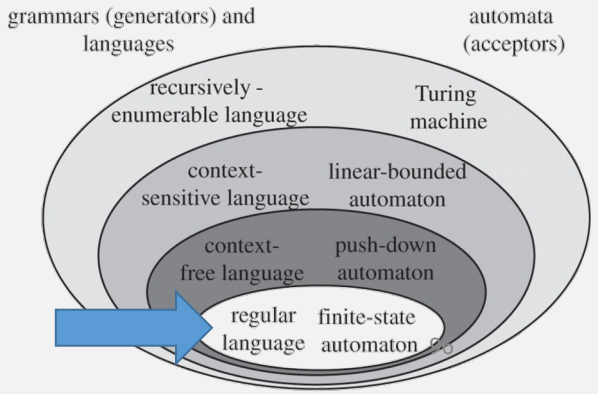
Regular Languages



Logistic Stuff

- Exam 1: 10AM – 8PM Saturday Oct 16
- Exam 2: 10AM – 8PM Saturday Nov 27 (???)
- See Schedule
- Thoughts?

- Homework Due Tonight.
- General questions I can answer?

The Computation Model *Informally*

- Computer = some finite automata
- Program = input string of chars
- Start in “start state”
- 1 char at a time, follow transition table to change states
- Result =
  - “Accept” if stops at an “Accept” state
  - “Reject” otherwise

Last Time: Finite Automata, Formally

5 components

DEFINITION 1.5

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

Last Time: Computation, Formally

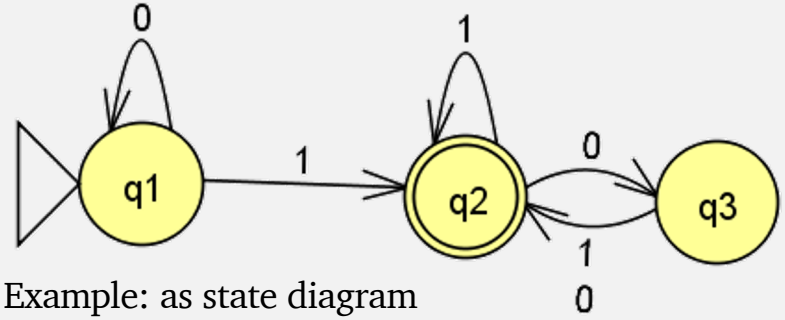
- A finite automata $M = (Q, \Sigma, \delta, q_0, F)$ is a computer
- We "run" on M an input string $w = w_1 w_2 \cdots w_n$, e.g. "1101"
- M **accepts** w if there is sequence of states r_0,...,r_n in Q where:
  - $r_0 = q_0$ **(start in "start" state)**
  - $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \ldots, n-1$ **("next" states follow transition table)**
  - $r_n \in F$ **(last state is an "accept" state)**

Last Time:

DEFINITION 1.5

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.



Example: as state diagram

Example: as formal description

$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

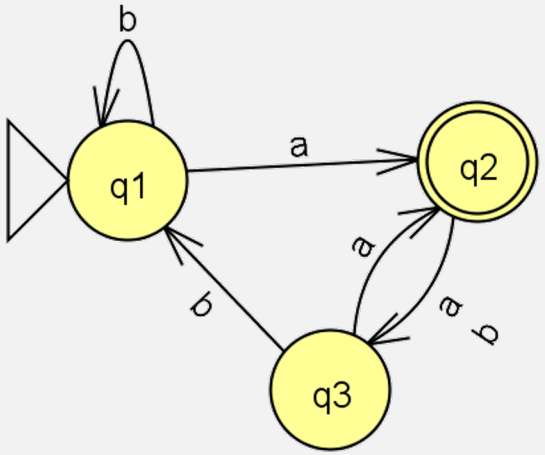
1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,
3. δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,

4. q_1 is the start state, and
5. $F = \{q_2\}$.

In-class exercise 1

- Come up with a formal description of the following machine:

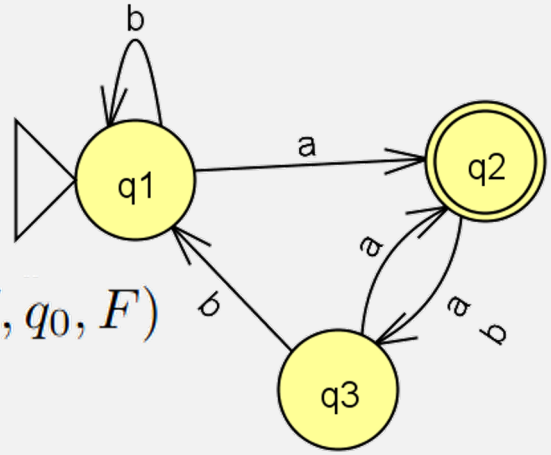


DEFINITION 1.5

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

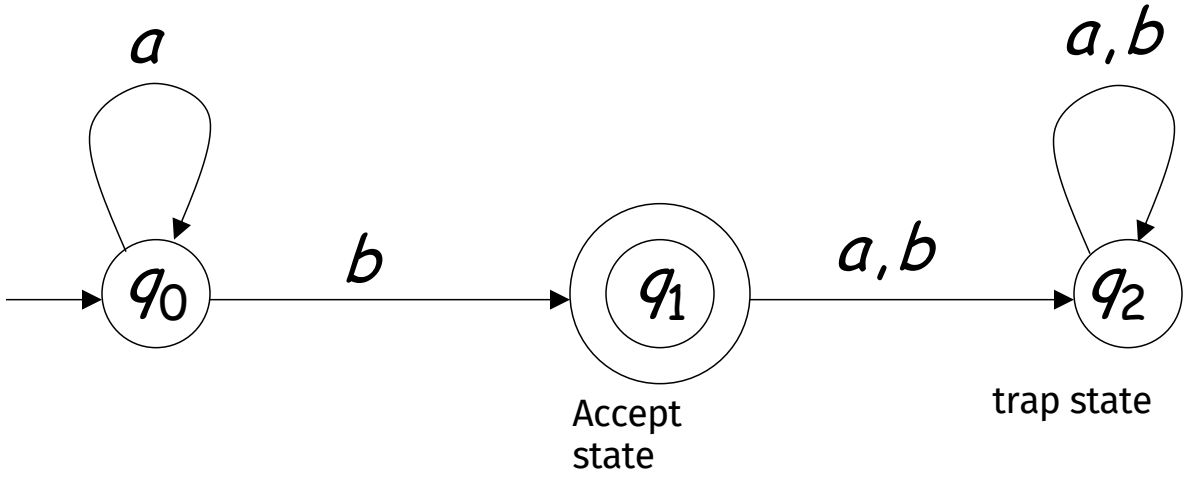
In-class exercise 1: solution



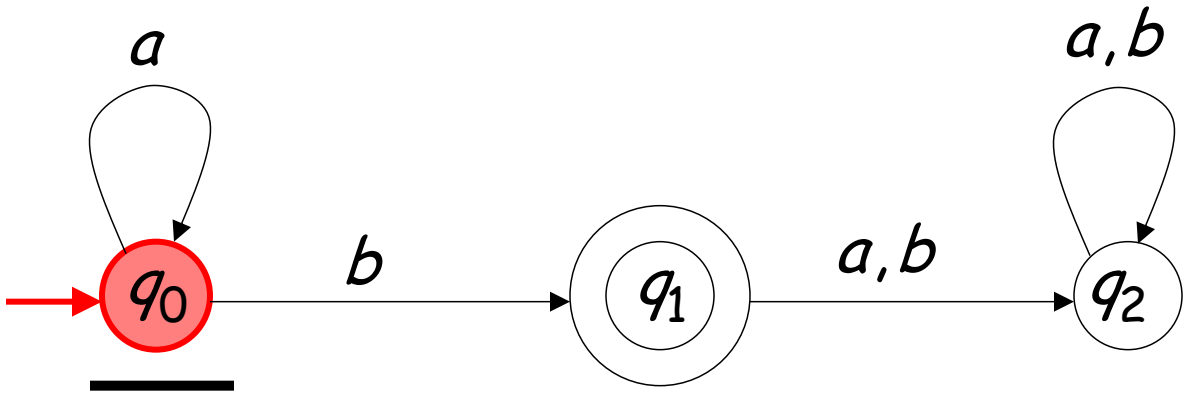
$$M = (Q, \Sigma, \delta, q_0, F)$$

- Q = {q1, q2, q3}
- Σ= {a, b}
- Delta
  - δ(q1,a) = q2
  - δ(q1,b) = q1
  - δ(q2,a) = q3
  - δ(q2,b) = q3
  - δ(q3,a) = q2
  - δ(q3,b) = q1
- q_0 = q1
- F = {q2}

