Teaching Computer Science with Interaction and Visualization

Susan H. Rodger
Computer Science Dept.
Duke University

December 4, 2006
Outline

• **Who am I?**

• **Teaching Strategies with Interaction**

• **Interaction and Visualization in First Year CS**
  – Project: Alice Programming Language
  – Project: Peer Led Team Learning

• **Interactive/Visual Software for Teaching**
  – Project: JFLAP
Who am I? Education

PhD, 1989
Computer Science

Assistant Prof.
1989-1994

Assistant Professor of the Practice
1994-1997
Associate Professor of the Practice
1997-present
My Research Interests

• Computer Science Education
• Visualization and Interaction
  – Instructional Tools for Theoretical concepts
    • Automata theory and formal languages
  – Teaching Introductory Computer Science
• Algorithm Animation
Who Am I? - Family
Who am I? Hobby – Shape cakes
How do you make those cakes?
What is Professor “of the Practice”?  

• Duke only – Special Position  
• PhD preferred, or appropriate professional experience  
• Focus on “education in the discipline”  
• Main tasks  
  – Teaching  
  – Research (related to education)  
  – Service  
• 20% of Faculty in Arts and Sciences Departments at Duke are “Professors of the Practice”
How does Professor of Practice differ from regular rank faculty in CS?

- Teach 2 courses/semester vs 1 course/semester
- Focus on undergrad curriculum, first two years
- Teach intro courses
  - Other grad and undergrad courses too
- Supervise undergraduates more than grad. studs.
- Publish Papers – in CS Education
- Write grants – CS education or education part of research grant
Outline

• Who am I?
• Teaching Strategies with Interaction
• Interaction and Visualization in First Year CS
  – Project: Alice Programming Language
  – Project: Peer Led Team Learning
• Interactive/Visual Software for Teaching
  – Project: JFLAP
Teaching Strategies with Interaction and Visualization – Why?

• 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
Teaching Strategies
Interaction with Your Class
First, Get to know your students!

- Get their picture
  - Pass around a camera the first day
  - Registrar photo lists
- Assigned Seating
- Calling on students
  - Pick-a-student system (rotate thru their pictures)
Interactive Lecture

- Lecture for 10-20 minutes
- “Time to Think” - Students solve a problem
  - Solve problem from scratch (longer)
  - Find what is wrong with a “solution” (shorter)
- Discuss solution
  - Ask how many did X? (gets students involved)
  - Give a possible solution (shorter)
  - Student present solution (longer)
- REPEAT
Interactive Lecture Notes and Handouts

• Create 3 versions of my lecture
  – Slides - with blanks to fill in
  – Handouts with blanks to fill in
  – My notes – blanks filled in
Interactive Lecture with Computers OR Interactive Lab

- Lecture for 10-20 minutes
- Students work on problem with computers
- Bring students back together
Ideal Room Layout with Computers

- 20 computers, 40 students
- Extra desks for group work
- Advantage: see what students are doing
Teaching Strategies
Group Dynamics

• Work with large or small classes
Divide Students into Groups

• Random assignment
  – Count off and assign groups on the spot
  – Assign in advance, bring in seating chart
  – Change groups every 2-3 weeks

• Students work on problems during class in groups
  – Short (2 min) or long problems (20 min)
Advantages to Random Groups
Large or Small classes

- Students help each other
- Students are more confident to answer questions – not feeling alone
- Students present different solutions
- Students meet other students
- Less work to grade for you
- Can pass graded work back quickly
  - Sort it by groups first
Groups in Lab - Pair Programming

• Work in pairs

• Responsibilities
  – One person is driver
  – One person is navigator

• “Pair Programming Illuminated” by Williams and Kessler, 2003
Teaching Strategies
Activities Without a Computer

• Get creative in bringing hands-on activities into the classroom
Interaction in Class – Props

Passing “Parameters” in Class

• Pass by reference – throw frisbee

• Pass by value – throw copy of frisbee

• Pass by const reference – throw “protected” frisbee
Interaction in Class – Props
Linked List and Memory Heaps

ITiCSE 98 – Astrachan – “Concrete Teaching: Hooks and Props as Instructional Technology
Interaction in Class – Props
Memory Heap
Interaction with Class
Binary Tree and Recursion

• Build a binary tree
  – Pick a root
  – Root picks two children – point at them
  – Repeat until everyone is part of the tree

• Recursively calculate height of tree
  – Start at root
  – Ask children their height
  – Leaf notes know their height is 0

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
Outline

• Who am I?
• Teaching Strategies with Interaction
• Interaction and Visualization in First Year CS
  – Project: Alice Programming Language
  – Project: Peer Led Team Learning
• Interactive/Visual Software for Teaching
  – Project: JFLAP
Motivation – Declining Enrollments

Figure 1. Computer Science Listed as Probable Major Among Incoming Freshmen
Source: HERI at UCLA
How do we Teach Science?

• Physics – experiments

• Chemistry - experiments

• Biology - experiments
How do we introduce Computer Science?

