IBM Journal of Research and Development 45.6 (2001): 797 – 805

DOI: http://dx.doi.org/10.1147/rd.456.0797

Determination of fractal dimensions from equivalent L systems

Alfonso Ortega, Manuel Alfonseca

Universidad Autonoma de Madrid, Dept. Ingenieria Informatica

{Alfonso.Ortega, Manuel.Alfonseca}@ii.uam.es

Keywords: Fractals, Fractal dimension, L Systems, Grammar systems

Abstract

This paper revises a few existing methods to compute 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 the 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 bi-dimensional: a sheet of paper, a picture, a table top... 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. But there are some, 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.
- A sphere (three dimensions) if the observer is comparable to the size of the ball (as 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 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 (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 a snowflake. As the Peano curve, it has no derivative at any point, and its longitude is infinite, even though its size is limited. Its dimension seems to be larger than one, although it does not fill the plane, and thus cannot reach two.

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

$$\frac{\log(4)}{\log(3)} = 1,2618595071429...$$

Other alternative definitions of dimension were proposed along 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. Let us mention the Hausdorff-Besicovitch dimension, the Minkowsky dimension and the box-counting dimension. Most of them are called *fractal dimensions* and 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 is interested only in the latter.

L systems, devised in 1968 by Aristid 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 very appropriate to represent fractal objects obtained by means of recursive transformations [9]. The initiator maps to the axiom of the L system, the generator becomes the production rules, while the recursive applications of the generator to the initiator correspond to the 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 have 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 most of the fractals in the literature can be represented by means of them. Our theorem makes it possible to focus here on turtle graphics without a significant loss of generality.

In another previous paper [11], we have 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 non-trivial 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 has been used to estimate the fractal dimension of self similar curves [12-14]. We shall 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 will take as unity the distance between the beginning and the end of the fractal line to be walked. The first measurement is p_l, 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, $N(p_l)$.

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

$$N(p_l) \approx p_l^{-D_{p_l}}$$

If we take logarithms in both sides of this equation, we get:

$$\log(N(p_l)) = -D_{p_l} * \log(p_l)$$

The fractal dimension is the limit of D_{pl} when p_l tends to zero.

$$D_{pl} = \lim_{p_l \to 0+} \left(\frac{-\log(N(p_l))}{\log(p_l)} \right)$$

Box dimension estimation.

This method computes the fractal dimension of a line as a function of two measurements taken while covering the fractal line by 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

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

If we take logarithms in both sides of this equation, we get:

$$\log(N(d)) = -D_b * \log(d)$$

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

$$D_b = \lim_{d \to 0+} \left(\frac{-\log(N(d))}{\log(d)} \right)$$

There is a wide set of 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 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 used too.

Calculating the fractal dimension from the equivalent L system

All the techniques described in the previous paragraphs try 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 bi-dimensional co-ordinates to the letters in each word. Once these co-ordinates 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 will 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 Seymour Papert [15], turtle graphics describe the trail left by an invisible "turtle", whose state at every instant is defined by its position and the direction it is looking to. The state of the turtle changes as it moves a step forward, or as it rotates a given angle in the same position.

Turtle graphics interpretations come in different levels of complexity. The version we are using here is the following:

- The angle step of the turtle is $\alpha = \frac{2k\pi}{n}$, 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, {+,-,(,)}. Each symbol in the alphabet is graphically interpreted thus:

- + 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 will call A a *non-graphic* letter.
- F in D moves the turtle one step forward, in the direction of its current angle, leaving a visible trail. We will 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 will call f a *move* letter.

In summary: a given fractal may be represented by means of two components: an L system, and a turtle 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.

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) the set of the D0L systems such that the right-hand side of all their rules is an angle-invariant string. In the following we will 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 non-graphic 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 end point of the walk, measured in turtle step units (this number can also be deduced from the string). The fractal dimension would then be:

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

The scale reduction at every derivation will be 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

```
F ::= F+F--F+F
    + ::= +
    - ::= -
```

with axiom F--F--F, and a turtle graphic interpretation, where {F} is a draw symbol, and the step angle is 60 degrees, represents the fractal whose fifth derivation appears in figure 1 (von Koch snowflake curve).

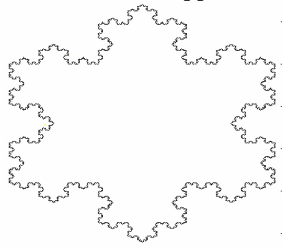

Figure 1: Von Koch snowflake curve, a well-known example of a fractal with a fractional dimension.

The only string to be considered is

```
F+F--F+F
```

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:

$$D = \frac{\log(4)}{\log(3)} = 1,2618595071429...$$

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

Problems in the previous definition

- The distance *d* in the denominator may be zero. Computed by our formula, D becomes zero.

  Example: the PD0L scheme

  ```
  F ::= F+F+F+F+
      +::=+
      -::=-
  ```

  with a step angle of 90 degrees. We will 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 one. Computed by our formula, D becomes infinite.

