L-system description of subdivision curves*

Przemyslaw Prusinkiewicz, Faramarz Samavati
Colin Smith and Radoslaw Karwowski

Department of Computer Science
The University of Calgary

pwp|samavati|smithco|radekk@cpsc.ucalgary.ca

Abstract

In recent years, subdivision has emerged as a major geometric modeling technique. Algorithms for generating subdivision curves are often specified in terms of iterated matrix multiplication. Each multiplication maps a globally indexed sequence of points that represents a coarser approximation of the curve onto a longer sequence that represents a finer approximation. Unfortunately, an infinite set of matrices is needed to specify these mappings for sequences of points of arbitrary length. Thus, matrix algebra is not well attuned to the dynamic nature of subdivision. In addition, matrix notation and the use of indices obscure the local and stationary character of typical subdivision rules.

We introduce parametric context-sensitive L-systems with affine geometry interpretation as an alternative technique for specifying and generating subdivision curves. This technique is illustrated using Chaikin, cubic B-spline, and Dyn-Levin-Gregory (4-point) subdivision schemes as examples. L-systems formalize subdivision algorithms in an intuitive, concise, index-free manner, reflecting the parallel and local character of these algorithms. Furthermore, L-system specification of subdivision algorithms directly leads to their computer implementation.

1 Introduction

The definition and generation of smooth curves and surfaces specified by a small set of control points is a fundamental problem of geometric modeling. One class of solutions is based on the concept of subdivision: an iterative replacement of coarser representations of a curve or surface by finer representations. The first subdivision algorithm for curves was described almost thirty years ago [5], and algorithms for surfaces soon followed [4, 11]. Nevertheless, subdivision was not recognized as a practical modeling technique until the late 1990's, when is was successfully applied to character animation [10]. This

*To appear in the *International Journal of Shape Modeling*. Used by permission from World Scientific.

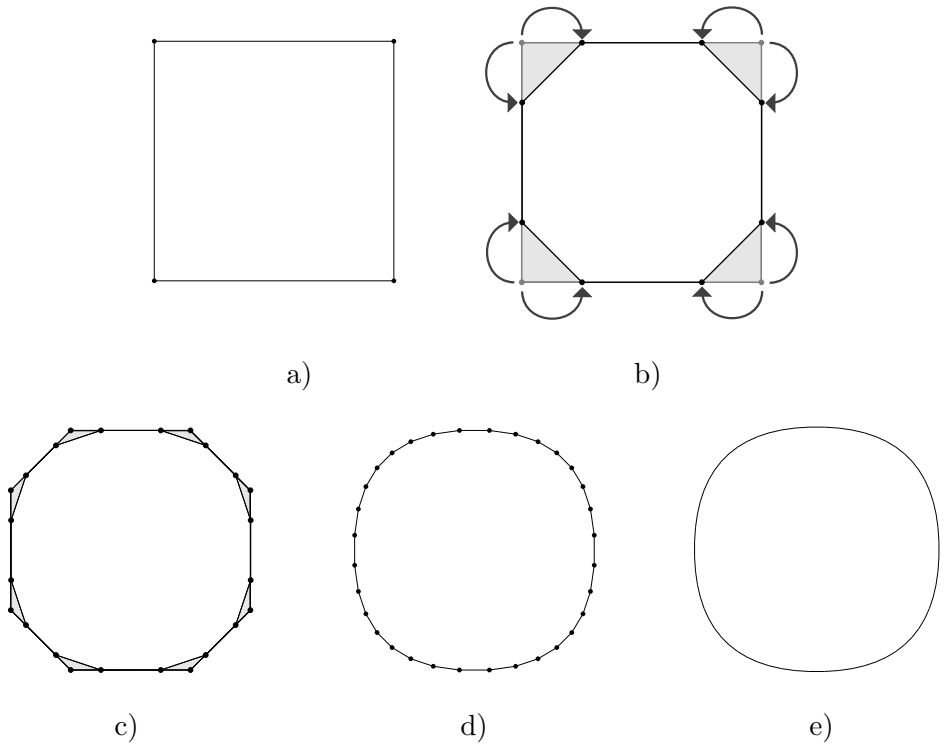


Figure 1: Curve generation using Chaikin's algorithm. a) A sample user-specified control polygon. b) The replacement of old vertices by pairs of new vertices in the first step of the algorithm. Shaded areas represent the cut-off corners. c–e) Illustration of the subsequent three subdivision steps.

development coincided with the explosion of research interest in subdivision curves and surfaces, which continues until now.

One appealing aspect of subdivision is that, at the intuitive level, it is easy to describe and understand. Unfortunately, this simplicity is not reflected in the index- and matrix-based notation often used to formalize subdivision algorithms. In this paper we propose context-sensitive parametric L-systems [22, 30] as an alternative formalism, rooted in formal language theory. L-systems make it possible to effectively specify growing sequences of symbols (words) without the use of indices. We extend this capability to polygonal approximations of subdivision curves. The L-system notation captures the local nature of subdivision algorithms in a formal yet intuitive manner, and leads directly to a computer implementation of these algorithms. This, in turn, is useful in practical applications, such as experimentation with different subdivision schemes, and expository presentation of subdivision algorithms.

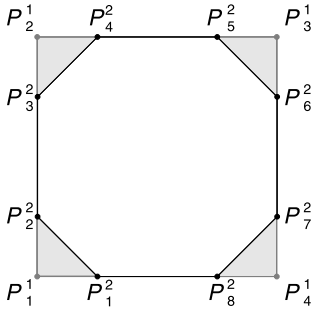


Figure 2: Example of the labeling of points in the first step of Chaikin's algorithm

2 Background

Let us consider the original Chaikin algorithm [5] to review the main concept of subdivision and its usual formalization. For simplicity, we will apply it to generate a closed curve, thus avoiding special-case rules that would be needed near the endpoints of an open curve. The initial approximation of the curve under construction is specified by a (circular) list of user-defined control points. Figure 1a shows a sample arrangement of these points, connected to form a polygon. The next approximation is obtained by cutting the corners of this polygon (Figure 1b). Each "old" vertex is replaced with a pair of "new" vertices, where each new vertex is situated one quarter of the distance from its parent point to one of its neighbors (Figure 1b). The subsequent approximations of the curve are obtained by iterating the same corner-cutting scheme (Figure 1c–e).

In the standard formalization of the subdivision process, points are globally enumerated and assigned unique labels. A possible labeling scheme is shown in Figure 2. The superscript k indicates the iteration step at which point P_i^k has been created. The subscript i is the ordering number of this point within the sequence of points created in the same step.

