DFA

COMP 4200 – Formal Language

ALPHABETS AND STRINGS

- An alphabet is any finite set of characters.
  - Examples: {0,1}, {a,b,c}, {0,1,#}, {a,….z,A,…..Z}
  - Typically represented by Σ.
- A string over an alphabet Σ is a finite sequence of characters from Σ.
  - Examples: Σ = {a, b, c} some valid strings include
    - abc,
    - baba,
    - aaaabbbbccc.
- Empty string, denoted by $\in$, with length 0.
- Length, number of characters in string, denoted by $|x|$

LANGUAGE

- A Language is a set of strings.
- We say that L is a *language over Σ* if it is a set of strings formed from characters in Σ.
- Example: The language of palindromes over Σ = {a, b, c} is the set

  {ε, a, b, c, aa, bb, cc, aaa, aba, aca, bab, … }
- One special language is Σ^*, which is the set of all possible strings generated over the alphabet Σ^*.
- Formally we can say, L is a language over Σ iff $L \subseteq \Sigma^*$.
- Example: Σ = {a, b, c} then Σ^* = {ε, a, b, c, aa, ab, ac, ba, . . . , aaaaaabbbaababa, . . .} .

SUMMARY

- A finite automaton is a collection of states joined by transitions.
- Some state is designated as the start state.
- Some states are designated as accepting states.
- The automaton processes a string by beginning in the start state and following the indicated transitions.
- If the automaton ends in an accepting state, it *accepts* the input. Otherwise, the automaton *rejects* the input.

FORMAL DEFINITION OF FA

- A finite automaton is a 5-tuple (Q,Σ, δ, q0, F), where

1. Q → a finite set called the states,
2. Σ → a finite set called the alphabet,
3. δ → Q x Σ, transition function,
4. q0 → the start/initial state, q0 ∈ Q
5. F → the set of accept/final states, F ⊆ Q

FA EXAMPLE

- Let M: ({q0,q1,q2,q3},{a,b},q0,q1,δ) where transition is given by δ(q0,a)=q1, δ(q1,a)=q3, δ(q2,a)=q2, δ(q3,a)=q2; δ(q0,b)=q2, δ(q1,b)=q0, δ(q2,b)=q2, δ(q3,b)=q2.
  - Represent M by its state table
  - Represent M by its state diagram
  - Which of the following strings are accepted by M ababa, aabba.

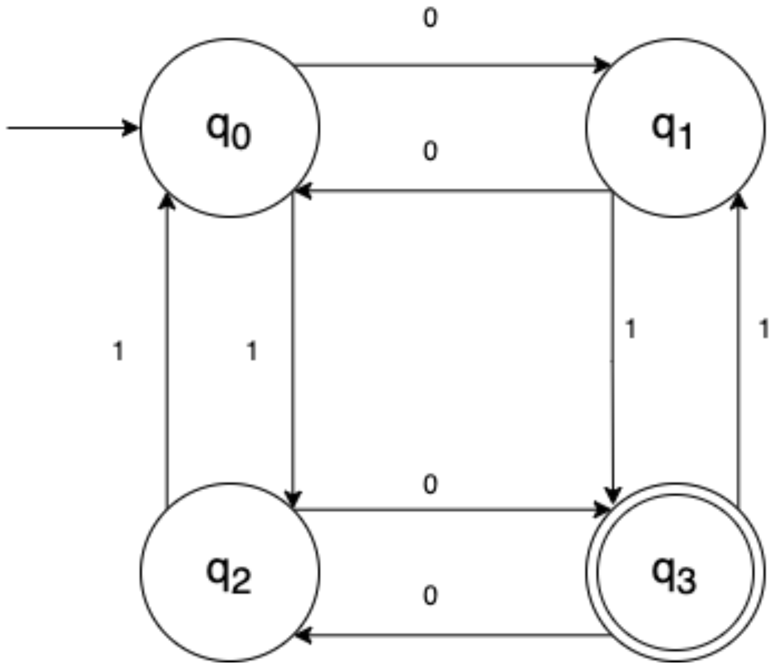
DETERMINISTIC FINITE AUTOMATON(DFA)

- A DFA is a
  - **D**eterministic
  - **F**inite
  - **A**utomaton
- DFAs are the simplest type of automaton.
- It has very limited memory

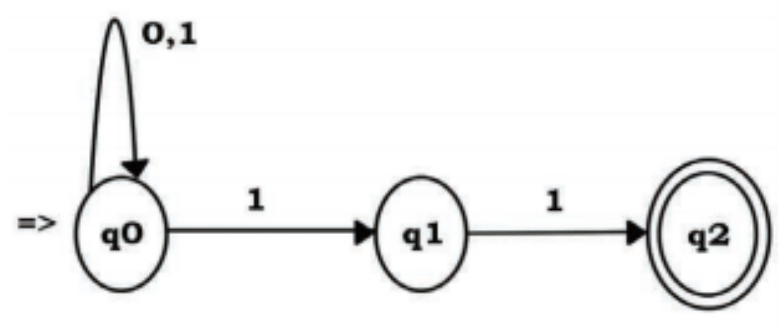
INFORMAL DEFINITION OF DFA

- A DFA is defined relative to some alphabet Σ.
- For each state in the DFA, there must be exactly one transition defined for each symbol in the alphabet.
  - This is the "deterministic" part of DFA.
- There is a unique start state.
- There are zero or more accepting states.

IS THIS A DFA?

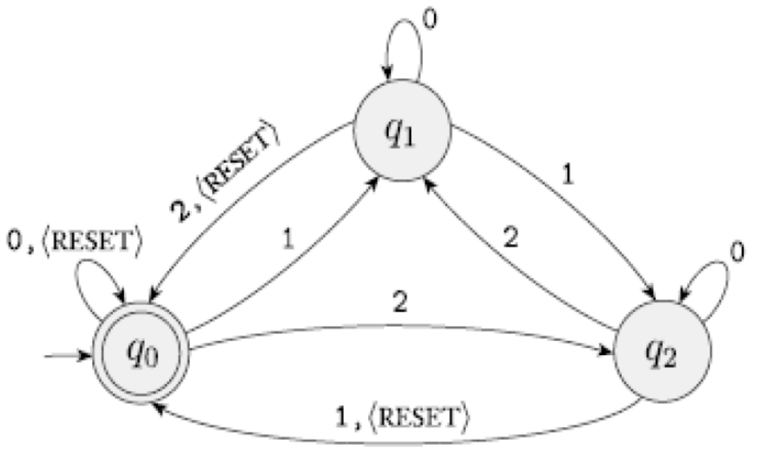


IS THIS A DFA?



