On the practical implementation of L-systems

David Brebera

***Abstract*—** L-system, sometimes called Lindenmayer system is a formal way of writing an algorithm first introduced in 1968 by Hungarian biologist Aristide Lindenmayer (1925-1989), and is primarily designed as a tool to describe the growth of plants [Lin68]. Over time, however, the L system has become a tool used not only in biology, but also today, mainly in computer science (as an example for the study of context-free grammars) and mathematics (a tool used to describe and exploration of fractal curves). As a visualization tool L-system often uses turtle graphics (below), i.e. LOGO programming language.

***Keywords*—** algorithm, fractal, L-system, LEGO, LOGO, recursion, turtle graphics

I. L-Systems

THE Definition of L-system is often stated as follows [Pru 90]: L-system is an ordered triplet $L = \langle V, \omega, P \rangle$, where V is an alphabet, V^* set of words compiled from the alphabet V and V^+ the set of all non-empty words drawn from the alphabet V. In addition, $\omega \in \varepsilon^+$ is the axiom defining the initial state of the entire L-system. Finally $P \subset V \times V^*$ is a finite set of rewriting rules to rewriting $s \in V$ to $w \in V^*$, in abbreviated labeling as $a \rightarrow w$. For each symbol $s \in V$ that is not on the left side of some rewriting rule P is defined identity $s \rightarrow s$. These symbols are called constants or terminals.

This definition of L-system is slightly differs from the conventional definition of deterministic context-free grammar [Sip06] thinking grammar $G = \langle V, \Sigma, R, S \rangle$ arranged as a ordered quartet with only one initial symbol $S \in V$ (unlike L-system where initial axiom can be a whole word - although this can be in the L-system neatly fixed by adding one more rewrite rules $\omega_0 \rightarrow \omega$, $\omega_0 \in V$) and a set Σ disjoint with V as a finite set of terminals.

The simplest L-system can be, for example, often referred to following a set of rules (which Lindenmayer used as a rule of algae growth) and the axiom, rule $a \rightarrow ab$, $b \rightarrow a$. After a few steps we get *a, b, aba, abaab, abaababa, abaababaabaab* etc. Certainly it is interesting that the number of symbols in each generation is consistent with members of the Fibonacci sequence and the ratio of *a* to *b* in the limit corresponds to the golden ratio $\varphi = 1.618$ ... which occurs in many different forms in nature.

This work was supported by the project No. SGSFES_2015001 Ekonomický a sociální rozvoj v soukromém a veřejném sektoru (Economical and social development in private and public sector), financed from Czech Republic funds.

D. Brebera, Institute of Mathematics and Quantitative Methods, Faculty of Economics and Administration, University of Pardubice, Pardubice, Studentská 84, 532 10 Pardubice, Czech Republic (e-mail: David.Brebera@upce.cz).

II. Turtle graphics

Turtle Graphics is commonly used term for a simple vector graphic language interpreter in the XY plane. To the public it is often presented as being a turtle tail, which in its simplest implementation has just five graphic commands: *TurnLeft, TurnRight, PENUP, PenDown, Forward*. If the tail of the turtle is down (PenDown), then draws a line in the sand, when the tail is up, then after the command *Forward* turtle only moves, but it is not drawing. These methods were used in the print driver for plotter described below. For the purpose of L-systems often leads to a simple modification, the trio commands *PenUp, PenDown a Forward* is replaced by a pair of *MoveForward* (without drawing lines) and *DrawForward* (while drawing lines). Likewise, a pair of commands *TurnLeft TurnRight* are replaced with a single command TurnLeft with a positive argument for turning to the left (counter clockwise) and a negative argument to turn to the right (clockwise). The logic is the same as the standard measurement angle in the plane when the default angle $\varphi = 0$ is a shift in the positive direction of the X axis negative analogy argument commands *MoveForward* and *DrawForward* causes movement / drawing opposite direction.

III. L-system with brackets

L-system with brackets extends the standard L-system of branches by introducing two new symbols: "[" as the beginning of branches and "]" as the end of the branches. The symbol "[" then stores on the stack the current state of the system, typically the current coordinates (position of the turtle) and the angle of rotation. Symbol "]" analogically restores the system state to the point stored in the stack (these values from the stack are picked up, move the turtle to coordinates and rotates it in the saved, respectively. The symbols "[" and "]" have become so integral part of the L-systems, that if today we speak about L-systems we completely automatically understand just as L-systems with brackets. Such a system will enable to create different types of trees (not necessarily binary-only). Let us say, however, that any system with brackets can be rewritten without parentheses simply that instead of the symbol "]" inverted repeat steps carried out after the preceding "[", i.e. the inverse shift and rotation turtles and inverse

mathematical operations. This procedure has been used to draw some further mentioned fractals.

