Computer Science Concepts Come Alive

Susan H. Rodger
Duke University

Feb. 6, 2015
ACM India
Goa, India

Supported by NSF Grants DRL-1031351, DUE 1044191, CRA-W, and IBM Faculty Awards
Thanks for inviting me!

- Tough choice

or

?
This is my classroom now

• Classroom rule:
  
  NO SITTING IN THE LAST FOUR ROWS

• Come forward

• Yes that is YOU who is sitting in the last four rows.
About me

Rode motorbikes in the past

Not sure I could do it here!!!
Graduate School

• PhD Purdue University 1989
  • Computational Geometry
  • Parallel Scheduling Algorithms
Became a professor - 1989

• New Assistant Professor at Rensselaer Polytechnic Institute

• New Course: Combined automata theory with CS1 and CS 2 (data structures)

• Student wanted feedback on all the answers in the book!
Back in 1990
Started developing education tools
Changed area to Visualization Tools and CS Education

- Tool – NPDA - to experiment with pushdown automata
CS Concepts Coming Alive- Back in 1989

• What data structure is this?
What is it?
2D-range tree

• Search in x-y plane
• Main tree organized by x-values
• Subtree organized by y values
1994 – Moved to Duke University
Professor of the Practice

• Position focuses on Education in the Discipline
  • Assistant Prof of the Practice
  • Associate Prof of the Practice
  • Professor of the Practice
Outline

• CS Concepts Come Alive with Software
  • Automata Theory with JFLAP
  • Learning Programming with Alice
  • Algorithm Animation

• Challenges in Designing Educational Software

• CS Concepts Come Alive in other ways
  • Manipulatives
  • Group Activities
  • Edible CS
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

• CS Concepts Come Alive with Software
  – Formal Languages and Automata Theory with JFLAP
  – Learning Programming with Alice
  – Algorithm Animation

• Challenges in Designing Educational Software

• CS Concepts Come Alive in other ways
  – Manipulatives
  – Group Activities
  – Edible CS
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
• Appears to be no programming? Unlike most other CS courses
Students Ready to learn Automata Theory!
Things start well enough ...
But soon, instead of pictures, there are **WORDS**.
Big words! The type with more than one syllable!
VIOLENCE AMONG STUDENTS AS NERVES FRAY!
We only wanted to learn automata theory! Isn’t there a better way?
Try JFLAP ...
Students Learning Automata with JFLAP
Why Develop Tools for Automata?

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_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:

![Automaton Diagram]

Interactive:

![Interactive Automaton Diagram]
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

*Over 20 years!*
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
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!
The diagram represents a finite automaton that recognizes strings with an odd number of 1's. It has the following states:

- **q0**: Initial state, labeled as "odd number of 1's".
- **q1**: State labeled as "only one 0".
- **q2**: State with an edge labeled 0 leading to itself, labeled as "odd number of 1's".
- **q3**: State with two edges labeled 0 and 1 leading to it.
- **q4**: Accepting state, labeled as "odd number of 1's".

The automaton transitions as follows:

- From **q0**, on input 1, transition to **q2**.
- From **q2**, on input 0, transition to itself.
- From **q2**, on input 1, transition to **q4**.
- From **q0**, on input 0, transition to **q1**.

The automaton accepts strings that have an odd number of 1's.
This is a state diagram for recognizing binary numbers.

- **State q0**: Initial state, with an arrow labeled 1 leading to q2.
- **State q1**: Only one 0 is allowed in this state, denoted by the text "only one 0".
- **State q2**: Represents a state with an odd number of 1's, indicated by "odd number of 1's".
- **State q3**: Even number of 1's, ends in 1, denoted by "even number of 1's, ends in 1".
- **State q4**: Final state, with arrows labeled 1 and 0 leading back to q2 and q4, respectively.

The transitions are labeled with 0 and 1, indicating the input symbols.
Test Multiple Inputs
Example: proof
NFA to DFA

1. Start with NFA

2. Construct new DFA
   On q0 with an a, go to q0, q1 and q2

3. Final DFA
DFA to Min DFA

- Start with DFA
DFA to Min DFA(2)

- Start tree of distinguishable states

- Complete tree!
DFA to Min DFA (3)

• Determine states in min DFA

• Add arcs to complete it
Another Example: Grammar

• Grammar – set of replacement rules to define a language

• Examples:
  – Grammar for English
    • defines English sentences
  – Grammar for Python programming language
    • defines syntactically correct programs
  – Grammar for a formal language (simpler)
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
  - aaabbbccc
  - aaaabbbbbcccc
  - aaaaaabbbbbbbcccccc
  - ...

