Through Visualization and Interaction, Computer Science Concepts Come Alive

Susan H. Rodger
Duke University
May 11, 2016
Evergreen State College

Supported by NSF Grants DRL-1031351, DUE 1044191, CRA-W, and IBM Faculty Awards
About Me

• Professor of the Practice of Computer Science

• Area: Visualization and Animation, Computer Science Education

• Passionate about education/diversity
  • SIGCSE Chair
  • ACM Education Policy Committee
  • CRA-W Board Member
About Me - Hobby – Baking Shape cakes

The Wiggles magazine Issue No. 42
How do you make those cakes?
CS 1
Sorting
Cookies
Graduate School

• PhD Purdue University 1989
  • Computational Geometry
  • Parallel Scheduling Algorithms
Outline

• Introduction

• CS Concepts Come Alive
  • Alice Programming Language
  • Algorithm Visualization
  • Automata Theory with JFLAP
  • Additional Ways to Engage with CS

• Diversity Efforts
CS Concepts Coming Alive

• What data structure is this?
YARN, in the shape of a binary tree. Subtrees made with molecule kit. What is it?
2D-range tree

- Search in x-y plane
- Main tree organized by x-values
- Subtree organized by y values
Binary Search tree of points in the plane – sorted by X-value

In the x-range

Each subtree organized by y-value

Search each subtree by y-value
Different Types of Learners

• 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
Learner Engagement Taxonomy with visualization software

• Different forms of Learner engagement
  – No Viewing
  – Viewing
  – Responding
  – Changing
  – Constructing
  – Presenting

• ITiCSE Working Group Report 2002 (Naps et al.)
Outline

• Introduction
• CS Concepts Come Alive
  – Alice Programming Language
  – Algorithm Visualization
  – Automata Theory with JFLAP
  – Additional Ways to Engage with CS
• Diversity Efforts
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
More on ... 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
Play Alice Animation

• Chicken rises, cow turns head and talks

Moo Moo Moo
Why Alice?

• Lots of other great tools for teaching programming
  
  Greenfoot

• Alice is easy to use, drag-and-drop, objects already exist

• Storytelling - Attractive to both girls and boys
Success - Alice attracts diverse group

• 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.
  • CompSci 4 Fall 2007 – 2 sections
    • 84 students - > 50% female
  • CompSci 4 Fall 2008 – 2 sections
    • 100 students - > 50% female
  • Same for Spring 2009, Fall 2009...
  • Advertised in school paper
    • picture of ice skater
    • Web site of animations
  • This course is now CompSci 94
Computer Science Concepts come alive with Alice - Examples

• Objects are visible
• Variables
• Inheritance
• Lists
• Array
Example: Objects are visible

Getting Started Tutorial teaches
• Placing objects
• Moving objects
• Setting up Camera tripods and moving between views
• Using built in methods and writing your own
  • Dragon flapWings
• Gluing objects together
• Adding sound, 2D pictures to enhance world
Getting Started Tutorial – 3 part
Variables – Timer and Score

To win this game, you must steer the boat through each ring and beat the clock. You receive one point for each ring, and there are 10 rings, so if your score is less than 10 at the end, you lose!
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**
  • Alice Programming Language
  • Algorithm Visualization
  • Automata Theory with JFLAP
  • Additional Ways to Engage with CS

• Diversity Efforts
Algorithm Visualization/Animation
Software/Aps/Videos

• Tango, Xtango, 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
• Lots of animations and systems on the web!
• Lots of videos of algorithm animations on the web!
Use of Algorithm Animation in CS 1/2

• Instructor
  • Make/Use animations for lecture
  • Stop/Pause – ask what will happen next
  • must be interactive

• Student
  • Create animations
  • Replay animations from lecture with same or new inputs
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

Student must have graduated. Link no longer works!
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.

![Diagram of Red Black Tree with Stream of Keys](image-url)
Python Tutor
Compute reverse of a list

```python
def reverse(numbers):
    answer = []
    for num in numbers:
        answer.insert(0, num)
    return answer

myList = [4, 7, 8, 3]
reversed = reverse(myList)
```

Edit code
Python Tutor
Compute reverse of a list

```python
def reverse(numbers):
    answer = []
    for num in numbers:
        answer.insert(0, num)
    return answer

myList = [4, 7, 8, 3]
reversed = reverse(myList)
```
Python Tutor
Compute reverse of a list

```python
def reverse(numbers):
    answer = []
    for num in numbers:
        answer.insert(0, num)
    return answer

myList = [4, 7, 8, 3]
reversed = reverse(myList)
```
Python Tutor
Compute reverse of a list

```python
def reverse(numbers):
    answer = []
    for num in numbers:
        answer.insert(0, num)
    return answer

myList = [4, 7, 8, 3]
reversed = reverse(myList)
```
Python Tutor
Compute reverse of a list

Python 2.7
1  def reverse(numbers):
2      answer = []
3      for num in numbers:
4          answer.insert(0, num)
5      return answer
6
7  myList = [4, 7, 8, 3]
8  reversed = reverse(myList)
Python Tutor
Compute reverse of a list

```python
def reverse(numbers):
    answer = []
    for num in numbers:
        answer.insert(0, num)
    return answer

myList = [4, 7, 8, 3]
reversed = reverse(myList)
```

Frames
- Global frame
  - reverse
  - myList

Objects
- list
  - 4 7 8 3
- reverse
  - numbers
  - answer
  - num 7
- list
  - 0 7 4
Python Tutor