FA EXAMPLE

- Σ = {<RESET>, 0, 1, 2}. <RESET> is considered as a single symbol.
- Keeps count of the sum of the numeric input symbols it reads, modulo 3.
- When it reaches <RESET>, it resets the count to 0.
- This machine accepts if the sum is a multiple of 3.



FA EXAMPLE

Let w be the string

10⟨RESET⟩22⟨RESET⟩012.

Then Machine accepts w according to the formal definition of computation because the sequence of states it enters when computing on w is

q0, q1, q1, q0, q2, q1, q0, q0, q1, q0,

which satisfies the three conditions.

The language of M is

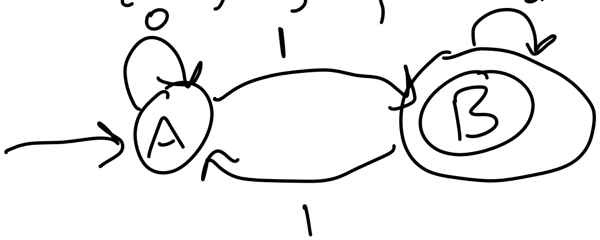
L(M) = {w | the sum of the symbols in w is 0 modulo 3, except that ⟨RESET⟩ resets the count to 0}.

As *M recognizes this language, it is a regular language.*

HOW TO DESIGN FINITE AUTOMATON?

The alphabet is {0,1} and that the language consists of all strings with an odd number of 1s. Construct a finite automaton E_1 to recognize this language.

$\Sigma = \{0, 1\}$ $L = \{01, 10, 111, \ldots\}$



0001

$A \xrightarrow{0} A \xrightarrow{0} A \xrightarrow{0} A \xrightarrow{1} B$

1110

$A \xrightarrow{1} B \xrightarrow{1} A \xrightarrow{1} B \xrightarrow{0} B$

HOW TO DESIGN FINITE AUTOMATON?

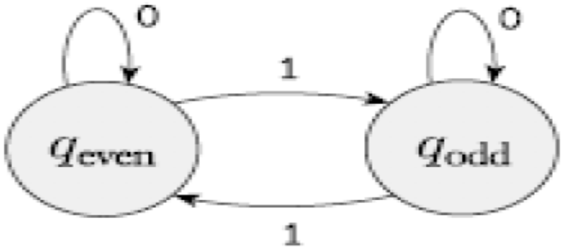
The alphabet is {0,1} and that the language consists of all strings with an odd number of 1s. Construct a finite automaton E_1 to recognize this language.

In this instance, the possibilities would be

1. even so far, and

2. odd so far.

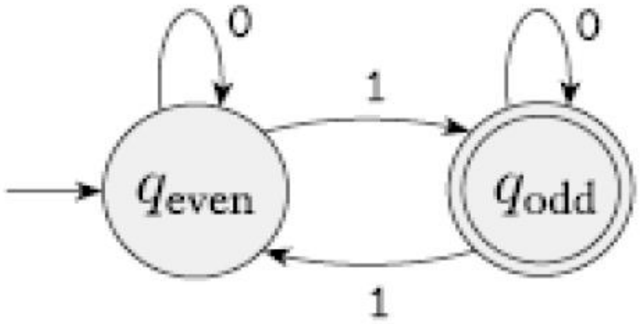
Assigning the transitions. If state q_{even} represents the even possibility and state q_{odd} represents the odd possibility, you would set the transitions to flip state on a 1 and stay put on a 0.



Things to do, remember the tuple,

Start state → q_{even} , because 0 is even

Accepting state → q_{odd} , because you want to accept when you have seen an odd number of 1s.



DESIGNING FINITE AUTOMATON EXAMPLE 2

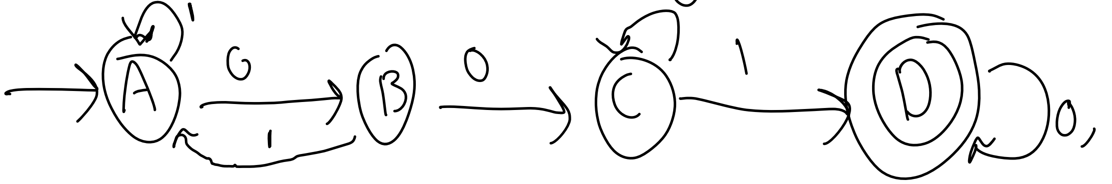
Design a finite automaton E_2 to recognize the regular language of all strings that contain the string 001 as a substring.

Example: 0010, 1001, 001, and 11111110011111 are all in the language, but 11 and 0000 are not.

How would you recognize this language if you were pretending to be E_2?

- Skip all the 1's (because the substring is 001)
- When you see 0, its just first of 3 symbols in the pattern 001
  - Next symbol is 1 → few 0's so go back to skipping 1's
  - Next symbol is 0 → still second symbol in the pattern 001
- Just scan the input until you get 1.
- Once pattern/substring found, scan all the inputs until end.

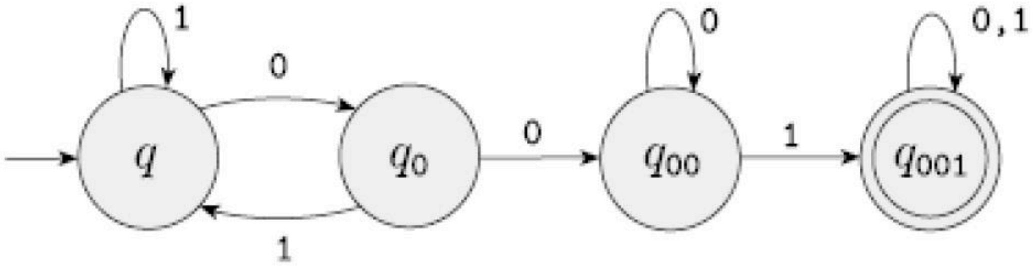
001



A
B
C
D
0
1
1
0
0
1
0, 1

- Possibilities
  1. haven't just seen any symbols of the pattern,
  2. have just seen a 0,
  3. have just seen 00, or
  4. have seen the entire pattern 001.

