Developmental Computing
(Extended Abstract)

Przemyslaw Prusinkiewicz

Department of Computer Science, University of Calgary
2500 University Dr. N.W., Calgary, AB T2N 1N4, Canada
pwp@cpsc.ucalgary.ca
http://algorithmicbotany.org

Abstract. Since their inception over forty years ago, L-systems have proven to be a useful conceptual and programming framework for modeling the development of plants at different levels of abstraction and different spatial scales. Formally, L-systems offer a means of defining cell complexes with changing topology and geometry. Associated with these complexes are self-configuring systems of equations that represent functional aspects of the models. The close coupling of topology, geometry and computation constitutes a computing paradigm inspired by nature, termed developmental computing. We analyze distinctive features of this paradigm within and outside the realm of biological models.

Keywords: natural computing, dynamic system with a dynamic structure, L-system, modeling of plant development, geometric modeling.

1 Introduction

Ideas in natural sciences and mathematics often feed off each other (Figure 1). An example of the resulting synergy is the development of calculus, which was prompted by physical questions regarding the motion of celestial bodies and subsequently found applications in many areas of science. Biological questions and processes have also led to the development of new mathematical formalisms and their applications (Table 1). For example, questions regarding the essence of

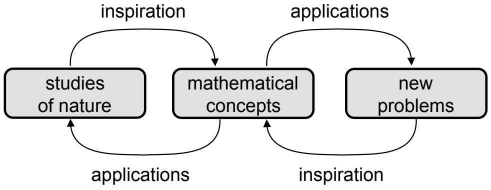

Fig. 1. The progress of ideas in science and mathematics. Studies of nature lead to the development of mathematical concepts that not only apply to the solution of the original problem, but also find applications in other areas of science. These applications, in turn, inspire further development of mathematical concepts and tools.

Table 1. Sample biological phenomena and mathematical formalisms they inspired

Phenomenon	Computational formalism
self-reproduction	cellular automaton
neural system	artificial neural network
evolution	genetic algorithm
DNA recombination	splicing system
molecular processes in a cell	membrane computing

self-reproduction lie at the origin of cellular automata, which subsequently found applications in physics, e.g. in the studies of percolation and diffusion-limited aggregation [29]. Here we focus on the development of multicellular organisms, in particular plants, as an inspiration for computational methods that can be applied to both biological and non-biological problems. The resulting paradigm of developmental computing is characterized by a close link between computation and topology.

2 The Essence of Developmental Computing

From a mathematical perspective, plants have the following characteristics:

1. A plant is a *dynamic system with a dynamic structure* [5,7]: it consists of discrete components (modules), the number and states of which change over time.
2. Development takes place in a *topological space*, which determines the neighboring relations between components.
3. The development is *symplastic*: the neighborhood relations can only be changed as a result of the addition or removal of components (in contrast to animal cells, plant cells do not move with respect to each other).
4. The state of each component advances over time as a function of its own state and the state of the neighboring components, i.e., according to *temporally and spatially local* rules or equations.
5. There is *feedback* between the topology of the system and the state of its components. On one hand, changes in topology (the addition or removal of components and connections between them) are controlled by the state of the components. On the other hand, the state of each component depends on the input from its neighbors.

The formalism of L-systems [12,13] captures these characteristics in the important case of filamentous (linear) and branching structures. L-system identify system components by their state and context (the state of the neighbors). This makes it possible to:

1. Describe indefinitely growing structures using a finite set of rules,
2. Simplify the notation by eliminating the use of indices.

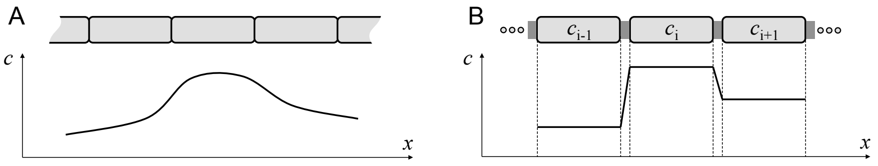

Fig. 2. Continuous (A) and discrete (B) model of diffusion in a filament

Let us consider diffusion in a filament as an example of the difference between traditional and L-system-based description of a physical process. Diffusion is often viewed as a spatially and temporally continuous process (Figure 2A), described in one dimension by the partial differential equation

