L-systems: from the Theory to Visual Models of Plants

Przemyslaw Prusinkiewicz1, Jim Hanan2, Mark Hammel1 and Radomir Mech1

1 Department of Computer Science, University of Calgary, Calgary, Alberta, Canada T2N 1N4
2 CSIRO — Cooperative Research Centre for Tropical Pest Management, Brisbane, Queensland, Australia

1 Introduction

In 1968, Aristid Lindenmayer introduced a formalism for simulating the development of multicellular organisms, subsequently named L-systems [18]. This formalism was closely related to abstract automata and formal languages, and attracted the immediate interest of theoretical computer scientists. The vigorous development of the mathematical theory of L-systems was followed by its applications to the modeling of plants. These applications gained momentum after 1984, when Smith introduced state-of-the-art computer graphics techniques to visualize the structures and processes being modeled [52]. Smith also attracted attention to the phenomenon of data-base amplification, or the possibility of generating complex structures from compact data sets, which is inherent in L-systems and forms the cornerstone of L-system applications to image synthesis. Subsequent developments (presented here from our personal perspective, without covering the fast-growing array of contributions from many other researchers) included:

- introduction of turtle interpretation of L-systems [32, 53] and refinement of a programming language based on L-systems [11, 43], which facilitated specification of the models for simulation purposes and promoted the use of L-systems as a language for describing models in publications;
- recognition of the fractal character of structures generated by L-systems, which related them to the dynamically developing science of fractals [32, 43, 38];
- increased interest in the application of computer simulations to the understanding of living processes and structures, related to the emergence of the field of Artificial Life;
- extension of the range of phenomena that can be modeled using L-systems, including, most recently, incorporation of environmental factors into the models [30, 41];
- increased understanding of the modeling process, providing a methodology for constructing models according to biological observations and measurements [45, 48].

In this paper, we revisit basic mechanisms that control plant development: lineage (cellular descent), captured by the class of context-free L-systems, and endogenous interaction (transfer of information between neighboring modules in the structure), captured by context-sensitive L-systems (c.f. [22]). Within this framework, we present several models that have been developed after the survey of L-systems in [43].

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 [18, 19]. The range of L-system applications has subsequently been extended to higher plants and complex branching structures, in particular inflorescences [8, 9], described as configurations of modules in space. 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 (Figure 1) [2, 12, 55]. 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.

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

The original version of this paper appeared in M. T. Michalewicz (Ed.): Plants to Ecosystems. Advances in Computational Life Sciences, CSIRO, Collingwood, Australia 1997, pp. 1–27.
rewriting, a production consists of a single module called the predecessor or the left-hand side, and a configuration of zero, one, 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.

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 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 [43, 4, 5, 7]. In this paper we focus on the rewriting of branching structures corresponding to cases (a) and (b).

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.

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.

It is interesting to contrast simulation of development using rewriting rules with the well known Koch construction for generating fractals [29, page 39]. The essence of the Koch construction is the replacement of straight line segments by sets of lines. Their positions, orientations, and scales are determined by the position, orientation, and scale of the segment being replaced (Figure 4a). In contrast, in models of plants, 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 bend, the subtended branches are rotated and displaced to maintain the connectivity of the structure (Figure 4b). Thus, development is simulated as a parallel application of productions, followed by a sequential connection of the child structures.

Rewriting processes maintaining the connectivity of branching structures can be defined directly in the geometric domain, but a more convenient approach is to express the generating rules and the resulting structures symbolically, using a string notation. A sequential geometric interpretation of these strings from the left (plant base) to right (branch extremities) automatically captures proper positioning of the higher branches on the lower ones. The rewriting...
of branching structures in the string domain is the cornerstone of L-systems.

The basic notions of the theory of L-systems have been presented in many survey papers [22, 20, 21, 24, 25, 26] and books [43, 38, 13, 50, 51]. Consequently, we only describe parametric L-systems, which are a particularly convenient programming tool for expressing models of plant development. Our presentation closely follows the formalization introduced in [43, 39] (see also [11, 40]).

4 Parametric L-systems

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 \( \mathbb{R} \). A module with letter \( A \in V \) and parameters \( a_1, a_2, ..., a_n \in \mathbb{R} \) is denoted by \( A(a_1, a_2, ..., a_n) \). Every module belongs to the set \( M = V \times \mathbb{R}^* \), where \( \mathbb{R}^* \) 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 \( M^* = (V \times \mathbb{R}^*)^* \) and \( M^+ = (V \times \mathbb{R}^*)^+ \), respectively.

