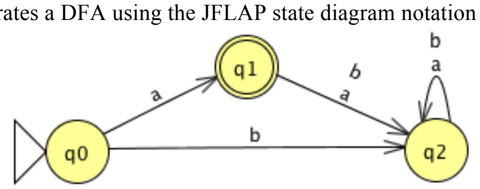
Deterministic Finite Automata (DFA)$_{JP}$

- Prerequisite knowledge:
  Automata
  Regular Languages
  Set Theory
  JFLAP Tutorial

Description of Deterministic Finite Automata

A *Deterministic Finite Automaton* (DFA) is a finite state machine that accepts or rejects finite strings of symbols and produces the same unique computation for each unique input string. For any given finite input string, the DFA will halt and either accept or reject the string. A DFA, M, is said to *recognize* a language, L(M), which is the set of all strings that M accepts.

The following figure illustrates a DFA using the JFLAP state diagram notation (see `DFA_a.jff`).



This DFA recognizes the regular language over the alphabet {a, b} consisting of only the string "a". That is, it accepts the string "a" and rejects all other strings.

Formally, a DFA is described by a 5-tuple (Q, Σ, δ, q0, F) in which
    Q is a finite set of states,
    Σ is a finite set of input symbols also known as the alphabet,
    δ is a state transition function ($\delta : Q \times \Sigma \rightarrow Q$),
    q0 is the start state (q0 ∈ Q), and
    F is a set of accept states (F ⊆ Q).

For example, the DFA whose state diagram is shown above is represented by the 5-tuple
    ({q0, q1, q2}, {a, b}, δ as given by state diagram above, q0, {q1})

The transition function given by state diagram above is also described by the following table:

	a	**b**
q0	q1	q2
q1	q2	q2
q2	q2	q2

In the transition function of a DFA, every state has exactly one transition associated with each symbol of the alphabet. Because the DFA determines the unique next state for each next input symbol, this is a deterministic finite automaton.

Martha Kosa 7/22/2014 9:39 PM
Comment [1]: Perhaps you could demonstrate one step-by-step trace to see the configurations in one simple accepting computation and one simple rejecting computation.

Example (see: `DFA_cxc.jff`)

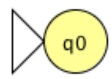
Consider the regular language L over the alphabet { a, b, c } comprised of all strings that begin and end with c. Let's construct a DFA M to recognize that language.

Some strings in the language include: c, cc, ccc, cac, cabc, cabcbabc

Some strings not in the language include: λ, a, ac, cb, ccca

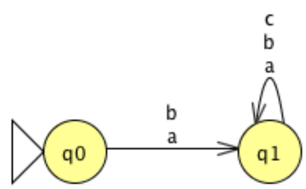
1. Since the empty string is not in the language, we know that the initial state must not be an accept state.

> **Create a new initial state.**



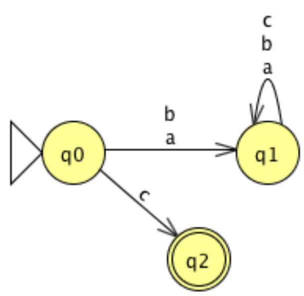
2. Since all strings that begin with a or b are not in the language, we can create a "trap" state to catch all such strings.

> **Create a new state, q1, as the destination of a or b from q0 and which traps all subsequent substrings.**



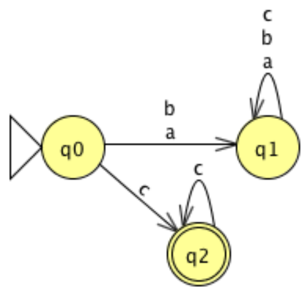
3. The string c is in the language, so we must create an accept state as the destination of transition c from q0.

> **Create a new state, q2, as the destination of transition c from q0 and which is an accept state.**



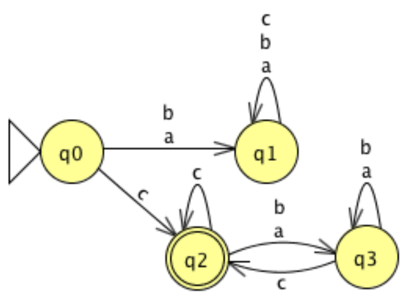
4. Since the string may begin or end with an arbitrary length substring of consecutive c symbols, the DFA can stay in the q2 accept state.

> **Add a c transition to and from q2**



5. All strings in the language that begin with c and end with c can have an arbitrary number of a and b substrings as long as they are followed by a c.

