.

Complex Systems in APL: Fractals, Evolving Cellular Automata and Artificial Life

Manuel Alfonseca

Alfonso Ortega

Universidad Autónoma de Madrid

e-mail: {Manuel.Alfonseca, Alfonso.Ortega}@ii.uam.es

Marina de la Cruz

CIEMAT

e-mail: marina@ciemat.es

Abstract

We have been working for several years on the representation, study and simulation of complex systems by means of formal methods. APL2 and other programming languages have been used to develop our tools and experiments. This paper summarizes our APL2 works on fractal sets, automatic programming of cellular automata and simulation of multi agent systems.

Acknowledgment

This paper has been sponsored by the Spanish Interdepartmental Commission of Science and Technology (CICYT), project numbers TEL1999-0181 and TIC2001-4937-E.

1. Fractals

Fractal curves are well-known mathematical constructions [1] that occupy intermediate positions between standard geometrical entities such as points and lines or lines and surfaces. An appropriate extension of the concept of dimension makes it possible to describe fractals by means of fractionary dimensions. In this way, Cantor's fractal set of points (see figure 1) has a dimension of 1/3, von Koch's snowflake's cdimension (see figure 2) can be expressed as log 4/log 3 (or 1.26...) and Peano's plane filling curve (see figure 3) has a dimension of 2, quite untypical for a curve.

In a previous paper [2, 3] we have described how this type of fractals can be represented in a straightforward way by means of a special kind of grammars called L systems, originally described by A.Lindenmayer in the nineteen sixties [4], and how to compute their dimension from their grammar [5, 6]. All of them can be obtained from some initiator, by applying iteratively a rule or iterator. The fractal curve is the limit of the generated curve when the number of iterations tends to infinite.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or comercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permision and/or a fee. APL02, 7/02, Madrid, Spain. © 2002 ACM 1-58113-577-7....$5.00

- The initiator of Cantor's fractal set is a straight segment. The iterator rule removes the central third. Figure 1 shows the six first steps of this fractal set.
- Von Koch snowflake's initiator is also a straight segment. The iterator rule replaces the central third by the opposite sides of an equilateral triangle. Figure 2.a shows the iterator. Figure 2.b shows an intermediate stage of the fractal curve obtained from an equilateral triangle.
- Peano's curve is continuous, but fills any arbitrary rectangular ball in R2. Figure 3.a shows its initiator and iterator; figure 3.b shows an intermediate phase of this fractal.

