Application of Parametric L-Systems to Generate the Figures of Two Series of Spherical Fullerenes

M. Vahidipour[1*], Gh. Vakili-Nezhaad[2]

1- Department of Computer Engineering, Faculty of Engineering, University of Kashan, Kashan, I. R. Iran
2- Department of Chemical Engineering, Faculty of Engineering, University of Kashan, Kashan, I. R. Iran

(*) Corresponding author: Vahidipour@kashanu.ac.ir

(Received: 09 Mar. 2010 and Accepted: 05 Jun. 2010)

Abstract:
In this work, two parametric L-Systems have been introduced to generate the figures of two series of nanostructures. These series consist of different spherical fullerenes with various number of carbon atoms. By specifying a parameter in each series a spherical fullerene can be determined. The members of the first series are C_{60}, C_{240}, C_{540}, C_{940}, ... which can be produced by applying parameter n=0, 1, 2, 3, ... on the first parametric L-system. In the same manner, the second parametric L-system can produce the figures of C_{80}, C_{180}, C_{320}, ... by applying parameter n=0, 1, 2, etc.

Keywords: *Fullerenes, L-System, Spherical Fullerenes*

1. INTRODUCTION

For the first time, application of L-systems for drawing of nanostructures was introduced in ref. [1]. A feed forward D0L-System can generate C_{20} and C_{60} figures (Figure 1). The only difference between these two L-Systems is the value of angles. There are 12 pentagons in every closed fullerene [2] that is created by bounds between carbon atoms. A *branch* or *link* connects two pentagons in C_{60} if we assume that every pentagon lies on a plane. Therefore, there is an angle between the pentagon plane and connected branch. The value of this angle in C_{60} is 31.7175 [1]. This scenario can be repeated for C_{20} (and maybe in all of spherical fullerenes). In C_{20}, the angle is 52.283 whereas in C_{80}, is 37.3760 [1].

Now, we are going to extend this L-system to other spherical fullerenes without knowing the value of more angles. In ref. [2], two series of spherical fullerenes are introduced. In each series that includes some spherical fullerenes, an arithmetic progression predicts the number of carbon atoms. The number of carbon atoms in the first series is started by 20 follows by 80, 180, 320… ($20\ n^2$). In the second series, the arithmetic progression predicts the number of carbon atoms as 60, 240, 540, …($60n^2$).

2. L-SYSTEMS

The idea of L-systems is to generate a string of symbols using some of productions, and to interpret this string as a sequence of commands which controls a "turtle". Suitable generalizations of the notions of the L-system and of a turtle have been introduced in [3, 4]. The resulting mathematical model can be used to create a variety of (finite approximations of) fractal curves, ranging from Koch curves, classic space-filling curves, to relatively realistic-looking pictures of plants and trees. All these pictures are defined in a uniform and compact way.

Using the central concept of L-Systems, rewriting, a complex object can be defined by successive replacing simple initial object. Lindenmayer introduced a new type of string-rewriting, subsequently named L-Systems, in 1968 [3]. This idea can be applied to define and drawing pictures of some complicated figures, such as trees, planet organs, cellular layers, fractal shapes and so on [4]. L-System uses production rules for its rewriting as the same as Chomsky grammars [5], but there is an essential difference between them. Productions are applied sequentially in Chomsky grammars, whereas in L-systems they are applied in parallel.

In other words, L-System can formulate complicated relations among simple initial objects by some simple rewriting rules that are applied simultaneously on a string and produced new and next string. This means that all letters in a given word simultaneously are replaced according to productions. But, how do the figures generate by L-Systems? Assume that for each letter in produced string by L-Systems there is a simple initial object such as line, curve, rotation …. A graphical interpreter maps between two spaces, alphabetic space of string and 3D space of figures.

Based on the formal properties of rewriting systems, deterministic Context-free L-Systems (called D0L-System) were created [5, 6]. Formal definitions describing D0L-systems and their operation are given below. Interested reader may refer to the works of Herman and Rozenberg [5] and Roaenberg and Saloma [7].

