



Finite Automata



Formal Specification of Languages

Generators

- Grammars
 - Context-free
 - Regular
- Regular Expressions

■ Recognizers

- Parsers, Push-down Automata
 - Context Free Grammar
- Finite State Automata
 - Regular Grammar

■ A Finite Automata is:

- a mechanism to recognize a set of valid inputs before carrying out an action.
- a notation for describing a family of language recognition algorithms.

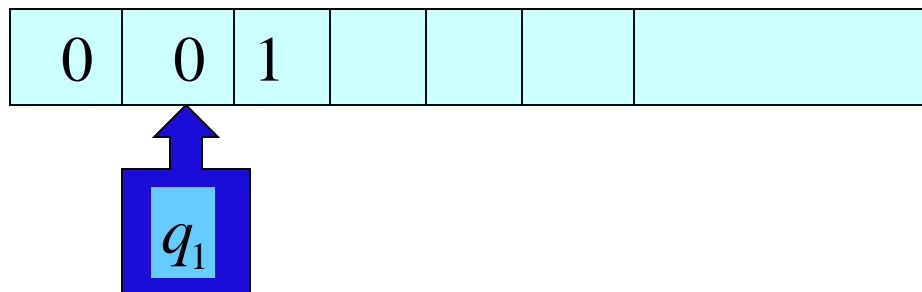
Operation of the machine



- Read the current letter of input under the tape head.
- Transit to a new state depending on the current input and the current state, as dictated by the transition function.
- Halt after consuming the entire input.

Operation of the machine

- Transitions show the initial state, input, and next state
 - Form: $\delta(q, a) = b$
- Example:
 - $\delta(q_0, 0) = q_1$ $\delta(q_0, 1) = q_2$
- Tape head advances to next cell, in state q_1
- What happens now?
 - What is $\delta(q_1, 0)$?



Associating Language with the DFA

- Machine configuration:

$$[q, \omega] \text{ where } q \in Q, \omega \in \Sigma^*$$

- Yields relation:

$$[q, a\omega] \mapsto_M^* [\delta(q, a), \omega]$$

- Language:

$$\{\omega \in \Sigma^* \mid \underbrace{[q_0, \omega] \mapsto_M^* [q, \lambda]} \wedge q \in F\}$$



Deterministic Finite Automaton (DFA)

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

Q : Finite set of states

Σ : Finite Alphabet

δ : Transition function

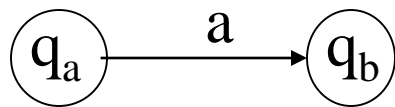
- a total function from $Q \times \Sigma$ to Q

q_0 : Initial/Start State

F : Set of final/accepting state

Finite State Diagram

- A graphic representation of a finite automaton
- A **finite state diagram** is a directed graph, where nodes represent elements in Q (i.e., states) and arrows are characters in Σ such that:



Indicates: $((q_a, a), q_b)$ is a transition in δ

The initial state is marked with: 

The final state(s) are marked with: 



Automata

- Deterministic automata – each move is uniquely determined by the current configuration
 - Single path
- Nondeterministic automata – multiple moves possible
 - Can't determine next move accurately
 - May have multiple next moves or paths



Automata

- An automaton whose output response is limited to yes or no is an acceptor
 - Accepts input string and either accepts or rejects it
- Measures of complexity
 - Running time
 - Amount of memory used



Automata

- Finite automaton
 - Uses a limited, constant amount of memory
 - Easy to model
 - Limited application



Automata

Given an automaton $A = (Q, \Sigma, \delta, q_0, F)$, and a string $w \in \Sigma^*$:

- w is **accepted** by A if the configuration (q_0, w) yields the configuration (F, ϵ) , where F is an accepting state
- the **language accepted by** A , written $L(A)$, is defined by:
$$L(A) = \{w \in \Sigma^* : w \text{ is accepted by } A\}$$



Automata

- Deterministic finite automaton
 - Every move is completely determined by the input and its current state
 - Finite control device
 - Can be in any one of the states, $q \in Q$
 - May contain trap or dead states
 - Contains accepting state(s)



Examples

Deterministic



Language Acceptor

Design a FSA to accept strings of a language

- strings of a's and b's that start and end with a

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

Q : States are required for the following:

- q_0 : Start state
- q_t : Trap for strings that start with a b
 - Accepting state can't be reached
 - Machine only accepts strings that start with an a
- q_f : State reached after any a in a string that started with an a
 - The final state
- q_1 : State reached after any b in a string that started with an a



Language Acceptor

$\Sigma : \{a, b\}$

$\delta :$ $\delta(q_0, a) = q_f$ Is the string a in the language?

