An Interactive Approach to Formal Languages and Automata with JFLAP

Susan H. Rodger
Duke University

NSF CCLI Showcase
March 9, 2007

Supported by NSF Grant DUE 0442513.
Outline

• Overview of JFLAP
• Examples and Demo
  – L-Systems
  – Turing Machine Building Blocks
  – Moore and Mealy Machines
  – Pumping Lemma
  – Batch Testing Mode
• JFLAP’s use in Teaching
• JFLAP Study
Formal Languages and Automata Theory

• Traditionally taught
  – Pencil and paper exercises
  – No immediate feedback

  – More mathematical than most CS courses
  – Less hands-on than most CS courses
Why study finite automata?

- Application: Compiler
- Compiler identifies your syntax errors
- Can write a big DFA to identify all words in a Java program
  - integers, doubles, boolean
  - keywords, variable names
  - arithmetic operators, punctuation symbols
## Why Develop Tools for Automata?

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

| Tabular  | \[
|---------|---------------------------------------------------------------|
|         | \[ \begin{array}{c|cc}
| q_0 & a & b \\
| q_1 & q_0 & q_0 \\
| q_2 & q_2 & \\
| \end{array} \] |

| Visual   | ![Visual Diagram](image1.png) |

| Interactive | ![Interactive Diagram](image2.png) |
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
Overview of JFLAP

• **Java Formal Languages and Automata Package**

• Instructional tool to learn concepts of Formal Languages and Automata Theory

• **Topics:**
  – Regular Languages
  – Context-Free Languages
  – Recursively Enumerable Languages
  – Lsystems
Thanks to Students - Worked on JFLAP and Automata Theory Tools

• NPDA - 1990, C++, Dan Caugherty
• 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
• JFLAP 6.0 – 2005-2006 Stephen Reading, Bart Bressler, Jinghui Lim
JFLAP – Regular Languages

• Create
  – DFA and NFA
  – Moore and Mealy
  – regular grammar
  – regular expression

• Conversions
  – NFA to DFA to minimal DFA
  – NFA $\leftrightarrow$ regular expression
  – NFA $\leftrightarrow$ regular grammar
JFLAP – Regular languages (more)

- Simulate DFA and NFA
  - Step with Closure or Step by State
  - Fast Run
  - Multiple Run
- Combine two DFA
- Compare Equivalence
- Brute Force Parser
- Pumping Lemma
JFLAP – Context-free Languages

• Create
  – Nondeterministic PDA
  – Context-free grammar
  – Pumping Lemma

• Transform
  – PDA $\rightarrow$ CFG
  – CFG $\rightarrow$ PDA (LL & SLR parser)
  – CFG $\rightarrow$ CNF
  – CFG $\rightarrow$ Parse table (LL and SLR)
  – CFG $\rightarrow$ Brute Force Parser
JFLAP – Recursively Enumerable Languages

• Create
  – Turing Machine (1-Tape)
  – Turing Machine (multi-tape)
  – Building Blocks
  – Unrestricted grammar

• Parsing
  – Unrestricted grammar with brute force parser
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.
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 ...
FA Edit & Simulation
Initial and Final State

• We set an initial and final state.
• Now we can simulate input on this automaton!
FA Edit & Simulation
Input to Simulate...

• 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."
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!
FA Edit & Simulation

Traceback

• One can then see a traceback to see the succession of configurations that led to the accepting configuration.
FA Multiple Run

• Select Multiple Run
• One can then enter many strings and receive acceptance info.

<table>
<thead>
<tr>
<th>Input</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Accept</td>
</tr>
<tr>
<td>aa</td>
<td>Accept</td>
</tr>
<tr>
<td>aab</td>
<td>Accept</td>
</tr>
<tr>
<td>aabb</td>
<td>Accept</td>
</tr>
<tr>
<td>acb</td>
<td>Reject</td>
</tr>
<tr>
<td>abcbb</td>
<td>Accept</td>
</tr>
<tr>
<td>abbcc</td>
<td>Accept</td>
</tr>
<tr>
<td>abcab</td>
<td>Reject</td>
</tr>
<tr>
<td>bc</td>
<td>Reject</td>
</tr>
</tbody>
</table>
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.
Students like L-systems
Turing Machine Building Blocks

• First, a problem.
• \( f(w) = \text{number of a’s in } w, \sum = \{a, b\} \)
• Examples:
  – \( f(aabab) = 111 \)
  – \( f(bbbaab) = 11 \)
Turing Machine Building Blocks

• Building Block
  – Build a Turing machine with a specific purpose
  – Name it and save it
  – Use it as a BlackBox in another Turing machine

