Loving to Learn Theory
Active Learning Modules for the Theory of Computing[1]

Michael T. Grinder, Seong B. Kim, Teresa L. Lutey, Rockford J. Ross, and Kathleen F. Walsh
Computer Science Department
Montana State University
Bozeman, MT 59717
ross@cs.montana.edu

1 Introduction

Can students love to learn the theory of computing? This topic is, after all, probably the most challenging in the computer science curriculum. The academic structure in the United States from grade school on is not particularly good at preparing students to deal with mathematical abstractions in any case, and the kinds of abstractions that permeate the theory of computing are unlike any encountered in other math courses: strings, formal languages, and models of computation. Furthermore, there is often little correlation provided in a standard theory textbook between the theory and the real world of software development; students wonder just what the point of the course is. It is no surprise, then, that most students struggle with learning the theory of computing.

It is tempting to forego teaching the theory of computing, since students not only struggle with it, but they often retain so little of it. This would be a mistake. Done properly, the theory course puts the "science" into computer science, giving aspiring practitioners a basis for understanding the fundamental laws that govern their discipline: there are problems that cannot be solved, there are intractable problems, there are limitations on the efficiency of the solutions to problems, and so on. Students not only need to know these fundamental truths about their field, but they also need to be able to apply this knowledge to their everyday work of programming.

In this paper we describe recent advances in our long-term efforts in the Webworks Laboratory at Montana State University to make the theory of computing accessible to students through active learning modules designed for use on the web. While we won't be so bold as to say that students will actually *love* to learn the theory of computing as a result of having access to these modules, we can confidently say from our own experience that they will find learning the theory to be more *fun*. And that, we would all agree, is a big step towards helping students learn.

[1]Support for some of the work described here has come from the National Science Foundation, grant number NSF-0088728.

2 Some Background

Interactive computer-based instruction has tantalized educators since at least the advent of computer video monitors, which made individual instructional delivery possible [4]. However, in spite of the fact that many good interactive educational systems have been developed for various topics in computer science, it is well known that these systems are not widely used. This problem and possible solutions have been discussed in the literature before (see, for example, [2]). The two most important issues are

- platform independence
- courseware integration

If active learning software is developed for a specific platform (e.g., Windows XP on a PC) it automatically excludes users who have a different platform (e.g., Mac OS on a Macintosh). If the software is a standalone tool for one specific, isolated topic, such as an animation of the red-black tree abstract data type, then the instructor must go to the effort of locating the software, learning to use it, fitting it into her course, and teaching the students to use it, all for a one- or two-lecture sequence. Each different software module would require the same effort, which most instructors are not willing to expend.

The solution we use in the Webworks Laboratory to these problems is to (1) design all active learning software as applets that run in standard browsers, and (2) integrate the applets into a comprehensive teaching and