> **Add state q3 with a transition of a or b from q2. Add transitions to remain in q3 for a or b. Add transition from q3 back to q2 for c.**



Questions to Think About

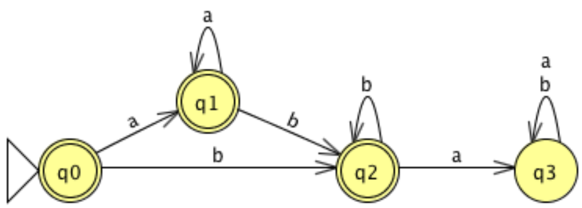
1. How many strings are in the language recognized by the previously developed DFA, `DFA_cxc.jff`?
   Answer: There are an infinite number of strings in the set; thus the language is infinite.

2. Enumerate at least six strings in the language
   { w | w is a string of symbols from alphabet {a, b} in which a never follows b }.
   Answer: λ, *a, b, aa, ab, aaa, aab, abb, abbb, bbbbb*

3. Enumerate at least six strings over alphabet {a, b} ***not*** in the language
   { w | w is a string of a and b symbols in which a never follows b }.
   Answer: ba, bab, aba, abba, bbba, bbab

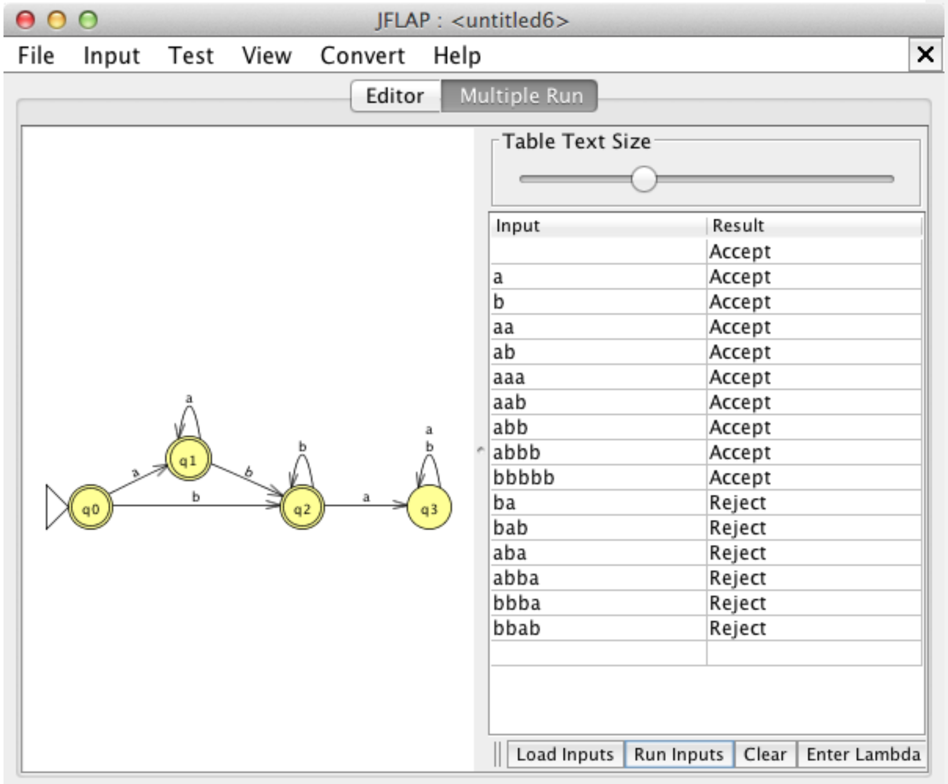
4. Create a DFA for the language
   { w | w is a string symbols from alphabet {a, b} in which a never follows b }.

Answer:



(see: `DFA_akbk.jff`)

5. Check your DFA to ensure that it accepts and rejects the strings you previously identified.
Answer:



References

Kun, Jeremy, *Determinism and Finite Automata – A Primer*
http://jeremykun.com/2011/07/02/determinism-and-finite-automata-a-primer/
[2 July 2011; Accessed on 16 June 2014]

Wikipedia, *Deterministic Finite Automaton*
https://en.wikipedia.org/wiki/Deterministic_finite-state_machine
[13 June 2014; Accessed on 17 June 2014]

Sipser, Michael, *Introduction to the Theory of Computation, Third Edition*
Course Technology / Cengage Learning, 2013.