Figure 1: Six steps in Cantor's fractal set

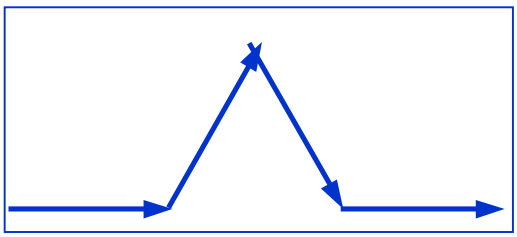

Figure 2a: Iterator for Von Koch's curve

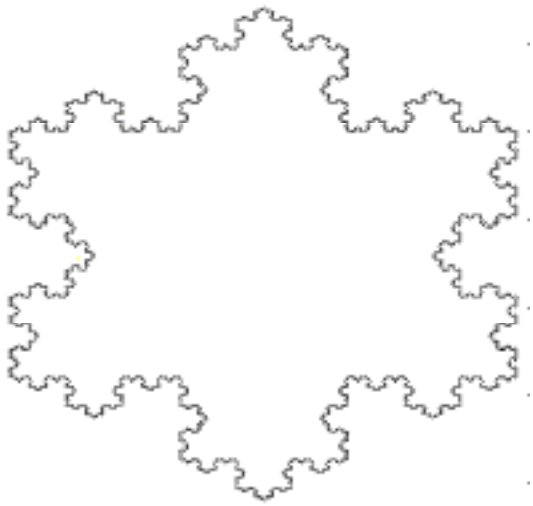

Figure 2b: Approximation to Von Koch's snowflake curve

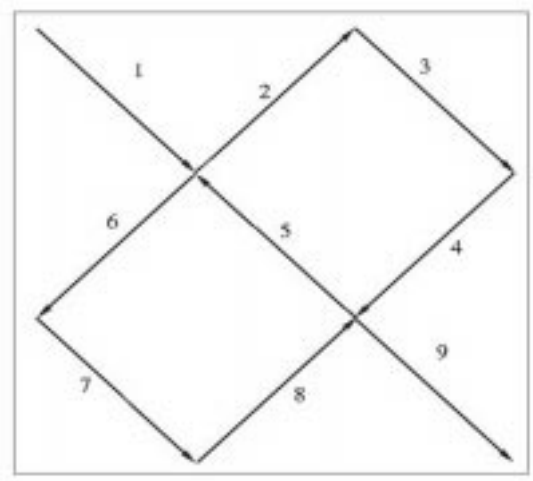

Figure 3a: Iterator for Peano's curve

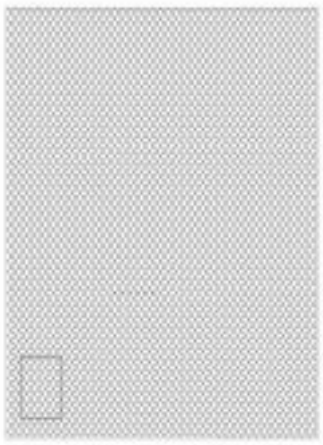

Figure 3b: Approximation to Peano's curve

There are two other kinds of fractals, less amenable to be represented by means of grammars:

- Complex curves obtained as the boundary between the domains of convergence and divergence of some mathematical function (Mandelbrot and Julia sets). A special case of these curves are Pickover's biomorphs [7, 8].
- Curves obtained by random means, similar to those generated by brownian movements.

In a recent paper [9] we have proposed a way in which an extended definicion of Lindenmayer grammars (parametric two-dimensional L systems) is capable of representing fractals of the first type. In this paper we describe how this set of algorithms may be programmed in APL2 to easily generate pictures showing selected sections of the Mandelbrot set, Julia sets, biomorphs and other fractals in the same family.

Mandelbrot set

The Mandelbrot set is the boundary between the convergence and divergence domains of the recursive complex function zn+1=f(zn)=zn2+c where c=cx+cyj is taken from a ball in the complex plane and z0=0 [1].

The following APL2 function generates the well-known Mandelbrot set in the form of a bit map (see figure 4):

```
[0]   Z←MANDEL4 N;A;B;C;W;D;F;F0;F1;F2;F3;F4;F5;F6;F7;A1;A2;D1;D2
[1]   ⍝ THRESHOLD, RESOLUTION, INITIAL COORDS., LENGHTS OF ZOOM AREA
[2]   A←150 350 350 ¯2.4 ¯1.2 3.2 2.4
[3]   W←A[2 3]⍴0
[4]   C←(A[4]+0J1×A[5])+((⍳A[2])×A[6]÷A[2])∘.+0J1×(⍳A[3])×A[7]÷A[3]
[5]   A1←A[2 3]⍴A2←0
[6]  L:F←A[1]<|B←C+W×W
[7]   F0←(A1=A2)∧(A1≤150)∧F
[8]   F1←(A1=A2)∧(A1>150)∧(A1≤160)∧F
[9]   F2←(A1=A2)∧(A1>160)∧(A1≤170)∧F
[10]  F3←(A1=A2)∧(A1>170)∧(A1≤180)∧F
[11]  F4←(A1=A2)∧(A1>180)∧(A1≤190)∧F
[12]  F5←(A1=A2)∧(A1>190)∧(A1≤200)∧F
[13]  F6←A1≠A2
[14]  F7←~F0∨F1∨F2∨F3∨F4∨F5∨F6
[15]  A2←A2+1
[16]  A1←(F7×A1+1)+(~F7)×A1
[17]  W←(F7×B)+(~F7)×W
[18]  →(0<N←N-1)/L
[19]  C←(⎕AF 'BM'),(⌽,256 256 256 256⊤1078+⍴,B),(4⍴0),⌽256 256 256 256
      ⊤1078
[20]  C←C,40 0 0 0,(⌽256 256 256 256⊤A[2]),(⌽256 256 256 256⊤A[3]),1 0
[21]  C←C,8 0,(24⍴0)
[22]  D1←(255-⌊(51÷30)×0,⍳150),105⍴255
[23]  D2←(151↑D1),(10/100 110 120 130 140),55⍴200
[24]  A1←((4×⌈(↑⍴A1)÷4),1↓⍴A1)↑A1
[25]  C←C,(,D1,D2,D1,[1.5]0),,⊖⍉A1
[26]  (⎕AF C)⎕FV 'MANDEL.BMP'
```