The positions of new points are expressed as affine combinations of the positions of old points. An affine combination of n points $P_1, P_2, \ldots, P_n$ is an expression of the form

$$\alpha_1 P_1 + \alpha_2 P_2 + \cdots + \alpha_n P_n, \tag{1}$$

where the scalar coefficients α_i add up to 1:

$$\alpha_1 + \alpha_2 + \cdots + \alpha_n = 1. \tag{2}$$

The meaning of the affine combination (1) is derived from its transformation to the form

$$P = P_1 + \alpha_2(P_2 - P_1) + \cdots + \alpha_n(P_n - P_1), \tag{3}$$

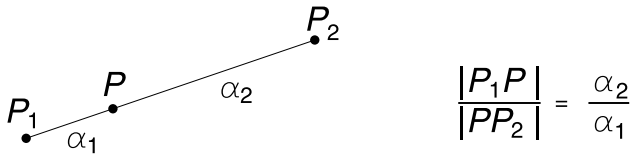


Figure 3: Definition of point P as an affine combination of points P_1 and P_2.

which is a well-defined expression of vector algebra. Specifically, for two points we obtain:

$$P = \alpha_1 P_1 + \alpha_2 P_2 = P_1 + \alpha_2(P_2 - P_1) = P_2 + \alpha_1(P_1 - P_2). \tag{4}$$

Thus, point P divides line $\overline{P_1P_2}$ in the ratio of $\alpha_2 : \alpha_1$ (Figure 3). Affine geometry and its applications to computer graphics have been described in detail by DeRose [8, 9]; for further insights see [20, 18].

Returning to Figure 2, the new point positions are related to the old point positions by the affine combinations:

$$P_1^2 = \frac{3}{4}P_1^1 + \frac{1}{4}P_4^1, \tag{5}$$

$$P_2^2 = \frac{3}{4}P_1^1 + \frac{1}{4}P_2^1, \tag{6}$$

and so on for the remaining points. In matrix notation,

$$\begin{bmatrix} P_1^2 \\ P_2^2 \\ P_3^2 \\ P_4^2 \\ P_5^2 \\ P_6^2 \\ P_7^2 \\ P_8^2 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & 0 & 0 & \frac{1}{4} \\ \frac{3}{4} & \frac{1}{4} & 0 & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & \frac{3}{4} & 0 \\ 0 & 0 & \frac{3}{4} & \frac{1}{4} \\ 0 & 0 & \frac{1}{4} & \frac{3}{4} \\ \frac{1}{4} & 0 & 0 & \frac{3}{4} \end{bmatrix} \begin{bmatrix} P_1^1 \\ P_2^1 \\ P_3^1 \\ P_4^1 \end{bmatrix}. \tag{7}$$

This equation is generalized to arbitrary control polygons and arbitrary derivation steps by writing:

$$\begin{bmatrix} P_1^{k+1} \\ P_2^{k+1} \\ P_3^{k+1} \\ P_4^{k+1} \\ P_5^{k+1} \\ P_6^{k+1} \\ \vdots \\ P_{2n-2}^{k+1} \\ P_{2n-1}^{k+1} \\ P_{2n}^{k+1} \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & 0 & 0 & 0 & \cdots & 0 & \frac{1}{4} \\ \frac{3}{4} & \frac{1}{4} & 0 & 0 & \cdots & 0 & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{4} & \frac{3}{4} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \frac{3}{4} & \frac{1}{4} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{3}{4} & \frac{1}{4} \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{4} & \frac{3}{4} \\ \frac{1}{4} & 0 & 0 & 0 & \cdots & 0 & \frac{3}{4} \end{bmatrix} \cdot \begin{bmatrix} P_1^k \\ P_2^k \\ P_3^k \\ P_4^k \\ \vdots \\ P_{n-1}^k \\ P_n^k \end{bmatrix} \tag{8}$$

Subdivision schemes other than Chaikin's can be specified in a similar way, using different subdivision matrices [2, 33, 36, 37].

Related to the use of matrices is the use of indices to identify and order the points. Unfortunately, the index notation is not well attuned to the needs of subdivision. Due to the local character of subdivision, the creation of a pair of new points is based on the information about their parent old point and its neighbors. Indexing makes it possible to access this information only in a circular way, by first globally assigning consecutive numbers to all points, then referring to the neighbors of point P_i^k using index arithmetic: $i-1$ and $i+1$. This is more complicated than the verbal description, in which we would use terms such as "previous" and "next" (or "left" and "right") to refer to the neighbors of a given point. At the same time, the index notation is too powerful: by providing a unique label to each point it makes it possible to access points at random, in violation of the algorithm's locality. This is true in both the spatial and temporal domains: in the latter case, the use of indices makes it potentially possible to refer to points from arbitrary iteration steps, whereas only the information from the previous step is available and needed.

3 From stencils and masks to productions

One alternative to the index-based notation is the representation of subdivision rules using *stencils*. Sabin [32] defines them as follows:

> **Stencil.** The weights due to various old vertices in computing a given new one. Also the pattern of relative positions of the old vertices around the new one.

Stencils (and the related notion of *masks*) are usually represented as graphs that depict short subsequences of old and new points. These points are connected by arrows labeled

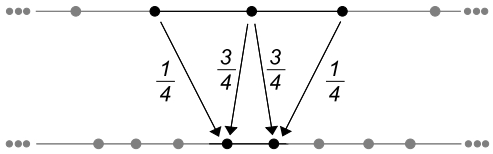


Figure 4: A stencil for Chaikin subdivision algorithm

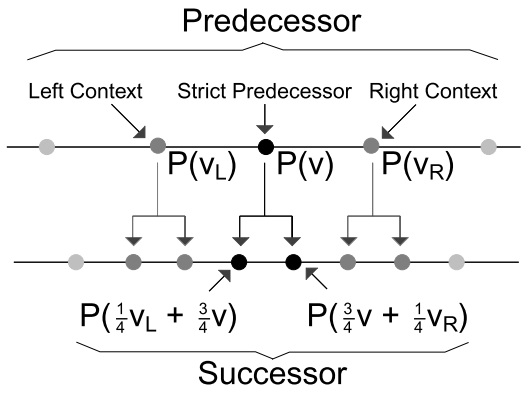


Figure 5: Chaikin's subdivision as a production

by coefficients α in the affine geometry combinations (Equation 1) that take the old points to the new ones. For instance, a stencil for the Chaikin subdivision algorithm is shown in Figure 4. For other examples see [14, 38].

