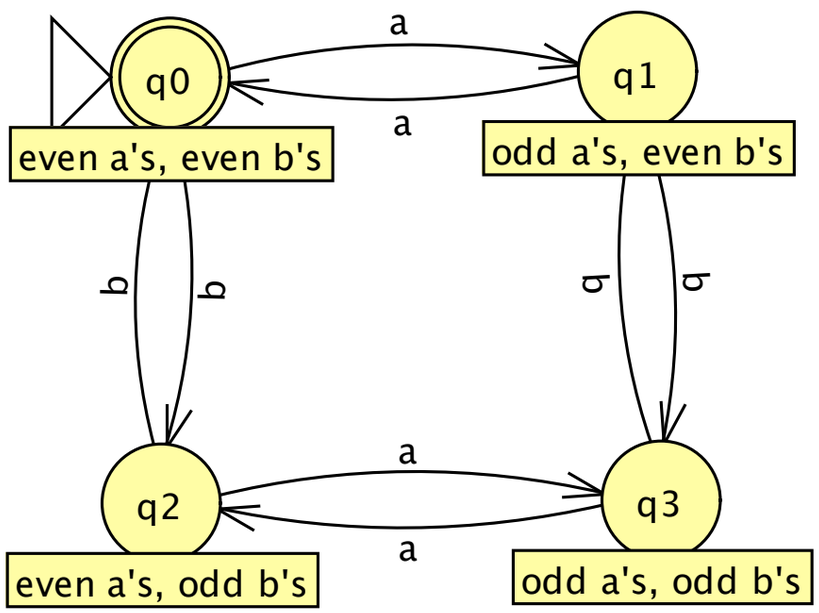
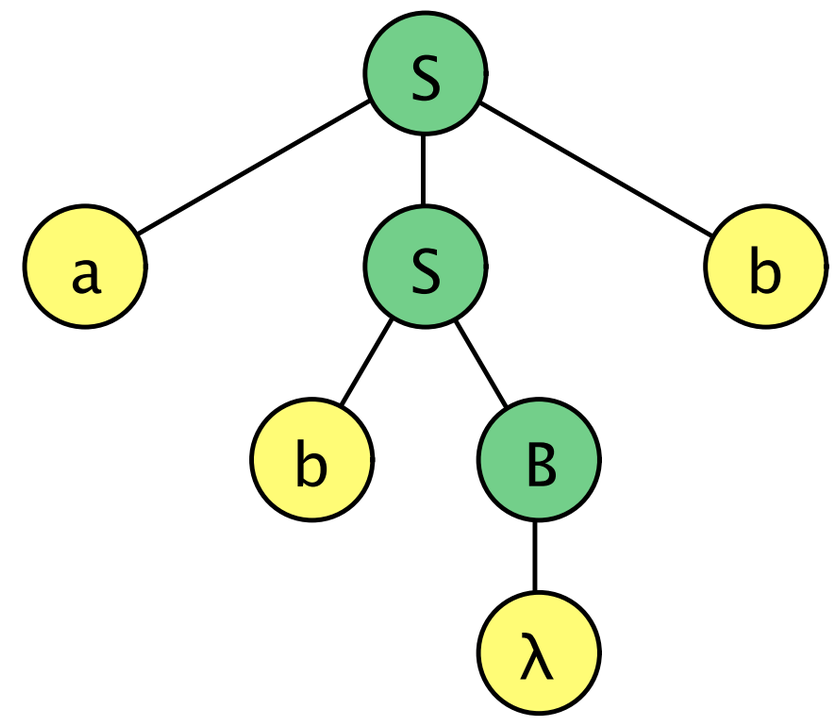
A Visual and Interactive Automata Theory Course with JFLAP 4.0



Ryan Cavalcante
Thomas Finley
Susan H. Rodger



Department of Computer Science
Duke University

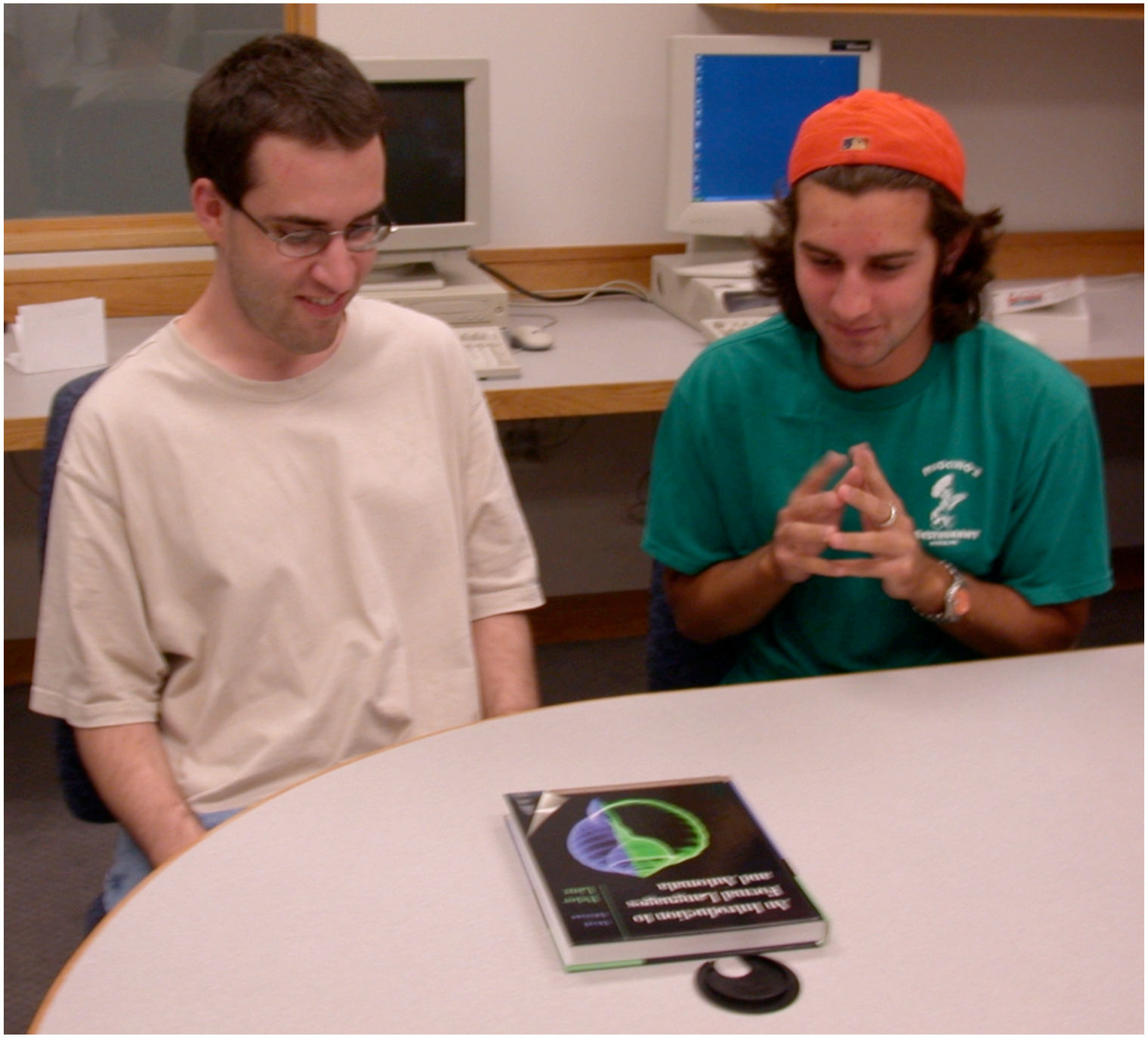
March 4, 2004

This work was supported by:
National Science Foundation DUE-9752583

Outline

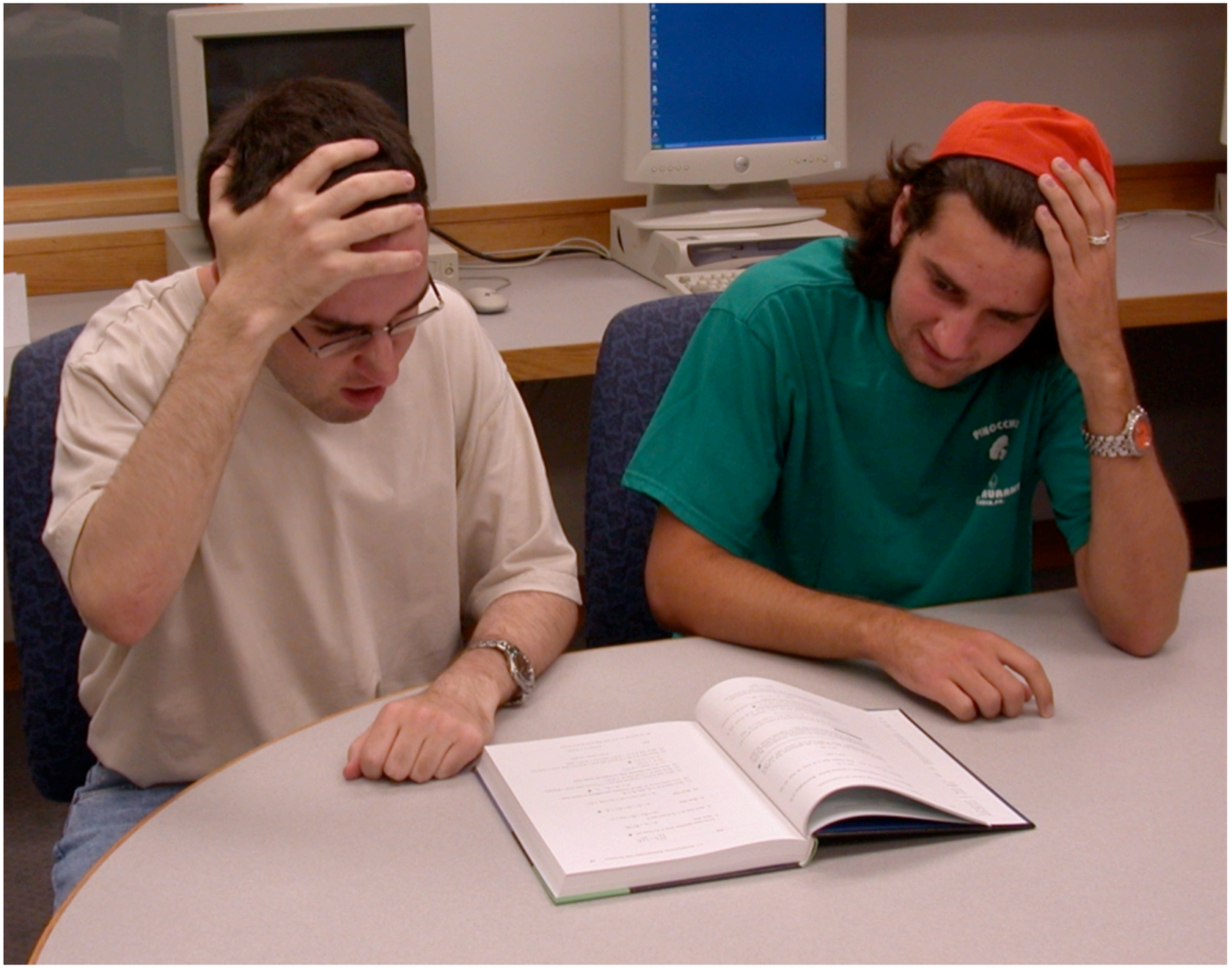
- Why develop Tools?
- Previous Work
- What is JFLAP?
- How Does JFLAP Fit In Automata Course?
- JFLAP's Use in the Course
- New Features of JFLAP
- Feedback and Use of JFLAP
- JFLAP's Future

Students Ready to Learn Automata Theory!



Things start well enough...

But soon, instead of pictures, there are ***WORDS***.

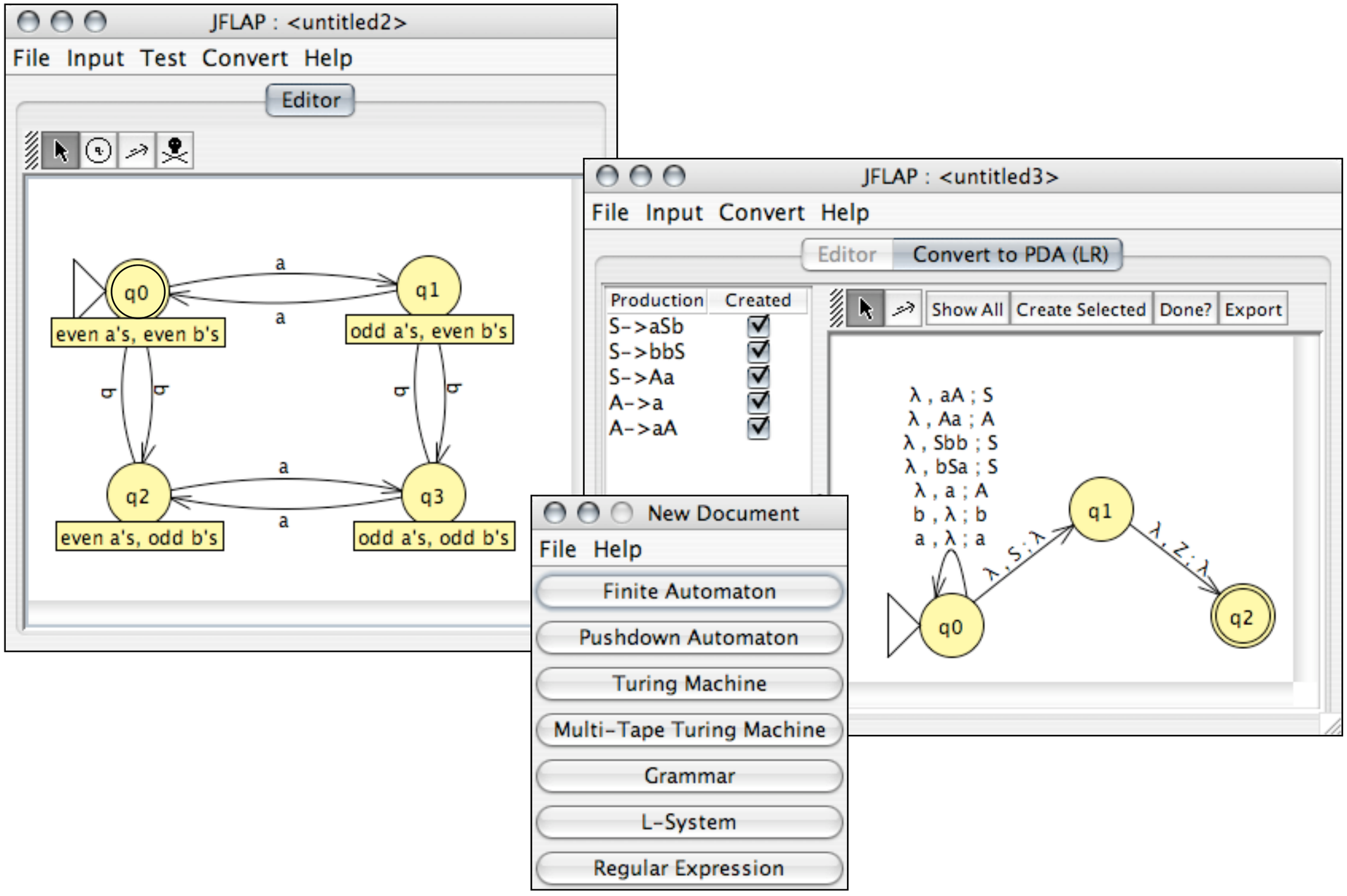


Big words! The type with more than one syllable!

VIOLENCE AMONG STUDENTS AS NERVES FRAY!

We only wanted to learn automata theory! Isn't there a better way?

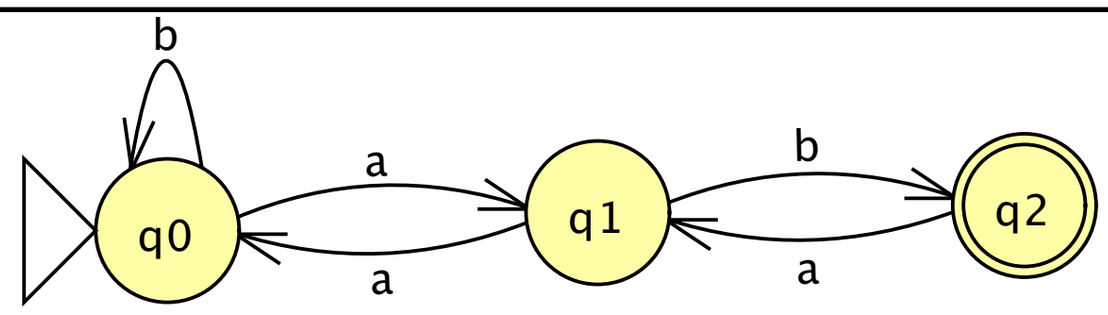
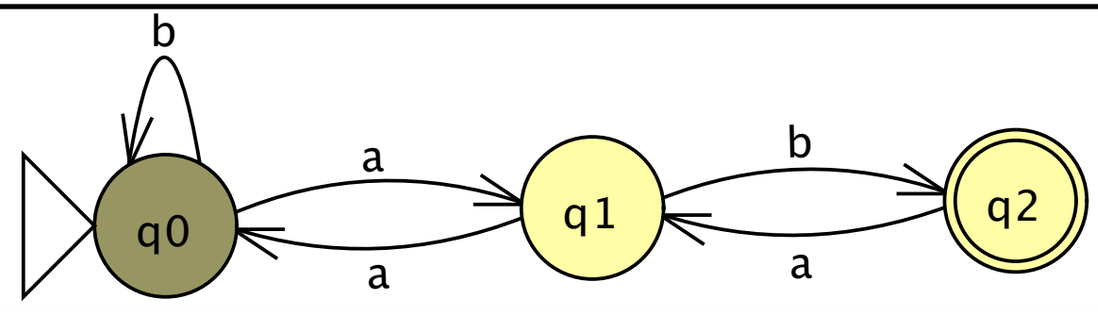
Try JFLAP ...


JFLAP : <untitled2>
File Input Test Convert Help
Editor
q0
q1
q2
q3
a
b
even a's, even b's
odd a's, even b's
even a's, odd b's
odd a's, odd b's
JFLAP : <untitled3>
File Input Convert Help
Editor
Convert to PDA (LR)
Production
Created
S->aSb
S->bbS
S->Aa
A->a
A->aA
Show All
Create Selected
Done?
Export
λ , aA ; S
λ , Aa ; A
λ , Sbb ; S
λ , bSa ; S
λ , a ; A
b , λ ; b
a , λ ; a
λ , S ; λ
λ , Z ; λ
New Document
File Help
Finite Automaton
Pushdown Automaton
Turing Machine
Multi-Tape Turing Machine
Grammar
L-System
Regular Expression

Students Learning Automata with JFLAP

Why Develop Tools for Automata Theory?

Different Representations of Presentation

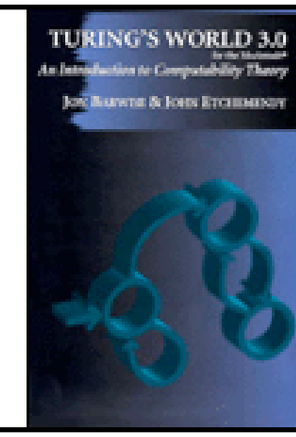
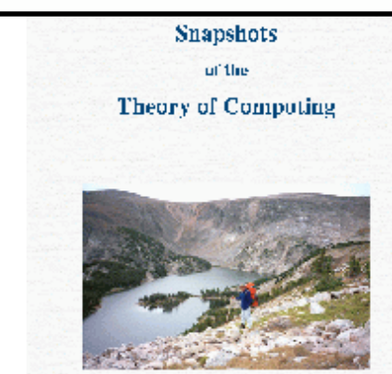
<table>
<tr><td>Textual</td><td>$(\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$
$\delta = \{(q_0, b, q_0), (q_0, a, q_1), (q_1, a, q_0), (q_1, b, q_2), (q_2, a, q_1)\}$</td></tr>
<tr><td>Tabular</td><td><table>
<tr><td></td><td>a</td><td>b</td></tr>
<tr><td>q_0</td><td>q_1</td><td>q_0</td></tr>
<tr><td>q_1</td><td></td><td>q_2</td></tr>
<tr><td>q_2</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>Visual</td><td> 
</td></tr>
<tr><td>Interactive</td><td> 
</td></tr>
</table>

Why Develop Tools for Automata Theory?