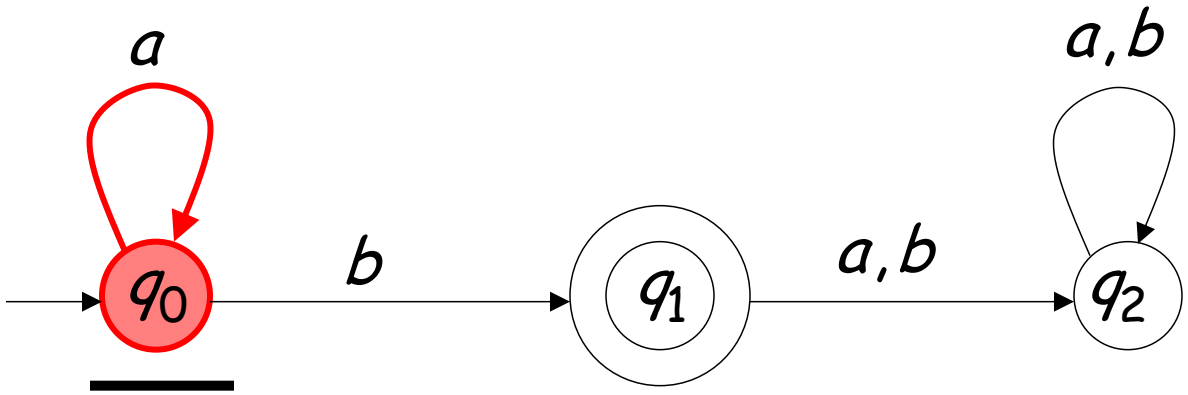
Another Example

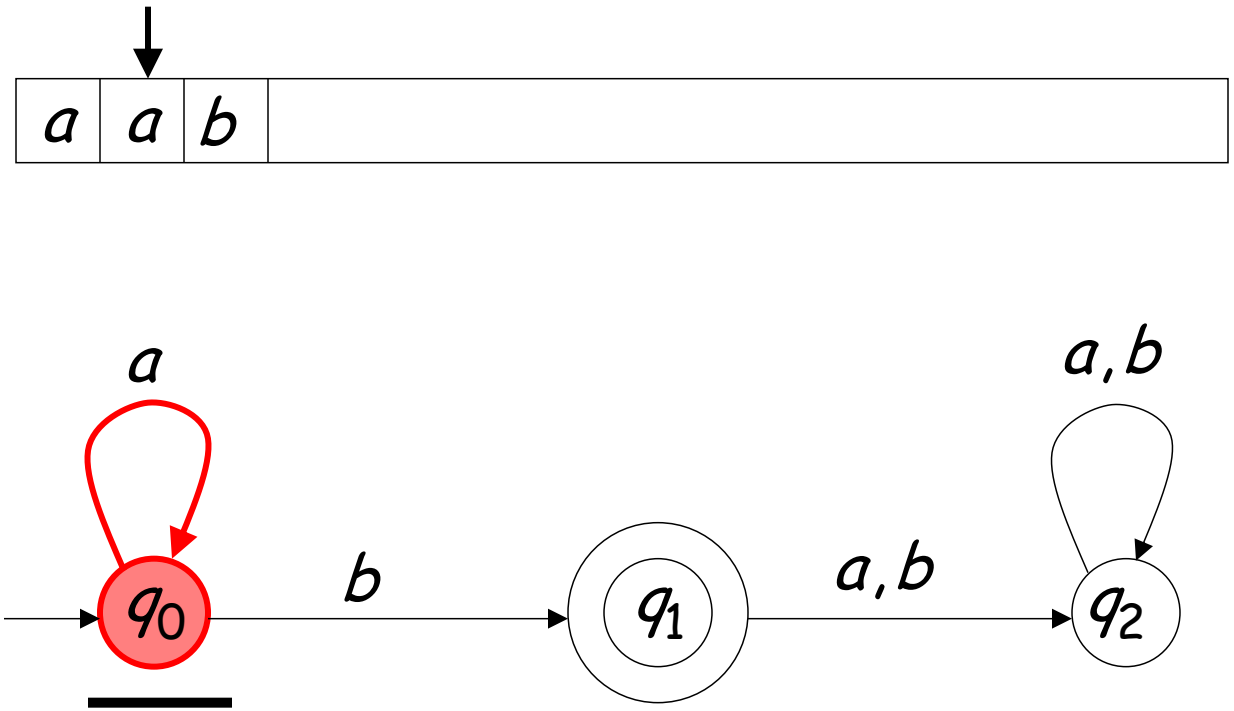


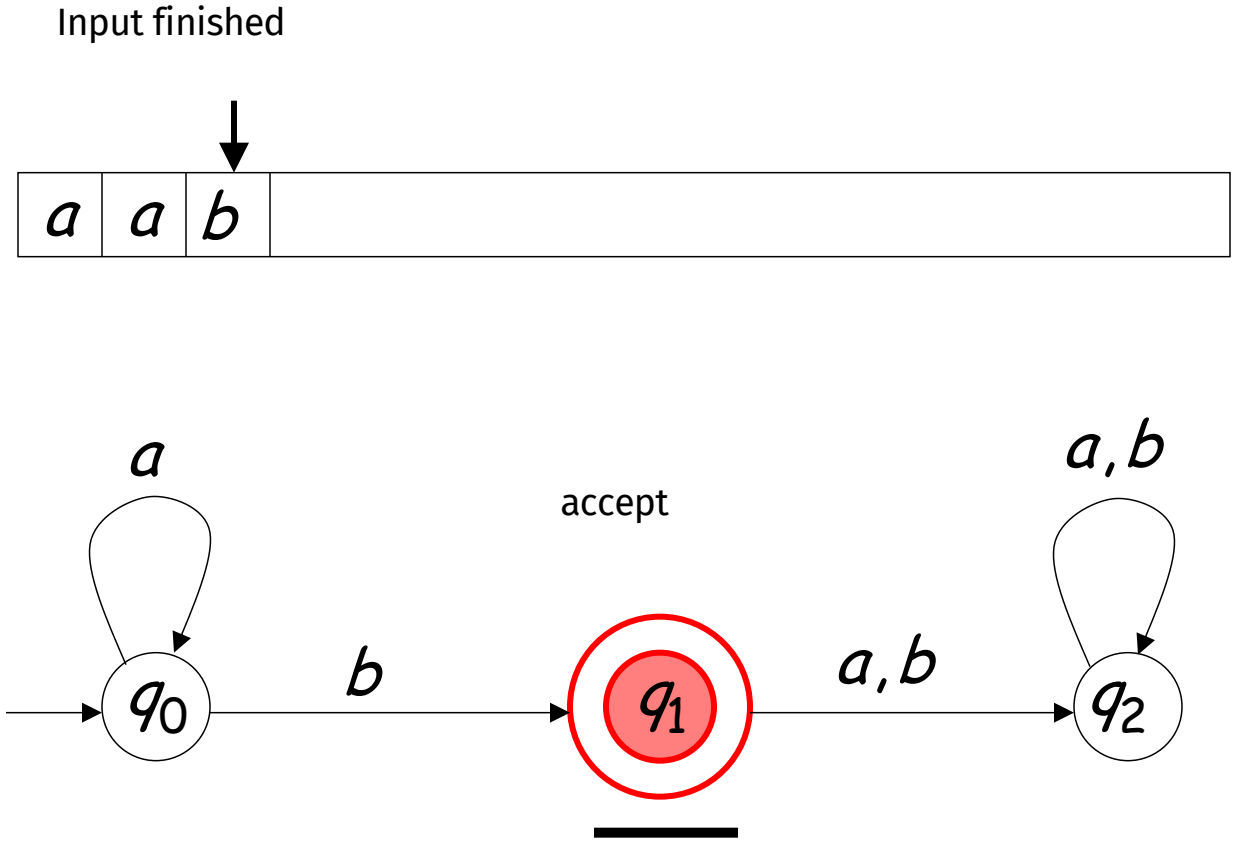
Input String