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}, \tag{1}$$

where c is the concentration of the diffusing substance and D is the diffusion constant. At the cellular level, however, diffusion may be more conveniently approximated as a system of ordinary differential equations, since the main obstacles for diffusion are cell walls (Figure 2B). In this case,

$$\frac{dc_i}{dt} = k[(c_{i-1} - c_i) + (c_{i+1} - c_i)], \quad i = 2, 3, \ldots, n-1. \tag{2}$$

where parameter k depends on the diffusion constant and the geometry of the walls (interface between the cells). Using forward Euler integration, the solution to this system can be expressed as

$$c_i^{t+\Delta t} = k[(c_{i-1}^t - c_i^t) + (c_{i+1}^t - c_i^t)]\Delta t, \quad i = 2, 3, \ldots, n-1, \;\; t = 0, 1, \ldots. \tag{3}$$

The standard notation employed in these equations creates three problems:

1. It requires the use of double indices, and thus is cumbersome;
2. It relies on a systematic indexing of system components, which is difficult to maintain in a growing structure (Figure 3);
3. It is not inherently local: formally correct expressions may refer to components that are distant in space and time (e.g. $c_{2i+100}^{t-200\Delta t}$ is distant from c_i^t).

These problems are eliminated in parametric L-systems [14,21]. For instance, the relations expressed by the system of equations (3) are summarily expressed as a production

$$M(c_l) < M(c) > M(c_r) \rightarrow M\left(c + k(c_l + c_r - 2c)\Delta t\right). \tag{4}$$

Symbol $M(c)$ denotes a cell M with the concentration of diffusing substance c. Spatial relations between system components are specified in an index-free manner using symbols $<$ and $>$ that separate the component being updated (strict predecessor) from its left and right neighbors (context). Temporal precedence is

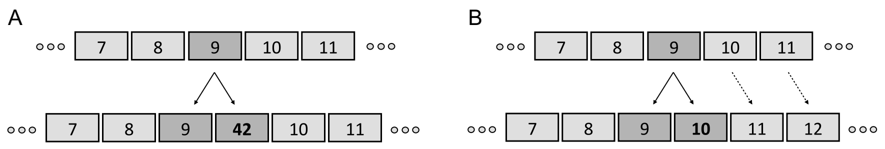

Fig. 3. Problems with component ordering in a growing multicellular structure. In case (A), one of the descendent cells receives a number out or order (42), preventing future use of index arithmetic to identify neighbors. In case (B) the sequence of indices is preserved, but some cells had to be renumbered (10 → 11 and 11 → 12), which makes it difficult to track cell identity over time.

indicated by the production application symbol →. Cell division, which can be expressed by production

$$M(c) \to M(c)M(c), \tag{5}$$

thus automatically reorganizes the set of variables and equations associated with specific components, without incurring the problems depicted in Figure 3. *Automatic changes in the set of variables and the system of equations describing a spatial system, governed by changes in the system topology that are controlled by these variables, constitute the essence of developmental computing* (Figure 4).

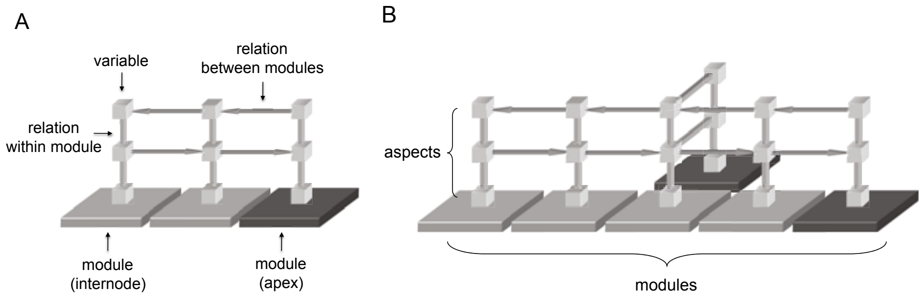

Fig. 4. The essence of developmental computing. At each point in time, the growing system consists of a set of modules with associated variables. Variables within the same or neighboring modules may be related by equations. The structure may grow, for example due to the creation of new modules by the selected "apical" modules (compare structures (A) and (B)). The newly created modules introduce new variables to the description of the system, which are automatically related to others by the updated neighborhood relations. The set of variables and equations describing the system thus grows with the structure. This growth is in turn controlled by the state of the system (values of the variables).

3 Progress of Ideas

In the context of the original biological applications, developmental computing is the paradigm behind diverse simulation models of plant functioning, which include: growth regulation and long-distance signaling by hormones [9,21]; acquisition, transport and allocation of water, nutrients, and sugars, e.g. [1]; bending of branches due to gravity [2,10,20]; and responses to pathogens or insects [18]. Nevertheless, many modeling problems remain open and require further advancements of the mathematical and computational concepts.