Stencils provide an intuitive, index-free representation of subdivision rules. Unfortunately, this is not a precise representation. For example, Figure 4 shows that three old points are involved in creating a pair of new points. It does not explicitly specify, however, whether the same old points may also be involved in the production of other new points. Looking at the same problem from a different perspective, it is not clear to what extent the stencils may overlap when applied to various subsequences of old points.

We address this imprecision by recasting the notion of a stencil into the framework of formal language theory (Figure 5). A sequence of points is viewed as a word over some alphabet. A circular sequence of points approximating a closed curve is represented by a circular word [31, 35]. The stencil is a grammar production, with the predecessor representing a finite subsequence of old points, and the successor representing a subsequence of new points. The predecessor is partitioned into the strict predecessor, left context, and right context. The strict predecessor represents the old point that is rewritten or "consumed" by the production application, that is, cannot be used anymore. In Chaikin's construction, it is the vertex of a corner being cut off. The

context consists of the neighbors of the strict predecessor that provide the additional information needed to specify the successor points. When rewriting a sequence of old points, production predecessors may overlap, as long as no point is used more than once as a strict predecessor.

The Chaikin construction requires that each old point be replaced by two new ones in every iteration of the algorithm. This corresponds to the notion of parallel rewriting as defined for L-systems [22, 23], as opposed to the sequential rewriting defined for Chomsky grammars [7].

4 L-systems

L-systems were originally introduced as a rewriting mechanism acting on words over a finite alphabet [22, 23]. Soon afterward, however, they were extended to strings of symbols with numerical attributes [1, 24]. This concept was formalized as parametrized [6] and parametric [19, 30] L-systems. Here we use an extension of the latter formalism. For its more detailed presentation see [26, 30].

Parametric L-systems operate on strings of modules. A module is a letter from a finite alphabet V with optional numerical parameters. For example, the string

$$A(1.5)B(2.0, 3.0)A(4.5) \tag{9}$$

is a parametric word over the alphabet $V = \{A, B\}$.

Starting with an explicitly defined initial word, or axiom, an L-system generates a developmental sequence of words using a finite set of productions that operate on limited-length subwords. The actual parameters in each word correspond to the formal parameters in the productions. Arithmetic expressions in the successor of a production determine new parameter values. In the case of context-sensitive productions, the left and right contexts are separated from the strict predecessor by the metasymbols (*i.e.*, symbols that do not represent modules) $<$ and $>$, respectively.

A developmental sequence of words results from a sequence of derivation steps. In each step, productions are applied in parallel to all modules of the predecessor word, so that each module is the strict predecessor of some production. For example, by applying the production set

$$A(x) \rightarrow A(2x + 1) \tag{10}$$

$$A(w) < B(x, y) > A(x) \rightarrow A(w + x)A(y + x) \tag{11}$$

to the parametric word (9), we obtain after one derivation step:

$$A(4.0)A(3.5)A(7.5)A(10). \tag{12}$$

Recent extensions of parametric L-systems [13, 21] make it possible to use not only numbers, but also compound data structures as parameters. We use this feature

to represent points as vectors of coordinates, and we overload standard arithmetic operators to specify affine combinations of points. With this convention, the L-system production that specifies Chaikin's subdivision becomes:

$$P(\mathbf{v}_l) < P(\mathbf{v}) > P(\mathbf{v}_r) \rightarrow P(\frac{1}{4} \cdot \mathbf{v}_l + \frac{3}{4} \cdot \mathbf{v})P(\frac{3}{4} \cdot \mathbf{v} + \frac{1}{4} \cdot \mathbf{v}_r) \tag{13}$$

This mathematical notation is closely reflected in the programming language `L+C`, which combines features of L-systems and the `C++` programming language [21]. `L+C` programs constitute an input to the graphical modeling program `lpfg` [21], which we have used to implement the algorithms presented in this paper. For example, the following `L+C` program generates the picture (curve with dots) shown in Figure 1d:

```
1   #include <lpfgall.h>
2   V2f v1(0, 0), v2(0, 1), v3(1, 1), v4(1, 0);
3   module P(V2f);
4
5   ring L-system: 1;
6   derivation length: 3;
7
8   Axiom: P(v1) P(v2) P(v3) P(v4) ;
9   P(vl) < P(v) > P(vr) :
10        { produce P(0.25*vl+0.75*v) P(0.75*v+0.25*vr); }
11
12  interpretation:
13  P(v) :  { produce LineTo2f(v) Circle(0.01) ; }
```

Line 1 is a reference to the file `lpfgall.h` that contains predefined constants and structure declarations. Specifically, our program makes use of two-dimensional vectors `V2f`. Line 2 defines four points that will be used in line 8 as the vertices of the control polygon. Line 3 declares module type `P` as being associated with one parameter — a vector of type `V2f`. Line 5 specifies that the L-system will operate on circular words, and line 6 specifies the required derivation length. Lines 9 and 10 are the essence of this program and contain the production responsible for the Chaikin subdivision. Finally, Line 13, preceded by the keyword in line 12, defines the homomorphism that will be applied at the end of the derivation (*c.f.* [28]). According to it, each point is represented as a small circle, connected by a line to its predecessor.

For compactness, in the following sections we will mainly use the mathematical notation exemplified by Equation 13, rather than complete program listings.

5 Inferring L-systems from subdivision matrices

Since subdivision curves are often defined using matrix notation [2, 33, 36, 37], the inference of L-systems from the subdivision matrices is an important practical problem. Unfortunately, it cannot entirely be resolved by algorithmic means, because dots in the general subdivision matrices, *e.g.* (8), require an interpretation. Furthermore, as we are going to see, many equivalent L-systems can be inferred from the same matrix, thus the inference process involves an element of decision.

As described in Section 2, the subdivision matrix globally maps an old sequence of points onto a new sequence. In contrast, an L-system production replaces an individual old point by a subsequence of new points. We must therefore partition the sequence of new points into subsequences, and establish a one-to-one mapping between the old points and these subsequences. For example, a mapping for the Chaikin subdivision (Equation 8) may take point P_i^k to points P_{2i-1}^{k+1} and P_{2i}^{k+1}, where $i = 1, 2, \ldots, n$. An instance of this mapping for $i = 3$ is illustrated below:

$$\begin{bmatrix} P_1^{k+1} \\ P_2^{k+1} \\ P_3^{k+1} \\ P_4^{k+1} \\ P_5^{k+1} \\ P_6^{k+1} \\ \vdots \\ P_{2n-2}^{k+1} \\ P_{2n-1}^{k+1} \\ P_{2n}^{k+1} \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & 0 & 0 & 0 & \cdots & 0 & \frac{1}{4} \\ \frac{3}{4} & \frac{1}{4} & 0 & 0 & \cdots & 0 & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{4} & \frac{3}{4} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \frac{3}{4} & \frac{1}{4} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{3}{4} & \frac{1}{4} \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{4} & \frac{3}{4} \\ \frac{1}{4} & 0 & 0 & 0 & \cdots & 0 & \frac{3}{4} \end{bmatrix} \begin{bmatrix} P_1^k \\ P_2^k \\ P_3^k \\ P_4^k \\ \vdots \\ P_{n-1}^k \\ P_n^k \end{bmatrix} \tag{14}$$

