Duke University — Department of Computer Science

Department of Computer Science
Box 90129
Duke University
Durham, NC 27708-0129
jflap@cs.duke.edu

July 11, 2007

2007 Premier Award

Greetings:

Enclosed is a submission of JFLAP, a software tool for formal languages and automata theory in computer science, and a tutorial on JFLAP for consideration for the 2007 NEEDS Premier Award. Included are 15 copies of the submission packet. JFLAP was submitted to NEEDS last year. The newest version of JFLAP, Version 6.1, is available for downloading from `www.jflap.org`. The web tutorial on JFLAP is also available from this same page.

I am the sole faculty member who has developed JFLAP and related tools over the past 17 years, supervising many students. I'm including several former and current Duke students I've contacted who worked on the current version of JFLAP and its tutorial: Thomas Finley (tomf@cs.cornell.edu) (now at Cornell University), Stephen Reading (srr27@duke.edu), Bartlett Bressler (bpb3@duke.edu), Ryan Cavalcante (RyanCav@microsoft.com) (now at Microsoft), Jinghui Lim (jl95@duke.edu), Chris Morgan (cmm24@duke.edu) and Kyung Min (Jason) Lee (kl44@duke.edu). I am authorized to submit the software for the Premier competition. My contact information is above. I hold the copyright on JFLAP.

Susan Rodger authorizes NEEDS to become a non-exclusive distributor of the JFLAP software as submitted.

JFLAP is software for experimenting with formal languages and automata theory and can be used with a course in Formal Languages, Discrete Mathematics, or Compilers. With JFLAP one can create and test automata, pushdown automata, multi-tape Turing machines, regular grammars, context-free grammars, unrestricted grammars and L-systems. One can also experiment with LL and SLR parsing and proofs such as converting an NFA to a DFA, to a minimal state DFA and to a regular expression. The JFLAP web page, www.jflap.org, has the JFLAP software, a JFLAP tutorial and additional resources available.

Sincerely,

Susan H. Rodger
Associate Professor of the Practice

Contact info for Dept Chair and Deans at Duke

Pankaj Agarwal
Professor and Chair
Department of Computer Science, Box 90129
Duke University, Durham, NC 27708-0129
Email: pankaj@cs.duke.edu
Phone: (919)-660-6548

Robert J. Thompson, Jr.
Dean of Trinity College
114 Allen Building
Box 90042
Durham, NC 27708
Email: bobt@asdean.duke.edu
Phone: 919-684-3465

George McLendon
Dean of the Faculty of Arts and Sciences
104 Allen Building
Box 90046
Durham, NC 27708
Email: george.mclendon@duke.edu
Phone: 919-684-4510

JFLAP - Software for Experimenting with Formal Languages

1 Introduction

A major problem in computer science education is that many students obtain only a superficial understanding of theory, even though theoretical concepts provide the fundamental basis for most areas of computer science. In particular, a thorough understanding of the theory of formal languages and automata (FLA) is crucial in designing programming languages and compilers. However, the traditional FLA course is taught in a nonvisual and pencil-paper problem solving manner with no programming component. Students find this approach frustrating as they have no visualization to relate to, they do not receive immediate feedback on problems, and furthermore it is tedious and uninteresting to them to determine if their solutions are correct. Many students turn in homework with errors as they do not bother to verify their solutions by hand. This contrasts starkly with the hands-on nature in most of their other computer science courses which contain programming assignments. In addition, many students leave the FLA course without understanding the importance of this material. They are told it has applications in other areas, but they don't experience the applications.

The material in the traditional FLA course is important to the computer science major, as mentioned in the Computer Science Volume of the ACM IEEE Computing Curricula 2001 document in several places. Under intermediate courses, the core course CS 210, Algorithm Design and Analysis, includes topics in "an introduction to automata theory and its application to language translation." Under advanced courses there is a course called CS 311 Automata and Language Theory. This course already includes such topics from the Introductory Courses including discrete mathematics, formal proofs, algorithmic thinking and testing and debugging.

We have developed a tool, JFLAP, that allows students in an FLA course to experiment with the theory in this course, receiving immediate feedback, and making the course more interesting to students. JFLAP (Java Formal Languages and Automata Package) is a long term NSF-supported software project (NSF grants DUE 9596002, DUE 9555084, DUE 9752583, and DUE 0442513) . With JFLAP one can interactively build automata and grammars and run test input on them. Also with JFLAP one can experiment with many of the construction-type proofs taught in this course.

JFLAP is freely available and has been used in several types of computer science courses around the world. We have published twelve papers, a JFLAP book, given over twenty presentations, and held three JFLAP workshops. We recently developed a web tutorial on JFLAP that is available for free on the JFLAP web site. This tutorial guides the user through the usage of JFLAP and includes many examples with JFLAP files for loading and experimenting with.

2 Background

The development of JFLAP began back in 1990 when Rodger was teaching automata theory at Rensselaer and students were turning in homework using pencil and paper. Problems to create an automaton and grammar were almost always incorrect as it was too tedious to test and trace inputs on a handwritten model. JFLAP began as several smaller tools (NPDA, LR Parser, LL Parser, TuBB, FLAP) for different concepts in formal languages, written in C++ and X windows. We made these tools freely available and universities started using them. However, with the many versions of C++ and X windows, some people had difficulty installing it. Around 1996 at Duke University, FLAP was rewritten in Java, became JFLAP and was much easier to install. Other smaller tools were continually developed (JeLLRap, Lsys, Pâté) and eventually integrated into JFLAP. Around 2002, JFLAP was rewritten using Swing and has continued to evolve.

Rodger has supervised many undergraduate and graduate students in the development of JFLAP and related tools at Rensselaer Polytechnic Institute from 1990-1994 and at Duke University from 1994 to the present. Those students who have worked on JFLAP related tools at Rensselaer include Dan Caugherty, Mike James, Steve Blythe, Bhasker Vasudevan, Grant Poladian, Eric Luce, Danny Daglas, Jeffrey Nesheiwat, and Jasper Wong. Those students who have worked on JFLAP related tools at Duke include Greg Badros, Steve Wolfman, Ben Hardekopf, Ken Leider, Jason Salemme, Anna Bilska, Edwin Tsang, Magda Procopiuc, Octavian Procopiuc, Alex Karweit, Robyn Geer, Eric Gramond, Lenore Ramm, Ted Hung, Ryan Cavalcante, Thomas Finley, Bart Bressler, Stephen Reading, Jinghui Lim, Chris Morgan and Jason Lee. Many of these students are co-authors with me on JFLAP papers and one (Finley) is a co-author on the JFLAP book.

JFLAP has been supported by four NSF grants. The first three grants, NSF DUE 9354791 from 1994-96 (this grant was transfered to Duke as NSF DUE 9596002), NSF DUE 9555084 from 1996-98, and NSF DUE 9752583 from 1998-2002) were focused mostly on the development of JFLAP and dissemination. The fourth and current grant, NSF DUE CCLI 0442513 from 2005-2008, continues the development of JFLAP but focuses on the evaluation of JFLAP and includes fourteen faculty adopters. More information on the study is described in Section 6.

3 Objectives

The educational objectives of the JFLAP software are the following.

- To provide students with a tool for visualizing and interacting with theoretical models and concepts in computer science.

  JFLAP provides visualizations of automata and grammars; transition diagrams (graphs) of the three types of automata and visual parse trees for the grammars. With JFLAP one can construct, test and debug several theoretical models of computation.

- To provide students with the means of experimentation of construction-type proofs in formal languages.

  After learning about the proof that the class of nondeterministic finite automata (NFA) are equivalent to the class of deterministic finite automata (DFA), one can use JFLAP to construct an NFA and then in steps construct the equivalent DFA. One can then run inputs on both of them to see their equivalence.

- To provide students with the means for experimenting with formal languages in a computational manner that can only be done with the use of a computer.

  Besides the ability to construct an automaton and run it on multiple inputs, JFLAP has many other uses that would be too tedious to do by hand. For example, JFLAP comes with a Universal Turing machine students can run. It would be too tedious for students to draw one on paper and try to trace it. As another example, in Section 3.7 of our SIGCSE 2006 paper included in the appendix, we present two grammars that are equivalent, the first with a λ-production. On input aaababaabbb, the first grammar generates 13286 nodes in the derivation tree, and the other grammar derives only 335 nodes.

- To provide students with hands-on experimentation of applications in theoretical computer science.

  L-Systems are used to model biological systems and fractals. With the L-Systems in JFLAP, students use another type of grammar that researchers use to model plants in environments. With SLR Parsing, students see how DFA and pushdown automata can be used in writing a compiler.

- To enhance (not replace) the theoretical nature of the formal languages course.

  JFLAP is not meant to replace the theoretical proofs taught in a formal languages course, but to complement them. Students should still study a theoretical proof with its mathematical notation, and then JFLAP will enhance and reinforce the theory with visualizations and experimentation with a concrete example

4 Overview of JFLAP

JFLAP (Java Formal Languages and Automata Package) is an extensive visual and interactive tool for designing and experimenting with different types of automata and grammars, studying proofs by the construction of examples, studying parsing through LL, SLR and brute force methods, and transforming grammars.

Automata in JFLAP

With JFLAP one can graphically design and animate three types of automata: finite automata, pushdown automata, and multi-tape Turing machines. These machines can be deterministic or nondeterministic. In using the tool, one graphically draws one of the machines as a transition diagram, enters an input string, and then controls the tracing of the input string through the machine. There are four simulation modes for testing input including two types of step run that allows one to step through the automaton, a fast run that returns information and validity of acceptance, and a multiple input tester that will check the acceptance of many input strings at the same time. In addition, students can save and retrieve their designs and load examples provided.

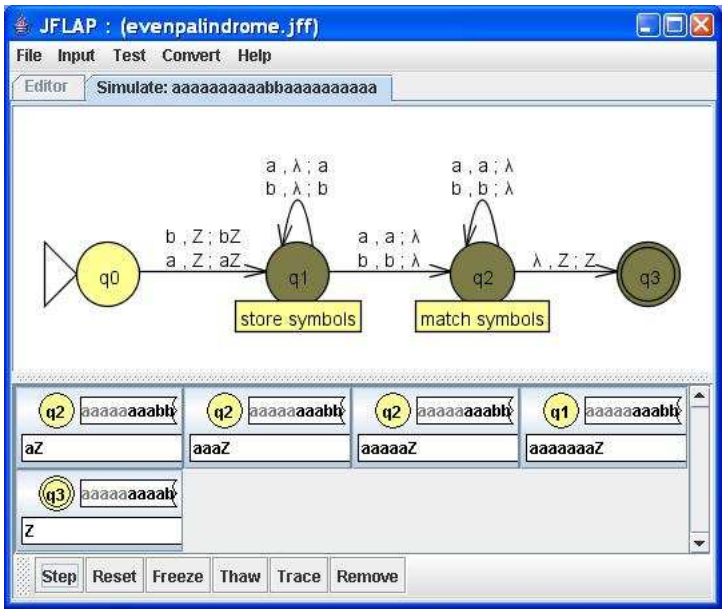


Figure 1: JFLAP NPDA example simulation

Figure 1 illustrates an NPDA created in JFLAP that represents nonempty palindromes of even length. This NPDA was created in the editor mode (see the editor tab faded out) but is now in simulation mode. This figure illustrates the nondeterminism in JFLAP. It shows one step in the execution of this NPDA on the simulation of input string $a^{10}bba^{10}$, showing five possible configurations at this time step. Each of the five configurations show the current state (shown as a circle), the input string (shown to the right of the state) with the part of the input already processed shaded, and the current stack contents (shown below the state). Notice that three of the states in the NPDA are shaded at this time step as these three states all have an active configuration. With nondeterminism, JFLAP creates a tree of all configurations (not shown to the user). At each step it expands any configuration that is a leaf node in the tree (with pruning), and then shows the current configurations (the current leaf nodes). Another type of simulation mode, fast run (not shown), with this same input string results in the input accepted, and displays only the acceptance path, which in this case would be of length 24. Also shown in the figure is the ability to annotate states. An annotation has been added to state $q1$ (store symbols) and an annotation has been added to state $q2$ (match symbols).

Studying Proofs in JFLAP

JFLAP has been enhanced to allow one to study the proofs of several theorems that focus on conversions of languages from one form to another by constructing an example and transforming the example into the other form. With JFLAP one can enter an NFA and convert it to a DFA and then to a minimum state DFA, or to a regular expression or a regular grammar. One can enter a regular expression and convert it to an NFA, or one can enter a regular grammar and convert it to an NFA. One can enter a context-free grammar and convert it to an NPDA in one of two formats, either an NPDA that models LL(1) parsing or an NPDA that models SLR(1) parsing. One can enter an NPDA and convert it to a CFG. The conversion process is interactive and provides helpful error messages as the user constructs the new form.

For example, in the conversion from NFA to DFA, the user would first construct an NFA. After selecting the conversion to DFA option, the user would start constructing the new DFA by adding states. For each state in the DFA, the user must enter the number from the corresponding states from the NFA. If the user enters the wrong state number, JFLAP informs them. The user must connect the states with the appropriate transition arc. JFLAP informs the user when their DFA is complete and the user has the option of placing the newly constructed DFA in a new editor window.

Parsing in JFLAP

There are three types of parsing in JFLAP, a brute-force parser, LL parser and SLR parser. These were incorporated from our tools JeLLRap and Pâté.

The brute-force parser allows one to parse restricted (regular and context-free) and unrestricted grammars, showing the derivation and parse tree (including a "parse tree" for unrestricted grammars!). The LL and SLR parser tools construct parse tables through a series of steps for LL(1), and SLR(1) grammars, and then simulate the parsing of input strings using the constructed table and a stack, at the same time building the parse tree. One can still build the parse table and also parse non-SLR(1) grammars by selecting choices for conflict items.

For example, one enters an SLR(1) grammar, a window appears and one enters in FIRST sets. When correct, one enters in FOLLOW sets. When correct, one builds the DFA that models the stack, including selecting marked rules for each state. When correct, one enters in the entries of the SLR parse table. When correct, one enters an input string, and steps through the parsing of the string seeing the stack and the parse tree being built.

Transforming grammars in JFLAP

JFLAP allows the transformation of grammars from one form to another. The transformation component is an instructional tool for converting a context-free grammar to CNF, including steps for removing lambda productions, unit productions, and useless productions.

L-systems in JFLAP

With JFLAP one can create and render L-systems. One enters an L-system including an axiom, grammar and graphical parameters. The strings of the L-system can then be visually displayed in order. For example, an L-system can be used to describe the growth of a tree or plant. When rendered the strings generated from the grammar are visually displayed showing the tree or plant growing.

Newest Version of JFLAP (6.1) and Current JFLAP work

We have included the newest version of JFLAP along with a new web tutorial in the submission. The new features in this version of JFLAP include Turing Machine Building Blocks, Moore and Mealy machines, experimentation with regular and context-free pumping lemma, and Batch grading to support the grading of JFLAP files. These new additions are explained in our SIGCSE 2006 paper and ITiCSE 2007 paper in the appendix. With Turing machine Building Blocks, one can build a Turing machine, name it and then reuse it as a submachine or building block in another Turing machine. The run control allows one to control the simulation, executing the building block in one step, or entering the building block and stepping through state by state. This allows one to create more interesting Turing machines easily such as multiplying two unary numbers.

We receive a steady stream of email from our users with requests for additions to JFLAP. Fo example, we received many requests for Moore and Mealy machines, batch grading and preferences to change the empty string to epsilon. All of these requests are in the current version of JFLAP.

5 Use of JFLAP Software

5.1 Types of Courses that use JFLAP

JFLAP's target audience is for an undergraduate formal languages course that includes any of the topics on automata, grammars and parsing. The majority of the letters in the appendix is for this type of course. JFLAP has been used in more than five other computer science courses. Berwick at MIT states in his letter that he uses JFLAP in an Artificial Intelligence course and a Natural Language Processing course. After receiving his letter we added Moore and Mealy machines to JFLAP. Brown at the College of Wooster and Harvey at Robert Morris University state in their letters they use JFLAP in a Discrete Mathematics course. Dooley at Knox College states in his letter that he uses JFLAP in three courses: Algorithm Design and Analysis, Theory of Computing and Software Development, and Software Design. Wallingford at University of Northern Iowa and Walker at Hiram College state in their letters that they use JFLAP in a Compiler course.

JFLAP is also being used in high school. Andreas Rittershofer, a mathematics, physics and computers science high school teacher at Dietrich-Bonhoeffer-Gymnasium in Metzingen, Germany writes in a letter "One part of the courses in computer science is about regular expressions, grammars, finite automata, stack automata, turing machines, ... In the case of some spare time we also create some flowers und bushes with L-Systems. ... JFlap is the ideal tool for my computer science courses... [JFLAP is] very easy to use, my students are familar with its handling in only a few minutes. I cannot image how to deal with automata without JFlap - and so do my students too."

5.2 The Usage of JFLAP in Courses

Instructors use JFLAP in many ways as shown in the letters of support and the included JFLAP publications in the appendices. One of the main uses is to use JFLAP during lecture. Instructors use JFLAP in the classroom to demo the use of JFLAP and to explain an algorithm by working through an example in JFLAP. Instructors can use JFLAP during lecture to work problems with students. For example, a Turing machine in JFLAP can be shown for a particular language and students are asked to determine if the Turing machine is correct and if not to correct it. Students can make suggestions and the instructor can make the changes and test them out in front of the class. Instructors can answer questions by "showing" the answer with JFLAP. JFLAP examples shown in class can be saved in files and made available for students to recreate the lecture.

Other uses of JFLAP by an instructor include the following. Instructors can use JFLAP in office hours one on one to help a student understand a concept. Baruah from UNC states in his letter that he uses it extensively in office hours. Instructors can use JFLAP to create pictures for slides for a lecture or for pictures in a homework writeup. JFLAP can be used as a separate lab. JFLAP can be used for homework assignments in a variety of ways. Students can load a JFLAP file and experiment with it to determine the language it represents. Students can construct their own examples of automata or grammars for a particular language and then turn in the file. Graders can have a set of test strings, load student files, and run them in the multiple run window easily determining the correctness. There are several examples of problems that could be used as homework that explore formal languages in computational ways included in the 2006 SIGCSE paper in the appendix. Many problems in automata theory textbooks can be worked using JFLAP. The JFLAP book has many exercises and all the files used in this book are available on the www.jflap.org site.

We have used FLAP/JFLAP since 1994 at Duke in the course CompSci 140, Mathematical Foundations of Computer Science. We use JFLAP extensively during lecture for demos and explaining algorithms and students use JFLAP for homework assignments, turning in JFLAP files. The course web site at `www.cs.duke.edu/courses/spring07/cps140` contains our lecture notes and homework assignments for the most recent version of this course. Many of the homework assignments have two parts, a theoretical part and a hands on experimentation part. Seven of the eight homework assignments used JFLAP.

Students use JFLAP in many ways. Students use it for homework whether it is required or not. Some of the letters in the appendix state that the instructor does not require JFLAP use, but instructors know students are using it as they turn in homework with JFLAP pictures. Students use JFLAP for recreating concepts learned in class and for working additional problems to understand a concept or to study for an exam.

5.3 JFLAP's use in Research

The JFLAP source is available and users are free to modify it for their needs. JFLAP has been used with modifications for research. A letter from Mariani from the University of Milano Bicocca states that he used JFLAP for his PhD Thesis, integrating it into a technique called BCT, for synthesizing models of component interactions from traces. JFLAP was successfully integrated into BCT to reuse its functionalities for handling and visualizing automata. A letter from Siamak Kolahi at Concordia University shows he used JFLAP in his MS thesis, as part of the interface for his automated composition platform. Tabakov, a PhD. student at Rice University states in his letter he used JFLAP for visualizing automata as part of his research. Bidder, a faculty member at the Swiss Institute of Technology in Zurich states in her letter that a student named Hildenbrand used JFLAP for his thesis, extending JFLAP to Mealy Automata.

5.4 World Usage

JFLAP is used around the world. When we released JFLAP 4.0 in January 2003, we created a user form to track and collect information on the downloading of JFLAP. From January 2003 to July 2006 there were over 35,000 downloads of the software JFLAP from over 160 countries. On `www.jflap.org` selecting "World Usage" gives statistics on these forms. All the countries in which someone has downloaded JFLAP are listed. The top five countries (from those who filled out the form) have the following number of downloads: USA (7779), Germany (2455), Brazil (2059), Mexico (1610), and Spain (1172). The number of downloads from each US institution are listed (from those who filled out the form). The top six number of downloads from US institutions had the following number of downloads: Harvey Mudd (325), RIT (276), Houston (224), Millersville (241), Duke (181), and UC Davis (161). 50 US schools have 20 or more downloads. Another 55 US schools have 10-19 downloads.

The JFLAP web site (www.jflap.org) and its predecessor page combined have received over 125,000 hits

since 1996. A search of JFLAP (which appears to be a unique name, and not a word in any language) on Google returns about 20,000 pages. These pages list JFLAP on several types of web pages including course web sites as a resource or as part of a homework, and as a reference to JFLAP as educational software such as Citidel. In some cases, the JFLAP web pages have been translated into another language. Note that Google can only search publicly available web sites and many schools use a system such as Blackboard for their courses which is password protected, so we expect that JFLAP appears on many more course websites. In the appendix we have included letters from mostly faculty and a few graduate students from fourteen countries who have used JFLAP.

6 Assessment of JFLAP Software

As shown in the previous section, JFLAP has obtained widespread use and as shown in the appendix, JFLAP has received positive feedback from faculty from a large variety of computer science courses and from high school to graduate school courses.

The current NSF grant supporting JFLAP, NSF DUE CCLI 0442513, includes an assessment of JFLAP. Eric Wiebe, Associate Professor in the Mathematics, Science and Technology Education Dept at North Carolina State University is in charge of the JFLAP study which will run from 2005-2007. In the first year there were twelve faculty adopters including Duke. The other eleven schools are University of California Davis, University of North Carolina Chapel Hill, Emory University, University of Richmond, Fayetteville State University, Winston-Salem State University, United States Naval Academy, Norfolk State University, Virginia State University, Rensselaer Polytechnic Institute and University of Houston. Four of these schools are predominately minority institutions. The faculty adopters attended a 2-day workshop in June 2005 and used JFLAP in their courses during the academic year 2005-2006.

We show some partial results from the first year of the study on the usage of JFLAP from 36 students from 6 institutions. See the appendix for more results.

- 20 of the 36 students replied that they used JFLAP to study for in-class exams.
- 30 of 36 students replied that it was easier to use JFLAP software than to draw it out by hand.
- 31 of 36 students replied that it was easy to install JFLAP and the other 5 stated it was neither easy nor difficult.
- When asked what the overall assessment of JFLAP was, 10 replied very good, 24 replied good, 2 replied neither poor nor good, and 0 replied poor or very poor.

For 2006-2007, we added two additional schools to our study, San Jose State University and Rochester Institute of Technology. Faculty adopters met for a second time in June 2006 at Duke for a 2-day workshop to discuss strategies and experiences with JFLAP. From this study we have over 130 responses. The analysis of the second year is under way this summer as San Jose State didn't complete their course until June 2007.

Ross, a Professor from Montana State University is a developer of educational software systems and a computer science evaluator on our current JFLAP NSF grant. He is also the editor of SIGACT News and the former editor of the SIGACT News Educational Forum. He includes a letter in the appendix stating 10 highlights of JFLAP. Four of those are:

- JFLAP is the only currently generally distributed software that animates the theory of computing.
- There is a class of students who find it relatively easy to master abstract concepts and do not really need a system such as JFLAP to help them learn. Such students nonetheless find such animation tools engaging and are often the ones who can be utilized to extend the software (i.e., enhance JFLAP).

- There is a much larger class of students who find it moderately to highly difficult to master abstract concepts. Many of these students are quite definitely helped by JFLAP. Perhaps the most appreciated aspect of JFLAP by these students is that they receive immediate and consistent feedback on exercises they perform within the software.
- A general trend noticed from the use of JFLAP is that most students, regardless of their innate talent, are more motivated and excited about learning the theory of computing when JFLAP is incorporated into the course. This may actually be the most important contribution of JFLAP.

He also states the following in his letter. "As already noted, JFLAP is a software system that supports the teaching and learning of the theory of computing. This subject is the foundation of the study of computer science, yet it is widely viewed by computer science students as the most dreaded and/or dry course in the curriculum due to its abstract, theoretical nature. In a nutshell, JFLAP aids the teaching and learning of many of the abstract concepts of the subject by brining them to life. That is, students are presented with computer-animated, visual depictions of the key abstract models of the theory, and they are allowed (and often required) to work with the models an active-learning mode as they progress through the material. The abstract models animated in JFLAP include deterministic and nondeterministic finite state automata, deterministic and nondeterministic pushdown automata, Turing machines, and grammars-the essential, key computational models of the theory of computing. In visualizing and animating these concepts, JFLAP effectively provides students with functionally correct mental models of these concepts with which they can interact until the concepts are understood. No static textbook presentation can match this kind of active learning environment."

7 Publications and Presentations

Rodger and her students have been publishing papers and giving presentations on JFLAP and related tools since 1992. This includes one book, twelve papers, twenty talks, three posters, two panels, one tutorial, and two demonstrations on JFLAP.

We have included three JFLAP papers in the appendix. The ITiCSE 2007 paper describes Moore and Mealy machines, pumping lemma, and batch grading. The SIGCSE 2006 paper includes a section on problem solving with JFLAP, describing thirteen ways to use JFLAP computationally that would be too tedious to do with pencil and papers. This paper also describes the Turing machine Building Blocks. The SIGCSE 2004 paper describes how JFLAP's topics fit into a formal languages course, describes many of the algorithms in JFLAP, and describes how we use JFLAP in CompSci 140 at Duke. This paper also illustrates the explosive use of JFLAP. The paper was completed in November 2003 and it states that JFLAP appears on over 2000 web pages and has over 3800 downloads since January 2003. In Section 5.4 we state that JFLAP since 2006 appears on over 20,000 web pages and has been downloaded over 35,000 times.

There have been three JFLAP workshops. In March at SIGCSE 2006, Rodger, Finley and Linz gave a 3-hour workshop on getting started with JFLAP. In June 2005 and June 2006 JFLAP faculty adopter workshops were held at Duke. Information on these workshops and the slides from the SIGCSE workshop are available on the `www.jflap.org` website under "Workshops".

Other people are citing and using JFLAP. Ganesh Gopalakrishnan states in his letter that he has just published a new textbook entitled "Computation Engineering: Applied Automata Theory and Logic". He states "in the treatment of finite automata, and especially pertaining to Turing machines, I have employed JFLAP to illustrate numerous subtle details." He has exercises in his book where he encourages the use of JFLAP to solve the exercises. Rakesh Verma at University of Houston received an NSF CCLI A&I Grant 0311407 that built on JFLAP and used Rodger as a consultant on the grant. Valerie Harvey from Robert Morris University in her letter lists four presentations/publications in which she has cited or demoed JFLAP.

8 Summary of How JFLAP Addresses NEEDS Criteria

1. Instructional Design

   - Learning Objectives

     Learning objectives are presented in this submission packet and are given on the JFLAP web site under Instructor Use.

     Coomes from William Patterson University states in her letter "When I first took over the course (CS 445), the student interest in it was dying. The course was very theoretical and abstract and it was failing to make connections with other topics and other courses. While I have maintained a substantial portion of theory and proof, there is now a healthy dose of application, experimentation, and project and presentations. JFLAP has become an integral part of the course. It has helped to make CS 445 a vital course again." Chaudhary from Millersville University states "I am certain that JFLAP improves the students' algorithmic thinking and thereby helps them better understanding the theory."

   - Interactivity

     JFLAP has been designed with interactivity in mind as students must construct models, design input, and run simulations. Students can create their own examples if they want to spend more time on a topic. JFLAP is helpful with its error messages in trying to guide the user through construction proofs. In combination with the JFLAP tutorial and described in the JFLAP papers in the appendix, there are different types of exercises to guide the user through a topic. Different forms of exercises include 1) load an NFA file and determine the language, 2) load an NFA file and determine what is wrong with it and fix it, and 3) construct an NFA for a particular language.

     Brown from USNA states in his letter that "the fact that these models can be run on inputs [with JFLAP] encourages students to question whether the machines they construct are correct, test them and debug them".

   - Cognition/conceptual change

     With JFLAP one can study a concept in multiple ways. For example, with SLR parsing. One can build a pushdown automaton for a language and run inputs on it, seeing that it is nondeterministic. For the same language, one can build an SLR(1) parse table for it, and run the same inputs, seeing how the lookaheads help in removing the nondeterminism.

     Many of the letters in the appendix state that JFLAP has improved student interest in computer science and in learning. For example, Busch from RPI states in his letter "In the past... I taught the same course without JFLAP where I noticed the student had a hard time in learning basic concepts...JFLAP helped to make a better connection between the basic theoretical concepts of the course and their applications in the practical domain". See Section 6 for more information on our current JFLAP study.

   - Content

     JFLAP's content covers a large number of topics in a formal languages course including the main computational models (finite automata, pushdown automata and several versions of Turing machines), and all the related grammars (regular, context-free and unrestricted). In addition, JFLAP covers many of the connections between them such as the proof that the class of context-free grammars are equivalent to the class of nondeterministic pushdown automata and vice versa. Furthermore, JFLAP covers the applications of L-systems and SLR parsing.

   - Multimedia Use

     The graphical editor has been designed with simplicity in mind, few colors are used. When simulating an NFA, red is used to indicate a configuration is stuck and green (and also a picture of a final state) is used to indicate acceptance. States can be annotated so the user can assign a meaning to it. Some windows have multiple panes that can be resized.

   - Instructional use/adaptability

JFLAP's web tutorial explains how to use the graphical editor, and the algorithms for the proofs. Also on JFLAP web page under *Instructor Use* we describe how instructors can use JFLAP. The JFLAP 2006 SIGCSE paper in the appendix explains many computational uses of JFLAP.

Section 5 on the usage of JFLAP shows the wide use of JFLAP by others in five different courses and even high schools. JFLAP is freely available and this section also shows how users have modified JFLAP for their needs.

Instructors have found JFLAP easy to use during lecture and many state in their letters that JFLAP has enhanced their teaching. Brown from The College of Wooster states "[JFLAP] has substantially enhanced the effectiveness of my teaching in the classes where it applies." Sloan from University of Illinois at Chicago states that "[JFLAP] has helped me to win a major teaching award at my university".

2. Software Design

   - Engagement

     JFLAP provides immediate feedback. Students want feedback and will use it whether it is required or not. Brown from USNA states "From that point on [JFLAP's] use is optional, but most students keep using it, as evidenced by the JFLAP screen captures that they turn in as part of their later homework assignments." The multiple run window encourages students to do thorough testing of their constructions.