Compute reverse of a list

```python
def reverse(numbers):
    answer = []
    for num in numbers:
        answer.insert(0, num)
    return answer

myList = [4, 7, 8, 3]
reversed = reverse(myList)
```

Edit code
Python Tutor
Compute reverse of a list

```python
def reverse(numbers):
    answer = []
    for num in numbers:
        answer.insert(0, num)
    return answer

myList = [4, 7, 8, 3]
reversed = reverse(myList)
```
AlgoViz – Repository of Algorithm Visualizations
Electronic Textbooks (ebooks) engage students

• OpenDSA (Shaffer, Virgina Tech)
  • Algorithm animations built in
• runestoneinteractive.org (Brad Miller)
  • Several books (Python)
    • Python - try and run code built in
    • Quizzes
• Zyante.com – interactive textbooks
• Track student progress
• Requirements and design strategies for open source interactive computer science eBooks
  • ITiCSE 2013 Working Group (Korhonen, Naps, et al)
Index Operator: Working with the Characters of a String

The indexing operator (Python uses square brackets to enclose the index) selects a single character from a string. The characters are accessed by their position or index value. For example, in the string shown below, the 14 characters are indexed left to right from position 0 to position 13.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13
```

```
L u t h e r  C o l l e g e
```

```
-14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
```

It is also the case that the positions are named from right to left using negative numbers where -1 is the rightmost index and so on. Note that the character at index 6 (or -8) is the blank character.

```python
school = "Luther College"
m = school[2]
print(m)

lastchar = school[-1]
print(lastchar)
```
Run and edit code in the book

```python
school = "Luther College"
m = school[2]
print(m)
lastchar = school[-1]
print(lastchar)
```
Integrates in Python Tutor
Questions for feedback

Check your understanding

strings-4-1: What is printed by the following statements?

```
s = "python rocks"
print(s[3])
```

- t
- h
- c
- Error, you cannot use the [ ] operator with a string.

Incorrect. Index locations do not start with 1, they start with 0.

Check Me  Compare me

strings-4-2: What is printed by the following statements?

```
s = "python rocks"
print(s[2] + s[-5])
```

- tr
- ps
- nn
- Error, you cannot use the [ ] operator with the + operator.

Correct! Yes, indexing operator has precedence over concatenation.

Check Me  Compare me
Outline

• Introduction

• CS Concepts Come Alive
  • Alice Programming Language
  • Algorithm Visualization
  • Automata Theory with JFLAP
  • Additional Ways to Engage with CS

• Diversity Efforts
Java Program to count number of words in a phrase

```java
public static void main() {
    String phrase = "baby fish should be fed three times a day";
    int numWords = 0;
    int pos = phrase.indexOf(" ");
    while (pos > 0) {
        numWords += 1;
        phrase = phrase.substring(pos + 1);
        pos = phrase.indexOf(" ");
    }
    numWords += 1;
    System.out.println("Number of words in phrase is " + numWords);
}
```

- Is this program syntactically correct?
First identify each word/token

```
public static void main() {
```

<table>
<thead>
<tr>
<th>TOKEN</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>keyword</td>
</tr>
<tr>
<td>static</td>
<td>keyword</td>
</tr>
<tr>
<td>void</td>
<td>keyword</td>
</tr>
<tr>
<td>main</td>
<td>variable</td>
</tr>
<tr>
<td>(</td>
<td>leftparen</td>
</tr>
<tr>
<td>)</td>
<td>rightparen</td>
</tr>
<tr>
<td>{</td>
<td>leftbrace</td>
</tr>
</tbody>
</table>
First identify each word/token (cont)

String phrase = "baby fish should be fed three times a day";

TOKEN
String
phrase
=
"

baby fish ... day
"
;

TYPE
keyword
variable
equals
doublequote
string
doublequote
semicolon
First identify each word/token (cont)

```c
int numWords = 0;
```

<table>
<thead>
<tr>
<th>TOKEN</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>keyword</td>
</tr>
<tr>
<td>numWords</td>
<td>variable</td>
</tr>
<tr>
<td>=</td>
<td>equals</td>
</tr>
<tr>
<td>0</td>
<td>int</td>
</tr>
<tr>
<td>;</td>
<td>semicolon</td>
</tr>
</tbody>
</table>
Define Rules for a valid program (Grammar)

\[\text{<program>} := \text{<procDef>} \ ( \ ) \ \{ \ \text{<body>} \ \} \]
\[\text{<program>} := \text{<procDef>} \ (\text{<arglist}> \ ) \ \{ \ \text{<body>} \ \} \]
\[\text{<procDef>} := \text{public static void main} \]
\[\text{<procDef>} := \text{<pubtype>} \ \text{<returntype>} \ \text{<variable>} \]
\[\text{<body>} := \text{<decllist>} \ \text{<stmtlist>} \]
\[\text{<declist>} := \text{<decl>} ; \ \text{<decllist>} \]
\[\text{<declist>} := \text{<decl>} ; \]
\[\text{<stmtlist>} := \text{<stmt>} ; \ \text{<stmtlist>} \]
\[\text{<stmtlist>} := \text{<stmt>} ; \]
Define Rules for a valid program (Grammar)

\[
\begin{align*}
<\text{decl}> & \quad := \text{int }<\text{variable}> = <\text{integer}> \ ; \\
<\text{decl}> & \quad := \text{String }<\text{variable}> = " <\text{string}> " \ ; \\
<\text{stmt}> & \quad := \text{while ( }<\text{cond}> \text{ ) } \{ <\text{stmtlist}> \} \\
<\text{stmt}> & \quad := <\text{variable}> += <\text{integer}> \ ; \\
<\text{stmt}> & \quad := <\text{variable}> = <\text{integer}> \ ; \\
\text{etc}
\end{align*}
\]
Now derive the program using the rules (grammar)

\[
\text{<program> := <procDef> ( ) { <body> }}
\]

\[
:= \text{public static void main () { <body>}}
\]

\[
:= \text{public static void main () { <declist> <stmtlist> }}
\]

\[
:= \text{public static void main() { <decl> ; <declist> <stmtlist> }}
\]
And so on until you derive the program...

