Introduction to Modeling with L-systems

Przemyslaw Prusinkiewicz[1]
Department of Computer Science
University of Calgary

Abstract

This note introduces L-systems as a programming language for developmental model construction. The examples presented here refer to a real biological structure, the vegetative segment of a cyanobacterium *Anabaena catenula*. Problems related to time management in developmental simulations are given particular attention.

1 Background

L-systems were introduced by A. Lindenmayer as a mathematical formalism for modeling multicellular organisms that form linear or branching filaments [11, 12]. The whole organism is treated as an assembly of discrete units, called *modules*. The nature of the modules is not predefined by the formalism. For example, they may represent individual cells in the models of lower organisms, or functional units, such as apical meristems, internodes, leaves and flowers, in the models of higher plants. Each module is represented by a symbol (a letter of the L-system *alphabet*), which specifies the module's *type*. In addition, a module can be characterized by one or more numerically-valued *parameters* [2, 13], which, together with the symbol, characterize the module's *state*. The resulting concept of *parametric L-systems* has been formalized in [5, 23], among others. Recent extensions make it possible to use parameters of compound data types, such as structures [7] and lists [3].

L-system models are inherently *dynamic*, which means that the form of an organism is viewed as a result of development: "an event in space-time, and not merely a configuration in space" [28]. The development of a

[1]Based in part on: P. Prusinkiewicz: Introduction to modeling using L-systems, in J. A. Kaandorp and J. E. Kübler (Eds.), *The Algorithmic Beauty of Seweeds, Sponges, and Corals*, Springer, Berlin, 2001, pp. 91–93, and P. Prusinkiewicz, J. Hanan and R. Měch, An L-system-based plant modeling language, *Lecture Notes in Computer Science* **1779**, Springer, Berlin, pp. 395–410.

structure is described in terms of *rewriting rules* or *productions* that change the module's state, or replace a module by zero, one or several new modules. Productions are applied in parallel, which is intended to capture the simultaneous progress of time in all parts of the growing organism.

L-systems were originally introduced as a mathematical tool for reasoning about development. Soon after their introduction, however, they began to be used as a formal basis for constructing simulation modeling programs and modeling languages. The availability of these languages is one of the key practical advantages of L-systems in modeling applications, because diverse structures and processes can be modeled using the same L-system-based procedural modeling program with different inputs. The L-system-based languages have initially been almost indistinguishable from the underlying mathematical notation of L-systems, but, with the increased complexity of the modeling tasks, they gradually diverged from this notation.

The notion of L-systems is often presented in a formal manner. In contrast, we will present it in the context of a specific modeling application. This will allow us to see the relationship between a natural phenomenon and its L-system models, highlight the conceptual essence of L-systems, see the motivation behind their extensions, and trace the evolution of L-system-based modeling languages. In addition, the examples address the question of time management in simulations, which is essential to the animation of development.

2 Model organism: *Anabaena catenula*

Anabaena is one of the earliest examples of multicellular organisms with cellular specialization [1, 4]. Like many other cyanobacteria, it is capable both of aerobic photosynthesis and fixation of nitrogen from the atmosphere. These processes, however, are incompatible with each other, because the enzyme nitrogenase responsible for nitrogen fixation is destroyed by oxygen, a byproduct of photosynthesis. Some cyanobacteria deal with this incompatibility by dividing their activities over time: they photosynthesize in the daytime, and fix nitrogen in the night. *Anabaena*, in contrast, divides these activities in space, using two different types of cells: *vegetative cells* responsible for photosynthesis, and *heterocysts* responsible for nitrogen fixation. This division of tasks is the rationale for the multicellular structure of the organism.

Anabaena cells are arranged into filaments. In the absence of ammonia or nitrate in the medium, individual heterocysts are separated by sequences of approximately ten vegetative cells of varying sizes. The filaments grow by asymmetric division of vegetative cells. As this process moves the existing heterocysts apart, new heterocysts differentiate from vegetative cells roughly midway between those already present. Consequently, the regular spacing of heterocysts remains approximately constant while the filament grows.

The developmental process outlined above raises two questions, which have attracted the interest of biologists and modelers alike:

1. What mechanism controls the asymmetric division of vegetative cells, resulting in the observed sequence of cell sizes along the filament; and

2. What mechanism regulates the spacing of heterocysts?

In this note we will address the first question.

3 Context-free L-systems

Mitchison and Wilcox [18] proposed to explain the observed pattern of long and short cells in a vegetative segment of *Anabaena* as a direct result of a developmental process. To this end, they divided the cells that form the filament into two classes: long cells L and small cells S. Furthermore, they assumed that each vegetative cell has one of two possible polarities, which can be indicated by an arrow: $\overrightarrow{L}$, $\overleftarrow{L}$, and $\overrightarrow{S}$, $\overleftarrow{S}$. During the development, cells S elongate and change their state to L. Long cells divide, producing a cell L and a cell S.