<table>
<thead>
<tr>
<th>S</th>
<th>→ AX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>→ aAbbc</td>
</tr>
<tr>
<td>A</td>
<td>→ aBbbc</td>
</tr>
<tr>
<td>B X</td>
<td>→ λ</td>
</tr>
<tr>
<td>B b</td>
<td>→ bB</td>
</tr>
<tr>
<td>B c</td>
<td>→ D</td>
</tr>
<tr>
<td>D X</td>
<td>→ Exc</td>
</tr>
<tr>
<td>D b</td>
<td>→ bD</td>
</tr>
<tr>
<td>D c</td>
<td>→ cD</td>
</tr>
<tr>
<td>a E</td>
<td>→ aB</td>
</tr>
<tr>
<td>b E</td>
<td>→ Eb</td>
</tr>
<tr>
<td>c E</td>
<td>→ Ec</td>
</tr>
</tbody>
</table>
Example Derivation for aabbcc

S → AX

rule: S -> AX
Example Derivation for aabbcc

\[ S \rightarrow AX \]
\[ \rightarrow aAbcX \]
rule: S -> AX
rule: A -> aAbc
Example Derivation for aabbcc

S $\rightarrow$ AX

$\rightarrow$ aAxbcX

$\rightarrow$ aABbcbcX

rule: S $\rightarrow$ AX

rule: A $\rightarrow$ aAxbc

rule: A $\rightarrow$ aABbc

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

S → AX  
→ aAbcX  
→ aaBbcbcX  
→ aabBcbbcX

rule:  S -> AX  
rule:  A -> aAbc  
rule:  A -> aBbc  
rule:  Bb -> bB
Example Derivation for aabbccc

S → AX  
    → aAbcX  
    → aaBbcX  
    → aabBbcX  
    → 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 → AX  rule: S -> AX
→ aAbcX  rule: A -> aAbc
→ aaBbcbcX  rule: A -> aBbc
→ aabBbcX  rule: Bb -> bB
→ aabDbcX  rule: Bc -> D
→ aabDcX  rule: Db -> bD
Example Derivation for aabbcc

\[
S \rightarrow AX \\
\rightarrow aAbcX \\
\rightarrow aaBbcbcX \\
\rightarrow aabBbcbcX \\
\rightarrow aabDbcX \\
\rightarrow aabbDcX \\
\rightarrow aabbCcDX
\]

rule:  
S \rightarrow AX  
A \rightarrow aAbc  
A \rightarrow aBbc  
Bb \rightarrow bB  
Bc \rightarrow D  
Db \rightarrow bD  
Dc \rightarrow cD
Example Derivation for aabbcc

S → AX
→ aAbcX
→ aaBbcX
→ aabBbcX
→ aabDbcX
→ aabbDcX
→ aabbcDX
→ aabbcEXc

Eventually ... → aabbcc

Note the c spit out on right end!
We could have done this derivation of \texttt{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: aabcbcbc
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

<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.
Spit out the “c” at the right end.
Input: aabbcc
String accepted! 51 nodes generated.

Derived Ec from cE.
String accepted! 51 nodes generated.

Derived Eb from bE.
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 Eb from bE.
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>(\lambda)</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 aB from aE.
<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>

String accepted! 51 nodes generated.

Derived bB from Bb.
Input: aabbcc
String accepted! 51 nodes generated.
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>aA</td>
</tr>
<tr>
<td>A</td>
<td>aB</td>
</tr>
<tr>
<td>Bb</td>
<td>bB</td>
</tr>
<tr>
<td>Bc</td>
<td>bD</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.
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 Eb from bE.
Input: aabccc
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: 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.
Input: aabbcc
String accepted! 51 nodes generated.

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

LHS | RHS
---|---
S  | AX
A  | aA\text{bc}
A  | aB\text{bc}
Bb | bB
Bc | D
Dc | cD
Db | bD
DX | EXc
BX | \lambda
\text{cE} | Ec
bE | Eb
aE | aB

Derived A from BX. Derivations complete.
Parsing in JFLAP

• Brute Force Parsing
  – Reg. Grammars, CFG, unrestricted grammars

• LL(1) Parsing

• SLR(1) Parsing
  – Application with
    • DFA
    • Pushdown Automata
  – Can parse grammars with conflicts!