The real-valued actual parameters appearing in the words have a counterpart in the formal parameters, which may occur in the specification of L-system productions. If \( \Sigma \) is a set of formal parameters, then \( C(\Sigma) \) denotes a logical expression with parameters from \( \Sigma \), 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 \( <, \leq, >, \geq, =, \neq \); the logical operators \( !, \&\& \), \( || \) (not, and, or); and parentheses \( () \). The expressions can also include calls to standard mathematical functions, such as a 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 [17]. 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 \( \Sigma \) are noted \( C(\Sigma) \) and \( E(\Sigma) \).

A parametric 0L-system is defined as an ordered quadruple \( G = (V, \Sigma, \omega, P) \), where:

- \( V \) is the alphabet of the system,
- \( \Sigma \) is the set of formal parameters,
- \( \omega \in (V \times \mathbb{R}^*)^+ \) is a nonempty parametric word called the axiom,
- \( P \subset (V \times \Sigma^*) \times C(\Sigma) \times (V \times E(\Sigma)^*)^* \) 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. Thus, a production has the format

\[
\text{pred} : \text{cond} \rightarrow \text{succ}.
\]

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

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

\[
\begin{array}{c|c|c}
\mu_0 : & B(2) & A(4,4) \\
\mu_1 : & B(1) & B(4) & A(1,0) \\
\mu_2 : & B(0) & B(3) & A(2,1) \\
\mu_3 : & C & B(2) & A(4,3) \\
\mu_4 : & C & B(1) & B(4)A(1,33,0) \\
\end{array}
\]

Figure 5: The initial sequence of strings generated by the parametric L-system specified in equation (2)

A production in a 0L-system 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 \( A(0) \) and one formal parameter in the production 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 \( B(1)CD(3, 7) \).

If a module \( a \) produces a parametric word \( \chi \) as the result of a production application in an L-system \( G \), we write \( a \rightarrow \chi \). Given a parametric word \( \mu = a_1a_2...a_m \), we say that the word \( \nu = x_1x_2...x_m \) is directly derived from (or generated by) \( \mu \) and write \( \mu \rightarrow \nu \) if and only if \( a_i \rightarrow x_i \) for all \( i = 1, 2, ..., m \). A parametric word \( \nu \) 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 \( \mu_0 \rightarrow \mu_1 \rightarrow ... \rightarrow \mu_n \).

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

\[
\begin{array}{l}
\omega : B(2)A(4,4) \\
p_1 : A(x, y) : y < 3 \rightarrow A(x + 2, x + y) \\
p_2 : A(x, y) : y > 3 \rightarrow B(x)A(x/y,0) \tag{2} \\
p_3 : B(x) : x < 1 \rightarrow C \\
p_4 : B(x) : x > 1 \rightarrow B(x - 1) \\
\end{array}
\]

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 5.

Productions in parametric 0L-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 general, a context-sensitive production has the format

\[
\text{le} \prec \text{pred} \succ : \text{cond} \rightarrow \text{succ},
\]
where symbols < and > separate the three components of the predecessor: a string of modules without brackets is called the left context, a module pred called the strict predecessor, and a well-nested bracketed string of modules rec called the right context. The remaining components of the production are the condition cond and the successor suc, defined as for parametric O-L-systems.

A sample context-sensitive production is given below:

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

(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

$$\ldots A(4) B(5) C(6) \ldots$$

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 $G = (V, \Sigma, \omega, P)$ is called deterministic if and only if for each module $A(t_1, t_2, \ldots, t_n) \in V \times \mathbb{R}^n$ the production set includes exactly one matching production. Within this paper we only consider deterministic L-systems.

5 The turtle interpretation of L-systems

Strings generated by L-systems may be interpreted geometrically in many different ways. Below we outline the turtle interpretation of L-systems, introduced by Szilard and Quinton [53], and extended by Prusinkiewicz [32, 33] and Hanan [11, 10]. A tutorial exposition is included in [43], and subsequent results are presented in [11]. The summary below is based on [43, 39, 33, 15].

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, 31] 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 $\vec{t}$, and the orientation is defined by three vectors $\vec{u}$, $\vec{e}$, and $\vec{v}$, indicating the turtle’s heading and the directions to the left and up (Figure 6a). These vectors have unit length, are perpendicular to each other, and satisfy the equation $\vec{u} \times \vec{e} = \vec{v}$. Rotations of the turtle are expressed by the equation:

$$[\vec{H} \quad \vec{U} \quad \vec{U}^r] = [\vec{H} \quad \vec{E} \quad \vec{V}] R,$$

where $R$ is a $3 \times 3$ rotation matrix [6]. 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

- $F(s), G(s)$ 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.
- $\odot O(r)$ Draw a sphere of radius $r$ at the current position.

