EVOLUTIONARY MODELING USING STATE-CONDITIONED L-SYSTEMS

Joseph Berleant (Little Rock Central High, Little Rock, AR, US)
Daniel Ashlock (University of Guelph, Guelph, Ontario, Canada)
Daniel Berleant (University of Arkansas at Little Rock, Little Rock, AR, US)

ABSTRACT: Plant modeling is used extensively in plant structure research and in creating virtual plants for virtual environments. Lindenmayer systems (L-systems), which consist of a set of replacement rules that iteratively modify a string, are an important tool in plant modeling, and have already been used to model the structures of plants like corn and soy. In this study, the L-system concept was combined with a finite state machine (FSM), a system consisting of a set of states with rules that dictate movement among the states. Each FSM state is associated with its own L-system. Although L-system and FSM hybrids have relatively rich and complex behavior, no method to create such hybrids that model specific plants exist. Effective use of these hybrids requires such a method.

This study examined the potential of using a set of mutation strategies to change an L-system/FSM hybrid to model a predetermined plant. An initial hybrid that represented a single diagonal line reminiscent of a slanted stalk was the starting point, and four mutation strategies were specified. Twenty-five mutated hybrids were each produced by randomly applying one of the four mutation strategies to the parent hybrid, and the hybrid that best resembled the target plant was chosen by the user for our human-in-the-loop selection strategy. This was used as the basis to produce 25 more mutations. This process continued until further mutations were unable to produce a hybrid with a significantly greater resemblance to the chosen plant.

The chosen mutation strategies successfully mutated the simple initial L-system/FSM hybrid into one resembling the target plant. This method's success shows that using mutation strategies as a way to produce an L-system/FSM hybrid that resembles a specific plant is feasible. Different mutation strategies could potentially produce even better results.

INTRODUCTION

The study of plant structures in real and virtual environments often uses plant modeling techniques. Lindenmayer systems (L-systems) are capable of modeling a variety of plant structures because of their ability to produce potentially recursive branching. An L-system consists of a group of rules that modify a string of characters over a series of iterations (Prusinkiewicz and Lindenmayer 1991a; Lindenmayer 1968). They have previously been used to model the growth and structure of certain plants, including soybeans (Sun and Tang 2008) and corn (Fournier and Andrieu 1998).

A standard L-system attempts to model an entire plant with a single set of replacement rules. In practice, features like developmental transitions from woody to herbaceous tissue and from stems to leaves or flowers suggests modifying the L-system algorithm with sentinels or calls to separate algorithms for leaves, flowers and other tissue structures (Prusinkiewicz and Lindenmayer 1991b). A state-conditioned L-system has the potential to implement this modeling requirement directly as well as to support more complex and varied branching structures.

In this study the L-system and the finite state machine (FSM, see e.g. Hopcroft and Ullman 1979) concepts were combined. Each state in the machine contains a distinct L-system, which helps provide the system with flexibility and expressiveness in modeling plant structures. Fusing L-systems with a finite state control structure generalizes the notion of an L-system in a manner that includes earlier techniques (Prusinkiewicz and Lindenmayer 1991a) that use sentry variables to create internal system states.

The combination of L-systems and FSMs has been explored in little previous work. Badr (1998, 2001) et al. (1999) explored using L-systems with a combination of FSMs and Petri nets for intelligent system design. In the plant domain, however, despite the demonstrated potential of L-systems in plant studies, the effective combination of L-systems and FSMs does not seem to have been addressed previously. In order to use this new type of L-system effectively in plant structure research, a way to produce a combined L-system and FSM model for a specific plant is needed. A mutation-based strategy can be useful (Droop and Hickinbotham 2011). The mutation-based algorithm we explored for creating state-conditioned L-systems that provide progressively improved models of a plant is a simple evolutionary strategy (Beyer 2001). The present study explains this novel approach, and gives its application in an example. The results suggest the potential value of developing this approach for use in additional and potentially more complex problems in plant modeling.