IV. Parametric L-system with brackets

Further expansion of L-systems is the introduction of the arguments on the individual symbols. Arguments in decimal form (or in general, any mathematical function) allow you to control the rendering algorithm, for example, by passing the length of the line or angle.

V. Selected L-systems and their graphical representation

Let us imagine now briefly some simple L-systems. For each, we will present its graphic (often fractal) form and write the algorithm in the system Malsys [Fis12]. All presented algorithms are also possible to draw on Lego plotter. For some we will also describe detail procedures for processing because the recursion is not supported by Lego (!) programming language.

VI. Levy C-fractal

Levy C-fractal is based on the axiom F, and one very frugal overwrite rule

$$F \rightarrow - F + + F$$

where the symbol + represents a turn to the left about 45 degrees symbol - clockwise rotation of 45 ° F symbol and move forward. Successively overwriting this chain then comprises the following sequence:

$$F - F + + F - - - F + F + - + - F + + F - - \text{ etc.}$$

However, since, as already mentioned, the programming language does not support recursion, such realization of L-system is impossible. If we replace doubles in the latter chain, a pair of symbols - - and + + we receive always one new same symbol with an angle of 90 ° (intermediate sequences - + + - cancel each other out), we can get a very similar record again with the axiom F and overwrite rule

$$F \rightarrow - F + F \;\; F + -.$$

Now broken down a few steps as follows: axiom F_0 and rewriting rules

$$\begin{aligned} F\_0 &\rightarrow - F\_1 + F\_1 \;\; F\_1 + F\_1 - \\ F\_1 &\rightarrow - F\_2 + F\_2 \;\; F\_2 + F\_2 - \\ &\vdots \\ F\_n &\rightarrow - F + F \; F + F - \end{aligned}$$

In the last, the n-th step, already do not call other nesting, but actually the first command Forward. The same trick, therefore multiple calls to the same rules, with only a slightly modified name (index added) was used also in the creation of additional L-systems - we will not repeat this process in all the upcoming listings.

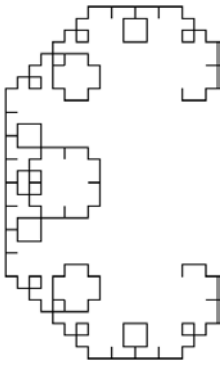

Fig. 1: Levy C-curve

```
lsystem Levy_C_curve_LEGO
{
   set symbols axiom = - F1 ;
   set iterations = 5;
   interpret F as DrawForward(16);
   interpret + as TurnLeft(90);
   interpret - as TurnLeft(-90);
   rewrite F1 to - F2 + F2 F2 + F2 - ;
   rewrite F2 to - F3 + F3 F3 + F3 - ;
   rewrite F3 to - F4 + F4 F4 + F4 - ;
   rewrite F4 to - F + F F + F - ;
}
```

Fig. 2: Levy C-curve algorithm

VII. H-fractal

H-fractal is the simplest example of a binary tree of parametric L-system with parentheses. Parameter is the length of the branch, then parentheses denote branching point. Again, a very simple example, however the implementation is impossible in Lego environment, since the entire system would need be programmed. To draw a tree was used the same principle as the C-Levy fractal, moreover, it was necessary to draw upon each branch return by the same route back, which replaces pick originally stored parameters on the stack.