a
a
b

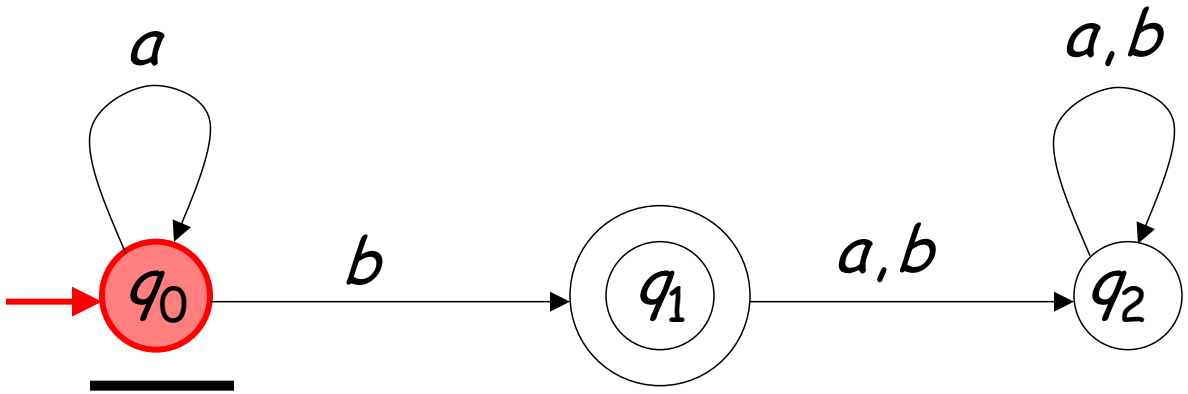

a
b
a, b
a, b
q_0
q_1
q_2

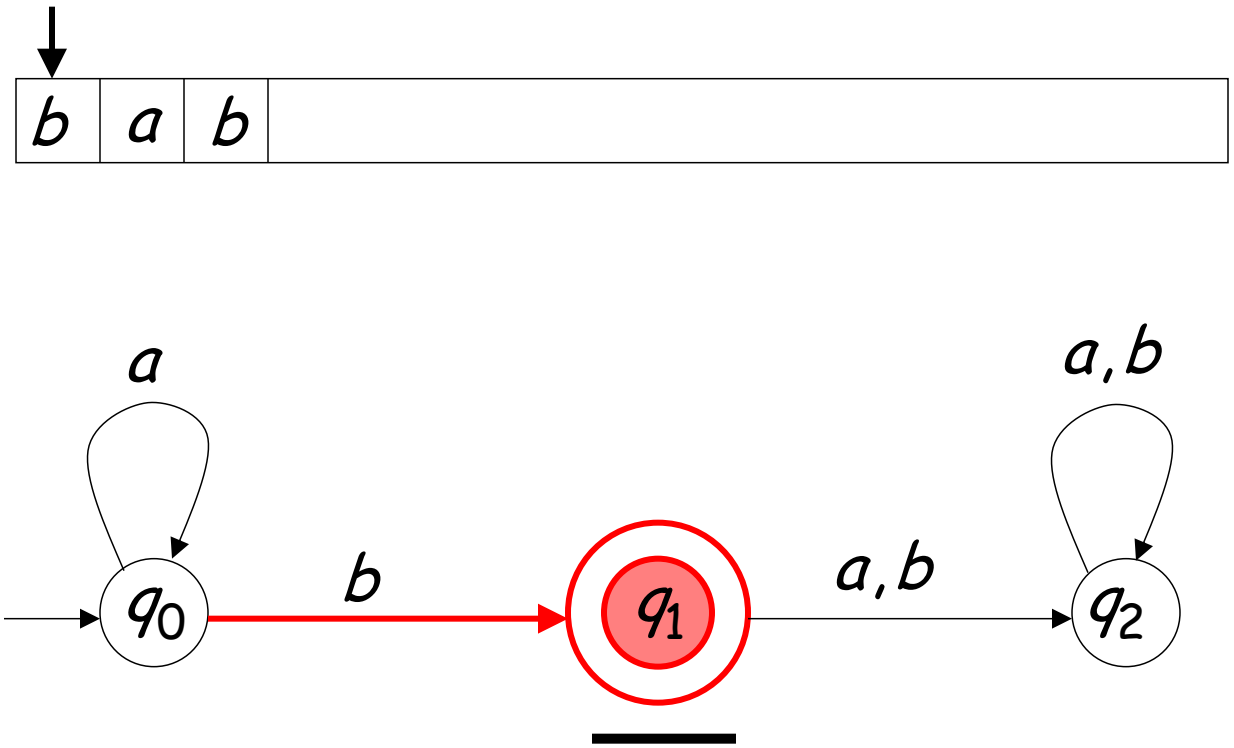

a
a
b
a
b
a, b
a, b
q0
q1
q2