Lindenmayer [16] observed that, taking polarities into account, the essence of the model of Mitchison and Wilcox can be written as a set of *rewriting rules*, or *productions*:

$$\overrightarrow{S} \longrightarrow \overrightarrow{L} \qquad \overleftarrow{S} \longrightarrow \overleftarrow{L} \qquad \overrightarrow{L} \longrightarrow \overleftarrow{L}\,\overrightarrow{S} \qquad \overleftarrow{L} \longrightarrow \overleftarrow{S}\,\overrightarrow{L} \tag{1}$$

These rules are the cornerstone of a DOL-system, the simplest type of L-systems [12] (see also [6, 26, 23, 27]). The acronym DOL stands for a **D**eterministic **L**-system with **0** interactions. This means that exactly one production applies to any symbol of the L-system alphabet, and the productions are *context-free*. The DOL-system productions have the form

$$\textit{predecessor} \longrightarrow \textit{successor},$$

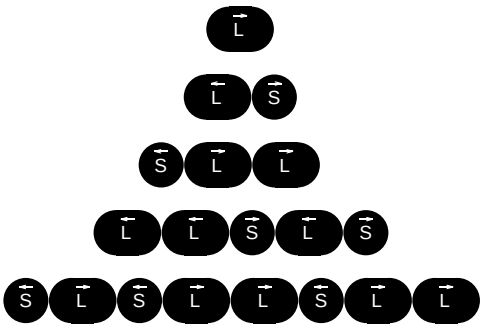

Figure 1: Developmental sequence of an *Anabaena* filament modeled with L-system 1.

where the *predecessor* is a single symbol of the L-system alphabet V, and the *successor* is a string of zero, one, or several symbols from the same alphabet.

Like all L-systems, DOL-systems operate on sequences of symbols called *strings* or *words*. In a single *derivation step*, each letter in the predecessor string is replaced by its *successor* using the applicable production from the *production set* P. The developmental process is simulated as a sequence of such derivation steps, beginning with a given initial string, called the *axiom*, and denoted ω. For example, Figure 1 shows the developmental sequence generated by L-system $\mathcal{G} = \langle V, \omega, P \rangle$ with alphabet $V = \{\overrightarrow{L}, \overleftarrow{L}, \overrightarrow{S}, \overleftarrow{S}\}$, axiom $\omega = \overrightarrow{L}$, and the production set P given by Equation 1.

The above model implies that the time between the formation and division of a short cell is twice as long as the time between the formation and division of a long cell. In reality, a short cell takes only about 20% longer to divide that a long cell. Lück and Lück [17], and Lindenmayer [15] incorporated this behavior into the model by introducing a sequence of state transitions to control the timing of cell division:

$$\begin{array}{ccccc} \overrightarrow{S} \longrightarrow \overrightarrow{L} & \overrightarrow{L} \longrightarrow \overrightarrow{A} & \overrightarrow{A} \longrightarrow \overrightarrow{B} & \overrightarrow{B} \longrightarrow \overrightarrow{C} & \overrightarrow{C} \longrightarrow \overleftarrow{L}\overrightarrow{S} \\ \overleftarrow{S} \longrightarrow \overleftarrow{L} & \overleftarrow{L} \longrightarrow \overleftarrow{A} & \overleftarrow{A} \longrightarrow \overleftarrow{B} & \overleftarrow{B} \longrightarrow \overleftarrow{C} & \overleftarrow{C} \longrightarrow \overleftarrow{S}\overrightarrow{L} \end{array} \tag{2}$$

Sequences of state transitions are not intuitively represented by the L-system formalism, but can be conceptualized using diagrams, similar to that shown in Figure 2. Similar diagrams have been used by Lindenmayer [10] and Lück and Lück [17]. Formally, they are a synchronously operating vari-

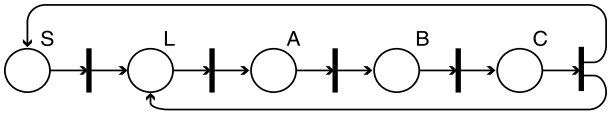

Figure 2: Petri net representation of L-system 2. A cell undergoes a sequence of transitions (vertical bars), which advance its state (circles). The last transition represents division of cell C into two children cells, S and L.