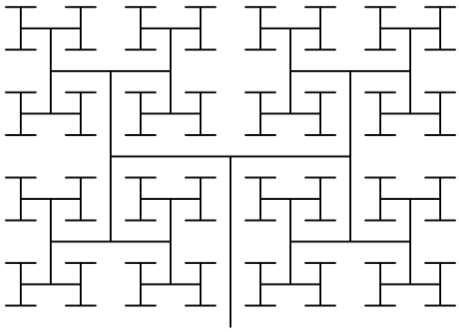

Fig. 3: the H-fractal

```
lsystem H_fractal
{
    set symbols axiom = + A(256) ;
    set iterations = 8;
    interpret F(x) as _
    DrawForward(256/(sqrt(2)^x));
    interpret + as TurnLeft(90);
    interpret - as TurnLeft(-90);
    interpret [ as StartBranch;
    interpret ] as EndBranch;
    rewrite A to F(currentIteration)
    [ - A] [ + A ] ;
}
```

Fig. 4: the H-fractal algorithm

VIII. Hilbert curve

The Hilbert curve, which tore down many prejudices about mathematical infinitely thin lines and their derivatives (as in the limit covers the entire plane dimension 2 and becomes flat) certainly deserves its place among L-systems deserves. Let's leave it without further comment and try it to draw the curve in hand. It's not as easy as it seems at first glance.

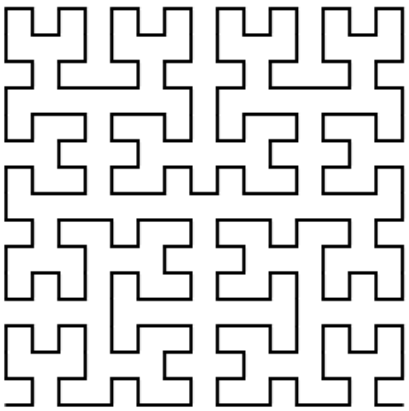

Fig. 5: the Hilbert curve

```
lsystem HilbertCurve
{
  set symbols axiom = L;
  set iterations = 6;
  interpret F as DrawForward(8);
  interpret + as TurnLeft(90);
  interpret - as TurnLeft(-90);
  rewrite L to + R F - L F L - F R +;
  rewrite R to - L F + R F R + F L -;
}
```

Fig. 6: the Hilbert curve algorithm

IX. The Dragon curve

Dragon curve is another famous fractal curve. Its origin is often demonstrated by means of folding strips of paper (still same direction - try as many times as you can) followed by unfolding with the preservation of right angles.

Fig. 7: paper folding (credit Wikimedia Commons)

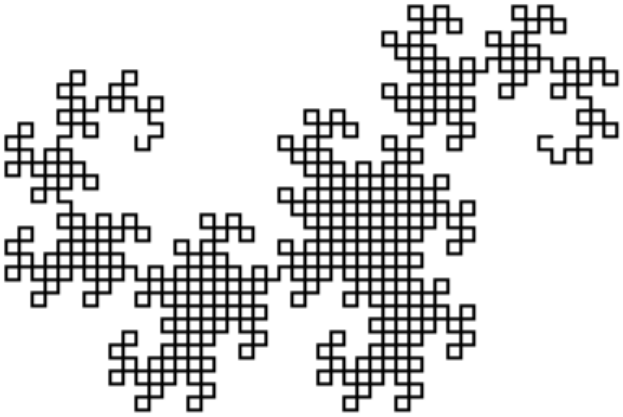

Fig. 8: Dragon curve in 1024 steps

```
lsystem DragonCurve
{
  set symbols axiom = - F X;
  set iterations = 10;
  interpret F as DrawForward(8);
  interpret + as TurnLeft(90);
  interpret - as TurnLeft(-90);
  rewrite X to X + Y F ;
  rewrite Y to F X - Y ;
}
```

Fig. 9: Dragon curve in 1024 steps

X. Recursion free algorithm

To draw a Dragon curve using rewriting rules and recursion is easy, but (as already mentioned) the recursion is impossible development environment Lego does not. It was therefore used other relatively interesting approach, where the whole curve is drawn in a single cycle using modular arithmetic when the rest of the division determines whether the left or right folding tape. We present pseudocode calculation.

At this point it opens an interesting moment to consider whether a similar procedure can also generate additional herein fractal curves, or whether each curve can be generally described recursive L-system converted to a mathematical function. It always had this rule only to find that the n-th step perform turn left or right (or to continue straight ahead).

```
for n = 1 to 1024
   forward 8
   m = 0
   repeat
      k = n / (2^m)
      m++
   until k%2 != 1
   if k%4 == 1 then
      left 90
   else
      left -90
   end if
next
```

