

# L-systems in Geometric Modeling

Przemyslaw Prusinkiewicz

Mitra Shirmohammadi

Faramarz Samavati

Department of Computer Science, University of Calgary,  
2500 University Dr. N.W., Calgary, AB T2N 1N4, Canada

pwp@cpsc.ucalgary.ca

mitrashm@yahoo.com

samavati@cpsc.ucalgary.ca

We show that parametric context-sensitive L-systems with affine geometry interpretation provide a succinct description of some of the most fundamental algorithms of geometric modeling of curves. Examples include the Lane-Riesenfeld algorithm for generating B-splines, the de Casteljau algorithm for generating Bézier curves, and their extensions to rational curves. Our results generalize the previously reported geometric-modeling applications of L-systems, which were limited to subdivision curves.

## 1 Introduction

L-systems were conceived by Aristid Lindenmayer as a mathematical formalism for reasoning about growing multicellular organisms [14]. They were originally introduced as an extension of cellular automata, allowing for the addition and removal of cells during an automaton's operation, but were soon rephrased in terms of rewriting systems [15]. This rephrasing led to an elegant definition of L-systems using notions and notation borrowed from formal language theory. The relation to formal languages colored early studies of L-systems, which were often focused on their ability to generate different classes of languages [10, 29].

Concurrent with these developments, although initially on a smaller scale, L-systems began to be used as a mathematical foundation for the computational modeling of plants. The fundamental observation was that a growing filament, i.e., a linear arrangement of cells, can be conveniently viewed as a word over an alphabet of cell states. This abstraction was generalized to branching structures and components larger than individual cells, leading to astonishingly succinct descriptions of growing plants [24].

In retrospect, the success of L-systems in plant modeling can be attributed to the following factors [21]:

1. L-systems specify development in terms of temporally and spatially local declarative rewriting rules — context-free and context-sensitive productions — which express developmental processes in a concise and intuitive manner.
2. L-systems describe growing structures in terms of topological (neighborhood) relations between structure components, which are automatically maintained when components are added to or removed from a structure. Consequently, the context for production application is always available.
3. L-systems refer to model components by their type, state and context, rather than a unique name or position in the structure. This simplifies the specification of developmental algorithms. In particular, indices are no longer needed.

These features are not only important to the modeling of biological development, but also make L-systems well suited to the description and implementation of some non-biological algorithms [22]. Typically, they are characterized by the repetitive application of relatively simple rules to discrete structures with a changing number of components. Early studies of such algorithms led to the concise L-system

description of linear fractals, including classic space-filling curves [19, 25, 31, 33]. The geometric interpretations of L-systems used in these studies were based on turtle geometry [1] or chain coding [7].

More recently, L-systems with an interpretation based on affine geometry [4, 5, 8, 9] have been demonstrated to provide a concise description of the subdivision curves used in geometric modeling [20, 26]. Similar to the algorithms for generating fractal curves, subdivision algorithms operate on discrete, polygonal lines with a changing number and configuration of components. The rules of change can be conveniently described in local terms and formalized as L-systems productions. A comparison between L-systems and the traditional specification of subdivision algorithms using indexed points and/or matrices reveals the advantages of L-systems [26].

Here we expand the range of geometric modeling applications of L-systems and show that L-systems allow for the succinct formulation of some of the most fundamental algorithms for geometric modeling of curves. These include the Lane-Riesenfeld algorithm for generating B-splines, the de Casteljau algorithm for generating Bézier curves, and their extensions to rational curves. Further examples are discussed in detail by Shirmohammadi [30], whose work was a stepping stone for the present paper. Compared to [30], we use a topologically more accurate representation of polygons, rooted in the notion of a cell complex [18], with both vertices and edges explicitly represented. This results in a more intuitive specifications of the algorithms, reflecting their inherent symmetries.

## 2 Preliminaries

The L-systems employed here are context-sensitive parametric L-systems [16, 24], extended in two directions:

- Parameters are not limited to numbers, but may also be compound data structures, for example representing points or vectors in two or three dimensions [11, 26];
- Productions may be grouped into subsets (tables [28]), with a control mechanism deciding which subset is applicable in each derivation step [3, 23, 34].

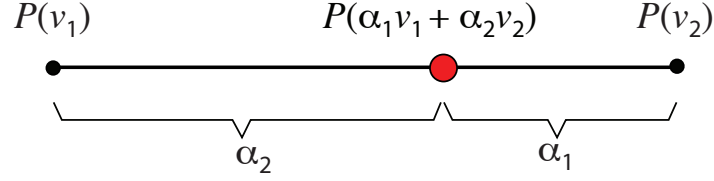
The key concepts can be summarized as follows. Parametric L-systems operate on parametric words, or strings of modules. Each module is a letter from the L-system alphabet  $V$ , which may be associated with one or more optional parameters. Beginning with an explicitly defined axiom (initial string), an L-system generates a developmental sequence of words using a finite set of productions of the form:

$$\text{label} : \text{left context} < \text{strict predecessor} > \text{right context} : \text{condition} \rightarrow \text{successor}. \quad (1)$$

In each derivation step, productions are applied in parallel to all modules in the predecessor string. If several productions apply, the first production in the list is chosen. If no production applies, a module is rewritten into itself. For example, the L-system with axiom  $\omega$  and production  $p_1$  to  $p_3$ , given below

$$\begin{aligned} \omega &: A(1.5)B(2.0, 3.0)A(4.5)C(1) \\ p_1 &: A(x) : x \leq 2 \rightarrow A(2x + 1) \\ p_2 &: A(x) : x > 2 \rightarrow B(2x + 1) \\ p_3 &: A(w) < B(x, y) > A(z) \rightarrow A(w + x)A(y + z) \end{aligned} \quad (2)$$

generates the word  $A(4)A(3.5)A(7.5)B(10)C(1)$  in the first derivation step. In addition to proper L-systems, in which the strict predecessor is always a single module, we consider pseudo-L-systems [19], in which the strict predecessor may be a nonempty parametric word over the alphabet  $V$ . We also consider L-systems operating on circular words, in which the first and last module are considered neighbors.