MATERIALS AND METHODS

User interface. Fig. 1 shows the output and interface screen. The main portion shows 25 different mutations, comprised of one mutation selected by the user from the preceding display of 25 mutations, plus 24 new mutations obtained by mutating that selection. From this screen, the user may in turn select one of the 25 shown by clicking on it. (One has been selected in the figure; it is shown with a highlighted border.) The "Expand" button near the bottom of the screen will zoom into the selected mutation and toggle the name of the "Expand" button to "Mutate." From the zoomed image the user may click the "Mutate" button to generate a new screen of 25 images.

Data representation and processing. Other elements of Fig. 1 provide information about the evolving plant model. "State Info" lists the states in the FSM for the selected image. Directly below that, the name of the highlighted state is given, and can be changed for mnemonic purposes if desired. While the mutation process often creates new states automatically, the "Add State" button permits manual creation of a new state in the FSM for test purposes. Below that is the "Variable Info" box. This lists the characters in the L-system. Clicking any of them highlights it. The replacement string for the highlighted variable is shown directly below the "Variable Info" box, and below that, the numerical arguments associated with the variable. These arguments are used to help define drawing actions, and can be mutated. The "Destination State" box shows the state that the FSM will transition to when highlighted character ("+" in the figure) is encountered by the L-system string interpreter. An action is associated with that character, indicated in the radio button list.

A small window near the bottom in Fig. 1 holds the number 4. This software parameter may be changed by the user for experimental purposes, and indicates the number of iterations through the string, starting from the initial string, in which character replacement is performed by the L-system character replacement rules. The drawing of the images occurs after the iteration is complete, so each image reflects the same number of iterations involved in producing the string that the interpreter draws the image from. More iterations yield longer strings and, potentially, more complex plant images. The string length depends on the substitution rules invoked, which in turn depend on the state when a particular character is being processed, and indirectly on the mutation history because that can change the substitution rules associated with a given state as well as creating new states. However, average string length will tend to naturally increase exponentially with the number of iterations.

Since each drawing is constructed by interpreting the string resulting from n expansion iterations (n=4 in Fig. 1), the computation involved in the drawing process also increases exponentially with the number of iterations. Thus overall system operation is governed by a time complexity of $O(x^n)$, where x is by the average number of characters in a character replacement rule and n is the number of iterations. Consequently the system is quite sensitive to the value of n. In practice we have found that values of n over 7 lead to significant delays in image generation that interfere with the interactive nature of the user interface.

L-system string interpretation. It is useful to understand how an L-system is represented in the software. Every character (i.e. variable) in each L-system is associated with four values: a replacement string, a destination state, a number representing an action associated with that variable, and a numerical magnitude m used by some of the actions. There are five actions, two for drawing, two for state handling, and null. The actions are 1) draw a line m pixels long, 2) turn m degrees clockwise, 3) “store state” by pushing the current status of the drawing mechanism onto a stack, 4) “restore state” by popping a status from the aforementioned stack and assigning it to the drawing mechanism, and 5) do nothing. Only two actions use their associated magnitude m, potentially allowing a magnitude mutation to have no effect until the action itself mutates later into an action that does use its magnitude parameter.

Mutations. The software is initialized with a simple initial L-system/finite state machine hybrid. The software then applies successive mutations. The evolutionary goal was to obtain an L-system/FSM hybrid that resembled a predetermined plant. Four mutation strategies were implemented in software and a simple initial hybrid system were created. The initial hybrid system expresses a single diagonal line. From this, 25 child mutations are created and drawn on a 5x5 grid on the screen (Fig. 1). Each mutation is obtained by randomly applying one of the 4 mutation strategies, each of which randomly produces a mutation from among a large space of possibilities. From these 25 mutations, we applied a human-in-the-loop selection strategy, in which the best picture of the predetermined plant was user-selected, then used as the parent of 25 more mutations. This may be referred to as a (1,25)-ES algorithm (Beyer 2001).