Let Σ denotes an alphabet, Σ^* the set of all words over Σ, and Σ^+ the set of all nonempty words over Σ. A 0L-system is an ordered triplet $G = \langle \Sigma, \omega, \rho \rangle$ where Σ is the alphabet of the system, $\omega \in \Sigma^*$ is a nonempty word called the *axiom* and $\rho \subset \Sigma \times \Sigma^*$ is a finite set of productions. We write $\alpha \to X$, if a pair (α, X) is a production. The letter α stands for the predecessor and the word X denotes the production successor. A 0L-system is deterministic (D0L-systems) *iff* for each $\alpha \in \Sigma$ there is exactly one $X \in \Sigma^*$ such that $\alpha \to X$.

Let $\mu = \alpha_1 \alpha_2 \cdots \alpha_m$ be an arbitrary word over Σ. The word $V = X_1 X_2 \cdots X_m \in \Sigma^*$ is directly derived from (or generated by) μ and write $\mu \Rightarrow V$, *iff* $\alpha_i \to X_i$ for all $i = 1, 2, \cdots, m$. A word V is generated by G in a derivation of length n if there exists a sequence of words $\mu_0, \mu_1, \cdots, \mu_n$ such that $\mu_0 = \omega$ and $\mu_n = V$ as well as $\mu_0 \Rightarrow \mu_1 \Rightarrow \cdots \Rightarrow \mu_n$.

Now we are going to the mapping procedure of the string to the picture. A (graphic) interpretation function $I : \Sigma^* \to R^3$ is mapping the set of strings over the alphabet Σ into the set of pictures. There are some functions that can be used [6, 7, 8] but Prusinkiewicz focused on an interpretation based on a LOGO-style turtle [9] and presented more examples of fractals and plant-like structures modeled using L-systems [10, 11]. The *sate* of turtle is basic idea of turtle interpretation. The *state* is defined as *Position* and *orientation*. The turtle's position is represented by a triplet (*x, y, z*) in the Cartesian coordinates. The current *orientation* of the turtle in space is represented by three perpendicular unit vectors. These vectors $\vec{H}, \vec{L}, \vec{U}$ indicating the turtle's *heading*, the direction to the *left*, and the direction *up* [7]. These vectors satisfy the equation $\vec{H} \times \vec{L} = \vec{U}$ .See figure 2.

Turtle read subsequently letters from string; L-Systems generate this string by applying productions. Each letter is a specific command that controls turtle position and orientation in space. Some of these symbols and commands are listed in table-1. Some letters change position of turtle and some of them make change in turtle orientation. String with brackets is introduced to delimit a branch. When the turtle finds the "[" in string, the current state of the turtle is pushed into a stack. Current state of the turtle is popped from the stack if turtle find "]" in string.

The *edge rewriting* and *node rewriting* are two modes of operation for L-systems with turtle interpretation, using terminology borrowed from graph grammars [12, 13, 14]. In the case of edge rewriting, productions substitute figures for polygon edges, while in node rewriting, productions operate on polygon vertices. Both approaches rely on capturing the recursive structure of figures and relating it to a tiling of a plane [4].

The recursive nature of the L-system rules leads to self-similarity and thereby fractal-like forms which are easily described with an L-system. Plant models and natural-looking organic forms are similarly easy to define by increasing the recursion level.

3. PARAMETRIC L-SYSTEMS

L-systems are now commonly known as *parametric* L-systems, defined as a ordered quadruplet [3] **G** = $\{V, S, \omega, P\}$, where

- ***V*** (the *alphabet*) is a set of symbols containing elements that can be replaced (*variables*)
- ***S*** is the *set of formal parameters*
- **ω** (*start*, *axiom* or *initiator*) is a string of symbols from ***V*** defining the initial state of the system
- ***P*** is a set of *production rules* or *productions* defining the way variables can be replaced with combinations of constants and other variables. A production consists of two strings - the *predecessor* and the *successor*.