• CYK Parsing
Example Parsing with SLR

- Ambiguous Grammar
- Will have conflicts in the parse table, but can still parse strings
SLR(1) Parsing

1. Define FIRST and Follow sets
2. Build DFA
3. Define parse table

orange is conflict
Parse of aaba with reduce conflicts

- Parse entry highlighted
- Stack
- Rule used
- Parse tree
Parse of aaba complete
Recall the conflicts

- When click on orange entry, can choose a different entry to resolve conflict
- For both, let’s choose the shift instead of the reduce
Parse of aaba with shift conflicts

• Note tree is a different shape
Comparison Reduce vs Shift Conflicts

With Reduce Entrees

With Shift Entrees
Compare SLR(1) with NPDA

• Convert the CFG to an NPDA

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>→ SS</td>
</tr>
<tr>
<td>S</td>
<td>→ a</td>
</tr>
<tr>
<td>S</td>
<td>→ b</td>
</tr>
</tbody>
</table>

Diagram:

- States: q0, q1, q2
- Transitions:
  - $\lambda, S; \lambda$ from q0 to q1
  - $\lambda, S; \lambda$ from q1 to q2
  - $\lambda, Z; \lambda$ from q2 back to q1
  - $\lambda, b; S$ from q0 to q1
  - $\lambda, a; S$ from q0 to q1
  - $\lambda, SS; S$ from q0 to q1
  - $b, \lambda; b$ from q1 to q2
  - $a, \lambda; a$ from q1 to q2
Trace same string: aaba

- Note the nondeterminism
- Discuss how lookaheads in SLR(1) make it deterministic
Finish the trace: aaba

- 5 paths accepted
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 ## T L - B # 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>

\( L\text{-System} = (A, \Sigma, R) \)
R ~ ## B
[@@R~#][~#TL-B++B]
JFLAP v8.0 (Beta) (exe10-tree-thick-fall-leaves.jff)

L System

L-S Render

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

Pitch 0
Roll 0
Yaw 0
Expansion contains 123 Symbols!
Add second T rule.

Axiom: \( R \sim \#\# B \)

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

**Name** | **Parameter**
---|---
angle | 15
color | brown
polygonColor | forestGreen
Expansion contains 259 Symbols!
L-Systems

The same stochastic L-system, rendered 3 different times all at the 9th derivation.
Students like L-systems
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

• Talk more on using JFLAP with teaching in my talk tomorrow
Two-year JFLAP Study 2005-2007

Fourteen Faculty Adopter 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

-small, large
-public, private
-includes minority institutions
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.
JFLAP 8.0 Beta—Where we are now!

- Formal definition
- Variables and terminals with multiple symbols
- Students: Genkins, McMahon, Godbey, Lin
JFLAP 8.0 Beta (cont)

- Language generator

![JFLAP 8.0 Beta interface with generated strings and grammar elements.](image)
JFLAP 8.0 Beta (cont)

- CYK animation
JFLAP is free

www.jflap.org

JFLAP tutorial
Outline

• CS Concepts Come Alive with Software
  – Automata Theory with JFLAP
  – Learning Programming with Alice
  – Algorithm Animation

• Challenges in Designing Educational Software

• CS Concepts Come Alive in other ways
  – Manipulatives
  – Group Activities
  – Edible CS
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
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
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
Example – Arrays
Shuffle, then Selection Sort

Sort by height
Outline

• CS Concepts Come Alive with Software
  • Automata Theory with JFLAP
  • Learning Programming with Alice
  • Algorithm Animation

• Challenges in Designing Educational Software

• CS Concepts Come Alive in other ways
  • Manipulatives
  • Group Activities
  • Edible CS
Algorithm Animation
Software/Aps/Videos

• AlgoViz.org – collection of algorithm visualizations
• 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.
Electronic Textbooks (ebooks) engage students

- OpenDSA (Shaffer, Virginia 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)
Outline

• CS Concepts Come Alive with Software
  • Automata Theory with JFLAP
  • Learning Programming with Alice
  • Algorithm Animation

• Challenges in Designing Educational Software

• CS Concepts Come Alive in other ways
  • Manipulatives
  • Group Activities
  • Edible CS
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

Enter grammar, press Done when finished.
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>bkgrd</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 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 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 . j flap . org to get started. JFLAP is written in Java to allow it to run on a range of platforms.
### Registration Form

<table>
<thead>
<tr>
<th>Name:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Designation:</td>
<td></td>
</tr>
<tr>
<td>Qualification:</td>
<td></td>
</tr>
<tr>
<td>Organization/Institute/University:</td>
<td></td>
</tr>
<tr>
<td>Address for Correspondence:</td>
<td></td>
</tr>
</tbody>
</table>

Phone(Mobile):... Email:...