The process of selecting among mutating requires a fitness function. In the case of human-in-the-loop systems fitness determinations are made by humans. In the present system these determinations were based on a prioritized list of criteria. The primary criterion was to choose the mutant with a branching morphology most similar to the target plant, in terms of which branches go on to branch again themselves. The secondary criterion, used in cases of ties, was based on which mutation resulted in angles more similar to the target plant, and the tertiary criterion used when there was still no clear best mutation, was to examine the relative lengths of the branches for similarity to the target. The process of successive iterations of mutations followed by choosing one of them continued until little or no further improvement was noted.

Mutation Strategies. The four mutation strategies are closely tied to the representation of the L-system and FSM data structures in the program code. The mutation strategies were as follows.

1) Choose two L-system variables and randomly interchange either their a) magnitudes, b) drawing actions, c) replacement strings, or d) destination states.
2) Randomly do one of the following:
    a. change one variable's magnitude by adding a random integer from -10 to +10,
    b. change the drawing action of one variable to a randomly chosen new action,
    c. change one variable's entire replacement string to be another randomly chosen (1-character) variable in the L-system, or
    d. change the destination state of one variable to a new state in the FSM.
3) Split one variable's replacement string at a random location and exchange the beginning and ending portions of the string.
4) Append a randomly chosen variable to the end of another variable's replacement string.

Initial System. The initial L-system/FSM hybrid contained a two-state FSM. Each state had its own associated L-system of 6 variables designated by 6 arbitrary characters. The axiom (i.e., initial string) of the system contained one character, an 'X.' Table 1 shows the starting values for each variable of the six in each state's L-system.

State 0				
Variable	Replacement String	Destination State	Drawing Action	Magnitude
+	+	0	Turn	25
-	-	0	Turn	-25
[	[	0	Store status	0
]	]	0	Restore status	0
F	FF	0	Draw	5
X	FX	0	Do nothing	0

(*a*)

State 1				
Variable	Replacement String	Destination State	Drawing Action	Magnitude
+	+	1	Turn	25
-	-	1	Turn	-25
[	[	1	Store status	0
]	]	1	Restore status	0
F	FF	1	Draw	5
X	FX	1	Do nothing	0

(*b*)

