Finite Automata (1.1)

COT 4210 Discrete Structures II

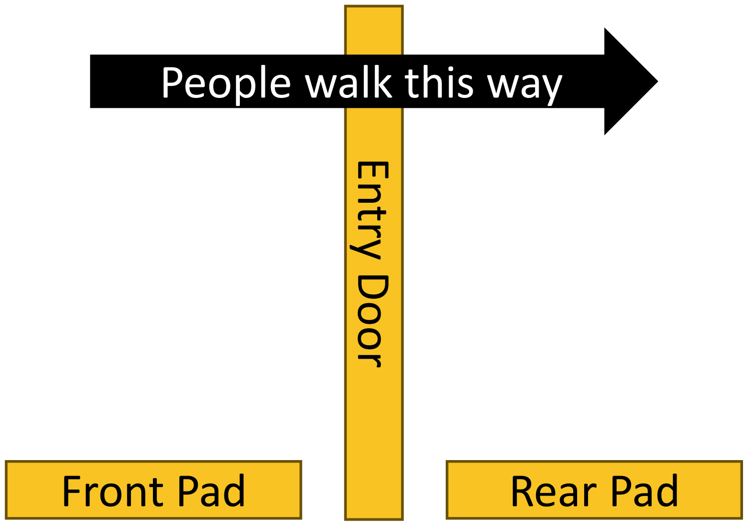
Summer 2025

Department of Computer Science

Dr. Steinberg

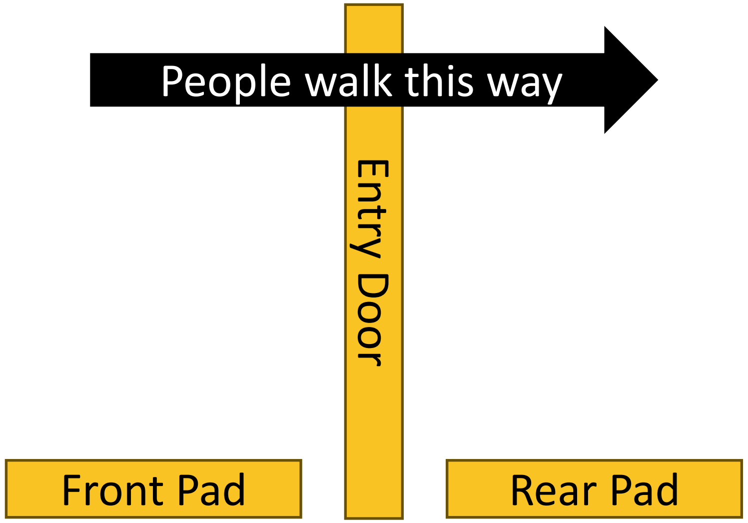
Introduction to Finite Automata

- Let’s start with the assumption that we have a computer limited memory (finite).
- With limited memory, what can we even do with that?
- Consider an automatic entry door.
  - When do we open and close the door?



Automatic Entry Door

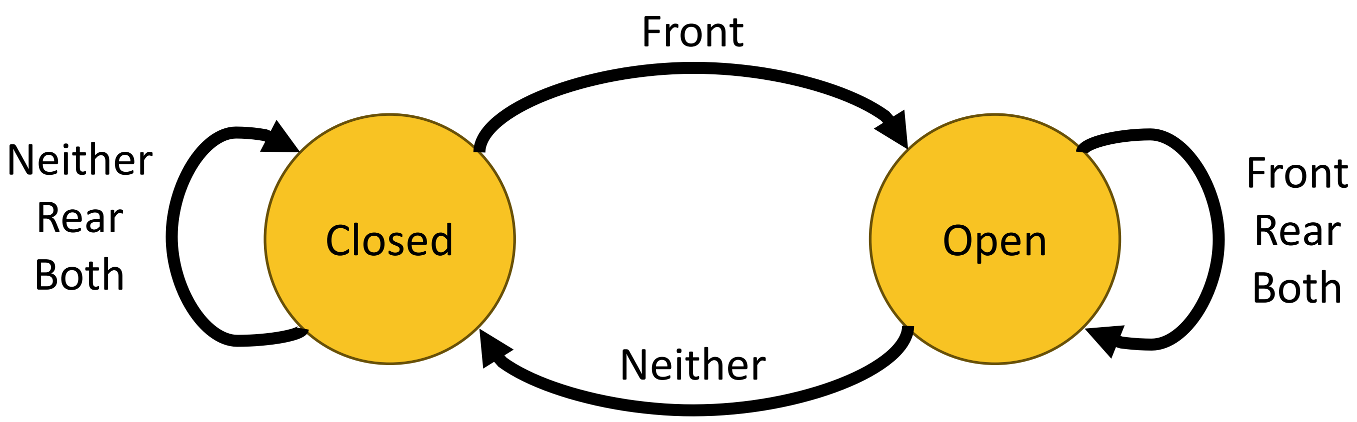
- When do we open and close the door?
- Four possible inputs: Outside, Inside, Both and Neither
- We ***open*** the door at Outside (only)
- We ***close*** the door at Neither
- Otherwise the door stays right where it is



