The Growth of Branching Structures with P Systems

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

Dpto. de Ciencias de la Computación e Inteligencia Artificial
E.T.S. Ingeniería Informática, Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012, Sevilla, España
Alvaro.Romero@cs.us.es, magutier@us.es, marper@us.es

Summary. L-systems have been widely used to model and graphically represent the growth of plants [18]. In [4], the use of membrane computing for such tasks has been proposed. In this paper we present a different approach, which makes use of the topology of membrane structures to model the morphology of branching structures. We also keep closer to reality by simulating their growth from buds, instead of rewriting existing structures, as L-systems do.

1 Introduction

The growth of plants, considered as a function of time, have attracted the attention of scientific community for a very 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 [9, 10]. These organisms are treated as an assembly of discrete units, which represent the individual cells. These systems 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 [2, 3]. In the first approach, the essence of development of lower organisms is the replacement of individual cells by sets of cells, in the terms established by the production rules of the system. On the other hand, in L-systems modeling higher plants the units of information represent

complex structures, such as branches or leaves, instead of individual cells. These structures are replaced by other ones using the production rules.

In [4, 5] 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, dealing in fact with 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 with sequential structures such as microorganisms or linear structure of fractals as the Koch curve [7, 8]. Nonetheless, the visual interpretation of strings of symbols as branching structures needs to add memory pointers in order to remember the point 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 relation. In this paper we use this feature to present a framework for modeling the topology of living plants, without the necessity of considering memory pointers. Besides, 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 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.

The paper is organized as follows. First L-systems and the usual way of 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. The next section is devoted to the graphical visualization of the configurations of these P systems, with some examples. 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 [19]), but rewriting systems started to be widely considered after Chomsky's work on formal grammars [1], 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 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 deterministic and context-free, called D0L-systems. 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 $a \to v \in P$. If no production rule is explicitly specified for a given predecessor $a \in V$, the identity production $a \to a$ is assumed to belong to the set of productions 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 $\rho = \phi_1 \dots \phi_m \in V^*$ is directly derived from (or generated by) μ, noted $\mu \Rightarrow \rho$, if and only if $a_i \to \phi_i \in P$ for all $i \in \{1, \dots, m\}$. A word ψ is generated by μ in a derivation of length n if there exists a developmental sequence of words $\mu_0, \mu_1, \dots, \mu_n$ such that $\mu_0 = \mu$, $\mu_n = \rho$, and $\mu_0 \Rightarrow \mu_1 \Rightarrow \dots \Rightarrow \mu_n$.

Note that in D0L-systems for all symbols of the alphabet $\alpha \in V$ there is exactly one rewriting rule. Starting with the axiom $\omega \in V^+$, a sequence of strings $\mu_0 = \omega$, μ_1, μ_2, ... is generated recursively, where the string $\mu_{(i+1)}$ is obtained from the preceding string μ_i by replacing *simultaneously* every symbol in μ_i.

3 Visualization of L-systems

Originally, L-systems were conceived for the study of multicellular organisms and the neighborhood relations among their different cells. After the incorporation of geometric features, L-systems became detailed enough to allow the use of computer graphics for realistic visualizations of these multicellular organism and their developmental processes. The first steps in this line can be found in the eighties' literature [15, 20], but the most popular graphical interface for L-systems was introduced by P. Prusinkiewicz [16, 17] based on the previous Papert's concept of *turtle graphics* [12]. In an informal description, we can consider a turtle standing on a sheet of paper facing in 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 $(x + l \cos\alpha, y + l \sin\alpha, \alpha)$. A line segment between (x, y) and $(x + l \cos\alpha, y + l \sin\alpha)$ is drawn.
- f: analogous to the previous command, the state of the turtle changes from (x, y, α) to $(x + l \cos\alpha, y + l \sin\alpha, \alpha)$. 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 representation of 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

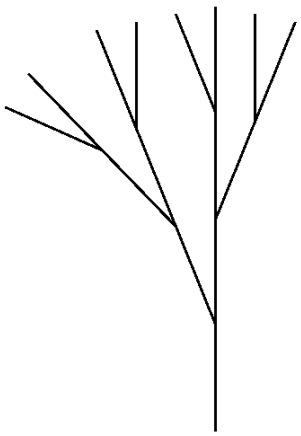

Fig. 2. The string $F[+F[+F[+F][F]][F[F][-F]]][F[F[+F][F]][-F[+F][F]]]$

defined by a cell-like membrane structure. This research area was initiated by Gh. Păun [14] at the end of 1998 and has flourished since then [13].

