A Preliminary Empirical Evaluation of the Effectiveness of a Finite State Automaton Animator

Michael T. Grinder
Computer Science Department
Montana Tech of the University of Montana
Butte, MT 59701
mgrinder@mtech.edu

Abstract

The FSA Simulator is a Java program created to allow computer science students to work and experiment with finite state automata (FSAs). One of its unique features is the ability to compare the languages of two FSAs. This FSA comparison feature lets the software give students feedback about the accuracy of their work as they do exercises, guiding them toward a correct solution. This paper discusses some preliminary experiments performed to determine the effect of this feedback mechanism on students' learning. Two experimental labs were conducted, the results of which suggest that this feature improved students' success rate when doing exercises, but did not appear to significantly improve the students' performance when the comparison feature was not available.

Categories and Subject Descriptors
K.3 [**Computers & Education**]: Computer & Information Science Education - *Computer Science Education.*

General Terms
Experimentation, Measurement, Theory

Keywords
Visualization, Animation, Theory of Computation, Evaluation

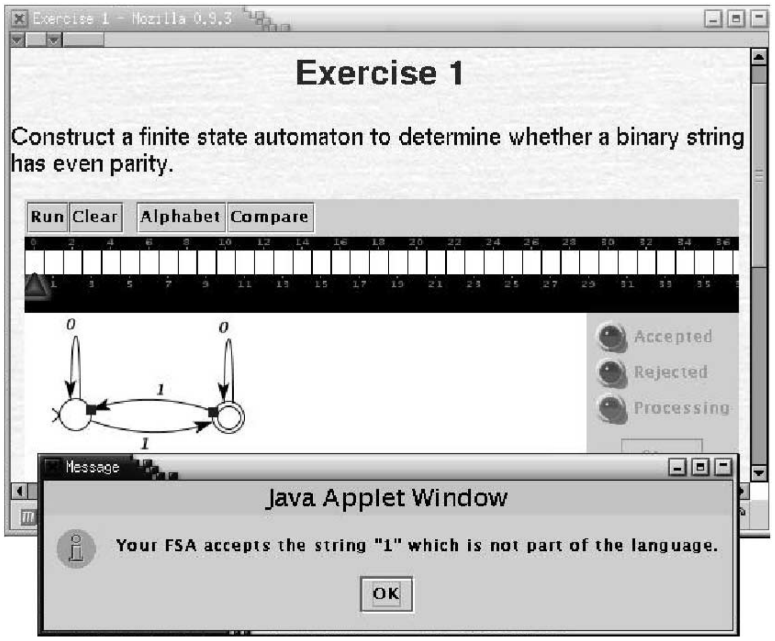
1 The FSA Simulator

The FSA Simulator is an animation program for simulating finite state automata. It was developed as a part of a larger project at the Webworks Laboratory at Montana State University. The goal of the Webworks Laboratory is to develop web-based educational resources that include interactive animation software. Webworks' main focus is to produce such resources for teaching the theory of computation.

Most of the features of the FSA Simulator are a subset of the features of similar software packages, such as JFLAP [3] and JCT [6]. Using the FSA Simulator, users can build FSAs using a graphical state diagram representation. Once an FSA has been built, users can test arbitrary strings for acceptance. The FSA Simulator's unique feature is its ability to load a pre-built FSA in the background and compare the language of the background FSA with the language of the FSA that the user is currently manipulating. If the languages of the two FSAs differ, the program will display example strings to illustrate where the differences lie (as depicted in Figure 1). This feature allows instructors to create exercises that give students enough feedback to guide them to a correct solution. More information about the FSA Simulator's features and implementation is available in [4].

2 Goals

The initial inspiration for the FSA comparison feature was the observation that, even with the help of the FSA Simulator, students would often build incomplete solutions to exercises without receiving any feedback about where their solutions were incorrect. Although the students had the ability to test their FSAs using strings from the language, their testing was nearly always perfunctory. The comparison feature allows students to detect when their solutions are incorrect and receive hints about where the problem lies. We hoped that over the course of several exercises, the students would be able to successfully complete exercises more quickly,


Exercise 1

Construct a finite state automaton to determine whether a binary string has even parity.