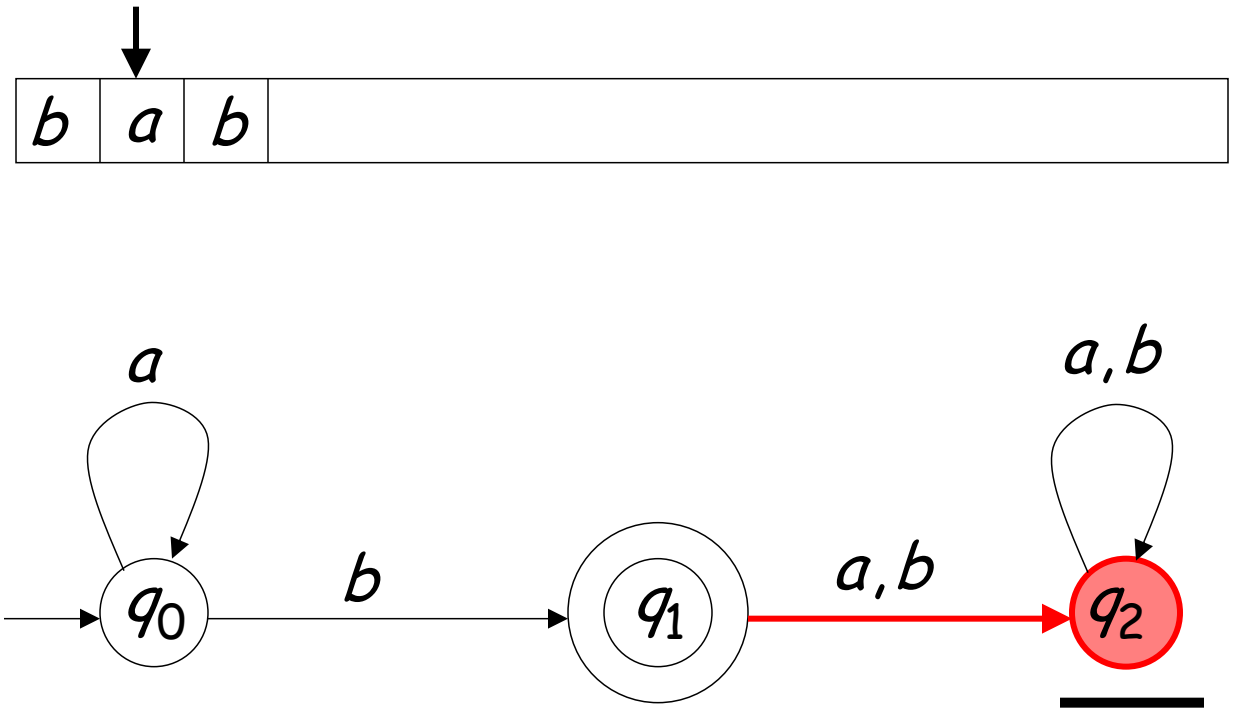

Input finished
a
a
b
a
accept
a, b
b
a, b
q_0
q_1
q_2

A rejection case

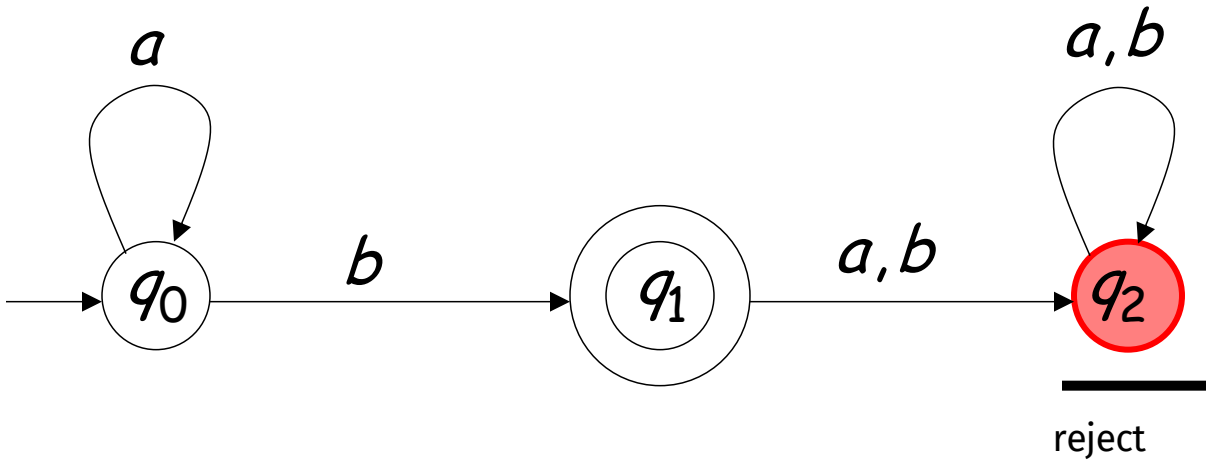