We can briefly describe this class of computing models as follows: consider a *membrane structure* as in Figure 3, consisting of several membranes arranged in a hierarchical structure inside a main membrane (called the *skin*) and delimiting *regions* (the compartments bounded by a membrane and the immediately lower membranes, if any). In the regions one places *multisets* of certain *objects*, that is, sets of objects with multiplicities associated with the elements. The objects correspond to chemicals evolving in the compartments of a cell and are represented by symbols from a given alphabet. They evolve according to given *evolution rules*, which are also localized, associated with the regions (hence with the membranes). The rules are applied non-deterministically in a maximally parallel manner (in each step, all objects which can evolve must do it). The objects can pass through the membranes and, in their turn, the membranes can be dissolved, divided, created. In this way, we get *transitions* from a *configuration* of the system to the next one. Note that the process is synchronized; a global clock is assumed, marking the time units for all the compartments of the system. A sequence of transitions constitutes a *computation*.

Since Gh. Păun presented the initial model of cellular computing with membranes, many different variants have been proposed. If the membrane structure is considered to set a classification among them, two big groups are obtained: P systems where the initial structure does not change along computations and P systems where the hierarchical structure of the membranes vary (or can do it) along computations. For P systems belonging to this latter group, we can find in the literature rules that reduce the number of membranes (*dissolution rule*) and

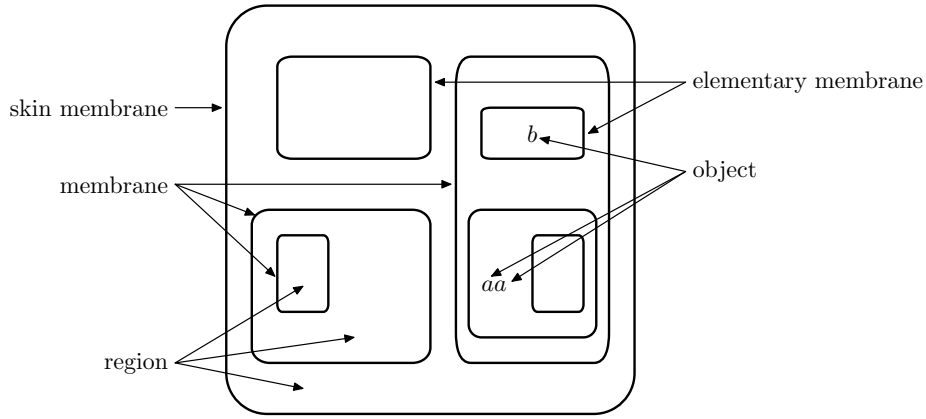

Fig. 3. A P system

rules that increase that number (*membrane division rule* and *membrane creation rule*).

In this paper we will consider P systems which make use of membrane creation rules, which was first introduced in [6, 11]. 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 construct $\Pi = (O, \mu, w_1, \ldots, w_m, R)$, where:

1. O is the alphabet of *objects*.
2. μ is the initial *membrane structure*, consisting of a hierarchical structure of m membranes.
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*, which can be of the two following kinds:
   a) $a \rightarrow v$, where $a \in O$ and v is a multiset over O. This rule makes an object a present in a membrane of μ evolve into a multiset of objects v.
   b) $a \rightarrow [v]$, where $a \in O$ and v is a multiset over O. This rule makes an object a present in a membrane of μ create within the latter a new membrane containing only the multiset of objects v.

The membrane structure (obtained from the membrane structure μ) together with the objects contained in its membranes constitute the 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 maximally parallel way: for a rule to be able to be applied in a membrane, this latter must contain an object matching the left hand side of the rule; this object is then consumed and the components indicated in the right hand 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 Visualization of Restricted P Systems with Membrane Creation

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

Once we have a membrane structure establishing the topology of the structure 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.

The simpler model to graphically represent a membrane structure is to make a depth-first search of it, drawing a segment of length l for each membrane containing an object F. This segment is drawn rotated an angle of $n \times \delta$ with respect to the segment corresponding to the parent membrane, 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.