The encircled points in the column matrices show that old point P_3^k will be replaced by new points P_5^{k+1} and P_6^{k+1}. In the subdivision matrix, the same replacement is indicated by encircling column 3, which represents the contribution of point P_3^k to the matrix multiplication, and rows 5 and 6, which yield points P_5^{k+1} and P_6^{k+1} of the result. The shaded area includes non-zero elements of these rows, and thus identifies the production predecessor and the coefficients of the affine combinations that will yield the successor points. The position of the encircled column with respect to this area partitions the predecessor into left context, strict predecessor, and right context. The replacement of point P_3^k by points P_5^{k+1} and P_6^{k+1} can therefore be written as a production,

$$P_{i-1}^k(\mathbf{v}_l) < P_i^k(\mathbf{v}) > P_{i+1}^k(\mathbf{v}_r) \rightarrow P_{2i-1}^{k+1}(\frac{1}{4} \cdot \mathbf{v}_l + \frac{3}{4} \cdot \mathbf{v}) P_{2i}^{k+1}(\frac{3}{4} \cdot \mathbf{v} + \frac{1}{4} \cdot \mathbf{v}_r), \tag{15}$$

where $i = 3$. The regular form of the subdivision matrix in Equation 14 suggests that production (15) applies for any values $i, k = 1, 2, \ldots$. This observation leads to the general L-system production for Chaikin subdivision in Equation 13.

The decision to replace point P_i^k with the points P_{2i-1}^{k+1} and P_{2i}^{k+1} was an arbitrary one. In general, there is an equivalent one-production L-system that generates the same Chaikin curve (up to a cyclical permutation of points) by taking point P_i^k to a pair of consecutive points $P_{j-1}^{k+1}\ P_j^{k+1}$ $(i = 1, 2, \ldots, n;\ j \equiv 2i + d \bmod 2n)$ for any integer d. The mapping performed by production 13 and illustrated by Equation 14 corresponds to $d = 0$. Two other mappings, corresponding to $d = -1$ and $d = 1$, are indicated below:

$$\begin{bmatrix} \frac{3}{4} & 0 & 0 & 0 & \cdots & 0 & \frac{1}{4} \\ \frac{3}{4} & \frac{1}{4} & 0 & 0 & \cdots & 0 & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{4} & \frac{3}{4} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \frac{3}{4} & \frac{1}{4} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{3}{4} & \frac{1}{4} \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{4} & \frac{3}{4} \\ \frac{1}{4} & 0 & 0 & 0 & \cdots & 0 & \frac{3}{4} \end{bmatrix}, \qquad \begin{bmatrix} \frac{3}{4} & 0 & 0 & 0 & \cdots & 0 & \frac{1}{4} \\ \frac{3}{4} & \frac{1}{4} & 0 & 0 & \cdots & 0 & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{4} & \frac{3}{4} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \frac{3}{4} & \frac{1}{4} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{3}{4} & \frac{1}{4} \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{4} & \frac{3}{4} \\ \frac{1}{4} & 0 & 0 & 0 & \cdots & 0 & \frac{3}{4} \end{bmatrix}. \tag{16}$$

The resulting L-system productions are, respectively:

$$P(\mathbf{v}_l) < P(\mathbf{v}) \rightarrow P(\tfrac{3}{4} \cdot \mathbf{v}_l + \tfrac{1}{4} \cdot \mathbf{v}) P(\tfrac{1}{4} \cdot \mathbf{v}_l + \tfrac{3}{4} \cdot \mathbf{v}), \tag{17}$$

$$P(\mathbf{v}) > P(\mathbf{v}_r) \rightarrow P(\tfrac{3}{4} \cdot \mathbf{v} + \tfrac{1}{4} \cdot \mathbf{v}_r) P(\tfrac{1}{4} \cdot \mathbf{v} + \tfrac{3}{4} \cdot \mathbf{v}_r). \tag{18}$$

Productions 17 and 18 lack the symmetry of production 13, but are shorter and in this sense simpler than production 13. Other values of constant d yield productions that are also asymmetric, but relatively longer. For example, for $d = 3$ we obtain:

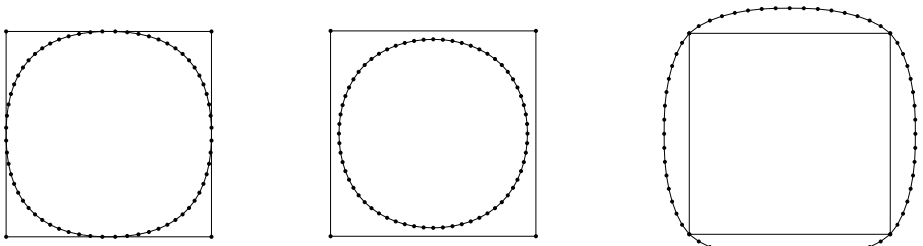
$$\begin{bmatrix}
\frac{3}{4} & 0 & 0 & 0 & \cdots & 0 & \frac{1}{4} \\
\frac{3}{4} & \frac{1}{4} & 0 & 0 & \cdots & 0 & 0 \\
\frac{1}{4} & \frac{3}{4} & 0 & 0 & \cdots & 0 & 0 \\
0 & \frac{3}{4} & \frac{1}{4} & 0 & \cdots & 0 & 0 \\
0 & \frac{1}{4} & \frac{3}{4} & 0 & \cdots & 0 & 0 \\
0 & 0 & \frac{3}{4} & \frac{1}{4} & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \cdots & \frac{3}{4} & \frac{1}{4} \\
0 & 0 & 0 & 0 & \cdots & \frac{1}{4} & \frac{3}{4} \\
\frac{1}{4} & 0 & 0 & 0 & \cdots & 0 & \frac{3}{4}
\end{bmatrix}, \tag{19}$$

$$P(\mathbf{v}) > P(\mathbf{v}_r)P(\mathbf{v}_{rr})P(\mathbf{v}_{rrr}) \rightarrow P(\frac{3}{4}\cdot\mathbf{v}_{rr} + \frac{1}{4}\cdot\mathbf{v}_{rrr})P(\frac{1}{4}\cdot\mathbf{v}_{rr} + \frac{3}{4}\cdot\mathbf{v}_{rrr}). \tag{20}$$

This production, along with other productions obtained for $|d| > 1$, appears to be of limited interest, because new points are increasingly distant from the old points they replace, contrary to the intuition of the Chaikin algorithm.

