A Visual and Interactive Automata Theory Course with JFLAP 4.0

Ryan Cavalcante
Thomas Finley
Susan H. Rodger

Department of Computer Science
Duke University

March 4, 2004

This work was supported by:
National Science Foundation DUE-9752583
Outline

• Why develop Tools?
• Previous Work
• What is JFLAP?
• How Does JFLAP Fit In Automata Course?
• JFLAP's Use in the Course
• New Features of JFLAP
• Feedback and Use of JFLAP
• JFLAP's Future
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
Why Develop Tools for Automata Theory?

Different Representations of Presentation

<table>
<thead>
<tr>
<th></th>
<th>Textual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( ({q_0, q_1, q_2}, {a, b}, \delta, q_0, {q_2}) )</td>
</tr>
<tr>
<td></td>
<td>( \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)} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Tabular</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>\begin{array}{c} \hline a &amp; b \ q_0 &amp; q_1 &amp; q_0 \ q_1 &amp; q_0 \ q_2 &amp; q_2 \ \hline \end{array}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Visual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="Image1" alt="Diagram 1" /> <img src="Image2" alt="Diagram 2" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Interactive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="Image3" alt="Diagram 3" /> <img src="Image4" alt="Diagram 4" /></td>
</tr>
</tbody>
</table>
Why Develop Tools for Automata Theory?

Different Representations of Presentation Examined

10 automata theory textbooks in past 11 years (Taylor, Cohen, Sipser, Kelley, Linz, Sudkamp, Lewis, Kinber, Hopcroft, Martin)

- One had software integrated for part of the book (Taylor)
- All had visual representation of DFA
- Only 6 had visual representation for PDA
  Those had few visual examples, only 2 or 3 states
- Only 6 had visual representation for Turing machine
  3 of those switched different visual representation
- Only 2 had picture of NPDA created in CFG to NPDA transformation
- None had picture of parse tree for unrestricted grammar
## Previous Work on Automata Tools by Others

<table>
<thead>
<tr>
<th>Title</th>
<th>Author(s)</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turing's World - Barwise and Etchemendy</td>
<td>(1993)</td>
<td></td>
</tr>
<tr>
<td>Deus Ex Machina - Savoiu (seven models of computation)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Models of Computation and Formal Languages</td>
<td>Taylor</td>
<td>(1998)</td>
</tr>
</tbody>
</table>
Our Previous Work on Automata Tools

• JFLAP - creating and experimenting with automata and grammars
  (SIGCSE 2000, SIGCSE 1999, SIGCSE 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 Caugherity

- FLAP - 1991, C++
  Mark LoSacco, Greg Badros

- 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

| Regular languages - create | ![Diagram]
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• DFA</td>
<td>q0 → a → q1 → a → q2 → a, b → q1, 0, 1, 2</td>
</tr>
<tr>
<td>• NFA</td>
<td>q0 → a → q1 → a → q2 → a, b → q1, 0, 1, 2</td>
</tr>
<tr>
<td>• regular grammar</td>
<td></td>
</tr>
<tr>
<td>• regular expression</td>
<td></td>
</tr>
</tbody>
</table>

| Regular languages - conversions | ![Diagram]
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• NFA → DFA → Minimal DFA</td>
<td>q0 → a → q1 → a → q2 → a, b → q1, 0, 1, 2</td>
</tr>
<tr>
<td>• NFA ↔ regular expression</td>
<td></td>
</tr>
<tr>
<td>• NFA ↔ regular grammar</td>
<td></td>
</tr>
</tbody>
</table>
What is JFLAP? (cont)

Context-free languages - create

- push-down automaton
- context-free grammar

Context-free languages - transform

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

<table>
<thead>
<tr>
<th>Recursively Enumerable languages</th>
<th><img src="image.png" alt="Turing machine multi-tape" /></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Turing machine (1-tape)</td>
<td></td>
</tr>
<tr>
<td>• Turing machine (multi-tape)</td>
<td></td>
</tr>
<tr>
<td>• unrestricted grammar</td>
<td></td>
</tr>
<tr>
<td>• unrestricted grammar</td>
<td></td>
</tr>
<tr>
<td>→ brute force parser</td>
<td></td>
</tr>
</tbody>
</table>

**L-Systems**

- Create L-systems
## How JFLAP Fits Into Topics In Formal Languages Course

<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>$\frac{3}{4}$</td>
<td>1</td>
</tr>
<tr>
<td>Properties of Regular Languages</td>
<td></td>
<td>$\frac{1}{2}$</td>
</tr>
<tr>
<td>Context-Free Grammars</td>
<td>$\frac{1}{2}$</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>$\frac{1}{2}$</td>
<td>$\frac{3}{4}$</td>
</tr>
<tr>
<td>Other Models of TM</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{2}$</td>
</tr>
<tr>
<td>Recursively Enumerable Languages</td>
<td>$\frac{1}{4}$</td>
<td></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 Programmes
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
Feedback from Students in CPS 140
Duke University - Spring 2003

- Used JFLAP and tools in 6 of 9 homeworks
- Questionnaire - 33 responses
  - “Was JFLAP easy to use?”
    All 33 answered yes.
  - “Did you look at the help at all? If so, what part did you look at and was it helpful?”
    6 used help and found it helpful, 27 did not look at help.
  - “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
New Features in JFLAP 4.0

- RE ↔ FA reworked
- Transform CF Grammars to Chomsky
- LL(1) and SLR(1) parsing
- Brute force parsing
- L-Systems

Minor New Features in JFLAP 4.0

- Compare Equivalence
- 3-5 Tape Turing Machines
- Combine Automata
- Graph Layout
Finite Automata Editing and Simulation

- 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.
• 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.
• 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.”
FA Edit & Simulation
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.
RE to FA

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

Determines if two FA recognize the same language.

They ARE equivalent!
Multiple Tape Turing Machines

For example, with 3 tapes, you can relatively easily define a Universal Turing Machine.
JFLAP’s Use Around the World

- JFLAP web page has over 42,000 hits since 1996
- Google Search
  - JFLAP appears on over 2300 web pages
  - JFLAP appears on automata theory class webpages at over 40 US universities
- Note: search only shows public web pages
- Note: appears to be many foreign sites
- JFLAP has been downloaded over 8000 times since Jan. 2003
- JFLAP appears in use (web pages or downloads) in over 40 countries
Future Work

• Filling in the missing pieces for automata theory course

• JFLAP User manual out in December 2004 (Rodger, Finley)

• JFLAP Automata Theory textbook out in Fall 2005 (Rodger, Linz)

JFLAP is FREE!