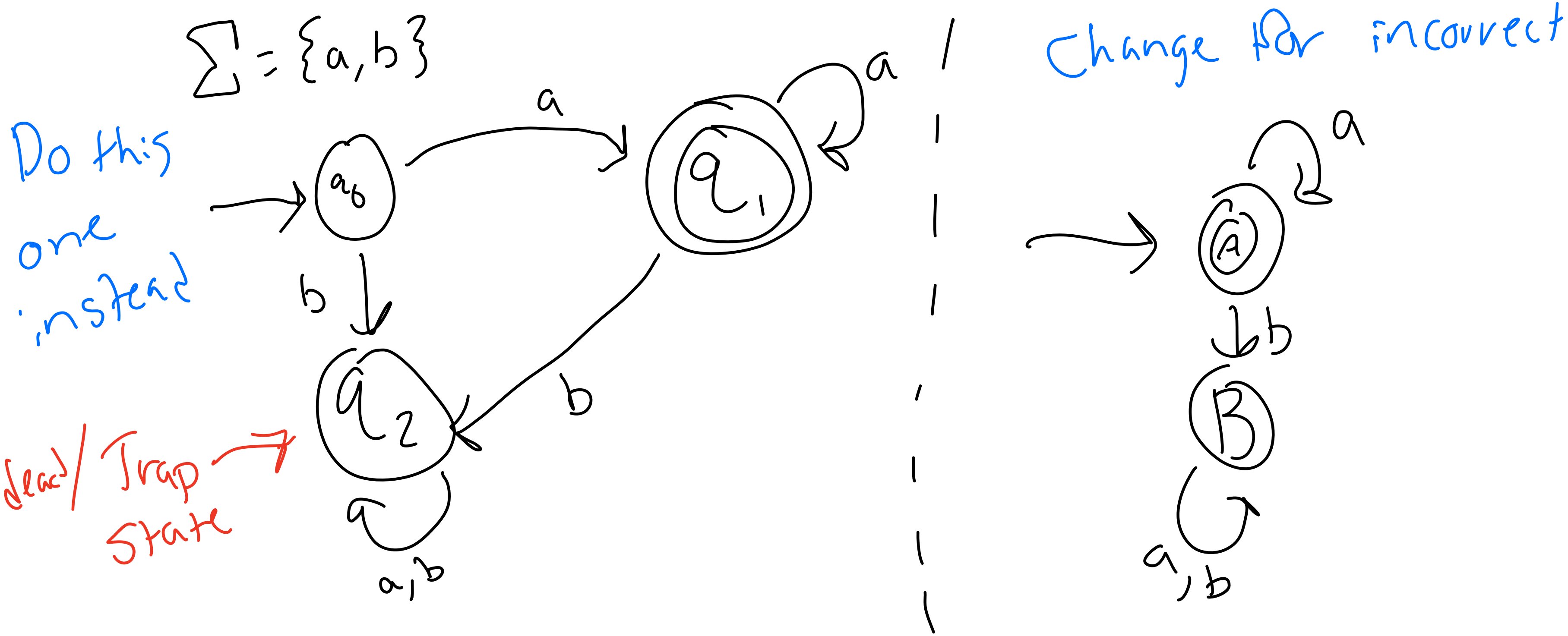
- Next step assign states q, q_0, q_{00}, q_{001}
- Assigning transitions, for
  - q → skip/stay in the same state for all 1's but reading a 0 you move to q0(next state).
  - q0 → reading a 1 you return to q but reading a 0 you move to q00(next state).
  - q00 → reading a 1 you move to q001 but reading a 0 leaves you in q00.
  - Q001 → reading a 0 or a 1 leaves you in q001.
- The start state is q, and the only accept state is q001.



EXAMPLES OF FINITE AUTOMATA

EXAMPLE – 1

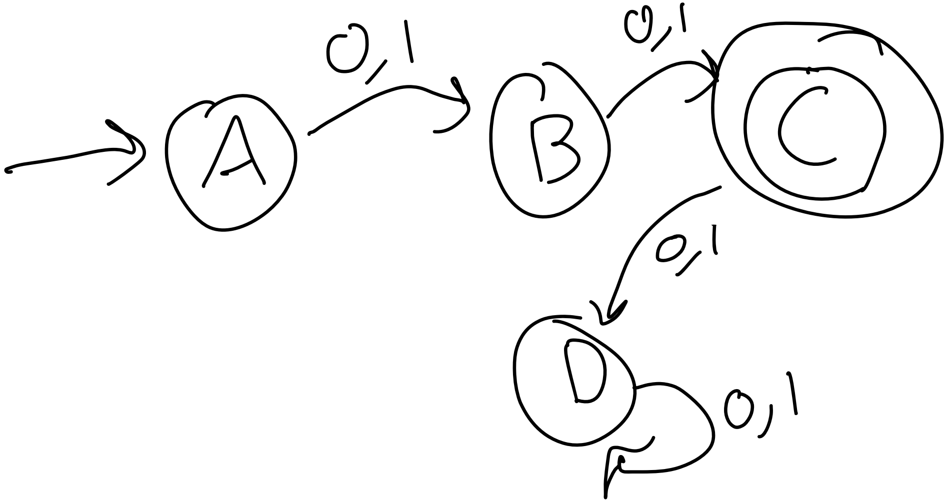
Construct DFA that accepts all the strings of only a's over the alphabet {a,b}