$\delta(q_0, b) = q_t$

$\delta(q_t, a) = q_t$

$\delta(q_t, b) = q_t$

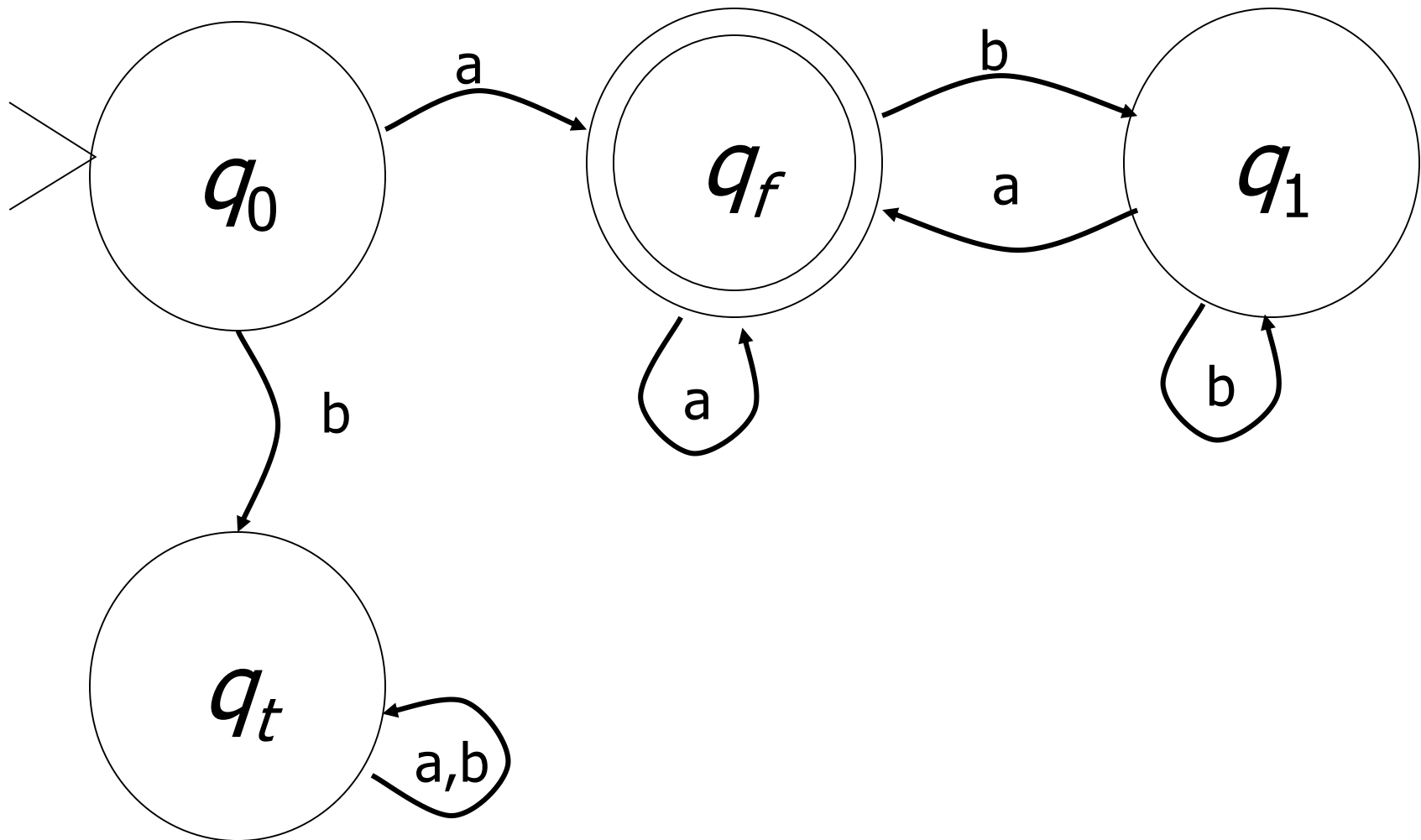
$\delta(q_f, a) = q_f$

$\delta(q_f, b) = q_1$

$\delta(q_1, a) = q_f$

$\delta(q_1, b) = q_1$

Language Acceptor





Vending Machine

- Suppose:
 - All items cost 40¢
 - Coins accepted are 5¢, 10¢, 25¢
 - Recall $M = (Q, \Sigma, \delta, q_0, F)$
 - What are these entities?
 - Q is a set of states
 - What are the possible states?
 - Σ is the alphabet
 - What are the input symbols?
 - δ are the transitions
 - How do we move from state to state?
 - q_0 is the starting state
 - Where does the machine start from?
 - F is the final state