Fig. 10: pseudocode for non recursive Dragon curve algorithm in 1024 steps

XI. LEGO MINDSTORMS

Lego Mindstorms robotic kit EV3 (2013) in sets Lego Mindstorms RCX (1999) and the Lego Mindstorms NXT (2009), currently the third-generation programmable robotic kits. It is offered in both the Home version (model 31313) and the Education version (model 45544 Core Set, Model 45560 Expansion Set). Each kit includes a programmable microcontroller (labeled simply as "brick"), formerly with proprietary OS, now with a full Linux OS. Programming is done via freeware LEGO EV3 software development kit and is perhaps somewhat unhappily done as Drag & Drop a jigsaw puzzle with only minimal use of classical programming techniques. For example, the system does not distinguish between local and global variables in the procedure call, can not pass parameters not implemented WHILE loop with condition at the start and hardly understandable reasons not support recursion. Especially the last mentioned may in connection with L-systems seem fatal - but even this is overcome difficulties. It should be noted that in the context of a very large community, there are many alternative versions of the firmware, some of which are fully compatible with major development environments (eg. LeJOS fully programmed in JAVA of middle Eclipse), but for demonstration purposes, it was decided to retention and use of the original LEGO firmware.

Plotter was developed and tested during May 2014 - it is a completely new unpublished model built without prior instruction. The construction of such a device is a very interesting experience, among other things, had to be solved for example. Calibrate the printhead calibration shift in X and Y to preserve the scale of 1: 1 in both directions and write low-level "printer driver" turtle graphics, which is subsequently used when printing fractals just described L-systems. In the next stage, there was a more complex experiments, eg. For printing Lissajous curves or vector models of 3D objects. The plan is to program and print the Mandelbrot set. It was interesting and monitoring public reaction to the functioning of the plotter, when many onlookers felt that the printer is not programmed, but draws only random patterns. Some other drawings are shown in the illustration at the end of this article.

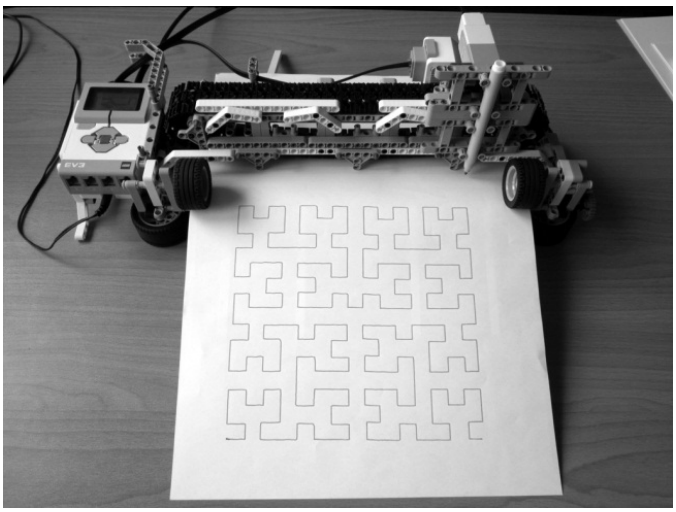

Fig. 11: The LEGO L-systems plotter

REFERENCES

[Coo71] Cook, Stephen (1971). "The complexity of theorem proving procedures". Proceedings of the Third Annual ACM Symposium on Theory of Computing. pp. 151–158.

[Fis12] Fišer, Marek (2012). L-systems online, Department of Software and Computer Science Education, MFF UK

[Fre04] Freitas, Robert A.; Ralph C. Merkle (2004). Kinematic Self-Replicating Machines. Georgetown, Texas: Landes Bioscience. ISBN 1-57059-690-5.

[Lin68] Lindenmayer, Aristid (1968). Mathematical models for cellular interactions in development. In: Journal of theoretical biology Parts I and II 18.3

[Pru90] Prusinkiewicz, Przemysław; Lindenmayer, Aristid (1990). The Algorithmic Beauty of Plants (The Virtual Laboratory). Springer-Verlag. ISBN 0-387-97297-8

[Sip06] Sipster, Michael (2006). Introduction To The Theory Of Computation, Second Edition, Thomson Course Technology, 2006, ISBN 0-534-95097-3G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

Mgr. David Brebera graduated in Mathematics in 1999 at Mathematics and Physics Faculty of Charles University. He was working as the senior actuary in Pojistovna Ceske sporitelny, a. s. (Insurance Company of the Czech Savings Bank), since 1999 to 2006. At the present he has been working at Faculty of Economics and Administration in Pardubice since 2005. He has been concentrated on informatics, statistics, actuarial science and management of financial risks.