The rules of the L-system grammar are applied iteratively starting from the initial state. In the rewriting systems, there is a parametric word that contains *modules* and their *actual parameters*. In the other hand, there are some parametric rules that contain variables and their formal parameters. As many rules as possible are applied simultaneously, per iteration; this is the distinguishing feature between an L-system and the formal language generated by a grammar.

The symbols: and → are used to separate the three components of a production: the predecessor, the *condition* and the successor. For example, a production with predecessor $A(t)$, condition $t > 5$ and successor $B(t + 1)CD(t \wedge 0.5, t - 2)$ is written as

$$A(t) : t > 5 \rightarrow B(t + 1)CD(t \wedge 0.5, t - 2).$$

A *Derivation* production *matches* a module in a parametric word if the following conditions are met:

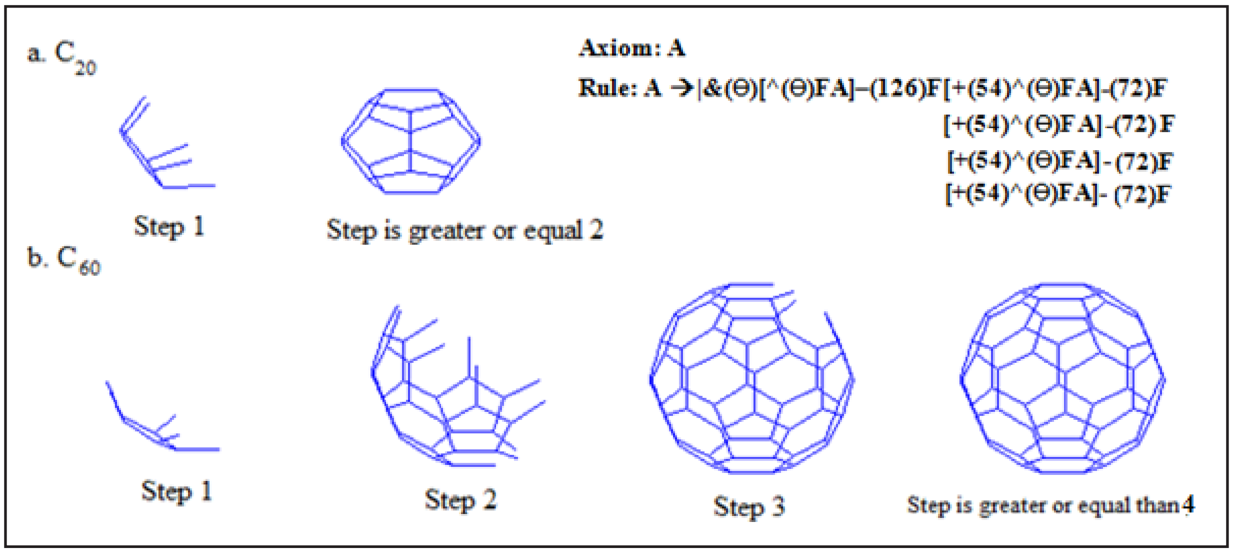

a. C20
Step 1
Step is greater or equal 2
Axiom: A
Rule: A →|&(Θ)[^(Θ)FA]–(126)F[+(54)^(Θ)FA]-(72)F
[+(54)^(Θ)FA]-(72)F
[+(54)^(Θ)FA]-(72)F
[+(54)^(Θ)FA]- (72)F
b. C60
Step 1
Step 2
Step 3
Step is greater or equal than 4

Figure 1: *Figures of C_{20} and C_{60} have been sketched in [13] by the shown L-system. If θ=52.283, The L-System can generate the figure of C_{20} after two derivation steps. After 4 steps, the L-System can generate the figure of C_{60} if θ=31.7175.*