```java
public static void main() { String phrase = "baby fish should be fed ...");
int numWords = 0; int pos = phrase.indexOf(" ");
while (pos > 0) {
    numWords += 1;
    ...
}
}
Determining if a Java program is syntactically correct

• Finite state machine (or deterministic finite automaton - DFA) – to identify the words or tokens of the program
• Context-free grammar – to write the rules of the programming language
• LR Parsing determining if the program fits the rules – trying to derive the program.

• This area is known as Formal languages and Automata theory
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 8.0Beta – 2011-14 Julian Genkins, Ian McMahon, Peggy Li, Lawrence Lin, John Godbey
- JFLAP in OpenDSA – 2015 Sung-Hoon Kim and Martin Tamayo

Over 25 years!
Why Develop Tools for Automata?

Textual

\[
\begin{align*}
\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\} \\
\delta = \{(q_0, b, q_0), (q_0, a, q_1), (q_1, a, q_0), (q_1, b, q_2), (q_2, a, q_1)\}
\end{align*}
\]

Tabular

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q_0</td>
<td>q_1</td>
<td>q_0</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 Diagram](image1)

Interactive

![Interactive Diagram](image2)
DFA Example

• Build a deterministic finite automaton (DFA) to recognize even binary numbers with an even number of 1s.

• 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, odd    Yes    No, ends    Yes
  no. of 1’s    In 1
Build with JFLAP

START

q0

1

q1

0

FINISH

q2

0

1

q3

FINISH

q4

0
Simulation on 1101010
Simulation on 1101010
Simulation on 1101010
Simulation on 1101010
Simulation on 1101010
Simulation on
1101010
Simulation on 1101010
Accepts Input!

1101010
Add meaning to states!
A deterministic finite automaton (DFA) for recognizing binary numbers with an odd number of 1's.

- **States:**
  - q0: Start state, no 0s.
  - q1: Only one 0.
  - q2: Odd number of 1's.
  - q3: No 1's.
  - q4: Accept state, only 0s.

- **Transitions:**
  - From q0:
    - On 1: Go to q2.
    - On 0: Go to q1.
  - From q1:
    - On 0: Go to q0.
    - On 1: Go to q2.
  - From q2:
    - On 0: Go to q3.
    - On 1: Go to q4.
  - From q3:
    - On 0: Go to q2.
    - On 1: Go to q4.
  - From q4:
    - On 0: Go to q4.
    - On 1: Go to q4.
The diagram is a state machine with the following states:

- **q0**: Start state, transitions to q1 on 0, to q2 on 1.
- **q1**: Only one 0 state, transitions to q4 on 0.
- **q2**: Odd number of 1's state, transitions to q3 on 1, to q4 on 0.
- **q3**: Even number of 1's, ends in 1 state, transitions to q2 on 0.
- **q4**: Even number of 1's, ends in 0 state, transitions to q2 on 0, to q3 on 1.
Test Multiple Inputs

A state diagram and a table showing the results of testing multiple inputs.
Example: Build an NFA for valid integers

• Example:
  • Valid integers {-3, 8, 0, 456, 13, 500, ...}
  • Not valid: {006, 3-6, 4.5, ...}
Example: NFA for all valid integers
NFA annotated and shortcut

• Shortcut: [1-9] on labels
Back to Recognizing whether a Java Program is syntactically correct or not...

• You would need a DFA to recognize all valid words in a program
  • An integer
  • A variable name
  • All keywords
  • All special symbols ; + - ( ) { }
  • etc
Another Example: Grammar

• Grammar – set of replacement rules to define a language
• Previously looked at grammar for Java (very small part of it!)
• Grammar for a formal language (simpler)
• Consider representing underlined words in a text file (to be interpreted later):
  • `cookie&&&&&&______                   cookie`
  & = go back one
Grammar for $a^n b^n c^n$

- Unrestricted grammar
- Generates strings with an equal number of a’s, b’s, c’s
- a’s first, then b’s, then c’s
- Example strings can derive:
  
  abc
  aabbcc
  aaabbbcccc
  aaaabbbbbbcccc
  aaaaabbbbbbbcccc
  ...