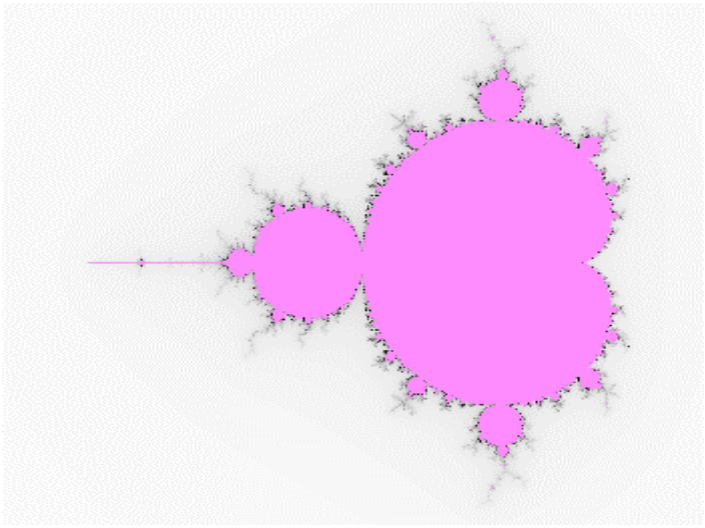

Figure 4: Mandelbrot set

To change the resolution in the preceding picture or to zoom in selected parts of the Mandelbrot set, it is only necessary to change the data in line [2]. Figure 5, for instance, shows the result of changing the line to:

```
[2]    A←150 350 350 ¯0.74591 0.11196,2⍴0.00143
```

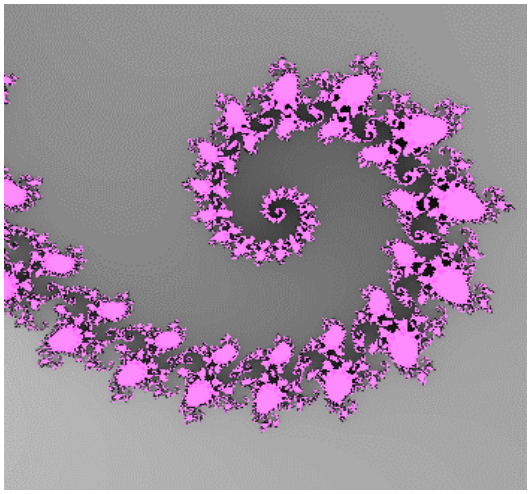

Figure 5: A part of the Mandelbrot set with higher resolution

Julia sets

The Julia set is the boundary between the convergence and divergence domains of the recursive complex function zn+1=f(zn)=zn2+c, where c=cx+cyj is a fixed complex point and z0 is taken from a ball in the complex plane [1].

The following APL2 function generates a Julia set in the form of a bit map (see figure 6):

```
[0]    Z←JULIA1 N;A;B;C;W;D;F1;F2;D1;D2
[1]    ⍝ THRESHOLD, RESOLUTION, INITIAL COORDS., LENGHTS, C
[2]    A←16 400 400 ¯1.5 ¯1.5 3 3 ¯0.745 0.113
[3]    C←A[8]+0J1×A[9]
[4]    W←(A[4]+0J1×A[5])+((⍳A[2])×A[6]÷A[2])∘.+0J1×(⍳A[3])×A[7]÷A[3]
[5]   L:F1←A[1]<|B←C+W×W
[6]    F2←~F1
[7]    W←(F1×W)+F2×B
[8]    →(0<N←N-1)/L
[9]    C←(⎕AF 'BM'),(⍬,256 256 256 256⊤1078+⍴,B),(4⍴0),⍬256 256 256 256
       ⊤1078
[10]   C←C,40 0 0 0,(⍬256 256 256 256⊤A[2]),(⍬256 256 256 256⊤A[3]),1 0
[11]   C←C,8 0,(24⍴0)
[12]   B←((4×⌈(↑⍴B)÷4),1↓⍴B)↑B
[13]   C←C,(,(⍬0,⍳255),(0,⍳255),(0,⍳255),[1.5]0),L,256||⊖⍉B
[14]   (⎕AF C)⍎FV 'JULIA1.BMP'
```