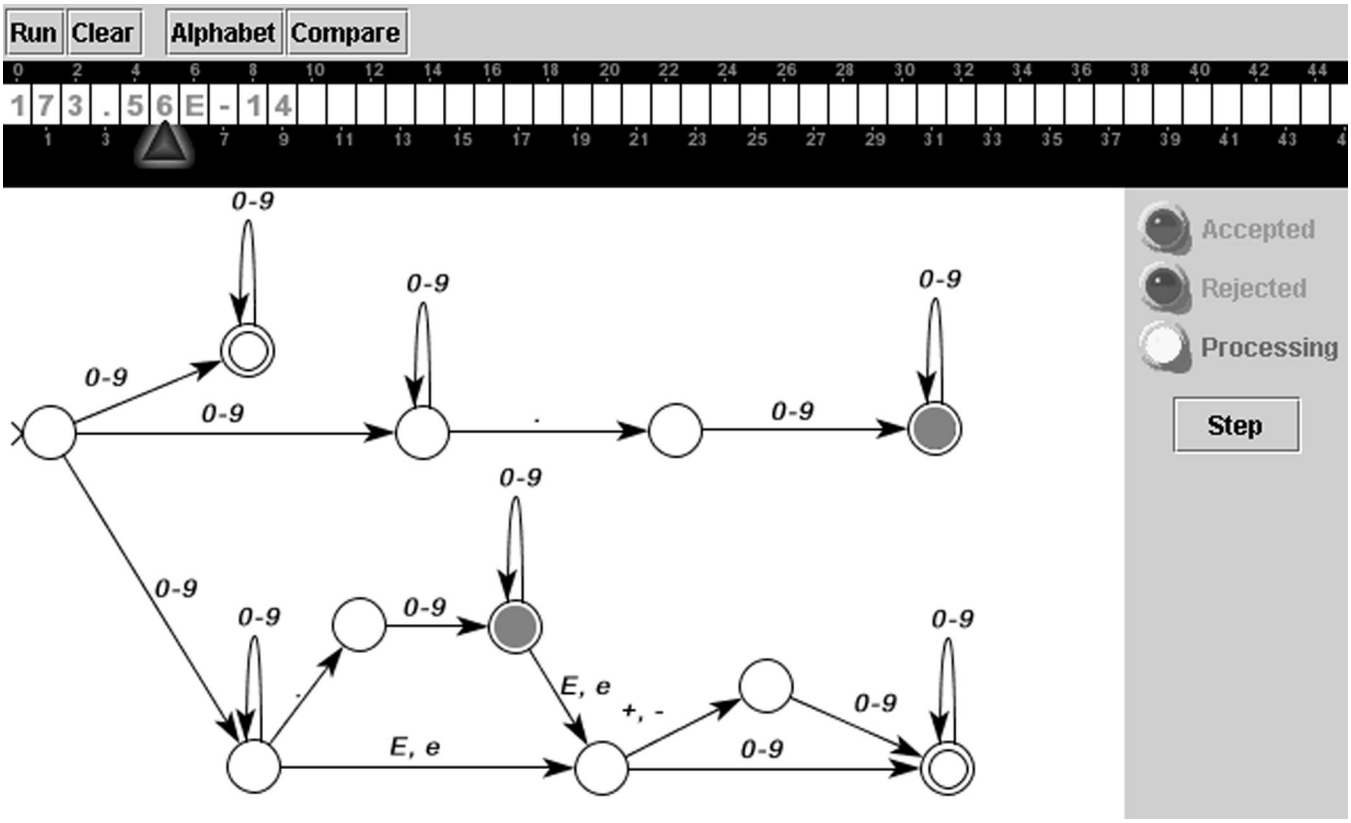

Figure 1: The finite state automaton applet

learning resource (e.g., a hypertextbook) that can augment or supplant a standard textbook as a primary resource for the class.

3 Active Learning Theory Applets

A standard first chapter in a traditional theory of computing course covers three equivalent representations of regular sets: finite state automata, regular expressions, and regular grammars. We have developed active learning applets for each of these models to incorporate into a hypertextbook which we call *Snapshots of the Theory of Computing* (we have reported on the hypertextbook and some of these applets earlier [2], but all have since undergone substantial revision and enhancement). We discuss each applet below, and then we describe how these are to be integrated into a comprehensive teaching and learning resource.

3.1 A Finite State Automaton Applet

The oldest of our applets is the finite state automaton (FSA) applet, the work of the first author. This applet has undergone a complete rewrite and several revisions as we have continued to explore more effective ways to reach students and better ways to incorporate active learning applets into a comprehensive teaching and learning resource, such as a hypertextbook.

Figure 1 gives a snapshot of this applet in action. The FSA illustrated is one that recognizes integer, fixed point, and floating point numbers. It is nondeterministic. At the point shown, the automaton has processed the prefix `173.5` of the input string and is about to consume the next input symbol (`6`) as seen on the input tape. Current states are marked with a disk (colored red on a computer monitor); state transitions are animated in that the red disks move from each current state to the next states in a smooth motion along the appropriate edges as an input symbol is consumed. Rudimentary sound effects accompany the moves. If there is no transition for a given input, the disk turns gray and then disappears. The user has complete control over the animation of the actions of the automaton.