Java Applet Window

Your FSA accepts the string "1" which is not part of the language.

Figure 1: The FSA Simulator after doing a comparison operation.

would begin to learn where they were making mistakes when constructing FSAs, and would learn how avoid such mistakes as they progressed. We also hoped that such guided exercises would improve the students' performance when the comparison feature was not available.

3 Previous Work

Although studies have shown that narrated animations help students learn about mechanical processes [5], little formal evaluation has been done to study the effects of interactive animation software in CS education. The most prominent examples of such evaluation within the CS education community are the studies conducted at Georgia Tech to evaluate algorithm animation software (for example, [2]), which produced mixed results.

4 Experiments

4.1 Subjects

Two experiments were performed in the lab sections of computer science courses offered at Montana State University during the spring semester of 2002. The first experiment was performed in a first-year course, CS 221. This course is mainly taken by computer science and computer engineering students during the second semester of their first year, after having taken an introductory programming class. The second experiment was performed in a second-year course, CS 223. This course is usually taken by computer science majors during the second semester of their second year. Some computer engineering students also take CS 223 as a professional elective. The content of the experimental lab was not directly related to the content of either course. Students were not graded on their performance in the lab, but did receive course credit for attending and participating.

Before the students began the lab assignment, they were asked to fill out a form to provide some demographic information about themselves. The form asked for each student's age, sex, major, year in college, approximate grade point average, and standardized test scores. Students were also asked whether they were familiar with terms related to the subject of the lab, such as *finite state automaton* and *regular expression*, to determine whether they had any previous exposure to the topic. To protect the students' privacy, the information provided in these forms was associated only with a number assigned to the student.

Fifty-two students participated in the first experimental lab and forty-four students in the second. Three of the students in Experiment 1 and six of the students in Experiment 2 were women. All of the women ended up being in the test groups. Most of the students in the first lab were male, traditionally-aged, first-year students majoring in computer science or computer engineering with little or no experience with topics related to regular languages. The students in the second experiment had similar demographics, although they had taken more computer science classes and some of them were also taking the theory of computation course, CS 350, concurrently with CS 223.

4.2 Design

For the first experiment, there were three sections of the lab, each lasting two hours. Each section was assigned to be in either the test or control group. The first and third sections, a total of 28 students, made up the test group. The second section, with 24 students, was the control group. The second experiment had two lab sections. The first section, 17 students, was designated the control group and the second section, 27 students, was the test group.

Overall, the demographics of the test and control groups for both experiments appeared to be equivalent. A more controlled way of choosing test and control groups would have been preferable, but was not practical at the time.

4.3 Treatments

The experimental labs were divided into two parts. In the first part, the students read through a web-based tutorial drawn from material in the Webworks Laboratory's theory of computation hypertextbook [1]. Examples in the test groups' tutorial used the FSA Simulator as an applet embedded in HTML pages. In place of the applet, the control groups viewed static images that conveyed equivalent information.

While they were reading through the tutorial, the students completed four exercises. Exercise 1 asked the students to identify the parts of an FSA, such as the start and final states. They were also asked to determine if the automaton accepted specific strings. Exercise 2 asked the students to construct a state diagram of an FSA from its formal description. For Exercise 3, the students were asked to construct a finite automaton that accepted identifiers for a programming language. In Exercise 4, participants were asked to construct a finite automaton that accepted floating point literals.

The test groups used the FSA Simulator with the comparison feature during all of the exercises. Images of solutions to the exercises were made available to the control groups as links from the exercise pages. Both groups were asked to write down their answers to the exercises on a paper worksheet.

The second part of the lab was a paper-and-pencil test with five problems similar to the exercises in the tutorial. Problem 1 of the test was similar to Exercise 1, but two additional questions were added. Students were also asked to identify a string that would be accepted and one that would be rejected by the FSA. For Problem 2, the students were asked to write a formal description of the FSA depicted in Problem 1. This was a bad idea, since the students did not have access to a description of the formal definition during the test. The answers to this problem were not graded. Problem 3 asked the students to draw a state diagram of a finite state automaton which accepts binary strings that start with 1 and end with 0. Problem 4 was similar to Problem 3. The language for this problem was binary strings with at least three 1s. Problem 5 was to create an FSA that accepted a language consisting of binary strings that do not contain the substring 110.