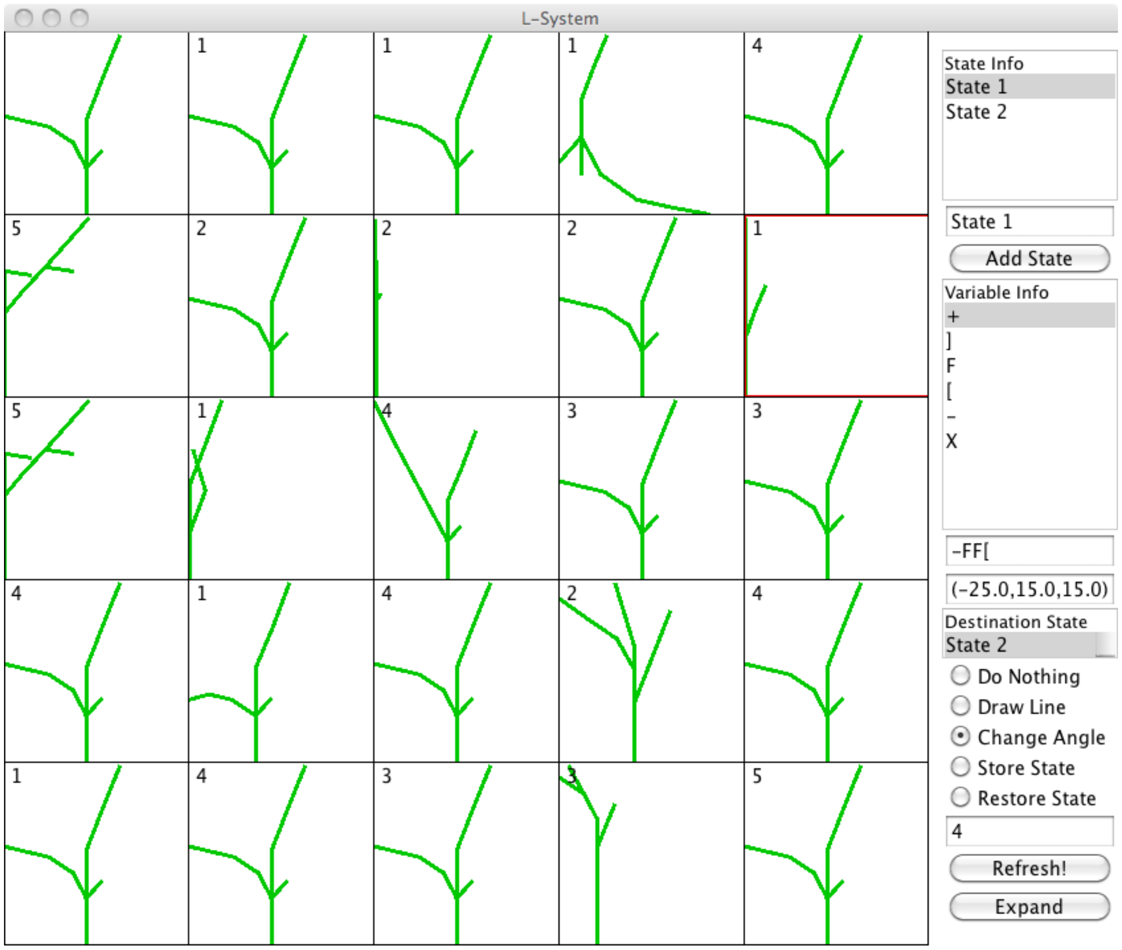
TABLE 1. Variable rules for (*a*) state 0, and (*b*) state 1 in the initial system.

RESULTS AND DISCUSSION

The L-system/FSM hybrid systems used in this study draw in two dimensions. Thus, the plant chosen for this study was a small branch of a northern white cedar (*Thuja occidentalis*), because of its two-dimensional character.

The initial system drew a single diagonal line, which was mutated and selected as described earlier until further iterations did not appear to significantly increase the model's resemblance to the chosen *Thuja occidentalis* branch. Note the resemblance in tilt and general layout of the main branches. Although the number of iterations varies from test to test, a typical run consisted of about 100 iterations. For example, Fig. 3 shows a result that took 119 iteration. Many mutations do not produce an immediately visible effect on the drawn image, but are unmasked later by future mutations. Bends typically appear after about 10 iterations, but branches often take about 50 iterations. As the images progressively resemble the branch more strongly, convergence toward it slows.

We found that mutations to the L-system replacement strings (mutation strategies 1c, 2c, 3, & 4 above) were especially potent in producing changes to the images. Although significant improvements to the system with continued iterations cease after a point, mutation strategies different from the ones outlined previously might have the potential to produce a final system that models the chosen plant even more closely.



State Info
State 1
State 2

State 1

Variable Info
+
]
F
[
-
X

-FF[

(-25.0,15.0,15.0)

Destination State
State 2

Do Nothing
Draw Line
[✓ Change Angle]
Store State
Restore State

4

FIGURE 1. Twenty-five mutated systems.

While an L-system with a larger rule set can sometimes simulate state conditioned L-systems, the question arises of whether the set of structures describable by a state conditioned L-system is larger than the set describable with either L-systems or finite state machines alone. Further work to resolve this interesting and potentially significant question is needed.

CONCLUSION

While L-systems are well known to be useful in modeling plant structure, the combination of L-systems and finite state machines in hybrid systems has been left remarkably unstudied. In this study we have demonstrated the possibility that this combination has value for plant modeling. Showing the feasibility of hybrid systems, as we have done in this investigation, suggests that continued research on hybrid combinations of L-systems and FSMs for plant modeling may prove fruitful.