Different Representations of Presentation Examined 10 automata theory textbooks in past 11 years (Taylor, Cohen, Sipser, Kelley, Linz, Sudkamp, Lewis, Kinber, Hopcroft, Martin)

- One had software integrated for part of the book (Taylor)
- All had visual representation of DFA
- Only 6 had visual representation for PDA
  Those had few visual examples, only 2 or 3 states
- Only 6 had visual representation for Turing machine
  3 of those switched different visual representation
- Only 2 had picture of NPDA created in CFG to NPDA transformation
- None had picture of parse tree for unrestricted grammar

Previous Work on Automata Tools by Others

Turing's World - Barwise and Etchemendy (1993)	
Deus Ex Machina - Savoiu (seven models of computation) Models of Computation and Formal Languages - Taylor (1998)	
Snapshots of the Theory of Computing - Ross (2002)	

Our Previous Work on Automata Tools

- JFLAP - creating and experimenting with automata and grammars
  (SIGCSE 2000, SIGCSE 1999, SIGCSE 1997)
- JeLLRap - LL and LR parsing
  (SIGCSE 1997)
- Pâté - Brute force parser, grammar transformer
  (SIGCSE 1997)
- Lsys - creating L-systems

The new JFLAP incorporates concepts from all of these.

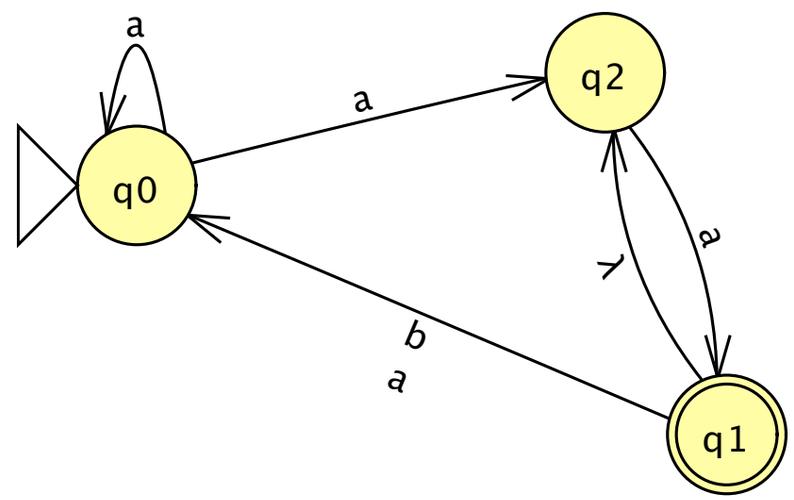
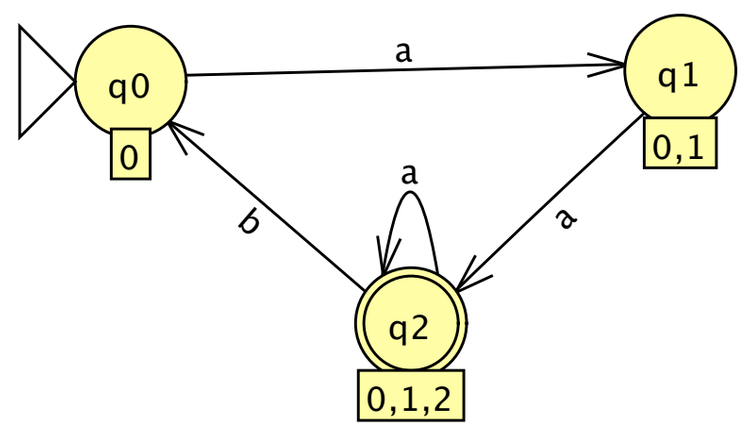
Thanks to Students Who Have Worked on JFLAP and Automata Theory Tools

- 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

What is JFLAP?

Java **F**ormal **L**anguages and **A**utomata **P**ackage

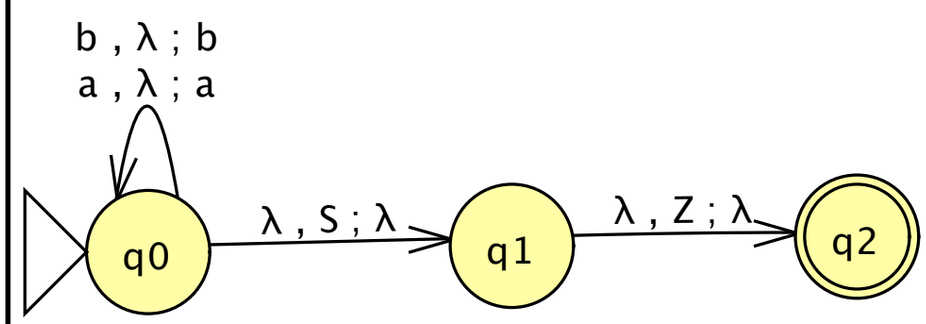
Instructional Tool to learn concepts of Formal Languages and Automata Theory

Regular languages - create • DFA • NFA • regular grammar • regular expression	
Regular languages - conversions • NFA→DFA→Minimal DFA • NFA ↔ regular expression • NFA ↔ regular grammar	

What is JFLAP? (cont)

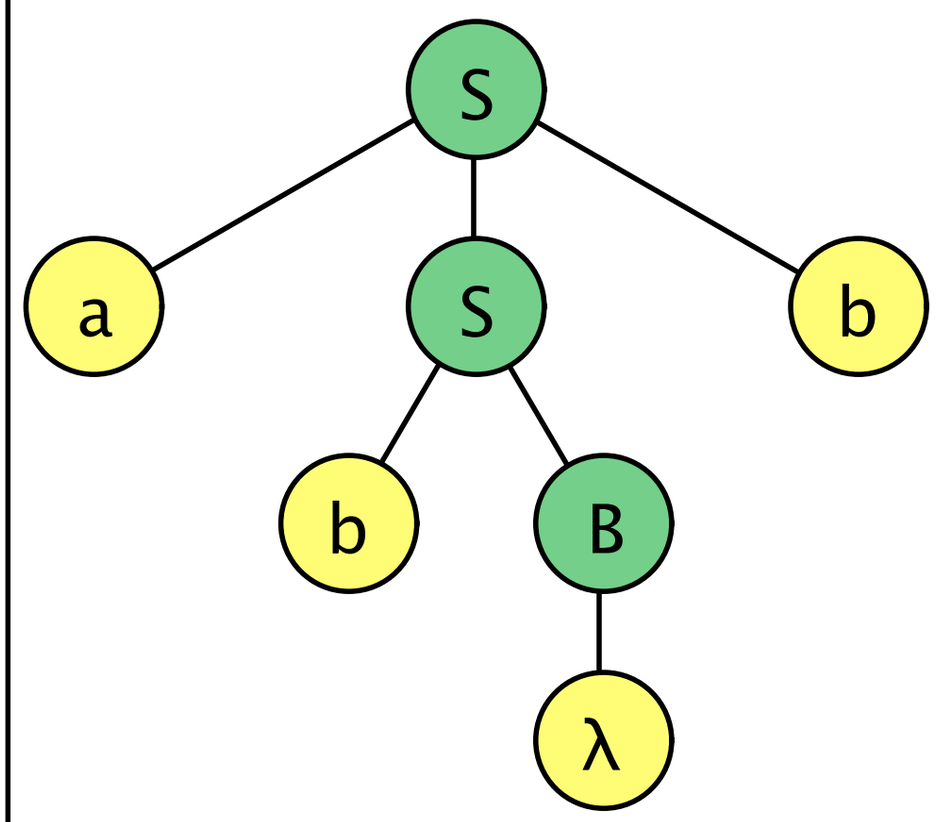
Context-free languages - create

- push-down automaton
- context-free grammar



Context-free languages - transform

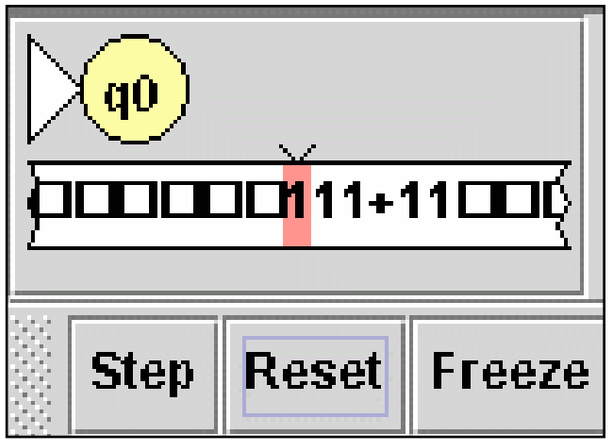
- PDA → CFG
- CFG → PDA (LL parser)
- CFG → PDA (LR parser)
- CFG → CNF
- CFG → LL parse table and parser
- CFG → LR parse table and parser
- CFG → Brute force parser



What is JFLAP? (cont)

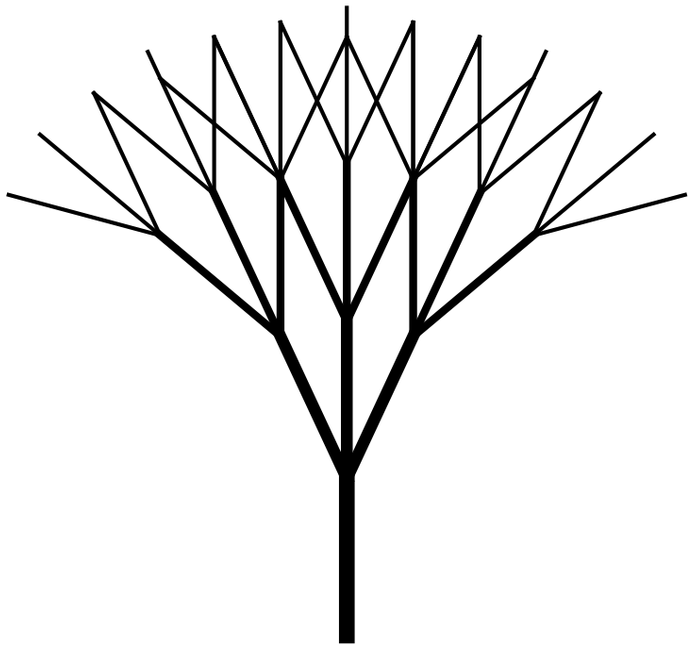
Recursively Enumerable languages

- Turing machine (1-tape)
- Turing machine (multi-tape)
- unrestricted grammar
- unrestricted grammar
  → brute force parser



L-Systems

- Create L-systems



How JFLAP Fits Into Topics In Formal Languages Course

Topic	JFLAP 3.1	JFLAP 4.0
Finite Automata	1	1
Regular Grammars/Expressions	¾	1
Properties of Regular Languages		½
Context-Free Grammars	½	1
Simplify CFL		1
Pushdown Automata	1	1
Properties of CFL		
Turing Machine (1-Tape)	½	¾
Other Models of TM	¼	½
Recursively Enumerable Languages		¼
LL and LR Parsing		1
L-Systems		1

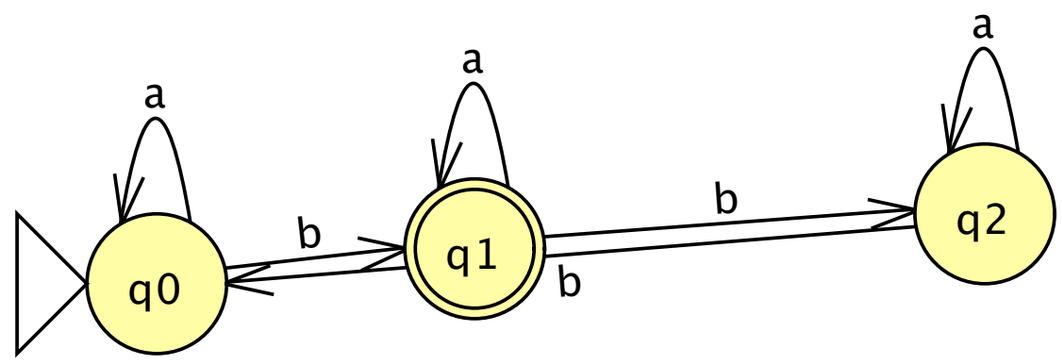
How JFLAP Fits Into Topics In Formal Languages Course Summary

- JFLAP 3.1 covers 4 chapters of material spread out over 6 chapters.
- JFLAP 4.0 covers 9 chapters of material spread out over 11 chapters.

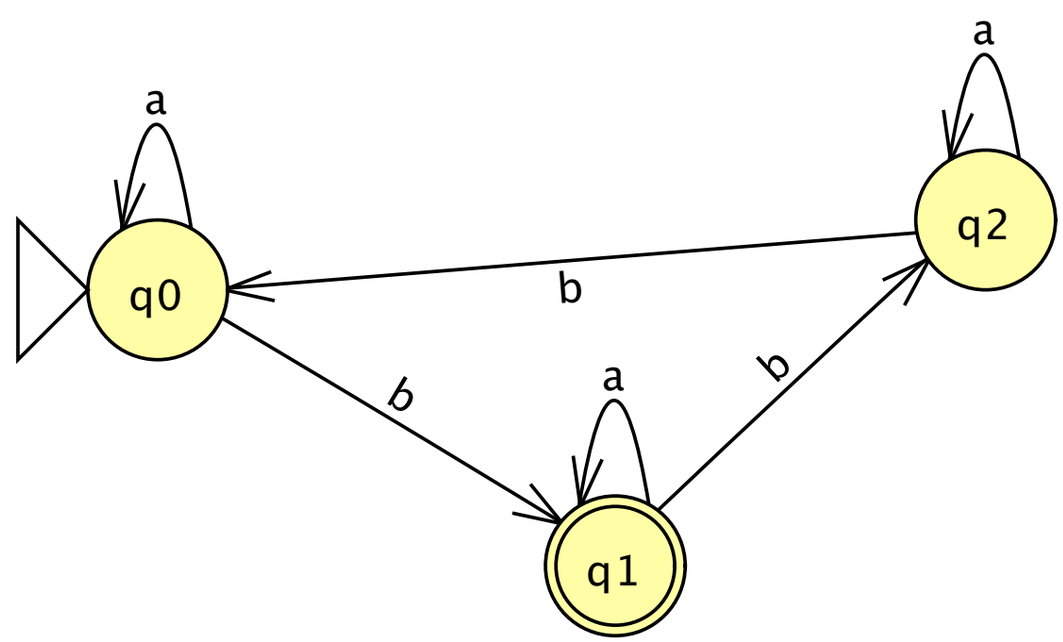
Use of JFLAP by Instructor

Showing how to layout items

Poor:

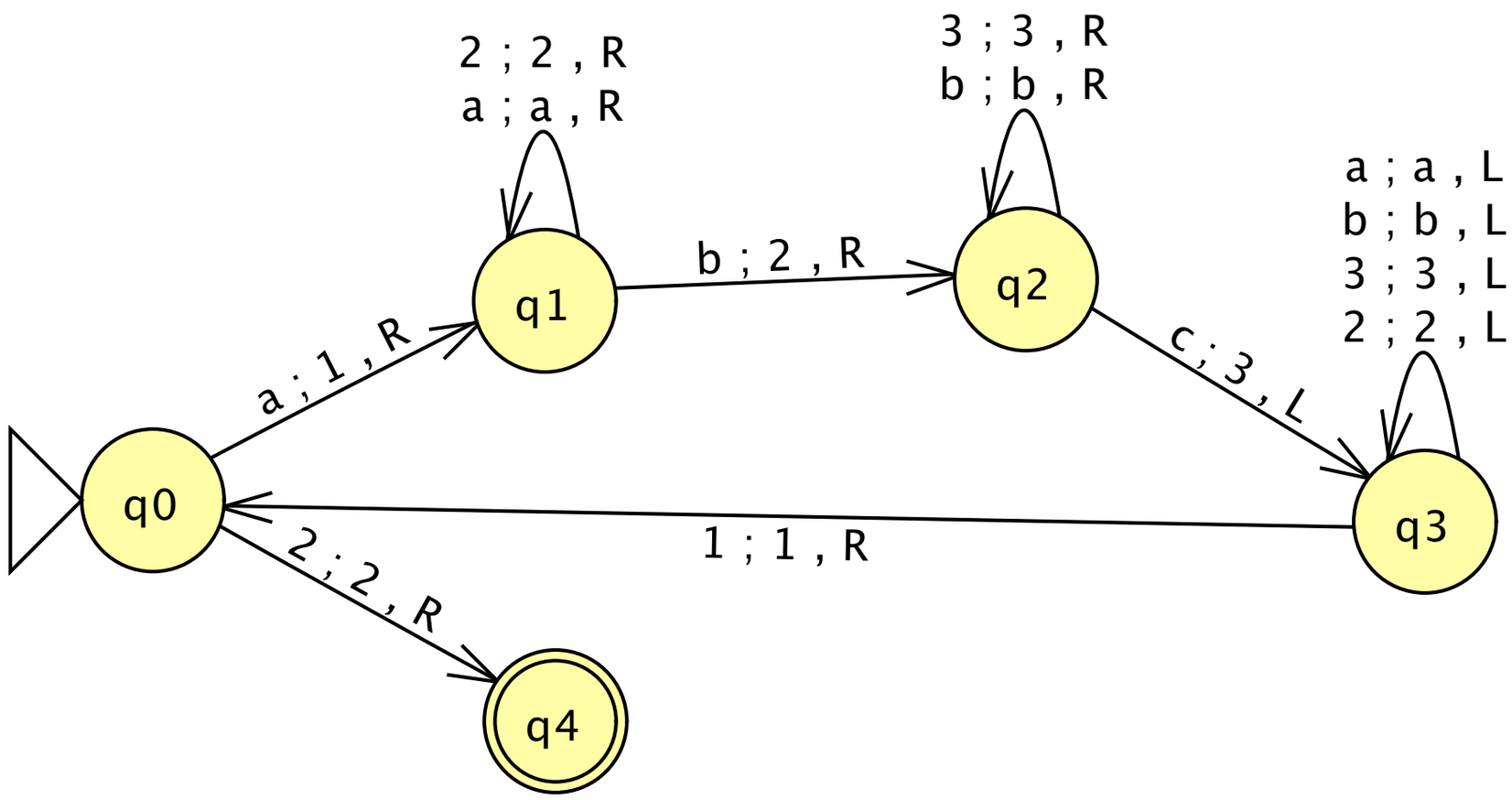


Better:



Use of JFLAP by Instructor

Is this correct for $a^n b^n c^n$?



