Graphical Modeling of Higher Plants Using P Systems

Alvaro Romero-Jiménez, Miguel A. Gutiérrez-Naranjo, and Mario J. Pérez-Jiménez

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla, Spain
Alvaro.Romero@cs.us.es, magutier@us.es, marper@us.es

Abstract. L systems have been widely used to model and graphically represent the growth of higher plants [20]. In this paper we continue developing the framework introduced in [21], which make use of the topology of membrane structures to model the morphology of branching structures.

1 Introduction

The growth of plants, considered as a function of time, have attracted the attention of scientific community for a long time. Features such as the bilateral symmetry of leaves, the central symmetry of flowers and, more recently, the study of self-similarity and fractal structure have been matter of study for computer scientists, mathematicians, and life scientists among others.

In 1968, Aristid Lindenmayer presented a theoretical framework for studying the development of simple multicellular organisms. The devices introduced in this framework are known as *parallel rewriting systems* or *L systems*.

L systems were introduced for modeling multicellular organisms in terms of division, growth, and death of individual cells [10,11]. These organisms are treated as an assembly of discrete units, which represent the individual cells. These sys-tems must be considered as dynamic models, which means that the form of the organism is the result of development along time. This development is described in terms of *production rules*, which are applied in parallel and are intended to capture the simultaneous progress of time in all parts of the growing organisms.

Several years later, the range of applications of L systems were extended to higher plants and complex branching structures [3,4]. In the first approach, the essence of development of less complex organisms is the replacement of individual cells by sets of cells, according to the production rules of the system. On the other hand, the units of information in L systems modeling higher plants represent complex structures, such as branches or leaves, instead of individual cells. These structures are replaced by other ones using the production rules.

In [5,6] a first approach for using P systems to simulate the growth and development of living plants is presented. This approach mixes L systems and

P systems, being in fact an L system "factorized" into several units, which are then computed in the compartments delimited by the membranes of the P system.

L systems use strings as data structures, which fits in a natural way in sequential structures such as microorganisms, or linear structure of fractals such as the Koch curve [8,9]. Nonetheless, the visual interpretation of strings of symbols as branching structures needs to add memory pointers in order to remember the location and orientation in which the branches were developed. These memory facilities are the key for developing several branches from the same point.

The topology of P systems is inherently a branching structure based on the inclusion relationship. This feature allowed us to present a framework for modeling the topology of living plants, without the necessity of considering memory pointers [21]. We think our approach is closer to reality than L systems, in the sense that we do not make "rewriting" over the membrane structure, but instead we use evolution rules to expand it. This is inspired by the fact that mature structures in higher plants, such as trunks and branches, keep their morphology along time. They change only in length and width, and the growth of new structures (leaves, flowers, new branches, and so on) is started only from specific points, already present.

In this paper we continue developing the framework introduced in [21], providing two means (randomness and non-determinism) for modeling the individual variations among different specimens of a same species that occur in nature.

The paper is organized as follows: first L systems and the usual way to visualize them are recalled in Sections 2 and 3. In Section 4, the variant of P systems used in this paper, a restricted version of P systems with membrane creation, is presented. Section 5 is devoted to the graphical visualization of the configurations of these P systems, whereas we discuss in Section 6 the differences between the representations obtained when stochastic and non-deterministic P systems are considered. Finally, conclusions and lines for future research are presented.

2 L Systems

The key idea of L systems for formalizing the development of plants is that of rewriting. This is a technique for defining complex objects by successively replacing parts of a simple initial object by using *production* rules.

The first formal definition of rewriting systems operating on strings of symbols was proposed by Thue at the beginning of the twentieth century (see [22]), but rewriting systems started to be widely considered after Chomsky's work on formal grammars [2], where the concept of rewriting is used to describe natural languages.

The essential difference of L systems with respect to Chomsky grammars lies in the method of applying the production rules. In Chomsky grammars, production rules are applied sequentially, whereas in L systems they are applied in parallel: in a derivation step all symbols of the string are rewritten.

The simplest class of L systems are the deterministic and context-free ones. Let V denote an alphabet, V^* the set of all words over V, and V^+ the set of all nonempty words over V. A *string 0L system* is an ordered triplet $G = \langle V, \omega, P \rangle$ where V is the *alphabet* of the system, $\omega \in V^+$ is a nonempty word called the *axiom* and $P \subset V \times V^*$ is a finite set of *production rules*. A production rule (a, v) is written as $a \to v$. The letter a and the word v are called the predecessor and the successor of this production rule, respectively. It is assumed that for any letter $a \in V$, there is at least one word $v \in V^*$ such that $a \to v \in P$. A 0L system is *deterministic* (noted D0L system) if and only if for each $a \in V$ there is exactly one $v \in V^*$ such that $a \to v \in P$.