   - Learner interface and navigation

     The JFLAP editor for all types of automata is very similar, making it easy for a user to learn the next type of automaton. The editor for grammars is also similar for all the types of grammars. For the complicated algorithms such as SLR parsing, grammar transformation to CNF, or NFA to DFA, JFLAP has multiple tabs, making it easy for the user to backtrack. Many of the letters included statements that students find JFLAP easy to use.

   - Technical reliability

     Elenbogen at the University of Michigan Dearborn states ""I have been using JFLAP for over 10 years and am writing concerning its benefits to teaching my classes. ... Its visual interface is so easy to use that it is wonderful for demonstrations in class. .. The output of JFLAP has improved over the years but it was always of such high quality that, I proudly posted the examples I did in class to the class website for examples. The interface is so well designed that students use it for homework problems even when it is not required."

3. Engineering Content

   - Accuracy of Content

     JFLAP covers general definitions of models to fit more closely with multiple formal languages textbooks.

   - Appropriateness of content

     JFLAP covers the majority of material from a computer science formal languages course, and others have found it useful in four other types of computer science courses.

9 Appendices

Following are five appendices that include a list of JFLAP publications and presentations, 3 JFLAP publications, the preface from the JFLAP book, some partial results from our JFLAP study in 2005-2006, and 68 letters from 14 countries in support of JFLAP.

Resubmission Documentation of JFLAP

We previously submitted the software JFLAP to the NEEDS 2006 competition. In this document we describe how JFLAP has changed and how the changes address the suggestions given by the judges last year.

We edited the body of the text to reflect the new changes, but we also describe these new changes here.

1 The changes to JFLAP

This year we are submitting the newest version of the JFLAP software and also submitting with it a JFLAP web tutorial that explains how to use JFLAP.

The additions of new topics to the JFLAP software were not requested by the judges, but rather we are continually adding new constructs and algorithms to JFLAP so that it will include a broader number of topics.

Additions to JFLAP include Moore and Mealy machines, Turing machine Building Blocks, Batch grading, regular pumping lemma proof, context-free pumping lemma proof and many small items such as adding preferences to select the type of empty string to appear (lambda or epsilon). These new structures and examples are explained in the 2006 and 2007 papers included in the appendix.

We have also created an extensive JFLAP Web tutorial that guides the user through the usage of topics in JFLAP, with example JFLAP files that can be loaded and used to follow along. This addition to the JFLAP web site is in response to the judges for JFLAP to provide more guidance in its use.

2 Addressing concerns of judges

1. The main concern of the judges was the lack of on-line help to guide the user.

   The newest change to the JFLAP web site is an online tutorial that covers the topics of JFLAP, guiding users through the creation of automata, grammars and other structures, and the running and testing of these creations. The tutorial guides the users through the many algorithms for construction proofs. Tutorial topics use examples with JFLAP files that can be loaded into JFLAP or created from scratch by the user. The user can load the example file and follow along in the tutorial. The tutorial includes many screen shots to show the user how JFLAP should look at each stage.

   The files for the tutorial can be downloaded for saving and loading into JFLAP as the user reads the tutorial, or one can download a .zip file containing all the example files.

2. One concern was the syntax of the input descriptors. These should now be explained in the web tutorial.

3. One concern is that the size of the working areas are too small. One can resize the JFLAP window to full screen and move the horizontal and vertical bars to expand one portion of the JFLAP window. This is explained in the tutorial.

4. One concern was the limit on the complexity of the models that can be constructed. JFLAP is an instructional tool to learn concepts and is not a tool for solving real (large) problems. With JFLAP one can learn how SLR parsing works with an example that has about 5 rules and 8 states. One would not want to use JFLAP to visualize an SLR parser for the programming language Java. That is not its intent.

5. One concern was they could not figure out how to create sub-machines. Building blocks can be created for Turing machines. There are two tutorial pages on this topic. Under "Turing machines," then "Construct and Run, " the tutorial guides you through constructing a Turing machine, saving it as a building block, and then using it to solve another problem. Under "Turing Machines," and then "Building Blocks," it describes building blocks in more detail. Also on the JFLAP web page under "Sample Files" you can get a library of Turing machine Building Blocks and some sample files.

6. The judges suggest adding a library of sample models. We have done that in our tutorial by including example files with all the concepts.

7. The judges suggest to improve on the use of icons. Many icons identify their name with the mouse over them. We have also identified many of these icons in the tutorial as they are needed.

8. The judges suggest not to give the end result but rather to animate the process. JFLAP goes further than that by having the user construct the result. Although JFLAP provides a "show" or "show step" button to show the final result on many construction proofs, the user can also construct on their own the final result. The tutorials guide the user in the step-by-step constructions.

9. The judges suggest to improve the efficiency of the software response. In some cases we do have JFLAP come up and tell you if it is taking too long. See the tutorial page under "Grammar," then "Convert CFG to PDA(LR)." At the end of the tutorial it shows how to use multiple run with strings that take too long to parse. A message appears and you have to cancel the parse.

List of Appendices

1. List of JFLAP Publications and Presentations
2. Three JFLAP publications
3. Preface from JFLAP book
4. Some Evaluation Results from JFLAP study 2005-2006
5. 68 Letters in Support of JFLAP

Appendix 1 - List of JFLAP Publications and Presentations

This is a list of JFLAP publications and presentations given by Rodger and her students. Many of these publications or slides of talks appear on the web site `www.jflap.org` under "JFLAP papers" or "JFLAP talks".

Publications

Susan H. Rodger, Jinghui Lim and Stephen Reading, Increasing Interaction and Support in the Formal Languages and Automata Theory Course, The 12th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2007), Dundee, Scotland, June 25, p. 58-62, 2007.

Susan H. Rodger and Thomas W. Finley, JFLAP - An Interactive Formal Languages and Automata Package, ISBN 0763738344, Jones and Bartlett Publishers, 2006.

Susan H. Rodger, Bart Bressler, Thomas Finley, and Stephen Reading, Turning Automata Theory into a Hands-on Course, Thirty-seventh SIGCSE Technical Symposium on Computer Science Education, p. 379-383, 2006.

Ryan Cavalcante, Thomas Finley and Susan H. Rodger, "A Visual and Interactive Automata Theory Course with JFLAP 4.0," *Thirty-fifth SIGCSE Technical Symposium on Computer Science Education*, p. 140-144, 2004.

S. H. Rodger, Using Hands-on Visualizations to Teach Computer Science from Beginning Courses to Advanced Courses, *Second Program Visualization Workshop*, Hornstrup Centert, Denmark, p. 103-112, June 2002.

T. Hung and S. H. Rodger, "Increasing Visualization and Interaction in the Automata Theory Course," *Thirty-first SIGCSE Technical Symposium on Computer Science Education*, p. 6-10, 2000.

E. Gramond and S. H. Rodger, "Using JFLAP to Interact with Theorems in Automata Theory," *Thirtieth SIGCSE Technical Symposium on Computer Science Education*, p. 336-340, 1999.

A. O. Bilska, K. H. Leider, M. Procopiuc, O. Procopiuc, S. H. Rodger, J. R. Salemme and E. Tsang, "A Collection of Tools for Making Automata Theory and Formal Languages Come Alive," *Twenty-eighth SIGCSE Technical Symposium on Computer Science Education*, p. 15-19, 1997.

M. Procopiuc, O. Procopiuc, and S. Rodger, "Visualization and Interaction in the Computer Science Formal Languages Course with JFLAP," *1996 Frontiers in Education Conference*, Salt Lake City, Utah, p. 121-125, 1996.

S. A. Blythe, M. C. James, and S. H. Rodger, "LLparse and LRparse: Visual and Interactive Tools for Parsing," *Twenty-fifth SIGCSE Technical Symposium on Computer Science Education*, p. 208-212, March 1994.

D. Caugherty and S. H. Rodger, "NPDA: A Tool for Visualizing and Simulating Nondeterministic Pushdown Automata," in *Computational Support for Discrete Mathematics, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 15, N. Dean and G. E. Shannon (ed.), p. 365-377, 1994.

E. Luce and S. H. Rodger, "A Visual Programming Environment for Turing Machines," *Proceedings of the 1993 IEEE Symposium on Visual Languages*, p. 231-236, August 1993.

M. LoSacco and S. H. Rodger, "FLAP: A Tool for Drawing and Simulating Automata," *ED-MEDIA 93, World Conference on Educational Multimedia and Hypermedia,* p. 310-317, June 1993.

Presentations that include JFLAP

"Computer Science Concepts Come Alive," Computer Science Department Colloquium, The Citadel, Feb. 20, 2007.

"Teaching Strategies and Learning Styles," CRAW Workshop - Managing the Academic Career for Faculty Women at Undergraduate Computer Science and Engineering Institutions, Cincinnati, OH, March 7, 2007.

"An Interactive Approach to Formal Languages and Automata with JFLAP," NSF Showcase at Thirty-eighth SIGCSE Technical Symposium on Computer Science Education, Cincinnati, Ohio, March 9, 2007.

"Turning Automata Theory into a Hands-on Course," Thirty-seventh SIGCSE Technical Symposium on Computer Science Education, Houston, Texas, March 3, 2006.

Workshop: "A Hands on Approach to Formal Languages and Automata with JFLAP," Thirty-seventh SIGCSE Technical Symposium on Computer Science Education, Houston, Texas, March 4, 2006 (with P. Linz and T. Finley).

Panel: "Automata Theory - Its Relevance to Computer Science Students and Course Contents," Panelist, *Thirty-seventh SIGCSE Technical Symposium on Computer Science Education*, Houston, Texas, March 2, 2006.

Panel: "Animation and Visualization in the Curriculum: Opportunities, Challenges, and Successes," Panelist, *Thirty-seventh SIGCSE Technical Symposium on Computer Science Education*, Houston, Texas, March 3, 2006.

Demonstration: "Learning Automata and Formal Languages Interactively with JFLAP," The Eleventh Annual Conference on Innovation and Technology in Computer Science Education, University of Bologna, Italy, June 28, 2006.

"A Visual and Interactive Automata Theory Course with JFLAP 4.0," *Thirty-fifth SIGCSE Technical Symposium on Computer Science Education*, Norfolk, Virginia, March 4, 2004.

"An Interactive and Visual Approach to Learning Computer Science," Department of Computer Science, University of Houston, Houston, Texas, November 30, 2004.

"Learning Computer Science Concepts via Interactive Visualizations," Department of Computer Science, Johns Hopkins University, Baltimore, MD, July 24, 2003.

"Using Hands-on Visualizations to Teach Computer Science from Beginning Courses to Advanced Courses", Second Program Visualization Workshop, Hornstrup Centret, Denmark, June 28, 2002.

"Using JFLAP for Visualization and Interaction in the Automata Theory Course," Software Visualization Workshop, Dagstuhl, Germany, May 25, 2001.

"Increasing Visualization and Interaction in the Automata Theory Course," *Thirty-first SIGCSE Technical Symposium on Computer Science Education*, Austin, TX, March 9, 2000.

"Using JFLAP to Interact with Theorems in Automata Theory," *Thirtieth SIGCSE Technical Symposium on Computer Science Education*, New Orleans, LA, March 27, 1999.

"Animation, Visualization and Interaction with Computer Science Concepts," Department of Mathematics and Computer Science, Wake Forest University, Winston-Salem, NC, December 1, 1998.

Poster: "JFLAP: An Aid to Studying Theorems in Automata Theory," Integrating Technology into Computer Science Education, Dublin, Ireland, August 20, 1998 (with E. Gramond).

"A Collection of Tools for Making Automata Theory and Formal Languages Come Alive," *Twenty-eighth SIGCSE Technical Symposium on Computer Science Education*, San Jose, CA, February 27, 1997.

"Visualization and Interaction in the Computer Science Formal Languages Course with JFLAP," *1996 Frontiers in Education Conference*, Salt Lake City, Utah, November 5, 1996.

"Visual Demonstrations of Automata and Parsing," Interactive and Visual Tools Workshop, Duke University, March 30, 1996.

"Integrating Hands-On Work into the Formal Languages Course via Tools and Programming," *Workshop on Implementing Automata*, London, Ontario, August 30, 1996.

Demo of JFLAP, *Workshop on Implementing Automata*, London, Ontario, August 30, 1996.

Poster: "Interactive Tools for Teaching and Learning the Foundations of Computer Science," American Society for Engineering Education (ASEE) Conference, Washington, DC, June 24, 1996.

Tutorial: "Using Visual Demonstrations," SIGCSE Technical Symposium on Computer Science Education, Nashville, TN, March 2, 1995 (with S. Grissom, R. Ross, D. Schweitzer, T. Naps, and D. Hunkins).

"LLparse and LRparse: Visual and Interactive Tools for Parsing," SIGCSE Technical Symposium on Computer Science Education, Phoenix, Arizona, March 11, 1994.

Poster and demos for "Visual and Interactive Tools for Teaching Computer Science," SIGCSE 94 Technical Symposium, Phoenix, Arizona, March 10, 1994.

"A Visual Programming Environment for Turing Machines," IEEE Symposium on Visual Languages 1993, Bergen, Norway, August 26, 1993.

"FLAP: A Tool for Drawing and Simulating Automata," ED-MEDIA 93, World Conference on Educational Multimedia and Hypermedia, Orlando, FL, June 24, 1993.

"Computers in Teaching the Foundations of Computer Science," Computers in Science Education and Training Workshop, Acadia University, Wolfville, Nova Scotia, June 15, 1992.

Appendix 2 - Three JFLAP publications

We include the following three JFLAP publications.

- Susan H. Rodger, Jinghui Lim and Stephen Reading, Increasing Interaction and Support in the Formal Languages and Automata Theory Course, The 12th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2007), Dundee, Scotland, June 25, p. 58-62, 2007.
- Susan H. Rodger, Bart Bressler, Thomas Finley, and Stephen Reading, Turning Automata Theory into a Hands-on Course, Thirty-seventh SIGCSE Technical Symposium on Computer Science Education, p. 379-383, 2006.
- Ryan Cavalcante, Thomas Finley and Susan H. Rodger, "A Visual and Interactive Automata Theory Course with JFLAP 4.0," *Thirty-fifth SIGCSE Technical Symposium on Computer Science Education*, p. 140-144, 2004.

Increasing Interaction and Support in the Formal Languages and Automata Theory Course*

[Extended Abstract]

Susan H. Rodger
Computer Science Department
Duke University
Durham, NC 27708
rodger@cs.duke.edu

Jinghui Lim
Computer Science Department
Duke University
Durham, NC 27708

Stephen Reading
Computer Science Department
Duke University
Durham, NC 27708

ABSTRACT

The introduction of educational software such as JFLAP into the course Formal Languages and Automata (FLA) has created a learning environment with automatic feedback on theoretical topics. In this paper we show how we further increase the interaction in the FLA course with the expansion of additional theoretical topics in JFLAP, and how we have added grading support into JFLAP for instructors.

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, Moore machine, Mealy machine

1. INTRODUCTION

The teaching of formal languages and automata (FLA) is moving from a pencil and paper environment to an interactive learning environment with the introduction of educational software to experiment with the many concepts in this course. This approach has both advantages and disadvantages. The advantages allow students to experiment with concepts that would be difficult and tedious to do on paper, and to receive immediate feedback on problem solving. The disadvantages include the difficulty for the instructor choosing to move this route with a large number and variety of tools being developed, the added burden of grading homeworks in a format other than paper, and the difficulty on students learning several tools with different formats.

A large variety of tools have been developed to experiment with one or more concepts from the FLA course. We list a few of those tools here, there are many more[5]. Turings World[1] is for experimenting extensively with Turing machines. Forlan[10] is a toolset for experimenting with regular languages. Language Emulator[12] is a toolkit for regular languages and includes Moore and Mealy machines. The tool jFAST[13] allows experimentation with finite automata, pushdown automata and Turing machines. Grinder[4] focuses on finite state automata. RegEx[2] allows experimentation with regular expressions.

A single tool that covers a large number of topics takes enormous development time, but can be easier on instructors and students in moving from one concept to another. There are three efforts to create a large collection of FLA concepts in one tool. Taylor[11] explores many types of machines including Turing machines, finite automata, pushdown automata, and linear bounded automata using the software *Deus Ex Machina*, but does not include support for other topics such as grammars or transformations between grammars and automata. Ross[3] is in the process of developing an extensive hypertextbook called Webworks that will include many topics in automata theory, and incorporates text, sound, pictures, illustrations, slide shows, video clips and active learning models. This is a huge development effort in the works for several years now. JFLAP[8, 9], described in more detail in Section 2, is a software tool under development for over fifteen years to incorporate many topics in automata theory in one tool including automata, grammars and experimentation with proofs.

In this paper we show how we further increase the interaction in the FLA course with the expansion of additional theoretical topics in JFLAP, and how we encourage the creation of JFLAP files for grading by providing grading support for instructors within JFLAP. The main additional theoretical topics in JFLAP are described in Section 3 and include Moore and Mealy machines, and games for learning about regular and context-free pumping lemmas. The batch grading support is described in Section 4, and includes an

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

ITiCSE'07, June 23–27, 2007, Dundee, Scotland, United Kingdom.

interface for loading and grading a large number of student JFLAP files.

2. OVERVIEW OF JFLAP

JFLAP (Java Formal Languages and Automata Package) is instructional software for experimenting with automata and grammars, but goes further in allowing one to experiment with proofs and applications related to these topics. JFLAP's main feature is the ability to experiment with theoretical machines and grammars. With JFLAP one can build and run user-defined input on finite automata, pushdown automata, multi-tape Turing machines, regular grammars, context-free grammars (CFG), unrestricted grammars, and L-systems. After constructing the automaton or grammar, one can trace through a single input string or receive automatic feedback on multiple inputs.

JFLAP's second feature is the ability to construct in steps the proof of the transformation of one form to another form. For example, after constructing a nondeterministic finite automata (NFA), one can step through its conversion to a deterministic finite automata (DFA), then to a minimal state DFA, and then to regular grammar. As another example, one can build a CFG and construct in steps the equivalent nondeterministic PDA.

JFLAP's third feature is the experimentation with applications of theoretical material. One example is experimenting with parsing by constructing the SLR(1) parse table in steps, and then stepping through the parsing of input strings and the construction of the equivalent parse tree. The construction of the parse table includes building a special DFA that models the parsing process, thus seeing a use for a DFA. Another application is building an L-system grammar of a plant, and rendering it to watch a simulation of the plant growing.

3. NEW TOPICS IN JFLAP 6.0

In this section we describe new topics we have integrated into JFLAP. All these topics allow students to experiment with them and receive immediate feedback.

One new topic is transducers. A transducer is a finite automaton that generates output. JFLAP now provides experimentation with two types of transducers: Mealy machines for generating output based on transitions and Moore machines for generating output based on states.

Another new topic is the pumping lemma. We provide pumping lemma games for both the regular pumping lemma and the context-free pumping lemma. Given a language, the student plays against the computer to help categorize the language. For example, if the language is not regular, the game will prove it is not regular, and if the language is regular, a partition is found to show the pumping lemma holds for that language.

3.1 Mealy Machines

A Mealy machine is similar to a finite state automaton, but it does not have final states and it can produce output through its transitions. Adding Mealy machines to JFLAP expands on applications of automata as Mealy machines can model problems in many disciplines. We first provide a definition of a Mealy machine and then give an example of a Mealy machine implemented in JFLAP.

Definition: A Mealy machine is defined as a 6-tuple, $(Q,\Sigma,\Gamma,\delta,\omega, q_0)$, where

- Q is finite set of states
- Σ is input alphabet
- Γ is output alphabet
- q_0 is initial state
- δ is the transition function, δ:Q×Σ→Q
- ω is the output function, ω:Q×Σ→ Γ

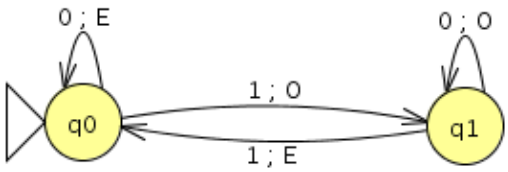


Figure 1: A Mealy Machine built with JFLAP

Input	Result
10	OO
110	OEE
101	OOE
1000	OOOO
1010	OOEE
1111	OEOE

Figure 2: Mealy Output from Several Inputs

Using JFLAP we build a Mealy machine to output information describing whether there is an odd number of 1's in a string. Since a Mealy machine doesn't have a final state, we will output the current status of a binary string based on how many digits have been viewed. Figure 1 shows the corresponding Mealy machine. There are two states q_0 and q_1, with q_0 indicated as the start state. The meaning of state q_0 is an even number of 1's have been seen and the meaning of state q_1 is an odd number of 1's have been seen. The transition $1; O$ from q_0 to q_1 means if a 1 is processed, output the letter O meaning an odd number of 1's have been processed, and the transition $1; E$ from q_1 to q_0 means if a 1 is seen output the letter E meaning there is an even number of 1's. The output for several inputs run in JFLAP is shown in Figure 2.

3.2 Moore Machines

A Moore machine is a transducer similar to a Mealy machine, but generates output based on states instead of transitions. Like the Mealy machine, the definition of a Moore machine is a 6-tuple, but with ω defined as $\omega : Q \rightarrow \Gamma$.

We build an example of a Moore machine in JFLAP shown in Figure 3 that divides binary numbers in half with truncation. One way to divide a binary number in half is to remove its last digit. However, Moore machines process the input string from left to right and do not have final states. Thus our algorithm keeps track of two symbols at a time (the previous and current symbol) and outputs the previous symbol.

This example also illustrates a feature of JFLAP that allows one to change the name of states to more meaningful names. The states in this machine have been renamed to be the previous and current symbol seen. Here the start

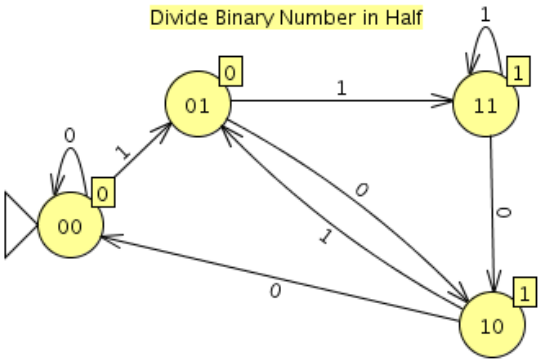


Figure 3: Moore Machine - Dividing Binary Numbers

Input	Result
10	001
110	0011
101	0010
1000	00100
1010	00101
1111	00111

Run Inputs | Clear | Enter Epsilon | View Trace

Figure 4: Moore Machine Output Runs in JFLAP

state (indicated by the triangular arrow) has been renamed from the default q_0 to 00. Assuming 0's preceeding any binary number are ignored, this start state represents the two previous and current nonexistant symbols as both 0 before processing any input. The output associated with each state is shown as a box attached to the top-right side of the state. The output for each state in this example is the previous symbol processed, which also happens to be the leftmost symbol of the state name. Thus, the output for state 00 is 0 and the output for state 10 is 1.

Let's process the string 101. Starting in state 00 automatically outputs the output for that state, 0. Processing the 1 moves to state 01 and outputs 0. Processing the 0 moves to state 10 and outputs 1. Processing the final 1 moves to state 01 and outputs 0. Thus the resulting output is 0010, which is equivalent to 10. Results of several inputs for this problem run in JFLAP are shown in Figure 4.

3.3 Regular Pumping Lemma Game

The pumping lemma is a more abstract concept than an automaton. In the FLA course, students experiment with a regular language in the form of an automaton, which is a concrete item that can be built and traced through with input. Then they are asked to identify a language, whether or not it is regular. If they can build a DFA for the language, they can easily answer the question. If they cannot, then they must consider proving the language is not regular. One such procedure is to use the pumping lemma. In this section we describe how we have created a pumping lemma game students can play to learn this concept. We modeled our pumping lemma game after the descriptions in [6].

First, we give the definition of the pumping lemma for regular languages.

Pumping Lemma: Let L be an infinite regular language. Then there is a constant $m > 0$ such that for any $w \in L$, with $|w| \geq m$, w can be partitioned into three parts $w = xyz$ with

$$|xy| \leq m$$
$$|y| \geq 1$$
$$xy^i z \in L \quad \text{for all } i \geq 0$$

The pumping lemma can be used to prove a language is not regular.

The game approach with JFLAP involves two players, the student and JFLAP. JFLAP's goal is to prove the language is not regular. The student's goal is to make it as hard as possible. The game proceeds as follows.

First the student selects a language for the game (one of nine choices) and selects an integer for m, the constant in the pumping lemma. JFLAP then picks a string in the language such that the string is of length greater than or equal to m. The student then has to partition the string into the three parts x, y, and z such that $xy^i z$ is in the language for all $i \geq 0$.

Here is an example shown in Figure 5 for the language $\{a^n b^n | n \geq 0\}$. In step 1, the user selected 7. In step 2 JFLAP shows the string $a^7 b^7$ and asks the user to select a decomposition. Step 3 starts with the slider bars all the way to the left. The user slid the first bar over two spaces to indicate the selection of $x = aa$, the second bar over an additional two spaces to indicate the selection of $y = aa$ and z becomes the rest of the string, $z = a^3 b^7$. JFLAP provides helpful messages to guide the user. When the user selects x, the second slider bar moves along also, and if the user slides the second bar back to the left, an error message is shown. If the user slides the bar too far to the right, making the condition $|xy| \leq m$ not true, JFLAP displays a message showing this condition has been violated and the user must make another selection. Once the user has a correct partition, in step 4, JFLAP shows a choice of i that gives a contradiction, 0 in this case and the generated string that is not in the language. Step 5 provides further explanation by providing an animation of building the pumped string that is not in the language.

One has a choice of nine languages to choose from, one of which is regular. In this case, the student can find a partition that allows the string to be pumped.

3.4 Context-Free Pumping Lemma

The pumping lemma game for context-free languages (CFL) is more complicated as there may be many cases to consider. We first give the definition of the pumping lemma for CFL's.

Pumping Lemma for CFL's Let L be an infinite context-free language. Then there is a constant $m > 0$, such that for any $w \in L$, with $|w| \geq m$, w can be partitioned into five parts $w = uvxyz$ with

$$|vxy| \leq m, \text{ (limit on size of substring)}$$
$$|vy| \geq 1, (v \text{ and } y \text{ not both empty})$$
$$\text{For all } i \geq 0, uv^i xy^i z \in \text{L}$$

In this pumping lemma game, the user starts by selecting a language from a list of nine possible languages. The window is partitioned into two parts. The left side looks similar

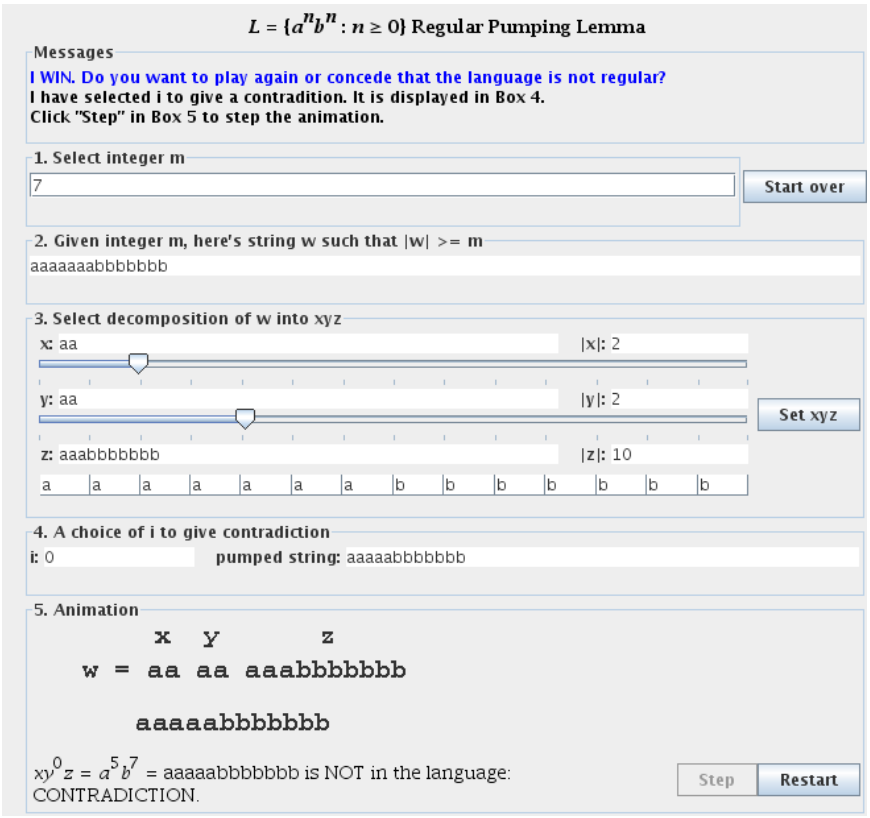


Figure 5: Regular Pumping Lemma Game

to the window for the regular pumping lemma. The right side of the window is for categorizing all the cases describing the choices for v and y (explained below), and starts as empty. The user plays the game by selecting the choice of m in step 1. JFLAP shows a string in step 2 that is in the language and whose length is greater than or equal to m. In step three the user moves the slider bars to partition the string into five parts u, v, x, y and z. In step 4 JFLAP finds an i to give a contradiction and step 5 shows an animation of building the string that provides the contradiction. At this point, the user can add the case to the right side of the window, and then start over to find another case.

We now explain the cases for a language. Consider the language $a^n b^n c^n$. The condition $|vxy| \leq m$ limits the size of this substring, but doesn't restrict where it can start. (Note that in the regular pumping lemma, the substring xy always started on the left end of the string.) To prove a language is not CFL, every possible partition must be considered and proven that it fails. Without a starting restriction for vxy we must consider all possible starting positions. We have done this by focusing on the contents for v and y and describing what type of letter they contain. For example, v has $a's$ then there are three cases for y: 1) y is all a's, 2) y is a's followed by b's and 3) y is b's. Note that y cannot have c's or $|vxy| > m$. In this example, there are 11 cases that the student must find. The user can find as many cases as they want, and at some point they can select "Find" to show all the cases.

3.5 Other New Features of JFLAP 6.0

We would like to mention a few other features that have been added to JFLAP. Annotations (called notes) can now be added in the editor pane at any location. In Figure 3 we added a note with a title to the Moore machine. Copying portions of a drawing is now easier by allowing one to select a bounding box area to copy.

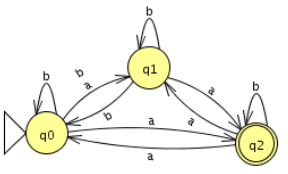
Multi-run input has been added to Grammars. In previous versions of JFLAP, only one input could be tested at a time on a grammar, with the result being either a derivation or a parse tree. Now one can run several inputs at the same time, and then select one at a time from the results to view the derivation or parse tree.