Figure 1: Affine combination of points  $P(v_1)$  and  $P(v_2)$ .

Parameters may be  $d$ -tuples of numbers representing positions  $v$  in a  $d$ -dimensional space [26]. An affine combination  $v$  of  $m$  positions  $v_1, v_2, \dots, v_m$  is given by the expression

$$v = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_m v_m, \quad (3)$$

where the scalar coefficients  $\alpha_i$  add up to 1:

$$\alpha_1 + \alpha_2 + \dots + \alpha_m = 1. \quad (4)$$

The meaning of the affine combination (3) is derived from its transformation to the form

$$v = v_1 + \alpha_2(v_2 - v_1) + \dots + \alpha_n(v_m - v_1), \quad (5)$$

which is a well-defined expression of vector algebra. Specifically, for two positions  $v_1$  and  $v_2$  we obtain:

$$v = \alpha_1 v_1 + \alpha_2 v_2 = v_1 + \alpha_2(v_2 - v_1) = v_2 + \alpha_1(v_1 - v_2), \quad (6)$$

which means that point  $P$  with position  $v$ , noted  $P(v)$ , divides line  $\overline{P(v_1)P(v_2)}$  in proportion  $\alpha_2 : \alpha_1$  (Figure 1).

### 3 B-splines

A simple example of the application of L-systems to geometric modeling is an L-system specification of the Chaikin algorithm [2]. Given a (closed) control polygon  $P(v_0)P(v_1)\dots P(v_m)$ , this algorithm produces a smooth (at the limit) curve, whose shape can be thought of as the result of iteratively cutting the corners of the control polygon and its descendants (Figure 2). This process can be succinctly specified by a context-sensitive L-system operating on circular words:

$$\begin{aligned} \omega &: P(v_1)P(v_2)\dots P(v_m) \\ p &: P(v_l) < P(v) > P(v_r) \rightarrow P(\frac{1}{4}v_l + \frac{3}{4}v)P(\frac{3}{4}v + \frac{1}{4}v_r) \end{aligned} \quad (7)$$

It is often convenient to explicitly represent not only the vertices, but also the edges of the polygons on which the Chaikin algorithm operates. Such a representation is provided by the following modification of L-system 7:

$$\begin{aligned} \omega &: P(v_1)EP(v_2)E\dots P(v_m)E \\ p_1 &: P(v_l) < E > P(v_r) \rightarrow P(\frac{3}{4}v_l + \frac{1}{4}v_r)EP(\frac{1}{4}v_l + \frac{3}{4}v_r) \\ p_2 &: P(v) \rightarrow E \end{aligned} \quad (8)$$

A point  $P(v)$  carries all the information needed to visualize it as, for example, a small circle centered at  $v$ . In contrast, the visualization of the edge  $E$  between points  $P(v_l)$  and  $P(v_r)$  requires an interpretation

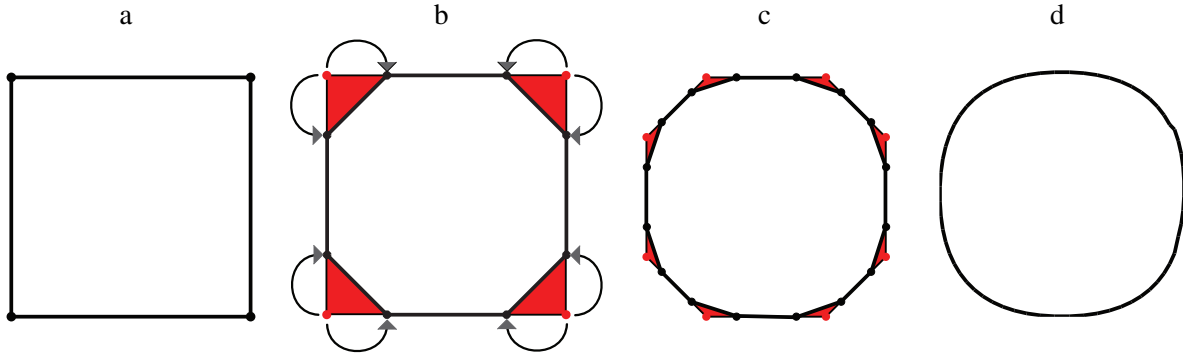


Figure 2: Illustration of Chaikin's algorithm (adapted from [26]). Beginning with a user-specified control polygon (a), the algorithm iteratively removes the polygon corners (red triangles in panels b and c). Each old vertex is thus replaced by a pair of new vertices, moved  $\frac{1}{4}$  of the distance toward the neighboring old vertices (arrows in panel b). An approximately smooth curve obtained after 4 derivation steps is shown in panel d.

rule [11, 12] that is applied at the end of the derivation and gathers the information about the line's endpoints:

$$h_E : P(v_l) < E > P(v_r) \rightarrow L(v_l, v_r) \quad (9)$$

Module  $L$  now carries all the information needed to draw a line from  $v_l$  to  $v_r$ . We assume that production 9 complements all L-systems using explicit edge representation (module  $E$ ).