Symbols that control turtle orientation in space (Figure 6a)

- $+\theta$ Turn left by angle $\theta$ around the $\vec{v}$ axis.
- $-\theta$ Turn right by angle $\theta$ around the $\vec{v}$ axis.
- $\&\theta$ Pitch down by angle $\theta$ around the $\vec{z}$ axis.
- $\wedge\theta$ Pitch up by angle $\theta$ around the $\vec{z}$ axis.
- $(\theta)$ Roll left by angle $\theta$ around the $\vec{u}$ axis.
- $\backslash(\theta)$ Roll right by angle $\theta$ around the $\vec{u}$ axis.

| | Turn $180^\circ$ around the $\vec{v}$ axis. This is equivalent to $+(180)$ or $-(180)$. |

Symbols for modeling structures with branches

- $[\quad]$ 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.

\( X(s) \approx \) 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 [33, 10].

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.

A sample string and its interpretation are shown in Figure 6b. 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 \( 45^\circ \).

6 Examples of parametric D0L-system models

This section presents selected examples that illustrate the operation of deterministic D0L-systems (D0L-systems) with turtle interpretation and their application to the modeling of plants. Many other examples are included in [11, 43, 39].

6.1 Fractal generation

Fractal curves provide a convenient means for illustrating the basic principle of L-system operation [32, 43, 38, 44]. For example, the following L-system generates the well-known snowflake curve [29, 54].

\[
\begin{align*}
\omega & : F(1) = (120)F(1) - (120)F(1) \\
pr_1 & : F(s) \rightarrow F(s/3) + (60)F(s/3) \\
& - (120)F(s/3) + (60)F(s/3)
\end{align*}
\]

The axiom \( F(1) - (120)F(1) - (120)F(1) \) draws an equilateral triangle, with edges of unit length. Production \( pr_1 \) replaces each line segment with a polygonal shape, as shown at the top of Figure 7. 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:

\[
\begin{align*}
pr_2 & : +(a) \rightarrow +(a) \\
pr_3 & : -(a) \rightarrow -(a)
\end{align*}
\]

The axiom and the figures obtained in the first three derivation steps are shown at the bottom of Figure 7.

6.2 Simulation of development

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

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

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 that consists of two internodes, two lateral apices, and a replica of the main apex (production \( pr_1 \)). An internode elongates by a constant scaling factor (production \( pr_2 \)). Production \( pr_3 \) is used to make the lines representing the internodes wider (3 units of width) than the lines representing the apices (1 unit). The branching angle associated with symbols \(+\) and \(-\) is set to \( 45^\circ \) by a global variable outside the L-system.

6.3 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 [46, 47] and the models of trees proposed by Honda [14] to study factors that determine overall tree shape. Parameter space exploration may reveal an unexpected richness of forms that can be produced by even the simplest models. For example, Figure 8 shows nine branching structures selected from a continuum generated by the following parametric D0L-system:

\[
\begin{align*}
\omega & : A(100, w_0) \\
p_1 & : A(s, w) : s \geq \min \rightarrow !(w)F(s) \\
& [+(\alpha_1)/(\varphi_1).A(s \ast r_1, w \ast q \ast e)] \\
& [+(\alpha_2)/(\varphi_2).A(s \ast r_2, w \ast (1 - q) \ast e)]
\end{align*}
\]

The single non-identity production \( pr_1 \) replaces apex \( A \) by an internode \( F \) and two new apices \( A \). The angle values \( \alpha_1, \alpha_2, \varphi_1 \), and \( \varphi_2 \) determine the orientation of these apices with respect to the subtending internode. Parameters \( s \) and \( w \) specify internode length and

![Figure 7: Visual interpretation of the production for the snowflake curve, and the curve after \( n = 0, 1, 2, \) and \( 3 \) derivation steps](image)
with different values of constants

Figure 8: Sample structures generated by a parametric D0L-system with different values of constants

Table 1: The values of constants used to generate Figure 8

<table>
<thead>
<tr>
<th>Figure</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p_1)</td>
<td>35.0</td>
<td>74.0</td>
<td>65.0</td>
<td>85.0</td>
<td>58.0</td>
<td>82.0</td>
<td>95.0</td>
<td>60.0</td>
<td></td>
</tr>
<tr>
<td>(p_2)</td>
<td>80.0</td>
<td>25.0</td>
<td>25.0</td>
<td>25.0</td>
<td>25.0</td>
<td>25.0</td>
<td>25.0</td>
<td>25.0</td>
<td></td>
</tr>
<tr>
<td>(m_0)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>(m_1)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>(m_2)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>(m_3)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>(m_4)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

width. The constants \(p_1\) and \(p_2\) determine the gradual decrease in internode length that occurs while traversing the tree from its base towards the apices. The constants \(m_0, q, r, e\) control the width of branches. The initial stem width is specified by \(m_0\) 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 [29, page 156] (see also [28, 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 1. The final column headed \(n\) indicates the number of derivation steps.