	Neither	Front	Rear	Both
Closed	Closed	Open	Closed	Closed
Open	Closed	Open	Open	Open

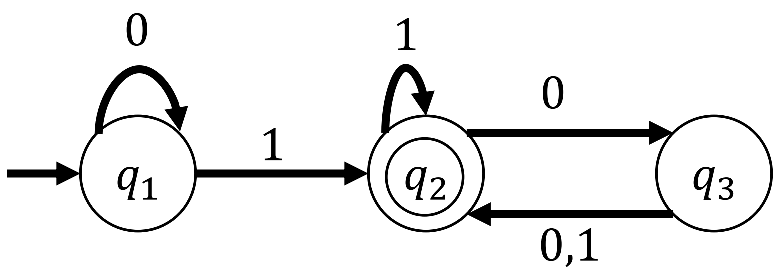
States and Transitions

- We can think of the door in terms of:
  - Its *states*: Open and Closed
  - Its *transitions*: When it opens and closes
- Two concise ways to depict these
  - A *state transition table*
  - A *state diagram*
  - Either may be better for a given machine
- We call logical constructs that we think of in these terms *automata*, or *machines*
  - Let’s make this less fuzzy
  - First, let’s remember strings



Another Machine (this we will see more of…)

- This machine can **read** strings over the binary alphabet
  - The incoming arrow on the left means q_1 is the **starting** state
  - The double circle around q_2 means it is an **accepting** state
  - This kind of machine, given any string over its alphabet, either **accepts** or **rejects** it
  - So, what strings will this machine accept?



Definition: Deterministic Finite Automata (DFA)

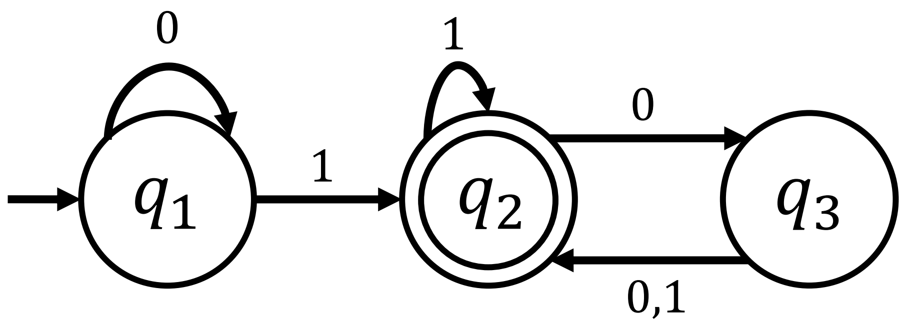
- A **deterministic finite automaton (DFA)** is a 5-tuple (Q, Σ, δ, q_0, F) that consists of:
  - Q — A finite set of *states*
  - Σ — An *alphabet*
  - $\delta: Q \times \Sigma \to Q$ — A *transition function*
  - $q_0 \in Q$ — A *start state*
  - $F \subseteq Q$ — A set of *accept* (or *final*) states

Example 1 (Sipser Example 1.6)

- $M_1 = (Q, \Sigma, \delta, q_1, F)$
  - $Q = \{q_1, q_2, q_3\}$
  - $\Sigma = \{0,1\}$
  - δ

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

  - q_1 is the start state
  - $F = \{q_2\}$

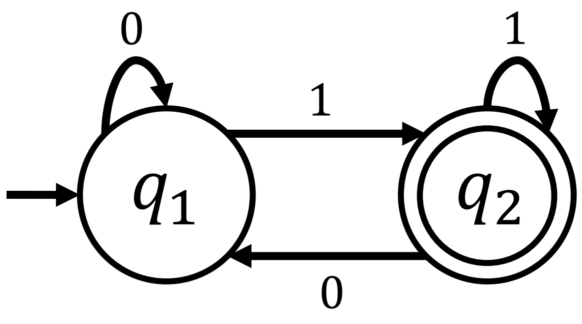


Example 2 (Sipser Example 1.7)

