An Interactive and Visual Approach to Learning Computer Science

Susan H. Rodger

Department of Computer Science
Duke University

November 30, 2004

This work was supported by:
National Science Foundation DUE-9752583
Eli Lilly
CRA Distributed Mentor
Why Use an Interactive and Visual Approach?
Students Ready to Learn Automata Theory!
Things start well enough...
But soon, instead of pictures, there are **WORDS**.
Big words! The type with more than one syllable!
VIOLENCE AMONG STUDENTS AS NERVES FRAY!
We only wanted to learn automata theory! Isn’t there a better way?
Try JFLAP ...
Students Learning Automata with JFLAP
The Role of Visualization and Engagement

• Working Group ITiCSE 2002 (Naps et al)

• Six Levels of Learner Engagement
  1. No Viewing
  2. Viewing
  3. Responding
  4. Changing
  5. Constructing
  6. Presenting

• Hypothesis: 1 and 2 equivalent, higher the number, better learning outcomes
Outline

• JAWAA

• JFLAP
Outline - JAWAA

• JAWAA Editor
• JAWAA Use in Courses by Student
• JAWAA Use in Courses by Instructor
• Feedback on JAWAA
What is JAWAA?

- Scripting Language for Animation
- Easily create, modify and move objects
- Runs over the web, no need to install
- More Advanced Students
  - Output JAWAA Command from Program
  - Animate Data Structures Easily
- SIGCSE 2003 and SIGCSE 1998
- Students: Pierson, Patel, Finley, Akingbade, Jackson
Related Work

- **Samba, Jsamba** - Stasko (Georgia Tech)
- **AnimalScript** – Roessling (Darmstadt Univ of Tech, SIGCSE 2001)
- **JHAVE** – Naps (U. Wisc. Oshkosh, SIGCSE 2000)
JAWAA Commands

circle cl 30 20 60 blue red
moveRelative cl 60 0
moveRelative cl 0 50
changeParam cl bkgrd blue
JAWAA Primitives

circle
rectangle
line
oval
polygon
text

jawaa
JAWAA Data Structures

• Array

array people 25 25 4.2 Owen running Gail boating
           Robert toys Susan cakes vert red yellow black
changeParam people index on
changeParam people[1] bkgrd white
changeParam people[0].1 text bubblesort
moveRelative people[2] 30 0
changeParam people[2] swap people[0]
JAWAA Data Structures

• Stack

  stack s1 200 200 4 Pop The Top Off black red
  pop s1
  pop s1

• Queue

  queue q1 200 200 6 A 1 B 2 C 3 red blue
  dequeue q1
  dequeue q2
JAWAA Data Structures

• Linked List

• Trees
JAWAA Editor

- Easily create animations
- Graphically layout primitives
- Modify across time
- No knowledge of JAWAA
- Export to JAWAA file
- Start with JAWAA editor, finish with JAWAA output from program
Making an **Animation**
with the JAWAAA editor

Hello

Hello

Hello

Hello
JAWAA in Courses – Student Use

• In first-year seminar
• In CS 1 – novice/arrays
• In CS 2 – data structures
• In other CS courses
Student Use of JAWAA in CPS 49S – First-year Seminar

• Animation and Virtual Worlds Course
• JAWAA – 2-3 week unit
• No programming experience
• No interest in CS major
• Two tutorials – JAWAA/JAWAA editor
• Worked in pairs in class
  • Use JAWAA editor/type JAWAA commands
  • Modify web page/ run animation
CPS 49S – Student Animations

• Traffic Assignment
• Project: Cow and Bird
• Project: Duke Fast Break
Student Use of JAWAA in CS 1

- First Course for Majors
- Focus on Learning C++ (now Java)
- Early on – JAWAA is hidden in classes
  - Students use class
  - JAWAA code generated automatically
- Later, students easily animate array in one of their programs
  - Insertion, deletion, search
Student Use of JAWAA in CS 2

• Second course for majors
• Know C++ syntax (now Java)
• Add JAWAA commands to their program
• Easily animate data structures in their program
  • Arrays, queues, stacks, trees, graphs
  • Animations: trees, Josephus, word ladder
Student Use of JAWAA in CPS 140 Automata

• JAWAA used in any programming assignment

• Three part assignment – Interpreter for ROBOGO new robot language (Sp 2002)
  • Part 1 – Scanner
  • Part 2 – Parser
  • Part 3 – Syntax tree, interpret, animate with JAWAA

• Sample Robogo statements

```plaintext
robot fred 20 40
obstacle 20 45
move fred south 4
```
Instructor Use of JAWAA in CS 1/2

