Determination of fractal dimensions from equivalent L systems

by M. Alfonseca
A. Ortega

This paper revises a few existing methods for computing fractal dimensions, underlines their dependency on the graphical properties of the curves, and proposes and discusses a new method, based on the representation of fractals by means of Lindenmayer systems, that makes use of the structure of L systems to compute the fractal dimension. The method is implemented in Prolog, and its limitations and usefulness are discussed.

Introduction

The concept of dimension is very old and seems easy and evident. We live in a space with three dimensions: length, width, and depth. Some of the objects in our environment are approximately bidimensional: a sheet of paper, a picture, a tabletop. Others have a single prevalent dimension: a distant road, a pencil line drawn on paper. What we call *dimension* may sometimes be defined as the number of directions in which movement is allowed.

Things appear very clear and elegant: Dimensions are consecutive integers: 0 (a point), 1 (a line), 2 (a surface), 3 (a volume), with no doubtful cases. Some do exist, however, as Mandelbrot proved in his famous book on fractals [1]. Depending on the size of the observer, a ball of thread can be considered as

- A point (zero dimensions) if the observer is very large (a mountain, a planet) or very far away.
- A sphere (three dimensions) if the observer is comparable to the size of the ball (a human being) and is located near the ball.
- A twisted line (one dimension) if the observer is smaller than the ball (an ant) and very near it.
- A twisted cylinder (three dimensions) if the observer is much smaller than the ball (a bacterium).
- A set of isolated points (zero dimensions) if the observer is even smaller and can see the atoms.
- A set of spheres (three dimensions), if the observer's size is comparable to that of the atoms.
- And so forth.

In 1890, the Italian mathematician Giuseppe Peano defined a curve with several strange properties, which was called a *monstrous curve*. It is a line (and therefore appears to be one-dimensional), but it fills a square (in the sense that it goes through every point in the square) and therefore could be considered two-dimensional. Another curious property of this curve is that it has no tangent or derivative at any point.

There are many other famous monstrous curves, such as the one devised by Helge von Koch in 1904, with a shape that reminds us of a snowflake. Like the Peano curve, it has no derivative at any point, and its longitude is infinite,

0018-8646/01/$5.00 © 2001 IBM

even though its size is limited. Its dimension seems to be larger than 1, although it does not fill the plane, and thus cannot reach 2.

In 1919, Hausdorff proposed a new definition of dimension, applicable to such doubtful cases, to distinguish them from normal surfaces and lines. With his definition, monstrous curves in the plane may have a fractional dimension between 1 and 2. Thus, Peano's curve has a Hausdorff dimension of 2, and Von Koch's snowflake has a Hausdorff dimension of

$$\frac{\log(4)}{\log(3)} = 1.2618595071429\ldots. \quad (1)$$

Other alternative definitions of dimension were proposed during the twentieth century [2, 3]. Most are similar to the Hausdorff dimension and have the same value in many cases, but differ in details and special circumstances—for example, the Hausdorff–Besicovitch dimension, the Minkowsky dimension, and the box-counting dimension. Most of them are known as *fractal dimensions*, and they are used in different situations.

The name *fractal*, introduced in 1975 by Mandelbrot [1, 4], applies to objects that have some special properties, such as self-similarity (containing copies of themselves), underivability at every point, and/or a fractal dimension greater than their integer topological dimension. They are appropriate for the description of natural shapes, and have been used successfully to code and compress images [5–7].

Fractals have been generated or represented by different means, such as fractional Brownian movements, recursive mathematical families of equations (such as those that generate the Mandelbrot set), and recursive transformations (generators) applied to an initial shape (the initiator). This paper discusses only the latter.

L systems, devised in 1968 by Lindenmayer [8], are also called parallel-derivation grammars, and differ from Chomsky grammars because derivation is not sequential (a single rule is applied at every step) but parallel (as many rules as possible are applied at every step).

L systems are highly appropriate for representing fractal objects obtained by means of recursive transformations [9]. The initiator maps to the axiom of the L system, and the generator becomes the set of production rules, while recursive applications of the generator to the initiator correspond to successive derivations of the axiom. The fractal corresponds to the limit of the word derived from the axiom when the number of derivations tends to infinity. Something else is needed, however: a graphic interpretation that makes it possible to convert each of the words generated by the L system into a visible graphic object.