<table>
<thead>
<tr>
<th>S</th>
<th>→ AX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>→ a Abc</td>
</tr>
<tr>
<td>A</td>
<td>→ ab Bc</td>
</tr>
<tr>
<td>BX</td>
<td>→  λ</td>
</tr>
<tr>
<td>Bb</td>
<td>→ b B</td>
</tr>
<tr>
<td>Bc</td>
<td>→ D</td>
</tr>
<tr>
<td>DX</td>
<td>→ EXc</td>
</tr>
<tr>
<td>Db</td>
<td>→ b D</td>
</tr>
<tr>
<td>Dc</td>
<td>→ c D</td>
</tr>
<tr>
<td>aE</td>
<td>→ a B</td>
</tr>
<tr>
<td>bE</td>
<td>→ Eb</td>
</tr>
<tr>
<td>cE</td>
<td>→ Ec</td>
</tr>
</tbody>
</table>
Example Derivation for aabbcc

S → AX

rule:  S -> AX
Example Derivation for aabbcc

S → AX
   → aAbcX

rule: S -> AX
      rule: A -> aAbc
Example Derivation for aabbcc

S → AX          rule: S -> AX
→ aAbcX        rule: A -> aAAbc
→ aaBbcbcX      rule: A -> aBbc

NOTE: We have generated the correct symbols, aabcbc, but they are in the wrong order!
Example Derivation for aabbcc

\[ \begin{align*}
S & \rightarrow AX & \text{rule: } S \rightarrow AX \\
\rightarrow aAbcX & \quad & \text{rule: } A \rightarrow aAbc \\
\rightarrow aaBbcX & \quad & \text{rule: } A \rightarrow aBbc \\
\rightarrow aaBBcX & \quad & \text{rule: } Bb \rightarrow bB \\
\rightarrow aabbBcX & \\
\end{align*} \]
Example Derivation for aabbcc

S → AX
    → aAbcX
    → aaBbcX
    → aabBcbcX
    → aabDbcX

rule: S -> AX
rule: A -> aAbc
rule: A -> aBbc
rule: Bb -> bB
rule: Bc -> D

Note: the D absorbed the c!
Example Derivation for aabbcc

S \rightarrow AX \quad \text{rule: } S \rightarrow AX
\rightarrow aAbcX \quad \text{rule: } A \rightarrow aAbc
\rightarrow aaBbcbcX \quad \text{rule: } A \rightarrow aBbc
\rightarrow aabBbcbcX \quad \text{rule: } Bb \rightarrow bB
\rightarrow aabDbcX \quad \text{rule: } Bc \rightarrow D
\rightarrow aabDbcX \quad \text{rule: } Db \rightarrow bD
Example Derivation for aabbcc

\[
\begin{align*}
S & \rightarrow AX & \text{rule: } S \rightarrow AX \\
& \rightarrow aAbcX & \text{rule: } A \rightarrow aAbc \\
& \rightarrow aaBbcX & \text{rule: } A \rightarrow aBbc \\
& \rightarrow aabbbbcX & \text{rule: } Bb \rightarrow bB \\
& \rightarrow aabDbcX & \text{rule: } Bc \rightarrow D \\
& \rightarrow aabbDcX & \text{rule: } Db \rightarrow bD \\
& \rightarrow aabbcDX & \text{rule: } Dc \rightarrow cD
\end{align*}
\]
Example Derivation for aabbcc

S → AX
   → aAbcX
   → aaBbcbX
   → aabBcX
   → aabD
   → aabbD
   → aabbcc

Eventually ... → aabbcc

Note the c spit out on right end!
We could have done this derivation of `aabbbc` with JFLAP.

Now let’s see how JFLAP visualizes this derivation with a “parse tree” [Parse DAG].
String accepted! 51 nodes generated.

<table>
<thead>
<tr>
<th>LHS</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>AX</td>
</tr>
<tr>
<td>A</td>
<td>aAbc</td>
</tr>
<tr>
<td>A</td>
<td>aBbc</td>
</tr>
<tr>
<td>Bb</td>
<td>bB</td>
</tr>
<tr>
<td>Bc</td>
<td>D</td>
</tr>
<tr>
<td>Dc</td>
<td>cD</td>
</tr>
<tr>
<td>Db</td>
<td>bD</td>
</tr>
<tr>
<td>DX</td>
<td>EXc</td>
</tr>
<tr>
<td>BX</td>
<td>λ</td>
</tr>
<tr>
<td>cE</td>
<td>Ec</td>
</tr>
<tr>
<td>bE</td>
<td>Eb</td>
</tr>
<tr>
<td>aE</td>
<td>aB</td>
</tr>
</tbody>
</table>
Input: aabbcc
String accepted! 51 nodes generated.

Derived AX from S.
Input: aabbcc
String accepted! 51 nodes generated.

<table>
<thead>
<tr>
<th>LHS</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>AX</td>
</tr>
<tr>
<td>A</td>
<td>aAbc</td>
</tr>
<tr>
<td>A</td>
<td>aBbc</td>
</tr>
<tr>
<td>Bb</td>
<td>bB</td>
</tr>
<tr>
<td>Bc</td>
<td>D</td>
</tr>
<tr>
<td>Dc</td>
<td>cD</td>
</tr>
<tr>
<td>Db</td>
<td>bD</td>
</tr>
<tr>
<td>DX</td>
<td>EXc</td>
</tr>
<tr>
<td>BX</td>
<td>λ</td>
</tr>
<tr>
<td>cE</td>
<td>Ec</td>
</tr>
<tr>
<td>bE</td>
<td>Eb</td>
</tr>
<tr>
<td>aE</td>
<td>aB</td>
</tr>
</tbody>
</table>

Derived aAbc from A.
Note all letters there, but wrong order: aabcbc
Input: aabbcc
String accepted! 51 nodes generated.

LHS               RHS
S → AX
A → aAbc
A → aBbc
Bb → bB
Bc → D
Dc → cD
Db → bD
DX → EXc
BX → λ
cE → Ec
bE → Eb
aE → aB

Derived bB from Bb.
Absorb the “c”
Input: aabbcc
String accepted! 51 nodes generated.