  Example: the PD0L scheme

  ```
  F ::= F+F++F++F+
      +::=+
      -::=-
  ```

  with a step angle of 60 degrees. We shall also exclude theses 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 to the string passes more than once along a set of points with a non-zero measure, as in the PD0L scheme:

```
F ::= F+FF+++F++F-FF+++F++F--F
          +::=+
          -::=-
```

with a step angle of 45 degrees 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, is 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 non-integer, 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).

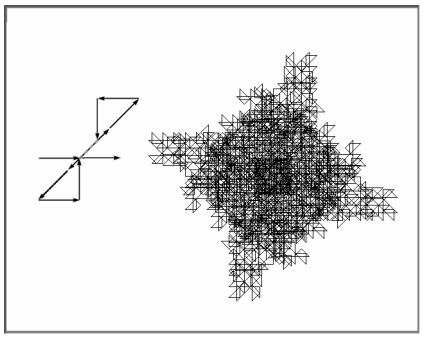

Figure 2: A curve obtained from a generator that passes twice through the same set of points.

- The turtle graphic associated to a derivation of the string passes more than once along a set of points with a non-zero measure, as in the PD0L scheme:

```
F ::= F+FF-F-FF+F
       +::=+
       -::=-
```

with a step angle of 90 degrees 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 by:

$$D = \lim \frac{\log(N)}{\log(d)}$$

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.

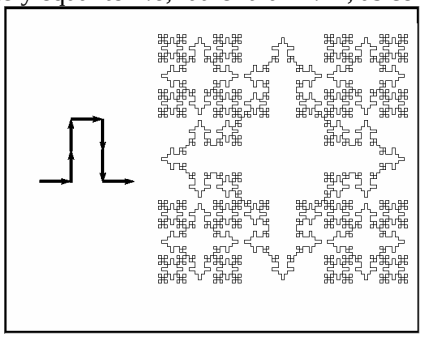

Figure 3: A curve that passes twice through the same set of points, although its generator does not.

Fractal curves represented by several equivalent symbols

Our algorithm is also immediately applicable to those L systems with more than one rule, where all the

right parts of the rules give rise to identical fractal dimensions. Let us see a couple of examples:

- The PD0L scheme

```
F ::= -G+F+G-
G ::= +F-G-F+
   +::=+
   -::=-
```

with axiom F, and a turtle graphic interpretation, where {F,G} are draw symbols, and the step angle is 60 degrees, represents the fractal whose first five derivations appear in figure 4.

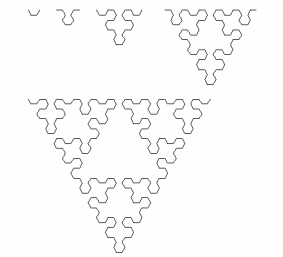

Figure 4: A fractal curve represented by two rules with the same derived dimension.

In this example, there are two strings to be considered:

```
-G+F+G-
+F-G-F+
```

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 results [1].

- The PD0L scheme

```
F ::= +A--A+
A ::= -F++F-
   + ::= +
   - ::= -
```

with axiom A--A--A--A--A--A, and a turtle graphic interpretation, where A is a non-graphic symbol, F is a draw symbol, and the step angle is 30 degrees, provides another way to represent the 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-
```

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 -> +A--A+ -> +-F++F----F++F-+
```

The string +-F++F----F++F-+ can also be considered a description of the fractal generator. Applying our algorithm to it, we again get 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::=ε
+::=+
-::=-

with axiom P++P++P, and a turtle graphic interpretation, where F is a graphic symbol, {P,Q,R,S,T,U} are non-graphic symbols, ε is the empty string and the step angle is 60 degrees, provides another way to represent the Koch snowflake curve in figure 1.

In this example, the dimension estimated by our algorithm gives the right 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 remarks:

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

So 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 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 assures 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 $\alpha = \frac{2k\pi}{n}$ 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 need to 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 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.

Let us look at an example: Let α be 60º, which means 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 have to make three steps with angle 0º, two with angle 120º, and one with angle 240º. 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 3 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 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 end point. The last one is the minimal, which we will take as the canonical representation of the end point.

It can easily be proved that this algorithm computes correctly the canonical integer representation of the end point 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 the previous ones and replacing canonical representations of end points. A simple algorithm can do this with a complexity of $O(N^2)$.