One change to make JFLAP fit with more textbooks has been a new preference to select the representation of the empty string, either *lambda* or *epsilon*. JFLAP stores your preference in a file and displays the empty string from any file with your preference.

4. BATCH GRADING SUPPORT

In previous versions of JFLAP, running multiple inputs at once for an automaton was added to aid in grading and to provide students with faster testing. In particular, one machine would be loaded and run on several inputs. The next machines loaded would use the same inputs, so the grader would not have to reenter them. However, grading could still take a long time for a large course as each machine has to be loaded and run.

In JFLAP 6.0, batch grading support has been added that



File	Input	Result
fa1.jff	aabbaabb	Reject(Accept)
fa1.jff	aaaa	Reject(Accept)
fa1.jff	aaab	Reject(Accept)
fa1.jff	aaabaaab	Reject(Accept)
fa2.jff	aabbaabb	Reject(Accept)
fa2.jff	aaaa	Reject(Accept)
fa2.jff	aaab	Reject(Accept)
fa2.jff	aaabaaab	Reject(Accept)
fa5.jff	aabbaabb	Accept
fa5.jff	aaaa	Accept
fa5.jff	aaab	Accept
fa5.jff	aaabaaab	Accept

Figure 6: Batch Grading Three Automata

allows one to load and produce output for multiple machines at once. Before using, the grader should place all the files in one directory and create an input text file with accept or reject status for each string. From the main JFLAP menu one selects *Batch* and then selects the automata files to run followed by selecting the input file. A window then appears with a picture of one of the automaton in the left side of the window, and the right side of the window lists the name of the files and the input strings to test. After selecting *Run Inputs* the results are displayed. For example, Figure 6 shows a run with three automata named fa1.jff, fa2.jff and fa5.jff. The machine *fa5.jff* has been selected and highlighted, and its graphic picture is shown on the left side of the window. On the right side are shown the results of running four input strings on each of these three files. The results for file *fa1.jff* are shown first, then the results for file *fa2.jff* and then the results for *fa5.jff*. The file *fa5.jff* accepts all four strings, and the other two files reject all four strings.

There are several options with batch grading. By selecting a different automaton on the right, the picture on the left changes to correspond to that automaton. One can add an additional automaton, and it is added with all the current inputs. One can add an input string, adding a line for each automaton. Files can be selected and edited to fix a bug. A trace for a specific automaton and string can be viewed. Most important, the results can be saved to files. Selecting *Save Results* writes each input string and its result for each file to a file of the same name with a ".txt" extension added. These files can be returned to students.

Batch grading is available for finite automata, pushdown automata, Turing machines and grammars.

5. USE OF JFLAP

JFLAP is used world wide in over 160 countries in formal languages courses, compiler courses, discrete math courses and artificial intelligence courses. The JFLAP web site [7] tracks usage of JFLAP. This web site allows users to download JFLAP for free and has had over 35,000 downloads since 2003. This site provides many resources including sample files, tutorial, slides, and papers.

JFLAP is used at Duke University in the course CompSci 140 and informal feedback from students over the past ten years has been very positive. JFLAP is currently undergoing a formal two-year study from 2005-2007 with twelve universities participating to evaluate JFLAP's effectiveness in learning automata theory topics.

6. CONCLUSION

We continue to develop JFLAP into an extensive tool covering many of the topics in a formal languages course including experimentation with automata, grammars, and proofs involving these concepts. We have recently added the experimentation of transducers to JFLAP, Moore and Mealy machines, and a new approach of games for learning the pumping lemmas. We also address concerns of grading JFLAP files by providing an interface for grading a large number of JFLAP files.

7. ACKNOWLEDGMENTS

Thanks to Peter Linz for suggestions on improving the pumping lemma interface.

8. REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

[5] V. J. Harvey and S. H. Rodger. Editorial for the special issue on software support for teaching discrete mathematics. *Journal on Educational Resources in Computing*, 5(2):1–16, June 2005.

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

[7] S. H. Rodger. Jflap web site, 2006. www.jflap.org.

[8] S. H. Rodger, B. Bressler, T. Finley, and S. Reading. Turning automata theory into a hands-on course. In *Thirty-seventh SIGCSE Technical Symposium on Computer Science Education*, pages 379–383. SIGCSE, March 2006.

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

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

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

[12] 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.

[13] 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.

Turning Automata Theory into a Hands-on Course *

Susan H. Rodger
Computer Science
Duke University
Durham, NC 27708
rodger@cs.duke.edu

Bart Bressler
Computer Science
Duke University
Durham, NC 27708

Thomas Finley
Computer Science
Cornell University
Ithaca, NY 14853

Stephen Reading
Computer Science
Duke University
Durham, NC 27708

ABSTRACT

We present a hands-on approach to problem solving in the formal languages and automata theory course. Using the tool JFLAP, students can solve a wide range of problems that are tedious to solve using pencil and paper. In combination with the more traditional theory problems, students study a wider-range of problems on a topic. Thus, students explore the formal languages and automata concepts computationally and visually with JFLAP, and theoretically without JFLAP. In addition, we present a new feature in JFLAP, Turing machine building blocks. One can now build complex Turing machines by using other Turing machines as components or building blocks.

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, pushdown automata, Turing machine, grammar, SLR parsing, LL parsing, L-system

1. INTRODUCTION

Traditionally, the formal languages and automata (FLA) courses have assigned pencil and paper homework exercises of two types: proofs and construction exercises. The second of these types of problems are limited to small examples. Even on a moderate-size example of constructing an automaton with eight states, students are unlikely to do much testing as it is tedious to trace by hand. Grading such problems is similarly slow and error prone.

We describe a hands-on approach to the FLA course that allows students to explore many of the FLA concepts computationally and visually using the tool JFLAP. We are *not* advocating to remove the proof type of exercises from the course, but rather to supplement them with hands-on explorations of related topics. For example, consider the problem of proving that if a language L is regular, then so is the language L^R (the language with all strings from L reversed). This is a common proof-type problem given in this course. For some students, before proving this, it might be helpful to visualize an example first. They start with some regular language L, build a deterministic finite automaton (DFA) for it, and then convert this DFA into a DFA for L^R. They must create test data for both DFA to convince themselves that the DFA are correct. This approach relates the FLA course more in line with the majority of their computer science courses which are hands on and involve constructing, debugging and testing.

Others have taken similar hands-on approaches to the FLA course, but focus on a smaller number of topics. Turing's World[1] allows one to create and experiment with Turing machines and automata. The focus is on Turing machines and submachines. Taylor[8] uses the software *Deus Ex Machina* letting users experiment with Turing machines, finite automata, pushdown automata, and several other types of automata. Forlan[7] is a toolset used in conjunction with Standard ML for creating and experimenting with finite automata, regular expressions and grammars. Language Emulator[9] is a toolkit for a number of forms of regular languages including Moore and Mealy machines, and many types of translations between the forms. Grinder[4] has developed the FSA Simulator for experimenting with finite state automata. It is part of Webworks[3], an extensive hypertextbook under development that will cover many topics in automata theory. It incorporates text, sound, pictures, illustrations, slide shows, video clips and active learning models.

In this paper we present an overview of JFLAP and then give several examples of how it can be used computationally and visually to explore FLA concepts in depth. We then present new features of JFLAP including the ability to build more interesting Turing machines with building blocks. Turing machines built can be named and reused as a component in another Turing machine. We conclude with an evaluation of JFLAP's use around the world and a description of future work.

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

SIGCSE'06 March 1–5, 2006, Houston, Texas, USA.

2. AN OVERVIEW OF JFLAP

JFLAP[5, 6, 2] is an instructional tool for creating and experimenting with several types of nondeterministic automata, grammars, regular expressions, L-systems, and experimenting with the conversion from one structure to another. With JFLAP one can build a finite automaton (FA), a pushdown automaton (PDA), or a multi-tape Turing machine (TM) and observe its simulation on several inputs. One can enter a regular grammar, a context-free grammar (CFG), or an unrestricted grammar and observe the brute-force parsing of strings in this grammar with the result shown either as a derivation or a parse tree.

JFLAP allows the conversion from one form to another. One can convert an NFA to a DFA to a minimal state DFA, convert between NFA and regular grammars, or convert between NFA and regular expressions. One can convert a nondeterministic PDA (NPDA) to a CFG or a CFG to an NPDA. One can convert a CFG to Chomsky Normal Form, along the way removing λ-productions, unit productions and useless productions. One can convert a CFG to either an LL(1) or SLR(1) parse table and then parse strings in the language. Finally, one can create an L-system, a different type of grammar that can be used for modeling the growth of plants and fractals.

3. PROBLEM SOLVING WITH JFLAP

The previous section described the construction and testing of automata and grammars in JFLAP, and the conversion from one form to another. That in itself allows for users to build and test automata more easily than can be created using pencil and paper.

We now describe several other types of problem solving with JFLAP that are tedious to do with pencil and paper.

3.1 Comparison of Finite Automata

Given two different FA, determine if they are equivalent and if not, then show that they do not accept the same language. A student can either be given the two FA in files, or can build them with JFLAP. The student must determine a good set of test data and then run simulations on the input strings. The multiple run window allows for the testing of multiple inputs simultaneously. Alternatively, they can minimize the two FA and compare their results. Finally, JFLAP's *Compare Equivalence* will announce if the two FA are equivalent. If the two FA are not equivalent, a student needs to determine one string that is accepted in one FA and not in the other.

3.2 Comparison of Regular Expressions

Given two regular expressions, determine if they are equivalent or not. In JFLAP one cannot test strings for a regular expression. However, one can convert a regular expression to an equivalent FA and then run a series of test strings similar to the comparison of two FA.

3.3 Working Backwards - DFA to NFA

This problem shows the understanding of how an NFA is transformed into a DFA, but works backwards. The students are given a DFA from JFLAP that was transformed from an NFA. The DFA has each state labeled with the numbers of the states from the NFA. The problem is to create the original NFA. There is an assumption that the original NFA did not have any λ-transitions. Once the original NFA has been created, students can use the *Compare Equivalence* option with the DFA to determine if they have created the correct NFA.

3.4 Creating Automata Based on Properties

JFLAP can be used to construct examples that illustrate the properties of languages. For example, given two automata, build an automaton that represents the union of the two. The two automata may already be constructed. Using the *Combine Automata* option, both automaton are placed in the same window, one of them losing its start state status as only one state can retain this status. The user can then connect and modify them. In this case, a new start state is created and λ-productions are added from the new start state to each of the previous start states. The user can then test the new automaton on multiple inputs.

A more complicated example is to consider the property called SwapFirstLast(L), which takes the first letter of each string in L and swaps it with the last letter of that string. Students are to show that if L is regular, than SwapFirstLast(L) is regular. Students would approach this problem in two ways. First, using JFLAP they would construct a simple DFA M with language L, and then convert it to the DFA M2 for the language SwapFirstLast(L). Second, without JFLAP they would formally prove SwapFirstLast(L) is regular.

3.5 Determining Distinguishable States

One of the transformations in JFLAP is converting a DFA to a minimal state DFA. The algorithm in JFLAP assumes initially that all the final states are indistinguishable and all the nonfinal states are indistinguishable, grouping indistinguishable states into two sets, one for final states and one for non-final states. The algorithm then attempts to determine if some of the states in a set are distinguishable, thus splitting a set into two or more sets. If a user suspects that two states in the same set are distinguishable, they can modify the DFA to make each of these states a start state (at different times) and determine a string that is accepted by one of the modified DFAs and not the other. If there is such a string, then these states are distinguishable and need to be placed into separate sets.

3.6 Exploring with Nondeterminism

Most students are used to thinking sequentially. When given a problem that can only be solved nondeterministically, many students struggle. The problem of determining if a string is a palindrome can be solved by a nondeterministic PDA (NPDA), and not by a deterministic PDA (DPDA). Students taking the sequential approach to this problem want to find the middle and then determine if the right and left half match up. But this approach does not work with a DPDA. With JFLAP, students can build an NPDA for this problem and then observe how the nondeterminism works. When running the simulation with JFLAP, each current configuration is shown. For a valid input string, one of those configurations reaches the middle of the string and the simulation begins matching the left and right halves, continuing to acceptance.

3.7 Exponential Growth in Grammars

JFLAP can show the exponential growth in grammars, and the result of transforming the grammar. For example,

students are given the grammar on the left that contains a λ-production and asked to transform it into an equivalent grammar with no *lambda*-productions or unit-productions, the resulting CFG shown on the right. They are then asked to compare brute-force parsing of the two grammars. For input *aaababaabbb*, the grammar on the left takes a long time to accept, generating 13286 nodes in the derivation tree. The grammar on the right accepts quickly after generating 335 nodes in the derivation tree.

S → aB	S → aB
S → Ba	S → Ba
B → aBb	S → a
B → BB	B → aBb
B → bBa	B → BB
B → λ	B → bBa
	B → ab
	B → ba

Similar examples can be shown with unrestricted grammars. Those grammars with more items on the left side of a production will proceed more quickly in parsing.

3.8 Determining the Language of a Grammar

JFLAP can be used in determining the language of a given CFG. In one approach, the CFG can be tested on multiple inputs. In another approach, the user can divide the problem into smaller components. For each variable, enter its productions to determine the capability of that variable (replacing other variables with temporary terminals). For example, consider the grammar below with seven productions. The user can enter all the B productions in a new grammar window, with a small s for S (otherwise derivations are not possible). The user then derives strings $\{b, ab, bs, aabs, absa, \ldots\}$ and determines that $B \stackrel{*}{\Rightarrow} a^*b(\lambda + S)a^*$. Entering only S productions with a special terminal to represent the variable B, the user can determine that $S \stackrel{*}{\Rightarrow} a^*bBa^*$. Putting them together, $S \stackrel{*}{\Rightarrow} a^*ba^*b(\lambda + S)a^*$ or $(a^*ba^*b)^*a^*$. The user can then test the language by developing a set of test strings.

S → aS	S → Sa	S → bB	B → aB
B → Ba	B → bS	B → b	

3.9 In Depth Study of FOLLOW sets

One of the early steps in LL or SLR parsing is to compute the FOLLOW set for each variable in the grammar. The FOLLOW set of a variable is the set of all terminals that can follow this variable in some derivation. Students are given an algorithm for computing the FOLLOW sets, and many can follow the algorithm, but it is not clear that they really understand the meaning of the FOLLOW sets. For this problem, students are given a grammar and are asked to compute the FIRST and FOLLOW sets for the variables in the grammar. Then they are asked to show the sentential form in a derivation for each symbol in a FOLLOW set that shows that terminal immediately following the corresponding variable. They can solve this problem partially using JFLAP by parsing strings with the brute-force parser. A derivation is shown for each string and one can observe the sentential forms in the derivation. Several strings may be needed and a few sentential forms may not be found due to the order JFLAP replaces productions when given a choice.

3.10 Parsing Algorithms: Two Approaches

We show how JFLAP can be used to extensively study SLR parsing from two approaches. A similar approach applies to LL parsing. Given an SLR(1) grammar, the first approach is to convert the grammar to an NPDA using the SLR parsing method. The resulting NPDA has three states and is likely to be nondeterministic. Students then run the NPDA on several inputs, making the run deterministic by choosing the lookahead each time and freezing configurations that are not chosen. Next students view the parsing from a second approach. They build an SLR(1) parse table for the grammar and step through the parsing of the same inputs with the same lookaheads.

3.11 Parsing grammars that are not SLR(1)

With JFLAP one can build an SLR(1) parse table even if there is a conflict in the table. For each entry that has a conflict, the user chooses one of them. Then the user can proceed and parse strings using the parse table. Not all strings in the grammar can be parsed using the choices chosen. Here is a problem given to students to test their understanding of the parse table. Given a CFG that is not SLR(1) and a given input string, find the correct choices for conflicts in the parse table so the string can be parsed.

3.12 Running a Universal Turing machine

With JFLAP's 3-tape Turing machine, we have constructed a Universal Turing machine that has 34 states. Using the Universal Turing machine, a student can encode a simple Turing machine with a few transitions, each encoded transition will be a string of 0's and 1's of about length 15. A student can then enter an input consisting of the encoded machine followed by an encoded input string and observe the simulation.

3.13 Comparison of one-tape and two-tape TM

Students are given the language $a^nb^nc^n$ and asked to build a one-tape TM in JFLAP for this language, and then to build a two-tape TM in JFLAP for this language. They then compare the running of several input strings of different lengths on each TM. In this example, an efficient one-tape TM will run in $O(n^2)$ time and an efficient two-tape TM will run in $O(n)$ time.

4. NEW FEATURE: BUILDING BLOCKS

A new feature of JFLAP is Turing machine building blocks. A building block is a Turing machine with a specific purpose that can be used as a component in building Turing machines. One can build a complex Turing machine more easily using building blocks than states.

4.1 Creation of Building Blocks

To create a building block, create a Turing machine using states and transitions, and save it in a file. The Turing machine can then be read in as a building block by selecting the *Building Block Creator*. The building block appears as a box and can be connected with transitions. We provide special transitions for hooking up building blocks more easily. Building blocks can also be formed using a combination of states and building blocks.

With building blocks one wants to start with a simple foundation. We list simple Turing machines that can form

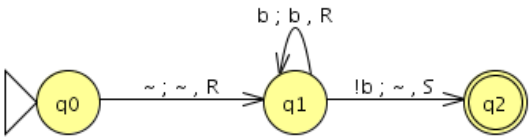


Figure 1: Building Block for Rnot_b

a library with which to build more complicated Turing machines. These building blocks can be provided for students to use or they can build some or all of them.

`R`	Move right once
`R_a`	move right once, keep moving right until an a
`Rnot_a`	move right once, keep moving right until not an a
`a`	write an a (don't move)
start	starting block
halt	halting block

Each of these represent a simple Turing machine. There are analogous machines for moving left `L`, `L_a` and `Lnot_a` and analogous machines for other symbols of the alphabet.

Building blocks can be connected using standard Turing machine transitions. In JFLAP the standard TM transition $a; b, R$ means to read the symbol a, write the symbol b and move right. We have created special symbols for transitions and a new type of transition. The symbol $\sim$ means to ignore a read or write. The transition $\sim; \sim, R$ means to ignore the symbol to read, ignore the symbol to write and move right. The symbol $!x$ used in the read position means to match any terminal that is not the terminal x. For example, Figure 1 is a Turing machine for moving right once and then continuing to move right until there is a symbol that is not a b. We have named this Turing machine Rnot_b and will use it as a building block.

One may want to connect two building blocks in one of two ways. First, one may want to connect them so they execute in sequence, connecting them with $\sim; \sim, S$ (S means stay put). Second, one may want to move to a second building block depending on the current symbol after processing the first building block. For example $a; \sim, S$ means if a is the current symbol on the tape then enter this building block and do not move on the tape head. To further simplify the connection between two building blocks, we have created a *Block Transition Creator* that only shows the read and assumes the write is $\sim$ and the move is S. To reduce the duplication of code that is similar except for one symbol, we allow the notation $a_1, a_2, \ldots a_n\}v$. This means if one of the a_i is read, any occurence of v later is replaced by a_i.

Figure 2 shows a Turing machine built solely with building blocks to represent the transducer $f(w) = w'$ such that w' has all the a's from w listed first, followed by all the b's in w. For example, $f(babba) = aabbb$. The Turing machine starts in a start building block, which represents a simple Turing machine of one state that is a start state and a final state. It then repeatedly moves right finding the first a past a b, replaces it with a b and then replaces the leftmost b with an a. When all the a's are to the left of all the b's, the tape head moves to the leftmost symbol and enters the halt building block. In this Turing machine, all the transitions were created with the *Block Transition Creator*. For example, the *start* block has an a transition to the *Rnot_a* block. This a transition means "if there is an a on the tape head, do not write on the tape and do not move the tape head, but go to the *Rnot_a* block for the next instruction."

Building block machines can be quite complicated. Turing machines built with building blocks can be named and saved in a file and used as a building block. Once a building block is read in using the Building Block Creator, a copy of its definition is stored in the new Turing machine. If the same building block is read in a second time, then it's old definition in the file is used. Once a building block is part of a Turing machine, it can be modified. When the *Attribute Editor* is selected, one can click on a block and then select the option *Edit Block*. The Turing machine for that block appears in a new tab and can be modified. Be cautious: if there are multiple uses of this block in a TM, modifying one creates a new definition only for that block. It is best to build and test a block before using it in Turing machines so that it does not need to be modified later.

The default name of a building block is the name of the file when the building block is read in. This name can be changed with the *Set Name* option.

4.2 Simulation with Building Blocks

There are five options for the simulation of input strings with Turing machines containing building blocks. One option, *Step*, provides a trace through the Turing machine one state at a time. When the trace enters a building block, the building block is highlighted as long as the trace is in a state within the building block. Selecting the *Focus* option will automatically change the view to display the transition diagram for the current building block, with the current state highlighted. Selecting the *Defocus* option changes the display back to the original Turing machine with its building block highlighted.

A second option, *Step by Building Block* displays the transition diagram of the original Turing machine and each block is considered one step in the simulation. Thus, the Turing machine moves quickly through a trace. If a block represents "Move right until a blank is seen," then *in one step* the tape head moves to the right to a blank. There are three fast run options. The *Fast Run* takes one input string and gives the result of acceptance or not without a trace. *Multiple Run* and *Multiple Run (Transducer)* return the result of several strings without providing a trace.

4.3 Other New Features

There are other new features of JFLAP. One is the ability to change the name of a state for all types of automata in JFLAP. The default names for states for all types of automata are qX where X is the number of the state starting with 0 . Figure 1 shows the default naming of states $q0$, $q1$ and $q2$. Figure 3 shows a Turing machine in which the four states have been renamed to *start*, *1*, *2*, and *3*. This Turing machine is a transducer for $f(w) = w'$ in which w must start with an a and have at least one b. The output is a string of b's equal in length to the first group of a's. Another new feature is that Turing machines that are transducers can be run with multiple inputs. Figure 4 shows the simulation of several input strings and the output of those strings. The fourth input string is invalid as it does not contain a b.

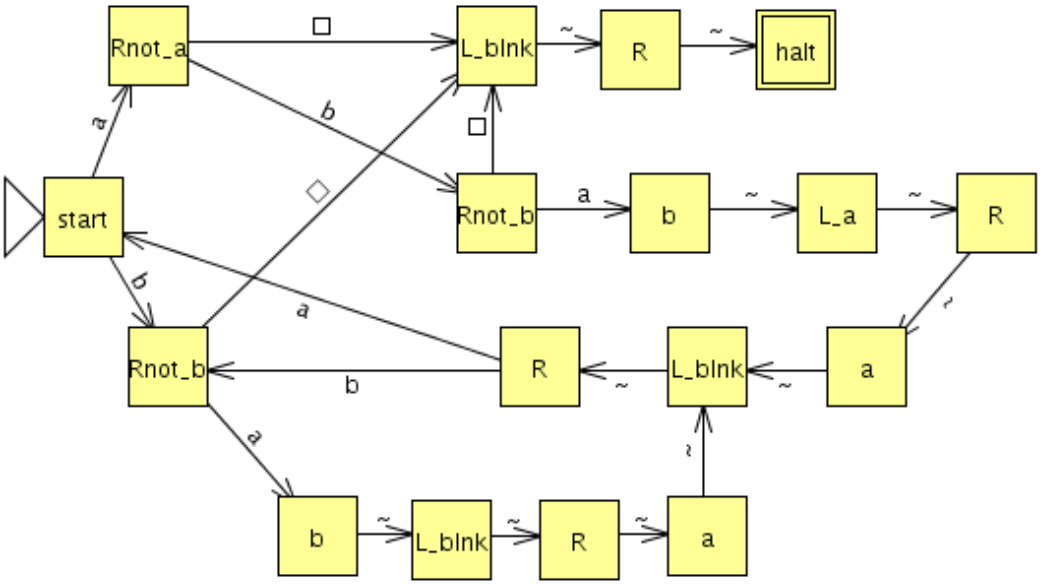


Figure 2: Turing machine to put a's first

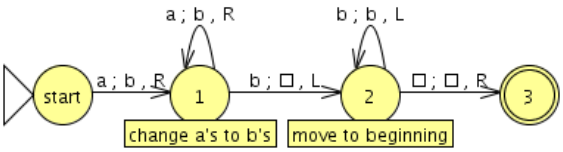


Figure 3: Change first group of a's to b's

Input	Output	Result
aaaaab	bbbbb	Accept
aaabab	bbb	Accept
ab	b	Accept
aaa		Reject

Figure 4: Multiple input for TM transducer

5. JFLAP'S USE AROUND THE WORLD

Since January 2003, JFLAP has been downloaded over 25,000 times in over 120 countries. The type of user was 29% undergraduate student, 18% graduate student, and 16% faculty. JFLAP was required use by 33% and not required by 36%. The type of use of JFLAP was 48% as a resource, 25% for homework, 15% as lecture and 12% as lab. The reason for using JFLAP was 46% taking a course, 14% teaching a course, and 6% research. These statistics do not add up to 100% as some users elected not to respond.

6. CONCLUSIONS AND FUTURE WORK

We are continuing to develop JFLAP with additional algorithms and ways to use JFLAP in the FLA course. We recently started a two-year study to evaluate JFLAP's effectiveness as a learning tool. A dozen universities are using JFLAP and participating in the study. A two-day JFLAP workshop was held in June 2005 and we received feedback on the use of JFLAP and now have many ideas for improvements that we plan to implement.

7. REFERENCES

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

[2] R. Cavalcante, T. Finley, and S. H. Rodger. A visual and interactive automata theory course with jflap 4.0. In *Thirty-fifth SIGCSE Technical Symposium on Computer Science Education*, pages 140–144. SIGCSE, March 2004.

[3] J. Cogliati, F. Goosey, M. Grinder, B. Pascoe, R. Ross, and C. Williams. Realizing the promise of visualization in the theory of computing. *JERIC*, to appear, 2006.

[4] 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.

[5] S. H. Rodger. Jflap web site, 2005. www.jflap.org.

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

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

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

[9] 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.

A Visual and Interactive Automata Theory Course with JFLAP 4.0

Ryan Cavalcante*
Dept of Computer Science
Duke University
Durham, NC 27708-0129

Thomas Finley†
Dept of Computer Science
Cornell University
Ithaca, NY 14853-7501

Susan H. Rodger‡
Dept of Computer Science
Duke University
Durham, NC 27708-0129
rodger@cs.duke.edu

ABSTRACT

We describe the instructional software JFLAP 4.0 and how it can be used to provide a hands-on formal languages and automata theory course. JFLAP 4.0 doubles the number of chapters worth of material from JFLAP 3.1, now covering topics from eleven of thirteen chapters for a semester course. JFLAP 4.0 has easier interactive approaches to previous topics and covers many new topics including three parsing algorithms, multi-tape Turing machines, L-systems, and grammar transformations.

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, pushdown automata, Turing machine, grammar, SLR parsing, LL parsing, L-system

1. INTRODUCTION

Many computer science students obtain only a superficial understanding of theory, even though theoretical concepts provide the fundamental basis for most areas of computer science. In particular, a thorough understanding of the theory of formal languages and automata (FLA) is crucial in designing programming languages and compilers. However, the traditional FLA course is taught in a nonvisual and pencil-paper problem solving manner. Students find this approach frustrating as they have no visualization to relate to and they do not receive immediate feedback on problems. This contrasts starkly with the hands-on nature in most of their other computer science courses which contain programming assignments.

Over the past twelve years, several software tools have been developed that attempt to change the traditional FLA course from a textual presentation to a more visual and interactive approach. Many of these tools focus on one particular concept such as finite automata[9, 10] or Turing Machines[1]. One tool[11] allows experimentation with several machines, with a focus on Turing machines. A tabular approach [4] covers many topics but has no graphical representation. A web-based textbook [3] that is partially complete allows experimentation with concepts through applets. JFLAP [5] is an instructional tool that covers many FLA topics in a visual and interactive manner, from experimenting with machines and grammars to experimenting with related proofs.

In this paper we describe JFLAP 4.0[8], a new version of JFLAP that can be used with topics from eleven chapters of an FLA course, double the number of chapters of material JFLAP 3.1[5] covered. JFLAP 4.0 includes many new topics such as multi-tape Turing machines, L-systems, grammar transformations, and three types of parsing (LL(1), SLR(1), and brute force). JFLAP 4.0 has new approaches for converting NFA to regular expressions (RE), and for the conversion of a regular expression to an NFA. JFLAP 4.0 has new features such as combining two automata and comparing the equivalence of two automata. With these new additions, JFLAP 4.0 can be used with almost all topics in an FLA course. We describe JFLAP 4.0's differences from JFLAP 3.1, JFLAP 4.0's new additions, its use in a FLA course at Duke University and feedback from using JFLAP.

2. JFLAP 4.0 VS. JFLAP 3.1

This section compares JFLAP 4.0 and JFLAP 3.1's coverage of topics by referring to a typical FLA textbook [6]. A typical automata theory course would cover Chapters 1 through 11 of this textbook and possibly two additional chapters of material either from this book or outside of this textbook. We list those 11 chapters plus two extra topics.

*The work of this author is supported in part by the National Science Foundation through grant NSF DUE-9752583.
†The work of this author is supported in part by the National Science Foundation through grant NSF DUE-9752583. This work was done while the author was at Duke University.
‡The work of this author is supported in part by the National Science Foundation through grant NSF DUE-9752583.

SIGCSE'04, March3–7,2004,Norfolk, Virginia, USA

Ch. 1	Mathematical Preliminaries
Ch. 2	Finite Automata
Ch. 3	Regular Languages and Grammars
Ch. 4	Properties of Regular Languages
Ch. 5	Context-Free Languages
Ch. 6	Simplification of Context-Free Grammars
Ch. 7	Pushdown Automata
Ch. 8	Properties of Context-Free Languages
Ch. 9	Turing Machines
Ch. 10	Other Models of Turing Machines
Ch. 11	A Hierarchy of Formal Languages
Extra 1	L-Systems
Extra 2	LL and SLR Parsing

The following shows what part of the chapter (where 1 is most of the chapter) that JFLAP 4.0 and JFLAP 3.1 can be used with.

Chapter	JFLAP 3.1	JFLAP 4.0
Ch. 1		
Ch. 2	1	1
Ch. 3	3/4	1
Ch. 4		1/2
Ch. 5	1/2	1
Ch. 6		1
Ch. 7	1	1
Ch. 8		
Ch. 9	1/2	3/4
Ch. 10	1/4	1/2
Ch. 11		1/4
Extra 1		1
Extra 2		1

JFLAP 3.1 covered four chapters of material from this textbook, spread out over six chapters. JFLAP 4.0 covered nine chapters of material, spread out over eleven chapters.