LHS | RHS
---|---
S  | AX
A  | aA Bc
A  | aB Bc
Bb | bB
Bc | D
Cc | cD
Db | bD
DX | EXc
BX | λ
cE | Ec
bE | Eb
aE | aB

Derived bD from Db.
Input: aabbcc
String accepted! 51 nodes generated.

Derived cD from Dc.
Spit out the “c” at the right end.
String accepted! 51 nodes generated.

Derived Ec from cE.
Input: `aabbcc`
String accepted! 51 nodes generated.

Derived `Eb` from `bE`. 

<table>
<thead>
<tr>
<th>LHS</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>AX</td>
</tr>
<tr>
<td>A</td>
<td>aA</td>
</tr>
<tr>
<td>A</td>
<td>aB</td>
</tr>
<tr>
<td>Bb</td>
<td>bC</td>
</tr>
<tr>
<td>Bc</td>
<td>D</td>
</tr>
<tr>
<td>Dc</td>
<td>cD</td>
</tr>
<tr>
<td>Db</td>
<td>bD</td>
</tr>
<tr>
<td>DX</td>
<td>EXc</td>
</tr>
<tr>
<td>BX</td>
<td>λ</td>
</tr>
<tr>
<td>cE</td>
<td>Ec</td>
</tr>
<tr>
<td>bE</td>
<td>Eb</td>
</tr>
<tr>
<td>aE</td>
<td>aB</td>
</tr>
</tbody>
</table>

**LHS** stands for the left-hand side of the production rules, and **RHS** stands for the right-hand side.
Input: aabbcc
String accepted! 51 nodes generated.

Derived Eb from bE.
String accepted! 51 nodes generated.

Derived aB from aE.
Input: aabbcc
String accepted! 51 nodes generated.

Derived bB from Bb.
Input: aabbcc
String accepted! 51 nodes generated.

Derived BB from Bb.
Absorb second “c”
Spit the “c” out at right end
Input: aabbcc
String accepted! 51 nodes generated.

<table>
<thead>
<tr>
<th>LHS</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>AX</td>
</tr>
<tr>
<td>A</td>
<td>aAabc</td>
</tr>
<tr>
<td>A</td>
<td>aABbc</td>
</tr>
<tr>
<td>Bb</td>
<td>bB</td>
</tr>
<tr>
<td>Bc</td>
<td>D</td>
</tr>
<tr>
<td>Dc</td>
<td>cD</td>
</tr>
<tr>
<td>Db</td>
<td>bD</td>
</tr>
<tr>
<td>DX</td>
<td>EXc</td>
</tr>
<tr>
<td>BX</td>
<td>λ</td>
</tr>
<tr>
<td>cE</td>
<td>Ec</td>
</tr>
<tr>
<td>bE</td>
<td>Eb</td>
</tr>
<tr>
<td>aE</td>
<td>aB</td>
</tr>
</tbody>
</table>

Derived Eb from bE.
String accepted! 51 nodes generated.

LHS | RHS
---|---
S   | AX
A   | aA
A   | aB
Bb  | bB
Bc  | D
Dc  | cD
Db  | bD
DX  | EXc
BX  | λ
cE  | Ec
bE  | Eb
aE  | aB

Derived Eb from bE.
Input: aabbcc
String accepted! 51 nodes generated.

LHS | RHS
---|---
S  | AX
A  | aAbc
A  | aBbc
Bb | bB
Bc | D
Dc | cD
Db | bD
DX | EXc
BX | λ
cE | Ec
bE | Eb
aE | aB

Derived aB from aE.
Input: aabbc
String accepted! 51 nodes generated.

Derived bB from Bb.
Input: aabbcc
String accepted! 51 nodes generated.

LHS   RHS
---   ---
S     AX
A     aA
A     aB
A     aB
B     bB
B     bB
Bc    D
Dc    cD
Db    bD
DX    EXc
BX    λ
CE    Ec
BE    Eb
aE    aB

Derived bB from Bb.
Input: aabbcc
String accepted! 51 nodes generated.

LHS  | RHS
---  | ---
S    | AX
A    | aAbCc
A    | aBbc
Bb   | bB
Bc   | D
Dc   | cD
Db   | bD
DX   | EXc
BX   | λ
cE   | Ec
bE   | Eb
aE   | aB

derived A from BX. Derivations complete.
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.
Axiom: $R \sim \## B$

<table>
<thead>
<tr>
<th>LHS</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B$</td>
<td>[$\sim ## TL - B + + B$]</td>
</tr>
<tr>
<td>$L$</td>
<td>[$\text{angle=15 { - g + + g % - g }}$]</td>
</tr>
<tr>
<td>$R$</td>
<td>![@@@ $R$]</td>
</tr>
<tr>
<td>$T$</td>
<td>$T g$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle</td>
<td>15</td>
</tr>
<tr>
<td>color</td>
<td>brown</td>
</tr>
<tr>
<td>polygonColor</td>
<td>forestGreen</td>
</tr>
</tbody>
</table>

$L$-System = $(A, \Sigma, R)$
R \sim \#\# \text{B}
JFLAP v8.0 (Beta)(ex10-tree-thick-fall-leaves.jff)

L System

L-S Render

!@R@R~#@~##Tg[ angle=15 { -g + + g % -g } ] - [ ~##T ]
Add second T rule

### Axiom: R ~ ## B

<table>
<thead>
<tr>
<th>LHS</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>[~,## TL - B ++ B]</td>
</tr>
<tr>
<td>L</td>
<td>[angle=15 { - g + + g % - - g}]</td>
</tr>
<tr>
<td>R</td>
<td>! @@ R</td>
</tr>
<tr>
<td>T</td>
<td>T g</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle</td>
<td>15</td>
</tr>
<tr>
<td>color</td>
<td>brown</td>
</tr>
<tr>
<td>polygonColor</td>
<td>forestGreen</td>
</tr>
</tbody>
</table>

$L\text{-System} = (A, \Sigma, R)$
L-Systems

The same stochastic L-system, rendered 3 different times all at the 9th derivation.
Students like L-systems
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 students thought time and effort using JFLAP helped them get a better grade.
Now a few tips if you ever write educational software...
Make your tool as interactive as possible – but not too tedious!