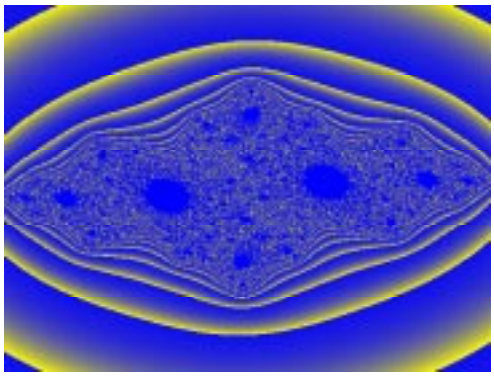

Figure 6: A Julia set

Pickover's biomorphs

"Biomorph" is the name that Clifford A. Pickover [7, 8] gave to the shapes he discovered as a result of a programming bug when studying fractal properties of several complex functions. They can be considered as a particular case of Julia sets, with a divergence condition slightly different.

The following APL2 function generates a Pickover biomorph in the form of a bit map (see figure 7):

```
[0]    Z←BIOM3 N;A;B;C;W;D;F1;F2;D1;D2;A1;A2
[1]    ⍝ THRESHOLD, RESOLUTION, INITIAL COORDS., STEPS, COLORS
[2]    A←10 200 200 ¯4 ¯4 8 8 0.05 0.75
[3]    C←A[8]+0J1×A[9]
[4]    W←(A[4]+0J1×A[5])+((⍳A[2])×A[6]÷A[2])∘.+0J1×(⍳A[3])×A[7]÷A[3]
[5]    D←A[2 3]⍴1
[6]    A1←A[2 3]⍴A2←0
[7]   L:F0←(A1=A2)∧((A[1]>|9○B)∨A[1]>|11○B)∧A[1]≤|B←C+W*5
[8]    F1←(A1=A2)∧((A[1]≤|9○B)∨A[1]≤|11○B)∧A[1]≤|B←C+W*5
[9]    F2←A1≠A2
[10]   F3←~F0∨F1∨F2
[11]   D1←(A[1]>|9○B)∨(A[1]≤|11○B)∨A[1]>|B
[12]   D2←0
[13]   D←(F3×D2)+(~F3)×D1
[14]   W←(F3×B)+(~F3)×W
[15]   A2←A2+1
[16]   A1←(F3×A1+1)+(F0×¯1)+(F1∨F2)×A1
[17]   →(0≤N←N-1)/L
[18]   C←(⎕AF 'BM'),(⍬,256 256 256 256⊤1078+⍴,B),(4⍴0),⍬256 256 256 256
       ⊤1078
[19]   C←C,40 0 0 0,(⍬256 256 256 256⊤A[2]),(⍬256 256 256 256⊤A[3]),1 0
[20]   C←C,8 0,(24⍴0)
[21]   D1←256↑0 214 219 147 192 252 231 248 168 218 41 255
[22]   D2←256↑0 1 63 3 149 172 217 62 73 110 50 255
[23]   D3←256↑0 217 234 246 6 112 79 146 213 133 201 255
[24]   C←C,(,D3,D2,D1,[1.5]0),,⊖⍉A1+1
[25]   (⎕AF C)⍎FV 'BIOM3.BMP'
```

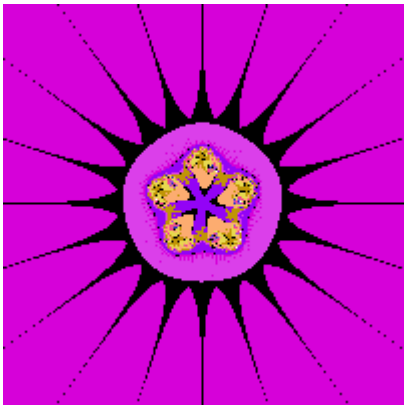

Figure 7: A Pickover's biomorph

General quadratic map basins

This kind of fractals can be described as the boundary between the convergence and divergence domains of a family of recursive complex functions whose real (x) and imaginary (y) components are defined as follows [10]:

$$Xn+1=a+bxn+cxn2+dxnyn+ey+fyn2$$

$$yn+1=g+hxn+ixn2+jxnyn+kyn+lyn2$$

where a through l are real numbers that define each member of the family.

