Increasing Engagement in Automata Theory With JFLAP *

Susan H. Rodger
Duke University
Durham, NC
rodger@cs.duke.edu

Eric Wiebe
NC State University
Raleigh, NC
eric_wiebe@ncsu.edu

Kyung Min Lee
Duke University
Durham, NC

Chris Morgan
Georgia Tech
Atlanta, GA

Kareem Omar
NC State University
Raleigh, NC

Jonathan Su
Duke University
Durham, NC

ABSTRACT

We describe the results from a two-year study with fourteen universities on presenting formal languages in a more visual, interactive and applied manner using JFLAP. In our results the majority of students felt that having access to JFLAP made learning course concepts easier, made them feel more engaged in the course and made the course more enjoyable. We also describe changes and additions to JFLAP we have made based on feedback from users. These changes include new algorithms such as a CYK parser and a user-controlled parser, and new resources that include a JFLAP online tutorial, a wiki and a listserv.

Categories and Subject Descriptors

F.4.3 [**Theory of Computation**]: Mathematical Logic and Formal Languages Formal Languages; D.1.7 [**Software**]: Programming Techniques Visual Programming

General Terms

Theory

Keywords

JFLAP, automata, formal languages, pumping lemma, CYK parser

1. INTRODUCTION

The formal languages and automata (FLA) course has traditionally been taught with little engagement or feedback, using pencil and paper exercises limited to small problems. We have developed JFLAP[18, 17], educational software used in this course to provide a more visual, interactive and applied experience with theoretical concepts. JFLAP allows one to create and simulate several types of automata, to create and parse strings in grammars, and to experiment with proof constructions such as converting a nondeterministic finite automaton (NFA) to a deterministic finite automaton (DFA) and then to a regular expression or regular grammar. In JFLAP one can *visualize* the finite automaton or parse tree in a graph format, *interact* with the automaton by running it with different input strings and *see applications* by creating a DFA as part of the SLR parsing process.

Many other tools for experimenting with automata theory have been developed. Turings World[2] is for experimenting with Turing machines. Forlan[24] and Emulator[26] are for experimenting with regular languages. The tool jFAST[27] allows experimentation with finite automata, pushdown automata and Turing machines. Grinder[9] focuses on finite state automata. RegEx[5] allows experimentation with regular expressions. Taylor[25] explores several types of machines. Ross[7] is developing a hypertextbook for many topics in automata theory. Many of these tools focus on a small set of topics. JFLAP is a software tool under development for over 17 years to incorporate an extensive number of topics in automata theory in one tool.

In the last few years, several tools have been developed in Algorithm Visualization for different areas of computer science with the majority for algorithms and data structures. Examples of these include Balsa-II [6], XTango [21], Samba [22], AACE [8], Animalscript [19] and JHAVÉ[13] for algorithms animation, tools for graph theory such as Guess [1], and tools for discrete mathematics such as Combinatorica [15, 20], SetPlayer [3] and LINK [4].

How effective are such tools in the learning process? There are some results in this area, but there is still much research to be done. A small study using Samba [23] showed that there was a learning advantage for students who interacted with algorithm animations in a lab. In [10] a metastudy of 24 experimental studies on the educational value of algorithm visualization is presented. Nine of the studies had no significant differences detected, but the remaining 15 had results such as participants viewing or constructing animations scored significantly higher or outperformed in some way. In [14] an ITICSE Working Group explored the role of visualization and engagement in computer science education and proposed several hypotheses to compare the types of engagement of 1) no viewing, 2) viewing, 3) responding, 4) changing, 5)constructing, and 6) presenting, with

*The work of all the authors was supported in part by the National Science Foundation through NSF grant DUE-0442513

SIGCSE'09, March 3–7, 2009, Chattanooga, Tennessee, USA.

the hypothesis that the engagements offer significantly better learning outcomes the higher the number associated with it (with *presenting* the best outcome).

How effective is JFLAP in enhancing the learning process and what additional value might JFLAP add to the FLA course? This paper describes a two-year study of fourteen universities using JFLAP in their FLA course including feedback from both students and faculty.

In Section 2 we give an overview of the study and its results. In Section 3 we describe changes and additions to JFLAP based on feedback from users in our study. In Section 4 we provide an analysis of the worldwide usage of JFLAP, and in Section 5 we give concluding remarks.

2. JFLAP STUDY

2.1 Overview

Our two year JFLAP study was held from Fall 2005 to Spring 2007 and included twelve universities the first year and an additional two universities the second year. The fourteen schools were Duke University, Emory University, Fayetteville State University, Norfolk State University, Rensselaer Polytechnic Institute, Rochester Institute of Technology, San Jose State University, United States Naval Academy, University of California Davis, University of Houston, University of North Carolina, University of Richmond, Virginia State University, and Winston-Salem State University. These schools included a large mix of public and private, large and small schools with several HBCU schools to include a racially diverse population.