EXAMPLE – 2

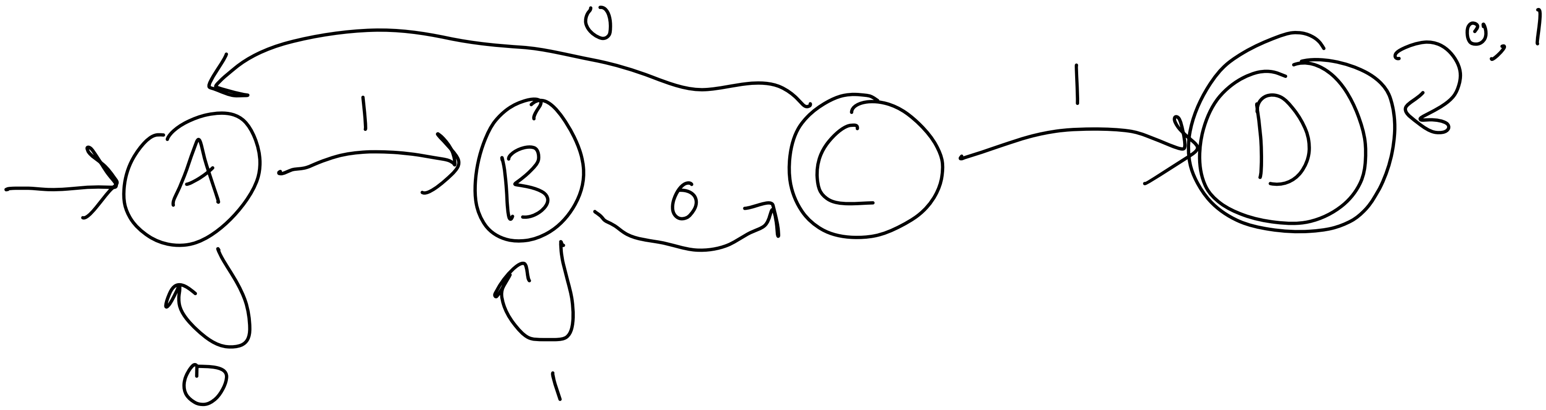
Construct DFA that accepts all the strings over the alphabet {0,1} of length 2.

$\Sigma = \{0,1\}$ $L = \{00, 01, 10, 11\}$



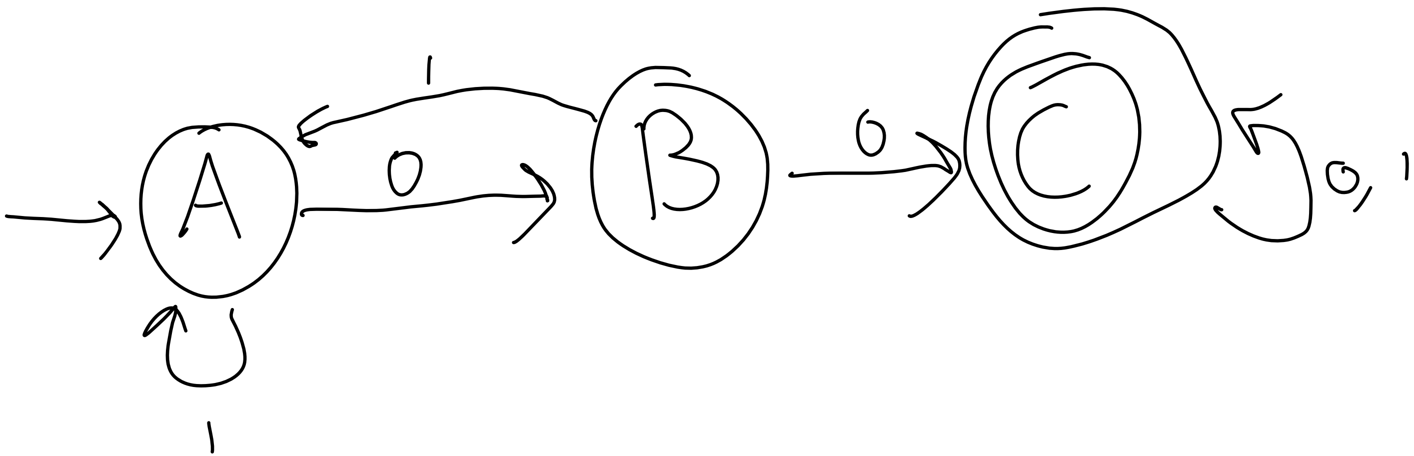
EXAMPLE – 3

- Design an FA M with L(M) = { w ∈ { 0,1 }* | w contains 101 as a substring }.



EXAMPLE – 4

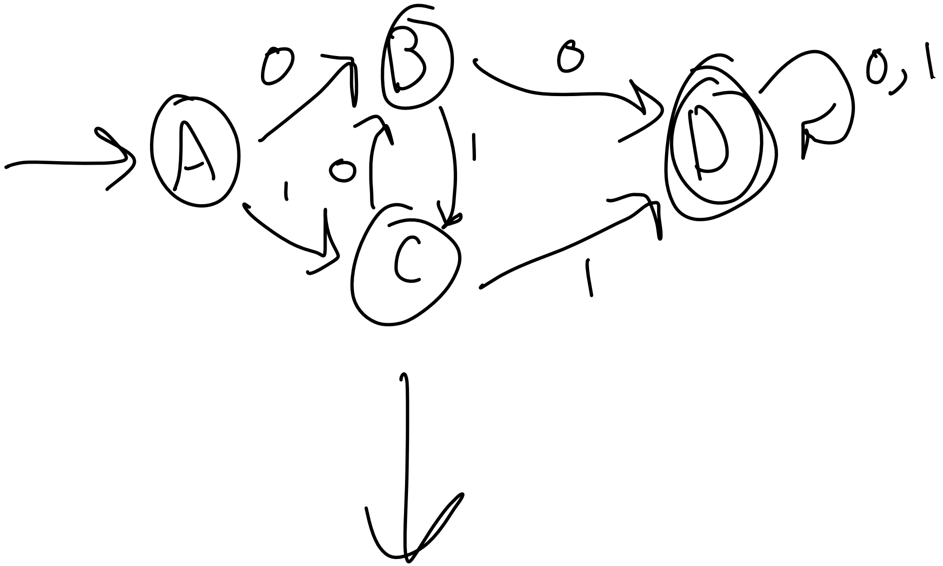
L = { w ∈ {0, 1}* | w contains 00 as a substring }



