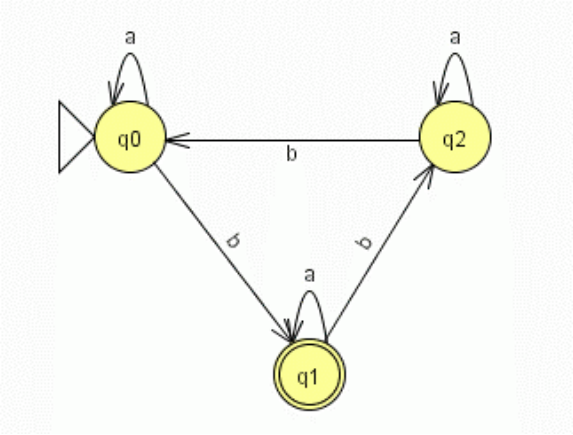
A Hands-on Approach to Formal Languages and Automata with JFLAP

Susan Rodger, Duke University

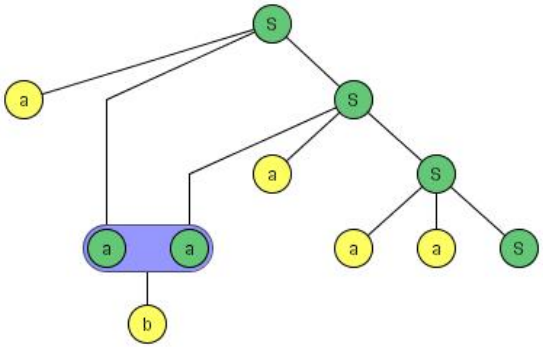
Thomas Finley, Cornell University

Peter Linz, University of California, Davis



SIGCSE 2006

March 4, 2006



Who We Are…….
Susan Rodger

- Duke University
  - Associate Professor of the Practice
  - Teaching automata theory for 16 years
  - Developing JFLAP for 15 years!
- Co-author of “JFLAP: An Interactive Formal Languages and Automata Package”, Jones and Bartlett, 2006.

Who We Are ……
Thomas Finley

- Cornell University
  - Graduate Student and Research Assistant
- One of the developers of JFLAP 4.0
- Duke University
  - BS Computer Science/Mathematics 2002
- Co-author of “JFLAP: An Interactive Formal Languages and Automata Package”, Jones And Bartlett, 2006.

Who We Are ……..
Peter Linz

- University of California, Davis
  - Professor Emeritus
  - Teaching automata theory a long time…
- Written five textbooks including
  - An Introduction to Formal Languages and Automata Theory, 4th Ed, Jones and Bartlett, 2006

Outline – What we will do!

- Overview of JFLAP
- Getting Started with JFLAP – Regular Languages
- JFLAP and Context-free Languages
- JFLAP and Turing Machines
- JFLAP and Parsing
- Integrating JFLAP into a course
- JFLAP and L-systems
- History of JFLAP and Wrap-up

- Exercises with JFLAP throughout

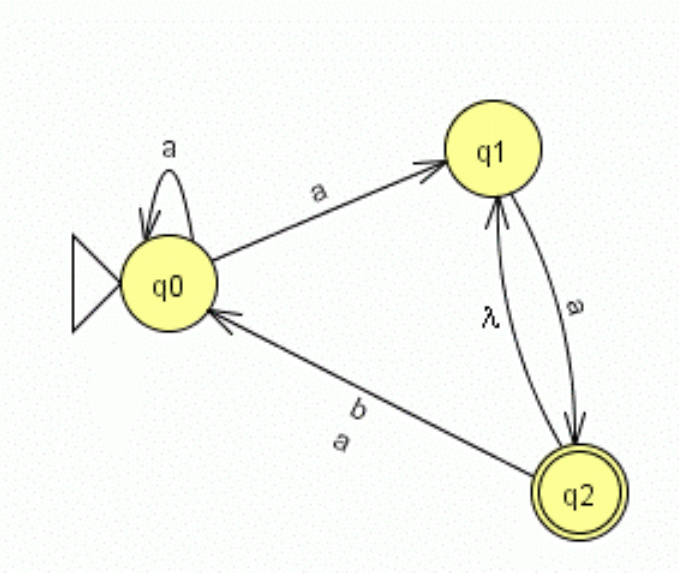
Why should people use JFLAP?