• User shouldn’t type everything
• Sometimes select
• Example: DFA to regular expression in JFLAP
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
  • Can integrate into ebooks
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

- Algorithm Animation tool

### Rectangle

**Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>a name uniquely identifying this rectangle</td>
</tr>
<tr>
<td>x</td>
<td>x-coordinate</td>
</tr>
<tr>
<td>y</td>
<td>y-coordinate</td>
</tr>
<tr>
<td>width</td>
<td>width of the rectangle</td>
</tr>
<tr>
<td>height</td>
<td>height of the rectangle</td>
</tr>
<tr>
<td>color</td>
<td>color of the rectangle outline</td>
</tr>
<tr>
<td>bgd</td>
<td>color of the rectangle's background</td>
</tr>
</tbody>
</table>

**Example:**

rectangle r1 10 20 100 120 black red
rectangle r2 150 20 180 60 cyan yellow

The first example will create a rectangle with its upper left corner at (10,20) and the rectangle will be red with a black outline, as shown in the figure below on the left. The second example will create a rectangle with its upper corner at (150,20) and the rectangle will be yellow with a cyan outline. This is shown in the figure below on the right.
JAWAA name is not unique

How popular is JAWAA?
FLAP

- **Formal Languages and Automata Package**

- 1996 – converted to Java
- FLAP -> JFLAP
JFLAP name is unique
Much more than Google Analytics
Forums, Blogs, Course websites

Newest 'jflap' Questions - Stack Overflow
stackoverflow.com/questions/tagged/jflap
We can use small letters for terminals and caps for Non-terminals in JFLAP while entering grammar. But this restricts to only 26 options. Can we have more ...

Blog:Recent posts - JFLAP
jflap.wikia.com/wiki/Blog:Recent_posts

CS 301: Using JFLAP
www.cs.colostate.edu/~massey/Teaching/.../JFLAP/gettingstarted.html
This course uses the JFLAP package. According to the JFLAP website, JFLAP is a package of graphical tools which can be used as an aid in learning the basic ...

[PDF] JFLAP Startup
Download JFLAP and the files referenced in this book from www. jflap. org to get started. JFLAP is written in Java to allow it to run on a range of platforms.
Patrons
Hon’ble Shri Dattaji Meghe
Chairman
Shri Sagarji Meghe
Secretary
Shri Sameerji Meghe
Treasurer
Dr. Avichal Kapur
Chief Executive Officer
Dr. M.M. Raghwanshi
Principal

Organizing Committee
Coordinator
Mr. M.M. Goswami
Co-Coordinator
Mr. Amit Khaparde
Mr. Alok Chauhan

ADDRESS FOR CORRESPONDENCE
SDP,
Rajiv Gandhi College of Engg. & Research,
Wanadongri, Hingna Road, Nagpur-441110
Maharashtra, India.
Ph.: 07104-649154,
Fax: 07104-287950
Website:
www.rgcer.edu.in/toc.html

AICTE Sponsored
One Week
Staff Development Programme
Workshop
On
Theory of Computation
09th July to 13th July 2012

Organized by
Department
of
Information Technology
Nagar Yuwak Shikshan Sanstha’s
Rajiv Gandhi College of Engineering & Research, Nagpur (M.S.)
COURSE CONTENTS

- Mathematical Foundation for Theory of computation
- Basic computational devices and their languages
- Finite Automata & Regular Languages
- Context Free Grammar & Context Free Languages, Pushdown Automata
- Turing Machines, Linear Bounded Automata & Context Sensitive Language.
- Undecidability, Reducability
- Practical applications of Theory of Computation
- Recent advancement in Theory of Computation

All the theory sessions will be supported by lab sessions with the hands on Practicals on tools such as LEX, YACC, JFLAP, Grammar Tool Box (GTB), etc.
JFLAP is free

www.jflap.org

JFLAP tutorial
• Introduction

• **CS Concepts Come Alive**
  • Alice Programming Language
  • Algorithm Visualization
  • Automata Theory with JFLAP
  • Additional Ways to Engage with CS

• Diversity Efforts
Middle School students sorting themselves with Bubblesort
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
What happens when your hobby and your career collide?

