Through Visualization and Interaction, Computer Science Concepts Come Alive

Susan H. Rodger
Duke University
rodger@cs.duke.edu

CWIC 2012
Columbia, SC
Feb. 17, 2012

Supported by NSF Grants DUE-0442513, ESI-0624642, DRL-1031351, DUE 10441-91, CRA, and IBM Faculty Awards
A bit about me, my background...

PhD, 1989
Computer Science

Assistant Prof.
1989-1994

Professor of the Practice
1994-present

Duke University
About Me - Research Interests

• Computer Science Education
• Visualization and Interaction
  – Instructional Tools for Theoretical concepts
    • Automata theory and formal languages
  – Teaching Introductory Computer Science
• Algorithm Animation
Outline

• Introduction

• CS Concepts Come Alive with Software
  – Automata Theory with JFLAP
  – Algorithm Animation software – JAWAA and others
  – Pre-CS 1 with Alice
  – CS Concepts Come Alive in other ways

• Challenges in Designing Educational Software

• Integrating Computer Science into K-12
Intro - Why Use Interaction and Visualization?

• Learning Styles
  – Visual Learners
    • Learn through seeing
    • Learn best from visual displays
  – Auditory Learners
    • Learn through listening
    • Learn best through verbal lectures, discussions
  – Kinesthetic Learners
    • Learn through moving, doing and touching
    • Learn best through hands-on approach
How do you reach all three types?

• You must do all three!
  – Provide pictures, diagrams
  – Discuss what you are doing
  – Provide activities for trying it
Outline

• Introduction

• CS Concepts Come Alive with Software
  – Automata Theory with JFLAP
  – Algorithm Animation software – JAWAA and others
  – Pre-CS 1 with Alice
  – CS Concepts Come Alive in other ways

• Challenges in Designing Educational Software

• Integrating Computer Science into K-12
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 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></td>
<td>q_2</td>
</tr>
<tr>
<td>q_2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Visual**

![Visual representation of automata]

**Interactive**

![Interactive representation of automata]
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

• **With JFLAP your creations come to life!**
Thanks to Students - 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
- JFLAP 6.0 – 2005-2008 Stephen Reading, Bart Bressler, Jinghui Lim, Chris Morgan, Jason Lee
- JFLAP 7.0 - 2009 Henry Qin, Jonathan Su
- JFLAP 8.0? – 2011-12 Julian Genkins

Over 20 years!
Example

• Build a deterministic finite automaton (DFA) to recognize binary numbers with an even number of 1s that are an even number.
• Only use symbols 0 and 1
• Binary numbers: 0, 1, 10, 11, 100, 101, 110, 111, ...
• When is a binary number an even number?
  – Ends in 0
• Which strings should be accepted?
• 11010, 10010, 1111, 10100
• no     yes     no     no     yes
A solution
Give meaning to states

- **q0**: even number of 1's, end in 1
- **q1**: odd number of 1's
- **q2**: even number of 1's, end in 0
- **q3**: only one zero
- **q4**:
Comes to Life!

q3
odd number of 1's

q1

q0
even number of 1's, end in 1
even number of 1's, end in 0

q2

q4
only one zero

1010

Simulate: 1010
Comes to Life!
What are DFAs useful for?

• When you write a program – you compile it before you run it
• How does it identify syntax errors?

• First phase of a compiler is a DFA that models and identifies every word in your program
  – 7648 is an integer
  – if is a keyword
  – + is a operator
  – item is a variable name

• Write one big DFA that identifies every word
What does this DFA recognize?
DFA annotated and w/ shortcut

- \( q0 \) to \( q1 \) with \( 0 \)
- \( q0 \) to \( q2 \) with \( \lambda \)
- \( q2 \) to \( q3 \) with \([1-9]\)
- \( q3 \) with \([0-9]\)

- \( q0 \) to negative or not
- \( q3 \) to at least one non-zero
What else can JFLAP do?

• Create other machines
  – Moore and Mealy
  – Pushdown Automaton
  – Turing machine

• Parsing of grammars
  – regular, context-free grammars
  – Unrestricted grammar

• Conversions for proofs
  – NFA to DFA to minimal DFA
  – NFA $\leftrightarrow$ regular expression
  – NFA $\leftrightarrow$ regular grammar
  – CFG $\leftrightarrow$ NPDA
JFLAP - 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.
JFLAP - 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.
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.
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
JFLAP’s Use Around the World

• JFLAP web page has over 250,000 hits since 1996
• Google Search
  – JFLAP appears on over 39,000 web pages
  – Note: search only public web pages
• JFLAP now used in several textbooks – JFLAP exercises
• JFLAP been downloaded in over 160 countries
Two-year JFLAP Study 2005-2007