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

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

$$B_{S_1} \to [FB_{S_2}B_R], \qquad B_L \to [+FB_LB_{S_1}],$$
$$B_{S_2} \to [FB_LB_{S_1}], \qquad B_R \to [-FB_LB_{S_1}].$$

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. 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. We can see in Figure 4 the evolution of the system along three computation steps, and in Figure 5 the corresponding graphical representation of each configuration,

where we fix a bottom-up orientation, a length l of 1cm, 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 the segments by complex and repetitive modules, but by expanding the figure from specific points with new segments (in a way similar to the way branches of higher plants spring from buds).

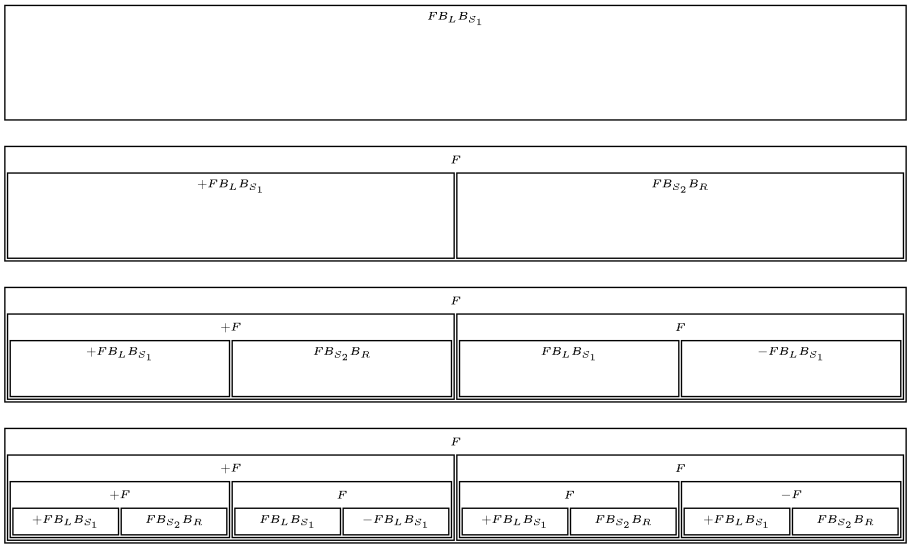

Fig. 4. First four configurations of the first example

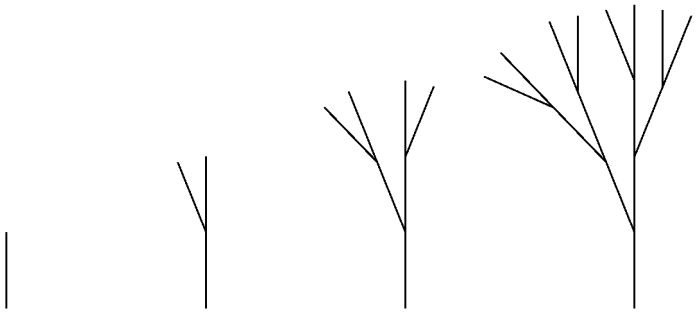

Fig. 5. Representation of the configurations of the first example

In the representation model introduced above, we use the multiplicity of the objects $+$ and $-$ to specify the angles by which the drawn segments are rotated. On the other hand, the multiplicity of the object F in the membranes does not affect the final result; a segment of length l is always drawn. The basic graphical model can therefore be extended in a natural way by making the length of the segments to be drawn depend on the number of objects F present in the corresponding membrane. Namely, this length would be $n \times l$, where n is the multiplicity of F.

As an example, let Π_2 be the restricted P system with membrane creation such that:

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

$$\begin{aligned} B_{S_1} &\to [LFB_{S_2}B_R], & B_L &\to [+LFB_LB_{S_1}], \\ B_{S_2} &\to [LFB_LB_{S_1}], & B_R &\to [-LFB_LB_{S_1}], \\ L &\to LF. \end{aligned}$$

As we can see in Figures 6 and 7, this P system models the same tree as Π_1, with the only difference being that the branches, besides ramifying, also lengthen themselves in each computation step. This is a step closer to reality, since the growth of the branching structure involves not only its morphology evolution but also the growth of its existing branches.