Table 1: *The symbols and commands to controls turtle in space*

$F(\sigma)$	Move forward a step of length $\sigma > 0$. The position of the turtle changes to (x', y', z'), where $x' = x + \sigma_{\vec{H}x}$, $y' = y + \sigma_{\vec{H}y}$ and $z' = z + \sigma_{\vec{H}z}$. A line segment is drawn between points (x, y, z) and (x', y', z').
$f(\sigma)$	Move forward a step of length $\sigma > 0$ without drawing a line.
$+(\alpha)$	Rotate around $\vec{U}$ by an angle of α degrees. If α is positive, the turtle is turned to the left and if α is negative, the turn is to the right. If α is negative, the symbol '–' can be used also.
$\&(\alpha)$	Rotate around $\vec{L}$ by an angle of α degrees. If α is positive, the turtle is pitched down and if α is negative, the turtle is pitched up. The pitch up rotation can be represented by $\wedge(\alpha)$.
$/(\alpha)$	Rotate around $\vec{\mathrm{H}}$ by an angle of α degrees. If α is positive, the turtle is rolled to the right and if α is negative, it is rolled to the left. The roll left can be represented by '\' symbol.
|	Turn around $\vec{U}$ by an angle 180.
[	Push the current state of the turtle onto stack. The information saved on the stack contains the turtle's position and orientation, and possibly other attributes such as the color and width of lines being drawn.
]	Pop a state from the stack and make it the current state of the turtle. No line is drawn, although in general the position of the turtle changes.

- the letter in the module and the letter in the production predecessor are the same,
- the number of actual parameters in the module is equal to the number of constants in the production predecessor, and
- The condition evaluates to *true* if the actual parameter values are substituted for the formal parameters in the production.

A matching production can be *applied* to the module, creating a string of modules specified by the production successor. The actual parameter values are substituted for the formal parameters according to their positions. For example, the above production matches a module *A*(9), since the letter *A* in the module is the same as in the production predecessor. There is one actual parameter in the module *A*(9) and one formal parameter in the predecessor *A*(*t*), and the logical expression *t*>5 is true for *t* = 9. The result of the application of this production is a parametric word *B*(10)*CD*(3, 7).

4. FULLERENES AND L_SYSTEMS

Now, we are going to extend the presented L-system in ref. 1 for drawing figures of two series of nanostructures introduced in ref. 2. In each series that includes some spherical fullerenes, an arithmetic progression predicts the number of carbon atoms. The number of carbon atoms in the first series is started by 20 follows by 80, 180, 320… ($20n^2$). In the second series, the arithmetic

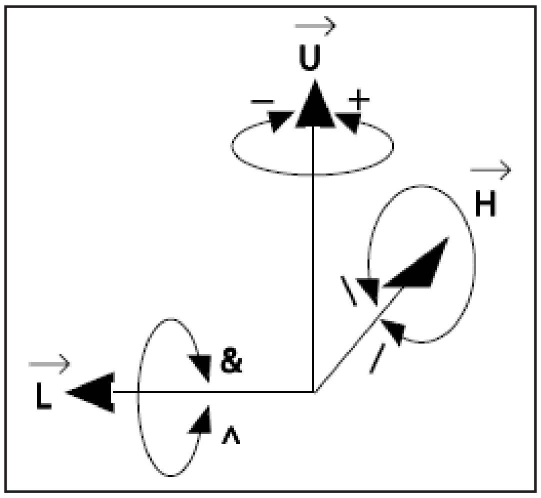


***Figure 2:** Controlling turtle in three dimensions*

progression predicts the number of carbon atoms as 60, 240, 540, …($60n^2$).

The whole story is determination of the distance between the nearest neighbor pentagons (See figure 3). In C_{60}, by connecting three pentagons, a hexagon is created (figure 3-a). If these pentagons are far from each other, a larger hexagon will be produced (figures 3-b and 3-c). This large hexagon must be partitioned into some smaller and regular hexagons to generate the figures of successive spherical fullerenes in this series.

The L-system in figure 4 generates the series which starts with C_{60}. By setting parameter n=0, figure C_{60} is produced, n=1 the figure of C_{240} is produced and so on. It must be mentioned that there is special symbol of *C* in the shown L-System in figure 4 that can save the location of the turtle. Finally by using these locations we can connect the most nearest points. This fact is shown in figure 5 by red lines. This is due to avoid the trial and error procedure for obtaining the right angles. Further works are needed to refine the obtained figures of fullerenes which is the subject of our future work.