- $M_2 = (Q, \Sigma, \delta, q_1, F)$
  - $Q = \{q_1, q_2\}$
  - $\Sigma = \{0,1\}$
  - δ

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

  - q_1 is the start state
  - $F = \{q_2\}$

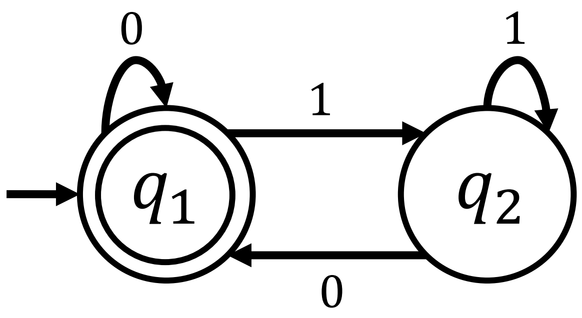


Example 3 (Sipser Example 1.9)

- $M_3 = (Q, \Sigma, \delta, q_1, F)$
  - $Q = \{q_1, q_2\}$
  - $\Sigma = \{0,1\}$
  - δ

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

  - q_1 is the start state
  - $F = \{q_1\}$



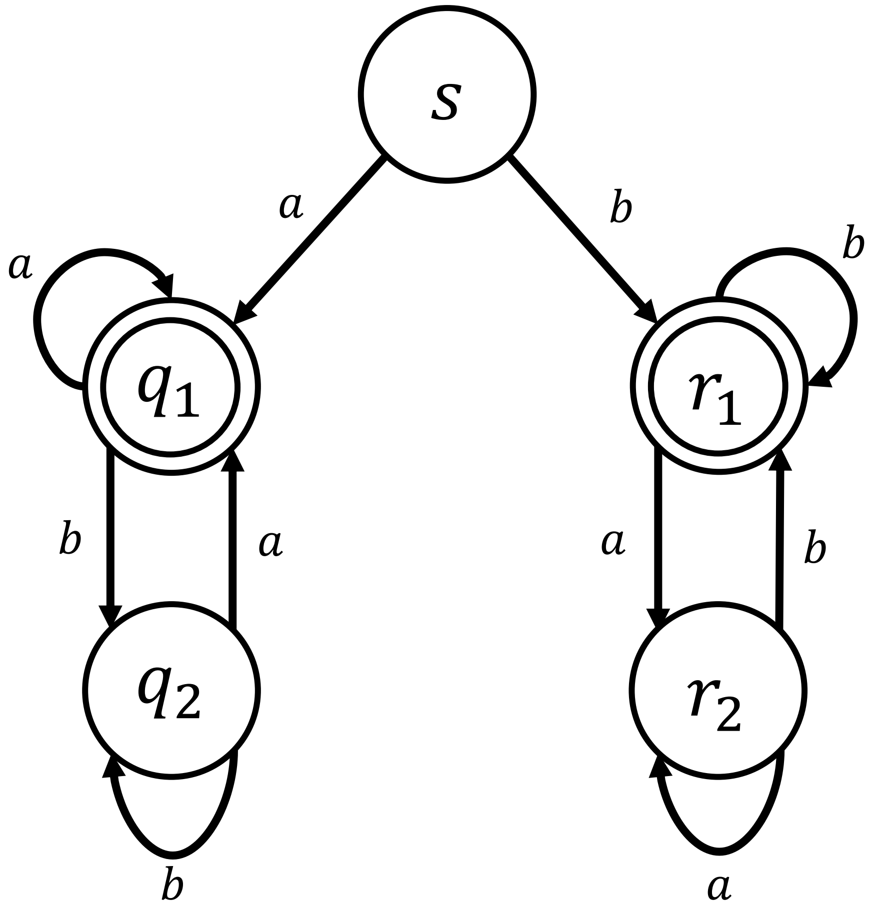
The difference between this example and the previous is the final state. The previous example had state q_2 as the final state while the example on this slide has q_1 as the final state.