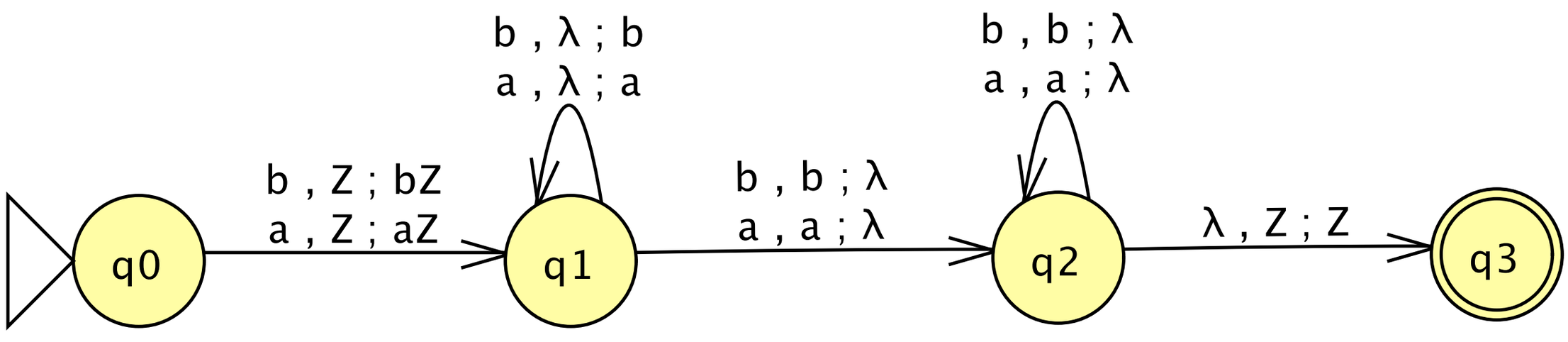
How do we fix it?

Use of JFLAP by Instructor

Experimenting with Difficult Concepts

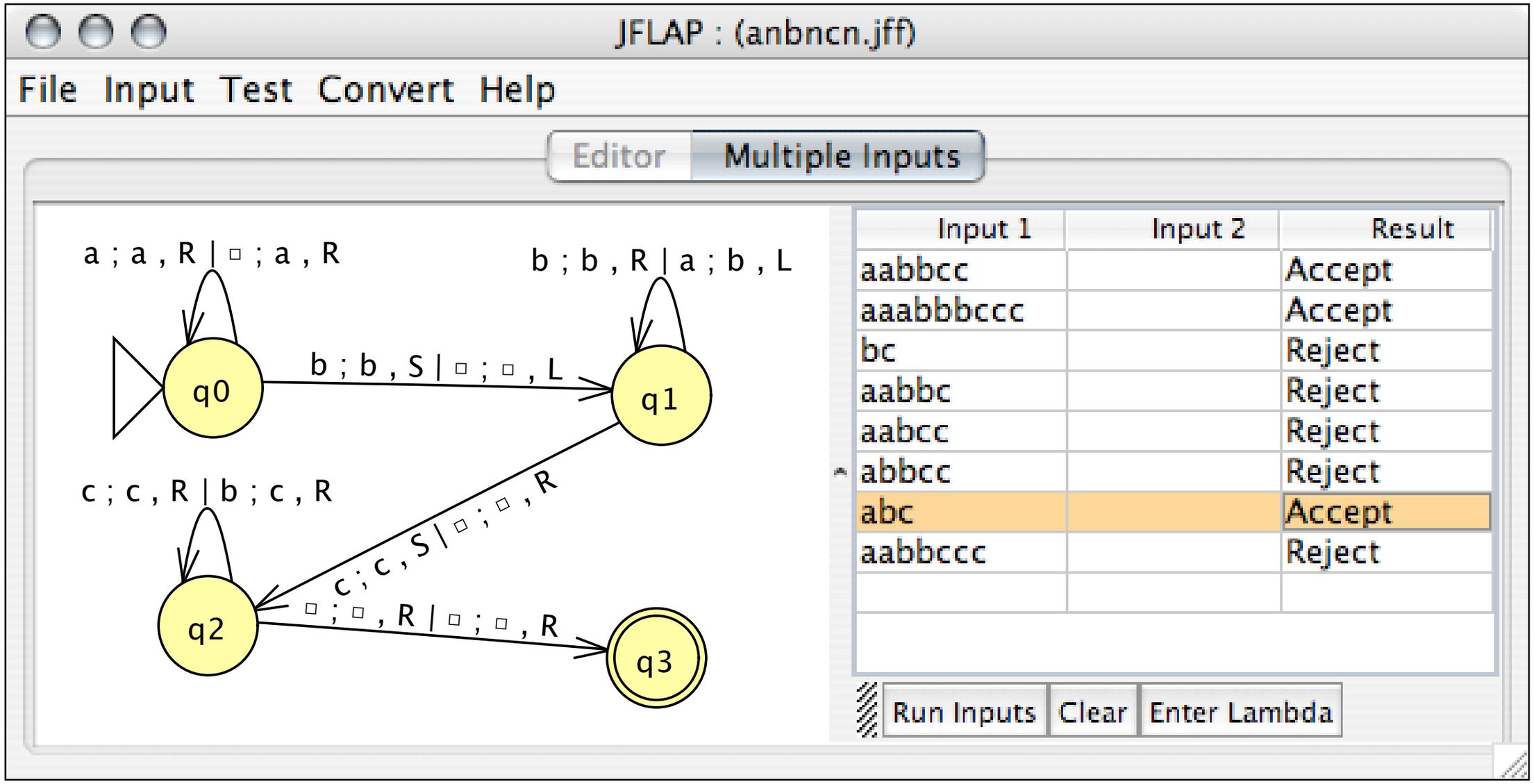
Nondeterminism: ww^R

- Students attempt at desk - difficult: want to find the "middle"
- Instructor solves with class using JFLAP



Use of JFLAP by Instructor

Testing Student Programs



Use of JFLAP by Instructor

Relate to other CS Concepts

Running Time

- Consider $a^n b^n c^n$
  - one-tape TM $O(n^2)$
  - two-tape TM $O(n)$

Other Uses of JFLAP by Instructor

- Demonstrate Nondeterminism
- Demonstrate the running of a CFG to a PDA using LR method

  Which lookahead do you choose?
- Demonstrate a transformation from one form to another

  Example: PDA to CFG
- And many other uses...

JFLAP Student Use

- Recreate and experiment with instructor's examples
- Use with Homework
- A study aid - create additional examples
  - explore concepts in depth
  - weaker students get more feedback

Feedback from Students in CPS 140 Duke University - Spring 2003

- Used JFLAP and tools in 6 of 9 homeworks
- Questionnaire - 33 responses
  - "Was JFLAP easy to use?"
    All 33 answered yes.
  - "Did you look at the help at all? If so, what part did you look at and was it helpful?"
    6 used help and found it helpful, 27 did not look at help.
  - "Do you prefer creating FA using JFLAP or drawing them on paper?"
    17 students - prefer to use JFLAP
    12 students - prefer paper first, then JFLAP for testing
    2 students - prefer paper

New Features in JFLAP 4.0

- RE ↔ FA reworked
- Transform CF Grammars to Chomsky
- LL(1) and SLR(1) parsing
- Brute force parsing
- L-Systems

Minor New Features in JFLAP 4.0

- Compare Equivalence
- 3-5 Tape Turing Machines
- Combine Automata
- Graph Layout

Finite Automata Editing and Simulation

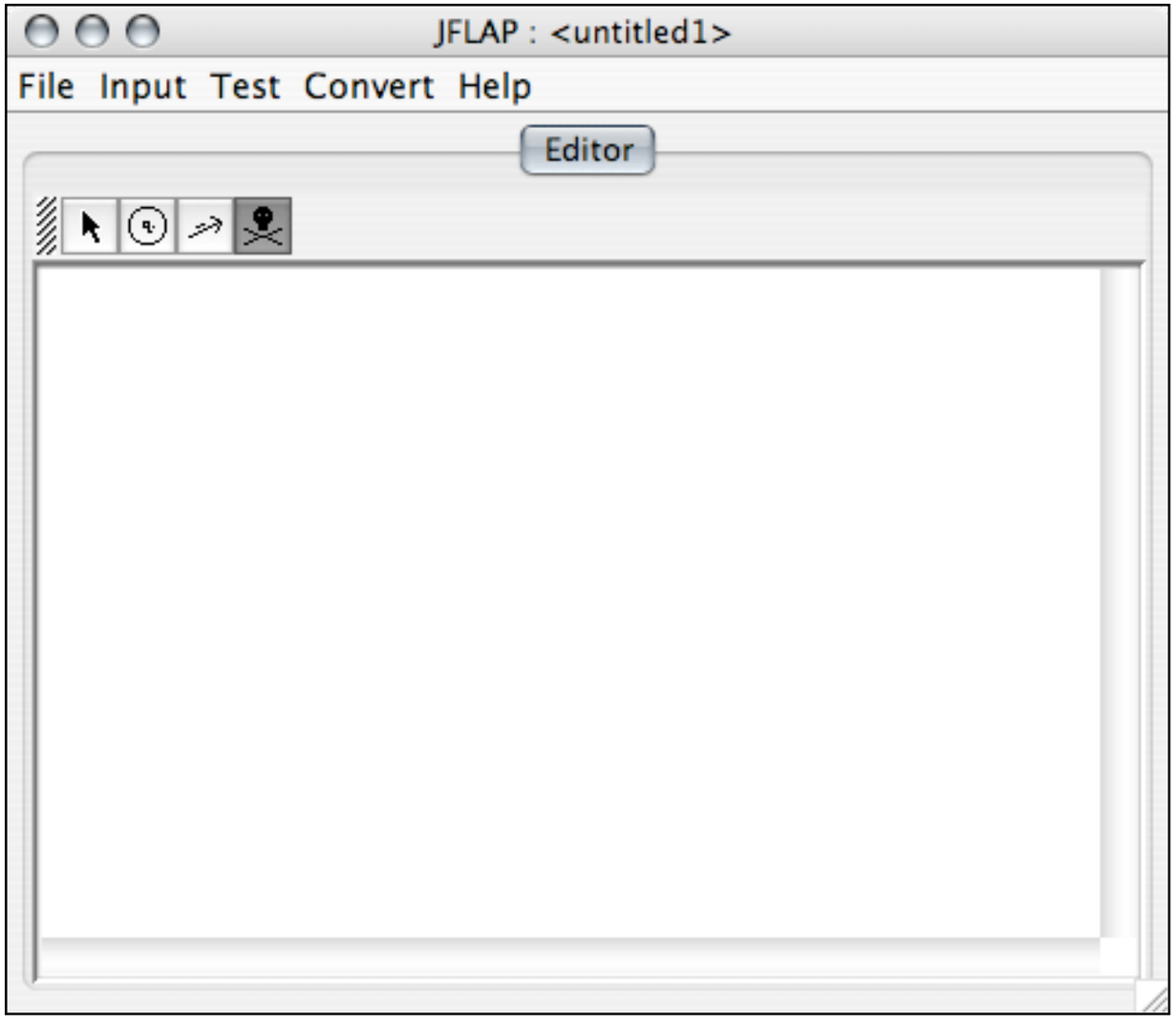
- The most basic feature of JFLAP has always been the creation of automata, and simulation of input on automata.
- Here we demonstrate the creation and simulation on a simple NFA.

FA Edit & Simulation
Start up JFLAP

- When we start up JFLAP we have a choice of structures.

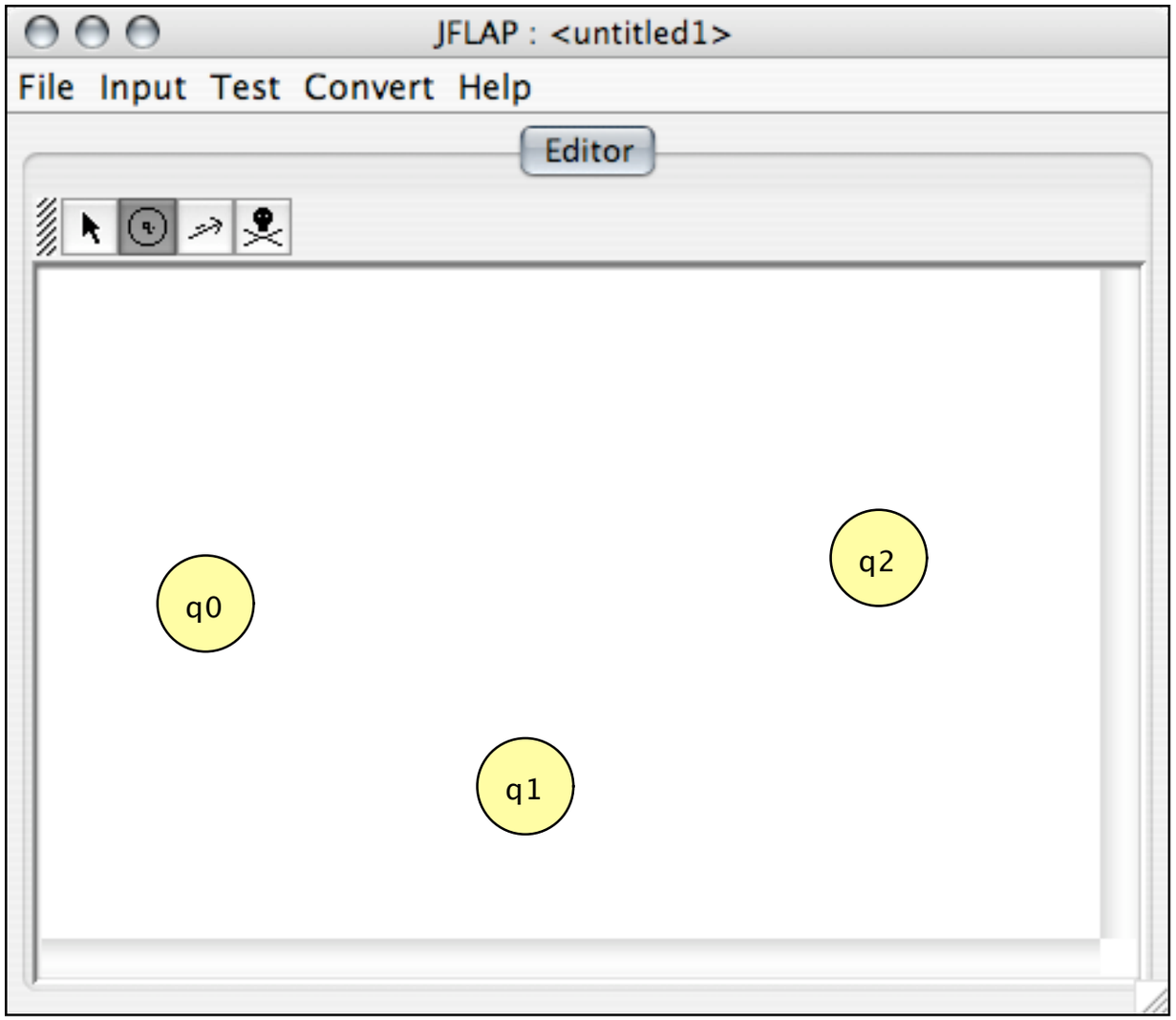
- The first of these is the Finite Automata!

FA Edit & Simulation
Start Editing!



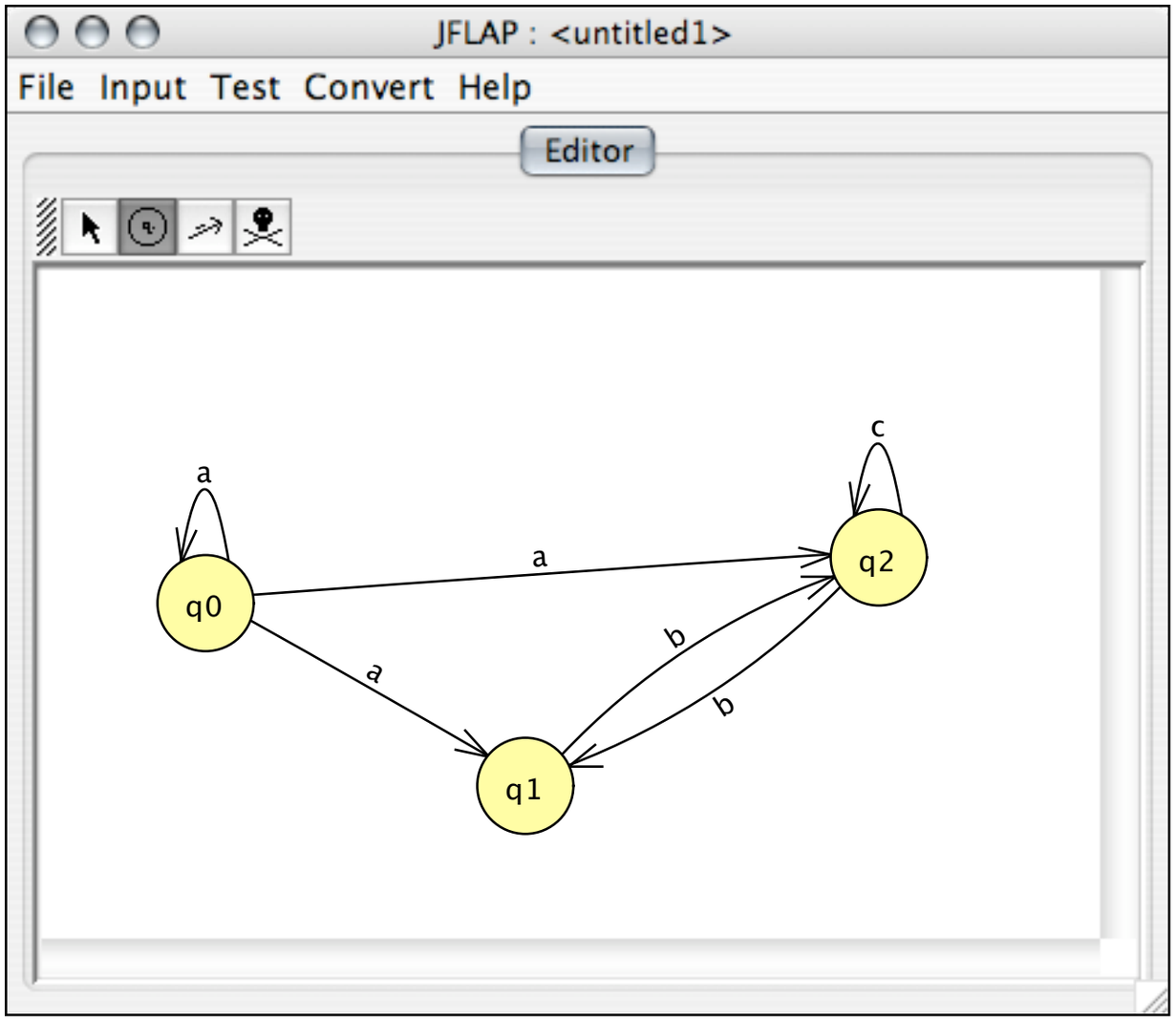
- We start with an empty automaton editor window.

