Tying it All Together
Creating Self-Contained, Animated, Interactive, Web-Based Resources for Computer Science Education[1]

Christopher M. Boroni, Frances W. Goosey, Michael T. Grinder, Jessica L. Lambert, Rockford J. Ross
CS Department, Montana State University, Bozeman, MT 59717
ross@cs.montana.edu

Tying What All Together?

For many years now, the annual SIGCSE Symposium has been the primary coming-out party for new software systems aimed at computer science education. Many nice systems have been developed and refined over the years that use interaction and animation to engage students in learning. In spite of their evident utility, most of these systems have languished, being used only at their home institutions and a few others. The reason can be stated succinctly: platform dependence. Pick your platform. There were systems that ran only with X-windows, some that worked only on Sun stations, others that required Microsoft Windows 95, several that demanded a Macintosh, and so on. Not only did this hinder the use of these systems in formal classroom settings, but it also made it virtually impossible for all students in a course to have off-campus access to the systems, a desirable capability for learning. A good (but incomplete) review of visualization software for education is found in [4].

Then came the World Wide Web and increasingly sophisticated browsers. Our mouths watered at the thought of what these new technologies promised for enhanced computer science education. The turning point, though, was the introduction of the Java Virtual Machine, which allowed educational software to be developed in the platform independent fashion of the Web. Very suddenly, the rules of the game changed, and the paradigm shifted [1]. The problem of platform dependence vanished (. . .well, pretty much, anyway) overnight. The learning curve for students adjusting to a new software system dropped dramatically, as nearly all students came to class already comfortable with Web browsers. It quickly became possible to reach virtually anyone virtually anywhere with Web-based educational materials.

The fact was not lost on computer science education researchers. In the proceedings of the 1998 SIGCSE Symposium alone there are at least nine papers with a specific emphasis on Web-based educational software. Established researchers turned their focus to the Web, and new projects were introduced. Algorithm animations, program code animations, and animations of other computer science concepts sprouted on the Web. In most cases, though, it was left to the instructors using these systems to weave them into the fabric of the curriculum. Needed—in conjunction with continued development and enhancement of quality visualization software—were concerted efforts to tie this software into cohesive, standalone educational resource materials, or hypertextbooks.

In this paper we describe our own initial efforts to develop a hypertextbook: a Web-based, standalone educational resource for computer science that incorporates the best of hypertext, interactive animations, audio, and other features. The accompanying talk will be a live, projected demonstration of the main features of the hypertextbook—its layout, links, and animations.

Hypertextbook Layout

There were many choices to make in creating the hypertextbook. The first was the topic: the theory of computing was chosen. The grand scheme is to create a work on the usual topics of computability, from simple machine models, such as finite state automata, all the way to uncomputability and intractability, including NP-completeness. The entire field of computing is ripe for animation, but theory seems to be the one area where students struggle the most. Besides, it sounded fun.

Once the topic was chosen, we needed a title for the work. After some bouncing around the title settled at *Snapshots of the Theory of Computing*, or just *Snapshots* for short. Why *Snapshots*? The title is actually quite appropriate for a Web-based work. Animation modules take considerable time to design, implement, and polish. So, for example, a finite state automaton animation module can be done that allows for an

[1] Support for some of the work described here has come in part from the National Science Foundation, grant number DUE-9752673, and a 1997–98 MONTS grant from MSU.

Figure 1: Bridger Bowl, Montana

entire "chapter," or "snapshot," on finite state automata and regular languages to be written well before the animation infrastructure is ready for subsequent chapters on pushdown automata, computability, or intractability. While it isn't the custom to publish a regular textbook a chapter at a time, the Web invites us to do so, allowing us to provide information as it becomes available.

Those were the easy decisions. A harder decision was the determination of the layout of the hypertextbook. We wanted to take advantage of hypertext links as much as we could in catering to different student backgrounds and learning styles. Our choice of layout was inspired by the skiing industry. Figure 1 is a map of Bridger Bowl in Montana. Without color, some of the effect is lost, but the basic idea can be seen. At the top of a run there are usually various ways down. Easy routes down are marked with a green circle, intermediate routes with a blue square, and more challenging runs with a black diamond. Multiple diamonds are used for yet more difficult terrain. Generally, at a handful of locations on a run, signs designate turning points, where a person can choose to deviate onto an easier or more challenging route. All routes eventually take you to the same location at the bottom of the mountain.