The automaton of this example was constructed to nondeterministically check for an integer, fixed point, or floating point number. At the point illustrated in figure 1, the branch checking for an integer has died because a decimal point was encountered in the input; the branches checking for fixed and floating point numbers are still alive, as seen by the shaded states.

Active learning on the part of the student is promoted throughout. The student must supply input strings and orchestrate the actions of the automaton. Processing can be directed one step at a time through mouse clicks,

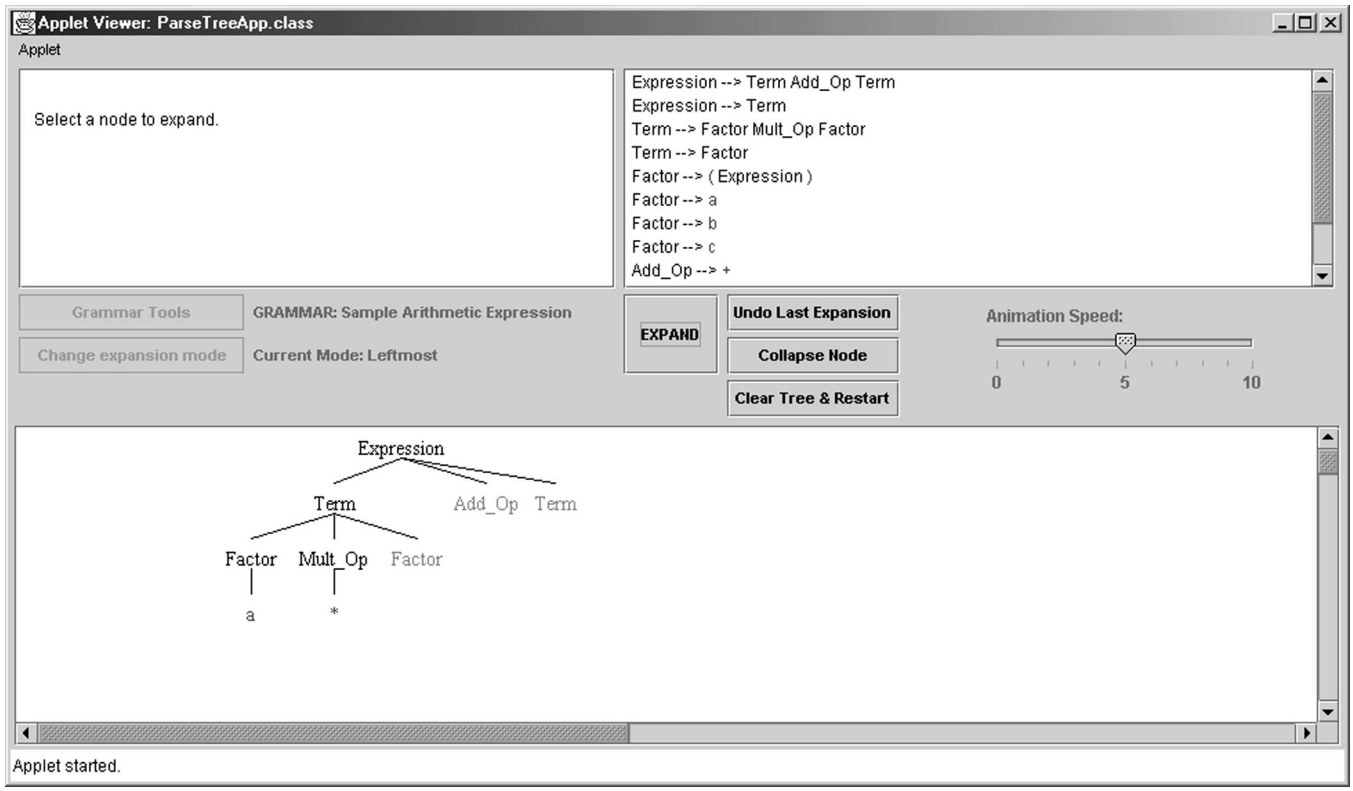

Figure 2: The context free grammar applet

or set to run continuously. The automaton can be edited arbitrarily by the student as well.