Gender & Age:...

Total Teaching / Research / Development Experience:...

Accommodation Required: YES / NO

D. D. No: ....... Dated: ...... Amount: ......... Name of Bank: .........

Date:............ Signature of the Candidate

Certified that the candidate is employed in our institute and is sponsored to attend the course. He will be relieved for the said period if selected for the above course. The Candidate will abide by the rules and regulations governing the staff development program and will attend with full sincerity.

Date:............ Signature of the Principal with seal

### Patrons

<table>
<thead>
<tr>
<th>Name:</th>
<th>Position:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hon’ble Shri Dattaji Meghe</td>
<td>Chairman</td>
</tr>
<tr>
<td>Shri Sagarji Meghe</td>
<td>Secretary</td>
</tr>
<tr>
<td>Shri Sameerji Meghe</td>
<td>Treasurer</td>
</tr>
<tr>
<td>Dr. Avichal Kapur</td>
<td>Chief Executive Officer</td>
</tr>
<tr>
<td>Dr. M.M. Raghunwanshi</td>
<td>Principal</td>
</tr>
</tbody>
</table>

### Organizing Committee

<table>
<thead>
<tr>
<th>Coordinator:</th>
<th>Name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mr. M.M. Goswami</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Co-Coordinator:</th>
<th>Name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mr. Amit Khaparde</td>
<td></td>
</tr>
<tr>
<td>Mr. Alok Chauhan</td>
<td></td>
</tr>
</tbody>
</table>

### 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](http://www.rgcer.edu.in/toc.html)
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.
Outline

• CS Concepts Come Alive with Software
  – Automata Theory with JFLAP
  – Learning Programming with Alice
  – Algorithm Animation

• Challenges in Designing Educational Software

• CS Concepts Come Alive in other ways
  – Manipulatives
  – Group Activities
  – Edible CS
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
Paper Fetch

If you've ever seen one of those wonders of nature in movies with lizards that catch flies on their tongues, you'll pick up this game much more quickly.

1. Lay five or six pieces of paper on the floor, in a circle with a diameter of, say, 6 or 7 feet. (The bigger the circle, the harder the game.) Get a partner with a similar fascination with lizards.
Cards

• Insertion Sort
• Card Class – shuffling, dealing hands
• Poker hands – Full house, Flush, etc.
Royal Flush
Notable Women in Computing Cards
bit.ly/NotableW

• Based on Wikipedia project – wrote guide on how to write a Wikipedia page on a Notable women in Computing
• Picked 54 Women - deck of cards
• Page on using cards to teach CS
• Poster of the women
3 female Turing Award Winners

6 women Eniac Programmers

Anita Borg

Grace Hopper

ACM Fellows

Other
Outline

• CS Concepts Come Alive with Software
  – Automata Theory with JFLAP
  – Learning Programming with Alice
  – Algorithm Animation
• Challenges in Designing Educational Software
• CS Concepts Come Alive in other ways
  – Manipulatives
  – Group Activities
  – Edible CS
CS Unplugged

• Great activities for exploring computer science concepts – without a computer
Middle School students sorting themselves with Bubblesort
Outline

• CS Concepts Come Alive with Software
  • Automata Theory with JFLAP
  • Learning Programming with Alice
  • Algorithm Animation
• Challenges in Designing Educational Software
• CS Concepts Come Alive in other ways
  • Manipulatives
  • Group Activities
  • Edible CS
About Me - Hobby – Baking Shape cakes

The Wiggles magazine Issue No. 42
How do you make those cakes?
What happens when your hobby and your career collide?

It is now time for engaging students with edible CS
Automata Theory
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
CS 2 – Data Structures
Red-Black Tree (cookies)
CS 1
Sorting Cookies
Discrete math

A graph and its Dual Graph
Discrete Math
Stacking Pancakes Problem
Alice Programming Language
More Alice Programming Language
Conclusions

• We have shown several ways for CS concepts to come alive

• Questions?