This is the layout we selected for *Snapshots*. We even borrowed the symbolism, making it easy for readers to choose which way to proceed through the hypertextbook simply by following links indicated by green circles (the easiest way through with lots of intuitive explanations and animated examples preceding the introduction of the mathematics), blue squares (for intermediate learners with less explanation, fewer examples, and earlier introduction of mathematical abstractions), or black diamonds (for the more abstract and mathematical thinkers, leading to concise mathematical presentations of the material) at various points based on their comfort level with the material.

Getting Animated

Hypertext links are a great aid in creating documents that allow a student to study material according to his or her background and learning style. However, hypertext is not new, and by itself probably not particularly engaging to a student. Animation is what gives the hypertextbook its life.

As an illustration, consider a chapter on finite state automata (fsa's). As the topic is introduced, examples are given. Suppose that instead of a simple picture of an fsa with a long description of its workings, an animation is provided. That is, a graphical representation of an fsa is inserted with point and click controls for managing its execution. Exercises could involve even more elaborate animations: students could be required to construct an fsa in the animation window through simple mouse actions and then run their creation on various inputs, modifying the fsa until it is correct.

Figure 2 is a snapshot of the finite state automaton animator in action that accompanies *Snapshots*. It has all of the features mentioned above. Students can run the fsa given (or choose others from a dropdown list), providing different input strings as desired. The fsa can be modified arbitrarily, and even discarded for one built from scratch. As students run the fsa, a colored disk moves along the correct arrow from one state to the next depending on the input. The current character being processed shows up under a read/write head. Sound is used to spice up the activities of the fsa.

It is also informative to see how an fsa can be implemented as a program. Figure 3 is an illustration of the program animator that accompanies *Snapshots*. Here, a portion of a program implementing the fsa given in figure 2 is shown being run on the same input string. In this illustration, the fsa is implemented with a single loop that processes each input character in succession. A case statement selects the code for the current state, and in this code is another case statement that further selects the proper next state according to the cur-