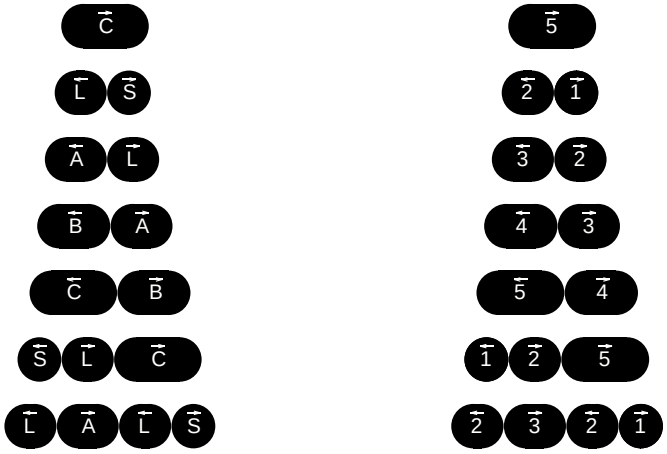

Figure 3: Improved developmental sequences of the *Anabaena* vegetative filaments. The models that take into account timing of the short and long cell divisions. Left: the sequence generated by non-parametric L-system 2. Right: the corresponding sequence generated by the parametric L-system 3.

ant of Petri nets [19, 25]. As illustrated in Figure 2, L-system 2 specifies a sequence of developmental stages, through which a cell progresses until it reaches the dividing state C. The children of cell C have different positions along this sequence, and therefore need different times to divide again. Specifically, the time to the next division is equal to 4 time units for a cell L and 5 time units for a cell S, thus resulting in the required ratio of 80%. This timing is also evident in the developmental sequence shown in Figure 3 on the right.

4 Parametric L-systems

Sequences of developmental stages significantly increase the size of the L-sysem alphabet and the size of the corresponding production set, and consequently complicate the specification of DOL-systems. *Parametric L-systems* [5, 23] offer a solution to this problem, by associating numerical parameters with L-system symbols. Context-free parametric productions have the form

$$predecessor : condition \longrightarrow successor,$$

where *predecessor* is a single module (L-system symbol with the associated parameters), *successor* is a sequence of modules, and *condition* is the logical expression that determines whether the production can be applied to a given module. For example, a parametric L-system equivalent to that in Equation 2 is given below:

$$\begin{array}{lcll} \vec{M}\,(s): & s<5 & \longrightarrow & \vec{M}\,(s+1) \\ \vec{M}\,(s): & s==5 & \longrightarrow & \overleftarrow{M}\,(2)\,\vec{M}\,(1) \\ \overleftarrow{M}\,(s): & s<0 & \longrightarrow & \overleftarrow{M}\,(s+1) \\ \overleftarrow{M}\,(s): & s==5 & \longrightarrow & \overleftarrow{M}\,(1)\,\vec{M}\,(2) \end{array} \tag{3}$$

The non-parametric L-system 2 and the parametric L-system 3 are equivalent because modules type in L-system 2 correspond one-to-one to parameter values in L-system 3:

$$S \leftrightarrow 1, \qquad L \leftrightarrow 2, \qquad A \leftrightarrow 3, \qquad B \leftrightarrow 4, \qquad C \leftrightarrow 5. \tag{4}$$

This correspondence is also illustrated by Figure 3, which compares the developmental sequences generated by both L-systems.

Parameters are useful in specifying various attributes of modules. For example, in the case of *Anabaena*, cell polarity can also be represented as a parameter. This results in the following L-system:

```
/* L-system 4 */
#define LEFT -1
#define RIGHT 1
Axiom: M(5,RIGHT)
M(t,p) : t<5 --> M(t+1,p)
M(t,p) : t==5 && p == LEFT --> M(1,LEFT)M(2,RIGHT)
M(t,p) : t==5 && p == RIGHT --> M(2,LEFT)M(1,RIGHT)
```

We have switched here from the mathematical notation to a programming language style of L-system specification. The particular language used has been devised for the L-system-based modeling program `cpfg` [22]. At this point, the differences between the mathematical notation and the corresponding `cpfg` specification are minimal, but the programming language style will allow us to more conveniently present further extensions to L-systems.