• Use JAWAA Editor to make quick animations for lecture
  • Fast - 4-8 minutes each animations, Fall 2002 CS 2 Course

• Create quick animation of data structure in an existing program, add JAWAA commands as output

• Show web pages with JAWAA animations in lecture

• Students replay animations later
Instructor Animations for CS 2 Lecture

• How Pointers Work in Memory
• Recursion
• Shellsort
• Linked List - Insert at the Front
• Quadratic Collision Resolution
• Build Heap and Heapsort
Evaluation/Use of JAWAA

- First-year Seminar
  - Spring 2001 – No JAWAA Editor
    - 50% students liked it
  - Fall 2002 – JAWAA Editor
    - 86% students liked it
    - Guinea pigs for this version
    - Given choice – chose JAWAA Editor
- CS 1 – Fall 2001 – several programming projects
- CS 2 – Fall 2002 – lecture, one prog. Project
- CPS 140 – Spring 2002 – one prog. project
Feedback from JAWAA Binary Trees – CS 2, Fall 2002

• Positive
  • “…assignment was cool because you could actually see an interesting display of what you had done…”
  • “After completing this assignment I am much more familiar with trees and their traversal.”
  • “I found it to be a powerful tool for visualizing code”
  • “Overall, I found the program interesting and I’d really like to use JAWAA more”
Feedback from JAWAA Binary Trees – CS 2, Fall 2002

• Negative

• “It only took me 2 hours to implement the binary tree stuff. The JAWAA implementation took 10 hours.”

• “Most of the time spent was spent on the minute details involved in the JAWAA animation rather than coding for the binary search tree functions.”

• Note: Trees are more tedious to do than using built-in data structures such as array, stack, queue, due to the placement of nodes.
JAWAA Editor, Nonmajors course

Spring 2001
No JAWAA Editor

Fall 2002
Using JAWAA Editor
Outline - JFLAP

• Why Develop Automata Tools?
• Previous Work
• What is JFLAP?
• Use of JFLAP by Instructor/Student
• Slide Show of Parts of JFLAP
• Feedback and Use Around World
• Future Work/Evaluation Study
Why Develop Tools for Automata?

Textual:

\[ (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\}) \]

\[ \delta = \{(q_0, b, q_0), (q_0, a, q_1), (q_1, a, q_0), (q_1, b, q_2), (q_2, a, q_1)\} \]

Tabular:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q_0</td>
<td>q_0</td>
<td></td>
</tr>
<tr>
<td>q_1</td>
<td>q_0</td>
<td></td>
</tr>
<tr>
<td>q_1</td>
<td>q_2</td>
<td></td>
</tr>
<tr>
<td>q_2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Visual:

![Automaton Diagram](image1)

Interactive:

![Automaton Diagram](image2)
Why Develop Tools for Automata?

Examined 10 Automata Textbooks

- One had software with book
- Only 6 had pictures of PDA, 2 or 3 states
- Only 6 had pictures of Turing machines, three of those switched representation
- Only 2 had picture of CFG to NPDA
- None had picture of parse tree for unrestricted grammar
Previous Work on Automata Tools by Others

Turing’s World Barwise and Etchemendy (1993)

Models of Computation – Taylor (1998), 7 models,
Deus Ex Machina by Savoiu

Snapshots – Ross (2002+)
Our Previous Work on Automata Tools

• JFLAP - creating and experimenting with automata and grammars (SIGCSEs 2004, 2000, 1999, 1997)

• JeLLRap - LL and LR parsing (SIGCSE 1997)

• Pâté - Brute force parser, grammar transformer (SIGCSE 1997)

• Lsys - creating L-systems

The new JFLAP incorporates concepts from all of these.
Thanks to Students Who Have Worked on JFLAP and Automata Theory Tools

- NPDA - 1990, C++, Dan Caugherty
- JFLAP - 1996-1999, Java version
  Eric Gramond, Ted Hung, Magda and Octavian Procopiuc
- Pâté, JeLLRap, Lsys
  Anna Bilska, Jason Salemme, Lenore Ramm, Alex Karweit, Robyn Geer
- JFLAP 4.0 – 2003, Thomas Finley, Ryan Cavalcante
What is JFLAP?