Most useful, perhaps, are special features built into the applet for use in exercises. For instance, an exercise can be designed that requires a student to construct an FSA to recognize a particular language. Known properties of FSAs are then exploited to provide feedback to the student as the exercise is attempted. This is accomplished by requiring that the instructor who creates the assignment also supply a correct FSA at the time of creation. The language of the student's FSA is then compared to the language recognized by the correct FSA. If the languages are not the same, a sample string is given that the student's FSA accepts or rejects in error, and the student can try again.

More about this particular applet can be found at the Webworks web site:

`www.cs.montana.edu/webworks`

A further description can also be found in [2].

3.2 A Context Free Grammar Applet

The second applet in our theory repertoire is the context free grammar animator of figure 2. This, too, has undergone a complete rewrite (by the third author) as we have learned more about active learning applets.

There are currently three possible ways to use the grammar applet. The first is in demonstrating how a parse tree for a string is constructed. In this case, the applet is preloaded with a grammar and a string. The student directs the parsing to proceed one step at a time by clicking on the "EXPAND" button. Explanations of what is occurring at each step appear in the upper left pane (these explanations will eventually also be provided in an audio file as well). In this mode the applet can be used to introduce the concept of grammars and parsing to novices.

A second use of the applet is to require the student to produce a parse of a given string as an exercise. In this case, the grammar to be used is either preloaded into the applet, or the student selects the appropriate grammar from a list. Then the student clicks on the nonterminal node to expand (in the bottom pane), the rule to use in expanding this node (in the top right pane), and then the "EXPAND" button to complete the expansion. Note that there is a button for selecting the type of parse to perform; the options are leftmost, rightmost, or arbitrary.

A third use of the applet is to require students to construct their own context free grammar within the applet and then perform parses of various strings using this grammar. The applet flags problems with the submitted grammar, allowing students to correct errors.

In every case, there are features that aid active learning. Rule expansions are done in a smooth fashion; that