3. JFLAP 4.0 NEW FEATURES

In this section we describe the many new topics and features of JFLAP 4.0. JFLAP is written in Java.

3.1 New Approach for NFA to RE

The new approach in JFLAP 4.0 for converting an NFA to a regular expression (RE) is more visual and easier to understand for the user than in JFLAP 3.1. In the new approach one first enters an NFA. For example, we have entered the NFA in Figure 1. The NFA must be in a particular format with exactly one final state, which cannot be the start state, and there must be exactly one transition from every pair of states. JFLAP helps the user in collapsing multiple transitions from pairs of states and in adding $\emptyset$ between two states that did not previously have a transition.

All states except the initial and final state are removed one at a time. In our example we first remove state q_2. When q_2 is removed, its meaning must be incorporated into all the other transitions. Figure 2 shows the regular expressions that will be placed on the arcs after state q_2 is removed, and Figure 3 shows the resulting NFA (actually a general transition diagram (GTG) since there are regular expressions on the arcs) after removing state q_2. Similarly, state q_1 is removed and the resulting regular expression is shown in Figure 4.

3.2 New Approach for RE to NFA

In JFLAP 3.1, a bottom-up approach was used to convert an RE to an NFA. One constructed NFA submachines corre-

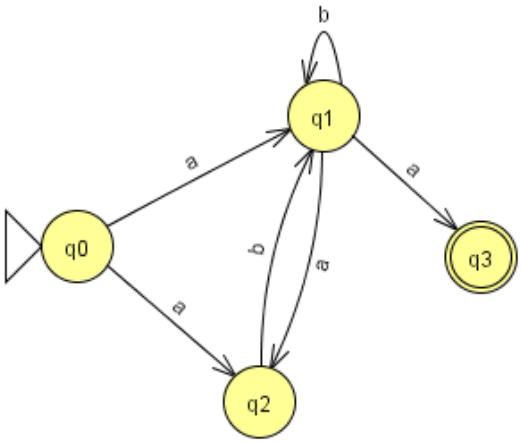


Figure 1: An NFA to convert

From	To	Label
0	0	ø
0	1	a+ab
0	3	ø
1	0	ø
1	1	b+ab
1	3	a
3	0	ø
3	1	ø
3	3	ø

Figure 2: Table in Removing state q_2

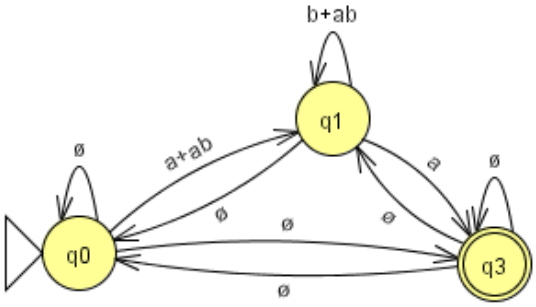


Figure 3: NFA after removing state q_2

((a+ab)((b+ab)*))a

Figure 4: The equivalent regular expression

sponding to low level operands of the RE and then connected these submachines into larger and larger submachines until the final equivalent NFA was constructed. For example, to convert the RE $ab + (c + d)*$ one first constructed two NFA that recognized the languages ab and $c + d$, then reformed the $c + d$ machine to recognize $(c + d)*$, and then combined the two machines to recognize $ab+(c+d)*$. How did JFLAP know to create the ab NFA, and then the $c + d$ NFA, and so forth? Implicitly JFLAP parsed the RE down to its lowest level operands. JFLAP 4.0 now makes this parsing more explicit without unduly complicating the conversion.

JFLAP 4.0 takes a top-down approach. The user works in the same direction the parser does, and observes and interacts with the parser. This serves to make the steps of the algorithm less "magical" by dividing the previously wholly implicit parsing of the RE into more easily comprehensible small chunks. One starts with a GTG with one initial state, one final state, and a single RE transition from the initial to the final state on the RE to convert. Using a modified automaton editor, the user repeatedly reduces each RE transition on an RE R into many RE transitions consisting of R's operands, and the user adds lambda transitions to duplicate the functionality of R's lost operators. At each stage the GTG is equivalent to the initial RE. After enough transition reductions the GTG becomes an NFA.

3.3 LL and LR Parsing

In JFLAP 4.0 one can start with a CFG and build an LL(1) parse table or an SLR(1) parse table through a series of steps, and then simulate the parsing of input strings. These tools were modeled after the LL and LR parsing in the tool JeLLRap[2].

We give an example of building the SLR(1) parse table for the grammar $S \rightarrow aSb, S \rightarrow b$. In JFLAP, the user first enters the CFG and selects the option to build the SLR(1) parse table. The window in Figure 5 appears with the grammar (not shown) and the fields in the window would be blank. The user would first enter in the FIRST sets for the variables, followed by the FOLLOW sets for the variables. Then the user would construct the DFA that models the parsing stack. Each state has marked rules or items associated with it. These items can be easily selected from a menu; they would be too tedious to type in as users had to do in JeLLRap. JFLAP gives hints along the way in the construction of the DFA such as an error message if you try to add a transition that doesn't exist, or if you have the wrong items for a state. Once the DFA is complete, the user enters in the entries in the parse table. Figure 5 shows the completed First and Follow sets, DFA and parse table.

Once the parse table is complete, the user can parse an input string. The parsing window is divided into four views: the grammar, the parse table, the parse tree and the input. The user enters an input string and steps through the parsing. For each step, the stack is adjusted, the current move in the parse table is highlighted, the new additions to the parse tree are added and a message such as "shifting a" is displayed. The user can view either a parse tree, an inverted parse tree or the derivation table.

If the grammar is not LL(1) or SLR(1), the user is allowed to continue, building a parse table that will have conflicts. If there is a conflict in the SLR(1) parse table, the user can choose one of the conflicts as valid and proceed to parse input strings.

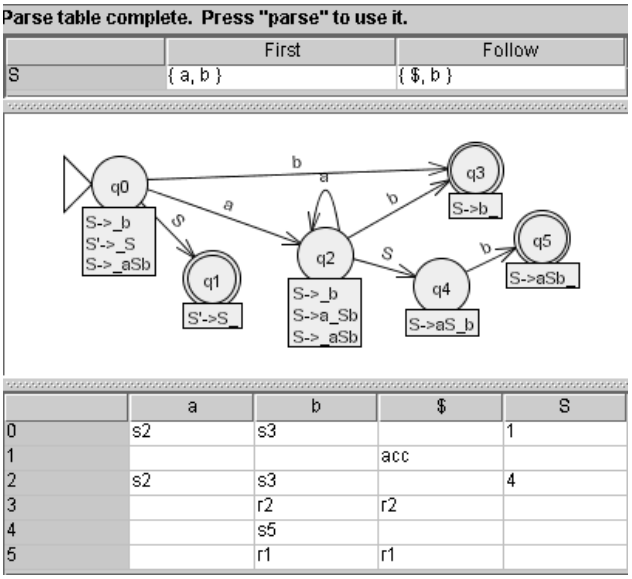


Figure 5: Building the SLR(1) Parse Table

3.4 Brute Force Parsing

The brute force parser in JFLAP 4.0 owes its heritage to Pâté's brute force parser[2]. The brute force parser allows the derivation of strings in the language of restricted and unrestricted grammars through exhaustive search. JFLAP's brute force parser improves over Pâté's in speed, memory requirements, pruning strategies, and aesthetics.

One first enters a restricted or unrestricted grammar in JFLAP and selects the option to perform brute force parsing. The interface is similar to JFLAP's interface for LL(1) and SLR(1) parsers. Figure 6 shows JFLAP's brute force parser. The naïvetè of the parser's exhaustive search explains two differences in the interface: the parse table's absence and the "pause" button's presence. Additionally, unrestricted productions allow multiple symbols to be replaced rather than single variables; if the step from one sentential form to another requires an unrestricted production that replaces multiple symbols, the corresponding nodes are grouped together in a bracket, and the replacement symbols appear as children of that "bracket node" as shown in Figure 6 with `aa` replaced by `b`.

3.5 L-Systems

In JFLAP 4.0 one may create and render L-systems[7]. Figure 7 shows the editor for L-systems: from top to bottom are components for the axiom, rewriting rules, and initial values of graphical parameters (colors, line width, turn angles, etc.). Figure 8 shows JFLAP's L-system renderer: the top of the window allows control of which derivation step to render (Figure 8 displays the 6th derivation), the large white panel shows derivation's rendering (a tree), and controls on the bottom turn the L-system, useful for 3D renderings.

JFLAP's L-systems have additional features beyond those standard in most L-systems tools. The turtle may turn in three dimensions, allowing the rendering of 3D structures. Some turtle commands take arguments, e.g. g will draw a line with default length, but $g(30)$ will draw a line 30 pixels long. Arguments may also be mathematical expressions. JFLAP also supports contextual rewriting rules that may

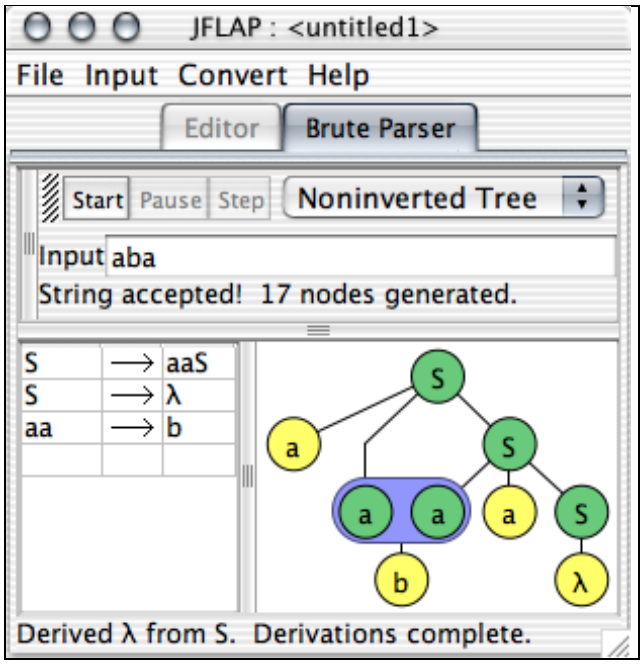


Figure 6: Brute Force Parser

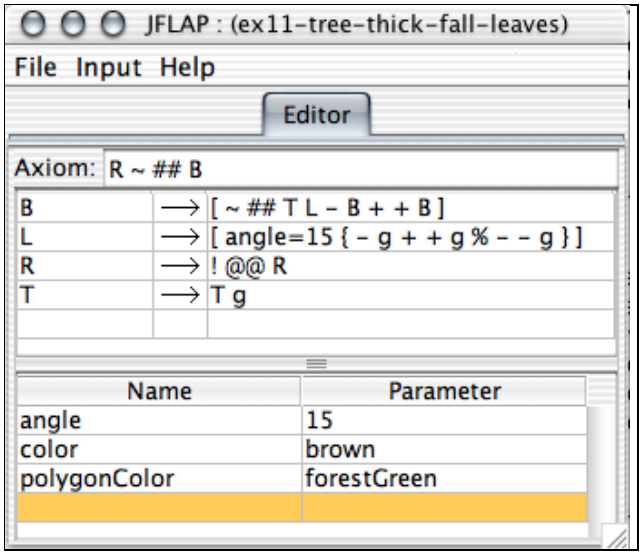


Figure 7: L-System Editor

rewrite a symbol only if it is prefixed and suffixed by certain symbols. JFLAP's L-systems are stochastic: if more than one rule may rewrite a symbol, a rule is uniformly randomly chosen from applicable rules to rewrite the symbol.

3.6 Transform Grammar

In JFLAP 4.0, one can transform a CFG to Chomsky normal form (CNF) in an easier to use interface than in Pâté[2]. One enters a CFG and then applies several algorithms to the CFG until the CNF is obtained. The algorithms are removing lambda productions, removing unit productions, removing useless productions, and converting the resulting grammar to CNF. Each algorithm has one or more interactive steps with some steps including a visual interpretation of the rules.

3.7 Other Features

Other new features in JFLAP 4.0 include the following. One can now create multi-tape Turing machines for up to five tapes. With three tapes, one can now create a Universal Turing machine. One can compare two finite automaton for equivalence. This allows students to check their design against another solution that may appear to be different. In particular, students can construct an automaton using properties (for example, reversal) and compare the result against a solution automaton. One can combine several automaton in the same window. This allows one to build a new automaton faster by using parts from other automaton.

4. JFLAP'S USE IN COURSE

We describe how we use JFLAP in teaching the FLA course CPS 140 at Duke University. JFLAP was designed to be general and to be used with most automata theory textbooks, such as [6]. We teach in a classroom with a computer whose monitor is displayed via a projector to a large screen. We use JFLAP during class in a number of ways.

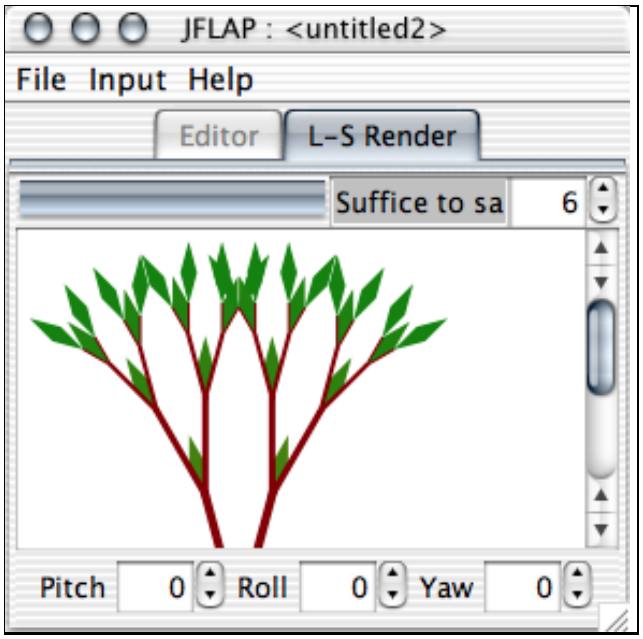


Figure 8: Rendering of Figure 7's L-system

- In the beginning we use JFLAP to show students how to use the editor to construct an automaton. For example, we demonstrate that states can be moved after they are created. In earlier years when students used JFLAP and we did not demo it in class, their designs were cluttered as they did not realize they could move states to make a prettier picture.
- We use JFLAP to solve problems during class. For example, we ask students to build an NPDA for the language ww^R (an even length palindrome). This NPDA cannot be built deterministically, and students are not used to thinking nondeterministically. They want to find the middle first. We build this NPDA with student input and test its correctness with several input strings.
- We use JFLAP to debug. For example, we will build an incorrect Turing machine for $a^n b^n c^n$ and ask students if it is correct and if not, for them to describe how to fix it. The machine we give them works for strings of the form $a^n b^n c^n$, however it also accepts incorrect strings. We ask for input strings that work and do not work and run them in class. Students then tell us how to fix the machine. We fix it and retry our input strings.
- We use JFLAP for illustrating proofs. For example we step through the conversion of a DFA to an RE.
- We use JFLAP to relate to other computer science concepts, such as running time. We compare the running time of an algorithm for $a^n b^n c^n$ on a one-tape Turing machine with an algorithm for it on a two-tape Turing machine.
- We use JFLAP to show how a PDA is used for LR parsing. We convert a CFG to a PDA and show it is nondeterministic. Then we ask the students to suggest lookaheads so that we can step through the parsing in linear time.

Students can use JFLAP for homework either to create an automaton to submit for grading, or to go through the steps in an algorithm. They can also use JFLAP to load examples done in class and to create additional problems to help them in understanding. JFLAP can be used outside of class by the instructor to grade students JFLAP submissions.

5. FEEDBACK

We have some feedback from JFLAP's use in CPS 140 at Duke University. In the spring of 2003 we used JFLAP for 6 of 9 homeworks. A questionnaire was given after their first use of JFLAP. In response to the question "Was JFLAP easy to use?", all 33 students responded yes. In response to the question "Did you look at the help at all? If so, what part did you look at and was it helpful?", 27 students did not look at the help. The 6 students who looked at the help said it was helpful. In response to the question "Do you prefer creating FA using JFLAP or drawing them on paper?", 17 students preferred to use JFLAP, 12 students preferred to use paper first, then JFLAP for testing and 2 students preferred paper.

JFLAP is being used around the world. A search of JFLAP (which appears to be a unique name) on Google showed over 2000 web pages have the word JFLAP on them. Many courses use JFLAP in some way and have it listed on their course web page. Our download site of JFLAP 4.0 has over 3800 downloads since January 2003. From a combination of course web pages and download feedback, we have determined that JFLAP is being used in over 40 countries.

6. CONCLUSION

JFLAP 4.0 can be used along with an automata theory textbook to create a hands-on FLA course. JFLAP 4.0 allows one to interact with concepts in eleven of thirteen chapters for an FLA course, doubling the amount of material JFLAP 3.1 covered. The instructor can use JFLAP during class to solve problems. Students can use JFLAP outside of class for homework and additional problems to aid in understanding of theory concepts. JFLAP is available for free [8].

7. REFERENCES

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

[2] A. O. Bilska, K. H. Leider, M. Procopiuc, O. Procopiuc, S. H. Rodger, J. R. Salemme, and E. Tsang. A collection of tools for making automata theory and formal languages come alive. In *Twenty-eighth SIGCSE Technical Symposium on Computer Science Education*, pages 15–19. SIGCSE, March 1997.

[3] C. Boroni, F. Goosey, M. Grinder, and R. Ross. Engaging students with active learning resources: Hypertextbooks for the web. In *Thirty-second SIGCSE Technical Symposium on Computer Science Education*, pages 65–69. SIGCSE, February 2001.

[4] D. Hannay. Hypercard automata simulation: Finite state, pushdown and turing machines. *SIGCSE Bulletin*, 24(2):55–58, June 1992.

[5] T. Hung and S. H. Rodger. Increasing visualization and interaction in the automata theory course. In *Thirty-first SIGCSE Technical Symposium on Computer Science Education*, pages 6–10. SIGCSE, March 2000.

[6] P. Linz. *An Introduction to Formal Languages and Automata, 3rd Edition*. Jones and Bartlett, Sudbury, MA, 2001.

[7] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[8] S. H. Rodger. Jflap web site, 2003. www.cs.duke.edu/∼rodger/tools/.

[9] M. Stallman, R. Cleaveland, and P. Hebbar. Gdr: A visualization tool for graph algorithms. In *Proceedings of Computational Support for Discrete Mathematics*, pages 17–28. American Mathematical Society, 1994.

[10] K. Sutner. Implementing finite state machines. In *DIMACS Workshop on Computational Support for Discrete Mathematics*, pages 347–363. DIMACS, March 1992.

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

Appendix 3 - Preface from JFLAP book

JFLAP: An Interactive Formal Languages and Automata Package

by

Susan H. Rodger and Thomas W. Finley

ISBN 0763738344

Preface (abridged)

This book is a hands-on guide through the Java Formal Language and Automata Package (JFLAP), an interactive visualization and teaching tool for formal languages. This book is intended as a supplement to an undergraduate automata theory course or an undergraduate compiler course. This book is not a textbook! We assume the user is using a textbook along with our book. Our book guides users interactively through many of the concepts in an automata theory course or the early topics in a compiler course, including descriptions of algorithms JFLAP has implemented and sample problems for reinforcement of concepts. However, our book assumes that the user has read briefly about these concepts first in an automata theory textbook or compiler textbook.

JFLAP allows users to create and operate on automata, grammars, L-systems, and regular expressions; the term *structure* is used to refer to any single automaton, grammar, L-system, or regular expression. JFLAP offers the following major groups of operations to apply to structures:

Explore the Language of Structures JFLAP has the ability to simulate input strings on nondeterministic automata, build parse tables and parse trees for grammars, and render successive expansions of L-systems. The automata represented in JFLAP are finite automata (FA), pushdown automata (PDA), and multitape Turing machines. The parsing algorithms in JFLAP are brute-force parsing, LL(1) parsing, and SLR(1) parsing.

Convert Between Equivalent Structures A wide range of conversion abilities are available, e.g., regular expression to FA, nondeterministic FA to deterministic FA (DFA), PDA to grammar, grammar to PDA, DFA to minimized DFA, context-free grammar to Chomsky Normal Form grammar, and others.

Miscellaneous Analysis of Structures JFLAP also offers a few sundry analysis tools to display properties of structures, like highlighting λ-transitions, highlighting nondeterministic states, and determining the equivalence of two FAs.

In addition to designing the structures listed above, our book guides the user through interactive alternative perspectives that would be difficult to do with pencil and paper. Here we list several examples from the book.

- In Chapter 2 users convert a deterministic finite automaton (DFA) to a minimal DFA. During the conversion, users must determine whether or

not two states p and q are distinguishable. With JFLAP users make two copies of the DFA — one with p a start state and the other with q a start state — and run both DFAs on the same set of input strings to determine whether the states are distinguishable.

- In Chapter 6 users parse context-free grammars using a brute-force parser. To determine the language of a grammar, users enter each variable's productions separately (when possible) and determine the language of each variable first, then put them together for the language of the grammar.

- In Chapter 8 users parse a grammar with an SLR(1) parser. They then build an NPDA that models the SLR(1) parsing process for this grammar and run the NPDA with the same strings. The NPDA is likely nondeterministic, so the students guide the run with lookaheads.

JFLAP uses general definitions of its structures to allow it to fit with a range of textbooks. We mention some of these definitions here. Definitions for each structure are in the corresponding chapter. Instructors might prefer to require students to use a subset of the JFLAP definition if that fits with their textbook.

- Finite automata allow users to enter strings of length zero or greater. Instead, an instructor might want to require students to enter strings of length zero or one.

- Pushdown automata can pop zero or more symbols from the stack in each transition. An instructor might want to require students to always pop one symbol.

- Turing machine movements for the tape head are Right, Left, and Stay. An instructor might want to require students to use only Right and Left.

Organization

This book covers topics from a formal languages and automata theory course, with additional topics on parsing and L-systems. We find that undergraduate students need to see an application to understand why this material is important. With SLR(1) parsing, students see the use of DFAs and PDAs. L-systems are included as an alternative grammar that results in interesting visual pictures.

We recommend chapters for an automata theory course and for the first part of a compiler course. Both courses should have students look at the JFLAP Startup section following the preface before reading the chapters.

For an automata theory course, the minimal coverage would be Chapters 1–6, and 9. Our book covers automata before grammars, so if your textbook covers grammars before automata then for regular languages you should cover Sections 3.1–3.2, 1, 2, 3.3–3.4. in that order. If context-free grammars are covered before PDA in your textbook, then cover them in the order 6.1, 5, 6.2 and 6.3. Optional topics include Chapters 7, 8, 10 and 11. If instructors want

to cover SLR(1) parsing only, but not LL(1) parsing, then cover Sections 8.1 and 8.3.

For a compiler course, related chapters on parsing are Chapters 1–2, 3.1–3.3, 5, 6.1, 6.2, 8, and 11. Skip Section 8.2 if you are not covering LL(1) parsing. Chapter 11 is optional but provides some interesting thoughts on how one parses other types of grammars, including the "parse tree" for unrestricted grammars in JFLAP. This is not really a tree, but rather a directed acyclic graph.

We now give a brief description of the highlights of each chapter.

Chapter 1, Finite Automata guides the user through editing and simulating both deterministic and nondeterministic automata.

Chapter 2, NFA to DFA to Minimal DFA guides the user through transforming an NFA into a DFA and then a minimal state DFA. In the NFA to DFA algorithm, a DFA state represents corresponding states reachable from the NFA. In the DFA to minimal DFA algorithm, states are grouped into sets of indistinguishable states using a tree method.

Chapter 3, Regular Grammars guides the user through editing and parsing grammars. Its conversion algorithms between FA and right-linear grammars are standard.

Chapter 4, Regular Expressions has conversion algorithms that use generalized transition graphs (GTG), wherein a transition label can be a regular expression. The regular expression to NFA algorithm starts with a GTG with one transition containing the regular expression and deexpressionifies one operation at a time until the result is an NFA. The FA to regular expression algorithm starts with an FA represented as a GTG with a transition between every pair of states (using $\emptyset$ if there was no transition). The algorithm removes one state at a time, replacing transitions with regular expressions, until there are only two states remaining.

Chapter 5, Pushdown Automata introduces the editing and simulation of NPDAs.

Chapter 6, Context-Free Grammars revisits the brute-force parser and goes into detail in terms of how it works. Two algorithms are presented. The CFG to NPDA algorithm uses the model for LL parsing. Students do not need to know LL parsing for this chapter. The NPDA to CFG algorithm can easily result in a rather large grammar with many useless productions, and thus seems magical to students. After the conversion students test the same strings in both the original NPDA and the resulting CFG.

Chapter 7, Transforming Grammars covers the transformation of a CFG to Chomsky Normal Form. Several transformations occur including removing λ-productions, unit productions, and useless productions. Students test the same strings in the transformed and original grammars.

Chapter 8, LL and SLR Parsing describes FIRST and FOLLOW sets that are needed in parsing, and the two parsing methods LL and SLR. For both LL and SLR parsing, students are shown how to convert a CFG to an NPDA for that method, how to build the parse table, and how to compare the NPDA with parsing. In parsing a string, students can view either the derivation or a parse tree.

Chapter 9, Turing Machines shows how to edit and simulate a one-tape Turing machine first, then expands to multi-tape Turing machines, including an example of a universal Turing machine.

Chapter 10, L-systems describes how to edit L-systems and use them to produce plants, fractals, and other graphical structures.

Chapter 11, Other Grammars in the Hierarchy describes how to use the brute-force parser to parse strings with unrestricted and context-sensitive grammars, and some insight about how to structure these grammars so brute-force parsing is not totally intractable.

JFLAP Software

JFLAP and all files referenced in this book are available at `www.jflap.org` and are free! We recommend that files be used in conjunction with the text.

Appendix 4 - Some Evaluation Results from JFLAP study 2005-2006

Partial results from the first year of the JFLAP study, receiving 36 valid responses on the Implementation section from a total of six participating schools.

Implementation Questions

Question	Response Options	Frequency
Were you required to use the JFLAP software for this course?	Yes	35
	No	1
Was use of the JFLAP software demonstrated in class?	Yes	36
	No	0
Did your professor use JFLAP to demonstrate concepts during office hours?	Yes	12
	No	4
	Not applicable	20
Was the JFLAP manual required reading?	Yes	18
	No	18
Was use of the JFLAP software required for homework assignments?	Yes	28
	No	7
Was use of the JFLAP software required for group projects?	Yes	3
	No	18
	Not Applicable	15
Was use of the JFLAP software required for take home exams?	Yes	6
	No	10
	Not Applicable	20
Did you use JFLAP software to study for in-class exams?	Yes	20
	No	16
Did the professor use JFLAP software as part of lectures?	Yes	33
	No	3
Did you use JFLAP software for in class exercises?	Yes	22
	No	13
Did you use JFLAP software for labs?*	Yes	10
	No	9
	Not Applicable	16
Did you feel that you had time to learn how to use the JFLAP software?	Yes	33
	No	2
Did you feel that using the software took time away from other study activities?	Yes	3
	No	33
Did the time and effort it took to use JFLAP help you get a better grade in the course?	Yes	23
	No	13
Was it easier to use the JFLAP software or was it easier to draw it out by hand?	It was easier to use the software.	30
	It was easier to draw it out by hand.	6
Did you feel that you would have done as well in the course if you had not used the JFLAP software?	Yes	18
	No	13
	Not Applicable	5
What percentage of the course utilized the JFLAP software?	0-25 percent.	16
	26-50 percent	15
	51-75 percent	3
	76-100 percent	2

Question	Response Options	Frequency of Response
How easy was it to install the JFLAP software?	Very easy	19
	Easy	12
	Neither Easy nor Difficult	5
	Difficult	0
	Very Difficult	0
How easy was it to use the drawing tools of the JFLAP software?	Very easy	18
	Easy	12
	Neither Easy nor Difficult	6
	Difficult	0
	Very Difficult	0
How easy was it to run the automata you designed with JFLAP software?	Very easy	19
	Easy	14
	Neither Easy nor Difficult	2
	Difficult	1
	Very Difficult	0
How easy was it to interpret results from the test run in JFLAP software?	Very easy	13
	Easy	16
	Neither Easy nor Difficult	6
	Difficult	1
	Very Difficult	0
How clear were the JFLAP software menus?	Very clear	8
	Clear	19
	Neither Clear nor Unclear	6
	Unclear	2
	Very Unclear	1
How flexible was the JFLAP software in simulating the automaton for your course?	Very flexible	11
	Flexible	22
	Neither flexible nor inflexible	3
	Flexible	0
	Very inflexible	0
How frustrating was it to use the JFLAP software?	Very frustrating	0
	Frustrating	3
	Neither frustrating nor appealing	13
	Appealing	18
	Very appealing	2
Does the JFLAP software have sufficient power to meet the programming challenges of the course?	Very weak	0
	Weak	1
	Neither weak nor powerful	9
	Powerful	18
	Very powerful	8
What is your overall assessment of the JFLAP Software?	Very poor	0
	Poor	0
	Neither poor nor good	2
	Good	24
	Very good	10

Appendix 5 - 68 Letters in Support of JFLAP

We have categorized the letters in support of JFLAP into four categories. 1) Letters from 48 faculty from 47 US institutions using JFLAP in teaching 2) Letters from 15 Faculty in Non-US institutions using JFLAP in teaching and 3) Five Other Letters including the use of JFLAP in research and a letter from Rockford Ross, one of the evaluators on our current NSF grant.

We provide a lists of who submitted letters, followed by the actual letters.