5 Modeling development in continuous time

All the L-systems discussed above advance time by unit interval per derivation step. In many applications, for example the animation of development, it is convenient to advance time by arbitrary user-defined increments. We can partially achieve this goal by reinterpreting the step-counting variable t in L-system 4 as a continuously-valued *developmental stage* or *physiological age* of the cell, and increment it by a constant dt. By applying this concept to L-system 4, and rescaling time so that the short cells have unit life span, we obtain:

```
/* L-system 5 */
#define LEFT -1
#define RIGHT 1
#define t_div 1.0       /* cell age at division */
#define t_s 0.0         /* initial age - short cell */
#define t_l 0.2         /* initial age - long cell */
#define dt 0.7          /* time increment per step */
Axiom: M(0,RIGHT)
M(t,p) : t+dt<t_div --> M(t+dt,p)
M(t,p) : t+dt>=t_div && p == LEFT -->
        M(t+dt-t_div+t_s,LEFT) M(t+dt-t_div+t_l,RIGHT)
M(t,p) : t+dt>=t_div && p == RIGHT -->
        M(t+dt-t_div+t_l,LEFT) M(t-t_div+t_s+dt,RIGHT)
```

Figure 4 shows developmental sequences generated by this L-system for two values of time increment, $dt_1 = 0.7$ and $dt_2 = 1.4$. A comparison of these sequences reveals that the results are correct for the smaller time increment dt_1, but incorrect for the larger increment dt_2. The reason for this discrepancy is presented in Figure 5, which places the branching process of cell division in the context of time increments used in both simulation. If

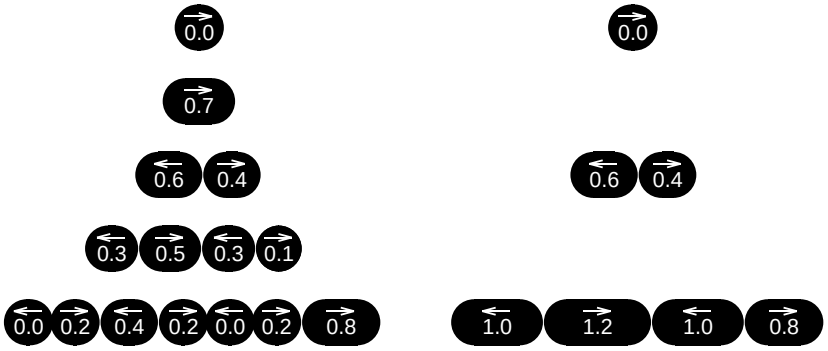

Figure 4: Developmental sequences generated by L-system 5 using time steps $dt_1 = 0.7$ (left) and $dt_2 = 1.4$ (right). The sequence on the left correctly captures the development of the filament, whereas sequence on the right does not, because it includes cells past their division age.

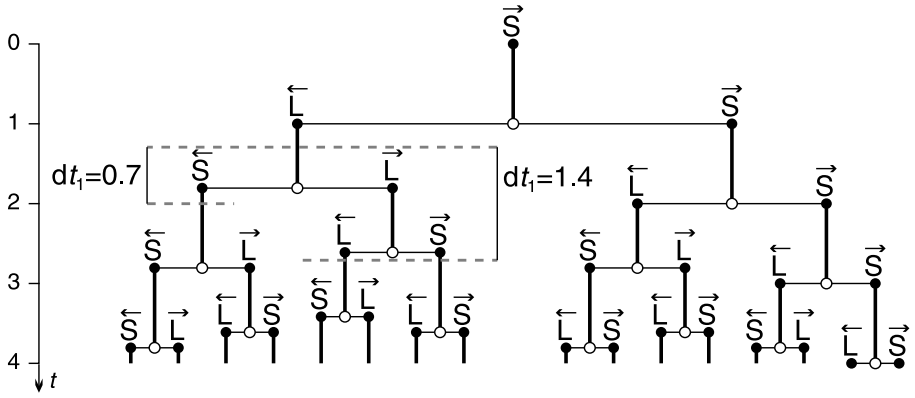

Figure 5: Development of an *Anabaena* filament considered in continuous time. Each cell divides at most once within any interval of duration $dt_1 =< 0.8$, but may divide more than once within an interval $dt_2 \geq 0.8$. Sample intervals $dt_1 = 0.7$ and $dt_2 = 1.4$ are shown.

the time increment dt is sufficiently small, for example $dt = 0.7$, each cell undergoes at most one division within an interval $(t, t+dt]$. This is correctly captured by L-system 5, according to which a cell either does not divide within a derivation step at all, or divides once. On the other hand, if the time increment dt is too large, for example $dt = 1.4$, some cells may undergo two divisions within an interval $(t, t + dt]$. This possibility is not accounted

for in L-system 5, which instead produces cells that exceed the age at which they should have divided.