 - When does the machine stop?



Vending Machine

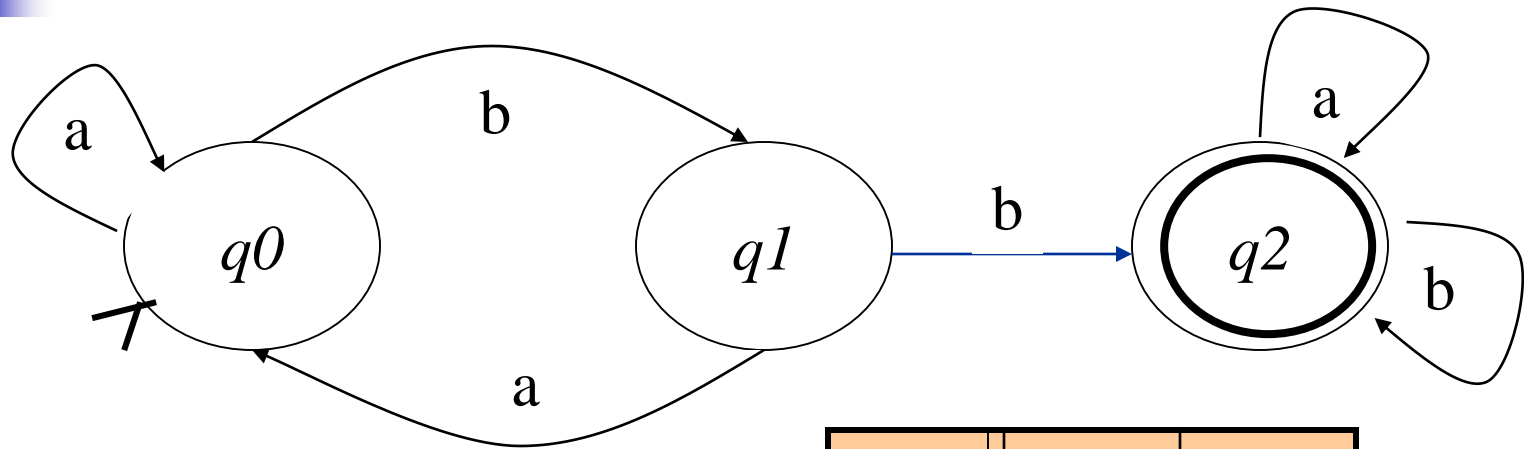
- Q : What are the possible states?
 - The status of the machine before and after any of the alphabet symbols have been applied
 - The present state represents how much money has been deposited
 - Could also represent how much is left to deposit
- Σ : What are the input symbols?
 - The coin denominations
- δ : How do we move from state to state?
 - Transition when a coin is deposited
- q_0 : Where does the machine start from?
 - The beginning!
- F is the final state
 - When does the machine stop?
 - Not before you've deposited enough money
- Wait! What if you put in more than 40¢?