A variant of L-system 8 is given below:

$$\begin{aligned} \omega &: P(v_1)EP(v_2)E \dots P(v_m)E \\ p &: P(v_l) < E > P(v_r) \rightarrow EP(\frac{1}{2}v_l + \frac{1}{2}v_r)E \\ q_1 &: P(v_l) < E > P(v_r) \rightarrow P(\frac{1}{2}v_l + \frac{1}{2}v_r) \\ q_2 &: P(v) \rightarrow E \end{aligned} \quad (10)$$

The set of productions is partitioned here into two subsets, labeled  $p$  and  $q$ . Production  $p$  inserts a new vertex in the middle of each edge, thus subdividing it into two halves. Productions  $q_1$  and  $q_2$  replace the predecessor polygon with a new polygon that has all vertices placed at the midpoints of the predecessor's edges. Derivation proceeds in cycles. Each cycle consists of a single application of production  $p$  followed by  $n \geq 0$  applications of productions  $q_1$  and  $q_2$  (Figure 3). For  $n = 1$ , such a cycle produces the same result as one derivation step in L-system 8 (compare the middle row in Figure 3 with Figure 2). Lane and Riesenfeld have shown that the polygons generated by L-system 10 converge to (uniform) B-spline curves of degree  $n + 1$  [13] (see also [17, 32]). B-splines are widely used in geometric modeling due to their well understood geometric properties, and the relative ease with which diverse curves can be defined by interactively positioning the control points. L-system 10 provides a very concise description of this class of curves.

## 4 The de Casteljau algorithm

The de Casteljau algorithm is considered "probably the most important algorithm of all of computer-aided geometric design" [6, p. 32]. Given an open control polygon  $P(v_1^0)P(v_2^0) \dots P(v_n^0)$  with  $n \geq 2$  ver-

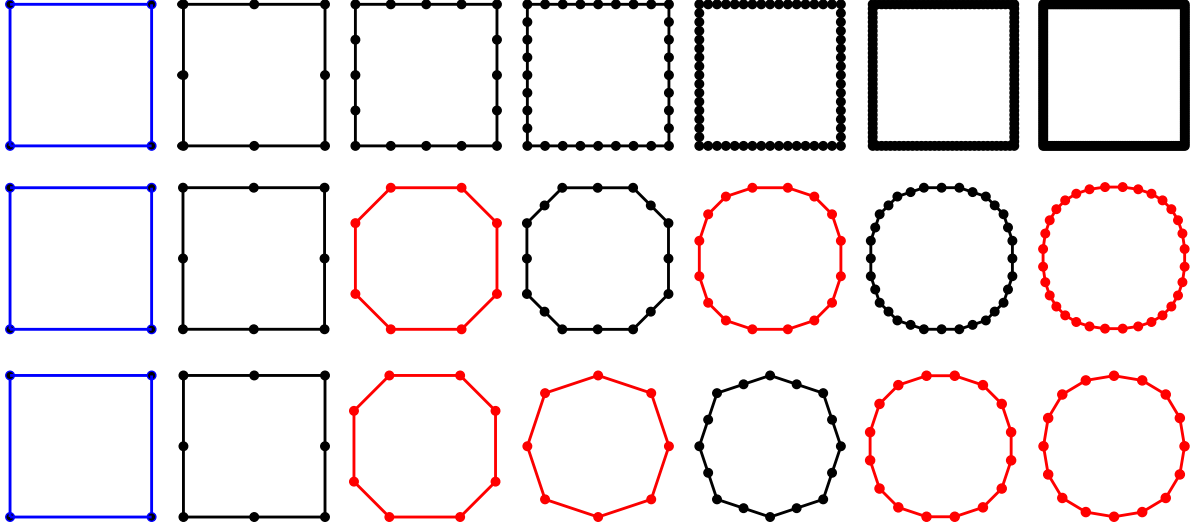


Figure 3: Illustration of the Lane-Riesenfeld algorithm for generating B-splines of arbitrary degree  $n + 1$  (L-system 10). Beginning with the initial polygon (shown in blue), the algorithm proceeds in cycles consisting of a single derivation step using production  $p$  (result shown in black) followed by  $n \geq 0$  steps using productions  $q_1$  and  $q_2$  (results shown in red). Derivations of a linear ( $n = 0$ ), quadratic ( $n = 1$ ) and cubic B-spline ( $n = 2$ ) are shown in consecutive rows.

tices  $P$  located at  $v_1^0, v_2^0, \dots, v_n^0$ , the algorithm constructs a polygon  $P(v_1^1)P(v_2^1) \dots P(v_{n-1}^1)$  such that each point  $P(v_i^1)$  subdivides the line segment  $\overline{P(v_i^0)P(v_{i+1}^0)}$  in proportion  $t : 1 - t$ . The number  $t$  is a parameter ranging between 0 and 1. This process is iterated  $n - 1$  times, ending with a single point  $P(v_1^{n-1})$ . The locus of points  $P(v_1^{n-1})$  obtained by the application of this algorithm for all  $t \in [0, 1]$  is a curve, called the Bézier curve of degree  $n - 1$  defined by the control points (or polygon)  $P(v_1^0)P(v_2^0) \dots P(v_n^0)$  (Figure 4). As with B-splines, the importance of Bézier curves stems from their well understood mathematical properties and the ease with which their shapes can be specified or changed by manipulating the control polygons.