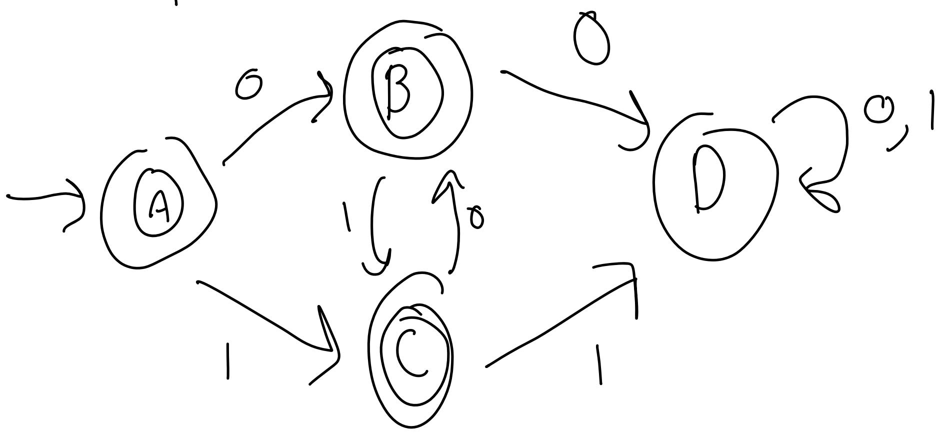
EXAMPLE – 5

- L = { w ∈ { 0,1 }* | w doesn't contain either 00 or 11 as a substring }.