Let $\mu = a_1 \dots a_m$ be an arbitrary word over V. The word $\rho = \phi_1 \dots \phi_m$, with $\phi_1 \dots \phi_m \in V^*$, is directly derived from (or generated by) μ, and we denote it by $\mu \Rightarrow \rho$, if and only if $a_i \to \phi_i \in P$ for all $i \in \{1, \dots, m\}$. Starting with the axiom $\omega \in V^+$, a sequence of strings $\mu_0 = \omega, \mu_1, \mu_2, \dots$ is generated recursively, where $\mu_i \Rightarrow \mu_{i+1}$, that is, the string μ_{i+1} is obtained from the preceding string μ_i by replacing *simultaneously* every symbol in μ_i according to production rules of the system.

3 Graphical Representation of L Systems

Originally, L systems were conceived for the study of multicellular organisms and the neighborhood relations between their different cells. After the incorporation of geometric features, L systems became appropriate for computer graphics representation that allows visualizations of these multicellular organism and their developmental processes. The first steps in this line can be found in the eighties' literature [17,23], but the most popular graphical interface for L systems was introduced by P. Prusinkiewicz [18,19] based on the previous Papert's concept of *turtle graphics* [14]. In an informal description, we can consider a turtle standing on a sheet of paper facing a given direction. The tail of the turtle is full of ink and it traces a line on the sheet when the turtle moves. The turtle obeys several commands: move forward by a fixed length l drawing or not the corresponding segment; turn left or right by a fixed angle δ.

More formally, a *state* of the turtle is defined as a triplet (x, y, α), where (x, y) represent the Cartesian coordinates of the turtle's position and the angle α represent the direction in which the turtle is facing. Given a step size l and an angle increment δ, the turtle responds to the following commands:

- F: the state of the turtle changes from (x, y, α) to $(x + l\,cos\alpha, y + l\,sen\alpha, \alpha)$. A line segment between (x, y) and $(x + l\,cos\alpha, y + l\,sen\alpha)$ is drawn.
- f: analogous to the previous command, the state of the turtle changes from (x, y, α) to $(x + l\,cos\alpha, y + l\,sen\alpha, \alpha)$. The segment is not drawn.
- $+$: the state of the turtle changes from (x, y, α) to $(x, y, \alpha + \delta)$.
- $-$: the state of the turtle changes from (x, y, α) to $(x, y, \alpha - \delta)$.

This method has been profusely used to interpret strings. For example, the representation of the string $F + F - -F + F$ with initial state (0, 0, 0), $l = 2$ cm and $\delta = 60$ degrees is depicted in Figure 1.

Fig. 1. The string $F + F - -F + F$

According to these rules, the turtle interprets a character string as a sequence of line segments. Note that different strings can lead to the same graphical representation.

If we want to model tree-like shapes and branching structures then new features have to be added. For that, an extension of turtle interpretation to strings with brackets is considered. Two new symbols are introduced to delimit a branch: the symbols "[" and "]". The *turtle interpretation* of the symbols is the following, when [is read, the turtle should remember its current direction and position. Then the branch can be drawn by the usual interpretation. Termination of the branch is marked by]. The turtle must then return to the location of the branch point, which it remembers. The formal interpretation of the symbols is the following.

- [: push the current state of the turtle onto a push-down 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.

For example, the representation of the string $F[+F[+F[+F][F]][F[F][-F]]][F[F[+F][F]][-F[+F][F]]]$ with initial state (0, 0, 90), $l = 2$ cm and $\delta = 22.5$ degrees is shown in Figure 2.

4 P Systems with Membrane Creation

Membrane computing is a branch of natural computing which abstracts from the structure and the functioning of the living cell. In the basic model, membrane systems (also frequently called P systems) are distributed parallel computing devices, processing multisets of symbol-objects, synchronously, in the compartments defined by a cell-like membrane structure[1].

[1] A detailed description of P systems can be found in [15] and updated information in [24].