• Write a calculator
• Write a banking program
• Etc…

```java
public class Simple {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```
Why Can’t the introduction of Computer Science be exciting?

• Add Visualization and Interaction!
What Is Alice?

• A modern programming tool
  – 3-D graphics
  – 3-D models of objects

• Animation
  – Objects can be made to move around virtual
    world (a simulation or video game)

• Developed at Carnegie Mellon University

• At Duke – Use Alice in CompSci 4
The Power of Alice

• Automatically keeps track of 3-D objects
  – What objects are in the virtual world
  – Types of objects
  – Positions of objects in the world
Classes and Objects

• Classes
  – In Alice, classes are predefined as 3D models

• Objects
  – An object is an instance of a class
    • Class: Chicken
    • Objects: Chicken, Chicken2, Chicken3
Objects in Alice

- Objects already exist
- Objects have parts
Galleries of 3D Objects

• Sources of 3D objects
  – Local gallery – comes with Alice
  – Alice web gallery
Object Position

• Objects
  – Are positioned in 3D space
  – Have six degrees of freedom
Program an Object - Demo
Can Teach Computer Science Concepts with Alice

- Conditional and looping structures
- Methods, functions
- Events
- Inheritance
- Recursion
- Lists, Arrays
Methods

- Built-in methods
- Write class methods
- Write world methods
Inheritance

- Dragon

- FlyingDragon
Example – parameters and events

- People are trapped in a burning building
- Select which person will be rescued
Parameters

- Types and number of parameters must match with arguments

Call

```plaintext
firetruck.savePerson whichFloor = burningBuilding.thirdFloor
whichPerson = randomGirl3
howFar = 3
```
Events

• Default event

• Other events
Three Events

• The argument sent to parameters depends on which person is mouse clicked

• Note - we positioned fire truck so distance from floor X is X meters (to floor 3 is 3 meters)
Example – Lists - WacAMole

- List of Moles
- Randomly moves one of them up and down
- Counter to keep track of score
- Event: when click on object, search through list to see if object is a mole
Example – Binary guessing Game

- Three switches represent 3 digit binary number
- Random number generated to guess
- Click on switch to move it (up = 1, down = 0)
Example - Arrays

- Shuffle, then sort by height
Function to return object with the tallest height from an Array

```plaintext
world.objectWithTallestHeight

world.objectWithTallestHeight No parameters

Obj maxHeightSoFar = ArrayVisualization

maxHeightSoFar set value to the value at ArrayVisualization[0]

Loop 123 index from 1 up to (but not including) ArrayVisualization's size incrementing by 1

If the value at ArrayVisualization[index] is taller than maxHeightSoFar

maxHeightSoFar set value to the value at ArrayVisualization[index]