By similar arguments we can obtain the figures of C_{80}, C_{180}, C_{320}, … using L-System in figure 6. Results are given in figure 7.

5. CONCLUSIONS

In this work, the new and powerful tool of L-Systems has been applied for generating the figures of some spherical fullerenes. A computer program has been

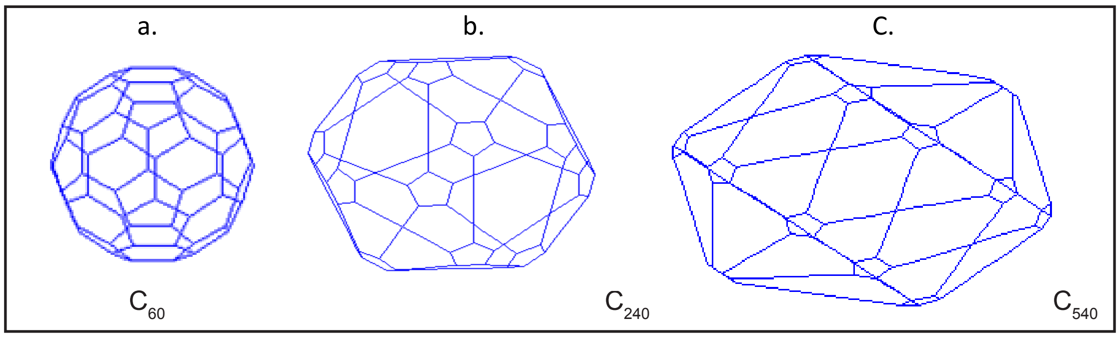


***Figure 3:** If the distance between the nearest neighbor pentagons is increased, the larger hexagon is formed that must be partitioned into smaller and regular hexagons.*

```
Axiom: A
Rule:      A → | & (31.3731) [ ^ (31.3731) FB(n)A ] + (-126) F [+ (54) ^ (31.3731) FB(n)A]
               + ( -72) F [+ (54) ^ (31.3731) FB(n)A]  + ( -72) F  [+ (54) ^ (31.3731) FB(n) A]
               + ( -72) F [+ (54) ^ (31.3731) FB(n)A]  + ( -72) F

     B (t): t>0 → (60)+ FC(60)-FC[B(t-1)](60)-F(60)+F B(t-1)
```

***Figure 4:** This parametric L-system can generate the figures of series spherical fullerenes which start with C_{60} follows by C_{240}, C_{540}, C_{960}, etc.*