In the summer of 2005 we held a workshop with 18 participants including mostly faculty adopters, three evaluators, and three student research assistants. A followup workshop was held in the summer of 2006 with 17 participants, five of them new. All fourteen schools participated in at least one of the workshops. However, with the nationwide trend of enrollments dropping in computer science and the automata theory course an optional course at some schools, not all schools were able to participate in the collection of data due to low enrollments or courses not held every year.

Data was collected from students and faculty adopters through knowledge tests, surveys, structured telephone interviews and face-to-face focus groups.

2.2 Results of Study - First Year

The first year of the study in 2005-2006 we were able to collect data from 55 students (pretest) and 33 students(posttest) at the seven schools: Duke University, Norfolk State University, University of Richmond, University of Houston, Virginia State University, Fayetteville State University and University of California Davis. The data collected from students included a pre and post knowledge test, and a survey of computer science attitudes, JFLAP implementation and usability.

The student usability survey can be summarized as follows. All the instructors did indeed use JFLAP materials in their course, with the primary use in class for demonstrating, and extensive use for homeworks. JFLAP was used very little as part of exams. Student opinion as to usefulness of JFLAP as a tool to improve their grade was mixed, but a large majority used JFLAP in 50% or less of their course. Students had high opinion of the JFLAP software in terms of its overall usability and power. They felt it was an easy and useful tool to design, run and interpret simulations. Specific questions from the usability survey are combined with year 2 and shown in the next section.

Selected questions from the software implementation survey shown below show that the majority of students used JFLAP to study for exams and thought JFLAP helped them to get a better grade.

Question	YES/NO	No.	%
Did you use JFLAP software to study for inclass exams?	YES NO	20 16	55% 45%
Did you feel you had time to learn how to use the JFLAP software?	YES NO	33 2	94% 6%
Did you feel that using the software took time away from other study activities?	YES NO	3 33	8% 92%
Did the time and effort it took to use JFLAP help you get a better grade in the course?	YES NO	23 13	64% 36%
Was it easier to use JFLAP software or was it easier to draw it out by hand?	software by hand	30 6	83% 17%
Did you feel you would have done as well in the course if you had not used JFLAP?	YES NO NA	18 13 5	50% 36% 14%

The attitudes survey showed that students were confident in their abilities as one would expect for upper level computer science majors, and enjoy engaging in the work and problem-solving that is part of their course work.

The pretest and posttest given at the beginning and end of the semester showed that the majority of students learned key content in the course. There were too few responses and no control group this year to statistically gauge the efficacy of JFLAP for improving learning.

At the end of each semester, the research team conducted structured telephone interviews with faculty adopters and at the end of the year a focus group of faculty during the second workshop. We give a summary from these faculty discussions.

- The faculty felt strongly that JFLAP not be used as part of exams. The reason cited in some cases had to do with concerns about cheating and/or the tests were all paper and pencil. One instructor did use it for a test but found that students spent too much time trying to perfect their machines.

- JFLAP was mostly used in homework and some as part of in class demonstrations. For those who did not use JFLAP as part of the class lecture/demonstration, their reasons were varied but usually centered around the fact that they delivered lectures with static multimedia (e.g., PowerPoint slides of text and static graphics) or did not use technology at all.

- One instructor noted that JFLAP was good for showing how to build machines, but not why they worked.

- One instructor felt that JFLAP was better at the early stages of learning but was cumbersome for more advanced problems.

- One instructor did try JFLAP in a group project with positive feedback from both the instructor and students.
- Another instructor indicated that they mostly used JFLAP as a tool in lecture but also for electronically submitted homework assignments.
- Some instructors used JFLAP as a graphics generation tool since it was often quicker to create a diagram with JFLAP and capture a graphic of it for use in lecture than to draw the diagram by hand.
- Another instructor used JFLAP for a robot programming assignment that the students really liked. They noted that it could also be used as an alternative to using lex or yacc.
- Unlike many of the instructors, one did not use it much in lecture, but instead used it more for assignments and office hours. They felt that it was a useful tool to help get students to see hard to visualize cases, but that it was not a substitute for theory instruction.