6 Decomposition rules

Time steps of a duration longer than the minimum lifetime of a cell could be handled using productions that create more than two cells in a single derivation step. A more general and concise solution, however, is to assume that a cell will divide recursively, as long as the age of the descendants is greater than the division age t_{div} (Figure 6). Recursive application of productions is a concept characteristic of Chomsky grammars rather than L-systems. We combine these concepts by distinguishing two types of productions: L-system productions, applied in the breadth-first fashion, and Chomsky productions, applied recursively [24]. In the `cpfg` language, we refer to Chomsky productions as *decomposition rules*, and separate them from the ordinary L-system productions with the keyword `decomposition` [22]. A derivation step consists of the application of L-system productions to all modules in the string, followed by the recursive application of decomposition rules. A model of *Anabaena* development using decomposition rules is given by L-system 6.

```
/* L-system 6 */
#define LEFT -1
#define RIGHT 1
#define t_div 1.0       /* cell age at division */
#define t_s 0.0         /* initial age - short cell */
#define t_l 0.2         /* initial age - long cell */
#define dt 0.7          /* time increment per step */
Axiom: M(0,RIGHT)
M(t,p) --> M(t+dt,p)
decomposition:
M(t,p) : t>=t_div && p == LEFT -->
        M(t-t_div+t_s,LEFT) M(t-t_div+t_l,RIGHT)
M(t,p) : t>=t_div && p == RIGHT -->
        M(t-t_div+t_l,LEFT) M(t-t_div+t_s,RIGHT)
```

The use of decomposition rules improved the structure of L-system 6 with respect to L-system 5, by distinguishing between an ordinary L-system production, which advances time, and decomposition rules, which describe the

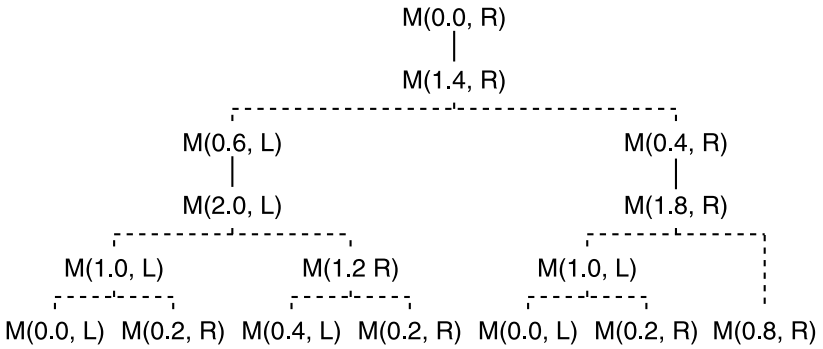

Figure 6: Development of an *Anabaena* filament simulated using L-system 6 with time step $dt = 1.4$. The application of the time-advancing L-system production (thick line) is followed by the recursive application of decomposition rules (thin lines).

structure of the descendants of each module. The underlying logical distinction between the relations "produces over time" and "is a part of" was formalized by Woodger and Tarski [29], and reviewed by Lindenmayer [14]. Figure 6 shows that L-system 6 works correctly for time increments exceeding the lifetime of a cell.

If we are only interested in the final state of the model at some time T, we can delegate all computation to the recursive application of decomposition rules by setting $dt = T$. This mode of operation is closely related to *timed L-systems*, defined in [23]. Timed L-systems make it possible to find the state of the developing structure at any time T, and consequently were the first variant of L-systems used to create "smooth" animations of plant development. As a limiting factor, the arbitrarily large time steps and recursive evaluation of L-systems are only possible in the deterministic context-free case, where components of the structure do not communicate with each other during the development, and always create the same lineage.

7 Interpretation rules

Up to now, we have not been concerned with the visualization of the models. This is consistent with the original definition of the L-system formalism, which only described ordering of cells in the filament, i.e., its topology. Nevertheless, in order to present generated models graphically, we need to

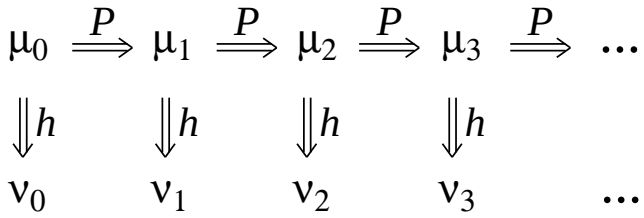

Figure 7: Generation of a developmental sequence using a `cpfg` model with interpretation rules. The progression of strings $\mu_0, \mu_1, \mu_2, \ldots$ results from the derivation steps $\stackrel{P}{\Longrightarrow}$ defined by productions and decomposition rules. The interpretation rules $\stackrel{h}{\Longrightarrow}$ map strings μ_i into the final strings ν_i that contain the graphical information.

assign a geometric interpretation to the modules. The problem is that the types of modules in terms of which a model is formulated and conceptualized (e.g., cells, apices, or leaves) is often very different from graphical primitives that are convenient to describe its geometry (e.g., lines, polygons, parametric surfaces). This problem was first addressed by Kurth [9], who introduced the notion of *interpretation rules*.