Does contain either 00 or 11 as a substring



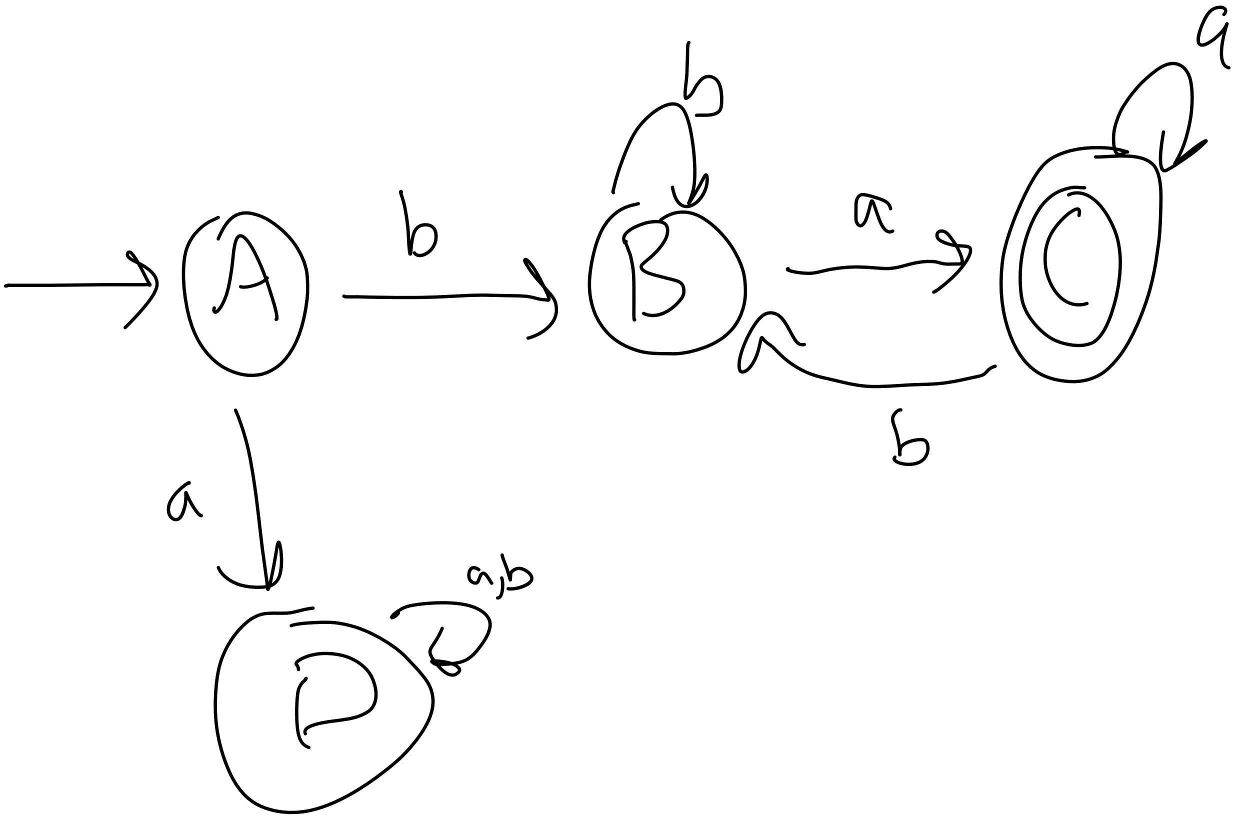
Flip final states



Now not accept those strings

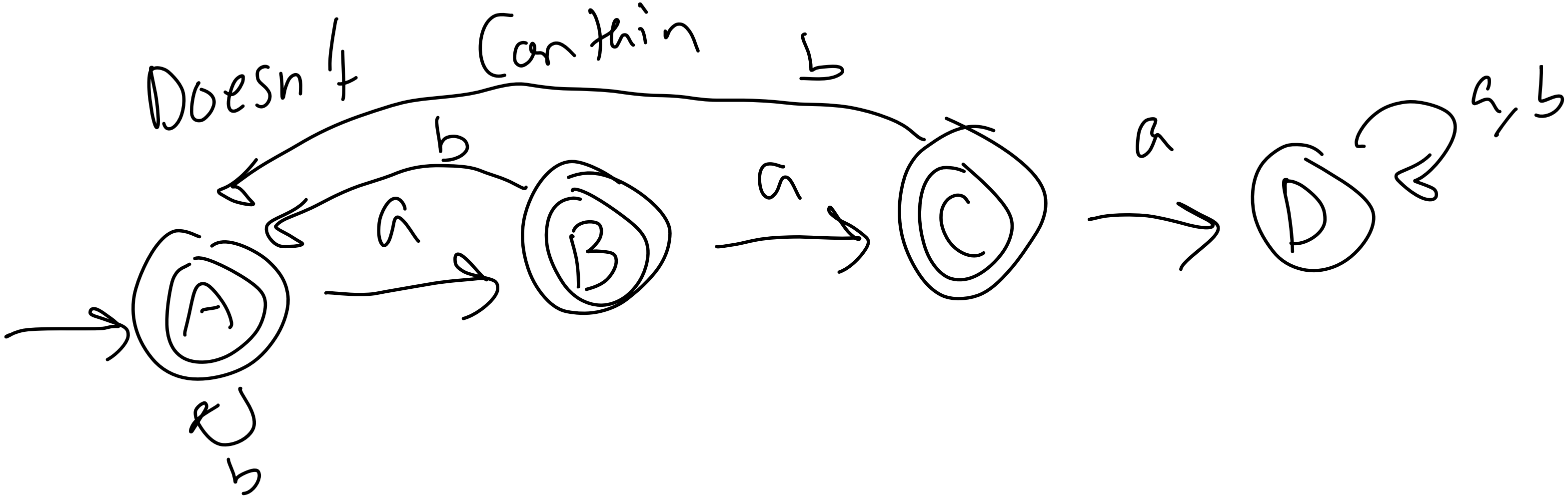
Example 6

For the following language over the alphabet S = {a, b}, give a DFA for it: "all strings that begin with b and end with a".



Example 7

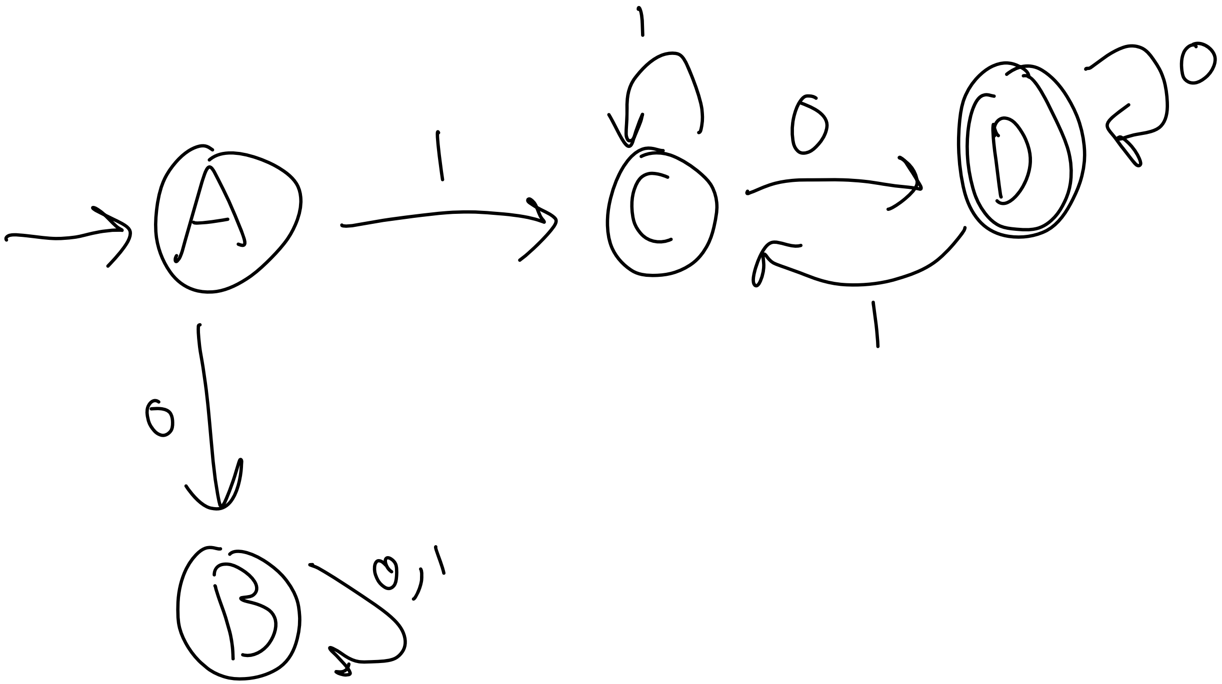
Design a DFA that accepts the the language consisting of the set of those strings over {a, b} that do not contain the substring “aaa”.



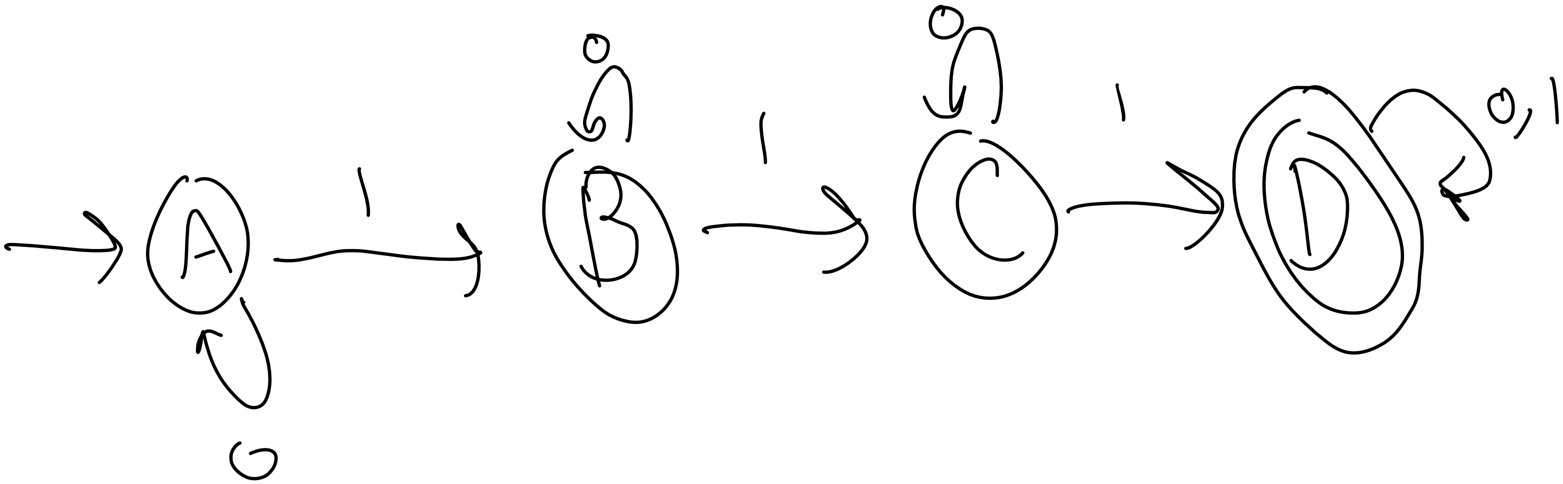
EXAMPLES TO TRY!