2.3 Results of Study - Second Year

In the second year of the study, two schools were added to the study. Feedback from instructors and results of the first year study were used to refine the evaluation instruments in preparation for a larger control group study in year 2. Changes included shortening the pre and post knowledge test with some of the remaining questions modified. The student survey was also modified based on instructor feedback. We were able to collect data from students at six universities including 75 students (pretest and posttest), 67 students (pretest only) and 34 students (posttest only). The six schools were Emory University, University of North Carolina, Duke University, San Jose State University, University of Houston, and Virginia State University. San Jose State University was our largest data collection site and had a control group.

The table below shows combined results from the usability survey in years 1 and 2, showing that students had an easy time using JFLAP and thought is was a good tool.

Question		
How easy was it to use the drawing tool of JFLAP? (134 respondents)	Very Easy	31%
	Easy	48%
	Neither	15%
	Difficult	5%
	Very Difficult	0%
How easy was it to run the automata you designed in JFLAP? (134 respondents)	Very easy	33%
	Easy	47%
	Neither	12%
	Difficult	6%
	Very Difficult	2%
How easy was it to interpret results from the test run in JFLAP? (134 respondents)	Very Easy	23%
	Easy	45%
	Neither	19%
	Difficult	10%
	Very Difficulty	3%
What is your overall assessment of the JFLAP software? (133 respondents)	Very Poor	2%
	Poor	4%
	Neither	11%
	Good	63%
	Very Good	20%

In the next table, we show two of the questions from the year two implementation survey that show one third of the students used JFLAP 21% or more of study time for exams, and 29% used JFLAP to work additional problems.

QUESTION	Time	Response
When preparing for exams what percentage of study time involved the use of JFLAP software? (100 responses)	0-20%	68%
	21-40%	16%
	41-60%	11%
	61-80%	3%
	81-100%	2%
How often did you use JFLAP to do additional practice problems? (99 responses)	Never	46%
	Rarely	25%
	Occasionally	21%
	Often	4%
	Very Often	4%

The next table shows new questions that were added in year two to the usability study that show the majority of students felt JFLAP made them more engaged and made learning concepts easier.

Question		
Using JFLAP made the course more enjoyable for me. (98 responses)	Strongly Agree	12%
	Agree	51%
	Neither	25%
	Disagree	6%
	Strongly Disagree	4%
Using JFLAP made me feel more engaged in the course. (98 responses)	Strongly Agree	13%
	Agree	59%
	Neither	15%
	Disagree	9%
	Strongly Disagree	3%
Having access to JFLAP made learning course concepts ... (97 responses)	Much harder	1%
	Harder	5%
	Neither	26%
	Somewhat easier	54%
	Much easier	14%

The attitudes survey showed that the population sample was a little less confident in their abilities. The pretest and posttest again showed that students provided significantly more correct answers at posttest. However the difference between the control group and the JFLAP group was not statistically significant.

3. EXPANDING INTERACTION IN JFLAP

This section describes changes to expand the interaction in JFLAP. Many of these changes result from feedback from users either directly or from the study. New interaction includes a user-controlled parser, and enhancements to the pumping lemma. New algorithms include an implementation of CYK parser, and a proof construction from Turing machine to unrestricted grammar. Several changes allow JFLAP to fit more closely with a larger variety of textbooks such as an option to automatically add in a trap state. New resources are also given in this section including a new online tutorial.

3.1 User-Control Parser

JFLAP had three parsers: brute-force parser, LL(1) parser and SLR(1) parser. The later two are used mostly by those studying compilers. The brute-force parser is used most

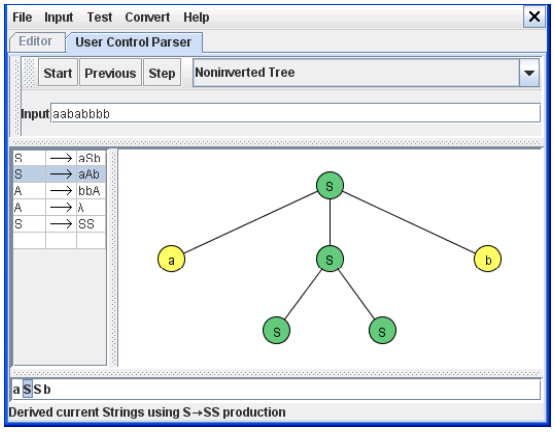

Figure 1: User-control Parser

heavily in testing strings but can result in poor performance. We have added a faster parser, the CYK parser described in the next section. However, all the parsers were lacking the interaction of experimentation with rejected strings. In all the parsers you type an input string and it is rejected or accepted. If it is accepted you can step through the derivation or parse tree. If it is not accepted you can only wonder why it was not accepted.