Interpretation rules provide a mechanism for complementing models with the geometric information needed for visualization purposes. They do not affect the sequence of strings $\mu_0, \mu_1, \mu_2, \ldots$ derived by productions and decomposition rules, but make it possible to temporarily replace modules in the derived strings by other modules or sequences of modules (Figure 7), which may have a predefined graphical interpretation. Formally, the interpretation rules are related to the notion of string *homomorphisms*, which have been studied in the mathematical theory of L-systems [6, 26]. The interpretation rules extend the concept of homomorphism, because they may operate on symbols with parameters, and can be applied in hierarchical and recursive manners. In that sense, they resemble decomposition rules.

In the `cpfg` language, the interpretation rules are specified using the same syntax as context-free productions and decomposition rules, following the keyword `homomorphism`. To see how they work, let us assume that the modeling program supports predefined modules `L(length,width)` and `R(length,width)`, which draw a round-cornered rectangles of given length and width. In addition, each module incorporates an arrow, pointing to the left or right, respectively. (In the actual `cpfg` implementation, modules `L` and `R` are defined in terms of more fundamental primitives, but this is

irrelevant to the basic concept.) We thus can define a complete L-system model of a vegetative segment of the Anabaena filament, by extending L-systems 6 with interpretation rules.

```
/* L-system 7 */
#define LEFT -1
#define RIGHT 1
#define t_div 1.0       /* cell age at division */
#define t_s 0.0         /* initial age - short cell */
#define t_l 0.2         /* initial age - long cell */
#define a 2.1673        /* exponential growth base */
#define WID 1.0         /* cell width */
#define dt 0.7          /* time increment per step */
Axiom: M(0,RIGHT)
M(t,p) --> M(t+dt,p)
decomposition:
M(t,p) : t>=t_div && p == LEFT -->
        M(t-t_div+t_s,LEFT) M(t-t_div+t_l,RIGHT)
M(t,p) : t>=t_div && p == RIGHT -->
        M(t-t_div+t_l,LEFT) M(t-t_div+t_s,RIGHT)
homomorphism:
M(t,p) : p==LEFT --> L(a^t,WID)
M(t,p) : p==RIGHT --> R(a^t,WID)
```

8 Geometric continuity

L-system 7 illustrates a nontrivial detail of the geometric interpretation of L-systems that can operate with arbitrarily small time steps: the need of preserving geometric continuity of the growing structure over time. In this specific example, we have assumed that the length of *Anabaena* cells increases exponentially with time: $l(t) = a^t$. Since the exponential function is continuous, the length of cells, and therefore the filament, will also be continuous functions of time in the intervals between cell divisions. In order to maintain the continuity during cell divisions as well, the length of a mother cell immediately before its division must be equal to the combined length of the daughter cells immediately after the division:

$$a^{t_{div}} = a^{t_s} + a^{t_l}$$

Given age values $t_{div} = 1.0$, $t_s = 0.0$ and $t_l = 0.2$, we obtain the basis a of the exponential growth by numerically solving the above equation: $a \approx 2.1673$. Thus, the value of a is a logical consequence of the timing of cell division and the exponential character of cell growth. The cell width WID has been set to $a^{t_s} = 1.0$, so that the smallest cells in the filament appear round. For a more general discussion of the problem of maintaining geometric continuity see [23].

In the above example, each cell keeps track of its age, and divides when it reaches the threshold age. An alternative approach is to use a cell's size itself as the independent variable. In this case, the continuity of the filament length is preserved in a more straightforward fashion, by explicitly partitioning the mother cell in the ratio $SMALLER : (1 - SMALLER)$. The resulting L-system is given below.

```
/* L-system 8 */
#define LEFT -1
#define RIGHT 1
#define x_div 2.1673    /* cell length at division */
#define SMALLER 0.4614  /* fraction of dividing cell length */
#define gr 1.7185       /* growth rate per simulation step */
#define WID 1.0         /* cell width */
Axiom: M(1.0,RIGHT)
M(x,p) --> M(x*gr,p)
decomposition:
M(x,p) : x>=x_div && p == LEFT
        --> M(x*SMALLER,LEFT) M(x*(1-SMALLER),RIGHT)
M(x,p) : x>=x_div && p == RIGHT
        --> M(x*(1-SMALLER),LEFT) M(x*SMALLER,RIGHT)
homomorphism:
M(x,p) : p==LEFT --> L(x,WID)
M(x,p) : p==RIGHT --> R(x,WID)
```

The constant values in L-system 8 have been chosen in terms of the constants of L-system 7, such that both L-systems produce the same developmental sequence. Specifically, $x_{div} = a^{t_{div}}$, $SMALLER = a^{t_s}/x_{div}$ and $gr = a^{dt}$.