FA Edit & Simulation
Create States



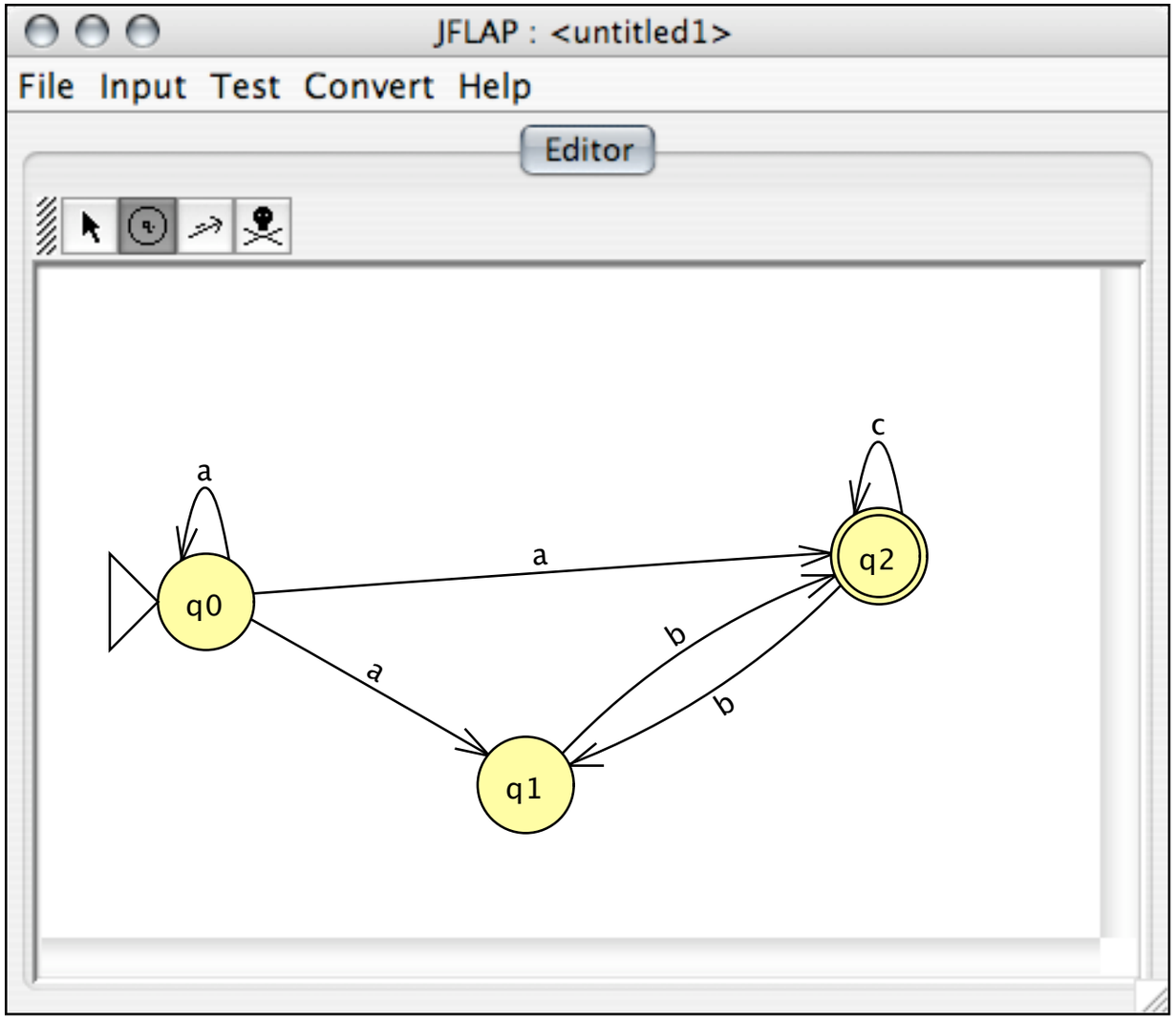
- We create some states ...

FA Edit & Simulation
Create Transitions



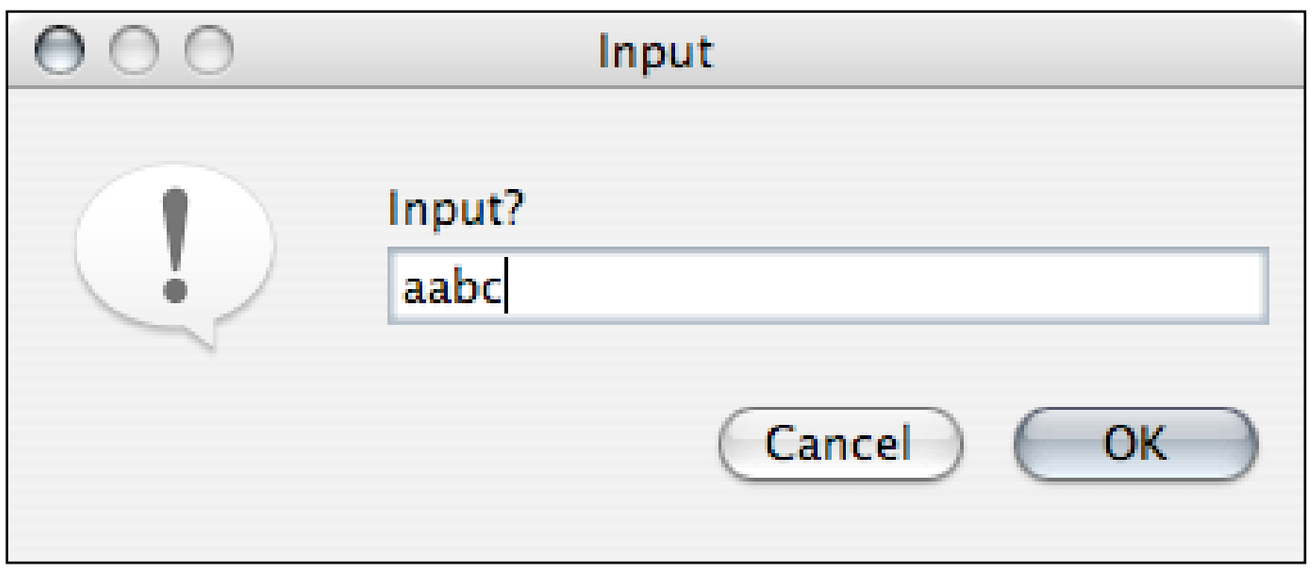
- We create some transitions ...

FA Edit & Simulation Initial and Final State



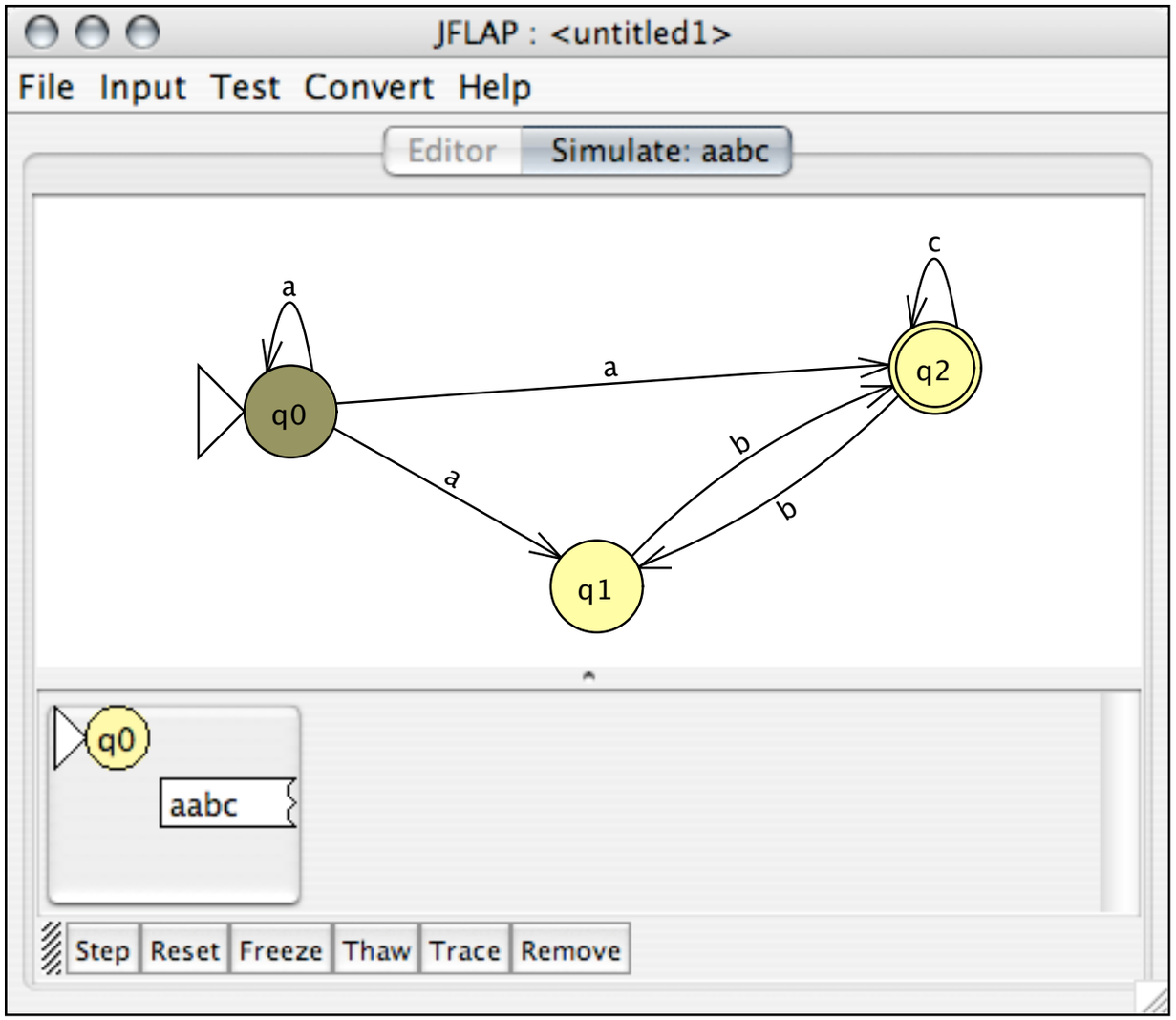
- We set an initial and final state.
- Now we can simulate input on this automaton!

FA Edit & Simulation Input to Simulate...



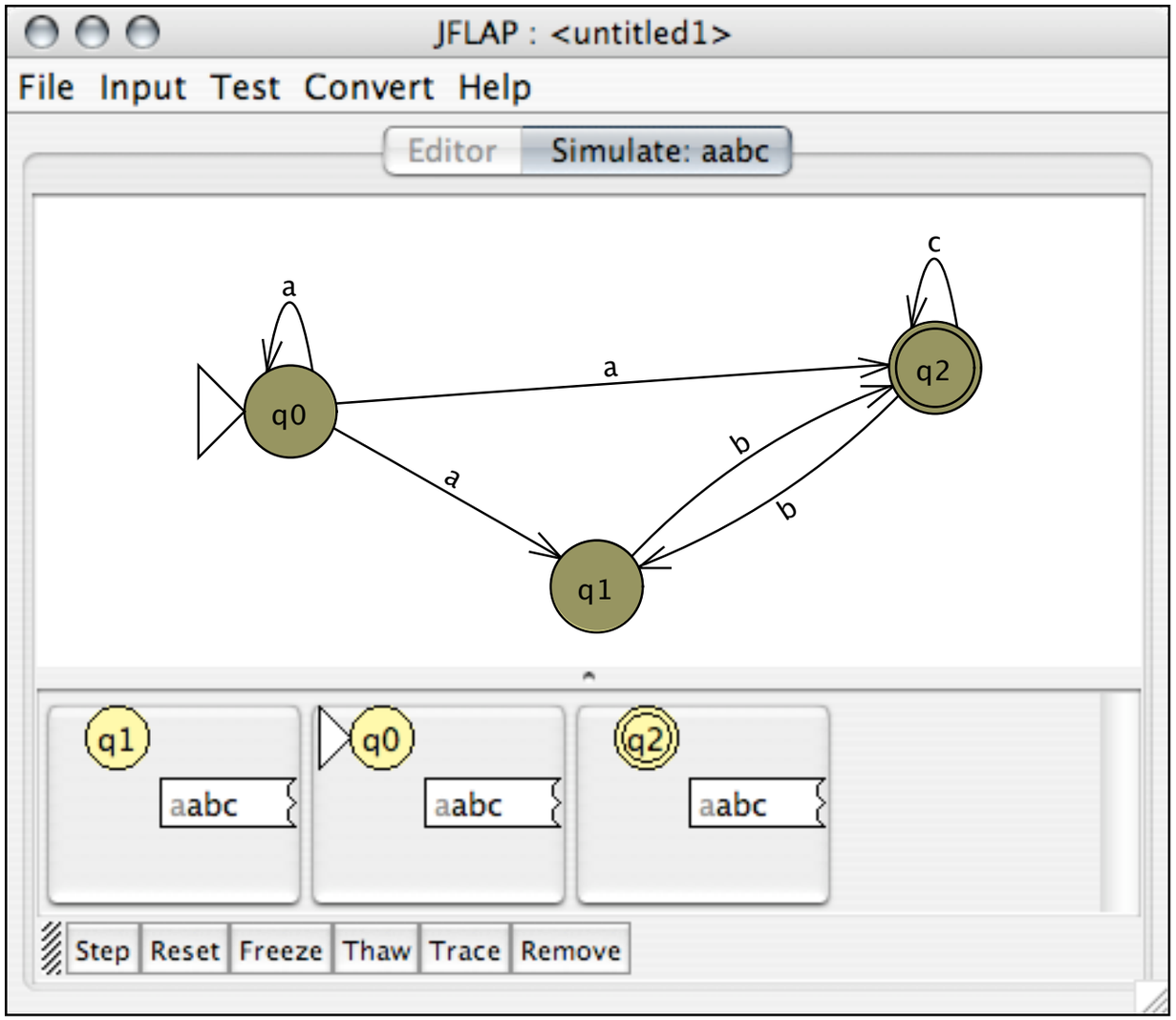
- When we say we want to simulate input on this automaton, a dialog asks us for the input.

FA Edit & Simulation Start Simulation!



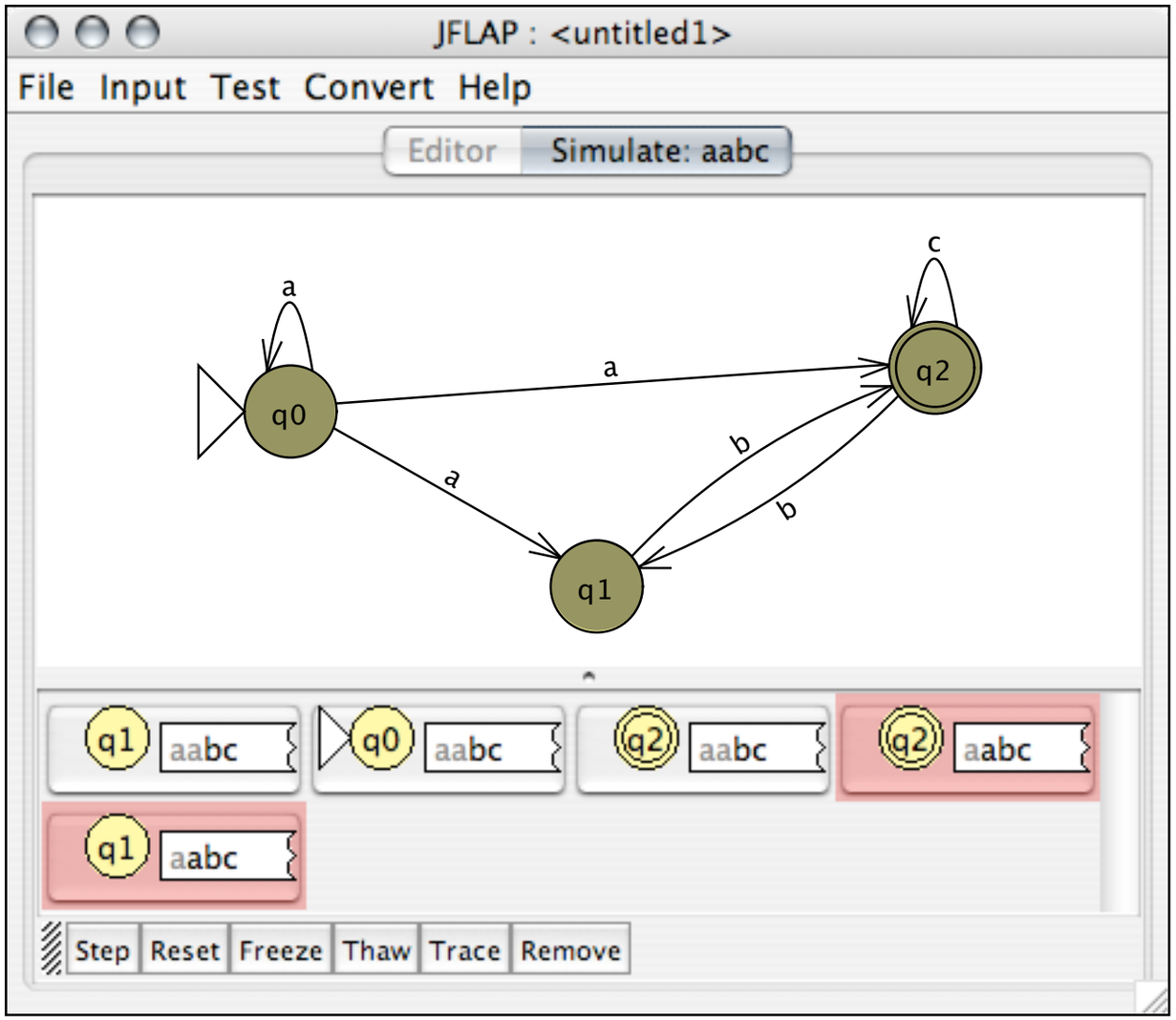
- When simulation starts, we have a configuration on the initial state with all input remaining to be processed.

FA Edit & Simulation After One Step



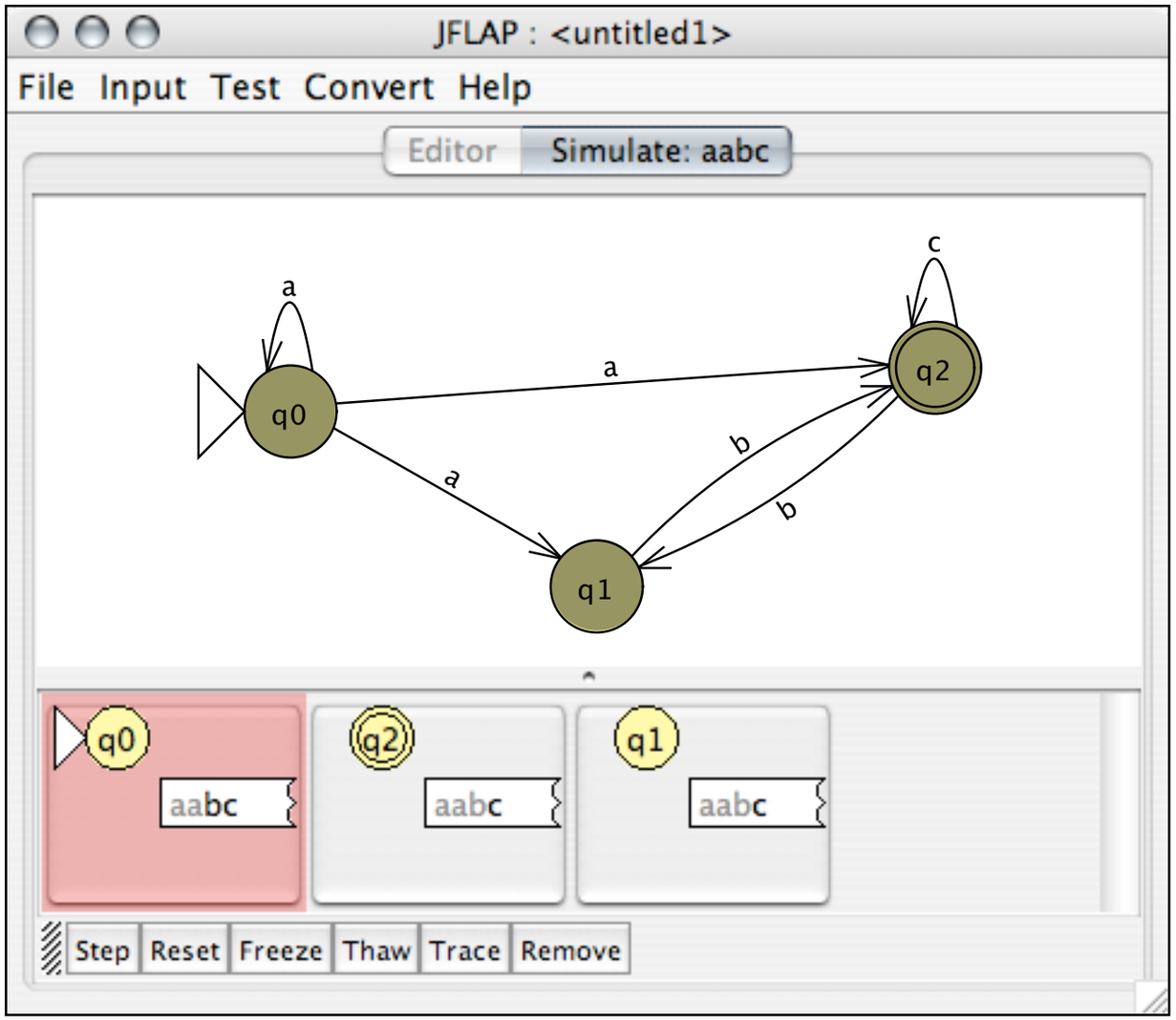
- This is a nondeterministic FA, and on this input we have multiple configurations after we "Step."