- {w| w begins with a 1 and ends with a 0}
- {w | w contains at least three 1s}
- Design FA to accept the string that always ends with 00.
- {w| w contains the substring 0101 (i.e., w = x0101y for some x and y)}
- Let Σ = {a, b}. Define A = {w ∈ Σ^* | w ends in "bba"}.
- Let Σ = {a, b}. Define A = {w ∈ Σ^* | w begins with 'a' and ends with 'b'}.
- Draw a DFA that recognizes the language over the alphabet {0, 1} consisting of all those strings that contain an odd number of 1's.
- Design a DFA which accepts strings with even number of 0's followed by single 1 over {0,1}.

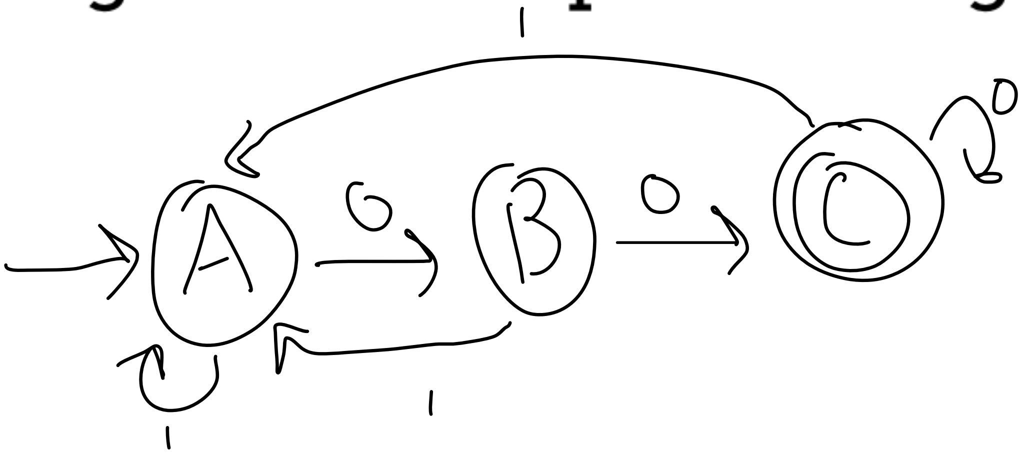
{w| w begins with a 1 and ends with a 0}



{w | w contains at least three 1s}



Design FA to accept the string that always ends with 00.



EXAMPLES TO TRY!

- Draw a DFA that recognizes the language over the alphabet {0, 1} consisting of all those strings that contain an odd number of 1's.
- Design a DFA which accepts strings with even number of 0's followed by single 1 over {0,1}.

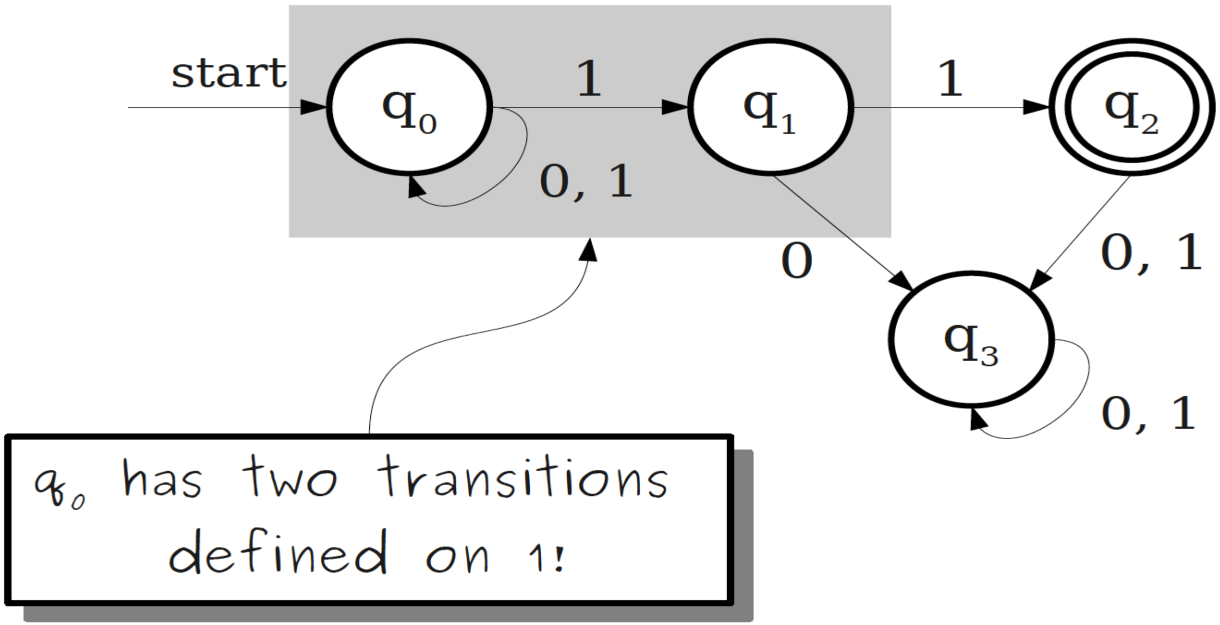
NFAs

- An NFA is a
  - **N**ondeterministic
  - **F**inite
  - **A**utomaton
- Conceptually like a DFA but equipped with the vast power of nondeterminism.
- Given the current state, there can be multiple next states
- The next state may be chosen at random or chosen in parallel
- Nondeterminism is a generalization of determinism, so *every deterministic finite automaton is automatically a nondeterministic finite automaton.*