The above process of finding a curve point corresponding to the parameter value  $t$  can be concisely expressed using a simple L-system with axiom  $\omega$  and productions  $p_1, p_2$ :

$$\begin{aligned}
 \omega &: P(v_1)P(v_2) \dots P(v_n) \\
 p_1 &: P(v) > P(v_r) \rightarrow P((1-t)v + tv_r) \\
 p_2 &: P(v) \rightarrow \varepsilon
 \end{aligned} \tag{11}$$

The first production replaces each point in the predecessor string with an affine combinations of this point and its neighbor to the right, thus capturing the essence of the de Casteljau algorithm. The second production erases the last point in the sequence. The same results are obtained by replacing the right-context-sensitive production  $p_1$  with its left-context-sensitive counterpart:

$$p'_1 : P(v_l) < P(v) \rightarrow P((1-t)v_l + tv) \tag{12}$$

Unfortunately, neither production  $p_1$  nor  $p'_1$  captures the inherent symmetry of the de Casteljau construction. As in the case of L-systems 8 and 10, this shortcoming can be addressed by representing the control

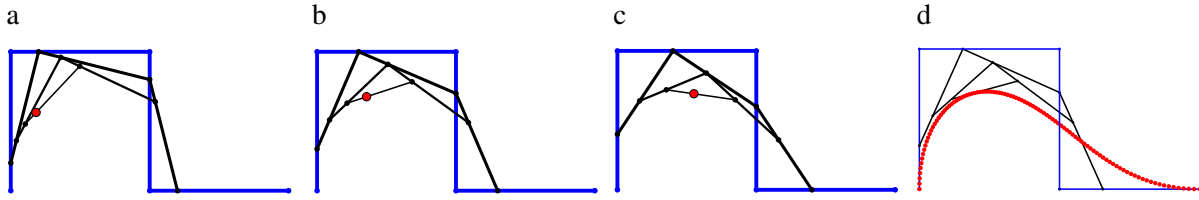


Figure 4: Illustration of the de Casteljau algorithm (L-system 13). a-c) Points obtained for parameter values  $t = 0.2, 0.3$  and  $0.4$ , respectively. The initial (open) control polygon is shown in blue, the polygons obtained in subsequent derivation steps are black, and the resulting point is shown in red. d) The quartic Bézier curve obtained as a locus of points generated by the de Casteljau algorithm with the parameter  $t$  increasing from 0 to 1 in increments  $\Delta t = 0.01$ .

polygon as a one-dimensional cell complex: a sequence of points  $P$  connected by edges  $E$ :

$$\begin{aligned}
 \omega &: P(v_1)EP(v_2)E \dots EP(v_n) \\
 p_1 &: P(v_l) < E > P(v_r) \rightarrow P((1-t)v_l + tv_r) \\
 p_2 &: E < P(v) > E \rightarrow E \\
 p_3 &: P(v) \rightarrow \varepsilon
 \end{aligned} \tag{13}$$

Production  $p_1$  now replaces an edge  $E$  with a vertex  $P$  dividing this edge in proportions  $t : 1 - t$ . Production  $p_2$  performs a dual operation, replacing the vertex between two edges with an edge. Finally, production  $p_3$  erases the first and the last vertex of the predecessor polygon from the successor polygon. The operation of this L-system is illustrated in Figure 4, and the corresponding information flow is given by the derivation in Figure 5a.

An alternative to the sequential evaluation of consecutive points approximating a Bézier curve can be described as follows. Points at the inclined boundaries of the derivation tree (Figure 5a) define two new control polygons with the same number of vertices as the original control polygon. It is known that Bézier curves defined by these new polygons subdivide the original curve into two segments [13] (see also [6, page 34], for example). Furthermore, the union of the new polygons is closer to the curve than the original polygon. A Bézier curve can thus be generated by subdivision, i.e., by constructing two control polygons that define the same curve as the original polygon, and iterating this process until a desired accuracy of approximation of the curve by the union of its control polygons has been reached.

The derivation tree for L-system 14 implementing one subdivision level is shown in Figure 5b. The consecutively established vertices of the new control polygons propagate from one derivation step to the next until the resulting string is complete. The algorithm builds new control polygons sequentially, by proceeding from the endpoints of the given control polygon inward. The meeting point — the final result of the de Casteljau algorithm — is shared by both new polygons. The L-system is given below:

$$\begin{aligned}
 \omega &: P(v_1, 2)EP(v_2, 0)E \dots P(v_{n-1}, 0)EP(v_n, 2) \\
 p_1 &: P(v_l, s_l) < E > P(v_r, s_r) \rightarrow P((1-t)v_l + tv_r, f(s_l, s_r)) \\
 p_2 &: E < P(v, s) > E \quad : s = 0 \rightarrow E \\
 p_3 &: E < P(v, s) > E \quad : s \neq 0 \rightarrow IP(v, s)I \\
 p_4 &: P(v, s) > E \quad : s \neq 0 \rightarrow P(v, s)I \\
 p_5 &: E < P(v, s) \quad : s \neq 0 \rightarrow IP(v, s)
 \end{aligned} \tag{14}$$

We now explain the operation of this L-system in detail. To produce the derivation tree in Figure 5b,