Two different families of graphic interpretations of L systems have been used: turtle graphics and vector graphics. In a previous paper [10] we proved a fractal-equivalence theorem between two families of L systems, one associated with a turtle graphics interpretation, the other with vector graphics. The two families are interesting because they can be used to represent most of the fractals in the literature. Our theorem makes it possible to focus here on turtle graphics without a significant loss of generality.

In another previous paper [11], we described a preliminary version of the algorithm presented here, written in APL2. The current paper, however, contains a full treatment of the different special cases that may arise, which would make our definition of fractal dimension invalid or divergent. We also consider in detail the problems due to the fact that a fractal curve may be self-overlapping. The algorithm is also applied to a new class of fractals, defined by L systems with more than one nontrivial symbol. Most of the examples in this paper are different, and have been chosen to demonstrate these new and problematic cases. Finally, a version of the algorithm is provided that has been written in Prolog, rather than APL2.

Calculating the fractal dimension of self-similar curves

A wide spectrum of techniques have been used to estimate the fractal dimension of self-similar curves [12–14]. We mention here two of the most important.

Ruler dimension estimation

This method computes the fractal dimension of a line as a function of two measurements taken while "walking" the fractal line in a number of discrete steps. We take as unity the distance between the beginning and the end of the fractal line to be walked. The first measurement is p_1, the length of the step used, or *pitch length*, which must be constant during the whole walk. The second is the number of steps needed to reach the end of the walk by following the fractal curve, $N(p_1)$.

We call D_{p_1} the number for which the following relation holds:

$$N(p_1) \approx p_1^{-D_{p_1}}. \quad (2)$$

If we take logarithms on both sides of this equation, we obtain

$$\log[N(p_1)] = -D_{p_1}\log(p_1). \quad (3)$$

The fractal dimension is the limit of D_{p_1} when p_1 tends to zero:

$$D_{p_l} = \lim_{p_l \to 0+} \left\{ \frac{-\log[N(p_l)]}{\log(p_l)} \right\}. \quad (4)$$

Box dimension estimation

This method computes the fractal dimension of a line as a function of two measurements taken while covering the fractal line with a number of discrete boxes. If we call $N(d)$ the number of boxes of linear size d necessary to cover a set of points distributed in a two-dimensional plane, the box dimension is defined as the exponent D_b in the equation

$$N(d) \approx d^{-D_b}. \quad (5)$$

If we take logarithms on both sides of this equation, we obtain

$$\log[N(d)] = -D_b \log(d). \quad (6)$$

The fractal dimension is defined as the limit of D_b when d tends to zero:

$$D_b = \lim_{d \to 0+} \left\{ \frac{-\log[N(d)]}{\log(d)} \right\}. \quad (7)$$

There are many variations to this simple scheme. Some of them assign a weight to each box, depending on the number of points it contains. Instead of counting the number of boxes, another variation estimates the information entropy for the set of boxes, where the number of points is considered to be the information.

Alternative ways of calculating the box dimension change the shape and the nature of the set of boxes. Square-shaped boxes are usually used to define the grid, but they can be placed at any position and orientation. Families of concentric circular boxes with increasing radius are also used.

Calculating the fractal dimension from the equivalent L system

All of the techniques described in the previous paragraphs attempt to measure the fractal dimension as a ratio between how much the curve grows in length and how much it advances. We have tried to reach the same result by operating directly on the L system that represents the fractal curve, without performing any graphical representation.

Each word in the derivation represents a given configuration of the recursive generation of the fractal curve. The production rules embody the allowed transformation between configurations. Therefore, the growth of the words is related to the corresponding growth of the curve. The graphic interpretation of the L system makes it possible to assign bidimensional coordinates to the letters in each word. Once these coordinates have been computed, it is straightforward to obtain the distance between different points. These distances may be used as a measure of how much the curve grows in length. Performing operations on strings should be an easier method of computing the fractal dimension than the computation of a limit.