It is now time for engaging students with edible CS
• 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

• Introduction

• CS Concepts Come Alive
  • Alice Programming Language
  • Algorithm Visualization
  • Automata Theory with JFLAP
  • Additional Ways to Engage with CS

• Diversity Efforts
Success - Alice Excites 4th-6th Grade Girls

- Duke Femmes Event, April 07
- 60 girls – 4 groups of 15
- Taught them Alice for an hour
- Handout to take home
- Event again in 2008, almost every year since
Adventures in Alice Programming
www.cs.duke.edu/csed/alice/aliceInSchools

• 2-week Teacher workshops
  • Over 200 teachers, middle school, high school, some elementary
  • First week Teach Alice, Practice
  • Second week - Develop Lesson Plans
• All disciplines: math, science, history, language arts, foreign language, art, music, business
• Summers 2008-2015, funding for lodging

• Main Sites:
  • Duke University, Durham, NC
  • Charleston/Columbia, SC
  • San Jose, CA (started 2014)
Curriculum Materials
www.cs.duke.edu/csed/alice/aliceInSchools

• Over 90 tutorials available for free
• Beginner, advanced, challenges, projects
• Paper handouts and video
• Over 200 Teacher lesson plans
  • Organized by discipline and grade level
Tutorial for Adventure Game – Find objects in order
Harry Potter Challenge

• Mix of programming and math challenges

_Hailey Programmer and the Goblet of Java_

You will receive a password at the end of each level that will be used to unlock the next level. WRITE THESE DOWN!
If this is your first time playing, select Charms.
Harry Potter – Math/computing
Level 1 Charms - before
Harry Potter – Math/Computing
Level 1 Charms - after
Science Example
How a volcano is formed
What a 6\textsuperscript{th} grader can do with Alice
- teacher Chari Distler
No Superheros in Alice
How Visible are Notable Women in Computer Science?

• Pondered this question in early 2012

• Looked at Wikipedia
  • The internet encyclopedia
  • Who writes those pages?
  • Why did some notables have pages and others not?

• Turing Award Winners
  • Only two women at that time
Fran Allen

- School teacher – got a job at IBM
- Compilers and Optimization Technology
- IBM Fellow – First Women
- Turing Award (2006) – First Woman
- The Turing Award was announced on Feb. 21, 2007
- Her Wikipedia page was created on...
  - Feb. 6, 2007
- On Feb 21, 2007 the Turing Award was added to her Wikipedia page.
Frances E. Allen

From Wikipedia, the free encyclopedia

Fran Allen has made outstanding contributions to the field of programming languages for more than forty-five years, and her work has significantly influenced the wider computer science community.

Ms. Allen is a pioneer in the field of optimizing compilers. Her achievements include seminal work in compilers, code optimization, and parallelization. In the early 1980s, she formed the Parallel TRANslation (PTRAN) group to study the issues involved in compiling for parallel machines. The group was considered one of the top research groups in the world working with parallelization issues. Her work on these projects culminated in algorithms and technologies that form the basis for the theory of program optimization and are widely used in today's commercial compilers throughout the industry.

Ms. Allen's influence on the IBM community was recognized by her appointment as an IBM fellow, the first woman to receive this recognition. She was also president of the IBM Academy of Technology. The Academy plays an important role in the corporation by providing technical leadership, advancing the understanding of key technical areas and fostering communications among technical professionals.

In 1997, Ms. Allen was inducted into the WITI Hall of Fame. Ms. Allen retired from IBM in 2002.
Three days later...

Frances E. Allen

From Wikipedia, the free encyclopedia

Fran Allen is a pioneer in the field of optimizing compilers. Her achievements include seminal work in compilers, code optimization, and parallelization.

In the early 1980s, she formed the Parallel TRANslation (PTRAN) group to study the issues involved in compiling for parallel machines. The group was considered one of the top research groups in the world working with parallelization issues. Her work on these projects culminated in algorithms and technologies that form the basis for the theory of program optimization and are widely used in today's commercial compilers throughout the industry.

Ms. Allen's influence on the IBM community was recognized by her appointment as an IBM Fellow, the first woman to receive this recognition. She was also president of the IBM Academy of Technology. The Academy plays an important role in the corporation by providing technical leadership, advancing the understanding of key technical areas and fostering communications among technical professionals.

In 1997, Ms. Allen was inducted into the WITI Hall of Fame. Ms. Allen retired from IBM in 2002.

This article has not been added to any categories. Please help out by adding categories to it so that it can be listed with similar articles.
In 1997, Ms. Allen was inducted into the WITI Hall of Fame. Ms. Allen retired from IBM in 2002.

Early 2007, she became the first woman to win the A.M. Turing Award.

Categories: Turing Award laureates
In the next three days

- Over 30 edits, added awards, boards

---

**Awards and honors**

Allen is a member of the **National Academy of Engineering**, a fellow of the IEEE, the **Association for Computing Machinery** (ACM) and the **American Academy of Arts and Sciences**. She is currently on the Computer Science and Telecommunications Board, the Computer Research Associates (CRA) board and National Science Foundation's CISE Advisory Board.

In 1997, Allen was inducted into the WITI Hall of Fame.[3] She retired from IBM in 2002 and won the **Augusta Ada Lovelace Award** that year from the Association for Women in Computing. In 2007, she became the first woman to win the A.M. Turing Award.[4]
Frances E. Allen

From Wikipedia, the free encyclopedia

For the early American nun, see Frances Allen (nun).

Frances Elizabeth "Fran" Allen (born August 4, 1932) is an American computer scientist and pioneer in the field of optimizing compilers. Her achievements include seminal work in compilers, code optimization, and parallelization. She also had a role in intelligence work on programming languages and security codes for the National Security Agency.[2][3]

Allen was the first female IBM Fellow and in 2006 became the first woman to win the Turing Award.[4]

Career [edit]

Allen grew up on a farm in Peru, New York and graduated from The New York State College for Teachers (now State University of New York at Albany) with a B.Sc. degree in mathematics in 1954.[8] She earned an M.Sc. degree in mathematics at the University of Michigan in 1957 and began teaching school in Peru, New York.[8]

Deeply in debt, she joined IBM on July 15, 1957 and planned to stay only until her school loans were paid, but ended up staying for her entire 45-year career.