1. *Extensions of L-system-based programming constructs and languages.* The required extensions include programming support for multiscale modeling [8] and for aspect-oriented programmingi [11]. In the latter case, the objective is to provide constructs for easily incorporating previously modeled aspects of plant behavior (e.g., responses to mechanical forces) into new models.
2. *Extensions of developmental computing to surfaces and volumes.* Although several software packages capable of simulating developing tissues and organs exist, e.g. [6,24,25,26], the question of what mathematical foundations, data structures and programming constructs are most appropriate as the basis for simulations in two and three dimensions (i.e., of systems represented by discrete 2- and 3-manifolds [3]) remains open. These data structures and constructs should conveniently represent the changing topology and geometry of the modeled systems, act as dynamic domains in which simulations can take place, and allow easy updates to the topology, geometry, and systems of equations during the simulations.
3. *Numerical methods for developmental computing.* Systems of equations describing developing structures continually change. One approach to solving these equations is to reformulate and resubmit them to a standard solver each time they are changed. A potentially more efficient alternative is to store and manipulate quantities at their geometrically meaningful location [3]; in other words, to reformulate numerical methods so that they operate directly on the dynamic data structures that represent the system topology [27,28]. Examples include a method for solving systems of linear equations in growing tree structures directly using L-systems [4,17], based on an adaptation of Gaussian elimination to tree graphs [15].

Consistent with Figure 1, many algorithms outside the domain of plant modeling and simulations benefit from recasting in terms of L-systems or general notions of developmental computing [19]. Geometric examples include the generation of fractals [16] and space-filling curves [22], as well as the generation of smooth subdivision curves [23] and surfaces [26]. Similar to biological models, these algorithms involve a growing number of components and relations between them, and thus benefit from the conceptual and notational clarity afforded by developmental computing.

4 Conclusions

Modeling and simulation of the development of multicellular organisms is characterized by:

1. a changing number of discrete components, state variables and equations that describe the system at each point of time, and
2. the spatio-temporal nature of the models.

The paradigm of developmental computing consists of using the spatio-temporal characteristic of the system under consideration to address computational problems created by the changing number of its components. Specifically, the state variables of individual components are linked to each other according to the neighborhood relations between the components. These relations are automatically updated following changes in the system topology. This approach makes it possible to organize computation in a conceptually elegant manner, making the models easy to specify and implement. In particular, identification of system components with indices, cumbersome in the case of growing spatial structures with arbitrary topologies, is avoided.

Parametric L-systems capture the concept of developmental computing in linear and branching structures, and have inspired current extensions of developmental computing to 2- and 3-dimensional multicellular structures. Current work also includes programming constructs that support efficient construction of complex developmental models, and numerical methods tailored to the framework of developmental computing.

Applications of developmental computing span both the modeling of plants and algorithms outside the scope of biological models. The latter include dynamic geometric modeling problems, such as the construction of subdivision curves and surfaces, and algorithms that are not inherently spatial, but can be recast in spatial terms. Despite a 40-year history [12] and the crucial impact of L-systems on plant modeling, developmental computing remains a fascinating, application-rich area bringing together computation and topology.

Acknowledgments. I would like to thank all the colleagues and students with whom I have had the pleasure of discussing the ideas outlined in this abstract. In particular, I acknowledge discussions with Jean-Louis Giavitto, Olivier Michel, and Eric Mjolsness, who have been working on similar concepts in the context of their modeling systems. I also thank the Natural Sciences and Engineering Research Council of Canada for financial support of the underlying research.

References

1. Allen, M., Prusinkiewicz, P., DeJong, T.: Using L-systems for modeling source-sink interactions, architecture and physiology of growing trees: the L-PEACH model. New Phytologist, 869–880 (2005)
2. Costes, E., Smith, C., Renton, M., Guédon, Y., Prusinkiewicz, P., Godin, C.: MAppleT: simulation of apple tree development using mixed stochastic and biomechanical models. Functional Plant Biology 35, 936–950 (2008)
3. Desbrun, M., Kanso, E., Tong, Y.: Discrete differential forms for computational modeling. In: SIGGRAPH 2006 Course Notes on Discrete Differential Geometry (2006)