The turtle graphics interpretation

As stated before, two different graphic interpretations may be used to relate a given L system to a fractal curve. We have proved elsewhere [10] the equivalence of two wide families of systems in both sets. Therefore, in this paper we consider only the turtle graphics interpretation. A fractal generated by means of the vector graphics interpretation may be converted by our algorithm to an equivalent L system that uses the turtle graphics interpretation.

Created in 1980 by Papert [15], turtle graphics describes the trail left by an invisible "turtle," whose state at every instant is defined by its position and the direction in which it is looking. The state of the turtle changes as it moves a step forward or as it rotates by a given angle in the same position.

Turtle graphics interpretations can exhibit different levels of complexity. The version we use here is the following:

- The angle step of the turtle is $\alpha = (2k\pi/n)$, where k and n are two integers.
- The alphabet of the L system can be expressed as the union of the four disjoint subsets: N, D, M, $\{+, -, (,)\}$. The symbols in the alphabet are graphically interpreted as follows:
  - $+$ increases the turtle angle by α.
  - $-$ decreases the turtle angle by α.
  - (stacks the current position and orientation of the turtle.
  -) moves the turtle invisibly to the position and orientation stacked at the top of the stack and pops it.
  - A in N leaves the turtle state unchanged. We call A a *nongraphic* letter.
  - F in D moves the turtle one step forward, in the direction of its current angle, leaving a visible trail. We call F a *draw* letter.
  - f in M moves the turtle one step forward, in the direction of its current angle, with no visible trail. We call f a *move* letter.

In summary, a given fractal may be represented by means of two components: an L system and a turtle graphics interpretation, with a given angle step. The length of the step (the scale) is reduced at every derivation in the appropriate way, so that the curve always occupies the same space.

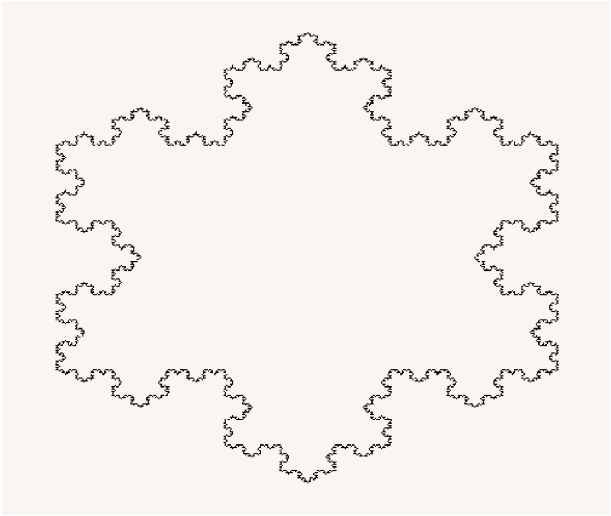

Figure 1

Von Koch snowflake curve, a well-known example of a fractal with a fractional dimension.

A string under a turtle graphics interpretation is said to be *angle-invariant* if the directions of the turtle at the beginning and the end of the string are the same. We call AID0L (*angle-invariant* D0L, where D0L describes a deterministic context-free L system [10]) the set of D0L systems such that the right-hand side of all of their rules is an angle-invariant string. In the following we restrict ourselves to AID0L systems.

Fractal curves represented by a single symbol

The fractal curves described in this section can be represented by an L system which contains a single draw symbol and no move or nongraphic symbols. The production set, therefore, consists of a single rule, apart from the trivial rules for symbols +, −, (, and).

Informally, the algorithm takes advantage of the fact that the right side of the only applicable rule provides a symbolic description of the fractal generator, which can thus be completely described by a single string. Our algorithm computes two numbers: The first is the length N of the visible walk that follows the fractal generator (equal in principle to the number of draw symbols in the generator string, but see below). The second is the distance d in a straight line from the start to the endpoint of the walk, measured in turtle step units (this number can also be deduced from the string). The fractal dimension is then

$$D = \frac{\log(N)}{\log(d)}. \tag{8}$$

The scale reduction at every derivation is such that, starting with an axiom equal to the left side of the only rule, the distance between the origin and the end of the graphical representation of the strings is always the same. The example given below illustrates the use of the algorithm.