Letters from 48 Faculty from 47 US institutions using JFLAP in teaching

Robert Berwick, Professor of Computer Science, Engineering and Computational Linguistics, MIT
Ran Libeskind-Hadas, Professor of Computer Science, Harvey Mudd College
Christoper Brown, Associate Professor, Dept of Computer Science, United States Naval Academy
Sanjoy Baruah, Professor, Dept of Computer Science, Univ. of North Carolina at Chapel Hill
Costas Busch, Assistant Prof, Dept of Computer Science, Rensselaer Polytechnic Institute
Ralph Wilkerson, Professor of Computer Science, University of Missouri Rolla
Eugene Wallingford, Associate Professor, Dept of Computer Science, University of Northen Iowa
Valerie Harvey, Professor, Computer and Information Systems, Robert Morris University
David Taylor, Assistant Professor, Dept of Computer Science, San Jose State University
Lynn Andrea Stein, Prof. of Computer \& Cognitive Science, Franklin W. Olin College of Engineering
Kelly Shaw, Assistant Professor, Dept of Math and Computer Science, University of Richmond
Shannon Pollard, Assistant Professor of Computer Science, Elon University
Andrew Phillips, Professor of Computer Science, University of Wisconsin - Eau Claire
Michelangelo Grigni, Associate Prof., Dept of Math and Computer Science, Emory University
Robert Sloan, University Illinois Chicago
Peter Linz, Emeritus Professor, Dept of Computer Science, University California Davis
Jacquelyn Long, Assistant Professor, Computer Science Department, Norfolk State University
Gary Lewandowski, Professor, Math and Computer Science, Xavier University
Chris Lane, Assistant Professor, Dept of Math and Computer Science, Pacific University
Martha Kosa, Associate Professor, Dept of Computer Science, Tennessee Tech University
David Kay, Lecturer S.O.E. in Computer Science, University California Irvine
Dwight House, Associate Professor, Dept of Math and CS, Fayetteville State University
Dawit Haile, Chair, Dept of Math and Computer Science, Virginia State University
Ganesh Gopalakrishnan, Professor, School of Computing, University of Utah
William Turkett, Assistant Professor, Dept of Computer Science, Wake Forest University
Dale Brown, Professor of Computer Science, College of Wooster
Judy Goldsmith, Professor, Computer Science, University of Kentucky
Judith Gersting, Chair, Computer Science Dept, University of Hawaii Hilo
John Dooley, Associate Prof and Chair, Dept of Computer Science, Knox College
Wanda Dann, Associate Professor of Computer Science, Ithaca College
Jose Cordova, Associate Prof and Chair of Computer Science, Univ. of Louisiana Monroe
Muhammad Chaudhary, Associate Professor of Computer Science, Millersville Univ.
Louis Ziantz, Instructor of Computer Science, Dickinson College
Denise Byrnes, Associate Professor of Computer Science, College of Wooster
Judith Coomes, Associate Professor, Dept of Computer Science, William Patterson Univ.
Alice Dean, Professor of Mathematics, Math and Computer Science Dept, Skidmore College
Bruce Elenbogen, Assoc. Prof., Computer and Information Science, Univ Michigan Dearborn
Joan Lucas, Associate Professor, Computer Science Dept, SUNY Brockport
James Riely, Associate Professor, School of Computer Science, DePaul University
Nancy Van Ness, Senior Lecturer II, Computer Science Dept, Univ Texas at Dallas
Rakesh Verma, Professor, Computer Science Deptartment, University of Houston
Soe Than, Professor, Dept of Math and Computer Science, Virginia Military Institute
Michael Gousie, Associate Professor of Computer Science, Wheaton College

Ellen Walker, Professor of Computer Science, Hiram College
Ralph Zegarelli, Computer Science Faculty, University of Hartford
Jill Zimmerman, Associate Professor, Math and CS Dept, Goucher College
Fereydoun Kazemian, Assoc Prof of Computer Science, Rochester Institute of Technology
Harry Harrison, Faculty, Park University, Missouri

Letters from 15 Faculty in Non-US institutions using JFLAP in teaching

Lars Asker, Assoc. Prof., Dept of Computer and System Sciences, Stockholm Univ, Sweden
Albert Hoogewijs, Tenured academic staff, Dept of Pure math and Algebra, Ghent Univ, Belgium
James Harland, Associate Professor, RMIT University Melbourne, Australia
Lila Kari, Assoc Prof, Dept of Computer Science, Univ of Western Ontario, London, Canada
Germana Nobrega, Professor, Computer Science, Universidade Catolica de Brasilia, Brazil
Mina Zolfy Lignvan, Lecturer of Information Technology Dept, University of Tabriz, Iran
Antonio Dourado, Professor, Dept of Informatics Engineering, University of Coimbra, Portugal
Piotr Dziurzanski, Assistant Professor, Sczcecin University of Technology, Poland
Stefano Crespi Reghizzi, Full Professor of Computer Science, Politecnico di Milano, Italy
Andreas Rittershofer, Computer Science HS teacher, Dietrich-Bonhoeffer-Gymnasium, Germany
Jairo Rocha, Associated Professor, Univ. of the Baleric Islands, Palma, Spain
Kamilla Klonowska, PhD Student, School of Engineering, Blekinge Inst of Tech, Ronneby, Sweden
Yih-Kuen Tsay, Associate Professor, National Taiwan University, Taiwan
Frank Neven, Hasselt University, Belgium
Rosa Rodriguez Sanchez, Faculty, University of Granada, Spain

Other Letters

Rocky Ross, Professor of Computer Science, Montana State University, USA
Siamak Kolahi, Masters Student, Dept of Computer Science, Concordia University, Canada
Leonardo Mariana, Researcher, University of Milano Biccoca
Diana von Bidder, Dipl Informatik-Ing, Swiss Institute of Technology, Zurich
Deian Tabakov, PhD. Student, Rice University, USA

Massachusetts Institute of Technology

June 15, 2006

Professor Susan Rodger
Associate Professor of the Practice
Computer Science Dept. Box 90129
Duke University, Durham NC
27708-0129

Dear Susan:

This letter is to voice my strong support for your quite marvelous Java program, JFLAP. I've been a professor in CS (and Brain and Cognitive Sciences) here at MIT for nearly a quarter-century, and for the past several years, I've used this program in several undergraduate (and a few graduate) classes. In particular, I've found it extremely helpful for students unfamiliar with formal automata to gain a quick, visual appreciation of what these devices are all about. Here, I've used it in our undergraduate class in AI, 6.034, since many of the students there come to the class without first taking a class in automata theory. Equally, I've found that for the Brain and Cognitive Science students who take my course in natural language processing, 6.863J, in order to understand the role of finite-state devices and transducers in morphology and language, JFLAP proves indispensable – I require warm-up exercises using the system in some of the problem sets. I also use the system in class to illustrate some simple points about the Chomsky hierarchy. All very effective! The students really appreciate this kind of visual demonstration. Indeed, some of the students in 6.863J have extended JFLAP so that it works as a finite-state transducer (i.e., an automaton with outputs), since this is a good model for linear morphology. So, JFLAP is also readily extendable as well – far more than one could hope for in even commercial software. The design choice of Java works well here.

So in brief, a great teaching tool that has worked well for undergrads here at MIT, and graduate students in other departments unfamiliar with automata theory – a great visual supplement. JFLAP serves as a wonderful model for this kind of interactive teaching aid.

Sincerely,

Robert C. Berwick
Professor of Computer Science and Engineering and Computational Linguistics

Robert C. Berwick
Professor of Computer Science
Stata Center
Building 32D–728

77 Massachusetts Avenue
Cambridge, Massachusetts
02139–4307

Phone 617–253–8918
Fax 617–253–3578
Email berwick(at)csail.mit.edu

Department of Computer Science
Harvey Mudd College
301 Platt Boulevard
Claremont, CA 91711

May 30, 2006

Needs Premier Award 2006 Selection Committee

Dear Members of the Selection Committee,

I am writing in strong support of the `Jflap` automata simulation tool developed by Dr. Susan Rodger at Duke University. I am a professor of Computer Science at Harvey Mudd College in Claremont, California and the Joseph B. Platt Endowed Chair for Effective Teaching.

According to the JFLAP web site, Harvey Mudd College is the largest user of the tool (based on the number of downloads). My colleagues and I have been using JFLAP in our "CS 60: Principles of Computer Science" for the last six years. This is the second course in our computer science sequence and is based on a "breadth-first" model in which students are exposed to some of the major intellectual ideas of computer science. The course is typically taken by freshmen and sophomore computer science majors but also by students in all disciplines with an interest in computer science.

In this course, JFLAP is required for several assignments dealing with deterministic and nondeterministic finite automata and Turing Machines. We give several demonstrations of JFLAP in class and then ask students to solve a number of challenging problems using the tool. The use of JFLAP is augmented by having students write formal proofs of the "optimality" (minimum number of states used) of the automata that they construct.

In my opinion, JFLAP is an outstanding learning tool. It is powerful and feature-rich, and yet is easy for students to quickly use and master. (The documentation in Haiku is a very nice touch!) The tool helps reinforce theoretical concepts in a concrete way. In my estimation, students understand automata and Turing Machines in a deeper way than when we simply drew them on paper in the "olden days". The ability to simulate the automata and Turing Machines is extraordinarily compelling and helps reinforce concepts. In particular, nondeterminism is a challenging concept for many students and JFLAP offers a very nice way of visualizing the behavior of

nondeterministic machines.

Our students report that they thoroughly enjoy JFLAP. One testament to its popularity is that students tend to work on optional bonus problems in CS 60 that use JFLAP to a greater extent than other optional pencil-and-paper or programming problems in the course. They also frequently explore the other models of computation that JFLAP supports but that are not discussed in CS 60. From my perspective as a teacher, the fact that JFLAP encourages students to explore material beyond the scope of the course is wonderful and exciting.

Sincerely,

Ran Libeskind-Hadas
Professor of Computer Science
Joseph B. Platt Endowed Chair

An Open Letter Concerning the Effectiveness of JFLAP

I have been using JFLAP in our course SI472, "Theory of Computing", for the last three years, and it has been used by the department for even longer. I use JFLAP to help with Finite Automata and Turing Machines. In both cases I spend one period in a lab with a sequence of JFLAP exercises for my classes, followed by an assignment that must be done in JFLAP. From that point on its use is optional, but most students keep using it, as evidenced by the JFLAP screen captures that they turn in as part of later homework assignments.

I think JFLAP is an important tool for teaching theory of computing. It addresses two key issues: the difficulty students have in thinking of computational models as concrete, and sloppiness of thought and work. The dynamic nature of JFLAP models and the manipulative nature of students' interactions with the models they build makes them seem more real to students than diagrams they draw. Moreover, the fact that these models can be run on inputs encourages students to question whether the machines they construct are correct, test them and debug them – something they are manifestly unwilling to do with machine models they create with pencil and paper. Moreover, there is an undeniable "fun factor" to creating machines in JFLAP. Students like it.

In short, I think JFLAP is an excellent system that really helps students learn some key concepts in the theory of computing.

Christopher W. Brown
Associate Professor
Department of Computer Science
United States Naval Academy

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Sanjoy Baruah, *Professor*
Phone: +1 919 962-1803
e-mail: baruah@cs.unc.edu

Department of Computer Science
Campus Box 3175, Sitterson Hall
Chapel Hill, NC 27599-3175

To whom it may concern

I am writing this letter concerning my experiences with the JFLAP package for teaching automata theory. I have used JFLAP with the course COMP 181: *Models of Language and Computation* at the University of North Carolina. This is a course taken primarily by junior and senior undergraduates in computer science, and by graduate students in computer science who feel that their undergraduate education was not rigorous enough in theoretical aspects of computer science.

I have used JFLAP as part of COMP 181 for about three years now. I recommend it to the class near the beginning of the semester (accompanied by a demonstration of its installation and use), and encourage the students to use it to complement the lectures. I also use it extensively during office hours to help those students that find the more abstract concepts harder to grasp. In addition, I have occasionally used it as a demonstration tool during lectures.

I find JFLAP to be particularly helpful in a one-on-one setting with students – students who have difficulty with abstraction find the "experimental" aspect of a tool like JFLAP immensely useful. COMP 181 is taught in a rather theoretical/ abstract manner, and there typically is a minority of students who find abstraction difficult to deal with. These students seem to find JFLAP invaluable in helping them with abstract ideas and concepts, by letting them visualize these ideas and concepts and hence rendering them more concrete. JFLAP's ease of use encourages students to experiment extensively and repeatedly to thoroughly understand and internalize difficult aspects of automata behavior.

In summary, I find JFLAP to be a very valuable tool that greatly facilitates the course objective of communicating difficult theoretical concepts. I fully intend to continue using it in class, and in fact consider it quite likely that I will be using it more and more in the future.

Sincerely yours,

Sanjoy K. Baruah

Costas Busch
Assistant Professor
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
Email: buschc@cs.rpi.edu
Web: http://www.cs.rpi.edu/~buschc
Phone: 518-276-2782

June 28, 2006

To whom it may concern:

I am very happy to write this letter of support for the courseware JFLAP developed by Prof. Susan H. Rodger.

I have been using JFLAP since the Fall of 2005 in the course CSCI-2400 "Models of Computation". The topic of the course is an introduction to the theory of computation which is a fundamental subject in computer science. The course explores a variety of subjects such as: theory of automata including finite automata, pushdown automata, and Turing machines; theory of grammars including context-free, context-sensitive, and unrestricted grammars; decidability and time complexity theory. Due to its theoretical nature, the students consider this course as one of the hardest courses in the computer science curriculum.

JFLAP is a software tool which helps to design and simulate automata, grammars, and parsers. It also supports conversions between automata and grammars. I used JFLAP for the first time in Fall 2005 when I taught the course to a class of 60 students. The course consisted of 10 homeworks in which the students were required to use JFLAP. In the homeworks, the students designed and simulated automata and grammars with JFLAP. In particular, they used the following options of JFLAP: Finite Automaton, Grammar, PushDown Automaton, and Turing Machine. They also used the additional option which converts grammars to automata. I have demoed JFLAP in the first lectures of the course after which the students were able to use JFLAP on their own.

JFLAP has proven to be an extremely useful learning tool for the theory of computation. In the past (before the Fall of 2005) I taught the same course without JFLAP where I noticed that the students had a hard time in learning basic concepts of computability theory such as non-determinism in automata and the functionality of grammars; the student's couldn't make a connection between the theoretical concepts and their applications. Ever since, I was looking for methods that could improve the learning level of the course. JFLAP proved to be the solution I was looking for since it made the course concepts clearer to the students and easier to understand. In essence, JFLAP helped to make a better connection between the

basic theoretical concepts of the course and their applications in the practical domain, since the students were able to build explicit automata and grammars which could be simulated. As a consequence of using JFLAP, the students got higher grades than previous years, which indicated that they learned the material better. Furthermore, I received very nice comments from the students which appreciated the usefulness of JFLAP and were proactive to solving the homeworks with it.

In sum, JFLAP is a well developed courseware aimed to improve the teaching of the difficult subject of theory of computation. JFLAP makes an important contribution to computer science by bridging the gap between theory and practice and making theory accessible to students. Therefore, I strongly support JFLAP.

Yours sincerely,

Costas Busch

UNIVERSITY OF MISSOURI - ROLLA

May 31, 2006

To Whom It May Concern:

I have been asked by Professor Susan Rodger to submit a letter of support for the academic software package that she developed for the teaching of Automata Theory, specifically the package JFLAP. I have been teaching CS 330, Automata Theory, for many years and the first time I encountered this software was during a talk given by Dr. Rodger at a national conference on computer science education. Since then I have supplied my classes with the links to the JFLAP website and also explained its use in class. While the use of the software is optional on the part of the students, they soon find that it is an invaluable aid in checking their solutions to problems that I assign in class. Typically, I cover most of the material in the first 9 chapters of Peter Linz's book "An Introduction to Formal Languages and Automata" during our 15 week semester. The class has a mixture of both undergraduate and graduate students on a campus that is 80% engineering and science students.

Students have remarked favorably to having the tool available for the course. I think this is partly due to the fact that most students want to actually see the application of a computational tool to an area of mathematics/computer science that is considered fairly abstract by most students. I have been very impressed by the ease with which most students are able to make effective use of JFLAP. The learning curve is not very steep for the beginning user. I just wish there were more tools for teaching the more abstract concepts of computer science.

Ralph Wilkerson
Associate Dean for Research and Graduate Studies
College of Arts and Sciences
Professor of Computer Science
Department of Computer Science
University of Missouri-Rolla
Rolla, MO 65401

Ph: 573-341-4653
Email: ralphw@umr.edu

University of Northern Iowa
Department of Computer Science
305 Innovative Teaching & Technology Center
Cedar Falls, Iowa 50614-0507
319-273-2618
Fax: 319-273-7123
Web: www.cs.uni.edu
E-mail: dept@cs.uni.edu

January 19, 2006

Susan H. Rodger, Ph.D.
Department of Computer Science
Dept. Box 90129
Duke University
Durham NC 27708-0129

Dear Dr. Rodger,

My name is Eugene Wallingford, and I am Associate Professor and head of the Department of Computer Science at the University of Northern Iowa. I use JFLAP in my compiler course, 810:155 Translation of Programming Languages. I have been using JFLAP in this course since the fall of 2003.

My students learn JFLAP when they need to create regular expressions in order to implement a language scanner, but they really begin to use it when they develop parsers. I do not require students to use JFLAP, but I know that students use it both from their questions and from the print-outs of finite state automata that they show me. I use the tool in lecture to show conversions, but I do not show students very much of JFLAP's power, leaving them to explore and learn a lot on their own.

JFLAP's support for nondeterministic and deterministic finite automata—and especially conversions from NFAs to DFAs and from DFAs to minimal DFAs—helps students to make connections between the ideas they see in their theory course (and in the theoretical material in their compiler text) and the ideas they learn about implementing a compiler. With a language of significant size, theory-based approaches to implementing scanners and parsers are essential, and JFLAP makes using thoase approaches possible. I also like that JFLAP embodies another key principle of computer science: building tools that do routine and tedious work for us. Students come to appreciate this mindset in this course.

Students find JFLAP to be a useful tool, and I find that it makes implementing a compiler—the true goal of my course—more doable. I will continue to use it and make it available to my students.

Sincerely,

Eugene Wallingford

Statement about the JFLAP teaching software regarding the Needs Premier Award 2006.

Valerie J. Harvey, RT(R) PhD
Professor, C&IS
Computer & Information Systems
Robert Morris University

INFS3450 – Quantitative Analysis for Information Systems Professionals. This is a discrete mathematics applications course designed by an interdisciplinary committee representing information systems, computer science, mathematics, mathematics education, and software engineering. Two of the faculty involved in the design of the curriculum and materials have engineering degrees (Peter Wu and Sushil Acharya).

I have been using JFLAP since 2003. This is clearly one of my most valuable teaching tools for discrete mathematics

Usage of JFLAP is required (although there are optional assignments). I think the experience with JFLAP is far too valuable to leave it to chance. However there is ample evidence that the students utilize the software beyond the requirement (1) for practice, (2) for review, and (3) to explore optional capabilities of the software.
A demonstration of the required assignment and one or more optional exercises is given in lecture. In additional to documentation internal to the software, which is fully adequate, documentation of the required assignment, which is quite simple, is provided to the students online (http://home.earthlink.net/~inforef/i3450fsa.htm) and in hard copy format. Additional documentation is furnished by a course textbook: Richard Johnsonbaugh, *Discrete Mathematics*, 5th ed. (Prentice Hall, 2001) in the following sections: §10.1 Sequential machines and Finite-State Machines, §10.2 Finite State Automata, §10.3 Languages and grammar, §10.4 Nondeterministic Finite-State Automata, and §10.5 Relationships between Languages and Automata.. With permission, this documentation is also included in the second edition of our custom textbook, Valerie J. Harvey, Sushil Acharya, E. Gregory Holdan, Mark M. Maxwell, David F. Wood, Peter Wu, eds., *Discrete Mathematics Applications for Information Systems Professionals, 2nd ed.* (Pearson, 2006), in press, Pearson/Prentice Hall, as a supplement to Richard Johnsonbaugh, *Discrete Mathematics*, 6th ed. (Prentice Hall, 2006).
Students give demonstrations when presenting projects involving JFLAP.
There are both required assignments and optional assignments. In the required assignment, student must deliver a print or screen-print showing trace and the finite state automaton (FSA) developed using JFLAP.

I use JFLAP with the following topics:

Finite State Automata (required for all INFS3450 students). Students must understand state transition conceptually as a fundamental building block of software and as a practical matter must be adept at interpreting documentation of process and thread management in operating systems. Among the applications and extensions to which students are guided in this regard: Finite state diagrams in documenting operating system concepts: see William Stallings, *Operating Systems, Internals and Design Principles*, 4th

ed. (Prentice Hall, 2001); Petri nets in enterprise process modeling: Nikunj P. Dalal, Manjunath Kamath, William J. Kolarik, and Eswar Sivaraman, "Toward an Integrated Framework for Modeling Enterprise Processes," *CACM* 47, 3 (March 2004): 83-87. Linear-Order String matching Algorithms in Data Mining - Sushita Mitra and Tinku Acharya, *Data Mining: Multimedia, Soft Computing, and Bioinformatics* (Wiley, 2003), §4.2, pp. 150ff. String Matching with finite automata, §4.3 String matching in Bioinformatics.

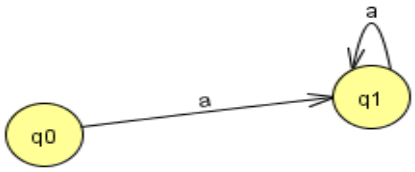
- Regular Expressions (required for RMU students doing project presentations on pattern matching and regular expressions where they complete an assignment in JFLAP as well as practical interactive demonstrations using *grep* (Unix/Linux/Xen) and/or ISO standard M pattern match syntax (used primarily in health care software and stock market and international financial transactions software). Capability with *grep* is required for use of Unix/Linux/Xen in all RMU operating systems courses, in RMU graduate networking courses for routing information protocol assignments and projects, and for intrusion detection assignments (to review contents of system logs looking for evidence of intrusion "exploits") for RMU graduate courses in information security and assurance. Capability with M (or MUMPS or InterSystems Caché ObjectScript) pattern match syntax is required for RMU students developing health care system (such as radiology system) demonstrations in our undergraduate health care track or students working with VistA/WorldVistA or OpenVistA hospital information systems or electronic health record (EHR) systems technology. JFLAP is a vital interactive conceptual practice environment and provides a general foundation for all of these practical applications of regular expressions.
- L-Systems (required for students doing project presentations on L-Systems and turtle graphics and their applications) L-systems experience is important for RMU students with an interest in bioinformatics. External documentation used with this includes Chris Lucas, "Classifier, IFS [Iterated Function Systems], L-Systems and Beyond," at http://www.calresco.org/lucas/classify.htm ; Lindenmeyer Systems and biological growth simulation, Texas A&M University, at http://acadia2000.tamu.edu/exhibit/DME/2000_49/moss1.htm
- Languages and grammars (optional). This topic is important so that students understand formal documentation of computing languages (BNF).

JFLAP is also used to develop finite automata for quiz and exam questions which the students might encounter, including on the final examination for INFS3450, where they could be required to devise strings that (1) would be accepted or (2) would not be accepted by a particular finite state automaton or (3) are asked to judge whether a particular string would or would not be accepted. A sample question of this type is provided online for students.

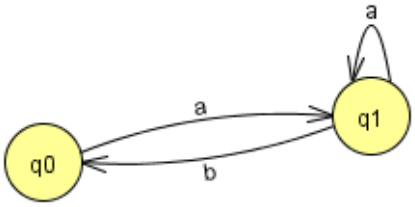
JFLAP is useful in teaching because of its applicability as described above and:

- it is easy to learn
- the graphic representation is excellent and approximates the examples seen in textbooks. It is very important that teaching software have this quality. For example, note that in developing a finite automaton, if there are transitions in both directions between the same two states, this is adjusted automatically in the

JFLAP graphics without explicit student action, so that a professional-looking result is achieved. Step 1, only a:q_0 -> q_1 and a:q_1 -> q_1 are shown:



Step 2: Student adds b:q_1 -> q_0 with the automatic result:



- Feedback is excellent in testing input strings (colors to show acceptance and non-acceptance of strings and traces.

Students are pleased to be able to develop professional-looking results in their work.

Perhaps most important is the integrated treatment of languages, grammars, and finite automata in JFLAP. One the most important experiences of discrete mathematics in the computing disciplines or for software engineering is to learn to recognize **alternative and equivalent representations** "of the same thing." Using JFLAP it is as easy to deal with equivalent representations as it is to see equivalent representations of relational database queries in SQL and QBE (Query by Example)-based tabular graphic format in Microsoft Access.

It is possible to use JFLAP equally well for introductory or advanced learning.

I can state that students like JFLAP software because they often do more elaborate exercises or more exercises than I assign, and because they report using it in the review process toward the end of the semester. Every semester I ask students which software they have found most useful in review and JFLAP is invariably cited. Because the graphic representations of the automata are self-documenting, they are easily used in correcting errors and helping the student understand concepts. All incorrect submissions must be corrected and resubmitted. I don't observe frustration in this resubmission process.

I have cited and/or demonstrated JFLAP in the following presentations or papers:

- "Workshop Report: Practical Examples for Teaching Discrete Mathematics in an Information Systems Curriculum," AMCIS 2005, available online as part of SIGCSE 2006 Discrete Math Workshop materials at

http://blue.butler.edu/~phenders/sigcse2006/activitiesworkshop/amcisa2c-1.doc Co-authors/presenters, Peter Y. Wu and John C. Turchek.

- "Coordinated Topic Presentations in Information Systems Core Curriculum and Discrete Mathematics Courses," presented at *ISECON 2005 in Columbus, OH, and published in ISECON* 2005 Proceedings. Co-presenters: Peter Wu and John Turchek. This has been accepted for publication in ISEDJ.
- "Insights from Teaching Discrete Mathematics in Information Systems Programs," Report for the Discussion Forum, CoLogNet/Formal Methods Europe Symposium on Teaching Formal Methods (TFM'04), November 19, 2004, Ghent, Belgium. Co-author with E. Gregory Holdan
- "Workshop on Discrete Mathematics for Programs Conforming to ABET Information Systems Accreditation," Co-presenters: Peter Y. Wu and John C. Turchek, ISECON, November 4, 2004, Newport, RI

Teaching of automata was cited in:

- "Life-long Learning: Making Discrete Math Relevant for Information Systems Professionals," IACIS 2005, available online at http://www.iacis.org/iis/2005_IIS/PDFs/Wood_Harvey_Kohun.pdf David F. Wood, Valerie J. Harvey, and Frederick G. Kohun.

I particularly respect the instructional usefulness of JFLAP because I taught automata theory in computer science in 1988 when I didn't have such a tool.

June 14, 2006

National Engineering Education Delivery System
Premier Award 2006 Nominations

Evaluation Committee Members,

I only became aware of the JFLAP software package in April, 2006. My spring semester course (CS 154, Formal Languages and Computability) was already almost 2/3 finished, so it was too late for me to thoroughly integrate the software into the course. However, I liked the software so much that I felt I at least needed to demonstrate it to my class. At that point, we had already completed our coverage of Finite Automata and Pushdown Automata, but were still covering Turing Machines.

Beyond showing students the software, and including a link to it on my website, I didn't require students to use it in any way. Nevertheless, about 75% of students used JFLAP for (at least) the next homework, in which they needed to give Turing Machines for two different functions.

I got only positive feedback from students, some of whom went back to study previously covered topics using JFLAP. Several commented that they found it quite fun to use, and two students (one of whom wasn't even in my class, but heard about it from the other) have expressed interest in helping to modify the software itself after having looked at the source code. These two students are from our student Linux Users Group, and both were attracted by the software engineering aspect of the project.

I was so impressed by the software that I wrote to Professor Rodger to compliment it, and to get textbook recommendations from her. To quote the first paragraph of that email:

> First of all, my compliments on the JFLAP system. It is such a nice tool that I hope it will provide me a much needed kick-in-the-pants to reorganize my automata/formal languages course. I had been thinking about how much work it would be to create such a system, but thankfully, you guys have already done the work for us.

To summarize, although I have just started using JFLAP, I am so impressed with it that I plan to completely reorganize my course next year in order to properly incorporate the software into it. I think that it helps students not only to learn the material, but to get excited about it.

Best regards,

David Scot Taylor
Assistant Professor
Department of Computer Science
San José State University
One Washington Square
San José, CA 95192-0249
taylor@cs.sjsu.edu

Franklin W. Olin College of Engineering

28 June 2006

Dear Susan,

I am pleased to write this letter in support of your JFLAP program. I've used JFLAP in my course (ENGR 3520, Foundations of Compuer Science) for several years now and find it to be the most compelling piece of software for the students of any that we use. I use it to show students how automata work, and they invariably choose to use it to produce their problem sets. JFLAP is designed to be easy to use and to bring to life a topic that is often difficult for the students to really experience; it makes automata theory into a hands-on learning experience! And, as all educators know, *doing* something is an excellent way to reinforce learning. It's just that we computer scientists sometimes find it hard to actually build things in the way that our more mechanically inclined colleagues do….With JFLAP, our students can have this kind of experience too. In fact, last year my TA was so taken with JFLAP that he wrote a little sprite animation program that takes JFLAP's XML output as its control program, so now my students can build finite state automata and watch them walk little creatures around the screen!

Thanks for making such a great tool available!

Sincerely,

Lynn Andrea Stein

Professor of Computer and Cognitive Science

las@olin.edu

June 10, 2006

To Whom It May Concern,

I am an Assistant Professor in the Department of Mathematics and Computer Science at the University of Richmond in Richmond, Virginia. I used JFLAP this fall when I taught the Theory of Computation course (CMSC 330) for the first time. I chose to use JFLAP in this course because I believe it enables computer science students to more easily grasp the very theoretical material presented in the course. For those students who are less mathematically inclined, JFLAP helps them understand the abstract ideas from lecture by allowing them to create, modify, and test the different types of automata, observing how the algorithms work in a step-by-step fashion. All of the students, including those with the greatest theoretical ability, better understand the subtleties of the course material because they are able to use JFLAP to effectively debug their understanding. When the students are unsure of some aspect of the course material, they can simply construct a corresponding example and observe how JFLAP handles that test case.

In this course, I required JFLAP to be used in the lab assignments and permitted students to use it on their homework assignments. (The course has a lab component, and the students are graded on their work.) The assignments over the course of the semester covered a large portion of JFLAP including finite state machines, pushdown automata, grammars, and Turing machines. The feedback I received from the students was positive. I found it extremely enjoyable to watch them use JFLAP in order to clarify their understanding of the algorithms.

I believe JFLAP is a fabulous tool which makes it easier to teach a wide variety of computer science students the material in this course. I look forward to using it again in lab the next time I teach this course, and I also intend to use it in my lectures when I illustrate the theoretical concepts with examples.

Sincerely,
Kelly A. Shaw

Sincerely,
Kelly A. Shaw

To Whom It May Concern:

I would like to recommend the JFLAP software for consideration for the Needs Premier Award. I am an Assistant Professor of Computer Science at Elon University in Elon, NC. I have been using JFLAP for three years in my Theory of Computation course, CSC 351.

The theory course is required for all Computer Science majors and is generally taken in the spring of either their junior or senior years. Math majors are also eligible for the class and often are drawn to the theoretical aspect of computing. The theory class is different from every other class in the Computer Science curriculum in that there is no programming component (the students write automata, of course, but do not write any programs in high-level languages.) This is generally quite worrisome to the students at first, as most of our Computer Science students do not rank Math classes as their favorite or best classes. The use of software in a theoretical, non-programming class is like a safety blanket for a Computing student.