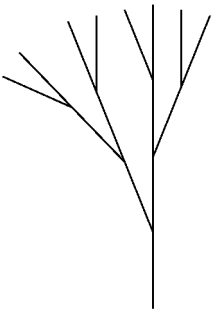

Fig. 2. The string $F[+F[+F[+F][F]][F[F][-F]]][F[F[+F][F]][-F[+F][F]]]$

In this paper we will consider P systems which make use of membrane creation rules, which was first introduced in [7,12]. However, our needs are far simpler than what the models found in the literature provide. This is the reason why we introduce the new variant of *restricted P systems with membrane creation*.

A restricted P system with membrane creation is a tuple $\Pi = (O, \mu, w_1, \ldots, w_m, R)$ where:

1. O is the alphabet of *objects*.
2. μ is the initial *membrane structure*, consisting of a hierarchical structure of m membranes (all of them with the same label; for the sake of simplicity we omit the label).
3. $w_1, \ldots, w_m$ are the multisets of objects initially placed in the m regions delimited by the membranes of μ.
4. R is a finite set of *evolution rules* associated with every membrane, which can be of the two following kinds:
   (a) $a \to v$, where $a \in O$ and v is a multiset over O. This rule replaces an object a present in a membrane of μ by the multiset of objects v.
   (b) $a \to [v]$, where $a \in O$ and v is a multiset over O. This rule replaces an object a present in a membrane of μ by a new membrane with the same label and containing the multiset of objects v.

A membrane structure (extending the membrane structure μ) together with the objects contained in the regions defined by its membranes constitute a configuration of the system. A computation step is performed applying to a configuration the evolution rules of the system in the usual way within the framework of membrane computing, that is, in a non-deterministic maximal parallel way; a rule in a region is applied if and only if the object occurring on its left–hand side is available in that region; this object is then consumed and the objects indicated in the right–hand side of the rule are created inside the membrane. The rules are applied in all the membranes simultaneously, and all the objects in them that can trigger a rule must do it. When there are several possibilities to choose the evolution rules to apply, non-determinism takes place.

5 Graphical Representation of Restricted P Systems with Membrane Creation

In this section we show how to use, through a suitable graphical representation, restricted P systems with membrane creation to model branching structures.[2] The key point of the representation relies on the fact that a membrane structure is a *rooted tree of membranes*, whose root is the skin membrane and whose leaves are the elementary membranes. It seems therefore a perfect frame to encode the branching structure.

Once we have a membrane structure establishing the topology of the object we want to model, we will follow a variant of the turtle interpretation of L systems. Let us suppose that the alphabet O of objects contains the objects $F, +$ and $-$ and let us fix the length l and the angle δ.

A simple model to graphically represent a membrane structure is to make a depth-first search of it, drawing, for each membrane containing the object F, a segment of length $m \times l$, where m is the multiplicity of F. If the number of copies of F in a membrane increases along the computation, the graphical interpretation is that the corresponding branch is lengthening. This segment is drawn rotated with respect to the segment corresponding to the parent membrane with an angle of $n \times \delta$, where n is the multiplicity of objects "+" minus the multiplicity of objects "−" in the membrane. That is, each object "+" means that the rotation angle is increased by δ whereas each object "−" means that it is decreased by δ.

In the real world, the branches of a plant not only lengthen but also widen themselves. Thus, we also fix the width w and use a new symbol W whose multiplicity will specify the width of the segments to be drawn as follows: if the number of objects W present in a membrane is n, then the segment corresponding to this membrane must be drawn with width $n \times w$.

For a better understanding let us consider the following example: let Π_1 be the restricted P system with membrane creation such that

- The alphabet of objects is $O = \{L, E, W, F, +, -, B_L, B_R, B_{S_1}, B_{S_2}\}$.
- The initial membrane structure together with the initial multiset of objects is $[LEWFB_LB_{S_1}]$.
- The rules are:

$$B_{S_1} \to [LEWFB_{S_2}B_R] \qquad B_L \to [+LEWFB_LB_{S_1}]$$
$$B_{S_2} \to [LEWFB_LB_{S_1}] \qquad B_R \to [-LEWFB_LB_{S_1}]$$
$$L \to LF \qquad E \to EW$$

In this system, the objects B_{S_1} and B_{S_2} represent straight branches to be created, whereas the objects B_L and B_R represent branches to be created rotated to the left and to the right, respectively. The objects F and W will determine the length and the width of the corresponding branch. The objects L and E do not have a graphical interpretation; they can be considered as seeds for growing the branch in length and width.

[2] A preliminary version with simpler examples can be found in [21].