FA Edit & Simulation After Two Steps



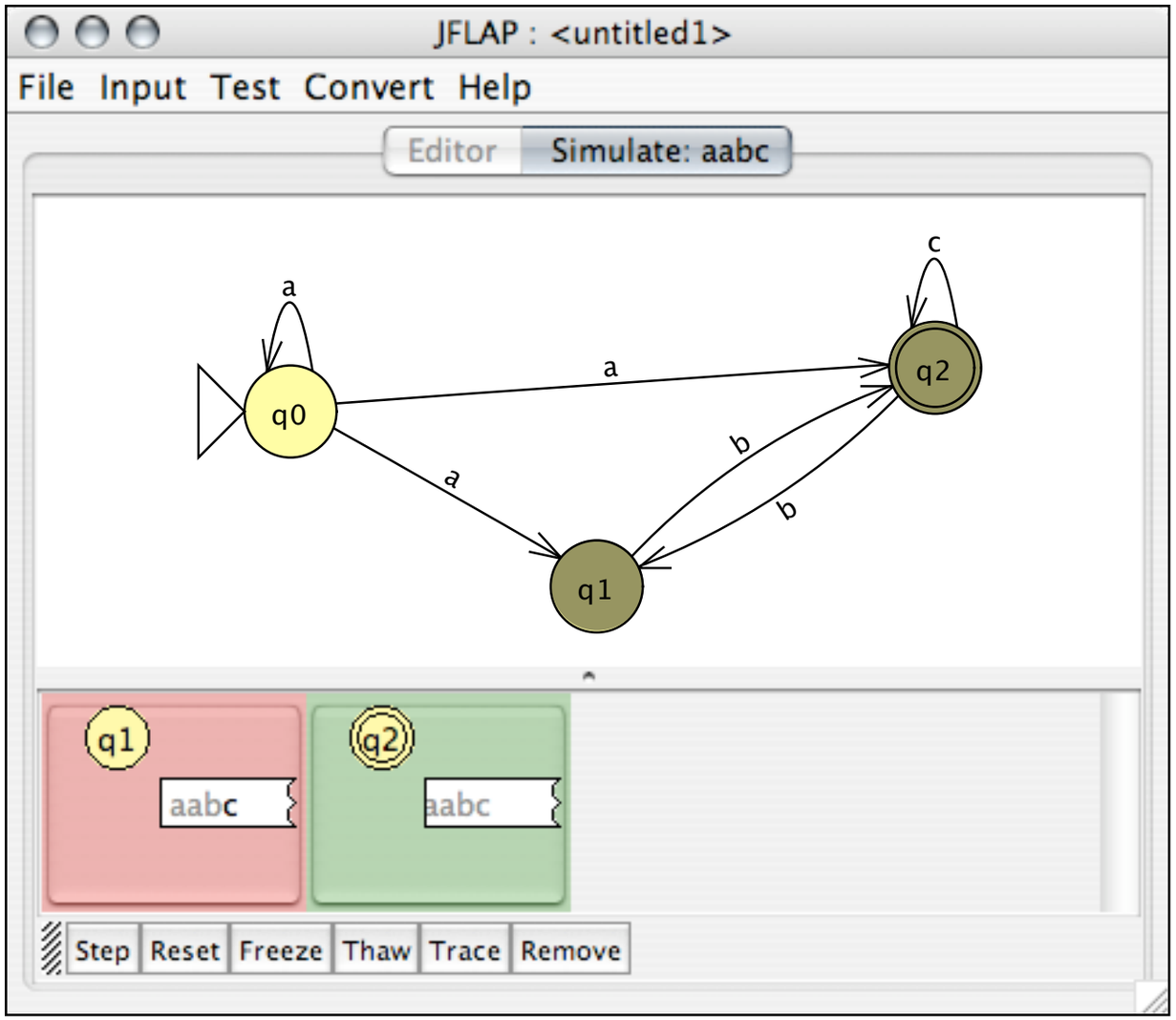
- The previous configurations on q_1 and q_2 are rejected, and are shown in red.
- The remaining uncolored configurations paths are not rejected, and are still open.

FA Edit & Simulation After Three Steps



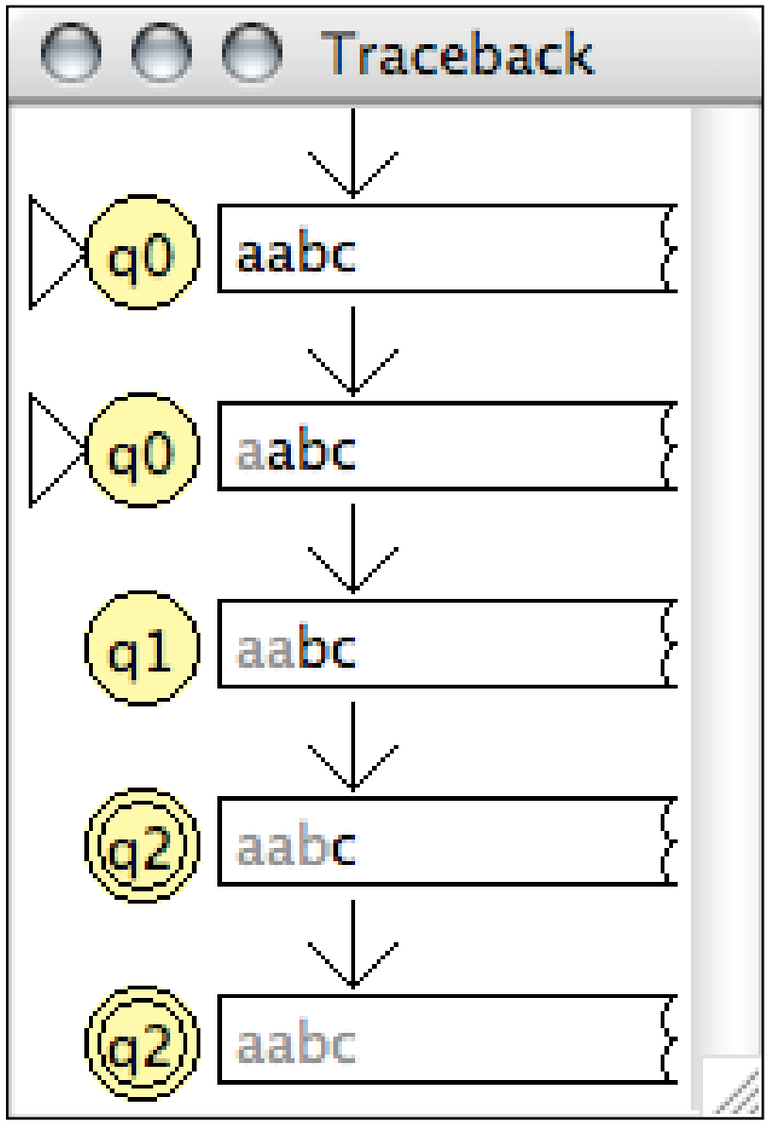
- Yet another step.

FA Edit & Simulation After Four Steps



- One of the final configurations has been accepted!

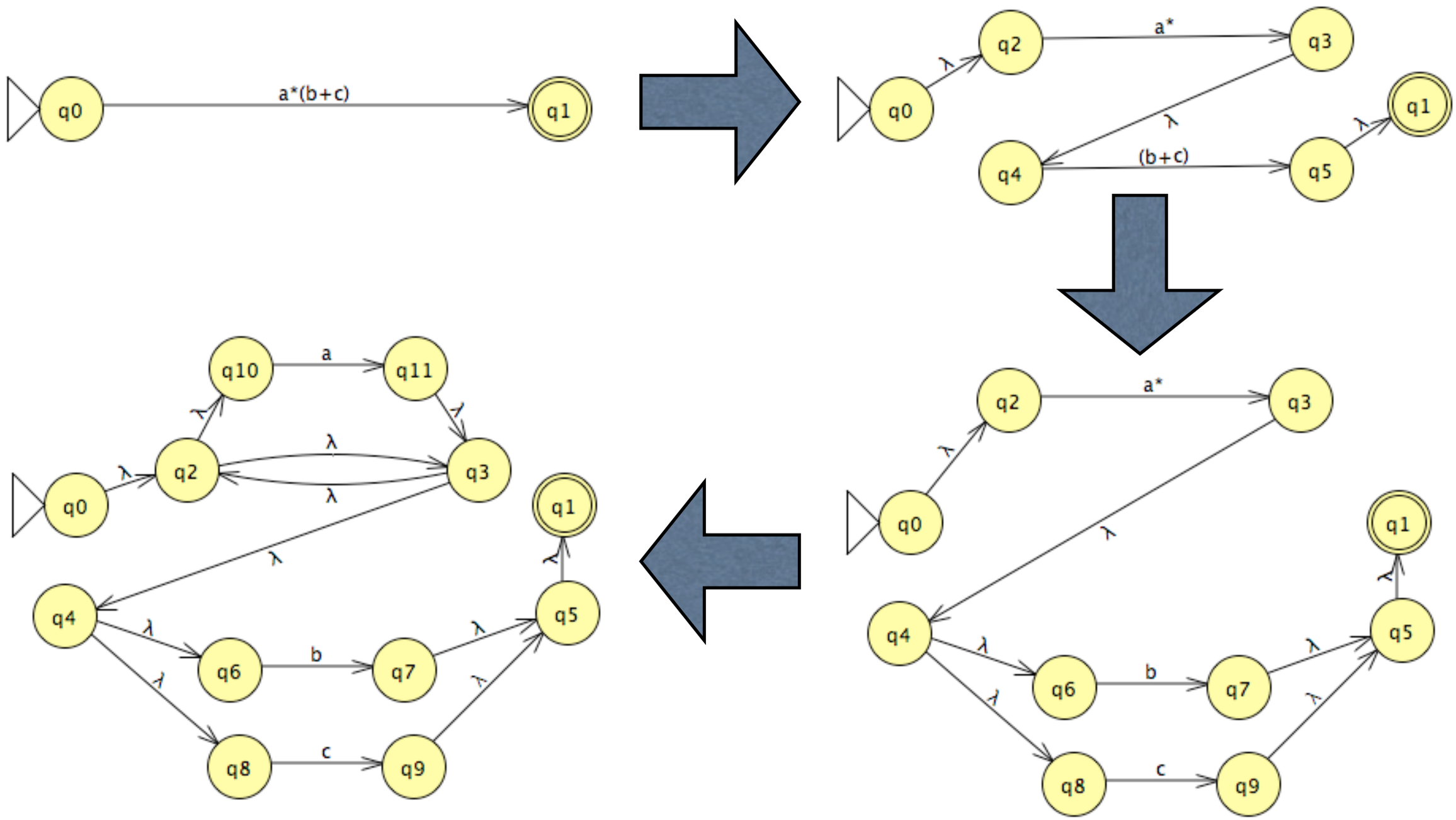
FA Edit & Simulation Traceback



- One can then see a traceback to see the succession of configurations that led to the accepting configuration.

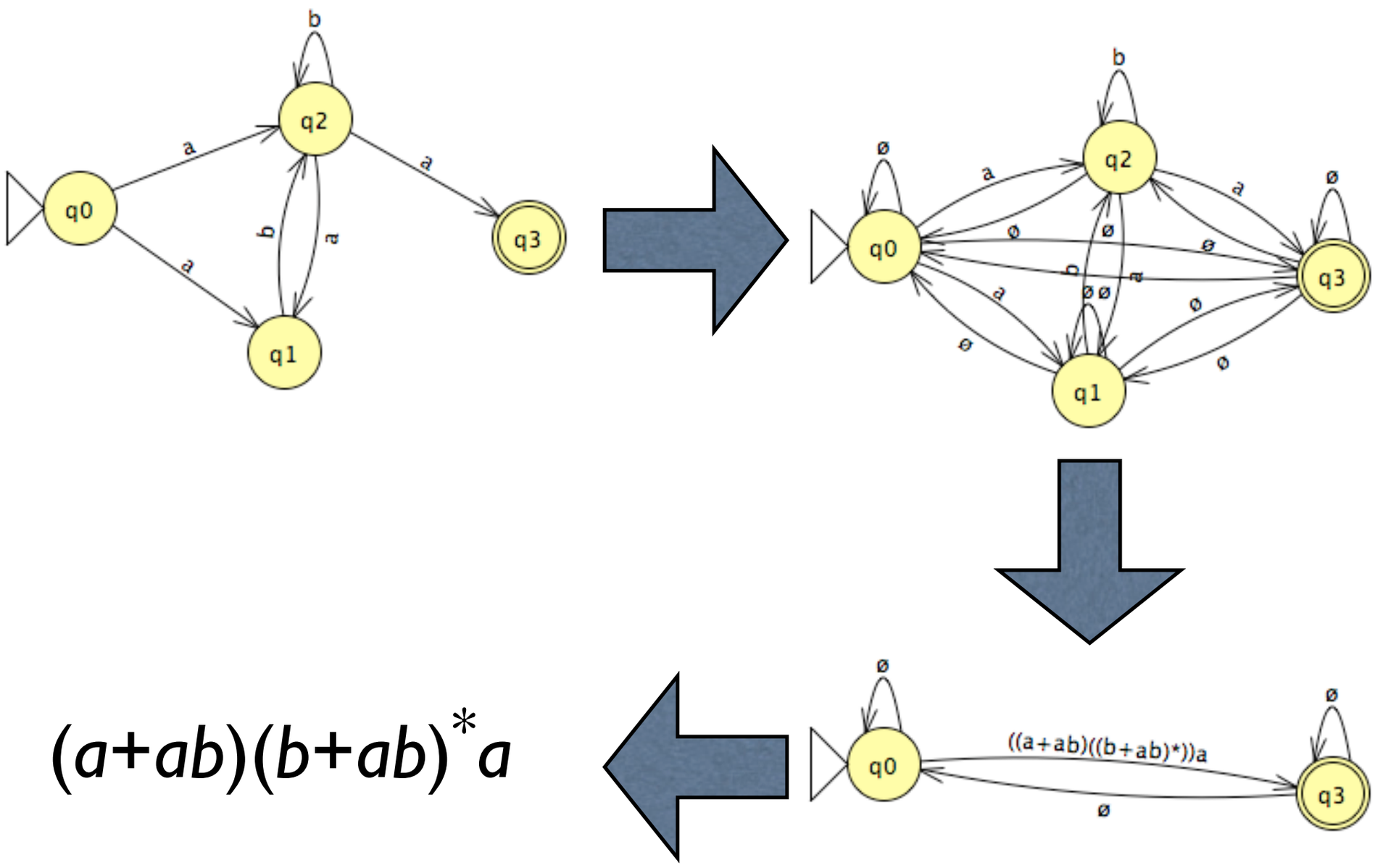
RE to FA

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.



FA to RE

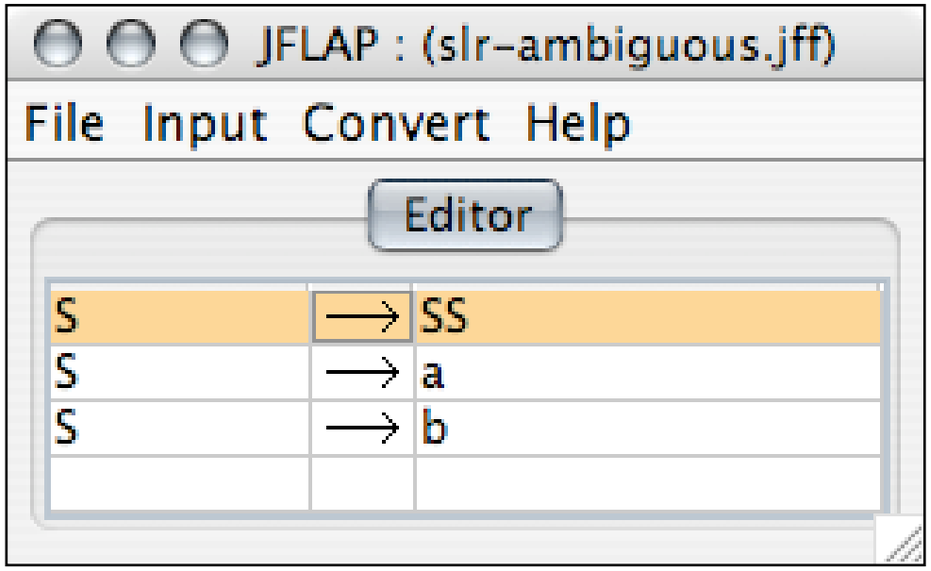
New algorithm transforms an FA to a GTG, and removes states until the GTG has only the initial and final states. At this point conversion becomes trivial.



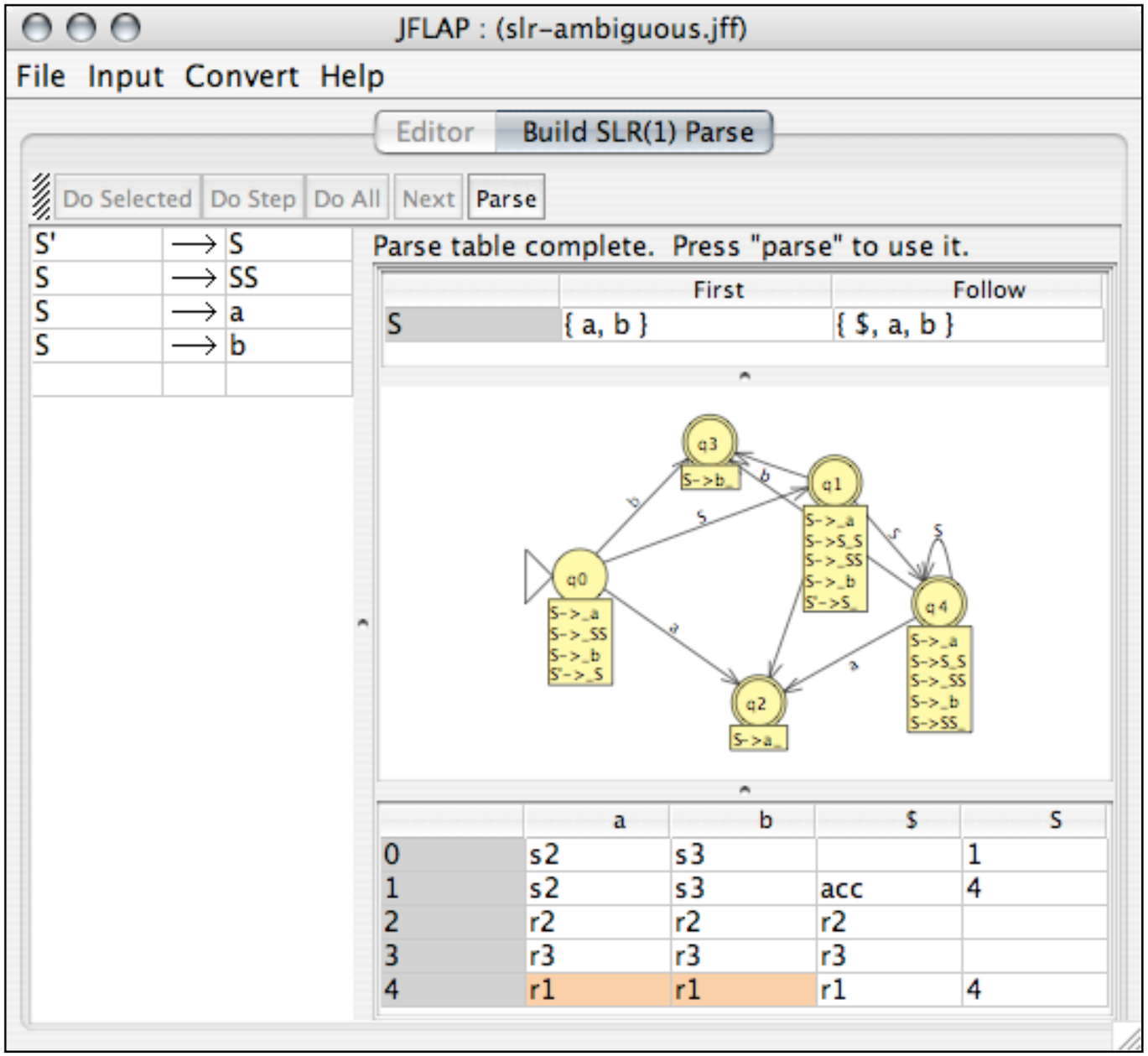
$(a+ab)(b+ab)^{*}a$

Ambiguous Grammar Parsing with SLR

- One can also parse strings with grammars in JFLAP using LL(1) or SLR(1) parsing.
- To the right is a trivial, obviously ambiguous grammar.
- We show how SLR(1) deals with ambiguity.