The user-control parser was developed so students could try "what-if" parsing. The user guides the parse, and if a deadend is reached, they can backtrack and try replacing a variable with another rule. Figure 1 shows an example of a step in a user-control parse of the string `aababbbb`. The user clicks start and the S (start) variable appears in the parse tree window. To move forward, the user must now select a replacement rule. When *step* is selected the replacement rule is executed. If there is a choice of variables to replace, the user must also select which variable from the sentential form shown in the bottom of the window. In Figure 1 the user has selected the `S`→`aSb` rule and the first of the two `S`'s in the sentential form `aSSb`.

3.2 CYK Parser

We have implemented the Cocke-Younger-Kasami (CYK) parsing algorithm in JFLAP. Internally, the grammar is first converted to the CNF grammar and then mapped back to the original grammar. The user only sees the accept or reject response and can step through the parsing of the string with the original grammar. This algorithm is substantially faster then the brute-force parser. The brute-force parser still has value in that its algorithm is fairly easy to understand, and its parsing difficulties are easy to see when JFLAP takes a long time to parse.

3.3 Expanding Pumping Lemma

The pumping lemma game in JFLAP described in [16, 12] was a one-way game, the user against JFLAP with the user always starting the game. To create more interaction we have modified the game so that either the user can go first or JFLAP can go first. This gives the user more experience with the choices needed in using the pumping lemma. This change was applied to both the regular and context-free pumping lemma games.

3.4 Turing Machine to Unrestricted Grammar

We have added the Turing machine to unrestricted grammar proof construction explained in [12] to JFLAP. This construction works with small examples and we provide two such examples in the new online JFLAP Tutorial.

The user starts with a Turing machine. The construction proof works by generating three sets of productions that include variables with special notation from the proof. The first set of productions are generated by clicking on the Start variable and include an encoded string with a start state, a start string and an arbitrary number of blanks on either side. The next set of productions are generated by clicking on a labeled transition and include encoded productions to mimic the transitions. The third set of productions are generated by clicking on the final state and include rules to remove everything but the derived string w. Once the grammar is generated it appears in a new window. Because it includes variables with special notation there is a parser specifically for this grammar that can be used to parse a string.

3.5 Other new items

We describe some of the other new items in JFLAP.

- To conform with more textbooks, we have added an option to automatically add a trap state and arcs to make a DFA complete.
- We have created additional transition diagram graph layouts.
- The user can load an input string from a file. This is helpful when viewing the universal Turing machine.
- There is an option to identify the type of grammar.
- We have written an online tutorial that includes JFLAP file examples that can be downloaded.

4. JFLAP USAGE AND RESOURCES

We have tracked JFLAP's usage around the world in a number of ways including an online form when downloading JFLAP. JFLAP has over 64,000 downloads from users in 161 countries since January 2004. The majority of users are undergraduates (41%). Other users include graduate students (26%), faculty(22%) and a small number of K-12 teachers (<2%, 713) and K-12 students (<2%, 1093). Of those using JFLAP in a course, the majority (51%) say it is not required, showing students are using it for courses even though it is not required. The majority of use for JFLAP is taking a course (60%) teaching a course (17%), Research (8%) or other. JFLAP and source are provided for free.

New resources on the JFLAP web pages[17] include a wiki, a listserv, and a list of books and papers referring to JFLAP. An example is the paper [11] in which they modify JFLAP to allow students to write Java programs to alter automata to aid in understanding topics such as Church-Turing thesis and undecidability.

5. SUMMARY

The results of our two year study showed that all the faculty used JFLAP in their courses and that students had a high opinion of JFLAP. The faculty in our study mostly used JFLAP for homework. Some used it in class demonstrations. JFLAP was used very little in exams. Four-fifths

of the students thought JFLAP was easy to use to draw automata, simulate and interpret the results. The majority of students felt that having access to JFLAP made learning course concepts easier, made them feel more engaged in the course and made the course more enjoyable. Over half of the students used JFLAP to study for exams, and thought that the time and effort spent using JFLAP helped them get a better grade in the course.

With feedback from the study and other users, we have made several additions and changes to both JFLAP and the JFLAP web site. JFLAP includes new algorithms for experimenting with concepts such as the user-control parser to try to derive a string from a grammar on your own. The JFLAP web site has new resources for learning such as the online tutorial and for sharing with the new wiki and listserv.

6. ACKNOWLEDGMENTS

We thank the evaluators and consultants involved in the study: Joseph Bergin, Rockford Ross, Thomas Finley and Peter Linz. We thank the fourteen faculty adopters for participating and providing us feedback.

7. REFERENCES

[1] E. Adar. Guess: A language and interface for graph exploration. In *SIGCHI*, pages 347–363, April 2006.