L-system productions for other subdivision algorithms can be inferred in a similar way. For example, below we present two views of the subdivision matrix for the cubic B-spline subdivision (c.f. [15, 33]):

$$\begin{bmatrix}
\frac{1}{2} & 0 & 0 & 0 & \cdots & 0 & 0 & \frac{1}{2} \\
\frac{3}{4} & \frac{1}{8} & 0 & 0 & \cdots & 0 & 0 & \frac{1}{8} \\
\frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots & 0 & 0 & 0 \\
\frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 & \cdots & 0 & 0 & 0 \\
0 & \frac{1}{2} & \frac{1}{2} & 0 & \cdots & 0 & 0 & 0 \\
0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & \cdots & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\
0 & 0 & \frac{1}{2} & \frac{1}{2} & \cdots & 0 & \frac{1}{2} & \frac{1}{2} \\
\frac{1}{8} & 0 & \frac{1}{2} & \frac{1}{2} & \cdots & 0 & \frac{1}{8} & \frac{3}{4}
\end{bmatrix}, \qquad
\begin{bmatrix}
\frac{1}{2} & 0 & 0 & 0 & \cdots & 0 & 0 & \frac{1}{2} \\
\frac{3}{4} & \frac{1}{8} & 0 & 0 & \cdots & 0 & 0 & \frac{1}{8} \\
\frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots & 0 & 0 & 0 \\
\frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 & \cdots & 0 & 0 & 0 \\
0 & \frac{1}{2} & \frac{1}{2} & 0 & \cdots & 0 & 0 & 0 \\
0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & \cdots & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\
0 & 0 & \frac{1}{2} & \frac{1}{2} & \cdots & 0 & \frac{1}{2} & \frac{1}{2} \\
\frac{1}{8} & 0 & \frac{1}{2} & \frac{1}{2} & \cdots & 0 & \frac{1}{8} & \frac{3}{4}
\end{bmatrix}. \tag{21}$$



a) b) c)

Figure 6: A comparison of curves generated with (a) Chaikin, (b) cubic B-spline, and (c) Dyn-Levin-Gregory subdivision algorithms, using the same control polygon.

The corresponding L-system productions are, respectively:

$$P(\mathbf{v}_l) < P(\mathbf{v}) > P(\mathbf{v}_r) \rightarrow P(\tfrac{1}{8}\cdot\mathbf{v}_l + \tfrac{3}{4}\cdot\mathbf{v} + \tfrac{1}{8}\cdot\mathbf{v}_r)P(\tfrac{1}{2}\cdot\mathbf{v} + \tfrac{1}{2}\cdot\mathbf{v}_r), \tag{22}$$

$$P(\mathbf{v}_l) < P(\mathbf{v}) > P(\mathbf{v}_r) \rightarrow P(\tfrac{1}{2}\cdot\mathbf{v}_l + \tfrac{1}{2}\cdot\mathbf{v})P(\tfrac{1}{8}\cdot\mathbf{v}_l + \tfrac{3}{4}\cdot\mathbf{v} + \tfrac{1}{8}\cdot\mathbf{v}_r). \tag{23}$$

Figure 6b shows a sample curve generated using either of these productions. Similar to Chaikin subdivision (Figure 6a), the cubic B-spline subdivision yields a curve that approximates the control polygon. In contrast, Dyn-Levin-Gregory 4-point subdivision [12] (see also [33]) generates an interpolating curve (Figure 6c). Its subdivision matrix, complemented with one of the possible mappings of an old point to new points, is given below:

$$\begin{bmatrix}
\frac{9}{16} & -\frac{1}{16} & 0 & 0 & 0 & \cdots & 0 & -\frac{1}{16} & \frac{9}{16} \\
1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
\frac{9}{16} & \frac{9}{16} & -\frac{1}{16} & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
-\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} & \cdots & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\
-\frac{1}{16} & 0 & 0 & 0 & 0 & \cdots & -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1
\end{bmatrix}. \tag{24}$$

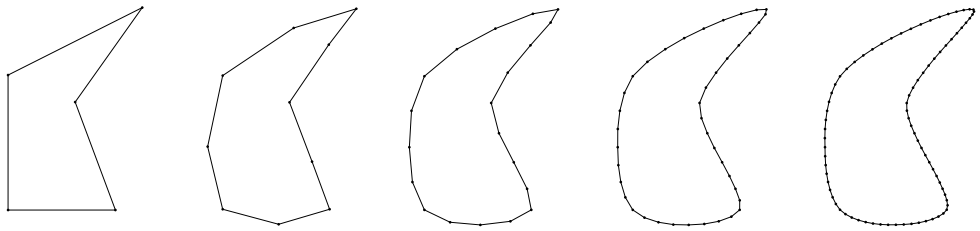


Figure 7: Five steps of a curve generation using the Dyn-Levin-Gregory algorithm

The resulting L-system production is:

$$\begin{aligned} &P(\mathbf{v}_l) < P(\mathbf{v}) > P(\mathbf{v}_r)P(\mathbf{v}_{rr}) \rightarrow \\ &\quad P(\mathbf{v})P(-\tfrac{1}{16} \cdot \mathbf{v}_l + \tfrac{9}{16} \cdot \mathbf{v} + \tfrac{9}{16} \cdot \mathbf{v}_r - \tfrac{1}{16} \cdot \mathbf{v}_{rr}). \end{aligned} \tag{25}$$

According to this production, the predecessor old point will be replaced by a copy of itself, followed by a newly inserted point. A different choice for mapping of old and new points results in an alternative production, in which the newly inserted point precedes the copy of the old point:

$$\begin{aligned} &P(\mathbf{v}_{ll})P(\mathbf{v}_l) < P(\mathbf{v}) > P(\mathbf{v}_r) \rightarrow \\ &\quad P(-\tfrac{1}{16} \cdot \mathbf{v}_{ll} + \tfrac{9}{16} \cdot \mathbf{v}_l + \tfrac{9}{16} \cdot \mathbf{v} - \tfrac{1}{16} \cdot \mathbf{v}_r)P(\mathbf{v}). \end{aligned} \tag{26}$$

The interpolating character of this subdivision scheme is further illustrated in Figure 7, which shows that points from the previous step are preserved in the next step.