SLR(1) Parsing



In order, students:

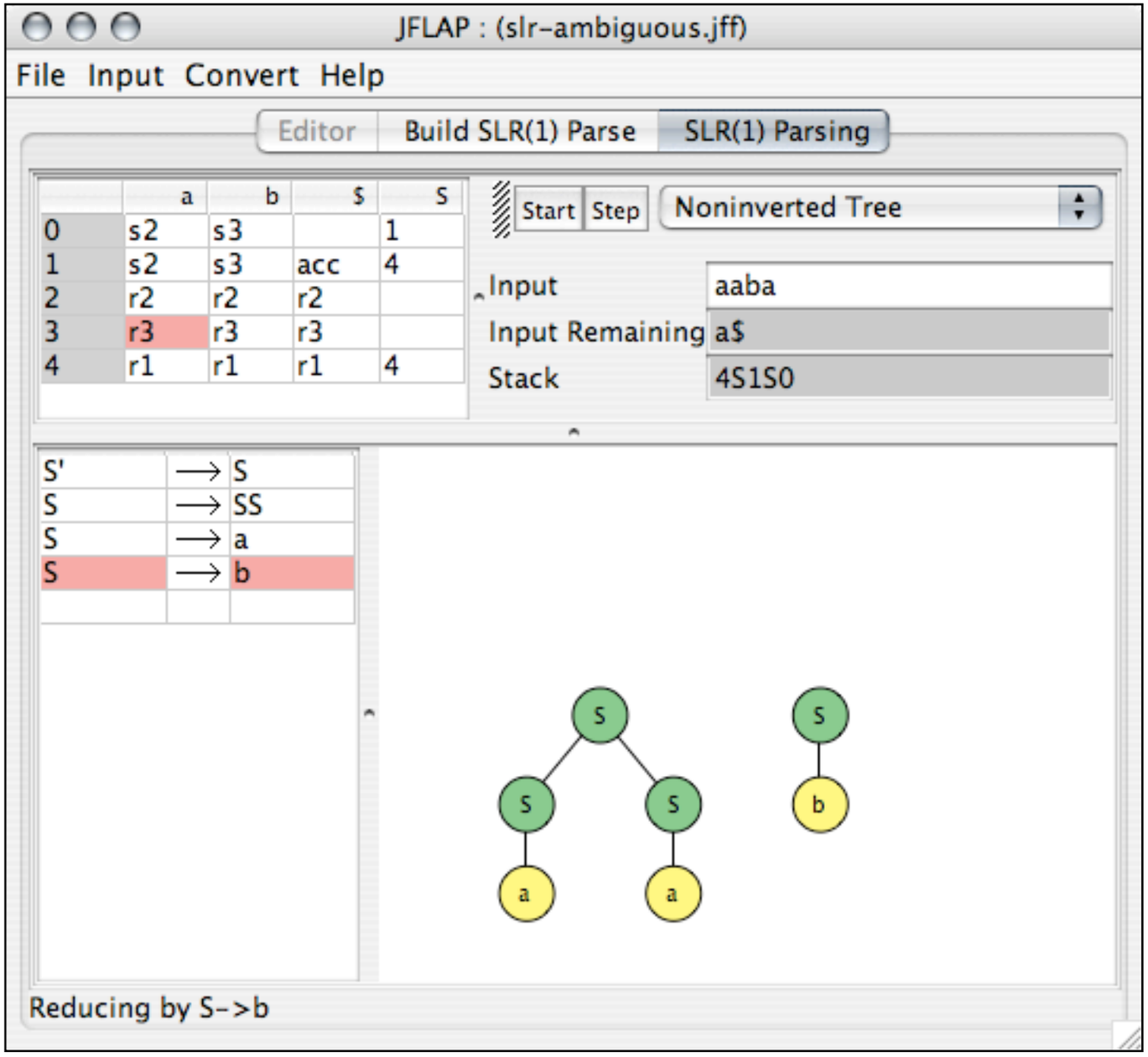
Define First and Follow Sets

Build the FA modeling stack.

Define the parse table.

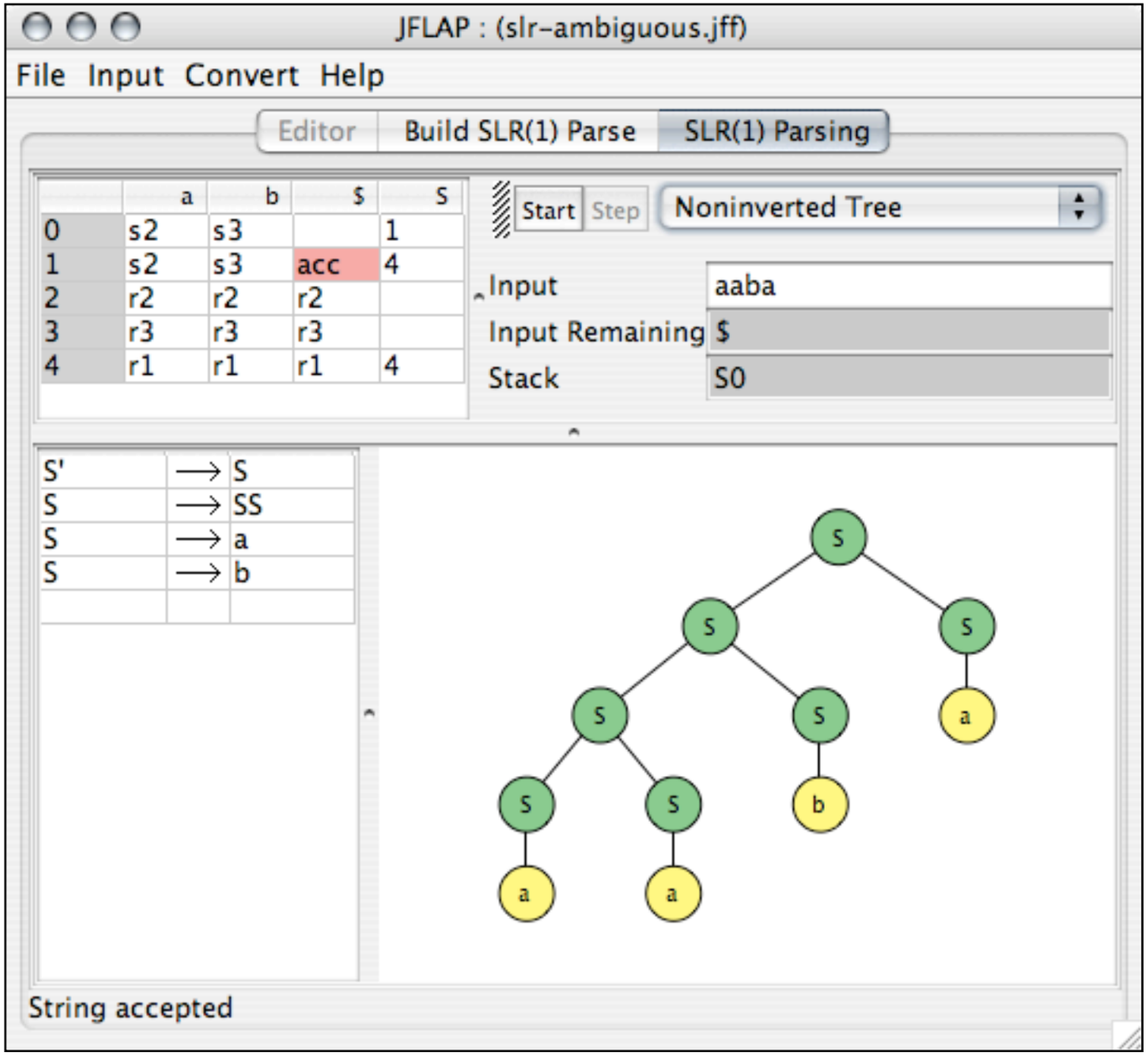
Orange entries indicate a conflict in the parse table. Current active value is displayed.

SLR(1) Parsing



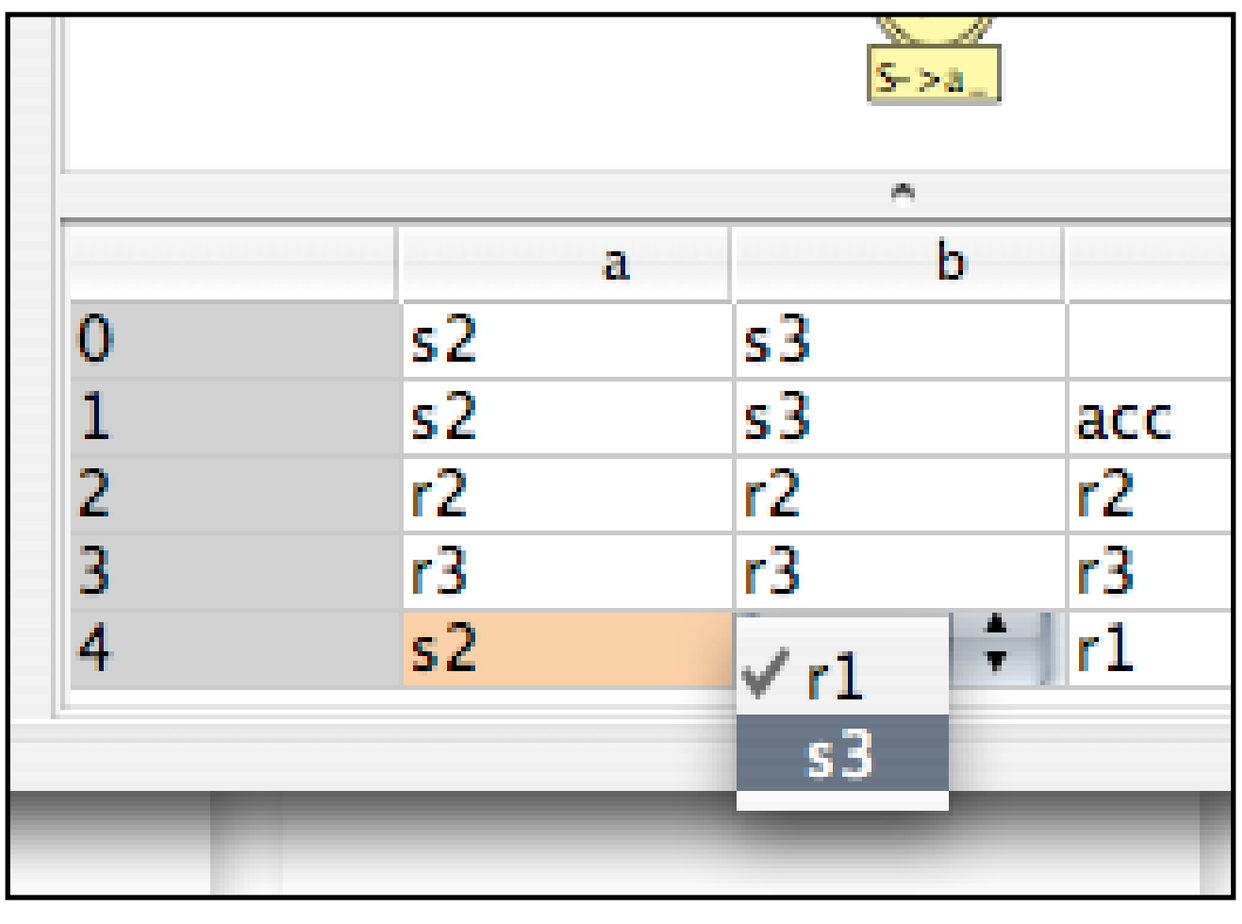
- Suppose we parse *aaba* with current conflicts both set to the default "reduce" entries.
- As students step, the parse table entry being used and grammar rule used (if a reduce) is highlighted.
- Notice also the input remaining and the stack.

SLR(1) Parsing



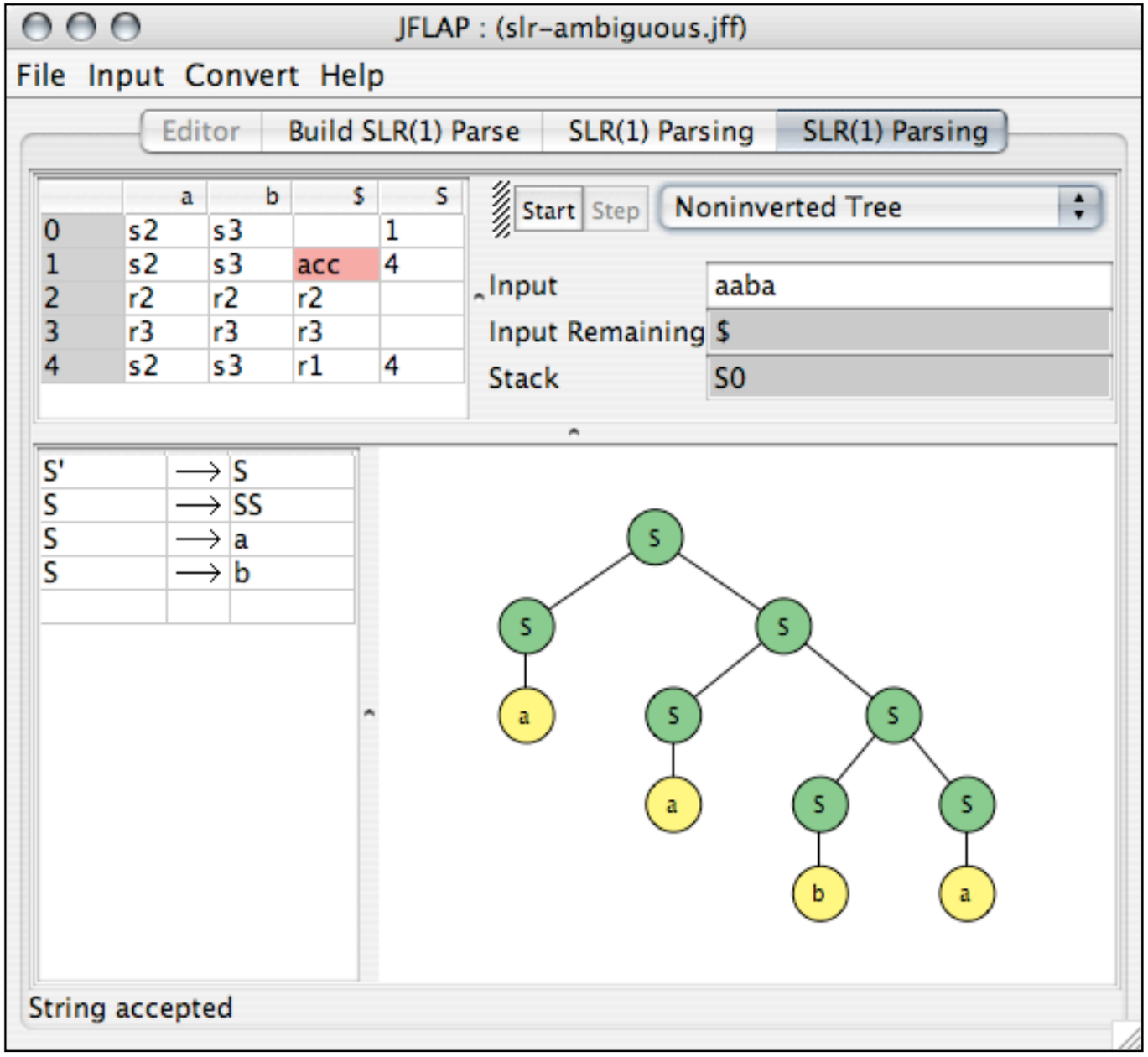
- Shown is the completed parse tree. Well done!

SLR(1) Parsing



- Recall the conflicts.
- When we click on the orange entry, we can choose a different entry to resolve the conflict.
- In this case we change the reduce operations to shift operations.

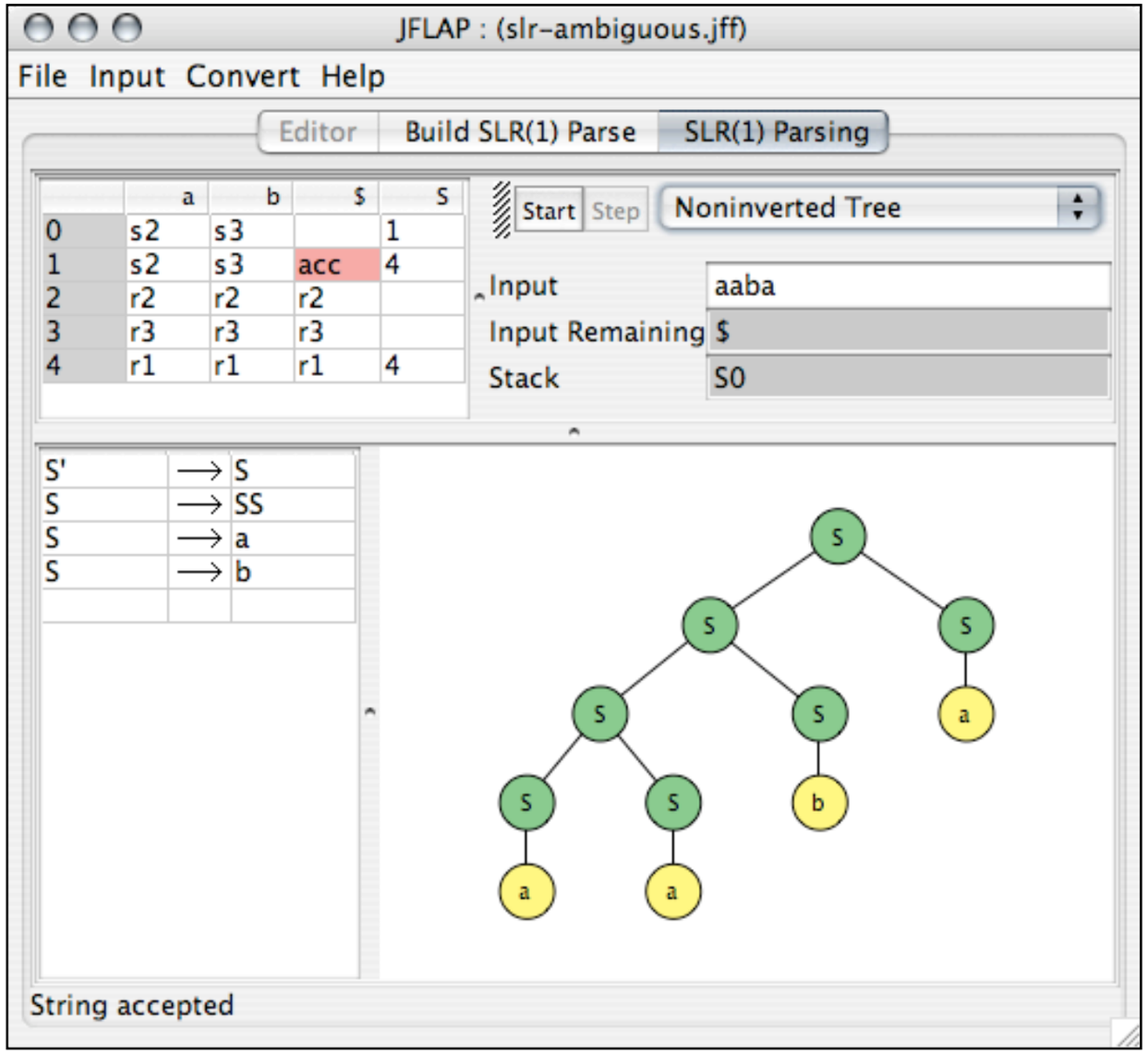
SLR(1) Parsing



- Notice, this change results in a very different parse tree.