[2] J. Barwise and J. Etchemendy. *Turing's World 3.0 for the Macintosh*. CSLI, Cambridge University Press, 1993.

[3] D. Berque and et al. *The SetPlayer System: An Overview and a User Manual*. Department of Computer Science, Technical Report 91-17, Rensselaer Polytechnic Institute, Troy, New York, 1991.

[4] J. Berry. Improving discrete mathematics and algorithms curricula with link. In *ACM SIGCSE/SIGCUE Conference on Integrating Technology in Computer Science Education*, 1997.

[5] C. W. Brown and E. A. Hardisty. Regexex: An interactive system providing regular expression exercises. In *Thirty-eighth SIGCSE Technical Symposium on Computer Science Education*, page (to appear). SIGCSE, March 2007.

[6] M. Brown. Exploring algorithms using balsa-ii. *Computer*, 21(2):14–36, May 1988.

[7] J. Cogliati, F. Goosey, M. Grinder, B. Pascoe, R. Ross, and C. Williams. Realizing the promise of visualization in the theory of computing. *JERIC*, 5, 2005.

[8] P. Gloor. Aace - algorithm animation for computer science education. In *Proceedings of the 1992 IEEE Workshop on Visual Languages*, pages 25–31, 1992.

[9] M. T. Grinder. A preliminary empirical evaluation of the effectiveness of a finite state automaton animator. In *Thirty-fourth SIGCSE Technical Symposium on Computer Science Education*, pages 157–161. SIGCSE, February 2003.

[10] C. Hundhausen, S. Douglas, and J. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, 13(3):259–290, 2002.

[11] J. Jarvis and J. Lucas. Incorporating transformations into jflap for enhanced understanding of automata. In *Thirty-ninth SIGCSE Technical Symposium on Computer Science Education*, pages 14–18. SIGCSE, March 2008.

[12] P. Linz. *An Introduction to Formal Languages and Automata, 4th Edition*. Jones and Bartlett, Sudbury, MA, 2006.

[13] T. Naps. Jhave: Supporting algorithm visualization. *IEEE Computer Graphics*, 25:49–55, 2005.

[14] T. Naps, G. Rossling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. A. Velazquez-Iturbide. Exploring the role of visualization and engagement in computer science education, report of the working group on improving the educational impact of algorithm visualization, 2002. ITICSE.

[15] S. Pemmaraju and S. Skiena. *Computational Discrete Mathematics Combinatorics and Graph Theory with Mathematica*. Cambridge University Press, 2003.

[16] S. Rodger, J. Lim, and S. Reading. Increasing interaction and support in the formal languages and automata theory course. In *The Twelfth Annual Conference on Innovation and Technology in Computer Science Education*, pages 379–383. ITICSE, June 2007.

[17] S. H. Rodger. Jflap web site, 2008. www.jflap.org.

[18] S. H. Rodger and T. W. Finley. *JFLAP - An Interactive Formal Languages and Automata Package*. Jones and Bartlett, Sudbury, MA, 2006.

[19] G. Roessling and B. Freisleben. Animalscript: An extensible scripting language for algorithm animation. In *Twenty-second SIGCSE Technical Symposium on Computer Science Education*, pages 70–74, Feb 2001.

[20] S. Skiena. *Implementing Discrete Mathematics*. Addison Wesley, Redwood City, CA, 1990.

[21] J. Stasko. Tango: A framework and system for algorithm animation. *IEEE Computer*, pages 27–39, Sept 1990.

[22] J. Stasko. Using student-built algorithm animations as learning aids. In *Twenty-eighth SIGCSE Technical Symposium on Computer Science Education*, pages 25–29, March 1997.

[23] J. Stasko and A. Lawrence. Empirically assessing algorithm animations as learning aids. In *Software Visualization*, pages 419–438. MIT Press, 1998.

[24] A. Stoughton. Experimenting with formal languages. In *Thirty-sixth SIGCSE Technical Symposium on Computer Science Education*, page 566. SIGCSE, February 2005.

[25] R. Taylor. *Models of Computation and Formal Languages*. Oxford University Press, New York, 1998.

[26] L. F. M. Vieira, M. A. M. Vieira, and N. J. Vieira. Language emulator, a helpful toolkit in the learning process of computer theory. In *Thirty-fifth SIGCSE Technical Symposium on Computer Science Education*, pages 135–139. SIGCSE, March 2004.

[27] T. M. White and T. P. Way. jfast: A java finite automata simulator. In *Thirty-seventh SIGCSE Technical Symposium on Computer Science Education*, pages 384–388. SIGCSE, March 2006.