Set of strings over $\{a,b\}$ that contain bb

- Design states by partitioning Σ^* .
 - Strings containing bb $q2$
 - Strings not containing bb
 - Strings that end in b $q1$
 - Strings that do not end in b $q0$
 - Initial state: $q0$
 - Final state: $q2$

State Diagram and Table



$Q = \{q0, q1, q2\}$

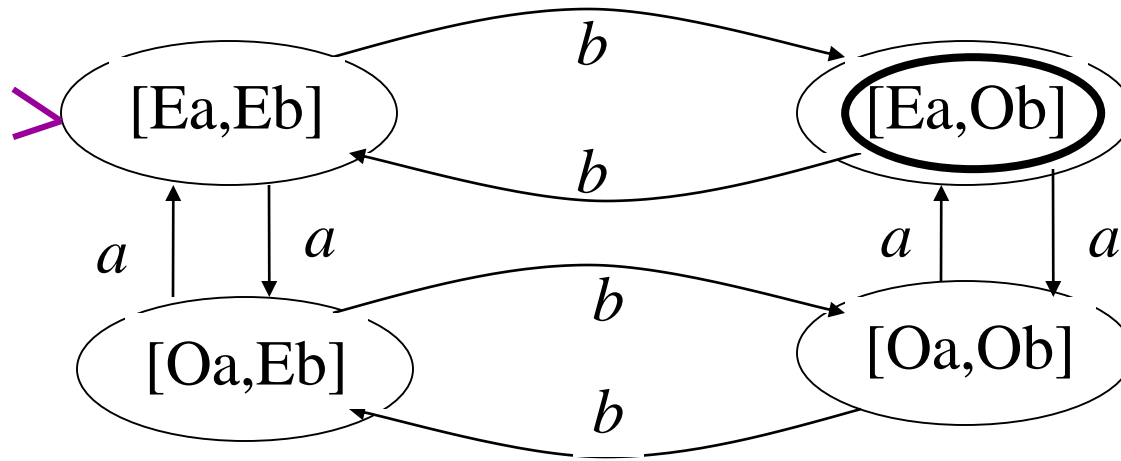
$\Sigma = \{a, b\}$

$F = \{q2\}$

$[q0, aab] \mapsto^* [q1, \lambda]$

δ	a	b
<i>q0</i>	<i>q0</i>	<i>q1</i>
<i>q1</i>	<i>q0</i>	<i>q2</i>
<i>q2</i>	<i>q2</i>	<i>q2</i>

Strings over $\{a,b\}$ containing even number of a 's and odd number of b 's.





Non-Determinism



Automata

- Non-deterministic finite automaton
 - More than one destination from a state with a distinct input
 - At least one state has transitions that cannot be completely determined by the input and its current state
 - It is possible to design a machine where a single input can have two paths to an accepting state
 - ϵ transitions
 - Move from a state without input

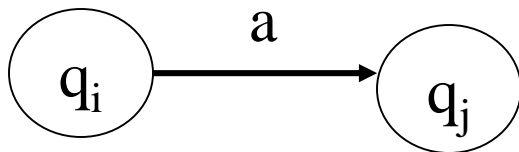
- Nondeterministic finite automata
 - Quintuple $A = (Q, \Sigma, \Delta, s, F)$ where
 - Q is a finite set of states
 - Σ Is an input alphabet
 - $\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$ is the transition relation
 - $S \in Q$ is the initial state of the automaton
 - $F \subseteq Q$ is the set of favorable states



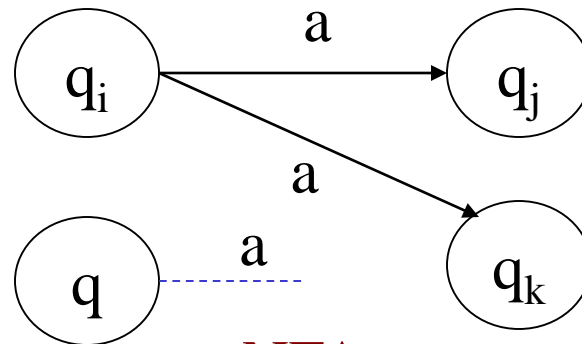
NDFA

- The range of Δ is not a single element of q , but a subset of q
 - Example: $\Delta(q_1, a) = \{q_0, q_2\}$
 - Either q_0 or q_2 could be resultant state
- Empty string (null input) can result in a change of state