The PD0L scheme

$$\begin{aligned} F &::= F+F--F+F \\ + &::= + \\ - &::= - \end{aligned} \tag{9}$$

with axiom $F--F--F$ and a turtle graphic interpretation, where $\{F\}$ is a draw symbol and the step angle is 60°, represents the fractal whose fifth derivation appears in **Figure 1** (Von Koch snowflake curve).

The only string to be considered is

$$F+F--F+F. \tag{10}$$

This string describes the fractal generator. The number of steps along the walk (N) is the number of draw symbols in the string, 4 in this case. The distance d between the extreme points of the generator, computable from the string by applying to it the turtle interpretation, is 3. Therefore, the dimension is

$$D = \frac{\log(4)}{\log(3)} = 1.2618595071429\ldots, \tag{11}$$

in accord with the results obtained by other methods, specified by Mandelbrot in Reference [1], p. 42.

Problems in the previous definition

- The distance d in the denominator may be zero. Computed by our formula, D becomes zero. An example is the PD0L scheme

$$\begin{aligned} F &::= F+F+F+F+ \\ + &::= + \\ - &::= - \end{aligned} \tag{12}$$

  with a step angle of 90°. We exclude these cases because they do not usually give rise to fractal curves, but to the same figure indefinitely repeated (in the example, a square).
- The distance d in the denominator may be 1. Computed by our formula, D becomes infinite. An example is the PD0L scheme

$$\begin{aligned} F &::= F+F++F++F+ \\ + &::= + \\ - &::= - \end{aligned} \tag{13}$$

with a step angle of 60°. We also exclude these cases because in every step of derivation the curve expands and is not limited to a finite space; therefore, it is not a fractal in the strict sense.

- The length N of the visible walk may not be equal to the number of draw symbols in the generator string. This may happen in two ways:
  - The turtle graphic associated with the string passes more than once along a set of points with a nonzero measure, as in the PD0L scheme

$$F ::= F+FF+++F++F-FF+++F++F--F$$
$$+ ::= +$$
$$- ::= - \qquad (14)$$

    with a step angle of 45° and axiom $F++F++F++F$. **Figure 2** represents the generator of the corresponding fractal curve and its third derivation. Our algorithm has been refined to take this case into account in such a way that the appropriate value of N is computed, where such sets of points are counted only once. This means that the value of N may be noninteger, as in this case, where its value is not 10 (the number of F in the string), but 9.4142 . . . (8 plus the square root of 2).
  - The turtle graphic associated with a derivation of the string passes more than once along a set of points with a nonzero measure, as in the PD0L scheme

$$F ::= F+FF-F-FF+F$$
$$+ ::= +$$
$$- ::= - \qquad (15)$$

    with a step angle of 90° and axiom $F+F+F+F$. **Figure 3** represents the generator of the corresponding fractal curve and its fourth derivation. In this case, we replace the definition of fractal dimension we are using with

$$D = \lim\left(\frac{\log(N)}{\log(d)}\right), \qquad (16)$$

    where the limit is taken on the string of derivations from axiom F. Our algorithm computes this case by taking a certain number of derivations until the quotient converges. The resulting dimension is approximately equal to 1.6, rather than 1.77, as computed from the string.

Fractal curves represented by several equivalent symbols

Our algorithm is also immediately applicable to those L systems with more than one rule, where all of the right parts of the rules give rise to identical fractal dimensions. Let us examine a couple of examples:

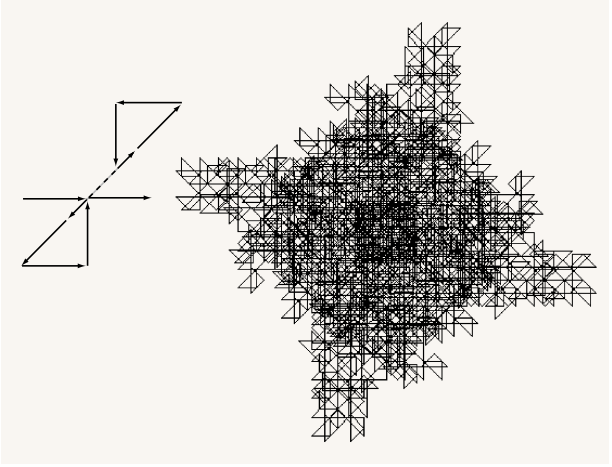

Figure 2

Curve obtained from a generator that passes twice through the same set of points.