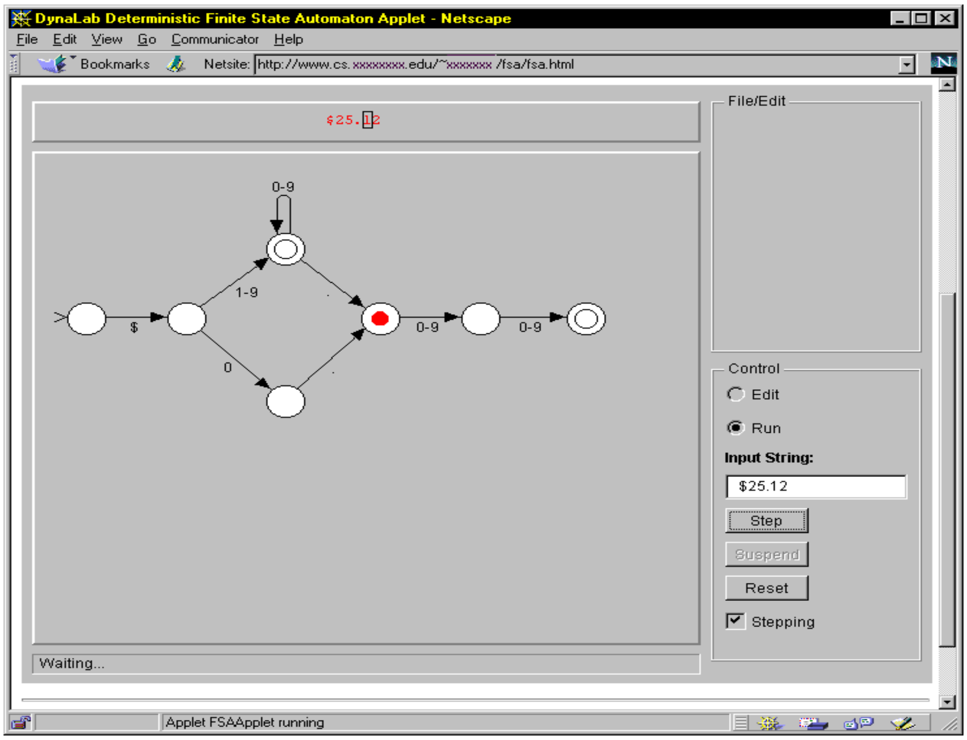

Figure 2: Snapshot of the FSA Animator Processing the String $25.12

rent input symbol. Different implementations can easily be incorporated.

A student using the program animator has full control over the execution of the program, including reversing program execution arbitrarily far to study puzzling sections as often as desired. Execution cost is even shown in one pane for time complexity experiments. The heart of the animator is a virtual machine that can execute its machine code in reverse, which provides the capability for animated reverse execution in the source code.

In making the mental connection between fsa's and formal languages, an animated grammar module is also helpful. Figure 4 shows the *Snapshots* grammar animator in the process of constructing a parse of the same string processed by the fsa animator in figure 2 and the program animator of figure 3. Again, this animator is flexible enough for students to input arbitrary context free grammars for processing.

Together these three animators cover most of the required concepts for a module on finite state automata and regular languages.

Snapshots

This now brings us to *Snapshots* itself. There is no good way to illustrate the hypertextbook here. As the used car salesmen always proclaim, "...must be seen to be appreciated." Work in progress can be viewed at `www.cs.montana.edu/webworks`. On each page of the hypertextbook, there are links to the next and previous pages in succession, as well as links to the table of contents, the glossary, and the index. At certain junctures, just as in the ski hill model, links are also inserted using the green circle, blue square, and black diamond symbols that lead students to easier or more challenging tracks. We expect that students will use different tracks on different occasions of reading the book, going down the easier routes upon a first encounter, while taking the more challenging tracks upon review.

The interesting and challenging aspect of *Snapshots* is the application of all of the tools of animation, sound, and text to help students understand the deeper aspects of the theory. For instance, students traditionally find the pumping lemma for regular languages to be inscrutable. Animated versions that let them study it from all angles with different parameters should help. The same is true for even the much more advanced topics, such as NP-completeness. This is the focus of *Snapshots*.

Really, learning to read and understand theory is an acquired skill that comes with practice, just as with skiing. Now if we could only inspire students to see the learning of theory as *fun*... Perhaps animated, interactive hypertextbooks like *Snapshots* will help.

Challenges

As the title and content of the paper indicate, *Snapshots* represents a concerted effort to tie together many years of work