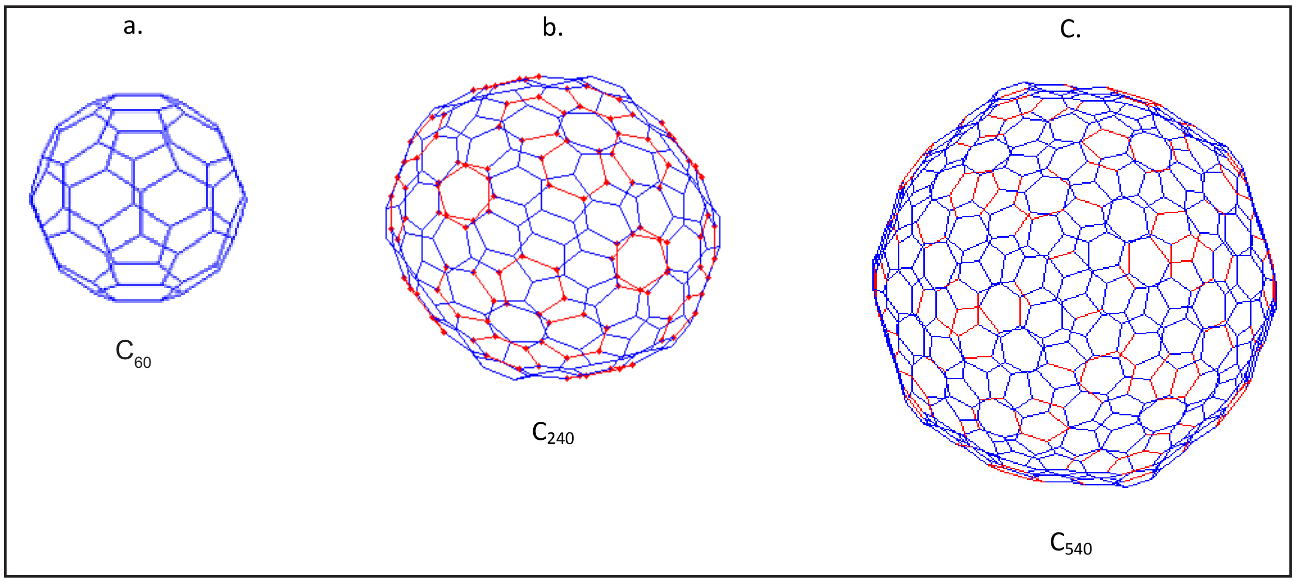
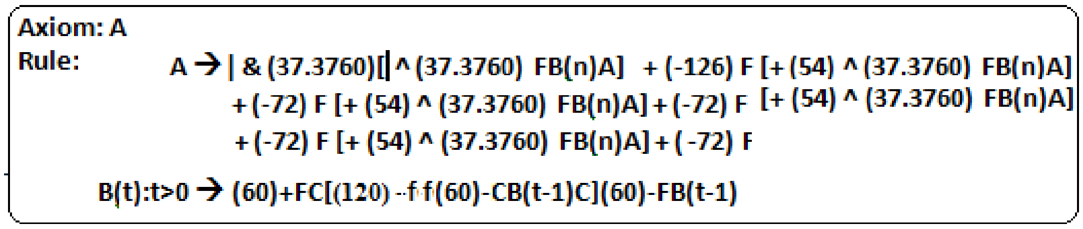


Figure 5: *By setting the parameter n=0,1,2 in the L-system in figure 3, the figure of C_{60},C_{240} and C_{540} is obtained consecutively*



```
Axiom: A
Rule:        A → | & (37.3760)[| ^ (37.3760)  FB(n)A]  + (-126) F [+ (54) ^ (37.3760)  FB(n)A]
                 + (-72) F [+ (54) ^ (37.3760)  FB(n)A] + (-72) F  [+ (54) ^ (37.3760)  FB(n)A]
                 + (-72) F [+ (54) ^ (37.3760)  FB(n)A] + ( -72) F

      B(t):t>0 → (60)+FC[(120) --f f(60)-CB(t-1)C](60)-FB(t-1)
```

Figure 6: *This parametric L-system can generate the figures of series spherical fullerenes which starts with C_{80} follows by C_{180}, C_{320}, etc.*