I explain all of this in order to highlight the part that the JFLAP software plays in the class. JFLAP allows the students to create automata using an easy, point-and-click interface, and then to check their work by working through example strings. The software allows them to visualize the process defined by the automata. I use it routinely in class when introducing new concepts or working through examples with the students. JFLAP is an excellent tool when the student originally proposes an incorrect answer. Their proposed automata can be created and tested in the software, and the student can see exactly where the problem is. Also, the student can fix the automata, and the new solution can be checked. Without the software, this testing and debugging phase would be too lengthy and abstract to do in class.

When I began teaching the class, I would show a power point presentation to begin a new topic. We would first discuss the properties of an automaton, and then we would use JFLAP to do examples. I found that the students were not engaged in the first half of the class, and it was also clear to me that they did not really understand the topic until they saw the JFLAP visualization. Now I introduce the automaton through JFLAP first. I use the software for modeling deterministic and non-deterministic automata and for creating a deterministic automaton from a non-deterministic one. I also use the software for pushdown automata and Turing machines. There are several topics that are included in the JFLAP software that I do not use. This is not because of the software, but rather because the other topics are beyond the scope of my class.

The students routinely use JFLAP on their own as well. They are required to use JFLAP for some of their homework, but not all. The students are told that those who use JFLAP to develop and test their work generally make higher grades than those who do not. This is indeed true, and the reason is simply the ease with which they can find and fix any mistakes in the work. For this reason, students routinely use JFLAP even on assignments that do not require it. The students really enjoy using the software, and I believe they enjoy the class as a whole more as well. Each year at least one senior (Computer Science senior, that is) chooses the theory class as his favorite class in the entire curriculum,

which includes classes such as Artificial Intelligence, Graphics and Video Game Programming, and our popular Senior Seminar.

In conclusion, I highly recommend the JFLAP software. I have never had the software crash, nor have I ever found a bug to report. The software is extremely easy to install and use, and the improvement in the class is immeasurable. It easily makes the class more engaged and interested. I believe that many students would never understand the material if they couldn't "see" it. I truly wish I had similar visualization software for all of my classes.

Sincerely,

Shannon Pollard

Asst Professor of Computer Science
Box 2189
Elon University
Elon, NC 27244

UNIVERSITY of WISCONSIN - EAU CLAIRE
Office of the Provost and Vice Chancellor• 105 Garfield Ave, Eau Claire, WI 54702 • (715) 836-2320

June 16, 2006

Dear Dr. Rodger:

Please accept this letter of support for your JFLAP software tool. For the past 18 years I have taught senior level computing courses on the theory of computation, first at the United States Naval Academy, and now at the University of Wisconsin – Eau Claire. While I now teach must less often in computer science because of my administrative responsibilities, I continue to teach – once per year – our Computer Science Department course entitled "Theory of Computation" (CS450). This is my main teaching obligation these days, and I am the only one on our campus to have ever taught this class.

Up until about three years ago, I had repeatedly bemoaned the lack of any quality tool for simulating finite state machines, pushdown automata, and Turing machines. I had engaged students in projects to create these tools for me to use in my classes, and while they did do that, the tools never quite made an impact or caught on with the students in my classes. The tools always lacked some key functionality, were not quite "professional" enough, or were cumbersome to use. In short, those efforts were valiant, but ineffective.

I recall hearing about JFLAP and thinking, "Oh sure, someone else tried what I did. I'll bet they failed, too." Of course, I was wrong about that latter part. I have now been using JFLAP in my course for three years and with great success right from the start. I spend NO class time discussing the tool. I give the students a pointer to the software on our system, and tell them they must use it for their assigned homework. The rest is up to them. Of course, the reason I can get away with that approach is that the interface and "usability" of the tool is simple and convenient. One rarely needs to read the online help; the use of the product is intuitive, the interface is clean, and the design is well planned, in spite of the fact that it is noncommercial.

The students like JFLAP, not so much because they like to simulate the machines to see them work (maybe they do – I can't be sure), but because they can design the devices, test them on sample inputs, redesign, and retest, all very quickly. The tool enables them to work fast and accurately, correctly mistakes and adding functionality. We use JFLAP for finite state machines, for pushdown automata, and for Turing machines. But I have also seen students experiment with some of the other models in JFLAP, simply out of curiosity. That's something that would never have occurred without the tool.

I want to make two special observations that I think really point out how useful this tool has become. I recently taught a course on Digital System Design (CS 278) that has a decidedly practical side involving the construction of digital systems implemented on VLSI system prototype boards. Some of my students in that class had previously taken my theory class and used JFLAP. While I never even mentioned JFLAP in the Digital System Design class, I

discovered that the students were using it to create the finite state controllers for their digital devices. They did all of their simulation and design work in JFLAP before implementing their work on the real system boards (in Verilog). The point here is that the students clearly recognized, without any prodding from me, that JFLAP was a valuable tool that extended beyond the course in "theory." It was a practical tool that they could apply to engineering design work in other areas. This is an instructor's dream – to have students see the link between theory and practice and do it without you pointing it out. The JFLAP tool was the "vehicle" for making that connection.

Second, I have also discovered that students are now using JFLAP as a "drawing tool" (I'll bet you didn't envision that in the beginning) for designing controllers that they want to embed as figures in technical papers that we are developing. While students unfamiliar with JFLAP fight with MS Word or MS Visio to create such diagrams, the students who have used JFLAP are quickly building the figures in JFLAP and then embedding those figures in our papers.

Obviously, I am a big fan of JFLAP, and so are my students. JFLAP is one of the highest quality, freeware packages that I know of. I'm always eager to hear about enhancements, and I intend to continue to use it in my classes. It's simply a great product.

Sincerely,

Andrew T. Phillips
Professor of Computer Science
Associate Vice Chancellor for Academic Affairs and Dean of Graduate Studies
University of Wisconsin – Eau Claire

EMORY
COLLEGE

Department of Mathematics and Computer Science

Michelangelo Grigni, Associate Professor
Department of Mathematics and Computer Science
Emory University
Atlanta, GA 30322

June 29, 2006

Susan H. Rodger
Dept. of Computer Science, Box 90129
LSRC, Room D237
Duke University
Durham NC 27708-0129

Professor Rodger,

I am writing this letter in support of JFLAP in the "Needs Premier Award" competition.

Since 1999, I have used the JFLAP tool to help teach automata theory concepts to undergraduates (and some graduates) at Emory University. It is an excellent tool for both the design and simulation of finite automata, grammars, and Turing machines. I have observed that it helps in at several kinds of learning situations:

1. During lectures or lab demonstrations, the "live" graphical presentation of a working automaton speeds student comprehension of their behavior, compared to a blackboard demonstation. This is true both during the demonstration of a prepared automaton (a didactic, introductory situation), and also during participatory design or debugging of a novel automaton (a more interactive, advanced situation).
2. In assigned work where the students are asked to design an automaton with some prescribed behavior, students using JFLAP (rather than just paper), I find that they are much more likely to produce a well-formed automaton, and furthermore they are also more likely to check its correct behavior on test inputs.
3. In group work (either group assignments, or student presentations) I find that having the common interactive tool greatly improves the ability of students to present and study each others designs. It certainly goes much better than a student blackboard presentation, especially if there is some mistake to correct.
4. During review or office hours, students have an easier time going over examples from lectures and homework, if those examples were done with JFLAP (and I've made the relevant files available).

Emory University
Mathematics and Science Center
400 Dowman Drive
Atlanta, Georgia 30322

Tel 404.727.7580
Fax 404.727.5611

The one exceptional situation is testing: since my main concern is their comprehension, rather than their JFLAP usage skills, my students do not use JFLAP on exams. However, I might still prepare exam figures with JFLAP; they have no trouble with this, since the JFLAP figures are quite close to standard textbook diagrams.

Similar remarks also apply with "grammar" or "Turing machine" in place of "automaton", and with non-deterministic variants of these models.

More specifically, I use JFLAP in these two Computer Science courses:

CS224: This is a our sophomore level introduction to discrete mathematics, the relevant module concerns finite automata. In my most recent edition of this course (Fall 2005) I used JFLAP for in-class demonstrations and design, and made it optional for assignments. I find that most of these students prefer JFLAP, the only issue being lab access (once they know how to use the lab machines, JFLAP itself is quite easy to use).

CS424: This is a senior level introduction to computability and complexity theory, the relevant module is Turing machines. In my most recent edition of this course (Spring 2006) I used JFLAP for in-class demonstration, and also required its use for a homework and for student presentations.

JFLAP could also be used in a compilers course just to introduce grammars and parsing, but I do not usually teach that.

Teaching these subjects just on paper, without a design and simulation tool like JFLAP, would be analogous to teaching programming without ever actually running a program! We could still do a good job with the abstractions, and get by with "by-hand simulations", but the practical and intuitive understanding of most students would suffer. JFLAP provides a design-and-test environment which is familiar to students of programming. By reducing the time necessary to work through examples, JFLAP also gives us more time to study the more abstraction concepts.

In summary, I strongly recommend JFLAP as a teaching tool for these subjects.

Sincerely,

Michelangelo Grigni
Email: mic@mathcs.emory.edu

From: Robert Sloan <sloan@uic.edu>
To: Susan Rodger <rodger@cs.duke.edu>
Date: Wed, 7 Jun 2006 11:32:47 -0500

To whom it may concern:

I am writing this letter to support the nomination of Susan Rodger's JFLAP software for the Needs Premier Award for courseware. I began to use the deterministic and nondeterministic finite automata portion of JFLAP for my course Computer Science 301, Languages and Automata in the fall of 2003. I require students to use JFLAP in the finite automata portion of the course; some students choose to use other parts of the software (push-down automata and Turing machines) later in the course.

The finite automata portion of my course could fairly be said to be centered around JFLAP. I use it in lecture to give examples and I assign homework problems that require its use. Students seem to really enjoy working with JFLAP, and it has clearly led them to a deeper understanding of nondeterminism, and of the algorithms for converting among the various representations of regular languages.

Incidentally, the reason that I use JFLAP so heavily in the beginning of my course and only incidentally in the end of the course is because the ideas that JFLAP promotes happen to be central to the beginning of the particular course I teach but only incidental to the material at the end of my course. If I were teaching a mildly different type of course, I might well use JFLAP throughout.

I could go on at some length about how wonderful I think JFLAP is. However, it might be more effective to say that it has helped me to win a major teaching award at my university, and to quote from those materials. The following is from my required letter when I won recognition from the University of Illinois at Chicago Teaching Recognition Program (a major award, carrying a $1,500 permanent salary increase, given each year to roughly 10 of our roughly 2,000 faculty members):

"One reason that my students have an excellent experience is not that I have hundreds of great teaching ideas, but rather that I have a few great teaching ideas, and a very good eye. I am always on the look out for good ideas of others. Many of my most successful classes have components that have come from hither and yon.

Let me give you some examples: (1) I regularly teach automata theory, a very abstract mathematical course. Since 2002-2003 I have used a visualization system (JFLAP) developed at Duke University that probably took about 10-15 person months to build and polish. My students now understand nondeterminism in a way that they never have before."

In conclusion, I strongly support the nomination of JFLAP for this courseware award. Fell free to contact me if you need any other information.

Sincerely,

Robert H. Sloan

Professor and Director of Graduate Studies
Department of Computer Science
University of Illinois at Chicago
sloan@uic.edu
(312) 996-2369

As a faculty member in the Department of Computer Science at UC Davis, I regularly teach our undergraduate course on formal languages and automata. I have published a text on this subject, “An Introduction to Formal Languages and Automata” with Jones and Bartlett, that is now in its fourth edition. A few years ago I was introduced to Susan Rodger’s JFLAP and, in spite of my initial reluctance for using software in this course, I quickly became an enthusiastic supporter. Since then I have been in frequent contact with Professor Rodger, have made suggestions for additions to JFLAP, and have participated in several workshops on it. In my opinion, JFLAP is a great teaching tool.

My students have benefited from JFLAP in several ways. It allows them to do their homework more quickly and more accurately. Constructing and testing automata, such as pushdown automata and Turing machines, is tedious and error-prone. JFLAP speeds up this process and makes finding and modifying errors a simple matter. Also, the ability to follow constructions step-by-step is a great aid in understanding some of the more difficult concepts. I am particularly impressed by the way in which JFLAP illustrates the issue of nondeterminism. Finally, JFLAP has improved my students’ attitude towards the course. Working with JFLAP is one aspect that most of them enjoy. In the most recent course evaluation, comments like “JFLAP is very useful”, “JFLAP is awesome”, and “JFLAP is a truly excellent resource” , were common.

JFLAP has also helped me as an instructor. With JFLAP I am able to give exercises that I had previously avoided, such as assigning problems that require explicit construction of complicated pushdown automata, Turing machines, or unrestricted grammars. I can give partial or incorrect solutions to a problem and ask that students complete or correct them, or make them analyze complex structures, such as context-sensitive grammars. Assignments of this kind are of course also possible without JFLAP, but they are much more time-consuming.

We also now use JFLAP to correct homework and in-class examinations. This saves a lot of time and increases grading accuracy considerably.

In summary, I consider JFLAP a great benefit to students and to any instructor who teaches a course in formal languages and automata.

Peter Linz
Professor Emeritus
University of California at Davis

June 30, 2006

Dr. Susan Rodger
Associate Professor
Duke University
Durham, NC

Dr. Rodger:

I used JFLAP for the first time during the spring semester of the 2005-2006 academic year for CSC 369 Theory of Computation. The students had the option to use JFLAP for some of their assignments and most chose to use JFLAP.

I introduced JFLAP after having lectured about finite automata. Most of the students did not seem to fully grasp the concepts of these theoretical machines and my construction of simple deterministic finite automata on the blackboard did not make much of an improvement of their understanding. After introducing JFLAP to them and having them learn to construct the fa's seemed to make it clearer to them. It immediately made it easier for the students to understand how the machines work.

I will be teaching the course again during the spring semester of 2006-2007. I intend to introduce JFLAP much earlier in the semester with plans to cover more material with the help of JFLAP.

Jacquelyn E. Long
Assistant Professor
Computer Science Department
Norfolk State University
Norfolk, VA 23504

21 June 2006

2006 NEEDS Competition
Evaluation Committee

XAVIER UNIVERSITY

Department of Mathematics
and Computer Science
3800 Victory Parkway
Cincinnati, Ohio 45207-4441
Phone 513 745-2882
Fax 513 745-3272

To the evaluators:

Xavier University uses Susan Rodger's JFLAP tool in our sophomore-level CSCI 250 Languages and Automata course. We have been using it since Spring 2000. It is used as an optional tool for the students and for classroom demonstrations. We use it most often when discussing finite automatons, because the tool makes it easy to quickly build and get a visual demonstration of the automaton, and also because if facilitates an easier understanding of non-deterministic automata vs. deterministic automata. We generally see a significant subgroup of students use the tool to produce their homework solutions, which we believe means they are also using it as they determine the solutions. Anecdotal evidence from the students is that those who use it appreciate the visual and easy interactive nature of the tool. We certainly believe the tool is useful for learning and when I teach the class I encourage students to use it as a way to explore the material.

While I have not used JFLAP for my own research, I suggested its use to a colleague, Matt Jadud, who was looking for a tool to represent automata graphically as a part of his dissertation research at the University of Kent. He tells me it provided precisely the functionality he needed to express his work.

My overall impression of JFLAP is that it is an excellent tool. Since I generally teach the CSCI 250 course only every other year or so, I have not had a chance to work with it recently, hence my opinions now are based on an older version than is probably current (Michael Goldweber also uses it when he teaches the course). I downloaded and started using the tool after attending Dr. Rodger's talk at a SIGCSE conference. To me, the real power of the tool is that you can build and test your ideas, much the same as students are used to building their programs. For sophomores, that hook is really nice as they wade into material that is very different from what they consider to be typical computer science work. When colleagues from other schools talk to me about teaching the Languages and Automata course, I cannot always recommend an excellent textbook, but I always recommend JFLAP as useful courseware.

I hope this letter is helpful in the evaluation process.

Sincerely,

Gary Lewandowski
Professor, Mathematics and Computer Science

June 15th, 2006

To whom it may concern,

I am writing this letter in support of the software JFLAP receiving the Needs Premier Award in 2006 for high-quality noncommercial courseware. I have used JFLAP during three semester-long CS 310 Theoretical Computer Science course offerings (Fall 2002, Fall 2004, Spring 2005) at Pacific University and found it very nearly indispensable. While I have required that JFLAP be used in each of these course offerings, during the Fall 2005 offering we used it extensively in the course. I demoed the software during class time, and using it was a crucial part of two of the six assignments in the course. In particular, the students used JFLAP to do the following:

1) Construct DFA's, NFA's that accepted the appropriate languages.
2) Experiment with CFG's and specifically LL(1) parse tables in a way that required that they understand deeply the notions of First and Follow.
3) Construct two different Turing Machines that would perform binary addition and subtraction.

Since I have been using JFLAP in CS 310, the professor of our Compilers course (CS 480, for which the Theoretical Computer Science course is a prerequisite) has commented to me how much more prepared the students are for his course. I have learned that there is no substitute for the ability of my students to see a DFA, NFA, CFG, or Turing Machine in action, and I would regard JFLAP now as such an integral part of the course that I cannot imagine teaching the course without it. The students, also, have found the software extremely helpful and easy to use, and we all appreciate the fact that is runs on any platform. Additionally, I have also used JFLAP in a seminar series whose intent was to teach non computer science faculty about the development of Turing Machines, and found the software extremely useful for conveying these ideas to non-specialists.

I should also add that any questions I have had regarding the software have been answered very quickly and in depth by either Susan Rodger or another member of the development team. They have clearly spent very much time developing this software with only the goal of student learning in mind, and I would very much like to seem them recognized for their hard work.

Thanks!

Chris Lane, Assistant Professor
Department of Mathematics and Computer Science
Pacific University
Forest Grove, OR 97116

Tennessee Tech
UNIVERSITY

Department of Computer Science
Box 5101 • Cookeville, TN 38505-0001 • (931) 372-3691 • Fax (931) 372-3686
http://www.csc.tntech.edu • info@csc.tntech.edu

June 5, 2006

NEEDS Premier Award Competition
NEEDS Headquarters
2111 ABC Etcheverry Hall
Berkeley, CA 94720-1740

To Whom It May Concern:

Dr. Susan Rodger, associate professor of the practice of computer science at Duke University, is entering the software package JFLAP (Java Formal Languages and Automata Package) that she and her students have developed and maintain in the 2006 NEEDS Premier Award Competition. I am very happy to write a letter in support of her application.

I have been using JFLAP since approximately 1999. I have used it in two courses that I have taught: Computer Science 2710 (Foundations of Computer Science) and Computer Science 4450 (Introduction to Automata Theory and Computation). Computer Science 2710 is a required course for our students, and Computer Science 4450 is an elective course with Computer Science 2710 as a prerequisite. I use JFLAP in the classroom when covering finite automata (both deterministic and nondeterministic and the conversions between them), pushdown automata, Turing machines, and context-free grammars. We design machines in class, and I assign students homework problems in which they are required to use JFLAP. Students are active participants in class when JFLAP is used. They have found mistakes in machines while we were designing them, and some have developed elegant alternative solutions, which they emailed to me during or after class. JFLAP is an excellent learning tool for theoretical concepts in computer science, and my students enjoy using it.

Dr. Rodger and her students have constantly strived to improve JFLAP through the years, adding new features while maintaining ease of use for the system. I am looking forward to using the new version with the Turing Machine building blocks this fall when I teach Computer Science 4450.

Thank you very much for considering JFLAP in this year's competition. I think that JFLAP is very deserving of the NEEDS Premier Award.

Sincerely,

Martha J. Kosa

Martha J. Kosa, Ph.D.
Associate Professor of Computer Science

UNIVERSITY OF CALIFORNIA, IRVINE

BERKELEY • DAVIS • IRVINE • LOS ANGELES • MERCED • RIVERSIDE • SAN DIEGO • SAN FRANCISCO SANTA BARBARA • SANTA CRUZ

DONALD BREN SCHOOL OF
INFORMATION AND COMPUTER SCIENCES

IRVINE, CALIFORNIA 92697–3440

DAVID G. KAY
LECTURER S.O.E. IN COMPUTER SCIENCE
DIRECTOR OF INTRODUCTORY PROGRAMS

TELEPHONE: (949) 824-5072
TELECOPIER: (949) 824-4056
ELECTRONIC MAIL: kay@uci.edu
WORLD-WIDE WEB: http://www.ics.uci.edu/~kay/

June 14, 2006

Professor Susan Rodger
Department of Computer Science
Duke University
Durham, NC 27708-0129

Dear Professor Rodger:

I would like to express my strong support and appreciation of your tool JFLAP.

My use of JFLAP may be a little unconventional, which speaks well of its flexibility and broad appeal. I teach second-quarter computer science, sometimes in honors sections, for majors in computer science and informatics. Our curriculum includes most of the traditional CS 2 topics (leaving sorting, balanced trees, and some advanced data structures for the third quarter). It also includes about three lectures (15% of the class meetings) on automata and formal languages, not primarily from the theoretical perspective but as a tool in programming certain kinds of tasks.

In some versions of the course, students do an early lab that requires them to tokenize the input stream into words. When they see later how to draw an FSA for the task and covert that trivially to code, there's a gasp of recognition that the right tool can make a big difference. As part of their lab work, students build and test a couple of simple FSAs in JFLAP. I find that they appreciate being able to run and test their designs. More generally, it's valuable for them to have a runnable tool that provides a clean, clear, and correct operational model to serve as a check on their understanding.

I strongly encourage you to continue your development and dissemination of JFLAP. It is a valuable tool of the highest quality.

Sincerely yours,

David G. Kay

Department of Mathematics and computer Science
Fayetteville State University
1200 Murchison Road
Fayetteville, NC 28301-4298

June 26, 2006

Dear Susan,

I have used JFLAP for three years in my course titled **Theory of Computation** (CSC 332) at Fayetteville State University. This course is typically offered during the spring semester, and the class size is approximately ten students. CSC 332 is not required for the major, and the prerequisites for the course are two semesters of discrete mathematics and the data structures course. My students are required to use it for eight assignments that they must submit electronically. Each assignment consists of one to three parts depending upon the difficulty of each part. I grade them by testing each submitted problem against a predetermined list (usually ten) of input strings. My JFLAP assignments cover the following topics: DFA, NFA, regular expressions, CFG, PDA, NPDA, and Turing machines (both single tape and multi-tape machines, acceptors and transducers).

Since classroom technology allows me to use JFLAP in the classroom, I use it during most of my classes. I have not used it for the pumping lemmas or complexity, although I did demonstrate some busy beavers that were part of an exercise in the textbook.

The students like it, and we believe it helps understanding for two reasons. (1) Students can get immediate feedback on their attempts to build a machine or construct a grammar. They can test what they have constructed to see if it is correct, and modify it if necessary. (2) Seeing the states change in the step-by-step mode is in my opinion a wonderful opportunity for students to understand how these things work. After all, “a picture is worth a thousand words”, and JFLAP provides many pictures.

I have much praise for JFLAP. It is relatively easy to use, yet it is a powerful tool to help students learn about computing theory. It is also easy to use in grading assignments since it can run multiple inputs strings at once. Finally, if I were still teaching the compiler course, I would certainly use JFLAP there to help students understand parsing

V. Dwight House
Associate Professor
Department of Mathematics and Computer Science

VIRGINIA STATE UNIVERSITY
Petersburg, Virginia 23806

SCHOOL OF ENGINEERING, SCIENCE AND TECHNOLOGY

MATHEMATICS & COMPUTER SCIENCE DEPARTMENT
P. O. Box 9068

PHONE (804) 524-5920/ 5553
FAX (804) 524-5746
TDD (804) 524-5487

June 12, 2006

Dr. Susan H. Rodger
Associate Professor of the Practice
Computer Science Department
P. O. Box 90129
Duke University
Durham NC 27708-0129

Dear Dr. Rodger: My name is Dawit Haile and I am an Associate Professor & Chair of the Department of Mathematics and Computer Science at Virginia State University, Petersburg, VA. After attending the JFLAP Faculty Adopter Workshop in June 2005, I adopted JFLAP in my Discrete Structures (CSCI 281) class during Spring 2006 semester. The software was required for students' use in one chapter of the course that dealt with *Modeling Computation*. The software was also used by students when completing their homework problems. I used the software during lecture class while discussing *Languages & Grammars*, *Finite-State Machines*, *Language Recognition*, and *Regular Expressions*. As a supplemental material, we covered the first four chapters from the draft version of the text on JFLAP – *An Interactive Formal Languages and Automata Tool* by S. Rodger and T. Finley.

I found JFLAP to be an easy-to-use visualizing tool that does help students to quickly grasp the concepts of formal languages. It helped my students to understand the material presented in class by enabling them to create and operate on automata, grammars and regular expressions. At several occasions most of my students in this course informed me that they really enjoyed working with the JFLAP software. Hopefully, the draft document will be completed and be available for students use in the next academic year.

Sincerely,

Dawit Haile

Dr. Dawit Haile, Chair
Department of Mathematics & Computer Science
P. O. Box 9068
Virginia State University
dhaile@vsu.edu

THE UNIVERSITY OF UTAH

School of Computing

50 S. Centeral Campus Drive RM 3190 Salt Lake City, Utah 84112 (801) 581-8224 FAX (801) 581-5843

June 7, 2006

Letter in support of JFLAP, a Software Tool for Enhancing Education of Computability and Logic, the "Engineering Mathematics" of Computations

Dear Awards Committee:

I have taught Automata Theory, Formal Methods, and Computability several times over my 20-year career at the University of Utah, School of Computing. I consider these topics to be fundamental to "Computation Engineering" - a term I prefer over the more traditional names such as "Computer Science" or "Computer Engineering." This letter is my very strong endorsement for JFLAP - a tool that greatly assists the learning of these topics. Before I elaborate on JFLAP, I would like to provide some context and motivation.

We, in the 21st century, are faced with the phenomenon of computation in every walk of our lives. Jeanette Wing, Professor of Computing, and Department Head, Carnegie Mellon University, has recently written an article entitled "Computational Thinking" in the Communications of the ACM, March 2006. In this article, Professor Wing explains why computation is going to be a paradigm for living in the 21st century - and not just something esoteric that computers carry out. Everything from traffic gridlocks to scheduling meetings to biological phenomena fall under this umbrella. Analyzing these computational phenomena, manipulating them, and properly understanding them requires help from multiple foundational areas of mathematics. Automata Theory and Computability are two of the central branches involved in these studies.

Unfortunately, Automata Theory and Computability are in a state of "pedagogical anachronism." On one hand, these subjects were born in the mid 20th century when basic aspects of compilers were, for the first time, being understood. However the topic has now moved well beyond its initial role, permeating the construction and analysis of complex computing systems that might control factories, airplanes, and microprocessors to which human lives are tied to. With the exposition of this topic remaining "stuck" in the 1960s, with examples and motivations drawn from the world of compiler construction, and with students expected to write long series of derivations pertaining to deeply mathematical phenomena, the subject matter has, unfortunately, become one of the most dreaded. This, clearly, need not be so. With the right set of software tools used to bring to life various formal models of computers, students can "play" with these machines and develop strong intuitions pertaining to them. It is the experience of those who have taken this approach that even the mathematics becomes far more approachable, as a result.

Five years ago, I was seeking a tool that helps animate automata and other formal models of computers. After trying many tools, I used JFLAP and found that (i) it was intuitive, (ii) it offered powerful methods to explore automata (for instance, how non-deterministic automata can go through various configurations), and (iii) it was a robust and reasonably efficient tool with a good graphical front end. JFLAP was superior to any of the tools I used. It truly made a difference to my class in that the students could revisit the concepts after my lecture and even come up with their own intuitions about computation that were refreshingly reflected in their answers.

My own textbook entitled "Computation Engineering: Applied Automata Theory and Logic" has just been released (published by Springer). I employ many tools in my book just because I cover

many more formal aspects than traditionally covered in foundational courses. However, in the treatment of finite automata, and especially pertaining to Turing machines, I have employed JFLAP to illustrate numerous subtle details.

In conclusion, JFLAP eminently deserves the Needs Premier Award for tools that have truly made a difference to engineering education. It is widely used, and it opens the path towards many more students understanding foundational subjects pertaining to Computation Engineering.

Sincerely yours,

Ganesh Gopalakrishnan
Professor

William Turkett
Assistant Professor
Department of Computer Science
Wake Forest University (Winston-Salem, NC)

I have used JFLAP in a course in my department – the course was titled "CSC 231 – Programming Languages". It was used for the first time this previous spring semester (Spring 2006), but I anticipate using it again when I re-teach the course (which is likely to be every spring).

I used JFLAP in teaching introductory regular expressions and finite automata to CS majors (CSC 231 is a sophomore/junior level core course in our department). There were approximately 12 students enrolled in the course during this past spring semester. Use was mandatory during one of the lab sessions and the students could optionally use JFLAP on their homework related to regular expressions/finite automata (most students did use it). The software was demonstrated to the students by myself during the first half of the lab period before the students were given assignments using it.

I believe JFLAP was very useful in making regular expressions/finite automata understandable. The way in which students could interactively design finite automata and then test whether those machines accepted the right language was very useful for understanding how regular expressions and finite automata are related. In addition, the ability to watch each step of the process of a finite machine accepting or rejecting a word in a language and the ability to watch a step-by-step algorithmic minimization of a finite machine were very useful.

I believe my students liked the JFLAP software and found it quite useful. As stated above, almost all of the students voluntarily completed their homework assignments on regular expressions/finite automata using JFLAP. As a whole, we had a little trouble with some minor interface issues (mainly from habits picked up using Windows), but these were all minor issues and the benefits of using the program for learning far outweighed such details.

In summary, I really found JFLAP to be an asset in teaching regular expressions and finite automata in my course. I plan to use it again in future semesters, as well as plan to extend my use of it to incorporate the relevant components when I teach grammars and push-down-automata (as they are part of the same CSC 231 course). I looked at a handful of similar programs at the beginning of the semester (such as "The Regular Expression Coach") and the extensive capabilities of JFLAP placed it well above the other programs and motivated my choice to use JFLAP.

To whom it may concern:

I have used the JFLAP tool for the past two years in my Discrete Mathematics and Theory of Computation classes at the College of Wooster. Dr. Denise Byrnes, our other faculty member who also regularly teaches the Theory of Computation class, introduced me to it. I use the tool as a required utility in homework assignments, and I also value it for demonstration purposes. My primary emphasis for JFLAP involves finite automata, pushdown automata, and Turing machines, but Dr. Byrnes also has used it in various classes to demonstrate Lindenmeyer systems.