Fourteen Faculty Adopter Participants

- small, large
- public, private
- includes minority institutions

- 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
Conclusions From Study

• Results of Study showed
  – All the faculty used JFLAP in their courses, mostly for homework, some in lecture
  – Students had a high opinion of JFLAP
  – Majority of students felt access to JFLAP
    • Made learning course concepts easier
    • Made them feel more engaged
    • Made the course more enjoyable
  – Over half the students used JFLAP to study for exams
  – Over half the student thought time and effort using JFLAP helped them get a better grade.
JFLAP is free

www.jflap.org

JFLAP tutorial
Outline

• Introduction
• CS Concepts Come Alive with Software
  – Automata Theory with JFLAP
  – Algorithm Animation software – JAWAA and others
  – Pre-CS 1 with Alice
  – CS Concepts Come Alive in other ways
• Challenges in Designing Educational Software
• Integrating Computer Science into K-12
JAWAA
Java and Web-based Algorithm Animation

- 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
- www.cs.duke.edu/~rodger/tools/
- Students: Pierson, Patel, Finley, Akingbade, Jackson, Gibson, Gartland
Related Work

• Samba, Jsamba - Stasko (Georgia Tech)
• AnimalScript – Roessling (Darmstadt Univ of Tech, SIGCSE 2001)
• JHAVE – Naps (U. Wisc. Oshkosh, SIGCSE 2000)
• TRAKLA2 – Software Visualization Group – TKK Finland
• AlgoViz portal – lots of animations!
• Lots of animations and systems on the web!
### JAWAA Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>circle cl 30 20 60 blue red</code></td>
<td></td>
</tr>
<tr>
<td><code>moveRelative cl 60 0</code></td>
<td>move right</td>
</tr>
<tr>
<td><code>moveRelative cl 0 50</code></td>
<td>move down</td>
</tr>
<tr>
<td><code>changeParam cl bkgrd blue</code></td>
<td></td>
</tr>
<tr>
<td>JAWAA Primitives</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td><strong>circle</strong></td>
<td><img src="image" alt="Circle" /></td>
</tr>
<tr>
<td><strong>rectangle</strong></td>
<td><img src="image" alt="Rectangle" /></td>
</tr>
<tr>
<td><strong>line</strong></td>
<td><img src="image" alt="Line" /></td>
</tr>
<tr>
<td><strong>oval</strong></td>
<td><img src="image" alt="Oval" /></td>
</tr>
<tr>
<td><strong>polygon</strong></td>
<td><img src="image" alt="Polygon" /></td>
</tr>
<tr>
<td><strong>text</strong></td>
<td><img src="image" alt="Text" /></td>
</tr>
</tbody>
</table>
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]
```

```
0
  Robert
toys

1
  Gail
boating

2
  Owen
bubblesort

3
  Susan
cakes
```
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
depqueue q1
depqueue q2
```

```
3C2B1A 3C2B1 3C2B
```
JAWAA Data Structures

• Linked List

• Trees
Use of JAWAA in CS 1/2

• Instructor
  – Use JAWAA Editor to make quick animations for lecture
  – Show web pages with JAWAA animations in lecture
  – Students replay animations later

• Student
  – Create animation of data structure in an existing program, add JAWAA commands as output
Instructor Animations for CS 2 Lecture

- Recursion
- Shellsort
- Depth First Search
- Quadratic Collision Resolution
- Build Heap and Heapsort
Lots of other software/programs for algorithm animation

- **Red Black Tree – animation on web page**

http://aleph0.clarku.edu/~achou/cs102/examples/bst_animation/RedBlackTree-Example.html
Another red-black tree animation

1. Search (top-down) and insert the new item $u$ as in Binary Search Tree.
2. Return (bottom-up) and
2.1 If $u$ is root, make it black and the algorithm ends or
2.2 if its parent $t$ is black, the algorithm ends
2.3 If both $u$ and its parent $t$ are red, do one of the following:
2.3.1. [change colors] If $t$ and its sibling $v$ are red, change colors: change $t$ and $v$ black and their parent $p$ red. Continue the algorithm in $p$ if necessary.
Outline

• Introduction

• CS Concepts Come Alive with Software
  – Automata Theory with JFLAP
  – Algorithm Animation software – JAWAA and others
  – Pre-CS 1 with Alice
  – CS Concepts Come Alive in other ways

• Challenges in Designing Educational Software

• Integrating Computer Science into K-12
Alice Programming Language

• Create interactive stories or games
• Learn programming in an easy way, drag-and-drop your code
• Problem solving with visual feedback
  – Objects are visual!
• Alice is free: www.alice.org
• Developed by Randy Pausch
Alice Programming Language

- Has libraries of 3D objects

- Keeps Track of objects you select
Objects Have Multiple Parts that are moveable
Alice Code is Easy to Learn

Select Code, Drag-and-Drop code in program
Computer Science Concepts come alive with Alice - Examples

- Objects are visible
- Variables
- Inheritance
- List
- Array
Variables – Scores/Timers

Game: Eragon

4 tasks to win the game
Example - Inheritance

• Start with a chicken object
• Rename it to 
  TalentedChicken
  – Change its color
  – Resize it larger
  – Add new methods (jump, fly, scurry)
  – Add events for this chicken
• Save this new class 
  TalentedChicken that inherits from the Chicken class
Example - List

The Alice Team Summer 2008
Example – Arrays
Shuffle, then Selection Sort

Sort by height
Outline

• Introduction
• CS Concepts Come Alive with Software
  – Automata Theory with JFLAP
  – Algorithm Animation software – JAWAA and others
  – Pre-CS 1 with Alice
  – CS Concepts Come Alive in other ways
• Challenges in Designing Educational Software
• Integrating Computer Science into K-12
Interaction in Class – Props
Edible Turing Machine

- TM for \( f(x) = 2x \) where \( x \) is unary

- TM is not correct, can you fix it? Then eat it!