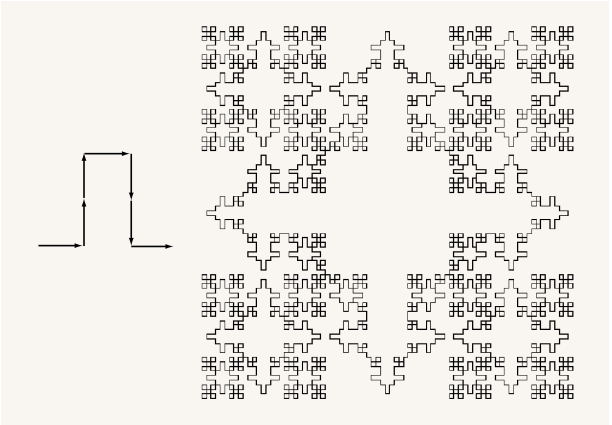

Figure 3

Curve that passes twice through the same set of points, although its generator does not.

- The PD0L scheme

$$F ::= -G+F+G-$$
$$G ::= +F-G-F+$$
$$+ ::= +$$
$$- ::= - \qquad (17)$$

  with axiom F and a turtle graphic interpretation, where $\{F, G\}$ are draw symbols and the step angle is 60°,

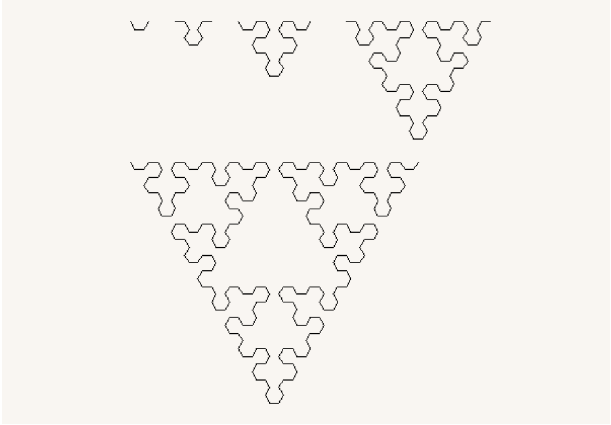

Figure 4

Fractal curve represented by two rules with the same derived dimension.

represents the fractal whose first five derivations appear in **Figure 4**. In this example, there are two strings to be considered:

$$-G+F+G-$$

$$+F-G-F+. \tag{18}$$

Applying our algorithm to each of them, we obtain the same estimation of the fractal dimension, 1.58496 . . . . Therefore, the fractal dimension of the corresponding curve must be the same, in accord with Mandelbrot's results [1].

- The PD0L scheme

$$F\ ::=\ +A--A+$$

$$A\ ::=\ -F++F-$$

$$+\ ::=\ +$$

$$-\ ::=\ - \tag{19}$$

with axiom $A--A--A--A--A--A$ and a turtle graphic interpretation, where A is a nongraphic symbol, F is a draw symbol, and the step angle is 30°, provides another way to represent the Von Koch snowflake curve in Figure 1, where only one of every two derivations generates a visible curve. In this example, there are two strings to be considered:

$$+A--A+$$

$$-F++F-. \tag{20}$$

Applying our algorithm to each of them, we obtain the same estimation of the fractal dimension, 1.261859 . . . . Therefore, the fractal dimension of the corresponding curve is again the same. Alternatively, taking advantage of the fact that one of every two steps is invisible, we could consider the result of the following two-step derivation:

$$F \rightarrow +A--A+ \rightarrow +-F++F----F++F-+. \tag{21}$$

The string $+-F++F----F++F-+$ can also be considered a description of the fractal generator. Applying our algorithm to it, we again obtain a result of 1.261859 . . . .