Overview of JFLAP

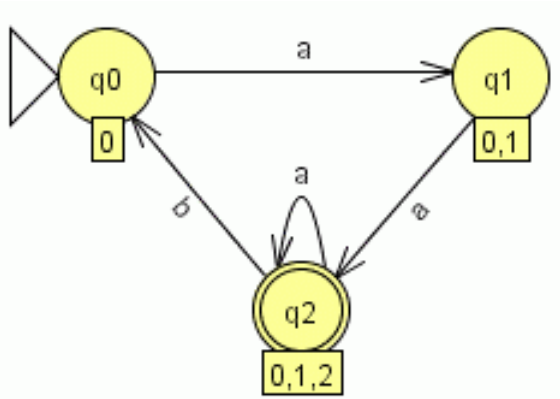
- **J**ava **F**ormal **L**anguages and **A**utomata **P**ackage
- Instructional tool to learn concepts of Formal Languages and Automata Theory
- Topics:
  - Regular Languages
  - Context-Free Languages
  - Recursively Enumerable Languages
  - Lsystems

JFLAP – Regular Languages

- Create
  - DFA and NFA
  - regular grammar
  - regular expression

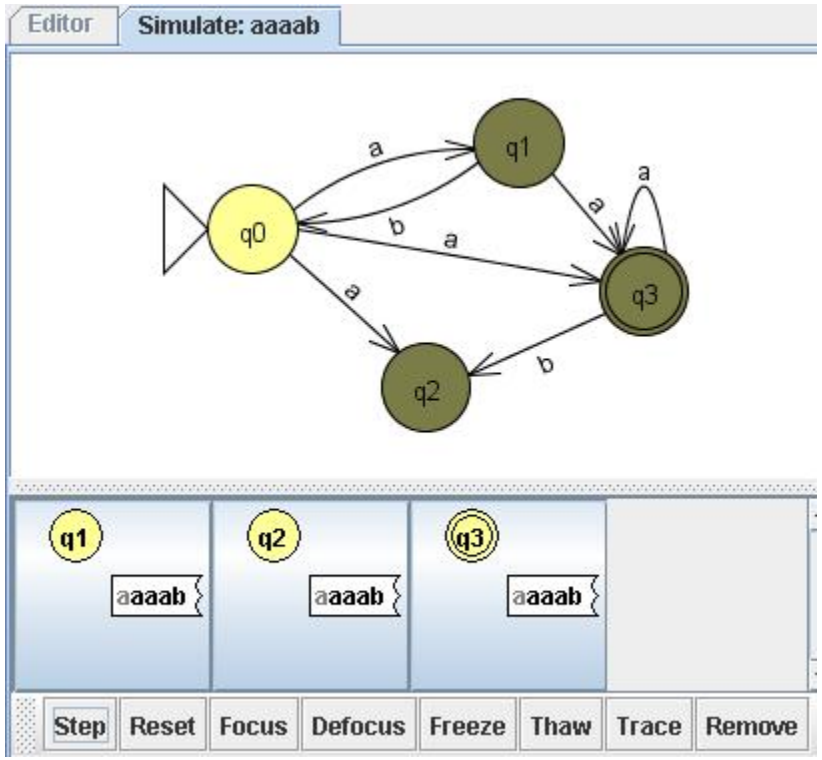


- Conversions
  - NFA to DFA to minimal DFA
  - NFA ←→ regular expression
  - NFA ←→ regular grammar



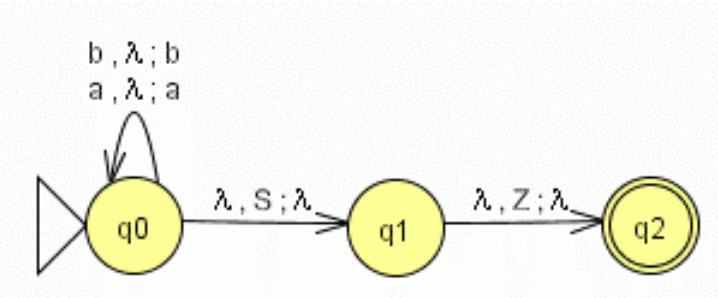
JFLAP – Regular languages (more)

- Simulate DFA and NFA
  - Step with Closure or Step by State
  - Fast Run
  - Multiple Run
- Combine two DFA
- Compare Equivalence
- Brute Force Parser