• Special Symbols
  ~ ignore read or write
  !x matches all symbols except for x
Simple Building Blocks

- start
- R – move right
- R_blknk – move right once, keep moving right until reach a blank
Simple Building Blocks (cont)

• **R_not_a** – move right once, keep moving until not an “a”

![Diagram of R_not_a state transition](image1)

• **a** - write “a” and stay put

![Diagram of a state transition](image2)
Create & Combine Building Blocks

- **New Buttons**

- **Conditional** – if the current symbol is \( b \), move to the next block (tape head not moving)

- **Move to the next block** – use \( \sim \)
  - Ignore read, ignore write, stay put
Problem again: Count number of a’s

- $F(abbaabb) = 111$
Building Block Run Choices
abbab

Step by building block
abbab
abbab
abbab
abbab
abbab
abbab

- Skip a few steps to ...
Mealy Machines

• Similar to finite automata
  – No final states
  – Produce an output on their transitions
  – deterministic
Example – Vending Machine

• Dispenses candy once enough money has been inserted
  – Money – n(nickel), d(dime) q(quarter)
  – Candy bars – 20 cents
  – Returns the appropriate amount of change – the number of nickels
  – C4 means “candy and 4 nickels”

• From Carroll and Long’s *Theory of Finite Automata* book
Mealy Vending Machine Example
Moore Machine

- Similar to Mealy Machine
  - No final state
  - Output is produced by states, not transitions
Example – Halve a Binary Number

<table>
<thead>
<tr>
<th>Input</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>0010</td>
</tr>
<tr>
<td>11111</td>
<td>001111</td>
</tr>
<tr>
<td>01001</td>
<td>000100</td>
</tr>
<tr>
<td>100</td>
<td>0010</td>
</tr>
<tr>
<td>110</td>
<td>0011</td>
</tr>
<tr>
<td>1</td>
<td>00</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
</tr>
</tbody>
</table>
Regular Pumping Lemma

Pumping Lemma: Let $L$ be an infinite regular language. \( \exists \) a constant \( m > 0 \) such that any \( w \in L \) with \( |w| \geq m \) can be decomposed into three parts as \( w = xyz \) with

\[
|xy| \leq m \\
|y| \geq 1 \\
xy^iz \in L \text{ for all } i \geq 0
\]
Pick an Example

<table>
<thead>
<tr>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L = {a^n b^n : n \geq 0}$</td>
<td></td>
</tr>
<tr>
<td>$L = {w \in {a, b}^* : n_a(w) &lt; n_b(w)}$</td>
<td></td>
</tr>
<tr>
<td>$L = {w w^R : w \in {a, b}^*}$</td>
<td></td>
</tr>
<tr>
<td>$L = {a^n : n \geq 0}$</td>
<td></td>
</tr>
<tr>
<td>$L = {(ab)^n a^k : n &gt; k, k \geq 0}$</td>
<td></td>
</tr>
<tr>
<td>$L = {a^n b^k c^{n+k} : n \geq 0, k \geq 0}$</td>
<td></td>
</tr>
<tr>
<td>$L = {a^n b^l a^k : n &gt; 5, l &gt; 3, k \leq l}$</td>
<td></td>
</tr>
<tr>
<td>$L = {a^n : n \geq 2, n \text{ is a prime number}}$</td>
<td></td>
</tr>
<tr>
<td>$L = {a^n : n \text{ is even}}$</td>
<td></td>
</tr>
</tbody>
</table>
User enters in steps 1 and 3
Context-Free Pumping Lemma

Pumping Lemma for CFL’s Let $L$ be any infinite CFL. Then there is a constant $m$ depending only on $L$, such that for every string $w$ in $L$, with $|w| \geq m$, we may partition $w = uvxyz$ such that:

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

$L = \{a^n b^n c^n : n \geq 0\}$ Context-Free Pumping Lemma

Messages

I WIN. Do you want to play again or concede that the language is not...
I have selected $i$ to give a contradiction. It is displayed in Box 4.
Click "Step" in Box 5 to step the animation or "Add" in the right pan...