Java Formal Languages and Automata Package

Instructional Tool to learn concepts of Formal Languages and Automata Theory

<table>
<thead>
<tr>
<th>Regular languages - create</th>
</tr>
</thead>
<tbody>
<tr>
<td>- DFA</td>
</tr>
<tr>
<td>- NFA</td>
</tr>
<tr>
<td>- regular grammar</td>
</tr>
<tr>
<td>- regular expression</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Regular languages - conversions</th>
</tr>
</thead>
<tbody>
<tr>
<td>- NFA $\rightarrow$ DFA $\rightarrow$ Minimal DFA</td>
</tr>
<tr>
<td>- NFA $\leftrightarrow$ regular expression</td>
</tr>
<tr>
<td>- NFA $\leftrightarrow$ regular grammar</td>
</tr>
</tbody>
</table>
What is JFLAP? (cont)

CFL - create

- push-down automaton
- context-free grammar

CFL - transform

- PDA → CFG
- CFG → PDA (LL parser)
- CFG → PDA (LR parser)
- CFG → CNF
- CFG → LL parse table and parser
- CFG → LR parse table and parser
- CFG → Brute force parser
**What is JFLAP? (cont)**

<table>
<thead>
<tr>
<th>Recursively Enumerable languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Turing machine (1-tape)</td>
</tr>
<tr>
<td>• Turing machine (multi-tape)</td>
</tr>
<tr>
<td>• unrestricted grammar</td>
</tr>
<tr>
<td>• unrestricted grammar</td>
</tr>
<tr>
<td>→ brute force parser</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L-Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Create L-systems</td>
</tr>
</tbody>
</table>

---

- Turing machine (1-tape)
- Turing machine (multi-tape)
- unrestricted grammar
- unrestricted grammar
  → brute force parser
<table>
<thead>
<tr>
<th>Topic</th>
<th>JFLAP 3.1</th>
<th>JFLAP 4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite Automata</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Regular Grammars/Expressions</td>
<td>¾</td>
<td>1</td>
</tr>
<tr>
<td>Properties of Regular Languages</td>
<td>½</td>
<td></td>
</tr>
<tr>
<td>Context-Free Grammars</td>
<td>½</td>
<td>1</td>
</tr>
<tr>
<td>Simplify CFL</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Pushdown Automata</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Properties of CFL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turing Machine (1-Tape)</td>
<td>½</td>
<td>¾</td>
</tr>
<tr>
<td>Other Models of TM</td>
<td>¼</td>
<td>½</td>
</tr>
<tr>
<td>Recur. Enumerable Languages</td>
<td></td>
<td>¼</td>
</tr>
<tr>
<td>LL and LR Parsing</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>L-Systems</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
How JFLAP Fits Into Topics In Formal Languages Course Summary

• JFLAP 3.1 covers 4 chapters of material spread out over 6 chapters.

• JFLAP 4.0 covers 9 chapters of material spread out over 11 chapters.
Use of JFLAP by Instructor

Showing how to layout items

Poor:

Better:
Use of JFLAP by Instructor

Is this correct for $a^n b^n c^n$?

How do we fix it?
Use of JFLAP by Instructor
Experimenting with Difficult Concepts

Nondeterminism: $ww^R$

• Students attempt at desk - difficult: want to find the “middle”

• Instructor solves with class using JFLAP
Use of JFLAP by Instructor
Testing Student Programs
Use of JFLAP by Instructor

Relate to other CS Concepts

Running Time

• Consider $a^n b^n c^n$

• one-tape TM $O(n^2)$

• two-tape TM $O(n)$
Other Uses of JFLAP by Instructor

• Demonstrate Nondeterminism
• Demonstrate the running of a CFG to a PDA using LR method

Which lookahead do you choose?

• Demonstrate a transformation from one form to another

Example: PDA to CFG

• And many other uses...
JFLAP Student Use

• Recreate and experiment with instructor’s examples
• Use with Homework
• A study aid - create additional examples
  • explore concepts in depth
  • weaker students get more feedback
Some Features in JFLAP 4.0

• RE ↔ FA reworked
• Transform CF Grammars to Chomsky
• LL(1) and SLR(1) parsing
• Brute force parsing
• L-Systems
• Compare Equivalence
• 3-5 Tape Turing Machines
• Combine Automata
• Graph Layout
The most basic feature of JFLAP has always been the creation of automata, and simulation of input on automata.

Here we demonstrate the creation and simulation on a simple NFA.
FA Edit & Simulation
Start up JFLAP

• When we start up JFLAP we have a choice of structures.

• The first of these is the Finite Automata!
FA Edit & Simulation
Start Editing!

- We start with an empty automaton editor window.
FA Edit & Simulation
Create States

• We create some states ...
FA Edit & Simulation
Create Transitions

• We create some transitions ...
• We set an initial and final state.