Fran Allen's work has had an enormous impact on compiler research and
Barbara Liskov

- Turing Award (2008)
- Currently Institute Professor at MIT
- One of the first women to get a PhD in CS in the U.S. (1968) – thesis on chess
- Research
  - Venus operating system
  - Design and Implementation of CLU
  - Argus, high-level language for distributed programs
  - Thor, object oriented database system
- Many awards – NAE, AAAS, Fellow ACM
- Had a Wikipedia page since before Turing Award
Barbara Liskov is a professor in the Electrical Engineering and Computer Science department at the Massachusetts Institute of Technology.
39 minutes later ...
Barbara Liskov

From Wikipedia, the free encyclopedia

Barbara Liskov (born November 7, 1939 as Barbara Jane Huberman) is an American computer scientist[2] who is an institute professor at the Massachusetts Institute of Technology and Ford Professor of Engineering in its School of Engineering's electrical engineering and computer science department.[3]

Contents [hide]

1 Life and career
2 Recognition and awards
3 See also
4 References
5 External links

Life and career [edit]

Liskov was born in 1939 California, the eldest of Jane (née Dickhoff) and Moses Huberman's four children.[4] She earned her BA in mathematics at the University of California, Berkeley in 1961. In 1968 she became one of the first women in the United States to be awarded a Ph.D. from a computer science department when she was awarded her degree from Stanford University.[5][6] The topic of her Ph.D. thesis was a computer system for supporting scientific information processing for a scientific collaborative working group. She was a postdoctoral associate at IBM's Thomas J. Watson Research Center from 1969 to 1970. While at IBM, she worked on the design of the System/360 family of computers, and collaborated with Richard Hamming on the development of the SIMULA programming language, which is regarded by many as the first computer programming language designed specifically for scientific computing.

Liskov in 2010.

Born
Barbara Jane Huberman
November 7, 1939 (age 75)
California

Nationality
American

Fields
Computer science
What about other Notable Women in Computer Science?

- ACM Fellows
  - Few women
    - 1994 first year over 130 Fellows
      - 9-12 were women? Less than 10%
  - About 20-50 Fellows per year
  - 2014 – 47 fellows, 6-8 women
  - Noticed few of Women had Wikipedia pages
Investigate New CRA-W Project

• Write Wikipedia pages for Notable women in Computing

• How hard is it to write a Wikipedia page?
  • Lots of rules you have to follow

• Another area with few women
  • 2013 study – 16% of Wikipedia writers are female
Some Rules in Writing Wikipedia Biography pages

• You cannot write your own page!
• Neutral point of view
• Person must be notable
• No original research
  • Must write only facts and reference them
  • Must be verifiable
  • Do not plagiarize – write in your own words
• Regard for subject’s privacy
  • NOT A TABLOID!
Wrote a Wikipedia page

• We had no idea what we were doing....
• At a CRA-W Board meeting in April 2012

• Who to write?
  • Female Turing Award winners had pages
  • All two of them
Mary Jane Irwin

• Professor at Penn State University
• VLSI Architecture and Automated Design
• Board Level Designs
  • Arithmetic Cube, MGAP, and SPARTA
• Architecture, Logic & Circuit Design Tools
  • ARTIST, PERFLEX, LOGICIAN, DECOMPOSER
• Awards
  • National Academy of Engineering
  • ACM Fellow, IEEE Fellow
• No Wikipedia page!
In writing her Wikipedia page,

Here is what happens when you don’t know what you are doing...
What happens when you don’t know what you are doing
Wrote a Guide on How to Write Wikipedia Biography
www.cs.duke.edu/csed/wikipedia

CRA-W and Anita Borg Institute Wikipedia Project
Writing Wikipedia Pages for Notable Women in Computing

About this project

This project started when it was recognized that there are very few notable women (or famous women or leading women) computer scientists who have Wikipedia pages. For example, a large number of women with notable awards such as ACM Fellow, IEEE Fellow, ACM Distinguished Educator, Scientist or Engineer, or other recognitions have no Wikipedia pages. As of April 2012, fewer than...
Our Database of Notable Women in CS

• Over 300 women
• Why notable
• Status of their Wikipedia page
• Forms for adding women and updating status

<table>
<thead>
<tr>
<th>Title/Position</th>
<th>Web page</th>
<th>Prestigious Award or why notable</th>
<th>Wikipedia page?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professor of Human-Computer Interaction, CS</td>
<td><a href="http://www.daimi.au.dk/~bodke">http://www.daimi.au.dk/~bodke</a></td>
<td>Member, CHI Academy</td>
<td>no page</td>
</tr>
<tr>
<td>Founder</td>
<td><a href="http://anitaborg.org/about/histor">http://anitaborg.org/about/histor</a></td>
<td>WITI Hall of Fame, Fellow ACM, EFF Pioneer</td>
<td>has a page</td>
</tr>
<tr>
<td>Mat</td>
<td><a href="http://polaris.gseis.ucla.edu/cb">http://polaris.gseis.ucla.edu/cb</a></td>
<td>ACM Fellow</td>
<td>has a page, needs work</td>
</tr>
</tbody>
</table>
3 female Turing Award Winners

6 women Eniac Programmers

Anita Borg

Grace Hopper

ACM Fellows

Other
Thank You

• Questions?