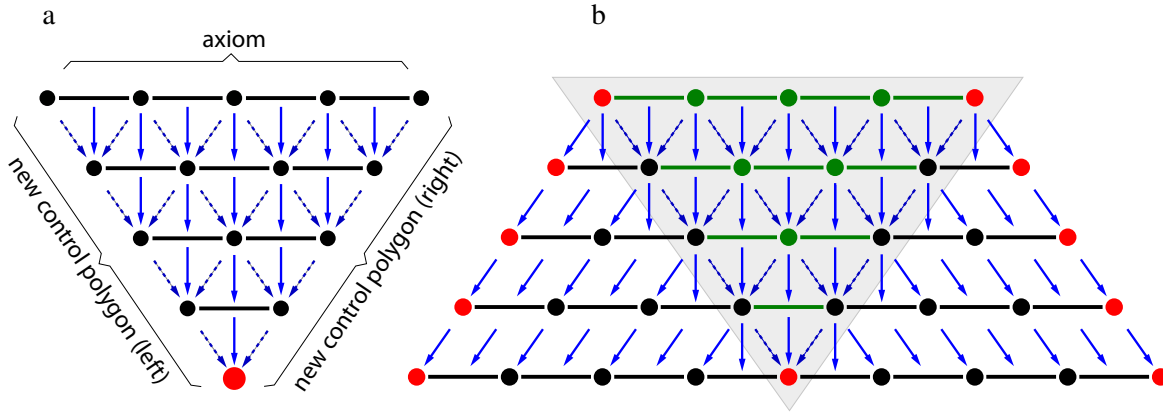


Figure 5: (a) Example of the derivation tree generated by L-system 13, which implements the de Casteljau algorithm. A quartic Bézier curve ( $n = 5$ ) has been assumed. The L-system operates on polygons defined as sequences of vertices (circles) and edges (horizontal lines). The L-system axiom is given in the first row. Solid arrows relate the strict predecessor of each production to its successor. Dashed lines indicate the flow of contextual information specifying the location of each vertex. The final result is a single vertex shown in red. (b) Example of the derivation tree generated by L-system 14, which approximates Bézier curves of arbitrary degree using repetitive subdivision. The first subdivision level of a quartic curve is shown. The central part of the subdivision tree (within the shaded inverted triangle) implements the de Casteljau algorithm as shown in panel (a). The left and right parts of the derivation tree propagate the vertices and edges established in the previous steps. In order to operate properly, L-system 14 distinguishes between vertices and edges of different states and types. For vertices, green denotes state  $s = 0$ ; black:  $s = 1$ ; red:  $s = 2$ . For edges, green denotes type  $E$ ; black: type  $I$ .

each point is characterized by its position  $v$  (first parameter) and state  $s$  (second parameter). The states represent the following information:

- $s=2$ : an endpoint of a control polygon;
- $s=1$ : a previously established interior vertex of the polygon being constructed;
- $s=0$ : any other point.

In addition, a distinction is made between edges  $I$  that connect pairs of resultant points (including the endpoints) and edges  $E$  that are still subject to changes by the algorithm. The axiom  $\omega$  specifies the initial control polygon, distinguishing between its endpoints and all other points. Production  $p_1$  and  $p_2$  correspond to productions with the same labels in L-system 13 and capture the essence of the de Casteljau algorithm. A new element is function  $f$  in production  $p_1$ , which assigns a state to the resultant point. The following rules apply to a vertex in state 0:

- a vertex adjacent to an endpoint or one interior point becomes an interior point ( $s=1$ );
- a vertex adjacent to two interior points becomes a new endpoint ( $s = 2$ ).

The remaining points retain their states. With the assumed coding of states, function  $f$  can be written as

$$f(s_l, s_r) = \min(s_l, 1) + \min(s_r, 1). \tag{15}$$

The assignment of states allows the L-system to distinguish between points and edges that are still subject of the Casteljau algorithm (productions  $p_1$  and  $p_2$ ) and points that are ready to be propagated to the

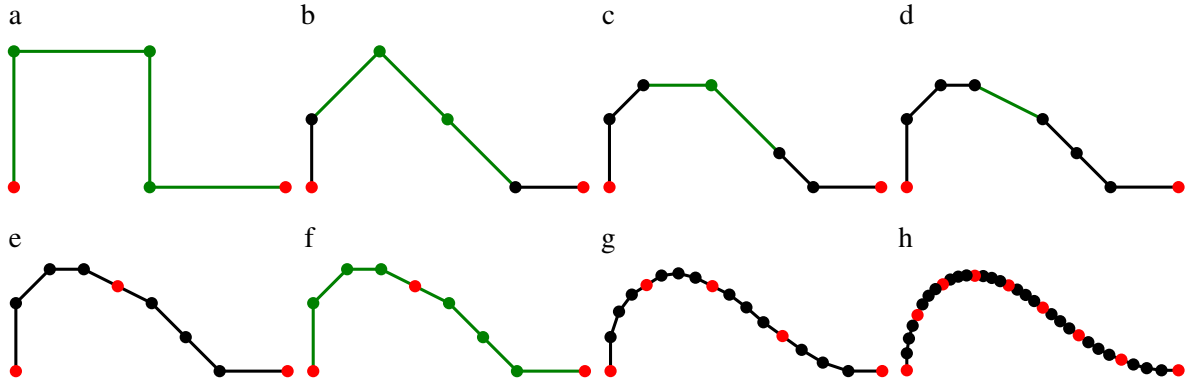


Figure 6: Generation of a quartic Bézier curve with the de Casteljau subdivision algorithm. a) The initial polygon. b-e) The first subdivision cycle using production set 14. f) Re-initialization preceding the second subdivision cycle using production set 16. g-h) The result of the second and third cycle of subdivision. Vertex states and edge types are color-coded as in Figure 5.

resulting string (productions  $p_3$  to  $p_5$ ). The propagation itself is effected by the identity productions assumed to operate on points and edges to which no other production applies.