Descriptions of subdivision schemes are often expressed in terms of "even" and "odd" points [38]. Odd points are newly created in the given algorithm step, whereas even points are the old points. The position of even points is preserved in the interpolating schemes, or adjusted in the approximating schemes. The L-system expression of subdivision rules makes the distinction between odd and even points unnecessary.

6 Subdividing open curves

Subdivision of open curves proceeds in a manner similar to the subdivision of closed curves, except that special subdivision rules must be applied near the curve endpoints. For example, let us consider the inference of an L-system for the Chaikin subdivision of an open curve, given the following subdivision matrix [33]:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{3}{4} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{3}{4} & \frac{1}{4} & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & \frac{3}{4} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{3}{4} & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{4} & \frac{3}{4} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{27}$$

Unlike previously considered matrices, which had n columns and $2n$ rows, this matrix has only $2n-2$ rows. Thus, it is no longer possible to substitute two new points for each old point. We address this problem assuming that the first and last old point will be replaced by single points, and the remaining old points will be replaced by pairs of new points. This leads to the following L-system productions:

$$\# < P(\mathbf{v}) \rightarrow P(\mathbf{v}) \tag{28}$$

$$\#P(\mathbf{v}_l) < P(\mathbf{v}) > P(\mathbf{v}_r) \rightarrow P(\frac{1}{2} \cdot \mathbf{v}_l + \frac{1}{2} \cdot \mathbf{v})P(\frac{3}{4} \cdot \mathbf{v} + \frac{1}{4} \cdot \mathbf{v}_r) \tag{29}$$

$$P(\mathbf{v}_{ll})P(\mathbf{v}_l) < P(\mathbf{v}) > P(\mathbf{v}_r)P(\mathbf{v}_{rr}) \rightarrow P(\frac{1}{4} \cdot \mathbf{v}_l + \frac{3}{4} \cdot \mathbf{v})P(\frac{3}{4} \cdot \mathbf{v} + \frac{1}{4} \cdot \mathbf{v}_r) \tag{30}$$

$$P(\mathbf{v}_l) < P(\mathbf{v}) > P(\mathbf{v}_r)\# \rightarrow P(\frac{1}{4} \cdot \mathbf{v}_l + \frac{3}{4} \cdot \mathbf{v})P(\frac{1}{2} \cdot \mathbf{v} + \frac{1}{2} \cdot \mathbf{v}_r) \tag{31}$$

$$P(\mathbf{v}) > \# \rightarrow P(\mathbf{v}) \tag{32}$$

$$\# \rightarrow \# \tag{33}$$

We assume that the control polygon is represented by a sequence of at least four modules $P(\mathrm{v})$, delimited by modules #. Productions 28 and 32 state that the first and the last point of the curve will be rewritten by themselves, as specified by the first and the last row of subdivision matrix 27. Production 29 is associated with the second and third point of the subdivision matrix, and production 31 is associated with the third and second last row of that matrix in a symmetric way. Production 30 captures subdivision away from the endpoints. In essence, it is the same production as production 13 for the Chaikin subdivision of closed curves. The additional context terms, $P(\mathbf{v}_{ll})$ and $P(\mathbf{v}_{rr})$, assure that production 30 will not be applied too close to the endpoints of the curve. Finally, production 33 rewrites the endmarkers by themselves.

The above L-system can be simplified using the following conventions [26, 30]:

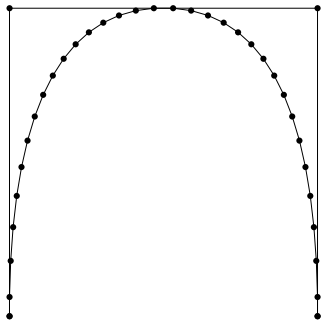


Figure 8: An open control polygon and the resulting Chaikin subdivision curve obtained using L-system productions 28 – 33 or 34 – 36.

- if no production for rewriting a particular module is explicitly listed, this module will be rewritten into itself;
- if more than one production could be used to rewrite the same module, the production that appears first in the ordered production list will be applied.

Under these conventions, the open-curve Chaikin subdivision can be defined using the following L-system productions:

$$\#P(\mathbf{v}_l) < P(\mathbf{v}) > P(\mathbf{v}_r) \rightarrow P(\frac{1}{2} \cdot \mathbf{v}_l + \frac{1}{2} \cdot \mathbf{v})P(\frac{3}{4} \cdot \mathbf{v} + \frac{1}{4} \cdot \mathbf{v}_r) \tag{34}$$

$$P(\mathbf{v}_l) < P(\mathbf{v}) > P(\mathbf{v}_r)\# \rightarrow P(\frac{1}{4} \cdot \mathbf{v}_l + \frac{3}{4} \cdot \mathbf{v})P(\frac{1}{2} \cdot \mathbf{v} + \frac{1}{2} \cdot \mathbf{v}_r) \tag{35}$$

$$P(\mathbf{v}_l) < P(\mathbf{v}) > P(\mathbf{v}_r) \rightarrow P(\frac{1}{4} \cdot \mathbf{v}_l + \frac{3}{4} \cdot \mathbf{v})P(\frac{3}{4} \cdot \mathbf{v} + \frac{1}{4} \cdot \mathbf{v}_r) \tag{36}$$

These L-systems provide a complete and compact specification of the Chaikin subdivision algorithm for open curves, and directly lead to its computer implementation (*c.f.* Section 4). An application example is given in Figure 8. The reference to the curve endpoints using context and markers is less error-prone than the use of numerical limits for index values. The same methodology can be used to specify L-systems for other subdivision schemes.

7 Reverse subdivision

Bartels and Samavati introduced the notion of reverse subdivision, in which the number of points representing a curve or surface is gradually reduced, while the resulting approximations are kept within tolerable error bounds [2, 33]. Specifically, local reverse subdivision [2] inverts the paradigm of the forward subdivision: instead of replacing

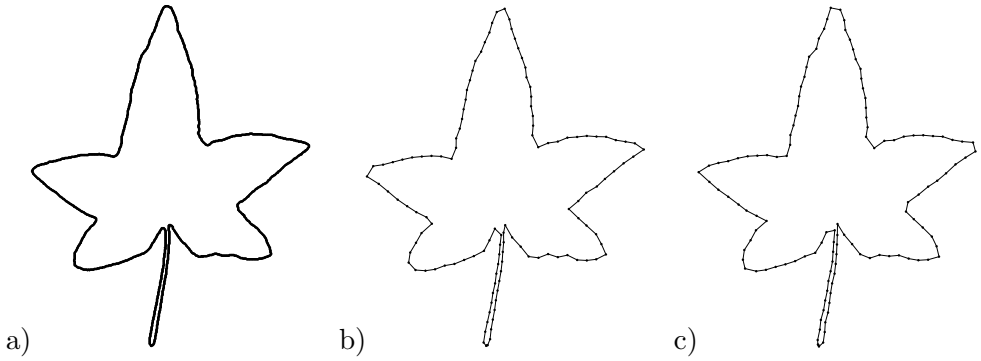


Figure 9: Reverse subdivision of a scanned ivy leaf contour. a) The input contour with 1925 points. b, c) Two approximations of the input contour obtained using production 38. Both approximations consist of 122 points, and have been obtained after four reverse subdivision steps using different circular permutations of the input string.