All students took the same test without reference to any of the online materials from the tutorial. The lab needed to be completed within 110 minutes. If students were not done with the tutorial within 80 minutes, they were asked to stop where they were and take the test during the last 30 minutes.

4.4 Observations

The students in both test groups spent much more time on the lab than the students in the control groups. It was observed that some of the test groups' students encountered difficulties while learning the applet's user interface. Students in the test groups spent more time working on the exercises than their classmates in the control groups. This may be attributed, in some cases, to the extra time needed by the test groups to learn the applet's interface. Also, the test groups' ability to receive feedback about their solutions may have encouraged them to keep working an a solution until it was correct. Many of the students in the test groups, therefore, did not have sufficient time to work through all of the exercises. Nearly a third of Experiment 1's test group were not able to begin Exercise 4. Most of the students in the control groups, on the other hand, completed the exercises and the test much more quickly. All of the students in the Experiment 1's control group completed the entire lab in less than 90 minutes.

Many of the students in the test groups appeared to enjoy working with the FSA Simulator applet. The ability to check the accuracy of solutions and receive hints when incorrect solutions were submitted made the exercises seem like a puzzle game. A few of the students even competed with each other to see who could complete the exercises first. The control groups were much more subdued. The students in these groups asked fewer questions and appeared to be much less enthusiastic about the lab.

4.5 Results

The results of labs are summarized in Tables 1 and 2. Although the test groups tended to do better on most of the exercises and test problems, the differences between the test and control groups were often not significant. The significance of the differences between the test and control groups for Exercise 1 and Problem 1 were tested using a one-tailed t-test with Student's t distribution

($\alpha = 0.05$). Fisher's Exact Probability Test (also with $\alpha = 0.05$) was used to determine the significance of the results for Exercises 2-4 and Problems 3-5.

5 Analysis

The original goal of the comparison feature was to guide students to the successful completion of exercises. The data from the two experiments appears to demonstrate that the comparison feature of the applet does improve students' peformance on exercises. The data suggests that the test groups had an advantage over the control groups when doing the exercises, especially as the exercises became more difficult. The groups in both experiments seemed to be equally matched on Exercise 1 and Problem 1. In Experiment 1, the test group did significantly better than the control group on Exercises 3 and 4. The results weren't quite as pronounced for Experiment 2, the test group did do significantly better than the control group on Exercise 4, but the difference on Exercise 3 did not quite fall into the significant range.

Although the success rates for the test group were higher than the control group for Exercises 2-4, they were lower than original expectations. It may be that some of the students in the test groups were able to successfully complete the exercises, but made mistakes when transferring the state diagrams from the computer screen to their worksheets. Future evaluations should probably use a completely electronic system for exercises to prevent such problems. Also, as mentioned above, the test groups spent much more time completing the exercises than the control groups. It may be that some of them could have successfully completed more of the exercises if they had had more time.

The results from the paper-and-pencil test were not as encouraging. There were no significant differences between the test and control groups on any of the test problems, although Experiment 2's test group did perform quite a bit better than the control group on Problem 4. It appears that the advantages that the test groups had when doing the exercises did not necessarily carry over when the applet was not used. There are several possible explanations for this. The students might not have spent enough time with the simulator to produce any lasting benefit, the students in the test groups might not have had enough time to complete the test, or working with the FSA Simulator might not provide any measurable benefit over pencil-and-paper exercises.

6 Future Work and Conclusion

Much more work needs to be done in evaluating how use of the FSA Simulator affects learning. The evaluations described in this paper were very limited and are not a sufficient base from which to draw definite conclusions.

The evaluation process needs to be improved to improve the quality of the results. A larger pool of subjects needs to be found for the testing, so that the subjects are more representative of the target student population. Also, the test instruments need to be more carefully designed to ensure that they are indeed testing students for the correct knowledge.