• Now we can simulate input on this automaton!
• When we say we want to simulate input on this automaton, a dialog asks us for the input.
FA Edit & Simulation

Start Simulation!

• When simulation starts, we have a configuration on the initial state with all input remaining to be processed.
This is a nondeterministic FA, and on this input we have multiple configurations after we “Step.”
After Two Steps

The previous configurations on $q_1$ and $q_2$ are rejected, and are shown in red.

• The remaining uncolored configurations paths are not rejected, and are still open.
FA Edit & Simulation
After Three Steps

- Yet another step.
FA Edit & Simulation
After Four Steps

- One of the final configurations has been accepted!
• One can then see a traceback to see the succession of configurations that led to the accepting configuration.
New approach starts with a single RE transition in a GTG, and recursively breaks RE transitions into normal FA transitions until the GTG becomes an FA.
New algorithm transforms an FA to a GTG, and removes states until the GTG has only the initial and final states. At this point conversion becomes trivial.

\[(a+ab)(b+ab)^*a\]
Ambiguous Grammar Parsing with SLR

• One can also parse strings with grammars in JFLAP using LL(1) or SLR(1) parsing.

• To the right is a trivial, obviously ambiguous grammar.

• We show how SLR(1) deals with ambiguity.
SLR(1) Parsing

In order, students:

1. Define First and Follow Sets
2. Build the FA modeling stack.
3. Define the parse table.

Orange entries indicate a conflict in the parse table. Current active value is displayed.
SLR(1) Parsing

- Suppose we parse `aaba` with current conflicts both set to the default “reduce” entries.
- As students step, the parse table entry being used and grammar rule used (if a reduce) is highlighted.
- Notice also the input remaining and the stack.
SLR(1) Parsing

- Shown is the completed parse tree. Well done!
SLR(1) Parsing

- Recall the conflicts.
- When we click on the orange entry, we can choose a different entry to resolve the conflict.
- In this case we change the reduce operations to shift operations.
SLR(1) Parsing

- Notice, this change results in a very different parse tree.
SLR(1) Parsing

With Reduce Entries

With Shift Entries
Brute Force Parsing

• Brute force parsing allows both CFG and unrestricted grammar parsing.

• To the right is an unrestricted grammar that generates the language $a^n b^n c^n$.

• We can build the unrestricted parse tree!
Brute Force Parsing

- We parse the string \textit{aabbcc} with the brute force parser.

- Notice how in this case multiple nonterminal nodes are grouped together to form a single node.

- This accomplishes the unrestricted grammar possibly replacing multiple symbols at once.
L-Systems

• L-Systems may be used to model biological systems and create fractals.

• Similar to Chomsky grammars, except all variables are replaced in each derivation step, not just one!

• Commonly, strings from successive derivations are interpreted as strings of render commands and are displayed graphically.
L-Systems

- This L-System renders as a tree that grows larger with each successive derivation step.
L-Systems

- L-systems may also be stochastic.

- The $T \rightarrow Tg$ rule adds $g$ to the derivation, which draws a line segment.

- We add another rewriting rule for $T$, $T \rightarrow T$.

- With two rewriting rules for $T$, the rule chosen is random, leading to uneven growth!
L-Systems

The same stochastic L-system, rendered 3 different times all at the 9th derivation.
Compare for Equivalence

• Determine if two FA recognize same language
Multiple Tape Turing Machines

For example, with 3 tapes, you can relatively easily define a Universal Turing Machine.
Feedback – CPS 140 – Spring 2003

• Used JFLAP and tools in 6 of 9 homeworks

• Questionnaire – 33 responses
  • “Was JFLAP easy to use?” All 33 yes
  • “Did you look at the help at all? If so, what part did you look at and was it helpful?” 6 found it helpful, 27 didn’t look
  • “Do you prefer creating FA using JFLAP or drawing them on paper?”
  • 17 students – prefer to use JFLAP
  • 12 students – prefer paper first, then JFLAP for testing
  • 2 students – prefer paper
JFLAP’s Use Around the World

- JFLAP web page over 49,000 hits since 1996
- Google Search
  - JFLAP appears on over 4000 web pages
  - JFLAP appears on automata theory class webpages at over 40 US universities
- Note: search only public web pages
- Note: appears to be many foreign sites
- JFLAP has been downloaded over 14,000 times since Jan. 2003
- JFLAP appears in use (web pages or downloads) in over 40 countries
Um das JFLAP-Applet zu starten, hier klicken.