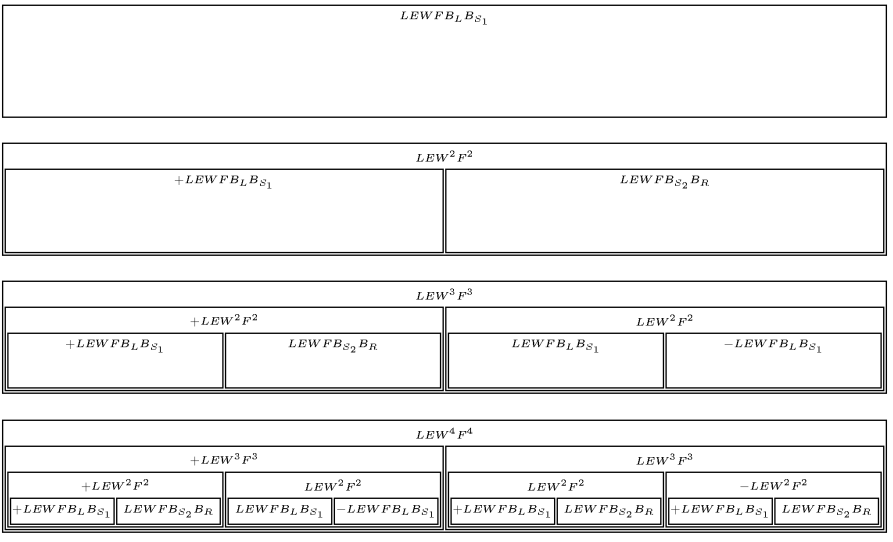

Fig. 3. First four configurations

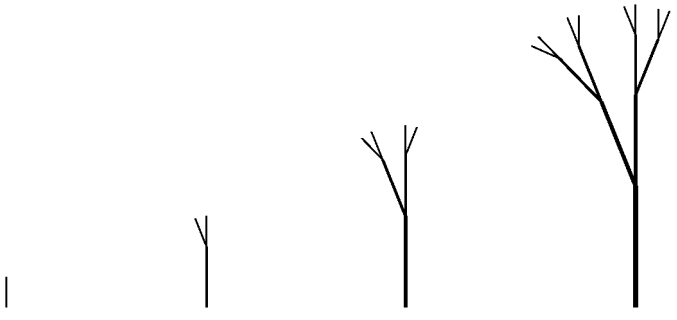

Fig. 4. Graphical representation of the configurations

This way, a thorough analysis of the initial membrane structure and of the rules shows that Π_1 models a branching structure consisting of a main trunk from which branches to the left and to the right come out alternatively. These new branches behave in the same way as the main trunk. Figure 3 shows the evolution of the system along three computation steps, and we can see in Figure 4 the corresponding graphical representation of each configuration, where we fix a bottom-up orientation with a length l of 1 cm, a width w of 0.4 pt and an angle δ of 22.5 degrees.

Note how the graphical representation of the configurations shows that the growth of the branching structure being modeled is not made by replacing segments by complex and repetitive modules, but by expanding the figure from

specific points with new segments (similar to how branches of higher plants spring from buds).

6 Stochastic Versus Non-deterministic P Systems

In the previous section we have introduced a technique to construct somehow realistic representations of branching structures in the framework of P systems. Each configuration of a given restricted P system with membrane creation can be translated to a set of graphical commands which produce the required picture.

An important drawback, as the example above shows, is that all the trees generated by the same deterministic P system are identical. It is then necessary to introduce specimen-to-specimen variations that preserve the general aspects of the branching structure but modify its details.

One possible way to achieve this is to consider stochastic P systems. Several alternatives to incorporate randomness into membrane systems can be found in the literature (see [1,13,16] and the references therein). One of them is to associate each rule of the P system with a probability. Thus, to pass from a configuration of the system to the next one we apply to every object present in the configuration a rule chosen at random, according to those probabilities, among all the rules whose left–hand side coincides with the object.

For example, let us consider Π_2 the following restricted P system with membrane creation:

- The alphabet of objects is $O = \{L, E, W, F, +, -, T, B\}$.
- The initial membrane structure together with the initial multiset of objects is $[LEWFTB]$.
- The rules are:

$$T \to [LEWFTB] \qquad L \to LF$$
$$B \xrightarrow{2/3} [+LEWFTB] \qquad E \to EW$$
$$B \xrightarrow{1/3} [-LEWFTB]$$

Note that we do not make explicit the probability of the rule when this is one. Also, it is easy to see from the probabilities assigned to the rules with left–hand side equal to B, that the trees generated by this system favors developing branches to the left.