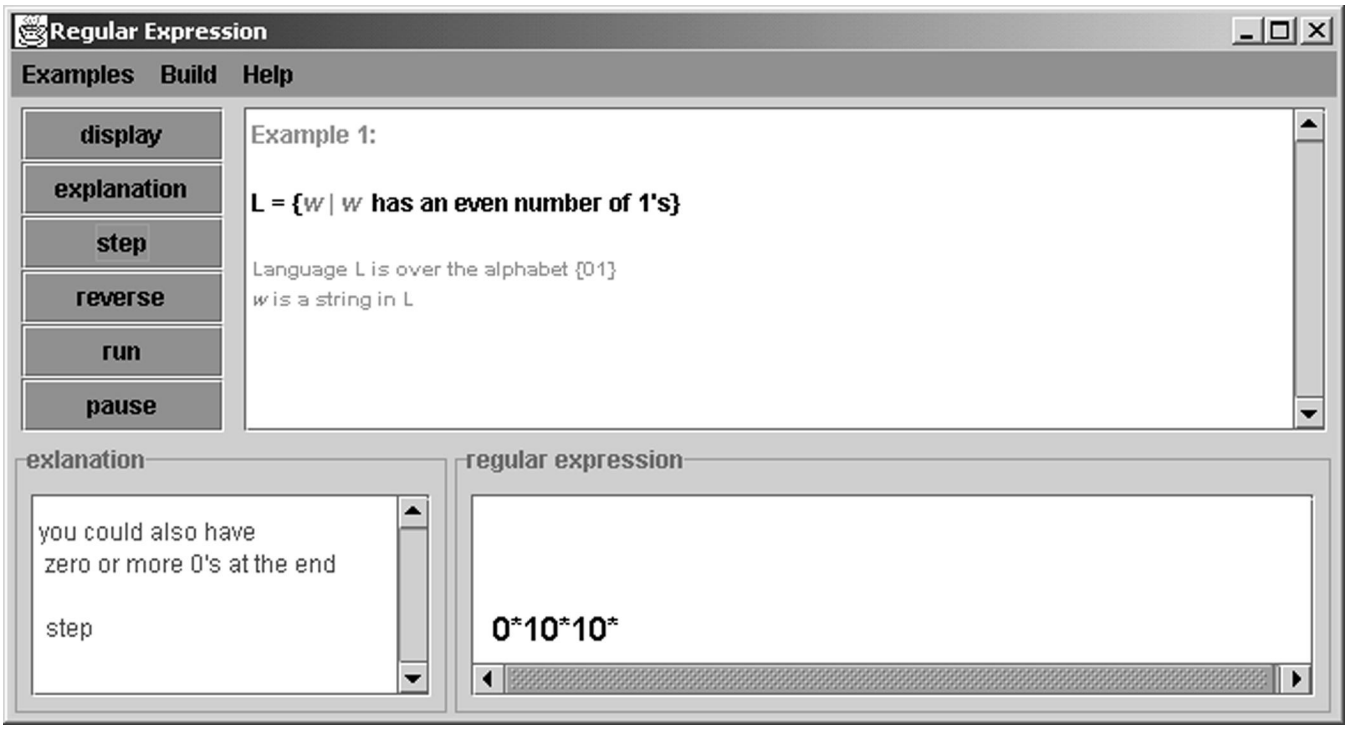

Figure 3: The regular expression applet

is, the applet actually draws the lines from the nonterminal being expanded to the symbols on the right hand side of the rule being applied one at a time in a manner similar to how it would be done by hand. The speed with which this drawing occurs can be controlled with the slider bar. A student can also back up in a parse one rule at a time to try different rules (through the "Undo Last Expansion" button), or an entire branch can be pruned (through the "Collapse Node" button).

3.3 A Regular Expression Applet

The final entry in our current repertoire of active learning theory applets is the regular expression animator (the work of the last author) shown in figure 3. It provides features similar to those of the other applets.

The regular expression applet can be used to introduce novices to the concept of regular expressions in that the applet can be preloaded with the description of a set (regular language) for which a regular expression is to be constructed. Then, a regular expression is constructed a step at a time each time the student presses the "step" button: the language in question is given in the top right pane, the bottom right pane shows the portion of the regular expression constructed so far, and the bottom left pane gives an explanation of how this partial expression was constructed from the previous partial expression. Since there is no single unique or correct way to produce a regular expression, it is up to the instructor who creates the animation and accompanying explanations to provide good insight into this process.

A second use for this applet is in exercises. In these cases, the applet is preloaded with a description of a regular language in the upper right pane and a (hidden) correct regular expression. The student is responsible for constructing a regular expression in the lower right pane that denotes the given regular language. Using known properties of regular expressions, the language represented by the student's regular expression is compared with the language denoted by the correct regular expression and feedback is given to the student about whether his or her regular expression is correct or not[2].

4 Integrating the Applets

As we discussed earlier, there are two major issues that need to be addressed in order to make active learning applets widely used in the curriculum: platform independence and integration with course teaching and learning resources. We have met the challenge of platform independence by creating our applets to run in the most common web browsers (i.e., Netscape and Internet Explorer) as long as the most recent versions of the Java Virtual Machine are installed.

Integrating the applets into regular course teaching and learning resources is a separate matter. We are tackling this issue on two fronts. First, we are ensur-

[2]This feature is still under construction at the time of this writing.

ing that the applets work with each other. Second, we are creating a hypertextbook that incorporates these applets and can be used to augment or supplant the traditional textbook used in a class.