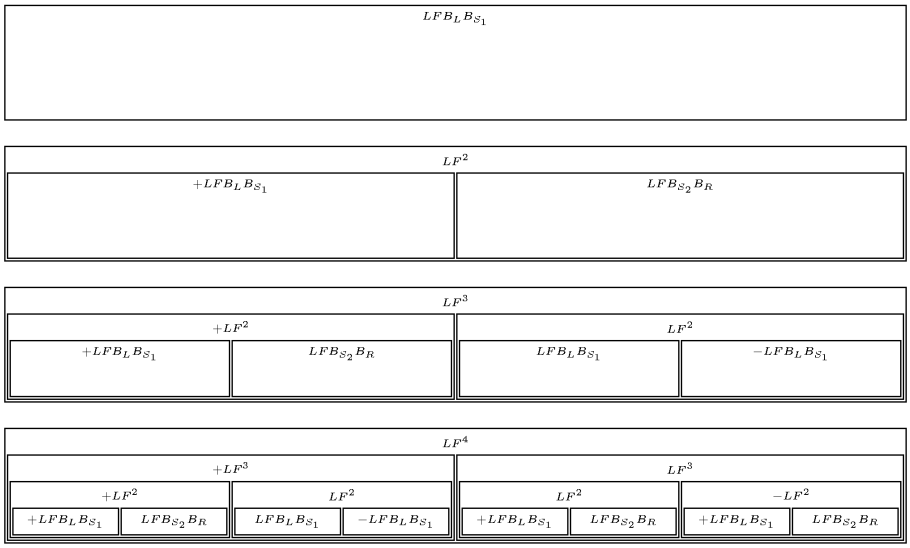

Fig. 6. First four configurations of the second example

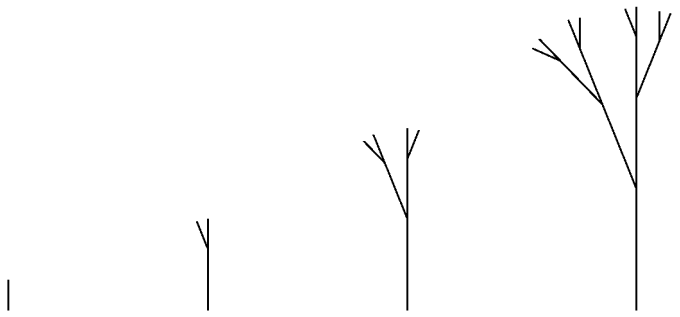

Fig. 7. Representation of the configurations of the second example

The second extension we will make to the representation model comes from the consideration that, in the real world, the branches of a plant not only lengthen but also widen themselves. Thus, we fix the width w and use a new symbol W whose multiplicity will specify the width of the segments to be drawn like 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 $n \times w$.

The restricted P system with membrane creation Π_3 with the following properties extends Π_2 to the new representation model:

- 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:

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

Figure 8 shows again the first four configurations of Π_3, and Figure 9 shows their graphical representation, where we can see that there exist a lengthening and widening of the existing branches in each computation step.

6 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 improvement with respect 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