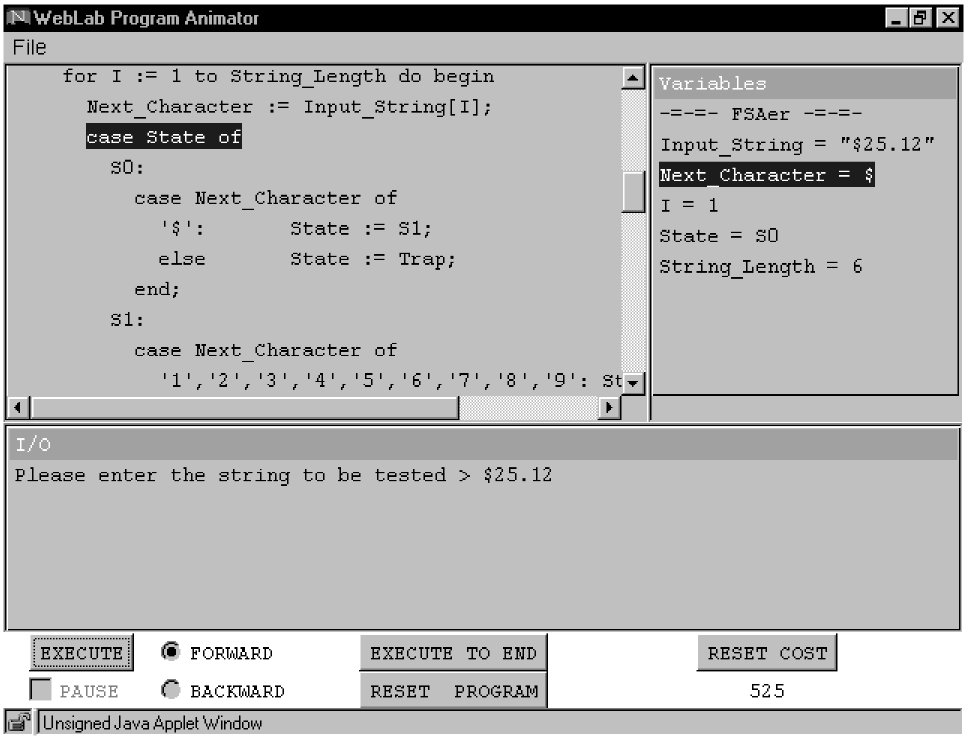

Figure 3: Snapshot of the Program Animation of the Implementation of the FSA of Figure 2

in animation research and development into a single, valuable educational resource. The work will undergo formal class testing soon. We—like many others—*have* used (and performed some formal evaluations of) the individual animations described here successfully in individual laboratory exercises, lectures, and self-study modules, but the creation of an integrated hypertextbook that covered a major component of the computer science curriculum was just beyond the capabilities of technology[2]. Our own work in animation research and development and that of other researchers in the field have now matured enough, as demonstrated above, to tackle such an effort. The platform independence and widely accessible nature of the Web provide the incentive.

That's not to say that it hasn't been tried before. A well-known book on algorithms was converted to Macintosh hypertext and augmented with animated examples [2]. Its largest impediment to success was the usual one: platform dependence. In the case of *Snapshots* we are taking the opposite tack, one that is sure to become the norm as other authors contribute to the inventory of Web-based instructional resources: we are constructing *Snapshots* from the ground up based on the facilities of the Web, its browsers, and animation technology, not on the paradigm of static, hardcopy textbooks.

[2] Not to mention our ambition level. Who wanted to invest a large chunk of time in something that would languish because of platform dependence or changes in computer hardware and/or software?

Oh, yes, there are still problems looking for solutions. The biggest hurdle facing an author of a Web-based theory hypertextbook is one of fonts. Expressing a common derivation in a grammar, such as

$$S \overset{*}{\Longrightarrow} wA\alpha \overset{1}{\Longrightarrow} w\beta\alpha \overset{*}{\Longrightarrow} wx$$

is impossible in HTML. Some utilities that convert LaTeX to HTML turn such phrases into pictures (.gif files). Others attempt to turn LaTeX mathematical expressions into reasonably similar views using pure HTML. Still others are plugins that use techniques unrelated to HTML to include mathematics in Web pages. But help is on the way in the form of MathML. This mathematics markup language will not only allow the creation of arbitrary mathematical expressions, but the expressions so created will also contain enough information to convey the *semantics* of the expressions (e.g., how they should be evaluated by some math package, such as Mathematica). A brief discussion of many of these systems can be found in [3]. We expect that innovations like MathML and XML will make the construction of hypertextbooks much easier in short order.

A few anticipated problems, on the other hand, have not materialized. Some feared that the animation activities would run too slowly in the context of the Java Virtual Machine and the Web. This fear was unfounded. Even our program animator, which is based on an emulated virtual machine, which in turn is executed on the Java Virtual Machine, runs without a hitch. If the hypertextbook with all of its an-