SLR(1) Parsing

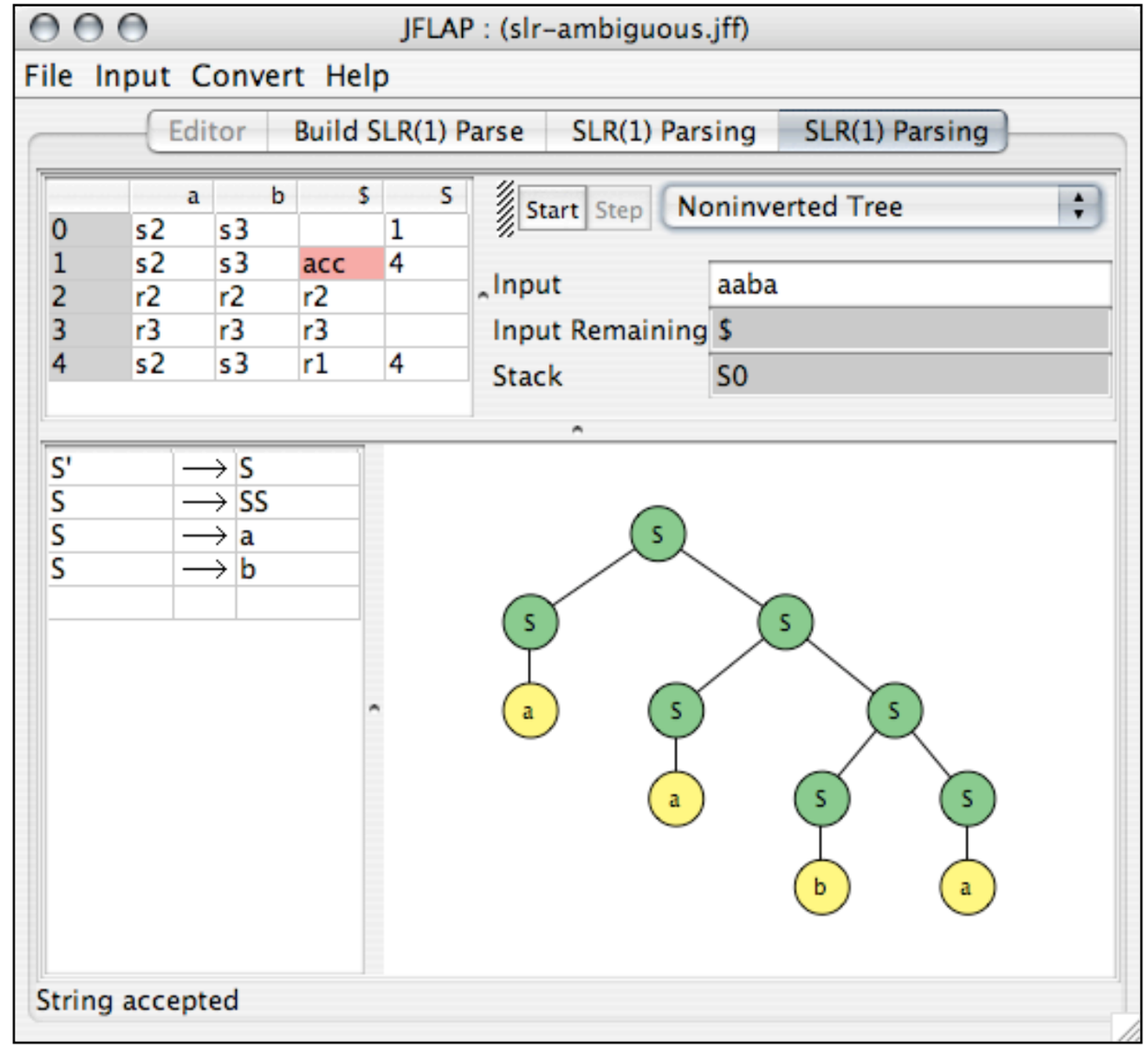
With Reduce Entries



	a	b	$	S
0	s2	s3		1
1	s2	s3	acc	4
2	r2	r2	r2	
3	r3	r3	r3	
4	r1	r1	r1	4

S'	→	S
S	→	SS
S	→	a
S	→	b

With Shift Entries

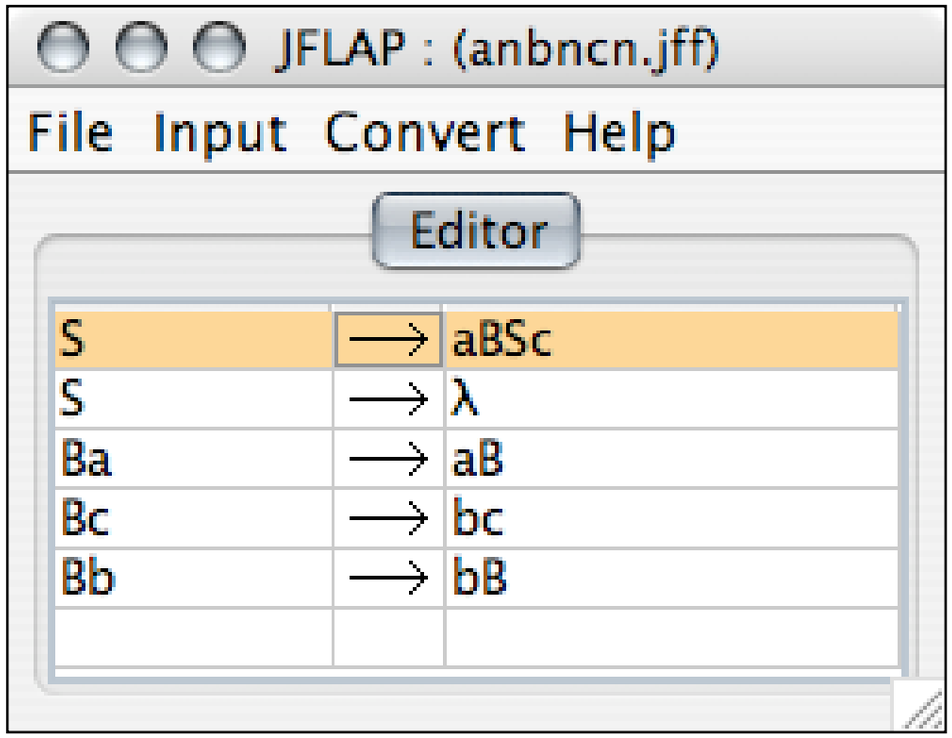


	a	b	$	S
0	s2	s3		1
1	s2	s3	acc	4
2	r2	r2	r2	
3	r3	r3	r3	
4	s2	s3	r1	4

S'	→	S
S	→	SS
S	→	a
S	→	b

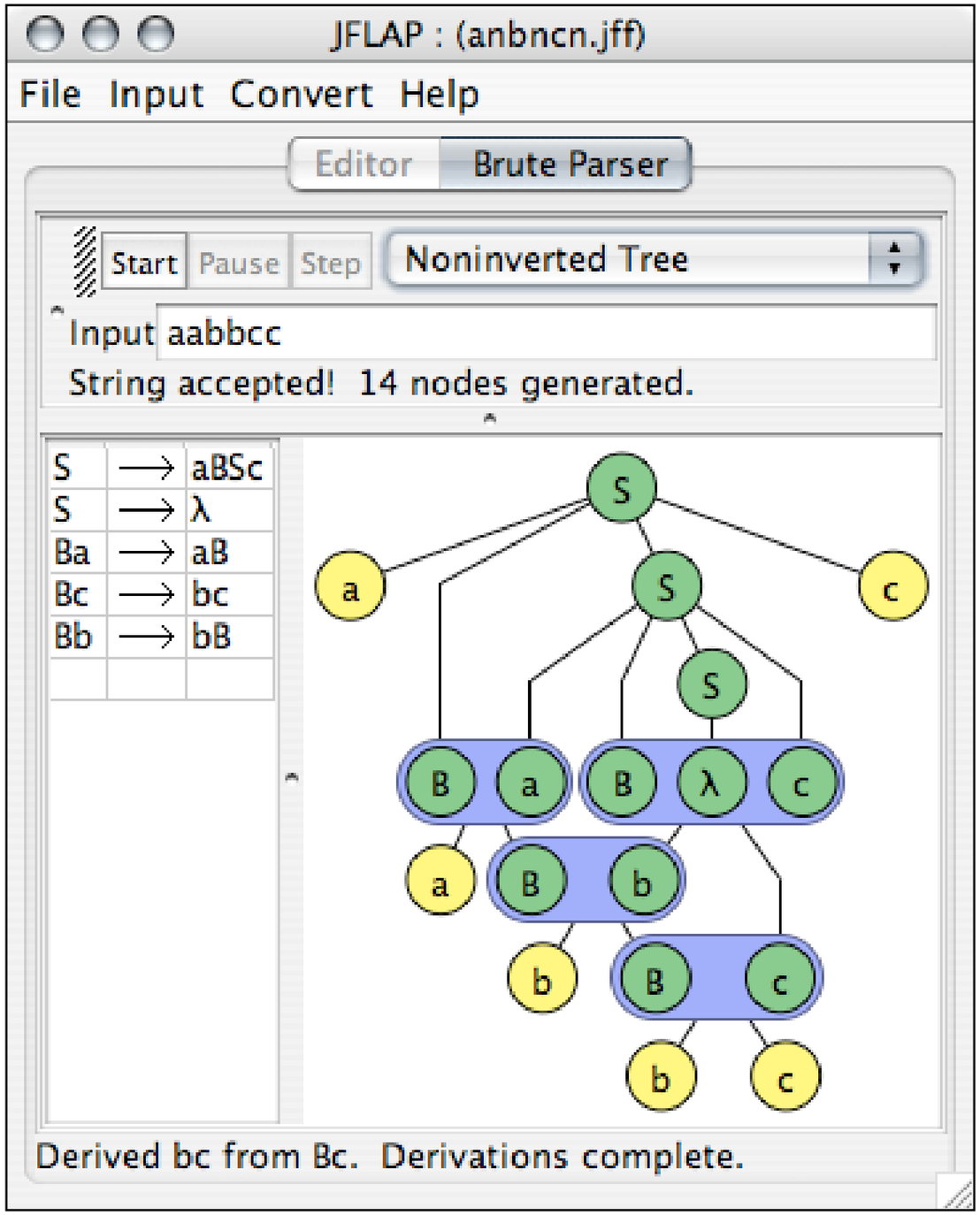
Brute Force Parsing

- 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!



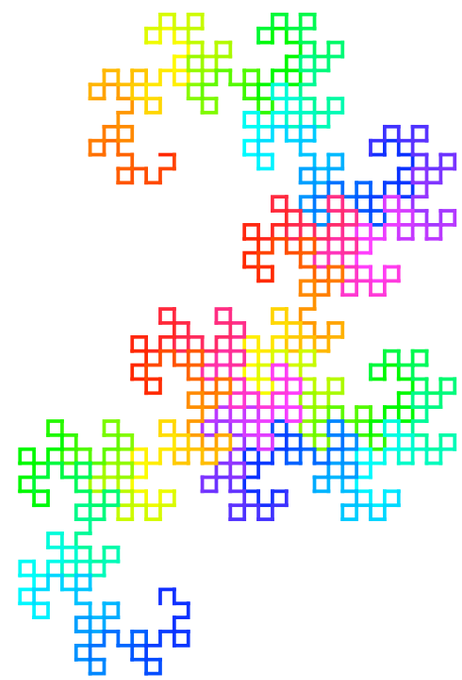
Brute Force Parsing

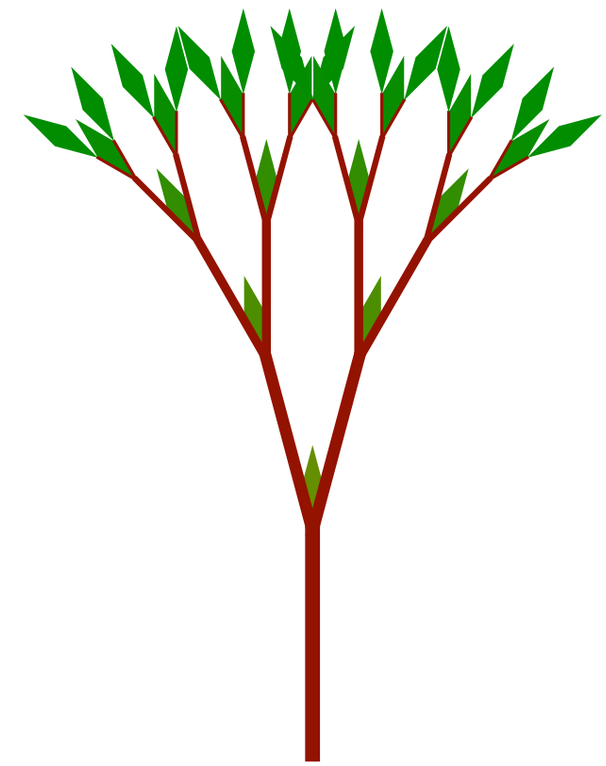
- We parse the string *aabbcc* with the brute force parser.
- Notice how in this case multiple nonterminal nodes are grouped together to form a single node.
- This accomplishes the unrestricted grammar possibly replacing multiple symbols at once.

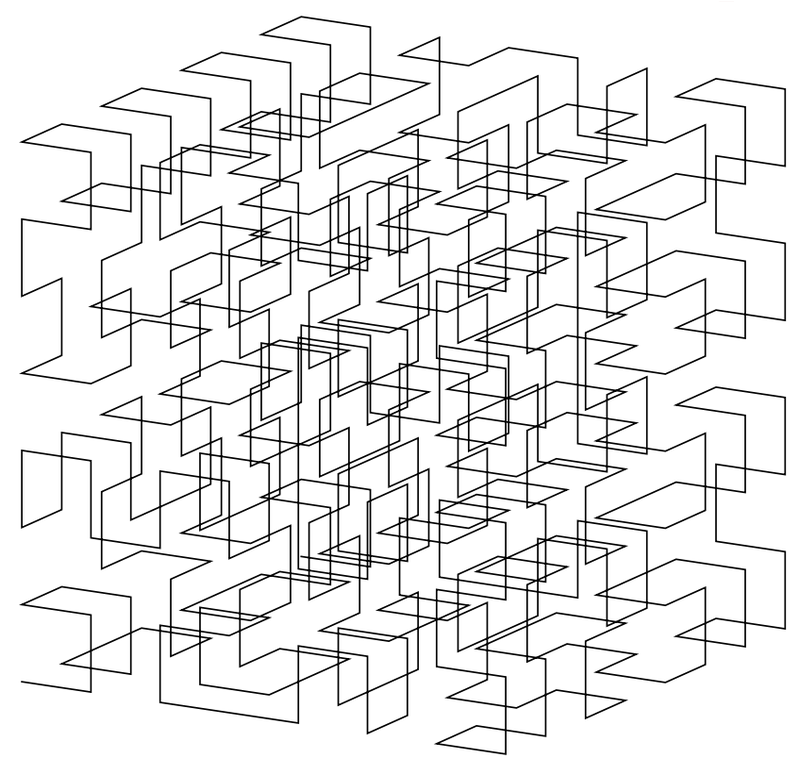


L-Systems

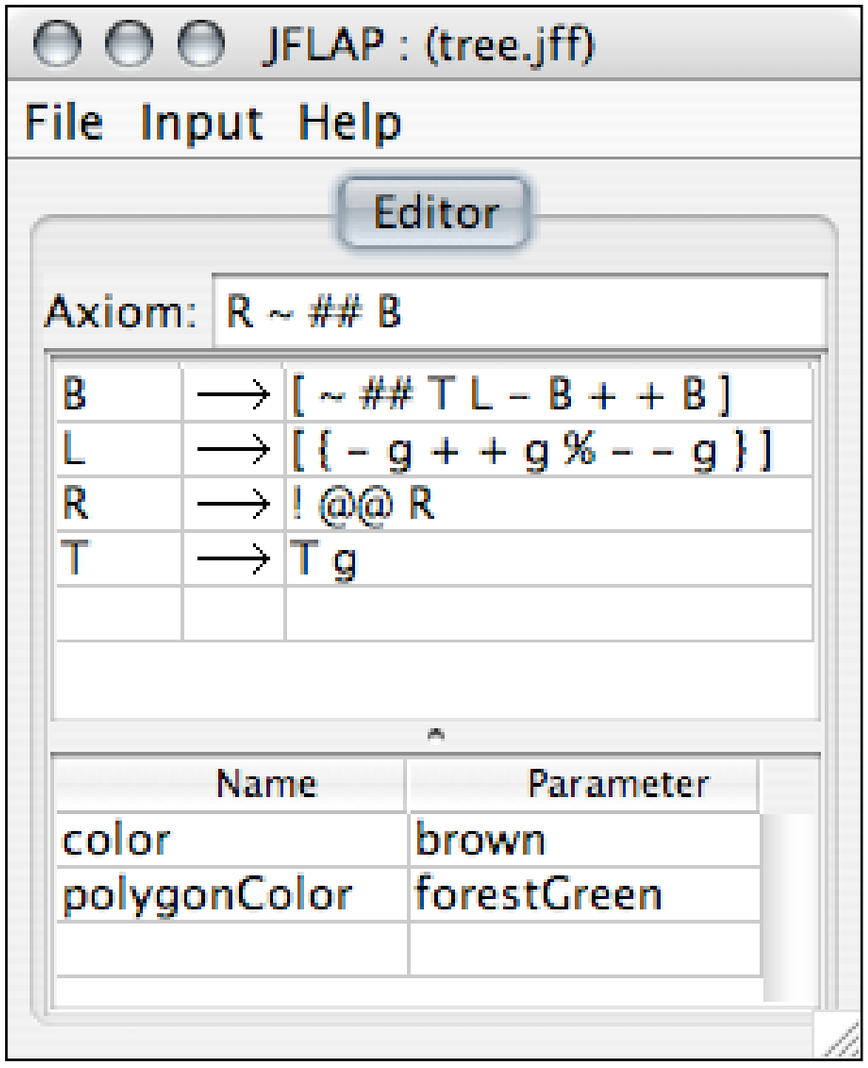
- L-Systems may be used to model biological systems and create fractals.
- Similar to Chomsky grammars, except *all* variables are replaced in each derivation step, not just one!
- Commonly, strings from successive derivations are interpreted as strings of render commands and are displayed graphically.