Let us fix a length l of 1 cm, a width w of 0.4 pt and an angle δ of 22.5 degrees. Thus, after two computation steps we could have obtain, depending on the rules chosen for the object B, any of the trees shown in Fig. 5, with the upper left one being the most probable, and this probability decreasing as we go from left to right and from top to bottom.

On the other hand, for non-deterministic P systems we could consider together all the trees generated by all of its computations. We would obtain in this way a "forest". Thus, taking Π_2 as a non-deterministic system instead of a stochastic one, we would obtain after two computation steps the graphical representation

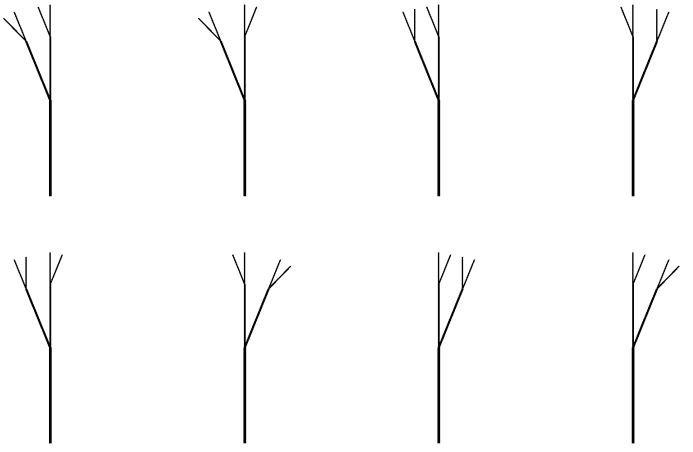

Fig. 5. Trees obtained from Π_2

depicted in Figure 5 (here the trees have been arranged in a 2×4 rectangular grid, but another methods to place the trees are possible).

7 Final Remarks

In this paper we have shown the suitability of P systems for modeling the growth of branching structures. It is our opinion that using membrane computing for this task could be an alternative to L systems, the model most widely studied nowadays, for several reasons: the process of growing is closer to reality, since for example a plant does not grow by "rewriting" its branches, but by lengthening, widening and ramifying them; the membrane structure of P systems supports better and clearer the differentiation of the system into small units, easier to understand and possibly with different behaviors; the computational power of membrane systems can provide tools to easily simulate more complex models of growing, for example taking into account the flow of nutrients or hormones.

Nevertheless, it is still necessary a deeper study of several features of our proposed framework as compared with that of Lindenmayer systems. Two aspects that have to be investigated are the complexity of the models that can be constructed, and the computational efficiency in order to generate their graphical representation. On one hand, the use of the ingredients of membrane computing can lead to more intuitive models; on the other hand, we lose the linear sequence of graphical commands that characterize the parsing algorithm of L systems.

It is also fundamental to develop computer tools that make easier the generation, within our framework, of the graphical representations from given models. For that, it should be interesting to analyze the possibility of transforming the models of higher plants constructed using membrane computing into equivalent ones using Lindenmayer systems. This way, we could reuse the available software for these latter systems.

The computation model considered here, restricted P system with membrane computing, is a very simple one (at least, in terms of membrane computing). We finish the paper by proposing extensions to this model in several ways that we think are interesting enough to be considered and studied:

- A labeling of the membranes could be useful to distinguish between different parts of the plant being modeled.
- The use of communication rules, allowing objects to cross the membranes of the system, are basic for modeling the flow of nutrients and hormones.
- Rules of the form $o \rightarrow \mu$, where o is an object and μ is a membrane structure, could lead to a clearer, faster and more compact representation of a plant.

Acknowledgement. This work was supported by the Project TIN2005-09345-C03-01 of the Ministry of Education and Science of Spain, cofinanced by FEDER funds, and by the Project of Excellence TIC-581 of the Junta de Andalucía.