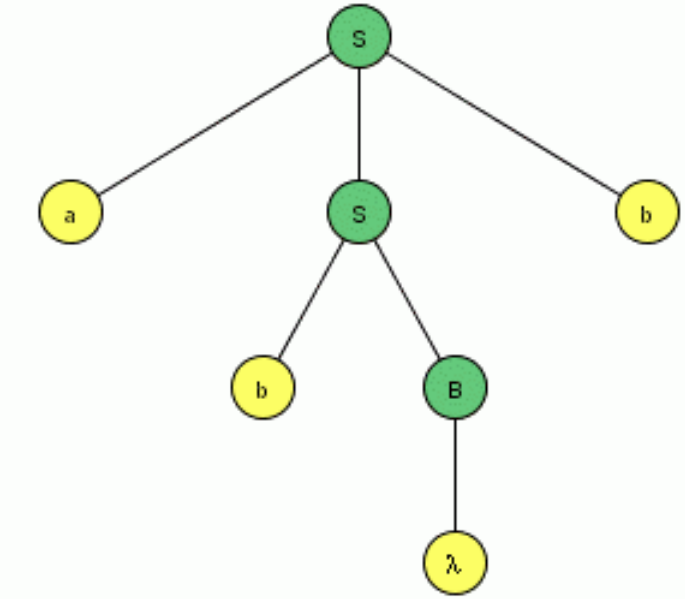


JFLAP – Context-free Languages

- Create
  - Nondeterministic PDA
  - Context-free grammar

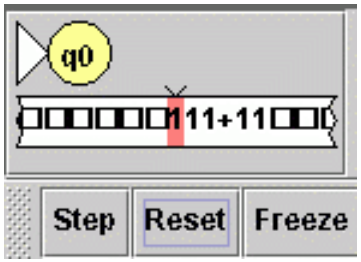


- Transform
  - PDA → CFG
  - CFG → PDA (LL & SLR parser)
  - CFG → CNF
  - CFG → Parse table (LL and SLR)
  - CFG → Brute Force Parser

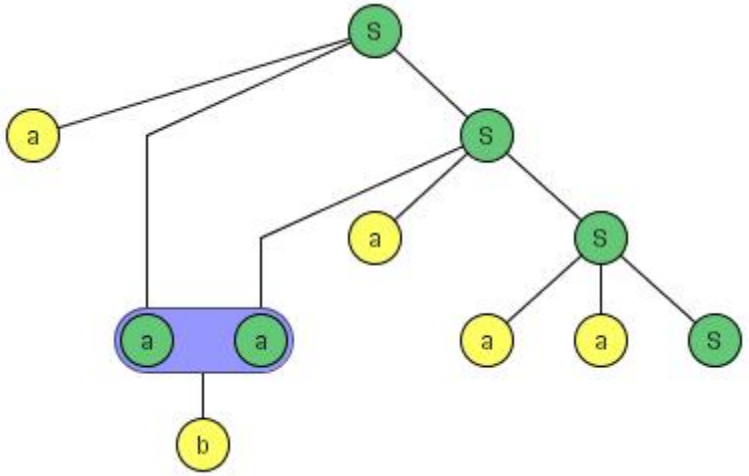


JFLAP – Recursively Enumerable Languages

- Create
  - Turing Machine (1-Tape)
  - Turing Machine (multi-tape)
  - Building Blocks
  - Unrestricted grammar

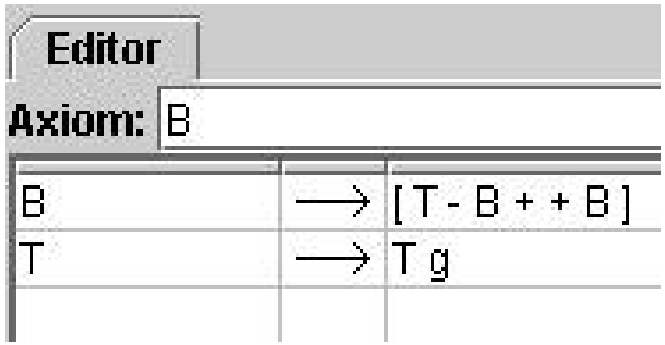


- Parsing
  - Unrestricted grammar with brute force parser



JFLAP - Lsystems

- Create an L-system



- Render the L-system