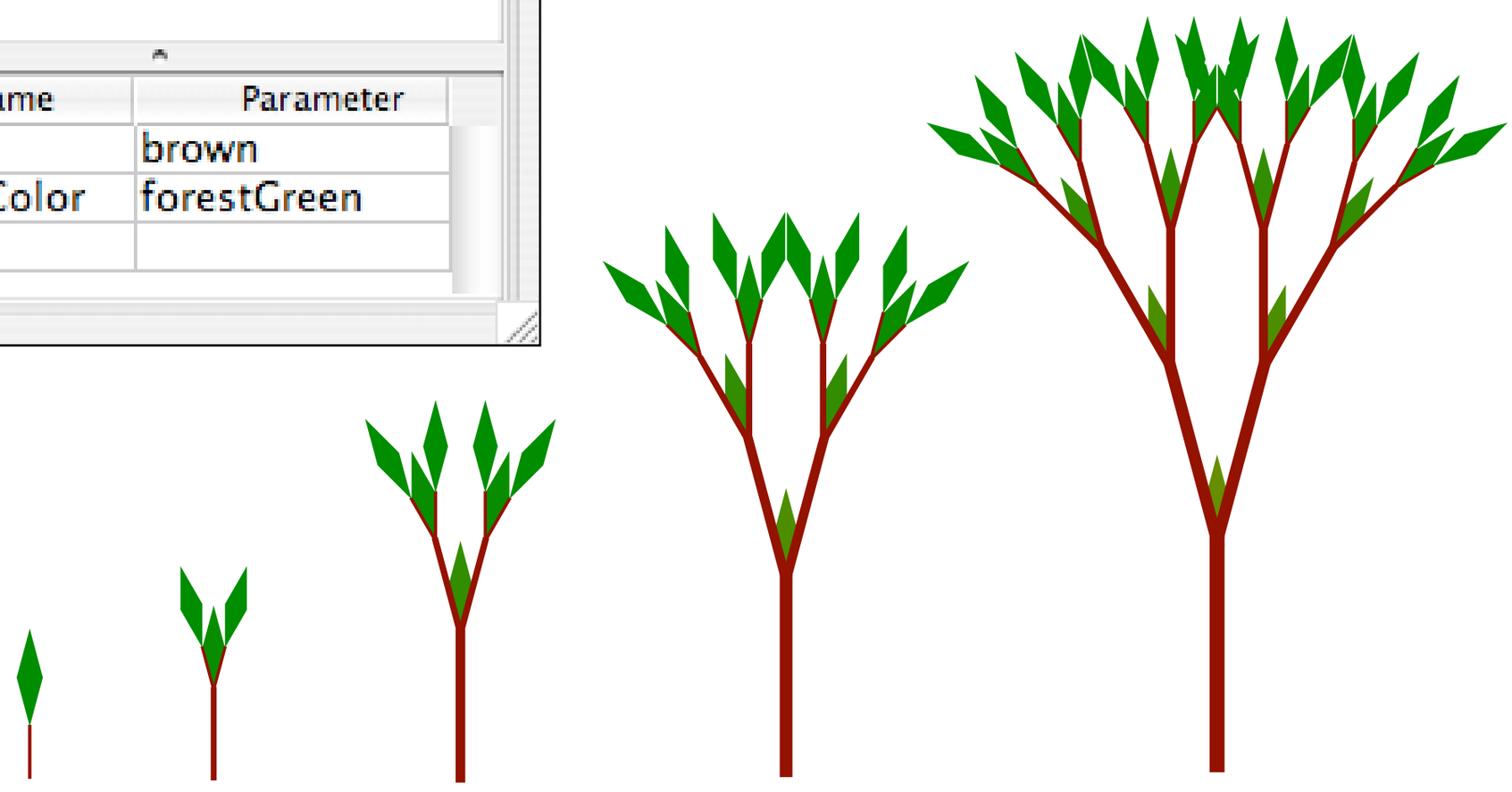




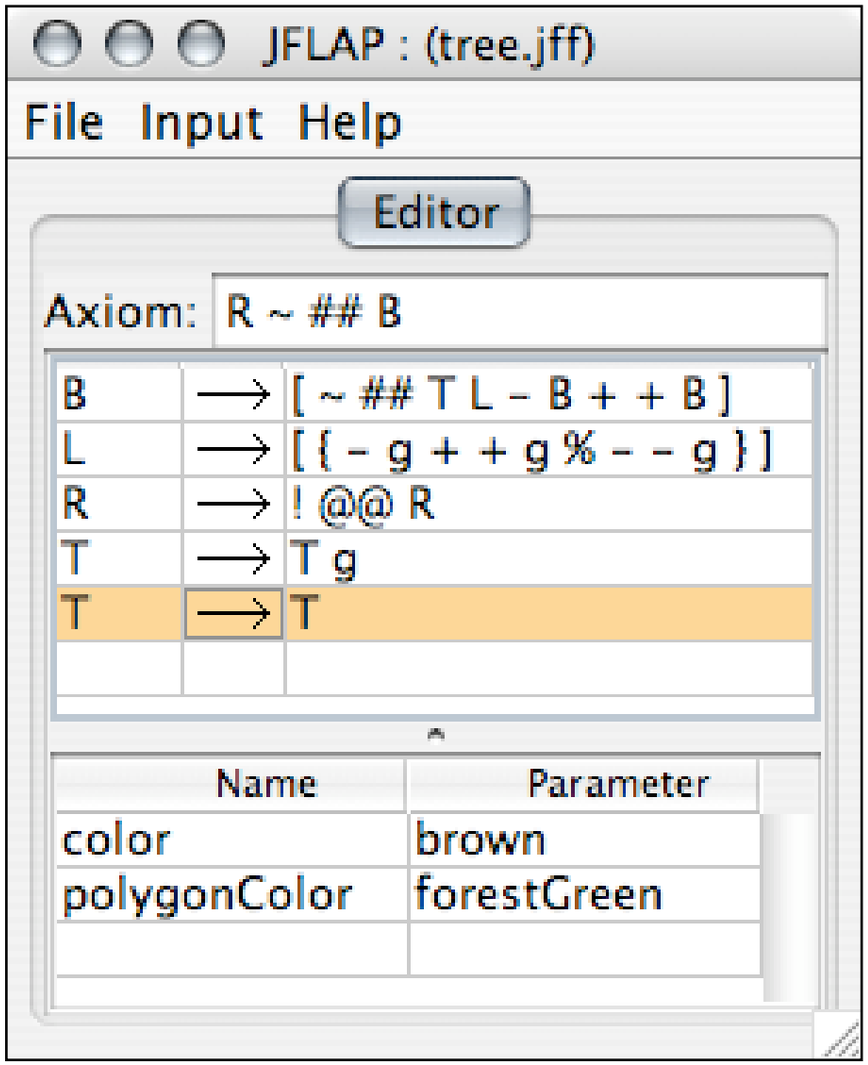
L-Systems



- This L-System renders as a tree that grows larger with each successive derivation step.



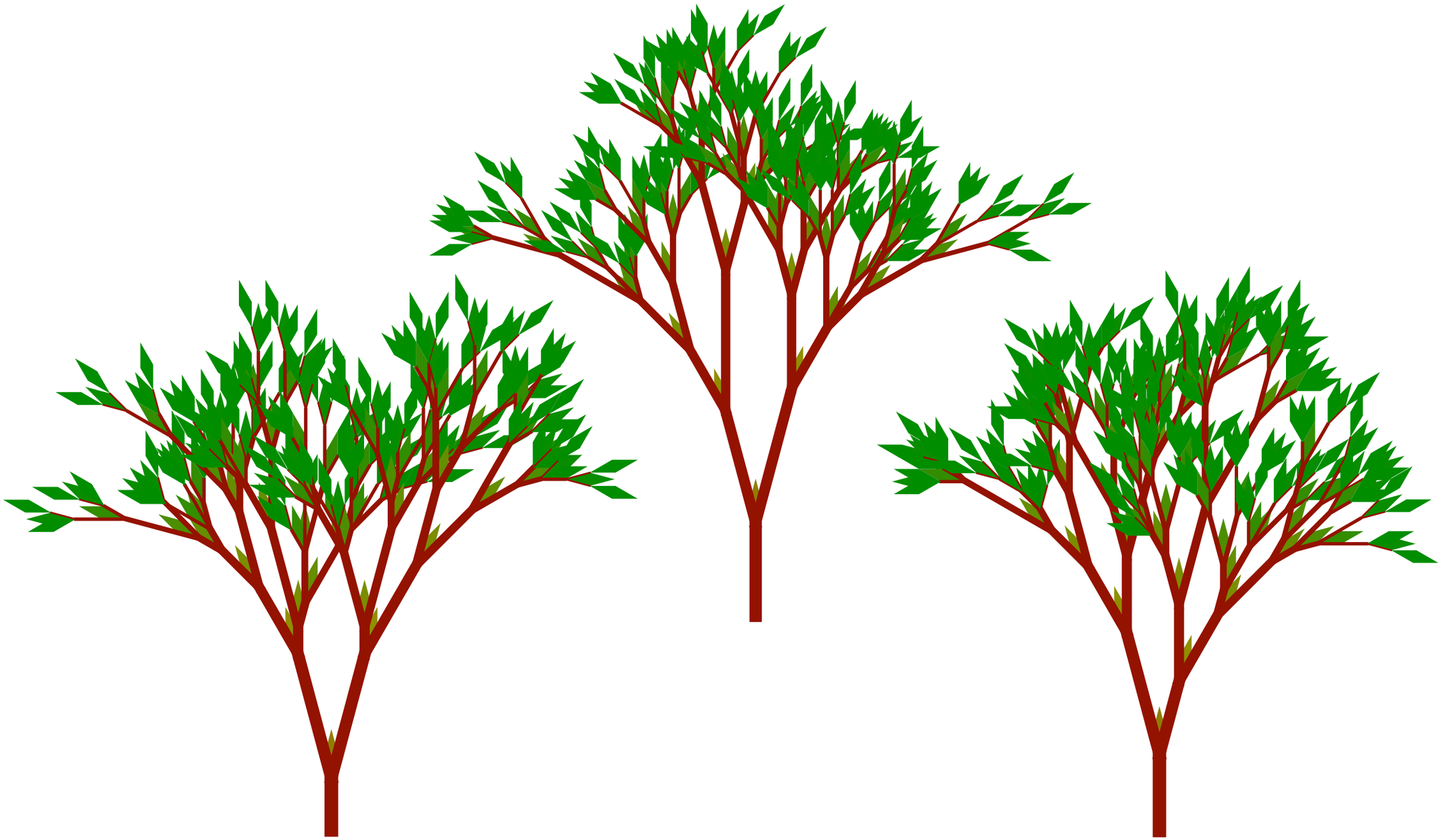
L-Systems



- L-systems may also be stochastic.
- The $T \rightarrow Tg$ rule adds g to the derivation, which draws a line segment.
- We add another rewriting rule for T, $T \rightarrow T$.
- With two rewriting rules for T, the rule chosen is random, leading to uneven growth!

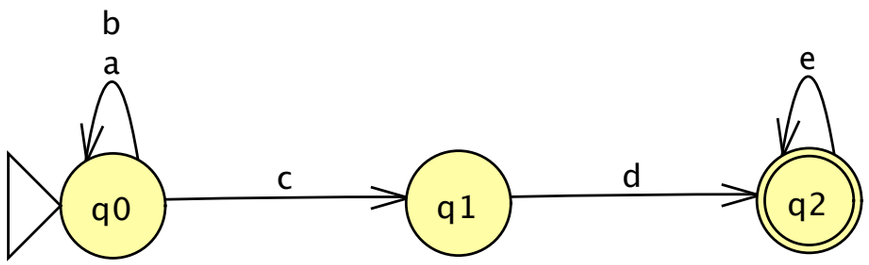
L-Systems

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

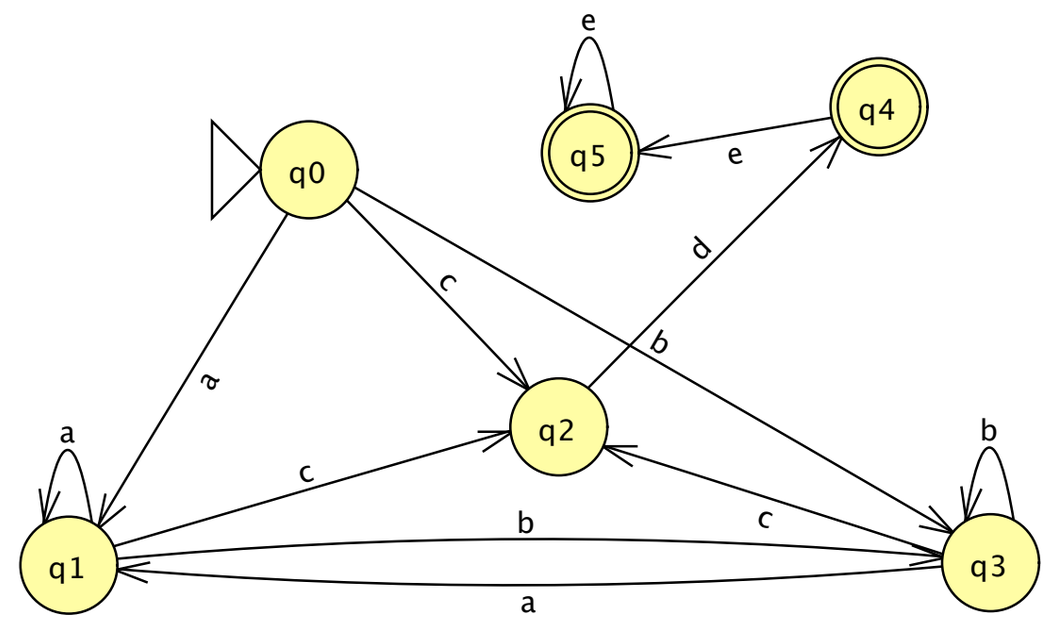


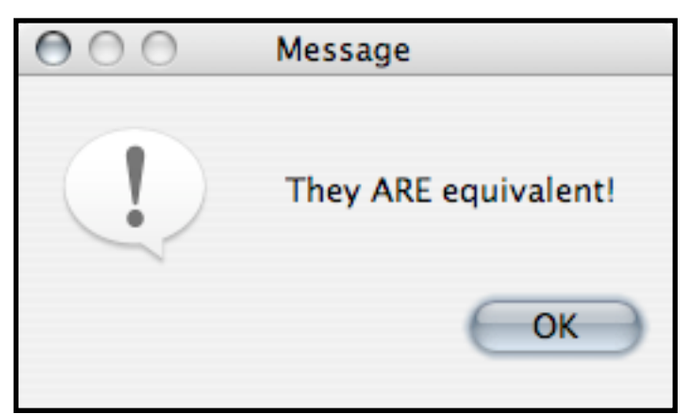
Compare for Equivalence

Determines if two FA recognize the same language.



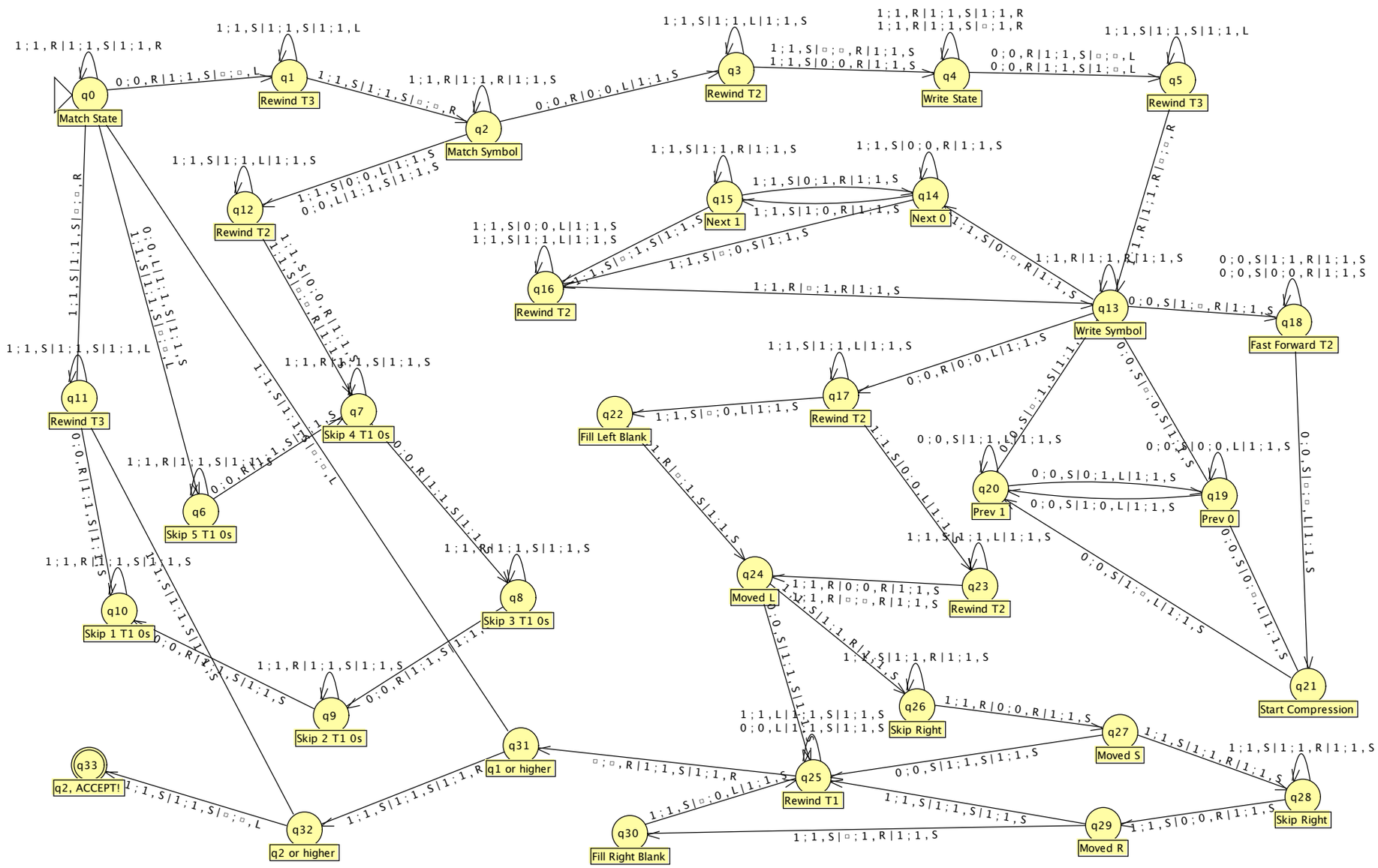
≡





Multiple Tape Turing Machines

For example, with 3 tapes, you can relatively easily define a Universal Turing Machine.



JFLAP's Use Around the World

- JFLAP web page has over 42,000 hits since 1996
- Google Search
  - JFLAP appears on over 2300 web pages
  - JFLAP appears on automata theory class webpages at over 40 US universities
  - Note: search only shows public web pages
  - Note: appears to be many foreign sites
- JFLAP has been downloaded over 8000 times since Jan. 2003
- JFLAP appears in use (web pages or downloads) in over 40 countries

Future Work

- Filling in the missing pieces for automata theory course
- JFLAP User manual out in December 2004 (Rodger, Finley)
- JFLAP Automata Theory textbook out in Fall 2005 (Rodger, Linz)

JFLAP is FREE!