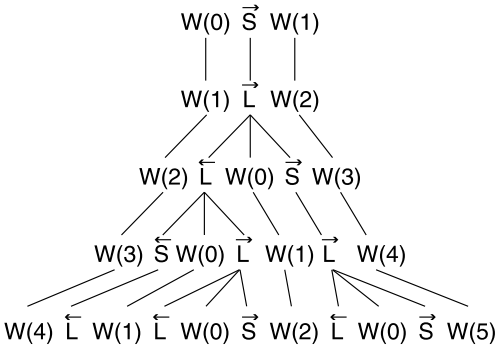

Figure 8: Developmental sequence of *Anabaena* filament (basic model, as in Figure 1), with explicitly represented cell walls. The parameter of cell walls indicated their age.

9 Context-sensitive L-systems

All models considered so far were united by one common tread: the fate of each module was determined by this module itself. A developmental process that proceeds this way is said to be controlled by *lineage* [12, 23, 20]. In reality, however, the fate of modules may also be controlled by *interactions* with their neighbors. In order to give a simple example of this fundamental concept, let us consider a variant of the *Anabaena* filament model, in which asymmetric cell divisions are controlled by properties of cell walls, rather than an attribute inherent in the individual cells. This variant of the model provides an insight into a plausible mechanism of polarity determination in nature.

The modified model treats the filament as a sequence of cells C separated by explicitly represented walls W. Similarly to cells, walls carry the age attribute. Figure 8 shows a developmental sequence corresponding to that of the basic *Anabaena* model (Figure 1). This sequence reveals that, during the asymmetric cell division, the shorter cell is always formed on the side of the older wall. It can be proven that this relationship between cell polarity and the age the walls that separate it from its neighbors also holds true for *Anabaena* models with improved timing (L-systems 2 to 8).

We now use this observation to a *context-sensitive* L-system [11] model, in which the asymmetry of cell division is controlled by the age of the neighboring walls (the context of the cell), rather than cell type or parameters transferred from mother to daughter cell by lineage. Context-sensitive para-

metric have the form

$$left\_context < predecessor > right\_context : condition \longrightarrow successor,$$

where $left\_context$ and $right\_context$ are strings of zero, one or more modules to the left and to the right of the strict *predecessor* [5, 23, 20]. During the derivation, the context modules may affect the applicability and outcome of a production application, but only the strict predecessor module is replaced by the successor. Using this notation, a context-sensitive variant of the *Anabaena* model based on L-system 5 is given below:

```
/* L-system 9 */
#define LEFT -1
#define RIGHT 1
#define t_div 1.0       /* cell age at division */
#define t_s 0.0         /* initial age - short cell */
#define t_l 0.2         /* initial age - long cell */
#define dt 0.7          /* time increment per step */
Axiom: W(0)M(0)W(dt)
W(t) --> W(t+dt)
M(t) : t<t_div --> M(t+dt)
W(tl) < M(t,p) > W(tr) : t>=t_div && tl<tr -->
        M(t-t_div+t_s+dt) W(t-t_div) M(t-t_div+t_l+dt)
W(tl) < M(t,p) > W(tr) : t>=t_div && tr>tl -->
        M(t-t_div+t_l+dt) W(t-t_div) M(t-t_div+t_s+dt)
```

In this model we have not used decomposition rules, and, consequently, the model does not operate properly for large time steps dt. The recursive cell division implemented using decomposition rules in L-systems 6 to 8 is not possible here, because it does not provide context information needed for cell divisions. More advanced time management techniques than the simple time slicing implemented by L-system 9 have been developed in the scope of simulation theory [8] and can be applied to L-systems as well [21], but their discussion is outside the scope of these introductory notes.

10 Conclusions

We have outlined basic types of L-systems introduced by Lindenmayer, context-free (DOL) and context-sensitive L-systems, and some of their extensions useful for model construction. These include the addition of attributes to L-system symbols (parametric L-systems), and decomposition

and interpretation rules. We have illustrated these concepts with a sequence of models of the vegetative segment of an *Anabaena catenula* filament. These examples also present a method for constructing developmental models that operate in arbitrarily small time steps.

References

[1] D. G. Adams. Heterocyst formation in cyanobacteria. *Current Opinoin in Microbiology*, 3:618–624, 2000.

[2] R. Baker and G. T. Herman. Simulation of organisms using a developmental model, parts I and II. *International Journal of Bio-Medical Computing*, 3:201–215 and 251–267, 1972.

[3] K. A. Erstad. L-systems, twining plants, Lisp. Master's thesis, University of Bergen, Norway, January 2002.

[4] J. W. Golden and H.-S. Yoon. Heterocyst formation in *Anabaena*. *Current Opinion in Microbiology*, 1:623–629, 1998.