The following APL2 function generates one of these maps, that looks like a smiling mouth, in the form of a bit map. (see figure 8):

```
[0]   Z←LIPS N;a;A;A1;A2;C;W;F1;F2;X;Y
[1]   ⍝ THRESHOLD, RESOLUTION, INITIAL COORDS., LENGTHS, C
[2]   A←1000000 300 300 ¯1.2714 ¯1.7664 3.1155 3.1155
[3]   a←0.1 ¯0.7 0.4 ¯0.3 ¯0.2 0.9 ¯0.4 ¯0.7 ¯0.4 ¯1 ¯0.7 ¯0.8
[4]   A1←A[2 3]⍴A2←0
[5]   W←(A[4]+0J1×A[5])+((⍳A[2])×A[6]÷A[2])∘.+0J1×(⍳A[3])×A[7]÷A[3]
[6]  L:F1←A[1]<|W×W
[7]   F2←(A1≠A2)∨F1
[8]   A1←(F2×A1)+(~F2)×A1+1
[9]   A2←A2+1
[10]  X←9○W
[11]  Y←11○W
[12]  C←a[1]+(X×a[2]+(a[3]×X)+a[4]×Y)+Y×a[5]+a[6]×Y
[13]  C←C+0J1×a[7]+(X×a[8]+(a[9]×X)+a[10]×Y)+Y×a[11]+a[12]×Y
[14]  W←(F2×W)+(~F2)×C
[15]  →(0<N←N-1)/L
[16]  C←(⎕AF 'BM'),(⌽,256 256 256 256⊤1078+⍴,A1),(4⍴0),⌽256 256 256 256
      ⊤1078
[17]  C←C,40 0 0 0,(⌽256 256 256 256⊤A[2]),(⌽256 256 256 256⊤A[3]),1 0
[18]  C←C,8 0,(24⍴0)
[19]  A1←((4×⌈(↑⍴A1)÷4),1↓⍴A1)↑A1
[20]  X←(76⍴189),(34⍴96),⌊(126+121×(0,⍳90)÷90),(247-121×(⍳51)÷51),4⍴255
[21]  Y←(76⍴208),(34⍴10),⌊(12+211×(0,⍳90)÷90),(223-211×(⍳51)÷51),4⍴255
[22]  F1←(76⍴255),(34⍴141),⌊(188+67×(0,⍳90)÷90),(255-67×(⍳51)÷51),4⍴255
[23]  C←C,(,X,Y,F1,[1.5]0),⌊,256||⊖⍉A1 ⍝ BGR
[24]  (⎕AF C)⎕FV 'LIPS.BMP'
```

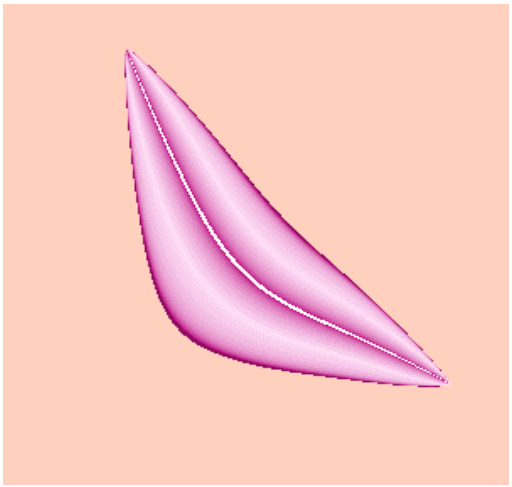

Figure 8: A fractal similar to a smiling mouth

2. Evolving Cellular Automata

Introduced by John Conway [11], the game of Life is a very simple cellular automaton that gives rise to extremely complicated behavior, and has been proved to be computationally complete, being able (in principle) to perform any computation which can be done by equivalent devices, such as digital computers, Turing machines or artificial neural networks.

The cellular automaton associated to the game of Life is defined thus:

- The grid is rectangular and potentially infinite.
- The set of neighbors to a point in the grid consists of the point itself plus the eight adjacent points in the eight main directions in the compass (Moore's neighborhood).
- Each finite automaton has two states: empty (also called dead, represented by a zero or a space character) and full (also called alive, represented by a one or a star symbol *). The set of states is thus represented by the two Boolean numbers {0,1} or the two characters ' *'.
- The transition function is defined by the following simple rules:

    - If the automaton associated to a cell is in the empty state, it goes into the full state if and only if the number of its neighbors in the full state is exactly three.
    - If the automaton associated to a cell is in the full state, it goes into the empty state if and only if the number of its neighbors in the full state is less than two or more than three.
    - In any other case, the automaton remains in the same state.