Availability and Requirements. The software written for this study runs under Java Version 1.6.0 or later. Testing was performed on a 2.00 GHz Intel T2500 CPU based PC running Windows XP Service Pack 3. Source code and compiled bytecode are available from the authors upon request. The system was written in Java, and requires Java 1.6.0 or higher. It is run by typing a Java command line from a terminal window on Windows or a Mac. Software available upon request.

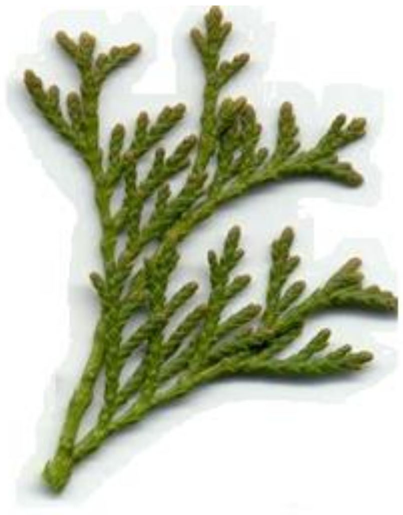


FIGURE 2. The *Thuja occidentalis* branch.

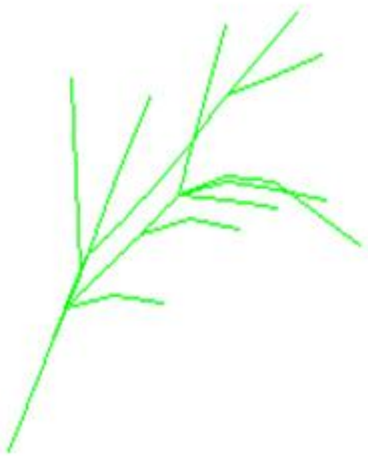


FIGURE 3. Model (produced after 119 iterations).

ACKNOWLEDGEMENTS
JB created the program, performed the study, and drafted the manuscript. DB revised the manuscript and provided technical advice. DA posed the problem. All authors participated in the design of the study and have read, reviewed and approved the final manuscript.

REFERENCES

1. Badr, A. 2001. “A Formal Analysis of the Computational Dynamics in GIGANTEC”. *Mathware & Soft Comp.* Vol. 8:137-152.
2. Badr, A. 1998. *A Hybrid Framework for Optimal System Design*. PhD Thesis, Department of Computer Science, University of Cairo, Egypt.
3. Badr, A., Farag, I., Eid, S. 1999. “A Hybrid Evolutionary Approach to Intelligent System Design”. *Mathware & Soft Comp.* Vol. 6:5-32.
4. Beyer, H.-G. 2001. *The Theory of Evolution Strategies*, Springer, New York, NY.
5. Droop, A.P., and S.J. Hickinbotham. 2011. “Properties of Biological Mutation Networks and their Implications for Artificial Life”. *Art. Life* Vol. 17:353-364.
6. Fournier, C., and B. Andrieu. 1998. “A 3D Architectural and Process-based Model of Maize Development”. *Ann. of Bot.* Vol. 81:233-250.
7. Hopcroft, J., and J. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA.
8. Lindenmayer, A. 1968. “Mathematical models for cellular interactions in development I”. *J. Theoretical Biol.* Vol. 18:280–315.
9. (a) Prusinkiewicz, P., and A. Lindenmayer. 1991. “Graphical modeling using L-systems.” In Prusinkiewicz and Lindenmayer, *The Algorithmic Beauty of Plants.* Springer-Verlag. New York, NY.
10. (b) Prusinkiewicz, P., and A. Lindenmayer. 1991. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, NY.
11. Sun, H., Ai, L., Tang, X. 2008. “Digital Design and Implementation of Soybean Growth Process Based on L-system”. In D. Li (Ed.), *Computer and Computing Technologies in Agriculture, Volume II*, pp. 791-797. Springer, Boston, MA.