Structured Dynamical Systems

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

Abstract

This note introduces the notion of structured dynamical systems, and places L-systems in their context.

1 Basic definitions

Many natural phenomena can be modeled as *dynamical systems*. At any point in time, a dynamical system is characterized by its *state*. A state is represented by a set of *state variables*. For example, in the description of planetary motions around the sun, the set of state variables may represent positions and velocities of the planets. Changes of the state over time are described by a *transition function*, which determines the next state of the system as a function of its previous state and, possibly, the values of external variables (input to the system). This progression of states forms a *trajectory* of the system in its *phase space* (the set of all possible states of the system).

Mathematical objects with diverse properties can be considered dynamical systems. For instance, state variables may take values from a continuous or discrete domain. Likewise, time may advance in continuous or discrete steps. Examples of dynamical systems characterized by different combinations of these features are listed in Table 1.

Table 1: Some formalisms used to specify dynamical systems according to the discrete or continuous nature of time and state variables.

C: continuous, D: discrete.	ODE	Iterated Mappings	Finite Automata
Time	C	D	D
State	C	C	D

[1]Adapted from: J.-L Giavitto, C. Godin, O. Michel and P. Prusinkiewicz, *Computational Models for Integrative and Developmental Biology*, LaMI Rapport de Recherche N^o **72-2002**, March 2002, Section 2.

In simple cases, trajectories of dynamical systems may be expressed using mathematical formulas. For example, the ODE (ordinary differential equation) describing the motion of a mass on a spring has an analytical solution expressed by a sine function (linear spring, in the absence of friction and damping). In more complex cases, analytic formulas representing trajectories of the system may not exist, and the behavior of the system is best studied using computer simulations.

By their nature, simulations operate in discrete time. Models initially formulated in terms of continuous time must therefore be discretized. Strategies for discretizing time in a manner leading to efficient simulations have extensively been studied in the scope of simulation theory, *e.g.* [6].

Dynamical systems with apparently simple specifications may have very complex trajectories. This phenomenon is called *chaotic behavior*, *c.f.* [13], and is relevant to biological systems (for example, in populations models) [10, 11].

2 Structured dynamical systems

Many dynamical systems can be decomposed into parts. The advancement of the state of the whole system is then viewed as the result of the advancement of the state of its components. For example, a developing plant can be described in terms of its functional units (modules), such as apices, internodes, leaves, and flowers (reviewed in [15]). Similarly, the operation of a gene regulation network can be described in terms of interactions between individual genes, and gene activities in interacting cells (e.g. [12, 14, 18, 19]).

Formally, we use the term *structured dynamical system* to denote a dynamical system divided into component subsystems (units). The set of state variables of the whole system is the Cartesian product of the sets of state variables of the component subsystems. Accordingly, the state transition function of the whole system can be described as the product of the state transition functions of these subsystems. Similarly to non-structured systems, structured dynamical systems can be defined assuming continuous or discrete state variables and time. In addition, the components can be arranged in a continuous or discrete manner in space. Some of the formalisms resulting from different combinations of these features are listed in Table 2.

Time management is an important issue in the modeling and simulation of structured systems [8]. For example, state transitions may occur *synchronously* (simultaneously in all components) or *asynchronously*. Furthermore, efficient simulation techniques may assume different rates of time

Table 2: Some formalisms used to specify structured dynamical systems according to the continuous or discrete nature of space, time, and state variables of the components. The heading "Numerical Solutions" refers to explicit numerical solutions of partial differential equations and systems of coupled ordinary differential equations.

C: continuous D: discrete	PDE	Coupled ODE	Numerical Solutions	Cellular Automata
Space	C	D	D	D
Time	C	C	D	D
States	C	C	C	D

progression in different components [5].

In many cases, the transition function of each subsystem depends only on a (small) subset of the state variables of the whole system. If the components of the system are discrete (i.e., excluding partial differential equations, or PDEs), these dependencies can be depicted as a *directed graph*, with the nodes representing the subsystems and the arrows indicating the inputs to each subsystem. We say that this graph defines the *topology* of the structured dynamical system, and call *neighbors* the pairs of subsystems (directly) connected by arrows.