I have found JFLAP to be a versatile and effective teaching tool that students easily comprehend and appreciate. Because of its simple interface, students learn to use it effectively within a few minutes. It has substantially enhanced the effectiveness of my teaching in the classes where it applies.

Dale A. Brown
Professor of Computer Science
The College of Wooster
Wooster, OH 44691
dabrown@wooster.edu

UNIVERSITY OF KENTUCKY
COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

Dr. Judy Goldsmith
763E Anderson Hall
Lexington, Kentucky 40506-0046

(859) 257-4245
FAX: (859) 323-1971
goldsmit@cs.uky.edu

June 15th, 2006

Letter of Support: JFLAP

Dear Needs Premier Award 2006 Evaluators:

I teach automata theory and Turing machines at least once/year, often once/semester, either in the required undergraduate discrete math/models of computation sequence (CS 375) or the mixed undergrad/grad course, Theory of Computation (CS 575).

While I find that the students quickly grasp finite automata, they have more trouble with push-down automata and even more with Turing machines. In particular, it is tremendously helpful for them to actually run a Turing machine simulator. And even more so to play with the nondeterministic features of the JFLAP simulations.

I have been using JFLAP since I was introduced to it in 1998. I primarily use it as a supplement to whichever text I assign. I give at least one homework in which students are expected to actually implement a Turing machine program and submit the JFLAP representation of the program. While I do not demand that they use JFLAP, I provide a link to the JFLAP site, and 99% of the student use it.

Because I downplay it, students often feel that they have discovered this wonderful, secret software. About twice a semester, a student comes to my office hours, all excited, and shows me this great software. "And look what it does!" he[1] exclaims.

I have found that JFLAP is consistently helpful and exciting for the students. They get to verify what they have understood, and that solidifies their control of the material. And, perhaps more important, they get to *play* with the subject matter.

It is my observations that my teaching evaluations are higher on the semesters that I ask them to use JFLAP than on the semesters I forget to assign a homework using it. In short, I find it useful as a teacher, and the students find it enjoyable. In a course that is usually in the category of "they wouldn't take it if it didn't satisfy a curriculum requirement," enjoyment is high praise!

If you have any questions, don't hesitate to contact me. I am available by email at goldsmit@cs.uky.edu.

Sincerely,

Dr. Judy Goldsmith
Professor, Computer Science

[1]For some reason, the men seem more likely to think that they discovered it. But perhaps my sample is not statistically significant.

UNIVERSITY
OF HAWAI'I
HILO

6/19/06

Susan Rodger
Computer Science Dept. Box 90129
Duke University
Durham NC 27708-0129

Dear Susan,

You asked some question about our use of JFLAP at the University of Hawaii at Hilo. I use JFLAP in our CS 470 course,Theory of Computation, which is required for all graduates of our B.S. in Computer Science degree program. The course description is

> *CS 470Theory of Computing (3)*
> *Study of various models of computation and their relation to formal languages: finite automata, pushdown automata, Turing machines, regular, context-free, and recursively enumerable languages. Unsolvability, NP completeness. Pre: CS 321.*

The course is taught every two years. I have used JFLAP for what seems like quite a long time, at least since 1998. In 2001 I had a small research project in which an undergraduate student developed a list of criteria for simulation tools of models of computation and did an extensive Web search to match available tools against these criteria. For modeling finite-state machines and pushdown automata, as well as for an understanding of grammars and their conversion to finite-state machines, we found that nothing could beat JFLAP. My student then went on to create a short web-based tutorial plus a list of example problems and solutions using JFLAP.

This resource is posted on the course webpage and students are encouraged, but not required, to use it. I do give a demo in class. Most of the students do use this resource and the JFLAP tool, and find JFLAP to be very helpful. It is much easier to build, test, and modify a finite-state machine in JFLAP than it is to do it by hand, especially since JFLAP has become more "user friendly" over time. For the course this fall I have required, in addition to the course textbook, the JFLAP: An Interactive Formal Languages and Automata Package book (Rodger and Finley).

All in all, JFLAP is a wonderful tool that I know has been a labor of love over the years for its authors and is much appreciated by those of us who teach automata and formal language theory.

Sincerely,

Judith L. Gersting

Judith L. Gersting, Chair
Computer Science Department

Computer Science

200 W. KAWILI STREET
HILO, HAWAI'I 96720-4091
808-974-7450
gersting@uhunix.uhcc.Hawaii.edu

Department of Computer Science
2 East South Street
Galesburg, IL 61401
June 18, 2006

Susan H. Rodger, Associate Professor of the Practice
Computer Science Department
Box 90129
Duke University
Durham NC 27708-0129

Dear Professor Rodger,

I have been using JFLAP for the past two years. I've used it primarily in two different courses:

CS 205 Algorithm Design and Analysis
CS 206 Theory of Computing

I use JFLAP for in-class demonstrations and I also recommend it's use for homework assignments in both classes. JFLAP is optional in the Algorithms course where we use it during a section on Turing machines. It is required for several homework assignments in the Theory course, when we discuss finite state machines, PDAs, and (again) Turing machines.

I think JFLAP is terrific in allowing me to illustrate topics in both these courses and I think it helps motivate students and increases their understanding of the material. Based on course evaluations, my students use JFLAP even when it's optional, and they find it a very useful tool.

In addition to the two courses mentioned above, I've had students in my Software Development course use JFLAP to model finite-state machines for software projects that they were creating. It's a terrific tool to model things like network protocols that fall naturally into an FSM.

I'm very enthusiastic about JFLAP, it helps me in my courses and allows my students to get a better understanding of material.

Best Regards,

John F. Dooley
Associate Professor and Chair
Department of Computer Science
jdooley@knox.edu
+1.309.341.7748

ITHACA

July 3, 2006

This letter is in support of Susan Roger and the JFLAP course software for the NEEDS 2006 Premier Award. At Ithaca College, I have been teaching our 312-411 Formal Languages and Automata course for seven years. Four years ago, I experimentally introduced one lab (one week) devoted to JFLAP. Student response was highly positive. They felt the concepts and principles of automata made much more sense when they could "see" the automaton in action. I know it sounds incredible that students would actually get excited about learning theoretical computer science, but today's generation of video-game, multi-media oriented students were enthusiastic about using JFLAP. As a result, students were more motivated to put in the necessary "time on task" to grasp the challenging concepts of automata, formal languages, and theoretical computer science.

My experience with using JFLAP has been so rewarding that over the last 3 years, I have gradually modified labs and assignments for a larger and larger portion of the semester. I now use JFLAP in the classroom to demo example of finite automata, pushdown automata, and Turing machines. Labs are designed to allow students to experiment with building automata that meet specifications and to determine the formal grammar for a given language. This fall, I will be using JFLAP for the entire semester.

Without reservation, I sincerely recommend JFLAP for the NEEDS 2006 Premier Award. This software and available instructional materials, including text, is a terrific tool for bringing theoretical computer science to a new generation.

Best regards,

Wanda Dann

Wanda Dann
Associate Professor of Computer Science
Department of Computer Science
Ithaca College

June 15, 2006

Dr. Susan H. Rodger
Associate Professor of the Practice
Computer Science Dept.
Box 90129
Duke University
Durham, NC 27708-0129

Dear Dr. Rodger:

I have used your JFLAP software for a number of years. As the instructor in charge of our undergraduate course in Theory of Computation, I am constantly looking for alternative methods and tools to deliver material that many students find difficult. I can honestly state that JFLAP is the best such tool I have found.

By using JFLAP, my students have been able to visualize many of the concepts studied in class as well as test their solutions to assigned homework. I normally introduce and demonstrate the software in class. Initially, I used it as an optional tool for demonstration and practice. In recent years, however, I have assigned problems whose solutions must be submitted as JFLAP files. The software allows the instructor and the students to use a common platform for the electronic submission and testing of assignments.

I have found that the software is particularly useful for the construction and testing of automata. Its user interface allows students to create machine elements graphically in a manner similar to that used in book examples. The various options for tracing the operation of automata using different inputs help students understand their function, particularly in the case of nondeterministic automata.

In short, JFLAP has been a very important tool in my Theory of Computation class. On behalf of my students, I sincerely appreciate your efforts in building and maintaining the software.

Sincerely yours,

Jose L. Cordova
Jose L. Cordova, Ph.D.
Associate Professor and Chair of Computer Science
University of Louisiana at Monroe
Monroe, LA 71203

To Whom It May Concern:

Our B.Sc. degree program in Computer Science was accredited in 1999. We had to revise our curriculum and courses to meet the requirements of CSAB. It took nearly three years to prepare and apply for accreditation. All of our Computer Science courses are 4 credits, 3 hours of lecture and 1 hour of lab. Once in fifteen days a closed 2-hour lab session in observed for every class.

For most Computer Science courses the laboratory component comes natural. For course like Discrete Mathematics and Automata Theory, designing a laboratory component is non-trivial. Our course that covers Automata Theory is CSCI 340 -- Computational Models. We use JFLAP to implement the laboratory component of the course. We are doing it since our program was accredited.

Keeping in mind the overall mathematical background of \under-graduate students, teaching theoretical computer science is not easy. I have done it for several years. Traditionally the programs in this field are written using pencil and paper. Usually, the areas covered are Finite State Machines, Pushdown Automata, Turing Machines, Regular Expressions, Grammars, and Parsing Techniques. Without using a programming platform like JFLAP, the students would write their programs using pencil and paper and verify the correctness of their endeavor using hand-simulation. Furnishing a mathematical proof for the correctness of their programs in beyond the scope of this class.

JFLAP makes it easy. The students code their solutions in JFLAP, test, debug, and run them. I am certain that JFLAP improves the students' algorithmic thinking and thereby helps them better understanding the theory.

Although I am the first who started using JFLAP in our Computational Models course, now all of my colleagues, who teach the course, use it.

I must say that I cannot think of teaching automata theory without using JFLAP.

Sincerely,

Dr. Muhammad Chaudhary
Associate Professor of Computer Science
Millersville University
Millersville PA 17551
Phone: (717) 872 – 3724
Email: Muhammad.Chaudhary@Millersville.edu

Louis Ziantz
Instructor of Computer Science
Department of Math and Computer Science
Dickinson College
Carlisle, PA 17013

June 20, 2006

Dear Sir or Madam:

I am happy to write this letter in support of JFLAP's submission to the Needs Premier Award 2006 competition. I used JFLAP in our COSCI 314 Theoretical Foundations of Computer Science course in the Spring of 2004. I will likely teach the course (now called COMP 314) next spring and plan to use it again if I do.

I have used the tool primarily for in-class demonstrations of automata, either by starting with a complete machine or building an automaton on the fly with the students' input. In the first case, the ability to use an automaton to process a string in a step by step fashion is a very powerful teaching tool. It can be especially helpful when introducing nondeterminism, since it clearly highlights that the machine can transition to several states on the same input character. When building an automaton on the fly, JFLAP allows an automaton to be quickly edited. In addition, a number of test strings can automatically be run through the current machine to test a proposed solution. This allows the instructor to build an automaton with the class, ask whether all cases have been considered, run tests to see that perhaps the machine is not complete, and then correct it, while making the whole process smoother and quicker than doing it on a whiteboard. I have used it in demonstrations of deterministic finite automata, nondeterministic finite automata, deterministic push down automata, nondeterministic push down automata, and Turing machines.

Additionally, I have given students the option to use JFLAP for certain homework problems, particularly those involving the construction of push down automata and Turing machines. Here the ability to run test cases greatly helped students who chose to take this option. I would say that most of the class used JFLAP for at least a few problems, and several began using it even for practice problems that were not submitted for grading. The students who used it found the interface to be extremely intuitive and liked the idea of being able to run a set of strings to test their proposed solutions. Constructing Turing machines on paper often leads students to settle for a solution, as it can be time consuming to trace and retrace test cases through the machine. By using JFLAP, the students found it easier to both test and make changes to their automaton,

which made the experience less frustrating while also giving them greater confidence in their submitted solution. The ability to give states labels allowed comments to be added to a machine much like a computer program, which made completed exercises easier to study from for an exam. Those students who used JFLAP to trace nondeterministic execution on their own said it was a useful experience.

In the future, I plan to continue using JFLAP in demonstrations and will expand its use in homework, allowing students the option of using it earlier and probably making its use required for some problems. I will also assign some problems that involve using JFLAP to trace through the acceptance or rejection of a string, particularly for a nondeterministic machine.

In closing, I have found JFLAP to be a very useful pedagogical tool and would highly recommend it for other instructors who are either introducing automata to students or using these constructs in other courses, such as compiler design.

Sincerely,
Louis Ziantz

From: "Denise Byrnes" <dbyrnes@wooster.edu>
To: <rodger@cs.duke.edu>
Date: Thu, 15 Jun 2006 09:42:44 -0400

Denise D. Byrnes
Associate Professor
Mathematical and Computer Sciences
The College of Wooster

Course: CS253 - Theory of Computation
Last 3 years

JFLAP is required, demoed in lecture, used for homeworks, labs and in class activities. Topics incluse FA, PDAs, TMs, REs, Grammars and L-Systems.

I love the tool and so do the students. Students give the highest praise for the labs that use JFLAP. We find it to have an intuitive interface, helpful in clarifying constructions just as conversion of an NFA to DFA and especially "fun" when constructing original L-Systems.

I have asked other faculty at the college to consider using the tool and all have included it in their course design.

Very nice tool - want to thank you for your effort in designing it and making it available to faculty.

Sincerely,
Denise D. Byrnes
Denise D. Byrnes
Associate Professor CS
The College of Wooster

From: "Coomes, Judith" <CoomesJ@wpunj.edu>
To: "Susan Rodger" <rodger@cs.duke.edu>
Date: Tue, 30 May 2006 14:46:20 -0400

Hi Susan Rodger,

I have been using JFLAP when I teach our Theory of Computation course (CS 445) for CS majors for the past three years. The first year we just tested it out. Subsequently, we have been using it to design, test, and simplify finite automata. I do require it at the beginning of the course so that my students can become familiar with it. They can use it to check their homework and/or they can use it to do their homework. I do spend one class period at the beginning of the course working some sample problems which we then do in JFLAP during the same class period (if it is a double class) or the next class period (if it meets twice a week). I demo it and then we use the rest of the period as a lab.

This course has a project requirement and the students can choose what they want to work on (within limits). They will frequently choose topics that they can do with JFLAP. For example, this year one group chose to model L-systems in two and three dimensions with JFLAP. Another student used it to model Turing machines with multiple tapes.

When I first took over the course (CS 445), the student interest in it was dying. The course was very theoretical and abstract and it was failing to make connections with other topics and other courses. While I have maintained a substantial portion of theory and proof, there is now a healthy dose of application, experimentation, and project and presentations. JFLAP has become an integral part of the course. It has helped to make CS 445 a vital course again.

I hope that this has answered your questions. If you prefer, I can write this in the form of a letter and mail it to you. Please let me know. In my opinion, JFLAP and its creators are deserving of an award for innovation in course based software. I wish you luck in this quest. Thank you for this opportunity to tell you how much we are enjoying working with JFLAP.

Sincerely,
Judith A. Coomes
Associate Professor
Department of Computer Science
William Paterson University
300 Pompton Road,
Wayne, NJ 07470
973-720-3383/2649

From: "Alice M. Dean" <adean@skidmore.edu>
To: Susan Rodger <rodger@cs.duke.edu>
Date: Mon, 19 Jun 2006 12:32:15 -0400

To Whom It May Concern:

My name is Alice Dean, and I am a professor in the Department of Mathematics and Computer Science at Skidmore College in Saratoga Springs, NY. The purpose of this letter is to comment on JFLAP, a software package which provides a graphical interface for constructing finite automata of various types.

I have used JFLAP for quite a few years in the course MC 306 Theory of Computation, which is a required course for the computer science major at Skidmore. I use JFLAP quite extensively, and I require my students to do so as well. When we study each of the main types of automata and associated grammatical constructs (finite automata and regular expressions, pushdown automata and context-free grammars, and Turing machines), the students and I all use this tool to construct and work with examples. I use it extensively during class meetings, since we have a classroom equipped with a computer and overhead projection. I also give homework assignments in which I require students to submit JFLAP files containing their solutions to problems involving construction of automata and other related operations (such as conversions from one type of object to another, determining properties of automata through test runs, etc.). For examinations, I use JFLAP to produce professional looking diagrams that I paste into my exam questions.

My experience with JFLAP over the years is that students really enjoy using it, and that they prefer using it to doing pencil and paper constructions, since modification and testing is so easy. It has been my experience that if students are given an assignment to construct an automaton that accepts a specified language, they will be much more thorough in their testing if they have JFLAP at their disposal. I believe that it really helps them achieve a much deeper understanding of the material than they otherwise would. Its ease of use is critical, since I need to spend almost no time at all showing them how to use it. The last thing I want is to have teaching the use of software overshadow the teaching of real material of the course.

When colleagues of mine teach this course for the first time, I strongly recommend JFLAP to them. I have found JFLAP to be an invaluable tool, and the fact that it is free is a wonderful bonus.

If there is any other information I can supply, please feel free to contact me.

Sincerely,

Dr. Alice M. Dean
Mathematics and Computer Science Department
Skidmore College
Saratoga Springs, NY 12866
Email: adean@skidmore.edu

From: "Dr. Bruce Elenbogen" <boss@umich.edu>
To: jflap@cs.duke.edu
Date: Mon, 05 Jun 2006 09:43:10 -0400

To whom it may concern

I am Dr. Bruce Elenbogen, Associate Professor of Computer and Information Science at the University of Michigan Dearborn. I have been using JFLAP for over 10 years and am writing concerning its benefits to teaching my classes. We just JFLAP to teach finite autotmata, Turing machines and grammars. Its visual interface is so easy to use that it is wonderful for demonstrations in class. I can easily draw examples with it during class as demos. Better yet, when a student asks a question, it is extremely easy to create new examples or make modifications to old ones. it is these types of "what happens if we change " questions where students truly master the material.

However the best part of JFLAP is the simulations that allow students to see step by step the processing of a word. It is this process where I find most students grasp the concept of Turing machine, finite automata or grammar.

The output of JFLAP has improved over the years but it was always of such high quality that, I proudly posted the examples I did in class to the class website for examples. The interface is so well designed that students use it for homework problems even when it is not required.

In short it is a wonderful tool and just keeps getting better. It has certainly improved my effectiveness in the classroom. I wish to thank Susan Roger for creating this wonderful teaching aid.

Dr. Bruce S. Elenbogen
boss@umich.edu

From: jlucas@brockport.edu
To: Susan Rodger <rodger@cs.duke.edu>
Date: Tue, 06 Jun 2006 18:57:24 +0000 (GMT)

Dear Dr. Rodger,

I am pleased to have the opportunity to comment on my experience using JFLAP. Currently I serve as an Associate Professor in the Computer Science Department at the State University of New York, College at Brockport. Brockport is a 4-year comprehensive liberal arts college enrolling approximately 8,000 students. About 40 students graduate from Brockport with a Bachelor's degree in Computer Science each year.

I teach the course CSC 483 - Theory of Computation. This is a required course in the ABET-accredited track of the Computer Science major. The course provides a thorough treatment of the standard topics in automata theory (finite state machines, context free languages, Turing machines, undecidability).

I have used JFLAP in CSC 483 for the past three years. As soon as I discovered this tool, I was completely hooked on it. I cannot now imagine teaching this course without this tool. I use JFLAP constantly in the classroom to develop and demonstrate example automata. It is far faster than drawing on the board, and is far more flexible and dynamic than using static PowerPoint slides. During class, I frequently ask the students to describe how they would design the automata, which I then precede to draw for all to see using the JFLAP tool. The ability to test run the sample automata on any variety of inputs is very valuable. I especially like JFLAP's ability to single-step simulate the execution of a non-deterministic automaton, illustrating all of the possible branches of execution. The class sessions are now far more interactive, spontaneous and fun than in past years, when I needed to use pre-planned examples.

My favorite "JFLAP moment" was when the students and I spontaneously developed a simple Turing Machine to perform division in unary notation. The solution requires programming a nested-loop. After developing the Turing Machine, we naturally tested it out on some sample inputs. One input was the "divide by zero" case. We had not considered that possibility when we had developed this Turing Machine on-the-fly in class. The divide-by-zero case caused the Turing machine to enter a non-trivial infinite loop. This served as perfect motivation for the following discussion on undecidability.

But JFLAP provides much more than a tool for defining the concept of automata, and drawing and executing sample machines. JFLAP is invaluable as a tool for teaching the underlying theory. JFLAP has a built-in ability to visualize such important theorems as the equivalence of non-deterministic finite automata and deterministic finite automata. Students are able to carry out the steps of this general transformation themselves, helping them to fully understand the proof. Similarly, JFLAP has an option that shows students how any finite automaton can be minimized.

Students are required to use JFLAP extensively in their homework assignments. JFLAP is very easy to learn, and has an intuitive interface.

The students are able to "hit the ground running" when doing their projects. I require the students to send their JFLAP files to me for testing. In the past, when students were asked to write non-trivial Turing Machines, there was no way for me to realistically check the correctness of their work. Now I do so routinely.

In conclusion, JFLAP has proved an extremely valuable learning tool, and I look forward to using for many years to come.

Respectfully, Joan Lucas

Dr. Joan M. Lucas
Associate Professor
Department of Computer Science
State University of New York, College at Brockport
350 New Campus Drive
Brockport, New York 14420
585-395-2196
585-395-2304 (fax)

From: James Riely <jriely@cti.depaul.edu>
To: "Susan H. Rodger" <rodger@cs.duke.edu>
Date: Tue, 23 May 2006 15:31:23 -0500

To whom it may concern,

I have taught Automata Theory several times at DePaul University, both for graduate and undergraduate students. Each time I teach the course, I have looked for the best tools available to support the material. The last two times I taught the class, in 2004 and 2005, I found JFLAP to be the best tool available.

I used JFLAP in lectures, since it neatly illustrated the concepts involved. I used it to explain the basic semantics of automata, and to illustrate some of the algorithms used in proofs of closure properties and equivalences.

While I did not require the use of JFLAP by students, I believe all of my students downloaded the tool. It is an excellent resource.

Yours,
James Riely
Associate Professor
School of Computer Science, Telecommunications and Information Systems
DePaul University
Chicago, IL

From: Nancy C Van Ness <nancyvn@utdallas.edu>
To: Susan Rodger <rodger@cs.duke.edu>
Date: Sat, 27 May 2006 10:13:12 -0500 (CDT)

In support of JFLAP

I am a Senior Lecture II and a graduate advisor in the Computer Science Department at UT Dallas. I have used JFLAP in CS 4384 Automata Theory (undergrad) and CS 5349 Automata Theory (grad) for 3 semesters. It is not required but strongly recommended. I know students use it because I am asked questions about it and asked to upload the demos from class to Web CT. I have used all parts for the finite automatas (regular expressions, minimization included). I do not use the changing a PDA to grammar capability as I have a slightly different algorithm. However, students like to construct and run PDAs in class. I have used the Turing Machine capability only sparingly but when I did use it, you could have heard a pin drop because all eyes were on the screen watching the input string being processed.

Attendance in class is very good when I am using JFLAP as illustrations of topics presented in lecture. I think it helps students see the underlying concepts and is very useful in explaining non-determinism. They also go home to construct their own machines or use the examples that are done in class. I think the over-all grades are up slightly. I do not have anyone totally lost and floundering. It makes the class quite lively and I can use one example that slowly changes into another to illustrate points with a wide variety of input that took much longer to describe when everything needed to be drawn by hand. Students seemed engaged and less bored throughout the entire class. They do not get lost in the explanation.

Students have volunteered in my office (without any prodding on my part) that they like and use JFLAP. I know that this is the case for a large portion of the class turn in machines that have been created using JFLAP. I also know that those who are in the other sections have been initiated by my students into its use.

One of the nicest uses though is when a student asks a question and we construct immediately a machine that can illustrate many subtleties. Frequently, there is discovery and the student essentially turns out answering his own questions. I think this very valuable in getting students to begin the process of learning how to learn.

I find this tool so very valuable and the student support so great that I have tried to recruit others to use it. I cannot imagine teaching these courses without it.

Regards,

Nancy Van Ness

Senior Lecturer II
Graduate Advisor
Computer Science Department
University of Texas at Dallas
PO Box 830699 MS ECS 31
Richardson, TX 75083-0688

Phone: (972) 883 - 4858
Office: ECS 4.706

Date: Thu, 15 Jun 2006 19:33:51 -0500 (CDT)
From: <rmverma@cs.uh.edu>
To: rodger@cs.duke.edu

This letter is in strong support of JFLAP, developed by Dr. Susan Rodger's group at Duke university, for the Needs Premier Award 2006 competition.

I have been using JFLAP for the last three years in my COSC 3340: Introduction to Theory of Computation course at the University of Houston. During this period, I have used it in my lectures for visualization of deterministic/nondeterministic finite automata, push-down automata/context-free grammars and turing machines, and on assigned homework problems for the course. Beyond using it in the course, two of my students have worked on extending JFLAP to add a random string generation feature to JFLAP. This effort to extend JFLAP was supported in part by a grant from the National Science Foundation.

My students and I have found JFLAP easy to use, well-designed and well-documented. The software is highly portable and has a very convenient and intuitive graphical interface. It has nice graphics that enables students to visualize all kinds of automata and interactively simulate them, getting ''instantaneous'' feedback. Feedback on JFLAP from the students has been uniformly very positive. I believe that JFLAP is definitely helpful in teaching and learning the rather abstract and difficult theoretical concepts in this course. The two students who worked on extending JFLAP were able to extend it in a timely manner and without any significant difficulties.

Ever since I started using JFLAP and redesigned the course materials, several students have been excited and motivated enough to pursue automata theory projects under my supervision. This is in stark contrast to the situation before using JFLAP and redesigning the course materials (hardly any student pursued automata theory projects with me over a decade). I believe that JFLAP is playing a significant and important role in this success.

In summary, I strongly recommend JFLAP for this award. Please do not hesitate to contact me if you need further information.

Sincerely,

Rakesh Verma
Professor
Computer Science Department
University of Houston
713-743-3348

June 20, 2006

This is a letter in support of JFLAP for the Needs Premier Award 2006 competition for high-quality, noncommercial courseware designed to enhance engineering education.

Name - Soe Than

Title or Position - Professor in Computer Science

Department - Mathematics and Computer Science

Institution - Virginia Military Institute

1. *Have you used JFLAP in a course?*

Yes

Course (title and number) JFLAP is used in at your school

CS 441 – Formal Languages and Automata

How long have you been using JFLAP?

I taught this CS 441 course for the first time in Spring 2006 and I used JFLAP at that time. I will teach CS 441 in the Fall 2006 and I plan to use JFLAP again.

How JFLAP is used
- *is it required or optional, if optional do you think students use it?*
- *is it demoed in lecture?*
- *assigned or optional to use for homeworks?*
- *which topics do you use JFLAP with?*

JFLAP is used to demonstrate how Finite Automata, Pushdown Automata, and Turing Machines work. It is used to illustrate construction of these machines and check whether the machines meet the desired specifications. Students were not required to use JFLAP. But students realized the benefits of interactive construction and verification provided by JFLAP, and used it in doing their homework. I used JFLAP with Finite Automata, Regular Expression, Regular Grammar, Pushdown Automata, Context Free Grammar, and Turing Machine topics.

Do you think JFLAP is helpful in learning material?

Yes, I think that JFLAP is helpful in learning the formal languages and automata material. It shows how the automata work, how the automata and grammar are related, and how to convert an automaton to corresponding grammar. Students get clear understanding of these concepts from JFLAP.

Opinion on how the students like it.

My students enjoyed using JFLAP and liked the help provided by JFLAP.

2. *Have you used JFLAP for you research or some other use? If so please explain.*

As my research area is different from the Formal Languages and Automata field, I haven’t used JFLAP for my research.

3. *Your opinion on or praise for JFLAP.*

I think JFLAP is wonderful, helpful, and useful software for me in my teaching of CS 441 course. It is comprehensive in coverage of all topics in the area of Formal Languages and Automata. Its interactive user interface with helpful comments provides students enjoyable environment to learn a highly theoretical (*reads boring*) subject. If a picture is worth 1000 words, then a demonstration using JFLAP is worth 1000 pictures.

For any questions regarding to this recommendation letter, I can be reached at thans@vmi.edu.

Sincerely,

Soe Than
Professor in Computer Science
Department of Mathematics and Computer Science
Virginia Military Institute
Phone: 540-464-7498
Fax: 540-464-7214

Department of Math & Computer Science
Wheaton College
Norton, MA 02766

June 20, 2006

Dear Susan,

I have not yet used JFLAP in a course, but I will use it next year when I next teach COMP 375 - Theory of Computation. We will use it in at least one lab session in addition to its use outside of class. I also plan to use it in class to interactively built machines and instantly see if they work properly.

In trying out JFLAP, I found that it is very easy to use and that it includes all of the standard models. Additionally, JFLAP includes a multi-tape Turing machine and even an L-System modeler. The latter I plan to show my COMP 365 - Computer Graphics class, as I usually have them write an L-System renderer with OpenGL. More importantly, however, is JFLAP in teaching students in the use of the various automata. Students often have difficulty creating and testing automata, especially when epsilon transitions are allowed. JFLAP gives students an easily-learned, platform independent tool to create and test machines quickly. This is much better than the old pen-and-paper method, where one small mistake can easily disappear among all the transitions. JFLAP also gives students a way to check their work, not only in determining if a particular string is accepted but also in converting one type of automata to another.

I find JFLAP very robust and easy to use. I wish I had such a tool when I first learned about automata. I am confident that students will like and use this tool in their work.

Sincerely,

Michael B. Gousie
Associate Professor of Computer Science

From: Ellen Walker <walkerel@hiram.edu>
To: "Susan H. Rodger" <rodger@cs.duke.edu>
Date: Tue, 30 May 2006 20:54:38 -0400

Dear Susan:

I'm writing to let you know how much I have appreciated your JFLAP software. I use the software when I teach Compilers at Hiram College (last taught 2001, also to be taught next spring), both in lecture and for student exercises. I have mainly used the DFA and PDA tools with my students. Students are encouraged, but not required, to use the tools for their homework, so they can test their automata before submission. Most students have chosen to use the tool, and (at least anecdotally) they were pleased with it. Students who used JFLAP turned in better (and easier to read) homeworks, and I plan to require its use next spring. Also, I hope to use the new JFLAP book as a supplement to my compiler text for the sections on lexical component extraction and parsing.

