Sandness. Koch curves as attractors and repellers. *IEEE Computer Graphics and Applications*, 8(6):26–40, November 1988.

[64] P. M. Room, L. Maillette, and J. Hanan. Module and metamer dynamics and virtual plants. *Advances in Ecological Research*, 25:105–157, 1994.

[65] G. Rozenberg, K. Ruohonen, and A. Salomaa. Developmental systems with fragmentation. *International Journal of Computer Mathematics*, 5:177–191, 1976.

[66] G. Rozenberg and A. Salomaa. *The mathematical theory of L-systems.* Academic Press, New York, 1980.

[67] 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.

[68] K. Ruohonen. Developmental systems with interaction and fragmentation. *Information and Control*, 28:91–112, 1975.

[69] K. Ruohonen. JL systems with non-fragmented axioms: the hierarchy. *International Journal of Computer Mathematics*, 5:143–156, 1975.

[70] A. Salomaa. *Formal languages.* Academic Press, New York, 1973.

[71] 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.

[72] G. Stiny. *Pictorial and formal aspects of shape and shape grammars*. Birkhäuser-Verlag, Basel and Stuttgart, 1975.

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

[74] C. E. Taylor. "Fleshing out" Artificial Life II. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 25–38. Addison-Wesley, Redwood City, 1992.