To iterate the subdivision process, the states of points and the types of edges are re-initialized before the next subdivision cycle using productions:

$$\begin{aligned} q_1 : P(v, s) : s = 1 &\rightarrow P(v, 0) \\ q_2 : I &\rightarrow E \end{aligned} \quad (16)$$

Given an initial polygon with  $n + 1$  points, the L-system with productions 14 and 16 thus generates a Bézier curve of degree  $n$  by iterating the cycle of  $n$  derivation steps using production set 14, followed by one step using production set 16. The parameter  $t$  in production  $p_1$  is typically set to 0.5 to achieve a relatively uniform distribution of points approximating the curve. An example of a quartic Bézier curve generated by this process is shown in Figure 6.

If the curve degree is known in advance, the result of a complete subdivision cycle applied to a set of internal points can be precomputed and encapsulated in a single production. For example, the following L-system generates quadratic Bézier curves:

$$\begin{aligned} \omega : P(v_1)EQ(v_2)EP(v_3) \\ p : P(v_l)E < Q(v) > EP(v_r) &\rightarrow Q(\frac{1}{2}v_l + \frac{1}{2}v)EP(\frac{1}{4}v_l + \frac{1}{2}v + \frac{1}{4}v_r)EQ(\frac{1}{2}v + \frac{1}{2}v_r) \end{aligned} \quad (17)$$

Here we distinguish points by their type ( $P$  or  $Q$ ) rather than state. Points  $P$  are the endpoints of a control polygon, and are not affected in the subsequent derivation steps. Points  $Q$  are the interior vertices and can be replaced with another set of vertices in the next derivation step. Generation of a quadratic Bézier curve using L-system 17 is illustrated in Figure 7a.

The principle of this construction carries over to Bézier curves of degrees  $n > 2$ . However, the number  $n - 1$  of the interior control points that need replacing is then greater than one. Such a replacement can be effected most simply using a pseudo-L-system, in which multi-module strict predecessors can be rewritten at once [19]. For example, a cubic Bézier curve (Figure 7b) is generated by the following



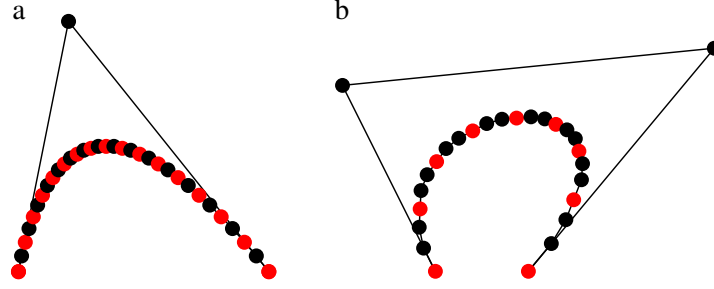


Figure 7: Illustration of the de Casteljau subdivision algorithm, specialized to curves of a given degree. a) A quadratic Bézier curve generated using L-system 17. b) A cubic Bézier curve generated using L-system 18 (or 19). Points  $P$  are shown in red, points  $Q$  in black.

pseudo-L-system:

$$\begin{aligned}
 \omega &: P(v_1)EQ(v_2)EQ(v_3)EP(v_4) \\
 p &: P(v_{ll})E < Q(v_l)EQ(v_r) > EP(v_{rr}) \rightarrow \\
 &\quad Q(\frac{1}{2}v_{ll} + \frac{1}{2}v_l)EQ(\frac{1}{4}v_{ll} + \frac{1}{2}v_l + \frac{1}{4}v_r) \\
 &\quad EP(\frac{1}{8}v_{ll} + \frac{3}{8}v_l + \frac{3}{8}v_r + \frac{1}{8}v_{rr})E \\
 &\quad Q(\frac{1}{4}v_l + \frac{1}{2}v_r + \frac{1}{4}v_{rr})EQ(\frac{1}{2}v_r + \frac{1}{2}v_{rr})
 \end{aligned} \tag{18}$$

An equivalent proper L-system can be obtained by dividing the predecessor and the successor of production  $p$  into three parts, for example:

$$\begin{aligned}
 \omega &: P(v_1)EQ(v_2)EQ(v_3)EP(v_4) \\
 p_1 &: P(v_{ll})E < Q(v_l) > EQ(v_r)EP(v_{rr}) \rightarrow Q(\frac{1}{2}v_{ll} + \frac{1}{2}v_l)EQ(\frac{1}{4}v_{ll} + \frac{1}{2}v_l + \frac{1}{4}v_r) \\
 p_2 &: P(v_{ll})EQ(v_l) < E > Q(v_r)EP(v_{rr}) \rightarrow EP(\frac{1}{8}v_{ll} + \frac{3}{8}v_l + \frac{3}{8}v_r + \frac{1}{8}v_{rr})E \\
 p_3 &: P(v_{ll})EQ(v_l)E < Q(v_r) > EP(v_{rr}) \rightarrow Q(\frac{1}{4}v_l + \frac{1}{2}v_r + \frac{1}{4}v_{rr})EQ(\frac{1}{2}v_r + \frac{1}{2}v_{rr})
 \end{aligned} \tag{19}$$

Thus, L-systems succinctly express both the basic de Casteljau algorithm and extensions that generate Bézier curves of arbitrary or fixed degree using subdivision.

## 5 Rational curves

The algorithms discussed so far have been illustrated with examples of planar curves. However, the assumption of planarity is not necessary, and the algorithms operate equally well in three and more dimensions. This provides a means of generating curves in space, and also leads to a useful extension of Bézier and B-spline curves to their rational counterparts [6, 27].