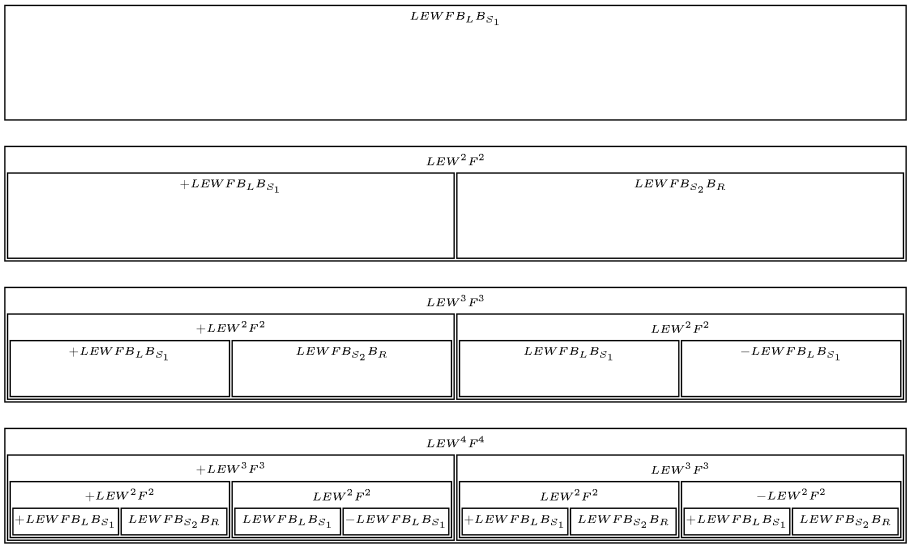

Fig. 8. First four configurations of the third example

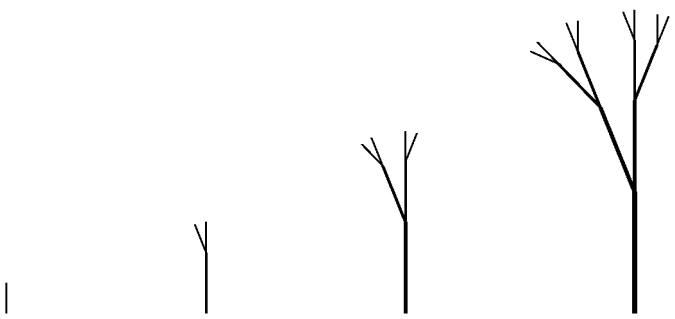

Fig. 9. Representation of the configurations of the third example

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 simulate more complex models of growing, for example, taking into account the flow of nutrients or hormones.

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.
- Stochastic P systems, where a probability is associated with each rule, are a natural way to introduce randomness into the model.

Acknowledgement

Work supported by project TIN2005-09345-C03-01 of Ministerio de Edicación y Ciencia of Spain, cofinanced by FEDER funds.

References

1. N. Chomsky: Three models for the description of language. *IRE Trans. on Information Theory*, 2, 3 (1956), 113–124.
2. D. Frijters, A. Lindenmayer: A model for the growth and flowering of Aster novae-angliae on the basis of table (0,1)L-systems. In *L-Systems* (G. Rozenberg, A. Salomaa, eds.), LNCS 15, Springer, 1974, 24–52.
3. D. Frijters, A. Lindenmayer: Developmental descriptions of branching patterns with paracladial relationships. In *Automata, languages, development* (A. Lindenmayer, G. Rozenberg, eds.), North-Holland, 1976, 57–73.
4. A. Georgiou, M. Gheorghe: Generative devices used in graphics. In *Preproceedings of the Workshop on Membrane Computing* (A. Alhazov, C. Martín–Vide, Gh. Păun, eds.) 2003, 266–272.
5. A. Georgiou, M. Gheorghe, F. Bernardini: Membrane–based devices used in computer graphics. In *Applications of Membrane Computing* (G. Ciobanu, G. Păun, M.J. Pérez–Jiménez, eds.), Springer, 2006, 253–282.
6. M. Ito, C. Martín–Vide, G. Păun: A characterization of Parikh sets of ET0L languages in terms of P systems. In *Words, Semigroups, and Transducers* (M. Ito, G. Păun, S. Yu, eds.), World Scientific, 2001, 239–254.
7. H. von Koch: Sur one courbe continue sans tangente, obtenue par une construction géometrique élémentaire. *Arkiv för Matematik*, 1 (1904), 681–704.
8. H. von Koch: Une méthode géometrique élémentaire pour l'étude de certaines questions de la théorie des courbes planes. *Acta Mathematica*, 30 (1906), 145–174.
9. A. Lindenmayer: Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 18 (1968), 280–315.
10. A. Lindenmayer: Developmental systems without cellular interaction, their languages and gramars. *Journal of Theoretical Biology*, 30 (1971), 455-484.
11. M. Madhu, K. Krithivasan: P systems with membrane creation: Universality and efficiency. In *Proc. Third. Intern. Conf. on Universal Machines and Computations* (M. Margenstern, Y. Rogozhin, eds.), LNCS 2055, Springer, 2001, 276–287.
12. S. Papert: *Mindstorms: Children, Computers and Powerful Ideas*. Basic Books, 1980.
13. Gh. Păun: *Membrane Computing – An Introduction*. Springer, 2002.
14. Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143.

15. P. Prusinkiewicz, A. Lindenmayer, J. Hanan: Developmental models of herbaceous plants for computer imagery purposes. *Computer Graphics*, 22, 4 (1988), 141–150.
16. P. Prusinkiewicz: Graphical applications of L-systems. *Proceedings of Graphical Interface '86*, Kaufmann, 1986, 247–253.
17. P. Prusinkiewicz, J. Hanan: Lindenmayer systems, fractals and plants. *Lecture Notes on Biomathematics*, 79, Springer, 1989.
18. P. Prusinkiewicz, A. Lindenmayer: *The Algorithmic Beauty of Plants*, Springer, 1990.
19. A. Salomaa: *Formal Languages*. Academic Press, 1973.
20. A. R. Smith: Plants, fractals and formal languages. *Computer Graphics*, 18, 3 (1984), 1–10.