Lernumgebung Automatentheorie mit JFLAP

Was ist JFLAP?
JFLAP ist ein an der Universität Duke (USA) entwickeltes interaktives Lernprogramm, welches die Automatentheorie mit praktischen Anwendungen ergänzt. JFLAP ermöglicht es dem Anwender, bestehende Beispiele durchzuspielen, sowie eigene Automaten zu konstruieren.

Was bietet diese Lernumgebung?
Wir haben JFLAP um zusätzliche Komponenten erweitert, welche zusammen eine einheitliche Lernumgebung bilden. Diese Lernumgebung besteht aus folgenden Komponenten:

- **Theorie**
  - eine Einführung in die Automatentheorie mit
    - **Themen** (Schritt-für-Schritt)
    - **Index** (Stichwortverzeichnis)

- **Hilfe**
  - eine Hilfedokumentation und Einführung in des Programm JFLAP mit
    - **Themen** (Schritt-für-Schritt)
    - **Index** (Stichwortverzeichnis)

- **Übungen**
  - Übungen zu den einzelnen Automatentypen und zu den JFLAP-Funktionen (mit Lösungen)

- **JFLAP-Applet**
  - das eigentliche Programm JFLAP zum graphischen Konstruieren und Testen von Automaten
  - Einstiegern wird dringend empfohlen, zuerst die Help-Themen anzuschauen, bevor erste Schritte mit JFLAP unternommen werden!

- **Info**
  - (diese) Informationen für den Benutzer
Ingeniería Técnica de Informática de Gestión / Sistemas

Asignatura Bases de lenguajes de programación
Curso 2002/03
Práctica opcional nº 1: Introducción a la herramienta JFLAP

Objetivo
El objetivo de la práctica es que el alumno se familiarice con la herramienta JFLAP, orientada a la práctica visual e interactiva de los conceptos sobre lenguajes formales y teoría de autómatas. Mediante el uso de esta herramienta se practicarán operaciones relacionadas con gramáticas regulares, autómatas finitos y obtención del árbol de derivación en gramáticas independientes del contexto.

Obligatoriedad
La práctica no es obligatoria.

Prerequisitos
El alumno debe conocer los elementos relacionados con los niveles 2 y 3 de la jerarquía de Chomsky (lenguajes regulares, expresiones regulares, gramáticas regulares, autómatas finitos, lenguajes y gramáticas independientes del contexto.
Es recomendable un conocimiento elemental de manejo del sistema operativo Windows.

Descripción
A continuación se enuncian diferentes operaciones para experimentar con la herramienta JFLAP.
JFLAP 3.0

Eftersom konstruktionen av automater i JFLAP utgår från en grafisk representation som användaren ritar och inte från reguljära uttryck (eller, för den delen, andra formella språk som hanteras av av JFLAP), så finns behov av att inga mekanismer för att explicit manipulera finita maskiner. På det sättet är JFLAP anormlunda än de andra systemen beskrivna i den här rapporten. Jämförelsen mellan JFLAP och XFST enligt de punkter som finns i avsnitt 3.1 ser ut som följer:

1. JFLAP är avsett för utbildning och går antagligen inte att använda för utveckling av större system.
2. Det går att spara automater i textformat på disk för att senare läsa in dem i JFLAP igen.
3. Eftersom JFLAP inte kan användas på ett sätt som kräver att det ska vara effektivt så kan man anta att det inte är det (det finns inget i den medföljande dokumentationen som pekar åt det ena eller det andra härlet).
4. Abstraktionsnivån i gränssnittet är visserligen hög (automatena uttrycks som transitionsdiagram och högre än så kan väl knappast abstraktionsnivån bli i sammanhanget?), men eftersom funktionaliteten inte är speciellt utbyggd och syftet med verktyget ganska lågt från vad XFST klarar av, så har abstraktionsnivån ingen betydelse.
5. JFLAPs gränssnitt kan manipuleras på ett relativt deklarativt sätt.
JFLAP in Chinese
Future Work - Evaluation

• Study runs 2 years starting Fall 2005
• 11 University sites
• Survey students on usage of JFLAP
• Comparison with courses not using JFLAP
• Funding by National Science Foundation CCLI Program
Future Work - JFLAP

• Visualize Pumping Lemmas
• Building Blocks of Turing Machines
• More Graph Layout Algorithms
• Provide View of Configuration Tree
• Experiment with Closure Properties
• Include other models such as LBA
Future Work - Books

• JFLAP User manual out in Spring 2005 (Rodger, Finley)

• JFLAP Automata Theory textbook out in Spring 2006? (Rodger, Linz)

JFLAP is FREE!
Questions?