### 6.4 Modeling mesotonic and acrotonic structures

In spite of their apparent diversity, the structures generated by L-system of Section 6.3 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 are 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 9a). In 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 9 b and c). As observed by Frijters and Lindenmayer [9], and formalized by Prusinkiewicz and Kari [42], 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. For example, the following parametric D0L-system generates the mesotonic structure shown in Figure 9b.

\[
\begin{align*}
\omega & : FA(0) \\
p_1 & : A(v) \rightarrow [-FB(v)][FB(v)]FA(v + 1) \\
p_2 & : B(v) : v > 0 \rightarrow FB(v - 1)
\end{align*}
\]

The axiom \(\omega\) defines the initial structure as an internode \(F\) terminated by an apex \(A\). In each derivation step, the apex \(A\) adds a new segment \(F\) to the main axis and initiates a pair of branches \(FB\) (production \(p_1\)). The value of parameter \(v\) assigned to the lateral apices \(B\) describes the maximum length to which each branch will grow (production \(p_2\)). This value is incremented acropetally (i.e., in the ascending order of branches) by production \(p_1\), yielding a sequence of branches of increasing length. This sequence is broken in the upper part of the structure, where the branches still grow. Consequently, the younger branches near the top are shorter than the older ones further down, and a mesotonic overall structure results.

A detailed discussion of the generation of mesotonic and acrotonic structures using a construct similar to parametric L-systems has been presented by Lück, Lück, and Bakkali [27].

### 6.5 The shedding of branches

The natural processes of plant development often involve shedding, or programmed removal of selected modules from the growing structure. In order to simulate shedding, Hanan [11] 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:

\[
ab'[\text{cut}]bc'[\text{cut}]def'[\text{cut}]ghi'[\text{cut}]jkl => ab'[\text{cut}]bc'[\text{cut}]def'[\text{cut}]ghi'[\text{cut}]jkl
\]

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

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

Figure 9: Schematic representation of a basitonic (a), mesotonic (b), and acrotonic (c) branching pattern. From [42].
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 \( p_8 \)), production \( p_8 \) 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 10. The initial steps capture the growth of a basitonic structure. 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.

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 11). 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.

7 Examples of context-sensitive L-system models

In this section we consider the propagation of control information through the structure of the developing plant (endogenous information flow [34]), 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.

7.1 Development of a mesotonic structure

As outlined in Section 6.4, arbitrarily large mesotonic and acrotonic structures cannot be generated using deterministic 0L-systems without parameters [42]. 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 the L-system discussed in Section 6.4 and those postulating control of development by signals [8, 16]. The following L-system simulates the development of the mesotonic structure shown in Figure 12 using an acropetal (upward moving) signal.

Figure 10: A developmental sequence generated by the L-system specified in Section 6.5. The images shown represent derivation steps 2 through 9.

Figure 11: A model of the date palm (Phoenix dactylifera). This image was created using an L-system with the general structure specified in Section 6.5.
the branches. Production returns to state is placed at the base of this structure. According to productions A similar mechanism, based on the pursuit of apices by acropetal a mesotonic structure develops as a result. lower branches have less time to grow than the higher branches, and and signal propagation rates specified be the #define statements, the reaches it, thus terminating the development of the corresponding to the left daughter branch (a). A downward moving insect of the main axis) is controlled by the constant parameter \( c \). The total number of segments in an axis is defined by con- of segments to be produced before the next branching oc- curs. The insect attempts to traverse the entire developing structure using the depth-first strategy. A context-sensitive L-system that integrates plant growth with the behavior of the insect is given below.

\[
\begin{align*}
\text{def} & I_L & 3 & \text{/* length of the left branch */} \\
\text{def} & I_R & 5 & \text{/* length of the right branch */} \\
\text{def} & d & 5 & \text{/* plastochron */} \\
\text{def} & w & 40 & \text{/* delay */} \\
\end{align*}
\]

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

Productions \( p_1 \) to \( p_2 \) describe the development of a simple branching structure. Starting with a single axis specified by axiom \( \omega \), 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 oc- curs. The insect changes this mark 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 the above L-system is shown in Figure 14. In this case, the insect feeds on the apices of a three-dimensional structure, and a branch that no longer carries any apices withers.