4.1 Coupling the Applets

All three of the applets described above are being developed to work together (this work is in progress). As described, known properties of FSAs are already utilized in the FSA applet to determine whether a student has constructed a correct automaton in an exercise and to provide feedback to the student. This work is being extended to ensure that a constructed automaton is deterministic, minimal, and/or completely specified, if any of these things are required in the exercise.

The same properties that allow two automata to be checked for equivalence can be applied for checking and providing feedback in the context free grammar and the regular expression applets. Algorithms exist for converting regular grammars and regular expressions to FSAs, so the grammar or regular expression supplied by a student in response to an exercise can be checked for correctness using the same process as described for the FSA applet. The only requirement is that the instructor creating the exercise also supply a correct grammar or regular expression, respectively.

The applets are also being extended to encompass all aspects of the theory of finite automata and regular languages, including active learning demonstrations and exercises for

- the conversion of nondeterministic FSAs to equivalent deterministic FSAs
- the conversion of deterministic FSAs to their minimal forms
- the conversion of regular expressions to and from FSAs
- the conversion of regular grammars to and from FSAs
- the pumping lemma for regular languages

to name a few.

Finally, the applets are being extended to include other models. The FSA applet will eventually be able to animate any of the usual models of computation up to Turing machines, and the context free grammar applet will be able to function with arbitrary grammars.

4.2 Courseware Integration

As applets, each of the animation systems described above can be embedded directly into comprehensive learning resources designed for the web. We have started this process in the form of the *Snapshots* hypertextbook mentioned earlier. There is not room in this paper to describe the hypertextbook project, but it can be found at the Webworks web site and in [1, 2]. Briefly, it presents the theory of computing in a way that allows learners with different academic needs to progress through the material at a comfortable level. At points where a traditional textbook would have examples of models of computation, conversion algorithms, and so forth, *Snapshots* instead includes the active learning applets described here.

The *Snapshots* hypertextbook is being released a portion at a time, as each portion becomes ready (hence the title, *Snapshots of the Theory of Computing*). As a comprehensive teaching and learning resource for the web, *Snapshots* solves the problems of platform dependence and non-integrated resources, and thus promises to be a useful active learning resource for teaching and learning the theory of computing.

5 Summary

Other researchers have produced some effective standalone applets for some of the components of the theory of computing discussed above. There is not room to cite each of these other efforts here, but references can be found in [2] and on the Webworks web site. Few others are developing the kinds of integrated, comprehensive, web-based, active learning teaching and learning resources of the kind we describe here. One other project of note is the Ganimal project, which does include an online textbook on finite state automata [3].

We have used our applets successfully in pilot form in courses at Montana State University, and they have been used at a few other institutions as well. We hope that our work will prove useful and encourage others to create such integrated active learning resources for the web as well.

References

[1] Boroni, C. M., Goosey, F. W., Grinder, M. T., Lambert, J. L., and Ross, R. J. Tying it All Together Creating Self-Contained, Animated Interactive, Web-Based Resources for Computer Science Education. In *Thirtieth SIGCSE Technical Symposium on Computer Science Education (SIGCSE Bulletin)* (Mar. 1999), vol. 31, number 1, pp. 7–11.

[2] Boroni, C. M., Goosey, F. W., Grinder, M. T., and Ross, R. J. Engaging Students with Active Learning Resources: Hypertextbooks for the Web. In *Thirty Second SIGCSE Technical Symposium on Computer Science Education (SIGCSE Bulletin)* (Mar. 2001), vol. 33, number 1, pp. 65–69.

[3] Reinhard Wilhelm, et. al. Ganimal. http://rw4.cs.uni-sb.de/~ganimal/, 2001.

[4] Stasko, J., Domingue, J., Brown, M. H., and Price, B. A., Eds. *Software Visualization: Programming as a Multimedia Experience*. MIT Press, 1997.