(NON)DETERMINISM

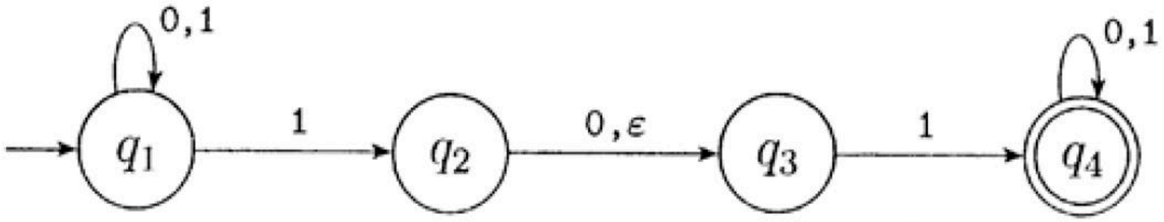
- A model is deterministic if at every point in the computation, there is exactly one choice that can make.
- The machine accepts if that series of choices leads to an accepting state.
- A model is nondeterministic if the computing machine may have multiple decisions that it can make at one point.
- The machine accepts if any series of choices leads to an accepting state.

SIMPLE EXAMPLE OF NFA

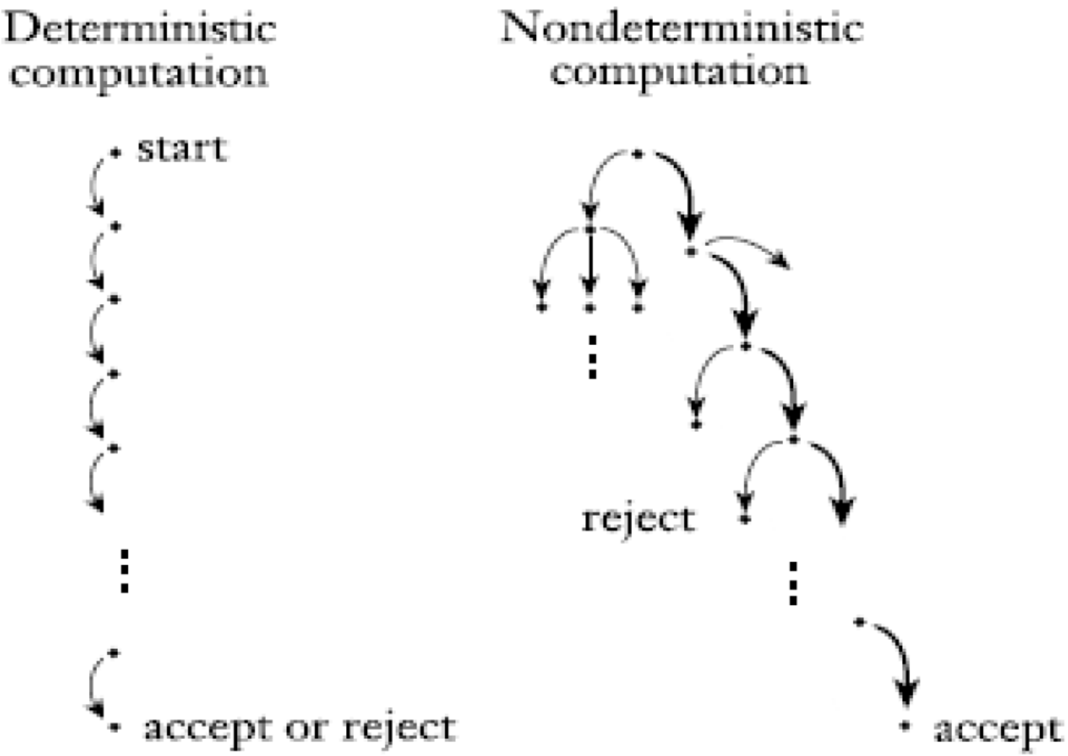


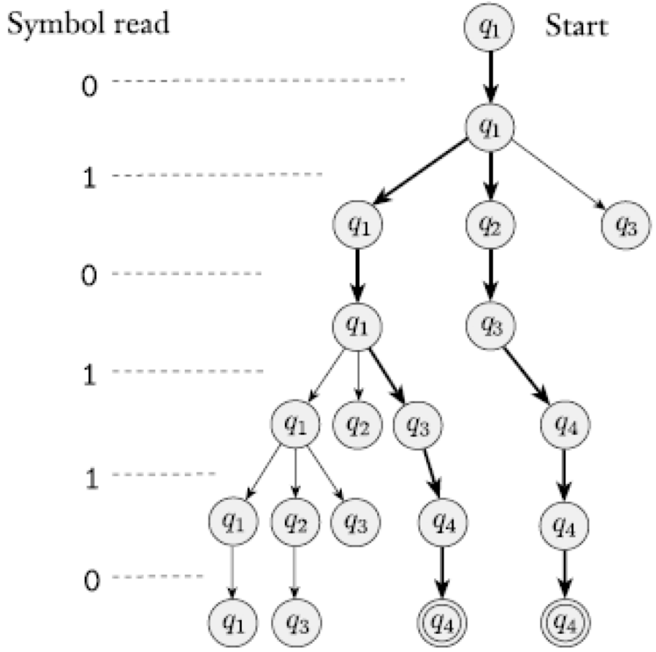
NFA

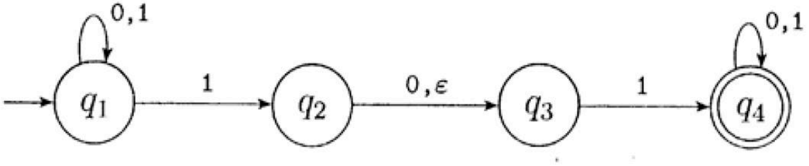
- The machine below violates the rule of unique state.
  - State q1 has one exiting arrow for 0, but it has two for 1;
  - q2 has one arrow for 0, but it has none for 1.
- *In an NFA, a <u>state may have zero, one, or many exiting arrows for each alphabet symbol.</u>*
- An NFA may have arrows labeled with members of the alphabet or ε. Zero, one, or many arrows may exit from each state with the label ε



HOW DOES AN NFA COMPUTE?







Input → 010110

FORMAL DEFINITION

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

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