- States are blueberry muffins
Students building DFA with cookies and icing
Red-Black Tree (cookies)
Outline

• Introduction
• CS Concepts Come Alive with Software
  – Automata Theory with JFLAP
  – Algorithm Animation software – JAWAA and others
  – Pre-CS 1 with Alice
  – CS Concepts Come Alive in other ways
• Challenges in Designing Educational Software
• Integrating Computer Science into K-12
Make your tool as interactive as possible – but not too tedious!

- User shouldn’t type everything
- Sometimes select
Allow user to proceed on if they got it

- Complete the rest for them

- Complete parts for them
Avoid Too Many Pop up windows

- OLD JFLAP LR PARSE TOOL
Add Pause/Checkpoint questions

• Allow for pause to think about what comes next
• Undo/go back

• Pop up a quiz question to see if the user understands what he/she just did
  – JHAVE tool does this
What can make the tool more useable?

- Annotations on states
- Multiple run window
  - Develop test data
  - Easier for grading
- General definitions
  - FA – recognize one or more symbols
  - NPDA – pop or push 0 or more symbols
- Batch processing
Naming your software

What is a “good” name for your tool?
JAWAA name is not unique

How popular is JAWAA?
JFLAP name is unique
Outline

• Introduction

• CS Concepts Come Alive with Software
  – Automata Theory with JFLAP
  – Algorithm Animation software – JAWAA and others
  – Pre-CS 1 with Alice
  – CS Concepts Come Alive in other ways

• Challenges in Designing Educational Software

• Integrating Computer Science into K-12
Engaging Middle School Teachers and Students with Alice in a Diverse Set of Subjects

Supported by the National Science Foundation Collaborative Grant ESI-0624642, 0624654, 0624528, NSF Supplement DRL-0826661, two CRA distributed mentor awards, and two Faculty Awards from International Business Machines.
Problem – Few students major in CS

- Students come to college with their mind made up on their career! This choice is based on what they know.
- Students don’t know what computer science is when in middle and high school
- In college, like Alice, but not staying with computer science
Where could Alice help in decisions?

- Students in middle school are starting to form decisions on careers
- They have exposure to Teachers, Doctors, Astronauts, etc.
- They learn about Biology, Physics, Chemistry
  - BUT DON’T KNOW WHAT COMPUTER SCIENCE IS
  - THEIR EXPOSURE is SPREAD SHEETS, POWERPOINT, etc.
Bring on Alice Virtual Worlds!

• Alice is
  – Hands-on!
  – Interactive!
  – Visual!
  – Less Error prone
  – Exciting Results right away!

• Alice has the potential to excite kids about computer science in the same way that experiments excite kids about chemistry, physics and biology!
Adventures in Alice Programming

• Summers 2008-2015
• 3-week Teacher workshops
  • Over 150 teachers, mostly middle school, some high school
  • All disciplines
  • Taught them Alice, Developed Lesson Plans

– 1-week middle school camps

• Sites:
  – Durham, NC
  – Charleston/Columbia, SC
  – Oxford, Mississippi
How to Use Alice in Middle/High Schools

• Teachers
  – Examples in lecture
  – Make interactive quizzes
  – Make worlds on concepts for students to view

• Students
  – Projects (in place of a poster, a model)
  – To take or build quizzes
  – To view and answer questions about a world
  – Older students can do more with Alice.
Examples of integrating Alice into K-12
Language Arts – Animate a story

By Betty Stone
Animated by Deborah Nelson

KITTY STORY
Science Example
How a volcano is formed
Math Example:
Teacher Lesson Plan on quadrant plane

- Click on lighthouse
- Enter x,y position
- Objects randomly move
Math Example – Rounding Numbers

Rounding World

Choose the level of difficulty by clicking on the handle:

- **Level 1**: round numbers up to the hundreds
- **Level 2**: round numbers up to the thousands
- **Level 3**: round numbers up to the millions

START
Our Free Materials
Over 40 Tutorials

1. Getting started tutorials
   – 1-4 hours

2. Tutorials on CS topics
   – Methods, conditionals, lists, etc
   – Variables (timers/scores).

3. Animation tutorials
   – Lights, camera, scene change, billboards, invisible objects,
Web site

• Adventures in Alice Programming
  www.cs.duke.edu/csed/alice/aliceInSchools
Conclusions

• We have shown several ways to visualize CS concepts and make them interactive
  – Software: JAWAA, JFLAP, Alice
  – Props: Food

• Questions?