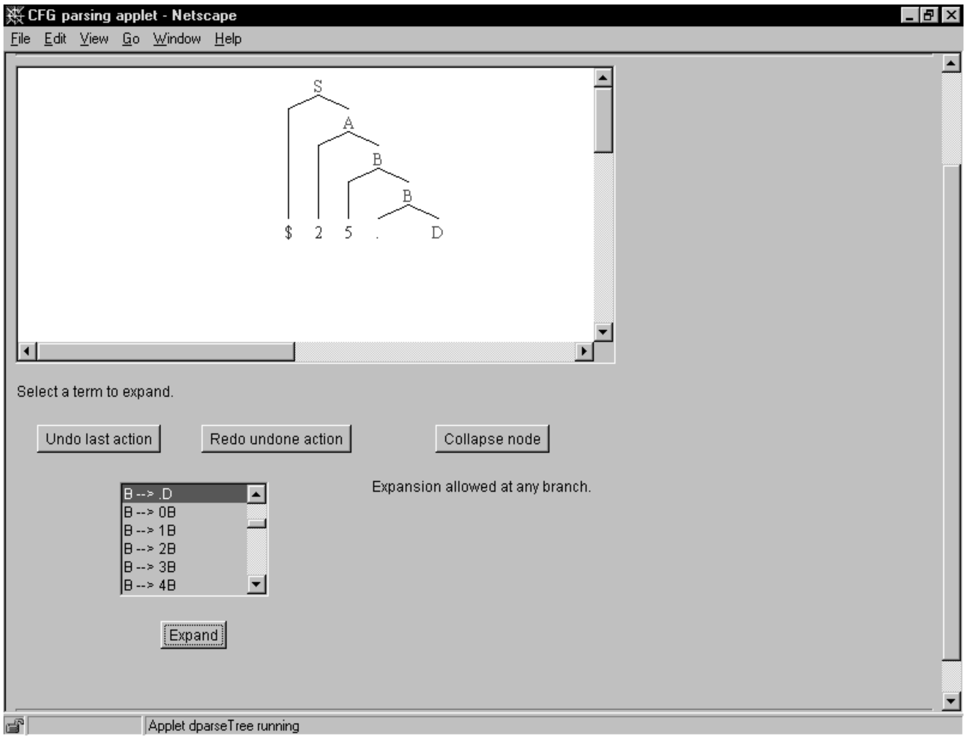

Figure 4: Snapshot of a Parse Tree Corresponding to the FSA of Figure 2

imations were to be accessed across the Web, there would indeed be a problem with download speed. However, our intent is to provide the entire hypertextbook on a CD or DVD, allowing it to be accessed locally, in which case it will run without untoward delays. This method of delivery also allows for regular modifications and updates to the material. New versions can be released regularly—every semester, for example. The cost of CD production is so low that the price of such a hypertextbook should be well below ten dollars.

The larger problem remains our responsibility as computer scientists. The research, development, and class testing of effective, interactive, animated educational systems is a daunting task requiring years of hard work. The animations described in this paper have their origins in the early 1980s. They contain well over 100,000 lines of code, all of which have undergone numerous revisions. The program animator, for example, required the design of a virtual computer capable of running its machine language in reverse, an emulator for the virtual machine, compilers that translate modern programming languages into the virtual machine code, and an animation interface for controlling the animation of the source code in execution, a project that has stretched across many students and a long period of time. These modules will be available for others to use in constructing educational resources, but many others await development.

This, then, is the fundamental challenge to computer science education researchers: to intensify efforts in the design, development, and formal testing of resources for computer science education that make effective use of the new paradigm presented by the evolving World Wide Web and its related technologies. It's fun. We can do it.

References

[1] Christopher M. Boroni, Frances W. Goosey, Michael T. Grinder, and Rockford J. Ross. A Paradigm Shift! The Internet, the Web, Browsers, Java, and the Future of Computer Science Education. In *Twenty-ninth SIGCSE Technical Symposium on Computer Science Education (SIGCSE Bulletin)*, volume 30, number 1, pages 317–321, March 1998.

[2] Peter Gloor, Scott Dynes, and Irene Lee. *Animated Algorithms: A Hypermedial Learning Environment for* Introduction to Algorithms . The MIT Press, 1993. An animated version of the textbook *Introduction to Algorithms* by Cormen, Leiserson, and Rivest; for the Macintosh only.

[3] Rockford J. Ross. Mathematics on the Web. *SIGACT News*, 29(2):33–41, June 1998.

[4] John Stasko, John Dominigue, Marc H. Brown, and Blaine A. Price. *Software Visualization: Programming as a Multimedia Experience*. MIT Press, 1997.