The topology of a structured dynamical system may reflect its *spatial organization*, in the sense that only physically close subsystems are connected. A dynamical system with this property is said to be *locally* defined. Locality is an important feature of systems that model physical reality, because physical means of information exchange ultimately have a local character (e.g., transport of signaling molecules between neighboring cells). On the other hand, physically-based models need not to be rigorously local. For example, when modeling plants, it may be convenient to assume that higher branches cast shadow on lower branches without simulating the local mechanism of light propagation through space.

When the number of components in a structured dynamical systems is large, the exhaustive listing of all connections between the components becomes impractical or infeasible. This limitation can be overcome in several ways. For example, if the components are arranged in a regular pattern, the neighbors of each component need not to be listed explicitly. This is the case of *cellular automata* (*e.g.* [17, 20]), in which cells are arranged in a

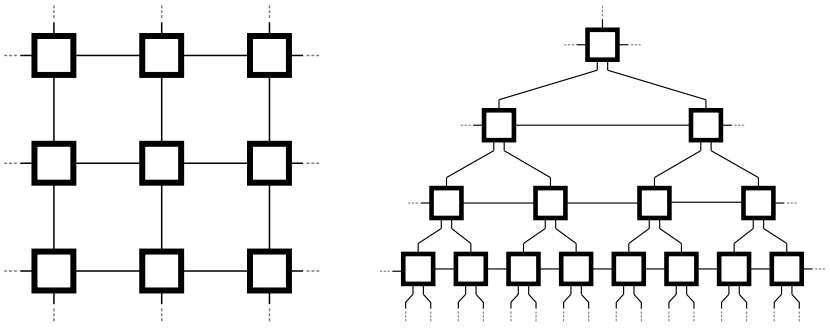

Figure 1: Two examples of regular patterns of connections in structured dynamical systems: Euclidean grid, in which each cell has four neighbors (von Neumann neighborhood, left), and hyperbolic grid, in which each cell has five neighbors (right).

square grid. The neighbor of each cell (excluding the cells, if present), can be referred to in a uniform way, using the directions North, South, East and West. Cellular automata operating in the hyperbolic plane [9] and *group-based fields* [1] are generalizations of this idea, allowing for a wider range of connecting patterns (Figure 1). Large structures can also be defined by *simulated development*, as it is discussed in the next section.

3 Dynamical systems with a dynamic structure

A developing multicellular organism can be viewed as a dynamical system in which not only the values of state variables, but also the *set* of state variables and the state transition function change over time. These phenomena can be captured using an extension of structured dynamic systems, in which the set of subsystems or the topology of their connections may dynamically change. We call these systems *dynamical systems with a dynamic structure* [1], or $(\mathrm{DS})^2$-systems in short.

For example, let us consider a multicellular plant structure, defined at the level of individual cells. A large network of interconnected cells can by gradually created by simulating the process of cell division. When a cell divides, the topology of this network is adjusted to:

- remove connections (neighborhood relations) between the mother cell and the rest of the organism,
- create connections between the daughter cells,

- insert connections between the daughter cells and the remainder of the system.

Other examples of natural processes that can be viewed as $(DS)^2$-systems include:

- chemical reactions, considered as interactions between molecules,
- developing plants, in which apical meristems create new meristems, branch segments, leaves, and flowers,
- developing ecosystems, in which plants and animals reproduce, interact with each other, and die.

The concept of $(DS)^2$-systems can also be extended to more abstract, mathematical constructs, such as:

- fractal generation using Koch construction,
- generation of curves and surfaces using subdivision and pyramid algorithms [4],
- numerical solution of PDEs using adaptive numerical methods, in which the discretization of space changes over time,
- parallel algorithms, in which the number of interconnected computing elements changes as the computation proceeds.

In the above examples, time is the independent variable that drives the system dynamics. It is also possible to consider $(DS)^2$-systems in which the independent variable has spatial character. These include multiresolution representations, in which the level of detail is controlled by the scale of observations.

From the computer science point of view, simulation of dynamical systems with a dynamical structure raises the problem of finding a programming paradigm (or the modeling language) well suited to the specification of such systems. The key difficulty is that, in $(DS)^2$-system, not only the values of variables that describe the system, but also the entire set of variables, and equations that relate them, change over time. If the number of these changes is small, they can be specified explicitly. In general, however, we need a formalism that supports these changes in an automatic manner. Table 3 presents some of the existing formalisms for handling systems that operate in discrete space.

Table 3: Some formalisms used for the modeling of (DS)^2, according to the underlying topology of space.