individual old points by subsequences of new points, it replaces subsequences of old points by individual new points. Bartels and Samavati specify this process using reverse subdivision matrices. For example, the matrix for the reverse Chaikin subdivision of a closed curve is [2]:

$$\begin{bmatrix} \frac{3}{4} & -\frac{1}{4} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & -\frac{1}{4} & \frac{3}{4} \\ -\frac{1}{4} & \frac{3}{4} & \frac{3}{4} & -\frac{1}{4} & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{4} & \frac{3}{4} & \frac{3}{4} & \cdots & -\frac{1}{4} & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & -\frac{1}{4} & \cdots & \frac{3}{4} & \frac{3}{4} & -\frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & -\frac{1}{4} & \frac{3}{4} & \frac{3}{4} & -\frac{1}{4} \end{bmatrix} \tag{37}$$

This matrix has $2n$ columns and n rows, which implies that pairs of predecessor points will be replaced by individual points. Using the grouping indicated by the encircled row, columns, and the shaded area, we obtain the following production:

$$P(\mathbf{v}_l) < P(\mathbf{v}_a)P(\mathbf{v}_b) > P(\mathbf{v}_r) \rightarrow P(-\frac{1}{4} \cdot \mathbf{v}_l + \frac{3}{4} \cdot \mathbf{v}_a + \frac{3}{4} \cdot \mathbf{v}_b - \frac{1}{4} \cdot \mathbf{v}_r) \tag{38}$$

Formally, this production is not consistent with the definition of L-systems, because its strict predecessor is not a single module. Nevertheless, an extension called pseudo-L-systems [25] makes it possible to use such productions. In a pseudo-L-system derivation step, strict predecessors are assumed to partition the predecessor string without overlaps. This is a source of nondeterminism, since different partitions may exist, leading to

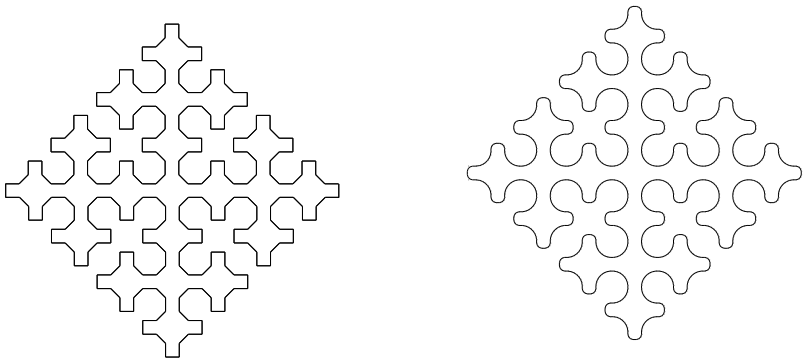


Figure 10: A Sierpinski space-filling curve (a) and its smooth version obtained using subdivision (b).

different results. For example, a circular word of length $2n$ can be partitioned into pairs $(1,2),(3,4),\ldots,(n-1,n)$ or $(2,3),(4,5),\ldots,(n,1)$. The existence of different results of a reverse subdivision step implies that the same original curve may be approximated in more than one way.

Reverse subdivision can be used, for example, to reduce the number of points approximating a measured curve. An example of such an application is shown in Figure 9.

8 Conclusions

We proposed context-sensitive parametric L-systems with affine geometry interpretation as a formal method for specifying subdivision algorithms for curves. L-systems formalize the notion of stencils and provide an intuitive yet compact and complete description of subdivision algorithms.

L-systems capture the local character of subdivision rules and the dynamic character of the subdivision process. This compatibility is closely related to the biological motivation of L-systems. They were originally proposed to describe the growth of linear structures made of locally communicating discrete elements. Subdivision can obviously be seen as an instance of such growth.

An important feature of L-system notation is that it identifies a module by its state and neighborhood. This stands in a contrast to standard mathematical notation, in which elements of a sequence are identified by indices. The index-free notation simplifies the specification and implementation of a dynamical system with dynamic structure [17]. The indices, if present, must be recalculated each time the number or configuration of components change, and thus do not provide convenient, stable identifiers of system elements. In addition, the use of indices obscures the local character of subdivision rules.

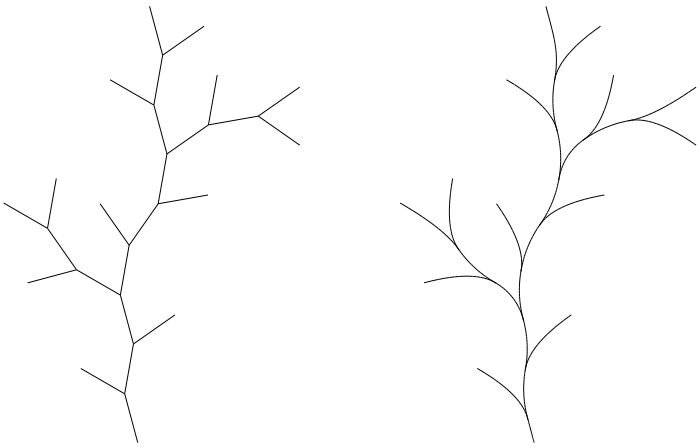


Figure 11: A branching structure (a) and the result of its smoothing (b).

We have considered the inference of L-systems, given subdivision matrices. We illustrated this inference using Chaikin, cubic B-spline, and Dyn-Levin-Gregory (4-point) subdivision schemes as examples. In addition to closed curves, discussed in more detail, we presented an example of an open curve subdivision, based on Chaikin's scheme. We have also shown that an extension of L-systems can be used to specify and implement local reverse subdivision algorithms.

We have implemented the programming language `L+C`, which makes it possible to specify L-systems as input to the modeling program, `lpfg`. This facilitates experimentation with various subdivision schemes, because not only the subdivision parameters, but also the entire subdivision algorithms, can easily be specified and modified. This makes L-systems particularly useful in research and teaching of subdivision curves.