I and my students appreciate the fact that the tool is platform-independent, free, and easy to use. Thank you for developing this excellent software that has improved both my teaching and my students' understanding of these topics.

Sincerely,
Dr Ellen Walker
Professor of Computer Science
Hiram College
PO Box 67
Hiram OH 44234

From: "ralph zegarelli" <zegarelli@hartford.edu>
To: "'Susan H. Rodger'" <rodger@cs.duke.edu>
Date: Tue, 23 May 2006 17:17:19 -0400

Name: Ralph Zegarelli

Institution: University of Hartford

Your Position: Computer Science Faculty

Course JFLAP is used in: CS 340 Formal Languages and Automata

When did you use JFLAP? . Last used in Fall semester 2005

How long you have been using JFLAP: I began using JFLAP in Fall 2002. I teach CS 340 every Fall and have used JFLAP in every class.

How is JFLAP used? JFLAP is required for homework and in class problem solving/testing

Anything you would like to say about JFLAP? This is a great tool. It gives students the ability to code and debug solutions in the same way a compiler is used to teach computer languages. I do assign coding problems to students but use JFLAP for all other assignments.

From: "Zimmerman, Jill" <jzimmerm@goucher.edu>
To: "Susan Rodger" <rodger@cs.duke.edu>
Date: Thu, 15 Jun 2006 09:45:33 -0400

I have used JFLAP for several years in teaching CS250, Theory of
Computation, at Goucher College. I have used it both in class
demonstrations on all the various automata and in required homework
assignments. The feedback that I have received from students is
uniformly positive, and I have been told by at least one student that it
was his favorite part of the course. It definitely motivates the
learning in the course.

Jill Zimmerman, Ph.D.
Associate Professor
Mathematics and Computer Science Department
Goucher College

From: "Fereydoun X Kazemian;;;" <fxk@cs.rit.edu>
To: rodger@cs.duke.edu
Date: Mon, 10 Jul 2006 17:05:39 -0400 (EDT)

This letter is in support of JFLAP software. My students have been using JFLAP for the last two academic years in "Introduction to CS Theory" course (4003-380) that I teach. Although using JFLAP is optional, all the students use many of its features for a number of homework assignments dealing with different types of automata. I demonstrate how to use the basic features of JFLAP in class. On a number of occasions during the lecture, I use JFLAP to show examples or demonstrate concepts relating to automata theory and formal grammars. I have found JFLAP to be an excellent educational tool for discussing and demonstrating many aspects of those topics. Based on the feedback that I have received from students, they find JFLAP very helpful in understanding concepts and doing exercises relating to finite automata, regular grammars, regular expressions, pushdown automata, context-free grammars, parsing, and Turing machines.

Overall, I consider JFLAP to be a high-quality software designed to enhance computer science and engineering education.

Fereydoun Kazemian
Associate Professor
Department of Computer Science
Rochester Institute of Technology

From: Maria Gabriel <cgabriel@Radix.Net>
To: "Susan H. Rodger" <rodger@cs.duke.edu>
Date: Wed, 24 May 2006 14:48:25 -0400 (EDT)
Subject: Re: JFLAP user: Need letter of support from you!

To Whom It May Concern:

Institution: Park University
Postion: Faculty
First Used: 2002
Use: Required in Lab assignments, used in class room demos
Comments: Very useful! We use the Finite State Machine and Grammar
Generator in our CS 380 Compiler class.

/signed/
Harry C. Harrison

DSV Institutionen för Data- och Systemvetenskap

STOCKHOLMS UNIVERSITET

KUNGLIGA TEKNISKA HÖGSKOLAN

9 juli 2006

To whom it may concern

I am an Associate Professor at the Department of Computer and Systems Sciences, Stockholm University, Sweden. As a faculty at the department I am among other things responsible for a course on "Formal Languages and the Theory of Computation" (DSV2:B/*:2B) that was last year given for approximately 160 students as part of their undergraduate education. I write this letter to express my appreciation of the JFLAP software package which was used by me for the first time during last years course. JFLAP was used both during lectures, as a convenient tool to demonstrate and provide insight into formal languages and automata theory, and also by the students when working with the course assignments. The software has been extremely appreciated by both myself as well as by the students and will be used also during this years course. I think it is safe to say that the quality of the course has increased considerably compared to previous years due to the fact that we started using JFLAP. It provides an excellent teaching framework that gives both instructors and students a better opportunity to focus on the core of a complex subject.

Best Regards

Dr Lars Asker
Associate Professor
Department of Computer and Systems Sciences
Stockholm University/Royal Institute of Technology
Email: asker@dsv.su.se

UNIVERSITEIT GENT

FACULTEIT WETENSCHAPPEN
Zuivere Wiskunde en Computeralgebra
Voorzitter: Prof. Albert Hoogewijs

To who it may concern

June 14, 2006

Letter of Appreciation on JFLAP

I have been using JFLAP (and JellRap) for about 10 years, as support for my course on Formal Languages and Automata.
Today the course notes mainly consist of the book:

- An Introduction to Formal Languages and Automata, Third Edition Peter Linz, University of California, Davis ISBN: 0763714224
- + extra chapters on LL(1) en SLR(1) parsing

This course is taught in the Third Year Bachelor of Informatics and
Master of Mathematical Informatics at Ghent University.
Current name of the course: Formal Languages, Finite Automata and Complexity
(Reference **CBINFO03000006).**

Over this period about 50 students per year took this course.

JFLAP is used to support the following topics:

- Finite automata
- Pushdown automata
- LL(1) parsing tables
- SLR(1) parsing tables
- Turing Machines

It is required for solving the problems the students get during the semester and the final examination.

JFLAP is an excellent tool and is very much appreciated by the students. It provides facilities for integrating automata theory with corresponding grammars. Lecturing LR parsing becomes a real pleasure for both teacher and student.

Albert Hoogewijs
tenured academic staff
head of department of Pure mathematics and computer algebra
Ghent University

albert.hoogewijs@ugent.be

From: James Harland <jah@goanna.cs.rmit.edu.au>
To: "Susan H. Rodger" <rodger@cs.duke.edu>
Date: Tue, 23 May 2006 14:01:48 +1000

G'day. Hope this is what you need. All the best with the award!

Cheers,
James.

To Whom It May Concern,

Re: JFLAP software

I am an Associate Professor in Computational Logic in the School of Computer Science and Information Technology at RMIT University in Melbourne, Australia. I have been using JFLAP as a teaching tool in my Computing Theory course for several years. I find it has been an invaluable asset for the teaching of this material. There are a number of laboratory exercises that students are strongly encouraged to complete which use this tool as a means of designing and executing finite state machines, pushdown automata and Turing machines. Apart from the concrete experience of seeing what a particular automaton does on a computer screen (rather than working it out painstakingly with pen and paper), I find the suite of algorithms implemented in JFLAP to be particularly useful. For example, students often find it difficult to understand the algorithm for converting a non-deterministic finite state automaton into a deterministic one. JFLAP is very helpful here, as the students can use it to perform this conversion on various examples, which they can make up themselves, rather than relying on a small set of pre-worked examples from a textbook. This instantaneous feedback on individualised examples makes an enormous difference to the student's conceptual understanding.

JFLAP continues to be an important part of the teaching material of this course, and the continual improvement of the package ensures that it does not become dated. In short, JFLAP has made it easier, for both students and lecturers, to concentrate on what is important rather than getting lost in details.

Yours sincerely,
James Harland

From: Lila Kari <lila@csd.uwo.ca>
To: rodger@cs.duke.edu
Date: Wed, 14 Jun 2006 16:25:49 -0400 (EDT)
Subject: JFLAP letter

To Whom it May Concern:

I am writing this letter in support of JFLAP as an educational tool. My name is Lila Kari and I am Associate Professor and Canada Research Chair in Biocomputing in the Department of Computer Science, University of Western Ontario, London, Canada. JFLAP is used in CS331, Foundations of Computer Science (required textbook John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman. Introduction to Automata Theory, Languages and Computation. 2nd Edition, Addison-Wesley, 2001. ISBN#0-201-44124-1). I used JFLAP for the first time in the 1997-1998 school year. I intend to use it again this coming academic year, 2006-2007.

In CS331 the use of JFLAP is required and several of the assignments are based on its use. In one of the first classes, a demo of the software is presented. JFLAP is mainly used for the illustration of regular languages and construction of automata as well as for the construction and illustration of Turing machines.

I found JFLAP to be an extremely useful and motivating tool for students, who in general have a low tolerance for theory courses. The fact that JFLAP is interactive made the whole theory course have a more concrete hands-on flavour. The fact that the students could actually see the steps involved in parsing a word by their designed automata made all the notions tought in the course more palatable and real to them. I think JFLAP was crucial in making students understand and appreciate theoretical concepts.

Thus, I found JFLAP to be an invaluable tool in teaching theoretical computer science. The students expressed both to me, verbally, and in their teaching evaluations how they found that JFLAP helped them conquer their fears and in fact enjoy theoretical computer science.

I would not consider teaching again without it.

Sincerely yours,

Lila Kari

Lila Kari, Associate Professor and Canada Research Chair in Biocomputing
Department of Computer Science Tel: 1-519-661 2111 ext.86894
University of Western Ontario Fax: 1-519-661 3515
London, Ontario Email: lila@csd.uwo.ca
N6A 5B7 Canada http://www.csd.uwo.ca/~lila

Universidade Católica de Brasília – UCB
Pró-Reitoria de Pós-Graduação e Pesquisa – PRGPP
Mestrado em Gestão do Conhecimento e da Tecnologia da Informação

To

Professor Susan Rodger
Computer Science Dept. Box 90129, Duke University, Durham NC 27708-0129

My name is Germana Menezes da Nóbrega, I take part as a professor both in the undergraduating program on Computer Science of Universidade Católica de Brasília, Brasília, Brazil, and at the Master program on Knowledge Management and Information Technology at the same University. Since 2004 me and some colleagues have been exploiting JFLAP with undergraduating students (on Formal Languages and Automata Theory, Theoretical Computer Science, and Compilers) in several situations: during lectures running demonstrations, in laboratory classes, and also recommending it for homework. The topics we mainly use include Finite Automata, Pushdown Automata, Turing Machines, Regular Expressions, and Grammars. I think the tool is really helpful for students to workout the constructs underlying Theoretical Computer Science. Yet, students seem to appreciate it a lot, so that we are exploiting JFLAP also from a research perspective within Computer Supported Collaborative Learning. We think to be able to assign the tool beyond the optional character it takes today in our courses.

Kind regards,
Germana Menezes da Nóbrega.

Germana Menezes da Nóbrega – Professor Adjunto
SGAN 916 – Módulo B – Asa Norte – Brasília, DF - CEP 70.790-160
Tel.: (61) 3448 7121 Fax: (61) 3347 4797
e-mail: gmnobrega@pos.ucb.br Homepage: http://www.ucb.br/prg/professores/germana

Dear Dr. Rodger,

I applaud your great software “JFLAP”. I use it in a course and find it very helpful for the students. JFLAP helps students to understand the related topics in shorter time and also to make sense.

Another feature of JFLAP is its user interface which makes it easy to use. Hence we didn’t spend any class time to show any demo to the students.

I asked my students about how they found the JFLAP and the statistic result shows that 95% of them agree on the benefits of it in their learning.

The lines below describe more how we use jflap in our course:

1. Course Name : Formal languages and automata
2. JFLAP was required in some computer assignments and in some other ones it was optional.
3. The topics that we use JFLAP for are as below:
    - Finite automata
    - Pushdown automata
    - Turing machine
    - Regular expression
    - Grammar

Best regards

Mina Zolfy Lighvan
Lecturer of Information Technology Dept.
Faculty of Electrical and Computer Eng.
University of Tabriz, Tabriz, IRAN.

From: Antonio Dourado <dourado@eden.dei.uc.pt>
To: rodger@cs.duke.edu
Date: Tue, 23 May 2006 15:26:18 +0100

To whom it may concern

I am Full Professor at the Department of Informatics Engineering of the University of Coimbra, Portugal, the third oldest University in Europe. I am teaching Theory of Computation , a compulsory course for the 150 students of BSc in Informatics Engineeering. I have been using JFLAP for three years in the classroom and my students have it in their computers. The software has been very useful and important for learning automata theory. As a teacher I benefited from it to plan exercises and to solve them. It is a very good support for prepairing studying materials and examinations. My students all of them mandatory use the software in classroom and (I hope) at home for supporting their learning activities.

I think that JFLAP is the best free software for automata theory and I hope that it will continue to be supported to become still better, since to teach this subject without a good software is not advisable.

Coimbra, May 23th 2006.

Antonio Dourado
Full Professor

From: =?iso-8859-2?Q?Piotr_Dziurza=F1ski?= <pdziurzanski@wi.ps.pl>
To: "Susan H. Rodger" <rodger@cs.duke.edu>
Date: Tue, 23 May 2006 13:34:48 +0200
Subject: Re: JFLAP faculty: Need letter of support from you!

Dear Ms. Rodger,

As I highly value JFLAP, here is some information about my usage of your tool. I hope you'll find it helpful and you'll get the award.

Regards,
Piotr Dziurzanski

Name: Piotr Dziurzanski
Institution: Szczecin University of Technology (Poland)
Position: assistant professor
Course: Theory of formal languages
No. of students (annually): 180
Used for: 4 years
JFLAP is used during lectures and students are encouraged to use it at home in order to deeply understand the subject. For me, JFLAP is a tool that is helpful in making more interesting lectures what is essential in such theoretical-oriented courses as formal languages. Moreover, it is good way for students to prepare to our table & chalk exercises, as they can check whether thay understand the matter.

From: Stefano Crespi <stefano.crespireghizzi@polimi.it>
To: "Susan H. Rodger" <rodger@cs.duke.edu>
Date: Tue, 23 May 2006 13:10:40 +0200
Subject: Re: JFLAP user: Need letter of support from you!

To Susan H. Rodger

Dear Colleague,

it is my pleasure to report to you the use I have been making of the
package JFLAP.
At the Universit della Svizzera Italiana (Lugano, Switzerland) I have
taught for some years the introductory course on
Formal Languages to students majoring in Communication (a blend of
humanities and technology).
Students were requested to use JFLAP in supervised hand-on sessions.
Due to the elementary level of the course, we did not use parser
generation tools.
We found that the friendly interfaces of JFLAP were very helpful for
this kind of student population.
At Politecnico di Milano, Dipartimento Elettronica e Informazione,
I currently teach the graduate course on Formal Languages and Compilers
to computer science majors,
over 200 students per year.
Here the use of JFLAP is also important though less systematic, because
students are requested to attend lab sessions to practice with
classical compiler generation tools such as Flex/Bison. I tell students
to use JFLAP for checking the exercises on
finite automata, regular expressions and grammars, when preparing for
the written exam.
Personally, I resort to JFLAP to check problems and solutions.
In my opinion JFLAP is a very nice toolset, especially attractive for
undergraduate teaching and also very valuable for teaching
theoretical computer science concepts to non-computer majors.
The enduring committment of the JFLAP staff to maintenance and
development of this reach and complex toolset
is unusual and much appreciated.

Yours
Stefano Crespi Reghizzi
(Full professor of Computer Science at Politecnico di Milano
Ph.D. UCLA
Chairman of the Ph.D. programme in ICT

--
Professor Stefano Crespi Reghizzi
Dipartimento Elettronica e Informazione - Politecnico di Milano
Piazza Leonardo 32, Milano I-20133
Tel. 0039 02 2399.3518 / 3405 - Fax 0039 02 2399.3411
crespi@elet.polimi.it - http://www.elet.polimi.it/

From: Andreas Rittershofer <andreas@rittershofer.de>
To: Susan Rodger <rodger@cs.duke.edu>
Date: Thu, 22 Jun 2006 13:44:38 +0200

Hi,

my name is Andreas Rittershofer. I'm a teacher for mathematics, physics and computer science at the Dietrich-Bonhoeffer-Gymnasium in Metzingen (Germany).

One part of the courses in computer science is about regular expressions, grammars, finite automata, stack automata, turing machines, ...

To make a good course at school I need a tool to demonstrate all these things and also to let the students work for themselves. At our school this tool is JFlap since two years.

The following example shows the usage:
- First I make a short introduction, i.e. explanation of what a turing machine is, how it works, ...
- This is accompanied with a demonstration via JFlap and a beamer, so that my students can see how a turing machine works and also how JFlap is used to simulate a turing machine.
- Now my students have to do some exercises (a turing machine as a parity checker, a turing machine as a binary counter, ...) with JFlap and put their result in a Moodle-course.

The homework of my students is similar.

We begin with regular expressions and regular grammars, followed by finite automata and stack automata and finally turing machines. In the case of some spare time we also create some flowers und bushes with L-Systems.

JFlap is the ideal tool for my computer science courses: It offers more than I will ever need at school and is nevertheless very easy to use, my students are familar with its handling in only a few minutes. I cannot image how to deal with automata without JFlap - and so do my students too.

--
E-Learning in der Schule:
http://www.dbg-metzingen.de/Menschen/Lehrer/Q-T/Rittershofer/E-Learning/

From: Jairo Rocha <jairo@uib.es>
To: "Susan H. Rodger" <rodger@cs.duke.edu>
Date: Tue, 23 May 2006 11:13:14 +0200

Dear Susan Rodger,

I take advantage in this letter to thank you for the first time on behalf of my colleagues and students for the wonderful tool you have created to teach automata, grammars and Turing machines.

I have used JFLAP in the past three years in my Formal Languages and Automata theory class. It allows my students to really understand that there are blind methods to solve certain symbolic problems so that computers can really solve them. I just show in class how JFLAP allows to, for instance, build completly a minimim automata from a regular expression, that,
of course, is correct. Usually, I bring my portable with JFLAP at the end of
each subject: automata, grammars, push down automata and Turing machines. Since it is in Java,
they can use it and play with it easily off class.

Please keep up with your interest on teaching.

Sincerely,

Jairo Rocha, PhD
Associated Professor
Department of Mathematics and Computer Science
University of the Baleric Islands,
Palma, Spain.

From: "Kamilla Klonowska" <kamilla.klonowska@bth.se>
To: "'Susan Rodger'" <rodger@cs.duke.edu>
Date: Wed, 14 Jun 2006 15:33:16 +0200

Hello Susan,

I'm sorry for a late e-mail. I work only 20% this season and I was very busy. You have an excellent idea! I think JFLAP should get an award because it is VERY useful in teaching and studying.

Name: Kamilla Klonowska
Title or Position: Ph.D. student
Department: The School of Engineering
Institution: BTH (Blekinge Institute of Technology), Ronneby, Sweden

Course (title and number) JFLAP is used in at your school
I teach "automata and formal languages" course (in Swedish: "automata och formella sprk") which has the number DVC005. It is the C-level at the university. You can check the Swedish version on the university home page:
https://idenet.bth.se/servlet/courseoccasion?course=2879.

How long have you been using JFLAP?
I started to use JFLAP in 2002 after I used this program when I studied a Ph.D. course "Algorithmics and Complexity" at Dalarna University in Sweden. It was a very useful and I realize that I also can use it in my course.

How JFLAP is used
- is it required or optional, if optional do you think students use it?
It is optional. They have two options: JFLAP or a pencil.
I hardly recommend JFLAP because they can check if they fault or not.

- is it demoed in lecture?
If the students need it - yes. But very shortly.

- assigned or optional to use for homeworks?
I hardly recommend it to use as a key to the right answer.

- which topics do you use JFLAP with?
DFA, NFA, PDA, TM and grammars,

Your opinion on or praise for JFLAP.
I am in a hurry right now so the answer is very short.
I think it should get the praise. It is a VERY useful tool that helps with teaching and understanding Automata and formal languages.

Do you think JFLAP is helpful in learning material?
YES

Opinion on how the students like it.
They liked it. They also think that it is a very useful tool.

Best regards
Kamilla Klonowska

National Taiwan University
Department of Information Management
國 立 臺 灣 大 學
資 訊 管 理 學 系 暨 研 究 所

June 7, 2006

Prof. Susan H. Rodger
Associate Professor of the Practice
Department of Computer Science
Duke University
Durham, NC 27708-0129
U.S.A.

Dear Prof. Rodger,

With this letter I wish to express my deepest gratitude to your creating and making freely available the JFLAP tool. I have used JFLAP for several years in my Theory of Computation course, which is an introductory course for junior and senior undergraduate students. Both my students and I have enjoyed and benefited from the illustrative visualization that the tool provides. Being able to see how an automaton, particularly a nondeterministic one, runs on an input greatly helps the students understand the subject matter. As the diagrams can be exported with a PDF printing support, the tool has also saved my time when preparing handout material and students' when writing up their homework.

As a foreign researcher and teacher, I greatly admire your tireless effort on developing JFLAP over the years and the generosity of NSF in providing the financial support. JFLAP has breathed life into an important foundational computer science course that would otherwise be dull to most students. I earnestly hope that the effort and the support will continue.

My best wishes to the continuing success of JFLAP!

Sincerely,

Yih-Kuen Tsay
Associate Professor

Tel : +886-2-33661200 Fax : +886-2-33661199 No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan, R.O.C.
電話：(02)33661200 傳真：(02)33661199 地址：台北市 10617 大安區羅斯福路四段 1 號
網址：http://www.im.ntu.edu.tw

From: Frank Neven <frank.neven@uhasselt.be>
To: rodger@cs.duke.edu
Cc: Geert Jan Bex <gjb@uhasselt.be>, Dirk Leinders
<dirk.leinders@uhasselt.be>
Date: Wed, 24 May 2006 15:32:42 +0200
Subject: JFLAP

Dear Susan,

JFLAP is a great and fun tool to use in class. We have used it in our
TCS course (although only the finite state machine part).
Thank you for developing such a tool!

I attach an interesting paper where the authors use NFAs to model
adventure graphs.
In one of our projects we have let the students transform
temporal logic formulas representing constraints on these
graphs to NFAs. It would be very appealing if such a thing
could be added to JFLAP (especially with the nice pictures).

all the best,
Frank Neven

Granada, 05/23/06

Name: Rosa Mª Rodríguez Sánchez
Institution:E.T.S.I Informatica. University of Granada. Spain
Position: Faculty

I have been using JFLAP since four years ago. JFLAP is used as a tool for illustrating the basic of the computation models.

It helps to students to write the solutions for problems involving automatas, grammars and regular expressions, etc. Also the students are able to understand the relations among the different models.

Sincerely,
Rosa Rodriguez.

Rockford J. Ross
Computer Science Department
EPS 357
Montana State University
Bozeman, MT 59717-3880
June 11, 2006

Phone: 406-994-4804
Fax: 406-994-4376
E-mail: ross@cs.montana.edu
URL: www.cs.montana.edu/ross

To Whom It May Concern:

I am writing this letter in support of Dr. Susan Rodger's software system, JFLAP, for the Needs Premier Award 2006 competition. JFLAP is a software system that supports the teaching and learning of the theory of computing, the foundational subject of computer science.

First, I should introduce myself. I am Rocky Ross, Professor of Computer Science at Montana State University. I have known Susan since she first began working on JFLAP. I have been working in a similar area since the late 1970s and first met Susan at the SIGCSE (the Association for Computing Machinery's Special Interest Group in Computer Science Education) Symposium, the world's premiere conference for computer science education, where we were both presenting our related educational software systems. Since that time we have maintained a productive professional relationship. I have followed Susan's work closely and have served as an evaluator of her JFLAP project. I am also the editor of *SIGACT News*, the newsletter for the Association for Computing Machinery's Special Interest Group in Algorithms and Computation Theory, which is the primary professional society of theoretical computer scientists.

As already noted, JFLAP is a software system that supports the teaching and learning of the theory of computing. This subject is *the* foundation of the study of computer science, yet it is widely viewed by computer science students as the most dreaded and/or dry course in the curriculum due to its abstract, theoretical nature. In a nutshell, JFLAP aids the teaching and learning of many of the abstract concepts of the subject by brining them to life. That is, students are presented with computer-animated, visual depictions of the key abstract models of the theory, and they are allowed (and often required) to work with the models an active-learning mode as they progress through the material. The abstract models animated in JFLAP include deterministic and nondeterministic finite state automata, deterministic and nondeterministic pushdown automata, Turing machines, and grammars—the essential, key computational models of the theory of computing. In visualizing and animating these concepts, JFLAP effectively provides students with functionally correct mental models of these concepts with which they can interact until the concepts are understood. No static textbook presentation can match this kind of active learning environment.

I will not attempt to include a thorough analysis of JFLAP here. Instead I will highlight some of its main contributions and impacts.

1. JFLAP is the only currently generally distributed software that animates the theory of computing.
2. JFLAP is widely used and attracts more adherents each year worldwide.
3. JFLAP has garnered substantial support from the National Science Foundation for development and dissemination.
4. JFLAP has been published in numerous venues, including regular presentations at the annual SIGCSE Symposium, where it has consistently attracted attention.
5. Many instructors who are asked to teach the theory course, particularly at smaller institutions, are themselves somewhat unsure of the topic and find JFLAP a very nice tool to help illuminate relevant concepts.
6. There is a class of students who find it relatively easy to master abstract concepts and do not really need a system such as JFLAP to help them learn. Such students nonetheless find such animation tools engaging and are often the ones who can be utilized to extend the software (i.e., enhance JFLAP).
7. There is a much larger class of students who find it moderately to highly difficult to master abstract concepts. Many of these students are quite definitely helped by JFLAP. Perhaps the most appreciated aspect of JFLAP by these students is that they receive immediate and consistent feedback on exercises they perform within the software.
8. A general trend noticed from the use of JFLAP is that most students, regardless of their innate talent, are more motivated and excited about learning the theory of computing when JFLAP is incorporated into the course. This may actually be the most important contribution of JFLAP.
9. JFLAP is a complete system that comes with a manual, not just a "one-off" toy system that has no support.
10. JFLAP will continue to evolve and is anticipated to become an integral part of an established textbook (by Peter Linz) on the theory of computing.

All of these points substantiate the fact that JFLAP is a uniquely valuable software system for the teaching and learning of the theory that underlies computer science. Without doubt it meets the criterion of being "high quality, non-commercial software designed to enhance engineering education." I heartily recommend it for the Needs Premiere Award.

If you have any questions, feel free to contact me.

Sincerely,

Rockford J. Ross
Professor

Appreciation Letter is Support of

Java Formal Language and Automata Package (JFLAP)

To whom it may concern:

I am writing this letter of appreciation in support of JFLAP. I have not used JFLAP for a teaching purpose, but instead I used it as a tool for my research. The JFLAP in my research acted as a part of the interface to my platform and its designing section for automata is used as a feature and the data structure of the automata used in my platform. My system is an automated composition platform and JFLAP, more specifically automata and NFA design section of that is used as a part of the interface of my tool.

My first encounter to JFLAP was through the search in the internet, looking for an open source interface to automata. Finding few platforms doing so and testing them for a while, JFLAP comparing to other systems showed a very good and clean performance. I then went further to its documentations and papers, and found it a comprehensive and trustable tool to work with.

The source code of JFLAP is well developed and designed. Also lots of comments and documentations can be found within the code which helps understanding it much easier. The clean and well developed source code of JFLAP makes it well pleasant to work with and also give the ability to further customize and extend it according to your own needs and future works. The data structure of JFLAP is well designed and the hierarchy of grammars and languages is well reflected in an object oriented environment like Java. It also provides a lot of interesting features which can be used during the development and many famous algorithms for processing automata are well implemented in it. It covers all the aspects in the automata and formal languages theory and you can find things which fits to your needs within it. I am sure that with its beautiful and user friendly interface, students can surf within the automata world and experience all the features from their formal language courses. I wish the bests for Susan and all the students using this useful tool, both as a research or teaching material.

Therefore, I strongly recommend and support JFLAP from my successful experience with it. Please don't hesitate to contact me, should you require any further reference and information regarding my experience with JFLAP.

Best Regards,

Siamak Kolahi
Email: s_kolahi@cs.concordia.ca

Masters Student
Telecom Science Research Lab
Department of Computer Science
Concordia University
Sir George Williams Campus.
1515 St. Catherine W., EV010.139
Montreal, Quebec, Canada, H3G 2W1

Office: (514) 848-2424-7176
Home: (514) 570-3596

Letter of support to JFlap

name: Leonardo Mariani
institution: University of Milano Bicocca
position: Researcher
use of JFlap: Leonardo Mariani used JFlap during his PhD Thesis. Leonardo Mariani developed a technique, called BCT, for synthesizing models of component interactions from traces. One kind of synthesized model is a finite state automaton, which represents sets of legal invocation sequences. The implementation of BCT includes an inference engine and a visualization tool. JFlap has been successfully integrated within the BCT implementation to reuse its functionalities for handling and visualizing automata.
Leonardo Mariani began to use JFlap about at the beginning of the year 2004, and he is currently using the tool.

The experience of Leonardo Mariani with JFlap has been extremely positive for three main reasons:

1) JFlap implements a complete set of API for manipulating automata
2) JFlap code is implemented in a way that is simple to inspect, understand and use, so that people can easily design new applications that integrate with JFlap
3) The visualization tool allows to visualize models derived with any technique, in this case it has been used to visualize models obtained from the BCT inference engine

Since of the positive experience with this high quality and helpful tool, I recommend JFlap to obtain its software award.

Sincerely,
Leonardo Mariani

From: Diana von Bidder <diana.bidder@inf.ethz.ch>
To: Susan Rodger <rodger@cs.duke.edu>
Date: Fri, 02 Jun 2006 15:35:45 +0200
Subject: Re: Request letter in support of JFLAP?

Hi Susan

I'm interested in specification-based firewall testing. For this test cases need to be generated from Mealy Automata. As it is easier for people to specify automata graphically, we wanted to implement a tool which allows graphical specification of Mealy Automata and then computes test cases for these. It showed that JFLAP was an optimal starting point for this. The student, Stephan Hildenbrand, completing this work for me (Title of Thesis: Generation of Test Cases from Automata) found it very easy to extend JFLAP to Mealy Automata, and also to add the test generation code.

Greetings
Diana von Bidder
--
Diana von Bidder
Dipl. Informatik-Ing. ETH
Information Security Group, ETH Zurich
http://www.infsec.ethz.ch/people/dsenn

From: Deian Tabakov <dtabakov@rice.edu>
To: "Susan H. Rodger" <rodger@cs.duke.edu>
Date: Wed, 31 May 2006 08:36:19 -0500 (CDT)

Hi Susan,

Here you go:

Deian Tabakov, Ph.D. student from Rice University

I have been using JFlap to visualise automata as a part of my research. It proved to be quite useful in our Automata Theory class (Comp 482). The ability to draw automata quickly and in a WYSIWYG mode is a breather from tools like fig.

Cheers,

Deian