Rational curves are generated in a higher-dimensional space, then projected to lower dimensions using a perspective projection. Assuming  $z \neq 0$ , the projection of a 3D point  $P(x, y, z)$  on the plane  $z = 1$  from the origin  $O(0, 0, 0)$  of the underlying coordinate system is the 2D point  $P'(\frac{x}{z}, \frac{y}{z})$ . Identifying point locations with their coordinates, this projection can be accomplished by the interpretation rule

$$h_P: P(x, y, z) \rightarrow P'(\frac{v_x}{v_z}, \frac{v_y}{v_z}) \tag{20}$$

The subsequently applied edge interpretation rule (production 9) should operate on the projected points  $P'$  rather than the original 3D points  $P$ , thus taking the form

$$h_E: P'(v'_l) < E > P'(v'_r) \rightarrow L(v_l, v_r) \tag{21}$$

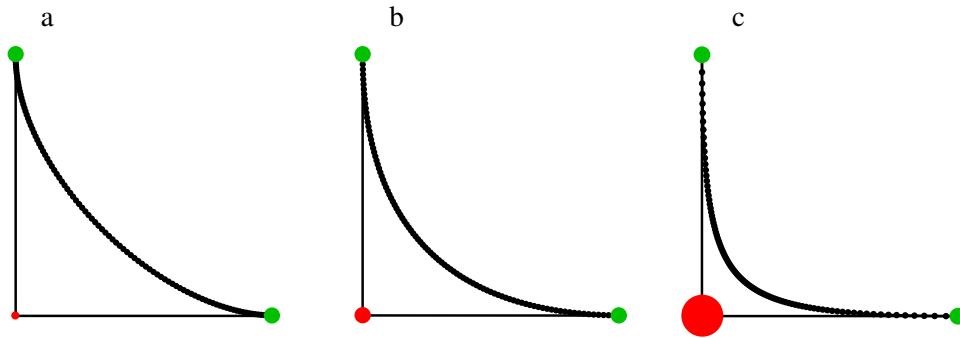


Figure 8: Examples of rational quadratic Bézier curves generated using the de Casteljau algorithm operating in three dimensions. The endpoints of the control polygon (green) have weight 1. The internal control point (red) has weight: a) 0.5, b) 1, and c) 2.5. The weights of the control points are indicated by their radius. Example based on [27, pp. 124–125].

A 2D point  $P'(v_x, v_y)$  can thus be represented by different 3D points of the form  $P(wv_x, wv_y, w)$ . The parameter  $w$  is called the weight of point  $P'$ . Weights of control points do not affect the shape of the control polygon, but bias the resulting curve towards the points with a higher weight. This provides an additional means of controlling the shape of rational curves, beyond the manipulation of the control polygon. Examples of quadratic rational Bézier curves generated with the de Casteljau algorithm (L-system 13) using the same control polygon, but different point weights, are presented in Figure 8.

## 6 Conclusions

We have shown that several fundamental algorithms for the geometric modeling of curves can be succinctly expressed using L-systems with affine geometry interpretation. Examples include the Lane-Riesenfeld algorithm for generating uniform B-splines of arbitrary degree, the de Casteljau algorithm for generating Bezier curves, and their extensions to rational curves. Our results generalize the previously reported geometric-modeling applications of L-systems, which were limited to subdivision curves [26]. Both the previous and current results demonstrate that L-system specifications closely match verbal descriptions of the modeling algorithms, and capture the information flow underlying their operation. This narrows the semantic gap between intuition and mathematical formalism, making L-systems useful as a notation for presenting the algorithms. These advantages of L-systems are similar to those observed in plant modeling, and stem from the same features. The key feature is the ease of expressing geometric algorithms that operate locally on structures with a varying number of components. This ease is achieved through index-free notation that emphasizes the topological relations between components (c.f. Section 1). Whether the use of L-systems could also lead to a notational and conceptual simplification of the theorem proofs considered in geometric modeling remains an interesting open question.

## 7 Acknowledgments

All examples presented in this paper were implemented using the L-system-based modeling program `lpfg` developed by Radoslaw Karwowski [11]. The correspondence between the mathematical notation used in this paper and the format of L-system specifications in `lpfg` is discussed in [26]. We thank

Thomas Burt for extending `lpg` with context-sensitive interpretation rules, and Adam Runions for insightful comments on the draft manuscript. The support of this research by the Natural Sciences and Engineering Research Council of Canada Discovery Grants to P.P. and F.S. is gratefully acknowledged.