Several problems relating L-systems to subdivision remain open for further research. For example, we observed that L-systems with affine geometry interpretation can also be used to generate fractals. This echoes the relation between subdivision curves and fractals pointed out by Warren and Weimer [37]. The possibility of integrating fractals and subdivision curves using the same L-system formalism is interesting from the theoretical perspective and may have useful applications. For instance, Figure 10 shows a finite approximation of the Sierpinski space-filling curve [34], and the result of its smoothing using Chaikin subdivision. The resulting curve is a kolam pattern, a representative of patterns that were developed as folk art in India and have attracted mathematical attention because of their self-similar structure [16, 27, 29].

Another open problem is the extension of subdivision algorithms to branching structures. An example of such a structure is shown in Figure 11a, and the result of its smoothing using Chaikin's algorithm is shown in Figure 11b. As pointed out by Bloomenthal [3], the use of curved lines increases the perception of realism in many models

of organic forms. The formalism of L-systems is useful in describing branching forms, and therefore may provide a convenient general basis for subdividing branching curves as well.

Acknowledgement

We thank Lynn Mercer for editorial help. The support of the Natural Sciences and Engineering Research Council of Canada is gratefully acknowledged.

References

[1] R. Baker and G. T. Herman. Simulation of organisms using a developmental model, parts I and II. *International Journal of Bio-Medical Computing*, 3:201–215 and 251–267, 1972.

[2] R. H. Bartels and F. F. Samavati. Reversing subdivision rule: local linear conditions and observations on inner products. *Journal of Computational and Applied Mathematics*, 119:29–67, 2000.

[3] J. Bloomenthal. *Skeletal design of natural forms*. PhD thesis, University of Calgary, January 1995.

[4] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[5] G. Chaikin. An algorithm for high speed curve generation. *Computer Graphics and Image Processing*, 3:346–349, 1974.

[6] 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.

[7] N. Chomsky. Three models for the description of language. *IRE Trans. on Information Theory*, 2(3):113–124, 1956.

[8] T. DeRose. Three-dimensional computer graphics. A coordinate-free approach. Manuscript, University of Washington, 1992.
`http://grail.cs.washington.edu/pub/`.

[9] T. DeRose. A coordinate-free approach to geomeric programming. In W. Strasser and H.-P. Seidel, editors, *Theory and practice of geometric modeling*, pages 291–305. Springer-Verlag, Berlin, 1989.

[10] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. Proceedings of SIGGRAPH 98 (Orlando, Florida, July 19–24, 1998), pages 85–94, ACM SIGGRAPH, New York, 1998.

[11] D. Doo and M. Sabin. Analysis of the behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):356–360, 1978.

[12] N. Dyn, J. Gregory, and D. Levin. A four-point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design*, 4:257–268, 1987.

[13] K. A. Erstad. L-systems, twining plants, Lisp. Master's thesis, University of Bergen, Norway, January 2002. `http://www.ii.uib.no/~knute/lsystems/`.

[14] K. Joy *et. al.* On-line geometric modeling notes. Computer Science Department, University of California, Davis.
`http://graphics.cs.ucdavis.edu/CAGDNotes`.

[15] G. Farin. *Curves and surfaces for CAGD. A practical guide. Fifth edition.* Morgan Kaufmann, San Francisco, 2002.

[16] P. Gerdes. Reconstruction and extension of lost symmetries: examples from the Tamil of South India. *Computers Math. Applic.*, 17(4–6):791–813, 1989.

[17] J.-L. Giavitto and O. Michel. MGS: A programming language for the transformation of topological collections. Research Report 61-2001, CNRS - Université d'Evry Val d'Esonne, Evry, France, 2001.

[18] R. Goldman. On the algebraic and geometric foundations of computer graphics. *ACM Transactions on Graphics*, 21(1):52–86, January 2002.

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

[20] M. Hausner. *A vector space approach to geometry*. Dover Publications, Mineola, 1998.

[21] R. Karwowski. *Improving the process of plant modeling: The L+C modeling language*. PhD thesis, University of Calgary, September 2002.

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

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

[24] 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.

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

[26] P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Měch. Visual models of plant development. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages,* Vol. III*: Beyond words*, pages 535–597. Springer, Berlin, 1997.

[27] P. Prusinkiewicz and J. Hanan. *Lindenmayer systems, fractals, and plants*, volume 79 of *Lecture Notes in Biomathematics*. Springer-Verlag, Berlin, 1989 (second printing 1992).

[28] P. Prusinkiewicz, J. Hanan, and R. Měch. An L-system-based plant modeling language. In M. Nagl, A. Schürr, and M. Münch, editors, *Applications of graph transformations with industrial relevance*, Lecture Notes in Computer Science 1779, pages 395–410. Springer-Verlag, Berlin, 2000.

[29] P. Prusinkiewicz, K. Krithivasan, and M. G. Vijayanarayana. Application of L-systems to algorithmic generation of South Indian folk art patterns and karnatic music. In R. Narasimhan, editor, *A perspective in theoretical computer science — commemorative volume for Gift Siromoney*, pages 229–247. World Scientific, Singapore, 1989. Series in Computer Science Vol. 16.

[30] 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.

[31] A. Rosenfeld. A note on cycle grammars. *Information and Control*, 27:374–377, 1975.

[32] M. Sabin. Subdivision surfaces. Tutorial notes, Shape Modeling International 2002 (Banff, Canada, May 18, 2002), 25 pp.

[33] F. F. Samavati and R. Bartels. Multiresolution curve and surface representation: reversing subdivision rules by least-squares data fitting. *Computer Graphics Forum*, 18(2):97–119, June 1999.

[34] W Sierpiński. Sur une nouvelle courbe qui remplit tout une aire plaine. *Bull. Acad. Sci. Cracovie, Série A*, pages 462–478, 1912. Reprinted in W. Sierpiński, *Oeuvres choisies*, S. Hartman et al., editors, pages 52–66, PWN – Éditions Scientifiques de Pologne, Warsaw, 1975.

[35] G. Siromoney, R. Siromoney, and T. Robinson. Kambi kolam and cycle grammars. In R. Narasimhan, editor, *A perspective in thepretical computer science. Commemorative volume for Gift Siromoney*, pages 267–300. World Scientific, Singapore, 1989.

[36] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for computer graphics: theory and applications.* Morgan Kaufman, San Francisco, CA, 1996.

[37] J. Warren and H. Weimer. *Subdivision methods for geometric design.* Morgan Kaufman, San Francisco, CA, 2002.

[38] D. Zorin, P. Schröder, A. DeRose, L. Kobbelt, A. Levin, and W. Sweldens. Subdivision for modeling and animation. SIGGRAPH 2000 Course Notes.