- The PD0L scheme

$$P::=PFU-F+Q+F-PF$$

$$Q::=Q+F-PFR++F--Q+F-$$

$$R::=R++F--Q+F-S+++F---R++F--$$

$$S::=S+++F---R++F--T--F++S+++F---$$

$$T::=T--F++S+++F---U-F+T--F++$$

$$U::=U-F+T--F++PFU-F+$$

$$F::=\varepsilon$$

$$+\ ::=\ +$$

$$-\ ::=\ - \tag{22}$$

with axiom $P++P++P$ and a turtle graphic interpretation, where F is a graphic symbol, $\{P, Q, R, S, T, U\}$ are nongraphic symbols, ε is the empty string, and the step angle is 60°, provides another way to represent the Von Koch snowflake curve in Figure 1. In this example, the dimension estimated by our algorithm gives the correct dimension for every symbol after the first iteration: 1.261859 . . . .

In this example there are several apparently different rules. In fact, the rules are very dissimilar. After carefully studying them one could state the following:

- The number of symbols on the right side is always equal to 8.
- Four of the eight symbols are graphics.

Therefore, the rules are structurally similar.

The algorithm

The crucial part of the algorithm is the computation of N, the length of the visible walk followed by the fractal generator or the sequence of derived strings, where repeated walks are eliminated. To do this, we need

to derive an exact unambiguous representation of all of the points in the walk, together with information about the visibility of each step. A typical Cartesian x–y representation is not appropriate, for we are dealing with irrational numbers for most turtle angle steps, and the precision of real numbers in a computer is finite, which means that only a subset of rational numbers can be represented. Our representation takes into account the fact that the turtle approach ensures that any point of interest in the plane can be reached by a finite sequence of unitary vectors taken from a set of n, where $\alpha = (2k\pi/n)$ is the turtle angle step. Thus, taking into account that vector addition is commutative, we can represent each point by a set of n integer numbers stating how many vectors of each kind are needed to reach that point from the origin by following the turtle movements, without specifying the order of the vectors.

To make the representation unique for every point, we must make sure that the walk from the origin to the point is minimal. We do this by performing the following additional computations on the set of integers that represent a point:

- For every n, we eliminate all n-sided regular polygons, represented by sequences of all ones.
- For odd n, we eliminate all smaller regular polygons with a number of sides that is a prime submultiple of n. They are easily recognized as sequences of ones and zeros.
- For even n (where, for every vector in the set, its opposite vector is in the set), we have to be subtler: A part of any regular polygon with a number of sides an odd prime submultiple of n, longer than half the polygon, can be replaced by a smaller set of vectors going around the remainder of the polygon in the opposite direction. This can also be done easily by looking at the sequences of ones and zeros in the point representation.

For example, let α be 60°, which means that $n = 6$. The turtle walk defined by

$$F{+}{+}F{-}{-}F{+}{+}F{+}{+}F{+}{+}F$$

can be defined by the set of integers: (3, 0, 2, 0, 1, 0), which means that we must make three steps with angle 0°, two with angle 120°, and one with angle 240°. This can be obtained immediately from the string by counting walks according to directions. Since n is even, we have to reduce polygons. The sequence (1, 0, 1, 0, 1, 0), which is included in (3, 0, 2, 0, 1, 0), represents a triangle (a polygon of three sides, where 3 is an odd prime submultiple of n). The sequence can be replaced by (0, 0, 0, 0, 0, 0), the remainder polygon in the opposite direction. The point representation thus becomes (2, 0, 1, 0, 0, 0). Next, we

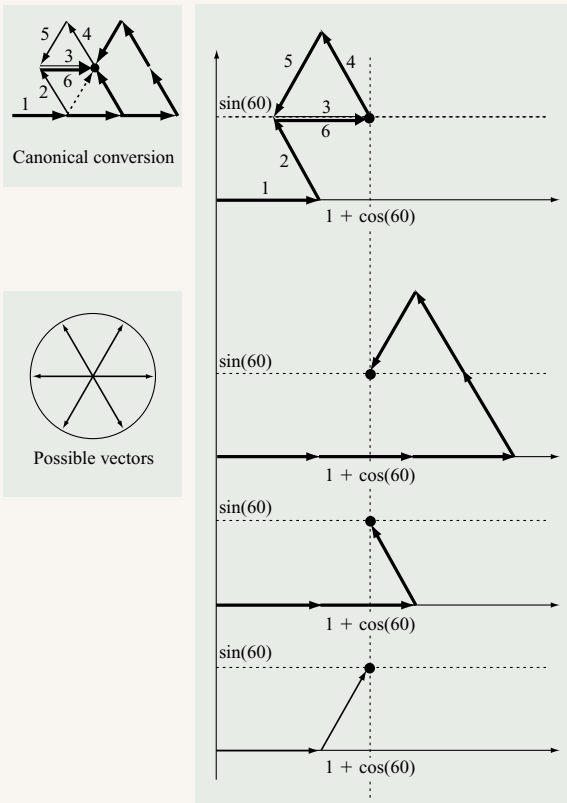

Figure 5

Obtaining the canonical representation of a point, independently of the path by which it was reached.