Topology	Multiset	Sequence	Array	Graph
Formalism	multiset rewriting	L-systems	cellular automata	map L-systems, graph grammars, MGS, VV-systems, Cellerator

In this table, the first line gives the type of the topology used to connect the subcomponents of a system. In a *multiset*, all elements are considered to be connected to each other. In a *sequence*, elements are ordered linearly; this case includes lists and extends to tree-like structures. *Array* structures represent regular neighborhoods; for example, the lattice shown in Figure 1. In this case, addition of new elements to the structure may only occur at its boundary. Finally, *graphs* are used to define arbitrary connections between discrete components. Formalisms for describing (DS)^2 operating on graphs are less well developed than the formalisms operating on multisets, sequences, and arrays. Current research includes MGS (acronym for *un Modèle Général de Simulation de système dynamique* [1, 2, 3]) and vertex-vertex systems, outlined later in these notes. Cellerator [16] is a recent example of a practical simulation system operating on dynamically reconfigured graphs.

4 Conclusions

Structured dynamical systems consist of interconnected components. In dynamical systems with a dynamic structure, or (DS)^2, the set of components and their connections may change over time. This raises the problem of characterizing these changes as a part of a (DS)^2 specification. One possibility is to assume that the (DS)^2 components exist in some topological space, and use proximity of components in this space to define their connections. These connections define, how the variables in one component are related to the variables in its neighbors. Within this general framework, several formalisms for modeling (DS)^2 exist. Specifically, L-systems [7] make it possible to model dynamically reconfigurable structures with linear or branching topology.

References

[1] J.-L. Giavitto and O. Michel. MGS: A programming language for the transformation of topological collections. Research Report 61-2001, CNRS - Université d'Evry Val d'Esonne, Evry, France, 2001.

[2] J.-L. Giavitto and O. Michel. Data structures as topological spaces. In *Proceedings of the 3nd International Conference on Unconventional Models of Computation UMC02*, volume 2509, pages 137–150, 2002. Lecture Notes in Computer Science.

[3] J.-L. Giavitto and O. Michel. The topological structures of membrane computing. *Fundamenta Informaticae*, 49:107–129, 2002.

[4] R. Goldman. *Pyramid algorithms. A dynamic programming approach to curves and surfaces for geometric modeling.* Morgan Kaufmann, San Francisco, 2003.

[5] D. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425, July 1985.

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

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

[8] N. A. Lynch. *Distributed algorithms.* Morgan Kauffman, Los Altos, CA, 1996.

[9] M. Margenstern. Celular automata in the hyperbolic plane. Report 99-103, Groupe d'Informatique Fondemantale de Metz, Université de Metz, 1999.

[10] R. M. May. Biological population models obeying difference equations: Stable points, stable cycles, and chaos. *Journal of Theoretical Biology*, 51:511–524, 1975.

[11] R. M. May. Simple mathematical models with very complicated dynamics. *Nature*, 261:459–467, 1976.

[12] E. Mjolsness, D. Sharp, and J. Reinitz. A connectionist model of development. *Journal of Theoretical Biology*, 152:429–453, 1991.

[13] H.-O. Peitgen, H. Jürgens, and D. Saupe, editors. *Chaos and fractals. New frontiers of science*. Springer-Verlag, New York, 1992.

[14] J. Reinitz and D. Sharp. Mechanism of *eve* stripe formation. *Mechanisms of Development*, 49:133–158, 1995.

[15] P. M. Room, L. Maillette, and J. Hanan. Module and metamer dynamics and virtual plants. *Advances in Ecological Research*, 25:105–157, 1994.

[16] B. Shapiro and E. Mjolsness. Developmental simulations with Cellerator. In *Proceedings of the Second International Conference on Systems Biology*, pages 342–351, 2001.

[17] T. Toffoli and N. Margolus. *Cellular automata machines: A new environment for modeling*. MIT Press, Cambridge, Massachusetts, 1987.

[18] G. von Dassow, E. Meir, E. Munro, and G. Odell. Ingeneue manual v. 0.7. `http://www.beakerware.com/ingeneue/`.

[19] G. von Dassow, E. Meir, E. Munro, and G. Odell. The segment polarity network is a robust developmental module. *Nature*, 406:188–192, 2000.

[20] S. Wolfram. *A new kind of science*. Wolfram Media, Champaign, IL, 2002.