The following APL2 program executes the game of life on a finite rectangular grid.

```
[0]    A←Y LIFE N;X1;G;⎕IO
[1]    ⍝ The game of life
[2]    G←⎕IO←1
[3]    ⍎(0=⎕NC 'Y')/'Y←2=?8 8⍴2'
[4]    A←Y
[5]   L:→(∧/,A=0)/0
[6]    →(N≤G)/0
[7]    G←G+1
[8]    X1←0,0,[1](A,[1]0),0
[9]    X1←(1⌽X1)+(¯1⌽X1)+(1∘.×X1)+(¯1∘.×X1)+(1⌽1∘.×X1)+(¯1⌽1∘.×X1)+(1⌽¯1∘.×X1)+
       ¯1⌽¯1∘.×X1
[10]   X1←1 1↓¯1 ¯1↓X1
[11]   A←(X1=3)∨A∧X1∈2 3
[12]   →L
```

We define life-related cellular automata as the set of cellular automata that comply with the following rules: the grid is rectangular; the set of neighbors to a point in the grid consists of the point itself plus the eight adjacent points in the eight main directions in the compass (Moore's neighborhood); each finite automaton has two states, represented by the Boolean numbers {0,1}; the transition function of the finite automaton associated to every point is deterministic.

Life-related cellular automata differ in the transition functions of their finite automata. Since the input to each automaton is made of the states of its neighbors (which are nine, including itself, and may be ordered arbitrarily), each possible input n be represented by a nine-bit Boolean vector or, alternatively, by a number in the [0,511] interval. Each member in the set of life-related cellular automata may thus be represented by its associated transition function elements. This means that the number of possible different transition functions is 2512, or approximately 10154, an unimaginably large number.

We have programmed in APL2 a genetic algorithm to obtain Conway's game of Life by evolution from a random population of life-related cellular automata, taken arbitrarily from the full set of 10154 members. The grid has been restricted to an 8x8 square matrix. Evolution is fast, and reaches perfect target in a few tens of thousand steps or generations.

The algorithm can be described as follows:

1. Create 60 random life-related cellular automata.
2. Choose random initial conditions for the 64 automata in the 8x8 grid.
3. Compute the result of executing a step in Conway's game of Life with the chosen initial conditions, using the implementation given above. This results in an 8x8 Boolean matrix.
4. Compute the result of executing a step in each of the 60 life-related cellular automata with the chosen initial conditions. All the results are also 8x8 Boolean matrices.
5. Compare each of the results in step 4 with the result of step 3 and assign a fitness value to each of the 60 life-related cellular automata. The fitness value (an integer in the [0,64] interval) is the number of coincidences between the elements of the two 8x8 Boolean matrices.
6. Order the 60 life-related cellular automata in the order of their fitness values.
7. The ten automata with top fitness values are paired two-by-two and reproduce, each pair generating two new automata, which replace the ten automata with bottom fitness values. The reproduction algorithm uses the two standard tools in genetic algorithms: mutation and crossing-over at a random point (genetic recombination of the two 512 Boolean vectors that represent the transition function of the two parent automata).
8. Go to step 2.

Initially, the average fitness values of the 60 automata compute around 32, the number of coincidences expected from two 8x8 random Boolean matrices. As evolution proceeds, the average fitness values increase towards 64.

The APL2 language is so concise and appropriate for this application, that each step in the above description of the algorithm is implemented by a single instruction. Additional auxiliary functions are used to implement the random density generator of initial conditions and the execution of a life-related cellular automaton.

We have experimented with different mutation rates. In standard genetic algorithms, the mutation rate is usually quite low, around 0.5 percent. However, higher mutation rates are used when both parents have the same genotype, because in that case crossing-over does not have any effect. In our case, a higher mutation rate was found to accelerate significantly the speed of convergence.