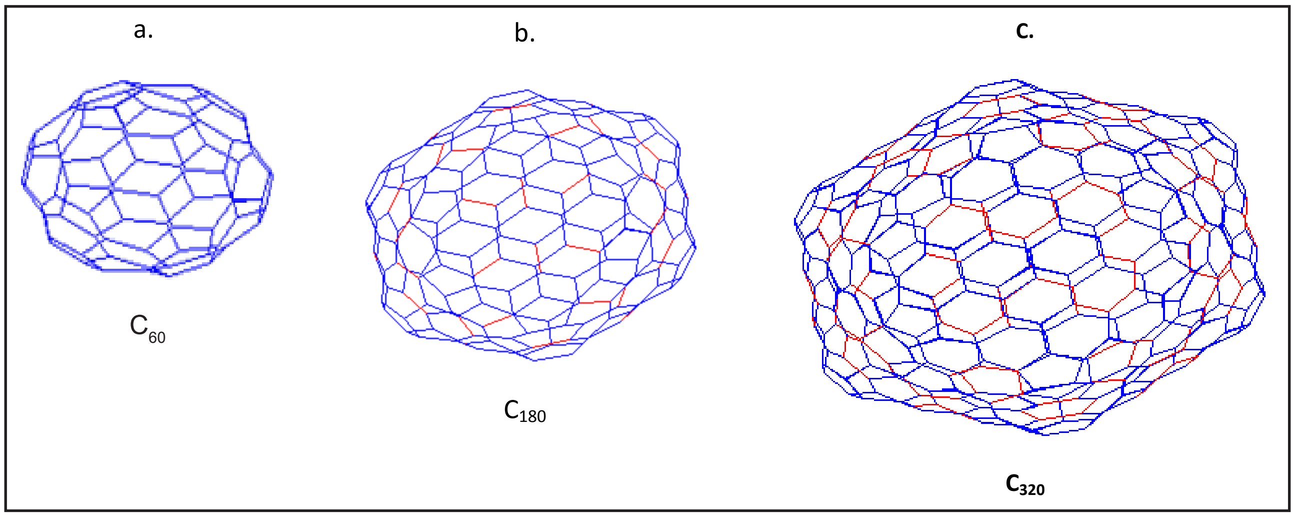


Figure 7: *By setting the parameter n=0,1,2 in the L-system in figure 5, the figure of C_{80},C_{180} and C_{320} is obtained consecutively*

written to sketch the figures of the fullerenes with the general formulas of $C20n^2$ and $C60n^2$. In other words, two grammatical formulas are proposed for drawing the figures of the above mentioned fullerenes without any detailed information.

REFERENCES

1. M. Vahidipour, H. Sabaghian-Bidgoli and G. R. Vakili-Nezhaad, Generating of the Figures of Some Fullerenes by Using L-Systems, Croatia Chemica Acta, Vol 81, 2008 (Accepted for publication).
2. H. Sabaghian-Bidgoli, G. R. Vakili-Nezhaad and M. Vahidipour, Prediction of the Number of Carbon Atoms in Various Nanostructures by Using Geometrical Approach, Croatia Chemica Acta, Vol 81, 2008 (Accepted for publication).
3. A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 1968, vol 18, pp. 280–315.
4. P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*. Springer-Verlag, New York, 1990. With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer .
5. G. T. Herman and G. Rozenberg. *Developmental systems and languages.* North-Holland, Amsterdam, 1975.
6. A. Salomaa. *Formal languages.* Academic Press, New York, 1973.
7. G. Rozenberg and A. Salomaa. *The mathematical theory of Lsystems.* Academic Press, New York, 1980.
8. A. L. Szilard and R. E. Quinton. An interpretation for DOL systems by computer graphics. *The Science Terrapin*, 1979, vol 4, pp. 8–13.
9. H. Abelson and A. A. diSessa. *Turtle geometry*. M.I.T. Press, Cambridge, 1982.
10. P. Prusinkiewicz. Graphical applications of L-systems. In *Proceedings of Graphics Interface '86 — Vision Interface '86*, 1986, pp. 247–253.
11. P. Prusinkiewicz. Applications of L-systems to computer imagery. In H. Ehrig, M. Nagl, A. Rosenfeld, and G. Rozenberg, editors, *Graph grammars and their application to computer science; Third International Workshop*, Lecture Notes in Computer Science 291, Springer-Verlag, Berlin, 1987, pp.534–548.
12. A. Lindenmayer. An introduction to parallel map generating systems. In H. Ehrig, M. Nagl, A. Rosenfeld, and G. Rozenberg, editors, *Graph grammars and their application to computer science; Third International Workshop*, Lecture Notes in Computer Science 291, Springer-Verlag, Berlin, 1987, pp. 27–40.
13. A. Habel and H.-J. Kreowski. On context-free graph languages generated by edge replacement. In H. Ehrig, M. Nagl, and G. Rozenberg, editors, *Graph grammars and their application to computer science; Second International Workshop*, Lecture Notes in Computer Science 153, Springer-Verlag, Berlin, 1983, pp. 143–158.
14. A . Habel and H.-J. Kreowski. May we introduce to you: Hyperedge replacement. In H. Ehrig, M. Nagl, G. Rozenberg, and A. Posenfeld, editors, *Graph grammars and their application to computer science; Third International Workshop*, Lecture Notes in Computer Science 291, Springer-Verlag, Berlin, 1987, pp. 15–26.