observe that the sequence (1, 0, 1, 0, 0, 0), contained in the latter, represents two sides of a triangle, which can be replaced by the vector corresponding to the third side in the opposite direction, (0, 1, 0, 0, 0, 0). Thus, the point reached by the above turtle string can be reduced to (1, 1, 0, 0, 0, 0). **Figure 5** shows the original string walk, plus those corresponding to the three subsequent representations of the endpoint. The last one is the minimum, which we take as the canonical representation of the endpoint.

It can easily be proved that this algorithm correctly computes the canonical integer representation of the endpoint of any turtle string. Thus, we can unambiguously locate and eliminate repeated walks (even parts of turtle movements, as in the example in Figure 3), simply by comparing any visible turtle step with all of the previous ones and replacing canonical representations of endpoints. A simple algorithm can do this with a complexity of $O(N^2)$.

Prolog implementation of the algorithm

The Prolog predicate in the listing below computes the fractal dimension of the fractal curve defined by an L system. The predicate receives the following arguments:

- Left: the left symbol of a rule in the L system.
- Draw: the set of draw symbols.
- Move: the set of move symbols.
- Nograph: the set of nongraphic symbols.
- Angle: the angle step of the turtle.
- InitialPoint: the coordinates of the initial point of the curve, usually (0, 0).
- *N*: the number of derivations.

The predicate returns the result of the computation in variable FractalDimension.

A set of facts describing the rules of the PD0L scheme must be stated before invoking the predicate `fractal_dimension`. These facts can be read from a file:

```
fractal_dimension(Left, Draw, Move,
                  Nograph, Angle,
                  InitialPoint, N,
                  FractalDimension) :-

% FIRST THE NTH DERIVATION IS OBTAINED
% FROM AXIOM ACCORDING TO THE PRODUCTION
% RULES.
  get_nth_derivation_from_axiom(Left, N,
                                String),

% THEN, THE GRAPHIC INTERPRETATION OF THE
% RESULTING STRING IS CALCULATED IN ORDER
% TO GET THE POINT REACHED.
  string_to_end_point(String, Draw, Move,
                      Nograph, Angle,
                      InitialPoint,
                      LastPoint),

% THE EUCLIDEAN DISTANCE BETWEEN THE TWO
% POINTS IS CALCULATED.
  distance(InitialPoint, LastPoint,
           Distance),

% THE EFFECTIVE LENGTH (WITHOUT
% OVERLAPPING SEGMENTS) DEPICTED BY THE
% CURVE IS CALCULATED.
  string_to_effective_length(String, Draw,
                             Move, Nograph,
                             Angle,
                             EffectiveLength),

% AND FINALLY THE FRACTAL DIMENSION IS
% ESTIMATED.
  FractalDimension is
  log(EffectiveLength)/log(Distance).
```

Conclusions

The L system that represents a fractal curve with a turtle graphics interpretation has proved to contain enough information for the computation of the fractal dimension of the curve, for an interesting family of systems. In some cases, the computation may have to be applied to a sequence of derivations, and thus will be exponentially slow, but this is a consequence of the inherent exponential growth of fractal curves. However, in many other cases it is not necessary to compute this limit, and the appropriate dimension can be obtained in one or two steps.

Fractals represented by L systems associated with a vector graphics interpretation are automatically covered by our algorithm, if they have previously been converted to equivalent L systems with a turtle graphics interpretation, using the algorithm described in Reference [10].

In the future, we will try to extend the method to more complicated turtle graphics interpretations and to different types of L systems, such as those which have rules whose left symbols do not lead to the same results. The primary handicap with these systems is that they have not been well documented. Most of the fractals in the literature belong to the same class as the examples in this paper.