3. Artificial Life

Finally, we have performed a simulation of artificial ants by means of the following APL2 function:

```
[0]   SIMUL;F_ANTH;XMAX;YMAX;FOOD;RANGE;ANTS;RW;IW;INFO;STEP
[1]   F_ANTH←100
[2]   (XMAX YMAX)←10 25
[3]   CREATE_FOOD
[4]   ANTS←ANT¨100⍴STEP←0
[5]   screen_mode 3
[6]   RW←window 1 1,(1+2×XMAX,YMAX),2 15 2
[7]   IW←window 1 55 10 20 2 12
[8]   open_windows
[9]  LOOP:
[10]  RANGE←(1+2×XMAX,YMAX)⍴' '
[11]  SHOW¨ANTS
[12]  RANGE[1+XMAX+FOOD[1];1+YMAX+FOOD[2]]←'F'
[13]  RANGE[1+XMAX;1+YMAX]←'A'
[14]  RW put_window_i⊖RANGE
[15]  INFO←('STEP:  ',⍕STEP)('FOOD:  ',⍕FOOD[3])
[16]  INFO←INFO,('KNOW:  ',⍕+/5⊃¨ANTS)('BRING: ',⍕+/6⊃¨ANTS)
[17]  IW put_window_i 20↑[2]⊃INFO
[18]  ANTS←MOVE¨ANTS
[19]  TELL¨(((0≠1⊃¨ANTS)∨0≠2⊃¨ANTS)∧5⊃¨ANTS)/⍳⍴ANTS
[20]  →LOOP,STEP←STEP+1
```

This artificial life system has the following properties:

- Things happen in a rectangular area of size 10x25.
- Ants belong to an anthill located in the center of the rectangular area.
- There is food somewhere in the territory.
- Ants go in search of food randomly.
- When an ant discovers food, it takes a piece to the anthill, rests for a time and goes back for more. It remembers the position of the food and goes there directly.
- If an ant which knows where food is, meets another one who doesn't, it tells its mate the location of the food.
- When the food source is depleted, another one appears in a different location, and ants start looking randomly again.

Ants in this experiment are not trying to simulate the behavior of real insects. For instance, they communicate directly, rather than at a distance, by means of pheromones.

4. Conclusions

The examples shown in this paper span many of the fields usually classified as complex systems: representation of fractal curves, cellular automata, genetic algorithms and artificial life.

APL2 has been found a very appropriate language to simulate these types of complex systems and experiment with them. Even when the execution speed is important, APL2 can still be used to write a prototype, which can later be translated to a faster language, such as C++.

References

[1] Mandelbrot, B.: The Fractal Geometry of Nature, W.H.Freeman and Company, New York, 1977.
[2] Alfonseca, M.; Ortega, A.: "A study of the representation of fractal curves by L systems and their equivalences", IBM Journal of Research and Development, Vol. 41:6, p. 727-736, Nov. 1997.
[3] Alfonseca, M.; Ortega, A.: "Representation of Fractal Curves by means of L Systems", APL Quote Quad (ACM SIGAPL), Vol. 26:4, p. 13-21, Jun. 1996. (Pres. APL 96, Lancaster, Jul. 1996).
[4] Lindenmayer, A.; Rozenberg, A.; Herman, G.: Developmental systems and languages. North-Holland/American Elsevier, Amsterdam, 1975
[5] Alfonseca, M.; Ortega, A.: "Using APL2 to Compute the Dimension of a Fractal Represented as a Grammar", APL Quote Quad (ACM SIGAPL), Vol. 30:4, p. 13-23, Jul. 2000. (Pres. APL Berlin 2000, Jul. 2000).
[6] Alfonseca, M.; Ortega, A.: Determination of fractal dimensions from equivalent L systems, IBM Jr. of Res. and Dev., Vol. 45:6, p. 797-805 Nov. 2001.
[7] Pickover, C.A. "Biomorphs: Computer Displays of Biological Forms Generated from Mathematical Feedback Loops". In Computer Graphics Forum, vol. 5, p. 313-316, 1986
[8] Pickover, C.A. "Computers, pattern, chaos, and beauty". New York: St. Marin's Press, 1990 (reprinted by New York: Dover Publications, 2001)
[9] Ortega, A.; de la Cruz, M.; Alfonseca, M.: Parametric 2-dimensional L Systems and recursive fractal images: Mandelbrot set, Julia sets and biomorphs, Computers and Graphics. Pergamon Elsevier Science Ltd. ISSN 0097-8493, (2002) vol. 26:1, pp. 143-149
[10] Sprott, J. C.; Pickover, C. A.: "Automatic generation of general quadratic map basins", Computers and Graphics. Elservier Science Ltd. (1995) vol. 19:2, pp. 309-313
[11] Berlekamp, E.R.; Conway, J.H.; Guy, R.K.: "Winning Ways for your Mathematical Plays", Academic Press, 1982.