An important follow-up evaluation would be to observe use of the FSA Simulator throughout a theory of computation course. How would the FSA Simulator affect learning if students saw it demonstrated during lectures, used it to complete several homework exercises outside of class, and then used it while taking an exam? It would seem that such extensive exposure would have a strong positive effect on how well students would understand FSAs, but the results of are difficult to predict.

It would also be helpful to do a formal evaluation of students' attitudes toward using the simulator. If it could be shown that use of the FSA Simulator significantly increased students' enthusiasm for the study of the theory of computation, that would be a compelling benefit even without an accompanying increase in students' understanding of the subject.

The results of these preliminary evalutions give a good basis for much fruitful research in the future. Over time, we should be able to gain a much firmer grasp of how to best use technology to improve computer science education.

References

[1] Boroni, C. M., Goosey, F. W., Grinder, M. T., and Ross, R. J. Engaging students with active learning resources: Hypertextbooks for the web. In *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education* (March 2001), vol. 33, pp. 65–70.

[2] Byrne, M. D., Catrambone, R., and Stasko, J. T. Do algorithm animations aid learning? Tech. Rep. GIT-GVU-96-18, Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA, August 1996.

[3] Gramond, E., and Rodger, S. H. Using JFLAP to interact with theorems in automata theory. In *Thirtieth SIGCSE Technical Symposium on Computer Science Education* (1999), pp. 336–340.

[4] Grinder, M. T. Animating automata: a cross-platform program for teaching finite automata. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education* (2002), ACM Press, pp. 63–67.

Experiment	**Question**	**Group**	**Average Score**	**Standard Deviation**	**One-tailed t-value** ($\alpha = 0.05$)
Experiment 1	Exercise 1	Test	5.1	0.994	-1.123
		Control	5.5	1.504	
	Problem 1	Test	7.9	1.571	-1.449
		Control	8.5	1.641	
Experiment 2	Exercise 1	Test	6.7	0.961	0.785
		Control	6.4	1.176	
	Problem 1	Test	8.8	0.934	1.431
		Control	8.4	0.966	

Table 1: Results for Exercise 1 and Problem 1

Experiment	**Question**	**Group**	**Number Correct**	**Percent Correct**	**Fisher Test p-value** ($\alpha = 0.05$)
1	Exercise 2	Test	16 of 28	57%	0.1281
		Control	9 of 24	38%	
	Exercise 3	Test	16 of 28	57%	$3.425 \times 10^{-5}\star$
		Control	1 of 24	4%	
	Exercise 4	Test	10 of 28	36%	$8.295 \times 10^{-4}\star$
		Control	0 of 24	0%	
	Problem 3	Test	8 of 28	29%	0.5111
		Control	6 of 24	25%	
	Problem 4	Test	15 of 28	54%	0.6258
		Control	13 of 24	54%	
	Problem 5	Test	1 of 28	4%	0.4413
		Control	2 of 24	8%	
2	Exercise 2	Test	20 of 27	74%	0.1331
		Control	9 of 17	53%	
	Exercise 3	Test	17 of 27	63%	0.0692
		Control	6 of 17	35%	
	Exercise 4	Test	18 of 27	66%	$0.0017\star$
		Control	3 of 17	18%	
	Problem 3	Test	16 of 27	59%	0.75
		Control	11 of 17	65%	
	Problem 4	Test	20 of 27	74%	0.0683
		Control	8 of 17	47%	
	Problem 5	Test	2 of 27	7%	0.371
		Control	0 of 17	0%	

Table 2: Results for Exercises 2-4 and Problems 3-5 (significant differences are indicated with $\star$)

[5] Mayer, R. E., and Anderson, R. B. The instructive animation: Helping students build connections between words and pictures in multimedia learning. *Journal of Educational Psychology 84*, 4 (1992), 444–452.

[6] Robinson, M. B., Hamshar, J. A., Novillo, J. E., and Duchowski, A. T. A java-based tool for reasoning about models of computation through simulating finite automata and turing machines. In *The proceedings of the thirtieth SIGCSE technical symposium on Computer science education* (1999), ACM Press, pp. 105–109.