Acknowledgment

This paper has been sponsored by the Spanish Interdepartmental Commission of Science and Technology (CICYT), Project No. TEL1999-0181.

References

1. B. B. Mandelbrot, *The Fractal Geometry of Nature*, W. H. Freeman and Company, New York, 1977.
2. K. Falconer, *Fractal Geometry: Mathematical Foundations and Applications*, John Wiley & Sons, Chichester, England, 1990.
3. Masaya Yamaguti, Masayoshi Hata, and Jun Kigami, "Mathematics of Fractals," *Translation of Mathematical Monographs*, Volume 167, American Mathematical Society.
4. S. D. Casey and N. F. Reingold, "Self-Similar Fractal Sets: Theory and Procedure," *IEEE Computer Graph. & Appl.* **14,** 73–82 (1994).
5. M. F. Barnsley, *Fractals Everywhere*, Academic Press, Inc., Boston, 1988.
6. T. Bedford, F. M. Dekking, M. Breeuwer, M. S. Keane, and D. van Schooneveld, "Fractal Coding of Monochrome Images," *Signal Process: Image Commun.* **6,** 405–419 (1994).
7. K. Culik II and S. Dube, "New Methods for Image Generation and Compression," *Proceedings of the Conference on Facts and New Trends in Computer Science*, H. Maurer, Ed., Springer-Verlag, Berlin, 1991, pp. 69–90.
8. A. Lindenmayer, "Mathematical Models for Cellular Interactions in Development" (in two parts), *J. Theor. Biol.* **18,** 280–315 (1968).
9. K. Culik II and S. Dube, "L-Systems and Mutually Recursive Function Systems," *Acta Informat.* **30,** 279–302 (1993).
10. M. Alfonseca and A. Ortega, "A Study of the Representation of Fractal Curves by L Systems and Their Equivalences," *IBM J. Res. & Dev.* **41,** 727–736 (1997).

11. M. Alfonseca and A. Ortega, "Using APL2 to Compute the Dimension of a Fractal Represented as a Grammar," *APL Quote Quad* **30,** 13–23 (2000).
12. F. M. Dekking, "Recurrent Sets," *Adv. Math.* **44,** 78–104 (1982).
13. F. M. Dekking, "Recurrent Sets: A Fractal Formalism," *Technical Report 82-32*, Technische Hogeschool, Delft, Netherlands, 1982.
14. TruSoft International Inc., Benoit application: *http://www.trusoft-international.com/benoit.html*.
15. S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books, New York, 1980.

Received January 21, 2001; accepted for publication August 16, 2001

Manuel Alfonseca *Universidad Autónoma de Madrid, Campus de Cantoblanco, 28049 Madrid, Spain (Manuel.Alfonseca@ii.uam.es).* Dr. Alfonseca is a professor at the University. He was formerly a Senior Technical Staff Member at IBM, having worked from 1972 to 1994 at the IBM Scientific Center in Madrid. Dr. Alfonseca was one of the developers of the APL/PC interpreter and related products; he has worked on computer languages, simulation, complex systems, graphics, artificial intelligence, object orientation, and theoretical computer science, and has published several books and about 170 technical papers, as well as 60 papers on popular science in a major Spanish newspaper. He is an award-winning author of 21 published books for children. Dr. Alfonseca holds a doctorate in electronics and an M.Sc. in computer science from the Universidad Politécnica de Madrid. He is a member of the Spanish Computer Society (SCS), the New York Academy of Sciences, the IEEE Computer Society, the ACM, the British APL Association, and the Spanish Association of Scientific Journalism.

Alfonso Ortega *Universidad Autónoma de Madrid, Campus de Cantoblanco, 28049 Madrid, Spain (Alfonso.Ortega@ii.uam.es).* Dr. Ortega is currently a lecturer at the University, and is also currently assigned for a year to CIEMAT (Center for Research on Energy and Environment). Formerly he was a lecturer at the Universidad Pontificia de Salamanca and worked at LAB2000 (an IBM subsidiary) as a software developer. He holds a doctorate in computer science from the Universidad Autónoma. Dr. Ortega has published ten technical papers on computer languages, complex systems, graphics, and theoretical computer science, and has collaborated in the development of several software products.