References

1. Ardelean, I.I., Cavaliere, M.: Modelling Biological Processes by Using a Probabilistic P System Software. *Natural Computing*, **2**, 2 (2003), 173–197.
2. Chomsky, N.: Three Models for the Description of Language. *IRE Trans. on Information Theory*, **2**, 3 (1956), 113–124.
3. Frijters, D., Lindenmayer, A.: A Model for the Growth and Flowering of Aster Novae-Angliae on the Basis of Table (0,1)L systems. In Rozenberg, G., Salomaa, A. (eds.): *L systems. Lecture Notes in Computer Science*, Vol. 15. Springer–Verlag, Berlin, 1974, 24–52.
4. Frijters, D., Lindenmayer, A.: Developmental Descriptions of Branching Patterns with Paracladial Relationships. In Lindenmayer, A., Rozenberg, G. (eds.): *Automata, Languages, Development*. North–Holland, 1976, 57–73.
5. Georgiou, A., Gheorghe, M.: Generative Devices Used in Graphics. In Alhazov, A., Martín–Vide, C., Păun, Gh. (eds.): *Preproceedings of the Workshop on Membrane Computing*. Technical Report, Vol. 28/03. Research Group on Mathematical Linguistics, Universitat Rovira i Virgili, Tarragona, 2003, 266–272.
6. Georgiou, A., Gheorghe, M., Bernardini, F.: Membrane–Based Devices Used in Computer Graphics. In Ciobanu, G. Păun, Gh., Pérez–Jiménez, M.J. (eds.): *Applications of Membrane Computing*. Springer–Verlag, Berlin, 2006, 253–282.
7. Ito, M., Martín–Vide, C., Păun, Gh.: A Characterization of Parikh Sets of ET0L Languages in Terms of P Systems. In Ito, M., Păun, Gh., Yu, S. (eds.): *Words, Semigroups, and Transductions*. World Scientific, 2001, 239–254.
8. von Koch, H.: Sur One Courbe Continue sans Tangente, Obtenue par une Construction Géometrique Élémentaire. *Arkiv för Matematik*, **1** (1904), 681–704.
9. von Koch, H.: Une Méthode Géometrique Élémentaire pour L'étude de Certaines Questions de la Théorie des Courbes Planes. *Acta Mathematica*, **30** (1906), 145–174.
10. Lindenmayer, A.: Mathematical Models for Cellular Interaction in Development, Parts I and II. *Journal of Theoretical Biology*, **18** (1968), 280–315.
11. Lindenmayer, A.: Developmental Systems without Cellular Interaction, Their Languages and Grammars. *Journal of Theoretical Biology,* **30** (1971), 455-484.

12. Madhu, M., Krithivasan, K.: P Systems with Membrane Creation: Universality and Efficiency. In Margenstern, M., Rogozhin, Y. (eds.): *Proceedings of the Third International Conference on Universal Machines and Computations*. Lecture Notes in Computer Science, Vol. 2055. Springer–Verlag, Berlin, 2001, 276–287.
13. Obtulowicz, A., Păun, Gh.: (In Search of) Probabilistic P Systems. *Biosystems*, **70**, 2 (2003), 107–121.
14. Papert, S.: *Mindstorms: Children, Computers and Powerful Ideas*. Basic Books, 1980.
15. Păun, Gh.: *Membrane Computing – An Introduction*. Springer-Verlag, Berlin, 2002.
16. Pescini, D., Besozzi, D., Mauri, G., Zandron, C.: Dynamical Probabilistic P Systems. *International Journal of Foundations of Computer Science*, **17**, 1 (2006), 183–204.
17. Prusinkiewicz, P., Lindenmayer, A., Hanan, J.: Developmental Models of Herbaceous Plants for Computer Imagery Purposes. *Computer Graphics*, **22**, 4 (1988), 141–150.
18. Prusinkiewicz, P.: Graphical Applications of L systems. In *Proceedings of Graphical Interface '86*. Kaufmann, 1986, 247–253.
19. Prusinkiewicz, P., Hanan, J.: *Lindenmayer Systems, Fractals and Plants*. Lecture Notes on Biomathematics, Vol. 79. Springer-Verlag, Berlin, 1989.
20. Prusinkiewicz, P., Lindenmayer, A.: *The Algorithmic Beauty of Plants*. Springer-Verlag, Berlin, 1990.
21. Romero–Jiménez, A., Gutiérrez–Naranjo, M.A., Pérez–Jiménez, M. J.: The Growth of Branching Structures with P Systems. In Graciani–Díaz, C., Păun, Gh., Romero–Jiménez, A., Sancho–Caparrini, F. (eds.): *Proceedings of the Fourth Brainstorming Week on Membrane Computing*, Vol. II. Fénix Editora, Sevilla, 2006, 253–265.
22. Salomaa, A.: *Formal Languages*. Academic Press (1973)
23. Smith, A.R.: Plants, Fractals and Formal Languages. *Computer Graphics*, **18**, 3 (1984), 1–10.
24. The P Systems webpage `http://psystems.disco.unimib.it/`