Input String




b
a
b
a
a, b
b
a, b
q_0
q_1
q_2

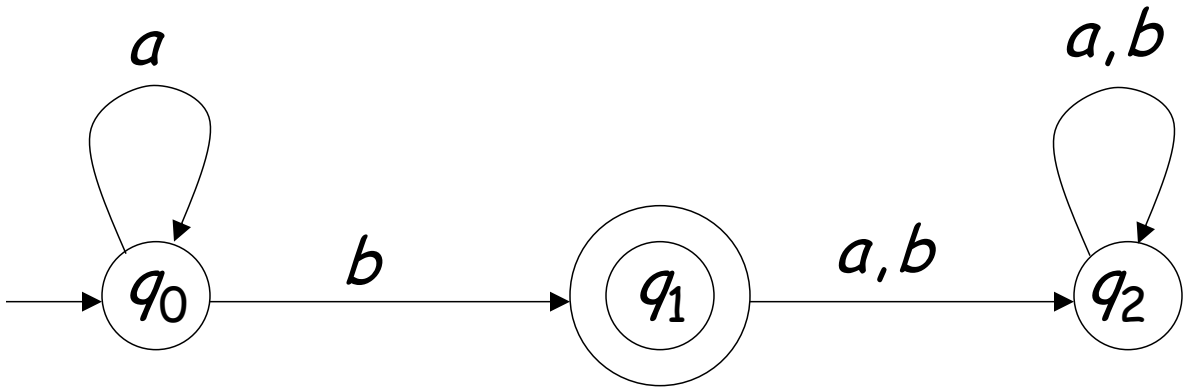

b
a
b
a
a, b
a, b
b
q_0
q_1
q_2

Input finished



Language Accepted:

$$L = \{a^n b : n \geq 0\}$$



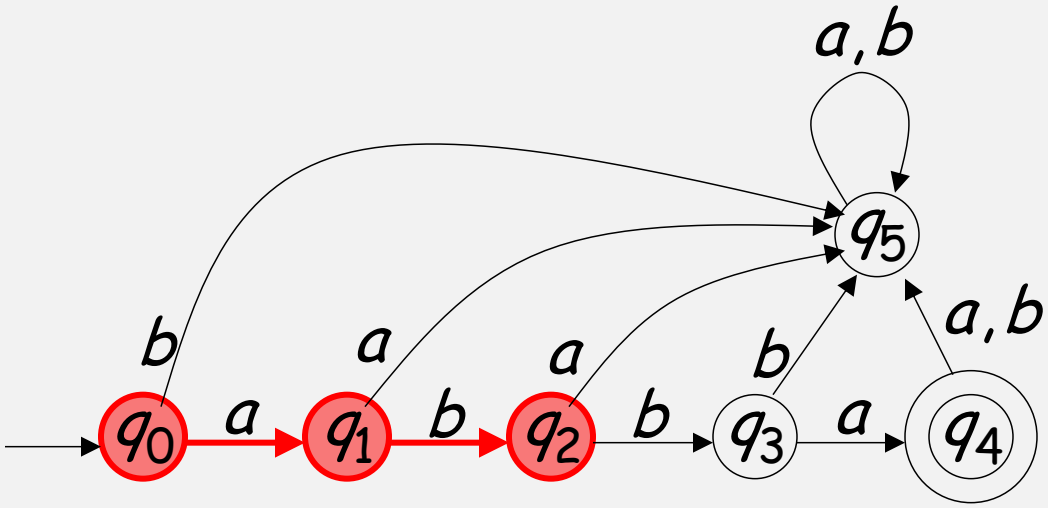
Extended Transition Function

- $\delta^* : Q \times \Sigma^* \to Q$

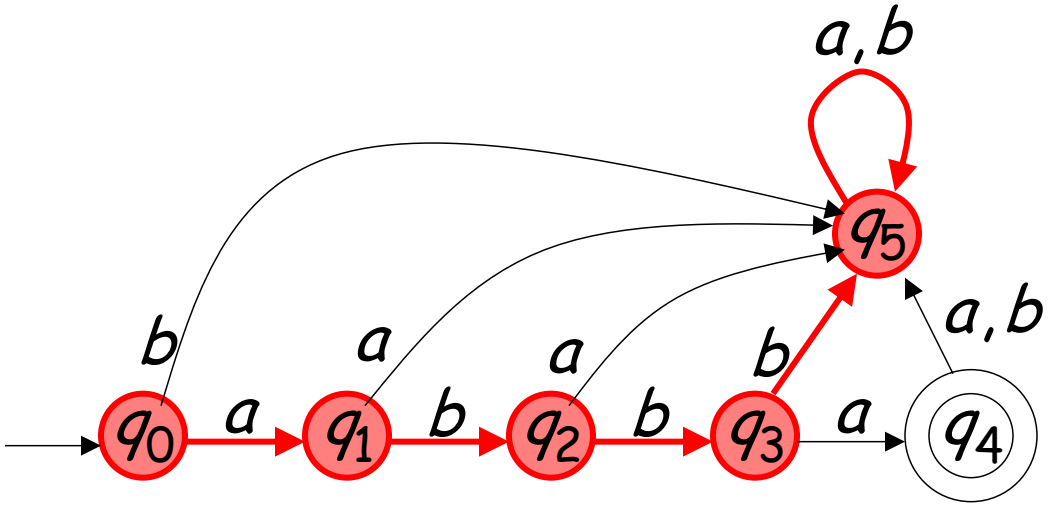
$$\delta^*(q, w) = q'$$

Describes the resulting state
after scanning string w from state q

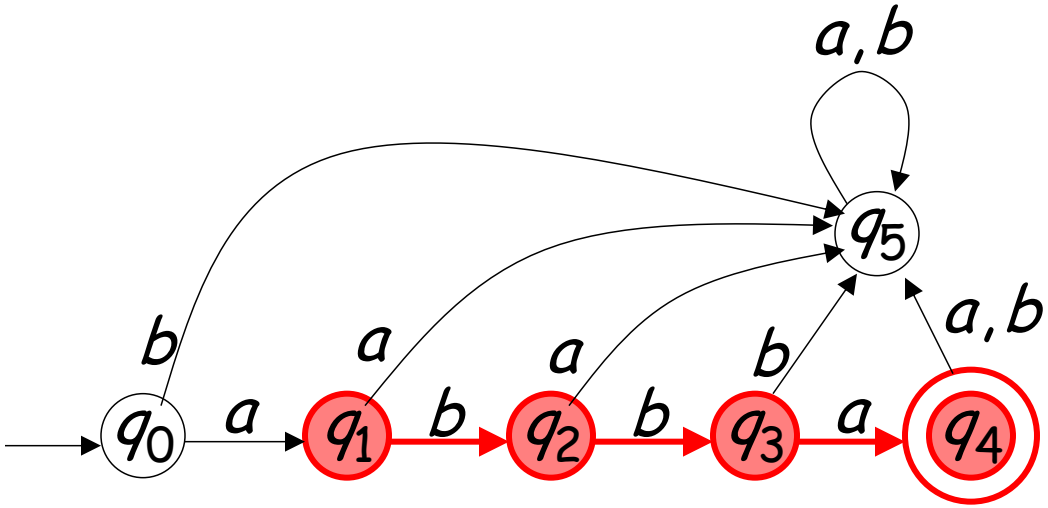
- Example: $\delta^*(q_0, ab) = q_2$



$$\delta^*(q_0, abbbaa) = q_5$$



$$\delta^*(q_1, bba) = q_4$$



Special case:

for any state: $\delta^*(q,\varepsilon) = q$

A ***language*** is a set of strings.

Terminology

- M ***accepts*** w
- M ***recognizes language*** A

$$\text{if } A = \{w | \, M \text{ accepts } w\}$$

“the set of all …” **“such that …”**

A ***language*** is a set of strings.