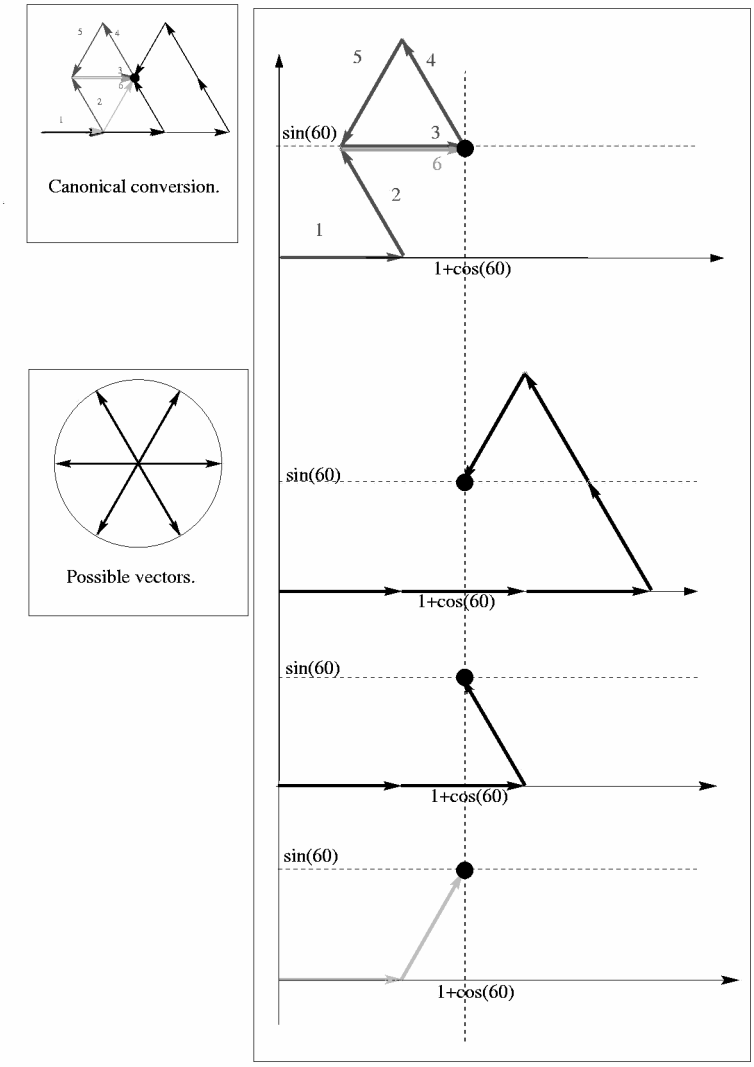

Figure 5: Obtaining the canonical representation of a point, independent of the path by which it was reached.

Prolog implementation of the algorithm

The Prolog predicate in listing 1 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 non graphic symbols.
- Angle: the angle step of the turtle.
- InitialPoint: the co-ordinates of the initial point of the curve, usually (0,0).
- N: 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 BOTH 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 ).
```

Listing 1: Prolog predicate to compute the fractal dimension

Conclusion

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 would be exponentially slow, but this is a consequence of the inherent exponential grow 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 to a vector graphics interpretation are automatically covered by our algorithm, if they are previously 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 main handicap with these systems is that they are little documented. Most of the fractals in the literature belong to the same class as the examples in this paper.

References

[1] Mandelbrot, B.B.: *The Fractal Geometry of Nature*, W.H.Freeman and Company, N.York, 1977.

[2] Falconer, K.: *Fractal Geometry: Mathematical Foundations and Applications*, John Wiley & Sons, Chichester, 1990.

[3] Masaya Yamaguti, Masayoshi Hata, Jun Kigami. *Mathematics of fractals*, Translation of Mathematical Monographs, volume 167. American Mathematical Society.

[4] Casey, S.D.; Reingold, N.F.: *Self-similar Fractal Sets: Theory and Procedure*, IEEE Computer Graphics and Applications, pp. 73-82, May 1994.

[5] Barnsley, M.F.: *Fractals everywhere*, Academic Press, Boston, 1988.

[6] Bedford, T.; Dekking, F.M.; Breeuwer, M.; Keane, M.S.; van Schooneveld, D.: *Fractal Coding of Monochrome Images*, Signal Procesing: Image Communication, Vol 6, pp. 405-419, 1994.

[7] Culik II, K.; Dube, S.: *New Methods for Image Generation and Compression*, Proc. Conf. on Facts and New Trends in Computer Science, ed. H. Maurer, Springer-Verlag, Berlin, 1991, pp. 69-90.

[8] Lindenmayer, A.: *Mathematical models for cellular interactions in development* (two parts). J. Theor. Biol. 18,. pp. 280-315, 1968.

[9] Culik II, K.; Dube, S.: *L-Systems and Mutually Recursive Function Systems*, Acta Informatica, Vol 30, pp. 279-302, 1993.

[10] Alfonseca, M.; Ortega, A.: *A study on the different representations of fractal curves by L systems and their equivalences*, IBM Journal of Research and Development, Vol. 41:6, pp. 727-736, 1997.

[11] Alfonseca, M.; Ortega, A.: *Using APL2 to Compute the Dimension of a Fractal Represented as a Grammar*, APL Quote Quad, Vol. 30:4, pp. 13-23, 2000.

[12] Dekking, F.M.: *Recurrent Sets*, Advances in Mathematics, Vol. 44:1, pp. 78-104, 1982.

[13] Dekking, F.M.: *Recurrent Sets: a fractal formalism*,Technical Report 82-32, Technische Hogeschool, Delft, 1982.

[14] TruSoft Int'l Inc., Benoit application. http://www.trusoft-international.com/benoit.html.

[15] Papert, S.: *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books, New York, 1980.