Similar models can be constructed assuming different traversing
Figure 14: Simulation of the development of a plant attacked by an insect

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 [48, 49].

7.3 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 (upwards), 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 [3]. 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 number 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.

```plaintext
#define α1 10 /* branching angle - straight segment */
#define α2 32 /* branching angle - lateral segment */
#define σ0 17 /* initial flux */
#define η 0.80 /* controls input flux changes */
#define λ 0.7 /* flux distribution factor */
#define ω 5.0 /* threshold flux for branching */

ignore: + - /

ω : N(1)I(0, 2, 0, 1, A)
p1 : N(k) < I(b, m, v, c) : b == 0 & m == 2
   → I(b, 1, σ0, k - 1) * (η & k), c
p2 : N(k) > I(b, m, v, c) : b == 0 & m == 2 → N(k + 1)
p3 : I(b1, m1, v1, c1) < I(b, m, v, c) : m1 == 1 & m == 1
   → I(b, m, v, c1 - 1) * ((c1 - c)/c), c
p4 : I(b1, m1, v1, c1) < I(b, m, v, c) : m1 == 1 & b == 2
   → I(b, m, v1 - 1 * λ) * (c/(c1 - c)), c
p5 : I(b, m, v, c) : A : m == 1 & v > v1h, A
   → /((180) - (c12)I(2, v1 * (1 - λ), 1, A)
   + (c1)I(1, 2, v * λ, 1, A)
p6 : I(b, m, v, c) : A : m == 1 & v < v1h → I(b, 2, v, c)
p7 : I(b, m, v, c) > I(b2, m2, v2, c2) → I(b, m1, v1, c1) :
   m == 0 & m1 == 2 & m2 == 2
   → I(b, 2, v1, c1 + c2)
p8 : I(b, m, v, c) : m == 1 → I(b, 0, v, c)
p9 : I(b1, m1, v1, c1) < I(b, m, v, c) : m1 == 2 & m == 2
   → I(b, 0, v, c)
```

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 p1 calculates an input flux value. The expression used for this purpose, \( v = \sigma_0 \lambda^{-1} 2^\eta \), 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 p2, which increments the current cycle number k in module N.

Productions p3 and p4 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 $\lambda$ 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 $c_a$ apices, the flux reaching the lateral branch is further multiplied by the ratio $c/c_a$. The number $c_a$ 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, $c_a = c_e - c$. In total, the flux directed towards the lateral branch is equal to $v_l(1 - \lambda)(c/c_e - c)$ (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 $v_l - v_l(1 - \lambda)(c/c_e - c)$ (production $p_4$).

Productions $p_5$ and $p_6$ control the addition of new segments to the structure. According to production $p_6$, 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. Both input numbers must be available ($n_1 = 2$ and $n_2 = 2$) 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 $\omega$. 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 15.

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 16. In accordance with [3], the L-system used in this case extends the L-system discussed above with parameters and productions needed to capture the effect of aging. Consequently, a branch that was unable to grow for a given number of developments dies: it loses the ability to develop further and stops taking any fluxes.

Similar behavior is shown in Figure 17. 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 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 assumes an increase of internode width over time and a gradual rotation of a lateral segment to the straight segment position, after the straight segment has...
been lost. The developmental sequence shown in the top row of Figure 17 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. This is a non-obvious consequence of the model, which illustrates the usefulness of L-systems in predicting the global behavior of plants, given the specification of their components.

8 Conclusions

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, and continues to play a key role in current biological research using L-systems. The emergence of global forms and developmental patterns is also important in the application of L-systems to computer graphics, because it makes it possible to create realistic representations of growing plants using relatively easy to specify, compact sets of data.

In principle, the mathematical formulation of L-systems should also 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. 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 that operate on non-branching strings without geometric interpretation (for examples, see [50]). In contrast, L-system models of biological phenomena often involve parameters, interactions between modules, and geometric features of the modeled structures. We hope that further development of the theory of L-systems will bridge this gap.

9 Acknowledgements

An overview of L-systems was the subject of several invited lectures and tutorials presented recently by P. Prusinkiewicz. Consequently, this paper includes sections of previous surveys [36, 37], and coincides with [35]. The idea of using L-systems to simulate the interaction between plants and insects was proposed by Peter Room. The reported research has been sponsored by grants and graduate scholarships from the Natural Sciences and Engineering Council of Canada.

References