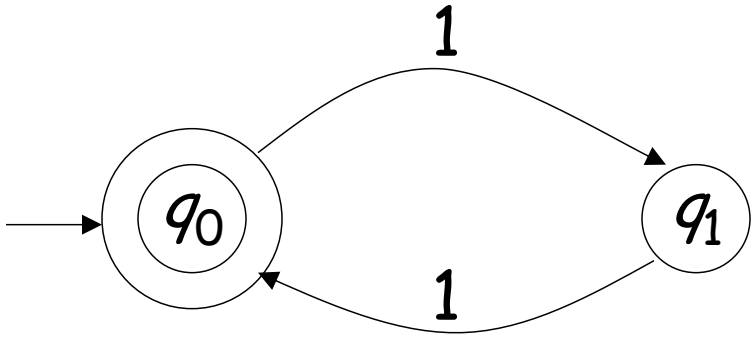
Terminology

- M ***accepts*** w
- M ***recognizes language*** A

$$\text{if } A = \{w | \ M \text{ accepts } w\}$$

Another Example

Alphabet: $\Sigma = \{1\}$



Language Accepted:

$$EVEN = \{x : x \in \Sigma^* \text{ and } x \text{ is even}\}$$
$$= \{\varepsilon, 11, 1111, 111111, \ldots\}$$

Regular Languages

A language is called a ***regular language*** if some finite automaton recognizes it.

- Definition:
- A language *S* is regular if there is
- a DFA M that accepts it (i.e. L(M) = S)

- The languages accepted by all DFAs
- form the family of regular languages

A ***language*** is a set of strings.

M ***recognizes language*** A
if $A = \{w | \ M \text{ accepts } w\}$

A language, regular or not?

- If given: Finite Automata M
  - We know: the language recognized by M is a regular language

- If given: some Language A
  - Is A is a regular language?
    - Not necessarily

There exist languages which are not Regular:

$$L = \{a^n b^n : n \geq 0\}$$

$$ADDITION = \{x + y = z \ : x = 1^n, y = 1^m, z = 1^k, \\ n + m = k\}$$

There is no DFA that accepts these languages

(we will prove this in a later class)

If given: some Language A

How do we determine, i.e., *prove*, that A is a regular language?

Kinds of Mathematical Proof

- Proof by construction
  - Construct the mathematical object in question
- Proof by contradiction

- Proof by induction

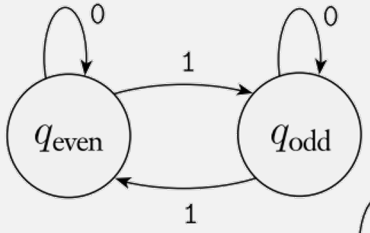
Designing Finite Automata: Tips

- Input may only be read once
- Must decide accept/reject after that
- States = the machine's **memory**!
  - Finite amount of memory: must be allocated in advance
  - Think about what information must be remembered.
- (For DFAs) Every state/symbol pair must have a transition
- Example: machine accepts strings with odd number of 1s

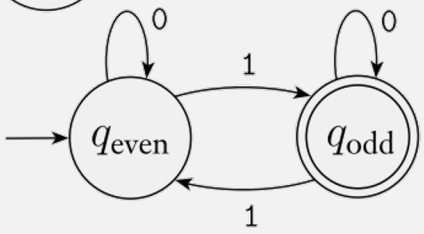
Design a DFA: accepts strs with odd # 1s

- States:
  - 2 states:
    - seen even 1s so far
    - seen odds 1s so far

- Alphabet: 0 and 1
- Transitions:



- Start / Accept states?



In-class exercise 2

- Prove that this language is a regular language:
  - {w | w has exactly three 1's}
  - i.e., design a finite automata that recognizes it!

- Where Σ= {0, 1},

- Remember:

DEFINITION 1.5

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

HW1 Pre-game

- In ToC we primarily learn about abstract mathematical objects
- But we may use code as a way to explore these math objects
- So it's important to understand the distinction: math vs code
- E.g., a set is an abstract mathematical object
  - contains other math objects like: strings, nums, characters, and other sets!
- A set's (data) representation in code can take many forms:
  - e.g., a list, an array, a space-separated string (hw0)

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	
Set	
Tuple (i.e., a small finite set)	
Function, i.e., a set of pairs	
Finite automata	

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	Int, BigInt, float, double
Set	
Tuple (i.e., a small finite set)	
Function, i.e., a set of pairs	
Finite automata	

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	Int, BigInt, float, double
Set	List, array, tree
Tuple (i.e., a small finite set)	
Function, i.e., a set of pairs	
Finite automata	

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	Int, BigInt, float, double
Set	List, array, tree
Tuple (i.e., a small finite set)	Struct, object, list
Function, i.e., a set of pairs	
Finite automata	

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	Int, BigInt, float, double
Set	List, array, tree
Tuple (i.e., a small finite set)	Struct, object, list
Function, i.e., a set of pairs	Function, dict, map, hash, tree
Finite automata	

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	Int, BigInt, float, double
Set	List, array, tree
Tuple (i.e., a small finite set)	Struct, object, list
Function, i.e., a set of pairs	Function, dict, map, hash, tree
Finite automata	XML str, <your choice here>

Sketch out “run” fn pseudocode:

Check-in Quiz

See Gradescope