## References

- [1] H. Abelson & A. A. diSessa (1982): *Turtle geometry*. M.I.T. Press, Cambridge.
- [2] G. Chaikin (1974): *An algorithm of high speed curve generation*. *Computer Graphics and Image Processing* 3, pp. 346–349.
- [3] J. Dassow (1986): *On compound Lindenmayer systems*. In: G. Rozenberg & A. Salomaa, editors: *The book of L*, Springer-Verlag, Berlin, pp. 75–85.
- [4] T. DeRose. *Three-dimensional computer graphics. A coordinate-free approach*. Manuscript, University of Washington, 1992.
- [5] T. DeRose (1989): *A coordinate-free approach to geometric programming*. In: W. Strasser & H.-P. Seidel, editors: *Theory and practice of geometric modeling*, Springer-Verlag, Berlin, pp. 291–305.
- [6] G. Farin & D. Hansford (2000): *The essentials of CAGD*. A. K. Peters, Natick, MA.
- [7] H. Freeman (1961): *On encoding arbitrary geometric configurations*. *IRE Transactions on Electronic Computers* 10, pp. 260–268.
- [8] R. Goldman (2002): *On the algebraic and geometric foundations of computer graphics*. *ACM Transactions on Graphics* 21(1), pp. 52–86.
- [9] R. Goldman (2003): *Pyramid algorithms. A dynamic programming approach to curves and surfaces for geometric modeling*. Morgan Kaufmann, San Francisco.
- [10] G. T. Herman & G. Rozenberg (1975): *Developmental systems and languages*. North-Holland, Amsterdam.
- [11] R. Karwowski & P. Prusinkiewicz (2003): *Design and implementation of the L+C modeling language*. *Electronic Notes in Theoretical Computer Science* 86(2), pp. 134–152.
- [12] W. Kurth (1994): *Growth grammar interpreter GROGRA 2.4: A software tool for the 3-dimensional interpretation of stochastic, sensitive growth grammars in the context of plant modeling. Introduction and reference manual*. Forschungszentrum Waldökosysteme der Universität Göttingen, Göttingen.
- [13] J. M. Lane & R. F. Riesenfeld (1980): *A theoretical development for the computer generation and display of piecewise polynomial surfaces*. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2(1), pp. 35–46.
- [14] A. Lindenmayer (1968): *Mathematical models for cellular interaction in development, Parts I and II*. *Journal of Theoretical Biology* 18, pp. 280–315.
- [15] A. Lindenmayer (1971): *Developmental systems without cellular interaction, their languages and grammars*. *Journal of Theoretical Biology* 30, pp. 455–484.
- [16] A. Lindenmayer (1974): *Adding continuous components to L-systems*. In: G. Rozenberg & A. Salomaa, editors: *L Systems*, Lecture Notes in Computer Science 15, Springer-Verlag, Berlin, pp. 53–68.
- [17] J. Maillot & J. Stam (2001): *A unified subdivision scheme for polygonal modeling*. *Computer Graphics Forum (Proceedings of Eurographics 2001)* 20(3), pp. 471–479.
- [18] R. Palmer & V. Shapiro (1993): *Chain models of physical behavior for engineering analysis and design*. *Research in Engineering Design* 5, pp. 161–184.
- [19] P. Prusinkiewicz (1986): *Graphical applications of L-systems*. In: *Proceedings of Graphics Interface '86 — Vision Interface '86*, pp. 247–253.
- [20] P. Prusinkiewicz (2002): *Geometric modeling without coordinates and indices*. In: *Proceedings of Shape Modeling International*, pp. 3–4. Extended abstract.

- [21] P. Prusinkiewicz (2009): *Developmental computing*. In: C. Calude, J. F. Costa, N. Dershiwitz, E. Freire & G. Rozenberg, editors: *Unconventional Computation. 8th International Conference, UC 2009*, Lecture Notes in Computer Science 5715, Springer, Berlin, pp. 16–23. Extended abstract.
- [22] P. Prusinkiewicz & J. Hanan (1992): *L-systems: From formalism to programming languages*. In: G. Rozenberg & A. Salomaa, editors: *Lindenmayer systems: Impacts on theoretical computer science, computer graphics, and developmental biology*, Springer-Verlag, Berlin, pp. 193–211.
- [23] P. Prusinkiewicz, R. Karwowski & B. Lane (2007): *The L+C plant-modeling language*. In: J. Vos, editor: *Functional-structural modeling in crop production*, Springer, Dordrecht, pp. 27–42.
- [24] P. Prusinkiewicz & A. Lindenmayer (1990): *The algorithmic beauty of plants*. Springer-Verlag, New York. With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer.
- [25] P. Prusinkiewicz, A. Lindenmayer & F. D. Fracchia (1991): *Synthesis of space-filling curves on the square grid*. In: H.-O. Peitgen, J. M. Henriques & L. F. Penedo, editors: *Fractals in the fundamental and applied sciences*, North-Holland, Amsterdam, pp. 341–366.
- [26] P. Prusinkiewicz, F. Samavati, C. Smith & R. Karwowski (2003): *L-system description of subdivision curves*. *International Journal of Shape Modeling* 9(1), pp. 41–59.
- [27] A. Rockwood & P. Chambers (1996): *Interactive curves and surfaces: a multimedia tutorial on CAGD*. Morgan Kaufmann, San Francisco, CA.
- [28] G. Rozenberg (1973): *TOL systems and languages*. *Information and Control* 23, pp. 357–381.
- [29] G. Rozenberg & A. Salomaa (1980): *The mathematical theory of L systems*. Academic Press, New York.
- [30] M. Shirmohammadi (2008): *Geometric modeling with L-systems*. Master’s thesis, University of Calgary.
- [31] R. Siromoney & K. G. Subramanian (1983): *Space-filling curves and infinite graphs*. In: H. Ehrig, M. Nagl & G. Rozenberg, editors: *Graph grammars and their application to computer science; Second International Workshop*, Lecture Notes in Computer Science 153, Springer-Verlag, Berlin, pp. 380–391.
- [32] J. Stam (2001): *On subdivision schemes generalizing uniform B-spline surfaces of arbitrary degree*. *Computer Aided Geometric Design* 18, pp. 383–396.
- [33] A. L. Szilard & R. E. Quinton (1979): *An interpretation for DOL systems by computer graphics*. *The Science Terrapin* 4, pp. 8–13.
- [34] T. Yokomori (1986): *Graph-controlled systems — An extension of OL systems*. In: G. Rozenberg & A. Salomaa, editors: *The book of L*, Springer-Verlag, Berlin, pp. 461–471.