1. Select integer $m$
   
   $5$

2. Given integer $m$, here's string $w$ such that $|w| \geq m$
   
   aaaaaabbbbbcccc

3. Select decomposition of $w$ into $uvwxyz$
   
   $u$: aa, $|u|$: 2
   $v$: a, $|v|$: 1
   $x$: a, $|x|$: 1
   $y$: abb, $|y|$: 3
   $z$: bbbcccc, $|z|$: 8

4. A choice of $i$ to give contradiction
   
   $i$: 2, pumped string: aaaaaaabbbbbbcccccc
CFL pump lemma (cont)

- Last step shows the contradiction

- In CFL – there are lots of cases to consider
Batch Testing Mode

• Select several files for testing

• Then select input file
### Batch Testing Mode (cont)

<table>
<thead>
<tr>
<th>File</th>
<th>Input</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>fa1.jff</td>
<td>aabbaab</td>
<td>Reject(Accept)</td>
</tr>
<tr>
<td>fa1.jff</td>
<td>aaaaa</td>
<td>Reject(Accept)</td>
</tr>
<tr>
<td>fa1.jff</td>
<td>aab</td>
<td>Accept</td>
</tr>
<tr>
<td>fa1.jff</td>
<td>aabbaaab</td>
<td>Accept(Reject)</td>
</tr>
<tr>
<td>fa2.jff</td>
<td>aabbaabb</td>
<td>Reject(Accept)</td>
</tr>
<tr>
<td>fa2.jff</td>
<td>aaaa</td>
<td>Reject(Accept)</td>
</tr>
<tr>
<td>fa2.jff</td>
<td>aab</td>
<td>Accept</td>
</tr>
<tr>
<td>fa2.jff</td>
<td>aabbaaab</td>
<td>Accept(Reject)</td>
</tr>
<tr>
<td>fa5.jff</td>
<td>aabbaabb</td>
<td>Accept</td>
</tr>
<tr>
<td>fa5.jff</td>
<td>aaaaa</td>
<td>Accept</td>
</tr>
<tr>
<td>fa5.jff</td>
<td>aab</td>
<td>Accept</td>
</tr>
<tr>
<td>fa5.jff</td>
<td>aabbaaab</td>
<td>Accept(Reject)</td>
</tr>
</tbody>
</table>
Using JFLAP in Teaching
Using JFLAP during Lecture

• Use JFLAP to build examples of automata or grammars
• Use JFLAP to demo proofs
• Load a JFLAP example and students work in pairs to determine what it does, or fix it if it is not correct.
Example: JFLAP during Lecture

• Ask students to write on paper an NPDA for palindromes of even length
• Build one of their solutions using JFLAP
  – Shows students how to use JFLAP
• Run input strings on the NPDA
  – Shows the nondeterminism
Example 2: JFLAP during Lecture

- Brute Force Parser
  - Give a grammar with a lambda-production and unit production
  - Run it in JFLAP, see how long it takes (LONG)
  - Is aabbab in L?
  - Transform the grammar to remove the lambda and unit-productions
  - Run new grammar in JFLAP, runs much faster!
Parse Tree Results

- First Grammar – 1863 nodes generated
- Second Grammar – 40 nodes generated
- Parse tree is the same.
With JFLAP, Exploring Concepts too tedious for paper

• Load a Universal Turing Machine and run it
• See the exponential growth in an NFA or NPDA
• Convert an NPDA to a CFG
  – Large grammar with useless rules
  – Run both on the same input and compare
  – Transform grammar (remove useless rules)
JFLAP’s use Outside of Class

• Homework problems
  – Turn in JFLAP files
  – OR turn in on paper, check answers in JFLAP
• Recreate examples from class
• Work additional problems
  – Receive immediate feedback
Ordering of Problems in Homework

- Order questions so they are incremental in the usage of JFLAP
  1. Load a DFA. What is the language?
     *Students only enter input strings.*
  2. Load a DFA that is not correct. What is wrong? Fix it.
     *Students only modifying a small part.*
  3. Build a DFA for a specific language.
     *Last, students build from scratch.*
JFLAP Study

- Study of JFLAP’s effectiveness in learning
  - Runs 2005-2007
  - Pretest/Posttest
  - Interviews
- Supported by National Science Foundation, grant NSF DUE 0442513
Fourteen Participants

• Duke
• UNC-Chapel Hill
• Emory
• Winston-Salem State University
• United States Naval Academy
• Rensselaer Polytechnic Institute
• UC Davis
• Virginia State University
• Norfolk State University
• University of Houston
• Fayetteville State University
• University of Richmond
• San Jose State University
• Rochester Institute of Technology
JFLAP’s Use Around the World

• JFLAP web page has over 110,000 hits since 1996

• Google Search
  – JFLAP appears on over 20,000 web pages
  – Note: search only public web pages

• JFLAP been downloaded in over 160 countries
Questions?

• JFLAP is free!

• www.jflap.org

• JFLAP book (Jones & Bartlett, 2006)
  – Use as supplement to a textbook