4. Federl, P., Prusinkiewicz, P.: Solving differential equations in developmental models of multicellular structures expressed using L-systems. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004, Part II. LNCS, vol. 3037, pp. 65–72. Springer, Heidelberg (2004)
5. Giavitto, J.-L., Godin, C., Michel, O., Prusinkiewicz, P.: Computational models for integrative and developmental biology. In: Proceedings of the Autrans seminar on modelling and simulation of biological processes in the context of genomics, pp. 65–72 (2002)
6. Giavitto, J.-L., Michel, O.: MGS: A programming language for the transformation of topological collections. Research Report 61-2001, CNRS - Université d'Evry Val d'Esonne, Evry, France (2001)
7. Giavitto, J.-L., Michel, O.: Modeling the topological organization of cellular processes. BioSystems 70, 149–163 (2003)
8. Godin, C., Caraglio, Y.: A multiscale model of plant topological structures. Journal of Theoretical Biology 191, 1–46 (1998)
9. Janssen, J.M., Lindenmayer, A.: Models for the control of branch positions and flowering sequences of capitula in Mycelis muralis (L.) Dumont (Compositae). New Phytologist 105, 191–220 (1987)
10. Jirasek, C., Prusinkiewicz, P., Moulia, B.: Integrating biomechanics into developmental plant models expressed using L-systems. In: Spatz, H.-C., Speck, T. (eds.) Plant biomechanics 2000. Proceedings of the 3rd Plant Biomechanics Conference, pp. 615–624. Georg Thieme Verlag, Stuttgart (2000)
11. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., Irwin, J.: Aspect-oriented programming. In: Aksit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997)
12. Lindenmayer, A.: Mathematical models for cellular interaction in development, Parts I and II. Journal of Theoretical Biology 18, 280–315 (1968)
13. Lindenmayer, A.: Developmental systems without cellular interaction, their languages and grammars. Journal of Theoretical Biology 30, 455–484 (1971)
14. Lindenmayer, A.: Adding continuous components to L-systems. In: Rozenberg, G., Salomaa, A. (eds.) L Systems. LNCS, vol. 15, pp. 53–68. Springer, Heidelberg (1974)
15. Parter, S.: The use of linear graphs in Gauss elimination. SIAM Review 3, 119–130 (1961)
16. Prusinkiewicz, P.: Graphical applications of L-systems. In: Proceedings of Graphics Interface 1986 — Vision Interface 1986, pp. 247–253 (1986)
17. Prusinkiewicz, P., Allen, M., Escobar-Gutierrez, A., DeJong, T.: Numerical methods for transport-resistance source-sink allocation models. In: Vos, J. (ed.) Functional-structural modeling in crop production, pp. 123–137. Springer, Dordrecht (2007)
18. Prusinkiewicz, P., Hammel, M., Hanan, J., Měch, R.: L-systems: from the theory to visual models of plants. In: Michalewicz, M.T. (ed.) Plants to ecosystems. Advances in computational life sciences I, pp. 1–27. CSIRO Publishing, Melbourne (1997)
19. Prusinkiewicz, P., Hanan, J.: L-systems: from formalism to programming languages. In: Rozenberg, G., Salomaa, A. (eds.) Lindenmayer systems: Impacts on theoretical computer science, computer graphics, and developmental biology. Springer, Berlin (1992)
20. Prusinkiewicz, P., Karwowski, R., Lane, B.: The L+C plant-modeling language. In: Vos, J. (ed.) Functional-structural modeling in crop production, pp. 27–42. Springer, Dordrecht (2007)

21. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer, New York (1990); With Hanan, J.S., Fracchia, F.D., Fowler, D.R., de Boer, M.J.M., Mercer, L.
22. Prusinkiewicz, P., Lindenmayer, A., Fracchia, F.D.: Synthesis of space-filling curves on the square grid. In: Peitgen, H.-O., Henriques, J.M., Penedo, L.F. (eds.) Fractals in the fundamental and applied sciences, pp. 341–366. North-Holland, Amsterdam (1991)
23. Prusinkiewicz, P., Samavati, F., Smith, C., Karwowski, R.: L-system description of subdivision curves. International Journal of Shape Modeling 9(1), 41–59 (2003)
24. Shapiro, B., Mjolsness, E.: Developmental simulations with Cellerator. In: Second International Conference on Systems Biology (ICSB), pp. 342–351 (2001)
25. Smith, C.: On vertex-vertex systems and their use in geometric and biological modeling. PhD thesis, University of Calgary (January 2006)
26. Smith, C., Prusinkiewicz, P., Samavati, F.: Local specification of subdivision algorithms. In: Pfaltz, J.L., Nagl, M., Böhlen, B. (eds.) AGTIVE 2003. LNCS, vol. 3062, pp. 313–327. Springer, Heidelberg (2004)
27. Smith, R.: Simulation models of phyllotaxis and morphogenesis in plants. PhD thesis, University of Calgary (January 2007)
28. Sowell, E., Haves, P.: Numerical performance of the spark graph-theoretic simulation program. In: Proc. IBPSA Building Simulation 1999, p. 14 (2001)
29. Wolfram, S.: A New Kind of Science. Wolfram Media, Champaign (2002)