Else Do Nothing

Return maxHeightSoFar
```
Why use Alice?

• There are very few women in computer science
  – Not uncommon to have 20% or fewer women in a course
  – Nationwide CompSci enrollments are down
Does Alice attract females?

• Build stories and interactive games
• Current two year study – several universities
• At Duke
  – CompSci 4 Spring 2005
    • 22 preregister, 30 enroll (12 female + 3 African Amer.)
  – CompSci 4 Fall 2005
    • 20 preregister, 31 enroll (17 female – 1 African Amer.)
  – CompSci 4 Fall 2006 – 2 sections
    • 64 students, 33 female, 7 African Amer.
  – Advertised in school paper
    • picture of ice skater
    • Web site of animations
Alice into High School and Middle Schools

• Pre-CS 1 course
• Middle Schools
  – Story-Telling Alice – not yet released
• Training of teachers
• Alice Symposium at Duke in June 2006
  – Over 100 High School, College Faculty
  – http://www.cs.duke.edu/csed/aliceworkshop
Alice Software – is free!

• Runs on Mac and PC
• CompSci 4 web site
  www.cs.duke.edu/courses/fall06(cps004/rodger/
• Textbook available
  – Learning to Program with Alice by Dann, Cooper, and Pausch

• Download from web
  www.alice.org
Outline

• Who am I?
• Teaching Strategies with Interaction
• Interaction and Visualization in First Year CS
  – Project: Alice Programming Language
  – Project: Peer Led Team Learning
• Interactive/Visual Software for Teaching
  – Project: JFLAP
Peer Led Team Learning in CS

• New program to try to attract and retain women in CS
• Collaborative effort – 8 universities
  – UW Madison, Purdue, Georgia Tech, Rutgers, Beloit, Duke, Loyola, UW Milwaukee
• At Duke
  – One-year program – Duke Emerging Scholars
  – Goal: 50% female – Reality: 30-50% female
  – Take CompSci 4 (Alice) followed by CompSci 6 (CS 1),
  – Take CompSci 18S (Problem Solving Seminar) whole year
• Sponsored by National Science Foundation
CompSci 18S Problem Solving

• Sessions led by Undergraduate Peer Leaders and/or faculty
• Students work in groups of 3 or 4 to solve problems – on paper or on computer
• Problems are on different areas of CS OR sometimes related to their other Computer Science class

• IDEA: Female students will make friends (support group) and not feel isolated in a male dominated field.
Example Problem Solving Topics

- Algorithms, Sorting, Searching
- Social Networking
- Parallelism
- Randomness (with Alice)
- Genomics
- Robotics
- Puzzles – Sudoku
- Finite Automata, Turing Machines, L-Systems
- Compression
Example: Be a Robot

- 4 People
  - Controller (head)
  - Sensors (eyes)
  - Manipulators (2 hands)

- Blindfolded except eyes
- Controller knows what to build
- Limited communication

SIGCSE 96, Rodger, Walker
Example: Sorting Over 100 Words

- An envelope with over 100 words, each word on one slip of paper
- Sort the words
- Write down the algorithm
- Early assignment, before sorting is covered

anchor
physiotherapist
pathetic
bootstrapped
acrimonious
polarization
firecracker
palindrome
observatory
controversial
orchestrate
statistician
confrontation
scrumptious
revolutionary
...
Outline

• Who am I?
• Teaching Strategies with Interaction
• Interaction and Visualization in First Year CS
  – Project: Alice Programming Language
  – Project: Peer Led Team Learning
• Interactive/Visual Software for Teaching
  – Project: JFLAP
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  | \[\begin{array}{c|cc}
   a & b \\
   \hline
   q_0 & q_1 & q_0 \\
   q_1 & q_0 & q_2 \\
   q_2 &   &   
\end{array}\] |

| Visual   | ![Visual Diagram](image) |

| Interactive | ![Interactive Diagram](image) |
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
JFLAP - Lsystems

- Create an L-system

- Render the L-system
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
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!
• 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.
FA Edit & Simulation
After One Step

• 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.
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.

\[(a+ab)(b+ab)^*a\]
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 $aabbcc$ 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 $T \to Tg$ rule adds $g$ to the derivation, which draws a line segment.
- We add another rewriting rule for $T$, $T \to T$.
- 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
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’s Use Around the World

• JFLAP web page has over 70,000 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
JFLAP in German

Um das JFLAP-Applet zu starten, [hier](http://www.mathematik.uni-kiel.de/flap/)
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:

<table>
<thead>
<tr>
<th>Theorie</th>
<th>Hilfe</th>
<th>Übungen</th>
<th>JFLAP-Applet</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>eine Einführung in die Automatentheorie mit Themen (Schritt-für-Schritt) und Index (Stichwortverzeichnis)</td>
<td>eine Hilfedokumentation und Einführung in das Programm JFLAP mit Themen (Schritt-für-Schritt) und Index (Stichwortverzeichnis)</td>
<td>Übungen zu den einzelnen Automatentypen und zu den JFLAP-Funktionen (mit Lösungen)</td>
<td>das eigene Programm JFLAP zum graphischen Konstruieren und Testen von Automaten</td>
<td>(diese) Informationen für den Benutzer</td>
</tr>
</tbody>
</table>

Einstiegern wird dringend empfohlen, zuerst die Help-Themen anzuschauen, bevor erste Schritte mit JFLAP unternommen werden!
JFLAP in Spanish

**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 automatas. Mediante el uso de esta herramienta se practicarán operaciones relacionadas con gramáticas regulares, automatas finitos y obtención del árbol de derivación en gramáticas independientes del contexto.

**Obligatoriedad**
La práctica no es obligatoria.

**Prerrequisitos**

El alumno debe conocer los elementos relacionados con los niveles 2 y 3 de la jerarquía de Chomsky (lenguajes regulares, expresiones regulares, gramáticas regulares, automatas 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 in Chinese

JFLAP Demo Applet

斐森恆鼎椁模إصابة逢短暂兼雄假日JAVA落驹謹尙駄撲撫卽築毎是歴發越雄僞 stakeholderships observes 数字化使察情越雄僞甜口凹越雄僞脭砸孜 stopwatch. * 柳寞桶湛宅 § 枥等蔚堆放婉励菌液柳寞桶湛宅蹦祥 3 隅 菌鸞发越越雄僞脭醛遥} 跳 fetch 菌鸞发越越雄僞脭醛遥 舐摀誇烂柳寞桶湛宅蹦祥 菌鸞发越越雄僞脭醛遥 跳 fetch 菌鸞发越越雄僞脭醛遥

JFLAP 堆驹

掛JAVA Applets剖猋IE4.0 麼 Netscape 4.5 睜欸哖掛腔銅擬腔猛苑卽樓<br>得住Applets剖猋瑁隅腔矣潔卽 驚陽脤渃卽
Future Work - Evaluation

- JFLAP 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
More on JFLAP

• JFLAP is free!

• www.jflap.org

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

• Interaction and Visualization
  – Make classes interactive
  – Use interactive and visual software to experiment with concepts
• Use engaging tools: Alice, JFLAP
• My web page:
  http://www.cs.duke.edu/~rodger/

• Questions?