NFA



DFA



NFA

$$\delta_{DFA} : Q \times \Sigma \rightarrow Q$$

total function

$$\delta_{NFA} : Q \times \Sigma \rightarrow Pow(Q)$$

total function

$$\delta_{NFA} \subseteq Q \times \Sigma \times Q$$

subset relation



NDFA

- Stuck state – no jumps out of state labeled with the input symbol
 - Can not lead to accepting state
 - Removes need for trap state
- Input is accepted if any resultant configurations lead to an accepting state



NDFA

- Given an NFA $A = (Q, \Sigma, \Delta, q_0, F)$, and a string $w \in \Sigma^*$:
 - w is **accepted** by A if at least one of the configurations yielded by (q_0, w) is a configuration of the form (F, ε) with f a favorable state
 - $L(A) = \{w \in \Sigma^* : w \text{ is accepted by } A\}$



Examples

Non-Deterministic



Language Acceptor (Revisited)

Design a FSA to accept strings of a language

- strings of a's and b's that start and end with a

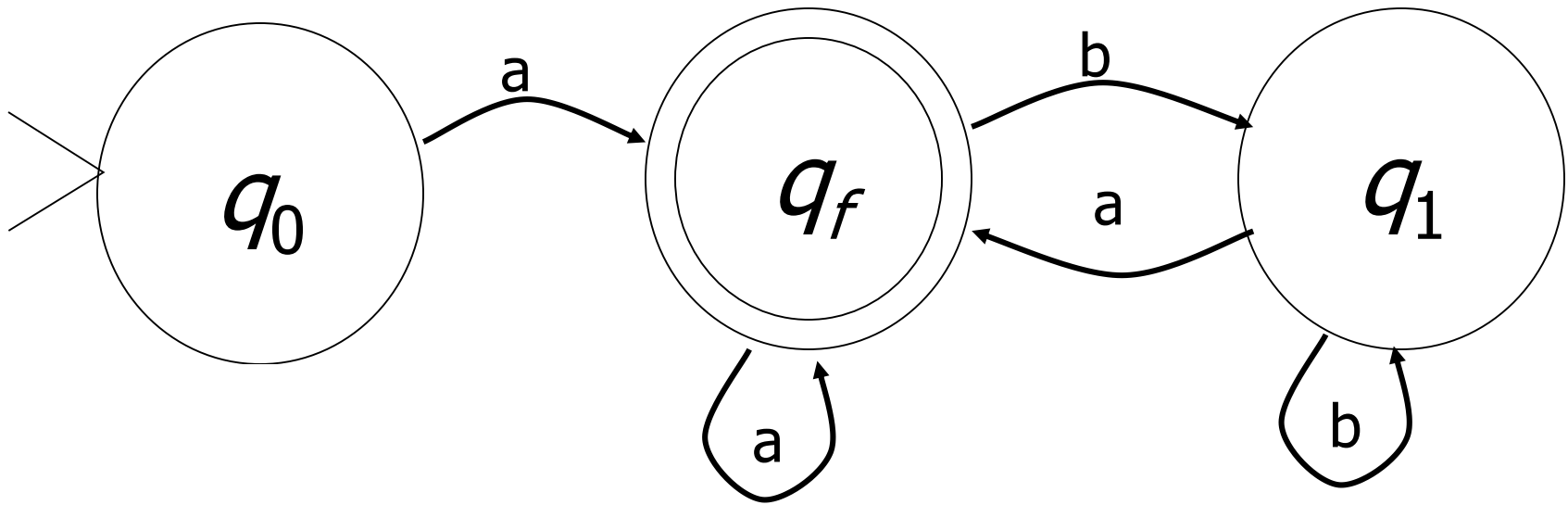
$$M = (Q, \Sigma, \delta, q_0, F)$$

- Only change is in δ morphing to Δ

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

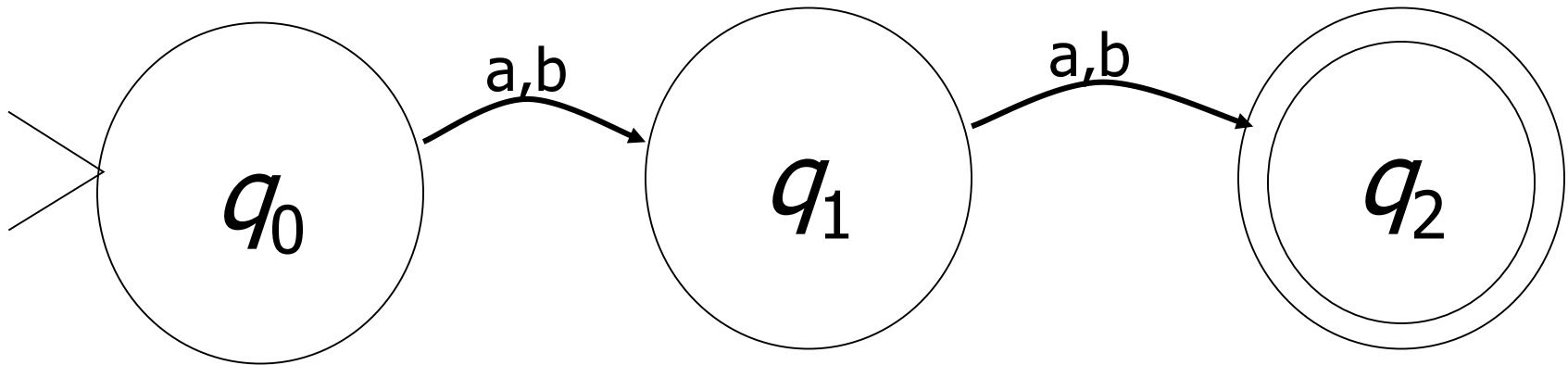
- Some transitions eliminated

Language Acceptor



Another Language Acceptor

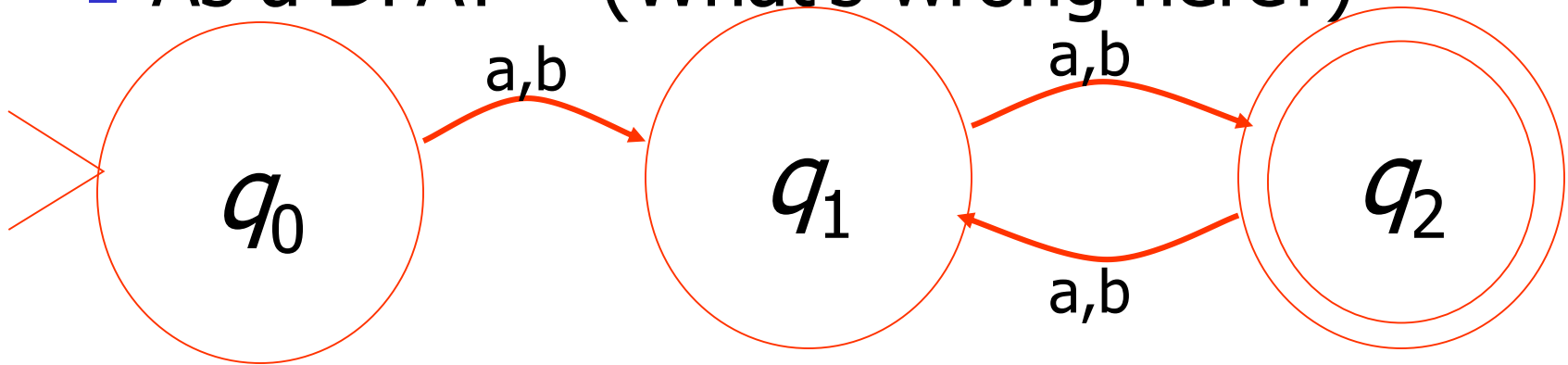
- Build a FA to accept strings of even length



- Wait! This only accepts strings of length 2
- How to update?

Another Language Acceptor

- As a DFA? (What's wrong here?)



- As an NFA (use ϵ transitions)