[5] J. S. Hanan. *Parametric L-systems and their application to the modelling and visualization of plants*. PhD thesis, University of Regina, June 1992.

[6] G. T. Herman and G. Rozenberg. *Developmental systems and languages.* North-Holland, Amsterdam, 1975.

[7] R. Karwowski. Improving the process of plant modeling: The L+C modeling language. Ph.D. thesis, University of Calgary, June 2002, submitted.

[8] W. Kreutzer. *System simulation: Programming styles and languages*. Addison-Wesley, Sydney, 1986.

[9] W. Kurth. *Growth grammar interpreter GROGRA 2.4: A software tool for the 3-dimensional interpretation of stochastic, sensitive growth grammars in the context of plant modeling. Introduction and reference manual*. Forschungszentrum Waldökosysteme der Universität Göttingen, Göttingen, 1994.

[10] A. Lindenmayer. Developmental systems and languages in their biological context. In G. T. Herman and G. Rozenberg, *Developmental systems and languages*. North-Holland, Amsterdam, 1975, pp. 1 – 40.

[11] A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 18:280–315, 1968.

[12] A. Lindenmayer. Developmental systems without cellular interaction, their languages and grammars. *Journal of Theoretical Biology*, 30:455–484, 1971.

[13] A. Lindenmayer. Adding continuous components to L-systems. In G. Rozenberg and A. Salomaa, editors, *L Systems*, Lecture Notes in Computer Science 15, pages 53–68. Springer-Verlag, Berlin, 1974.

[14] A. Lindenmayer. Theories and observations of developmental biology. In R. E. Butts and J. Hintikka, editors, *Foundational problems in special sciences*, pages 103–118. D. Reidel Publ. Co, Dordrecht-Holland, 1977.

[15] A. Lindenmayer. Algorithms for plant morphogenesis. In R. Sattler, editor, *Theoretical plant morphology*, pages 37–81. Leiden University Press, The Hague, 1978.

[16] A. Lindenmayer. Developmental algorithms: Lineage versus interactive control mechanisms. In S. Subtelny and P. B. Green, editors, *Developmental order: Its origin and regulation*, pages 219–245. Alan R. Liss, New York, 1982.

[17] H. B. Lück and J. Lück. Cell number and cell size in filamentous organisms in relation to ancestrally and positionally dependent generation times. In A. Lindenmayer and G. Rozenberg, editors, *Automata, languages, development*, pages 109–124. North-Holland, Amsterdam, 1976.

[18] G. J. Mitchison and M. Wilcox. Rules governing cell division in *Anabaena*. *Nature*, 239:110–111, 1972.

[19] J. L. Peterson. *Petri net theory and the modeling of systems*. Prentice Hall, Englewood Cliffs, 1981.

[20] P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Měch. Visual models of plant development. In G. Rozenberg and A. Salomaa, editors, *Handbook*

of formal languages, Vol. III*: Beyond words*, pages 535–597. Springer, Berlin, 1997.

[21] P. Prusinkiewicz, M. Hammel, and E. Mjolsness. Animation of plant development. Proceedings of SIGGRAPH 93 (Anaheim, California, August 1–6, 1993). ACM SIGGRAPH, New York, 1993, pp. 351–360.

[22] P. Prusinkiewicz, J. Hanan, and R. Měch. An L-system-based plant modeling language. In M. Nagl, A. Schürr, and M. Münch, editors, *Applications of graph transformations with industrial relevance*, Lecture Notes in Computer Science 1779, pages 395–410. Springer-Verlag, Berlin, 2000.

[23] 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.

[24] P. Prusinkiewicz, L. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. Proceedings of SIGGRAPH 2001 (Los Angeles, California, August 12–17, 2001). ACM SIGGRAPH, New York, 2001, pp. 289–300.

[25] P. Prusinkiewicz and W. Remphrey. Characterization of architectural tree models using L-systems and Petri nets. In M. Lebrecque, editor, *L'Arbre - The Tree 2000. Papers presented at the 4th Internatinal Symposium on the Tree at the Montreal Botanical Garden, Aug. 20-25, 2000.*, pages 177–186. Isabelle Quentin and Institut de Recherche en Biologie Végétale, Montreal, 2001.

[26] G. Rozenberg and A. Salomaa. *The mathematical theory of L systems.* Academic Press, New York, 1980.

[27] G. Rozenberg and A. Salomaa. *Handbook of formal languages.* Springer, Berlin, 1997.

[28] d'Arcy Thompson. *On growth and form.* University Press, Cambridge, 1952.

[29] J. H. Woodger. *The axiomatic method in biology.* University Press, Cambridge, 1937. With appendices by A. Tarski and W. F. Floyd.