Example 4 (1.11)

- $M_4 = (Q, \Sigma, \delta, q_1, F)$
  - $Q = \{q_1, r_1\}$
  - $\Sigma = \{a, b\}$
  - s is the start state
  - $F = \{q_1, r_1\}$

δ

	a	b
s	q_1	r_1
q_1	q_1	q_2
q_2	q_1	q_2
r_1	r_2	r_1
r_2	r_2	r_1



In this example, we see that this DFA has 2 final states!

Some Basic Definitions

Time to start building the big picture of what we are going to embark on!

Definition: Acceptance

- Let $M = (Q, \Sigma, \delta, q_0, F)$ and $w = w_1 w_2 \ldots w_n$ be a string of length n over Σ.
- M **accepts** w if there exists a sequence of states in Q $r_0, r_1, \ldots, r_n$ so that:

1. $r_0 = q_0$
2. For i from 0 to $n - 1$, $\delta(r_i, w_{i+1}) = r_{i+1}$
3. $r_n \in F$

Definition: Acceptance

- Let $M = (Q,$ … over Σ.
- M **accepts** … so that:

1. $r_0 = q_0$
2. For i fro…
3. $r_n \in F$

Guess What! The sequence of states describes the computation of a machine.

Definition: Acceptance (or *Computation*)

- Let $M = (Q, \Sigma, \delta, q_0, F)$ and $w = w_1 w_2 \ldots w_n$ be a string of length n over Σ.
- M **accepts** w if there exists a sequence of states in Q $r_0, r_1, \ldots, r_n$ so that:

1. $r_0 = q_0$
2. For i from 0 to $n - 1$, $\delta(r_i, w_{i+1}) = r_{i+1}$
3. $r_n \in F$

- **This sequence of states describes the computation of the machine.**

Definition: Recognition

A machine M **recognizes** language A if $A = \{w \mid M \text{ accepts } w\}$.

Definition: Regular Language

A language is **regular** if and only if it can be recognized by a DFA.

(Equivalently, language A is regular if there exists DFA M that recognizes it.)

Building DFAs

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- We know that are alphabet Σ contains the symbols 0 and 1.

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- We know that are alphabet Σ contains the symbols 0 and 1.
- From our basic definition of mathematics, we know that there exist an even number and an odd number since we are interested in counting the number of 1's. This means that there are TWO states. Note: If we know that we want to recognize an odd number of 1's, the odd state must be accepted (or final) state.

DFA Example 1 (Number of 1’s is odd)

We want do design some DFA that recognizes binary strings (consisting of 0’s and 1’s) where the number of 1’s **odd**.

- We know that are alphabet Σ
- From our basic definition of … n number and an odd number since we are interested in c… re are TWO states. Note: If we know that we want to rec… ust be accepted (or final) state.

Let’s draw this DFA!

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- $M_{odd} = (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}, 1) = q_{odd}$
  - $\delta(q_{even}, 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- $M_{odd} = (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}, 1) = q_{odd}$
  - $\delta(q_{even}, 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$

Let's Create the two states!

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- $M_{odd} = (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}, 1) = q_{odd}$
  - $\delta(q_{even}, 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- $M_{odd} = (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}, 1) = q_{odd}$
  - $\delta(q_{even}, 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$

Let's update odd state as the final state!

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- $M_{odd} \ = \ (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}\ , 1) = q_{odd}$
  - $\delta(q_{even}\ , 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$

DFA Example 1 (Number of 1’s is odd)

We want do design some DFA that recognizes binary strings (consisting of 0’s and 1’s) where the number of 1’s **odd**.

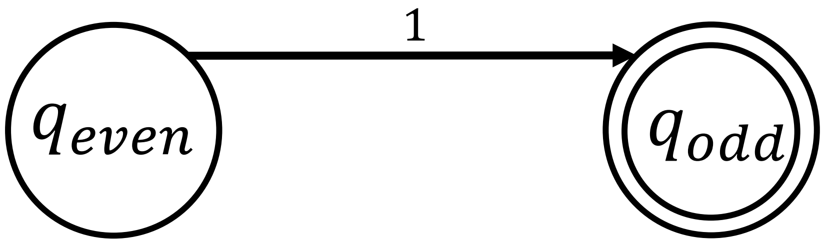
- $M_{odd} = (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}, 1) = q_{odd}$
  - $\delta(q_{even}, 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$

Let’s add the transitions!

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- $M_{odd} = (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}, 1) = q_{odd}$
  - $\delta(q_{even}, 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$

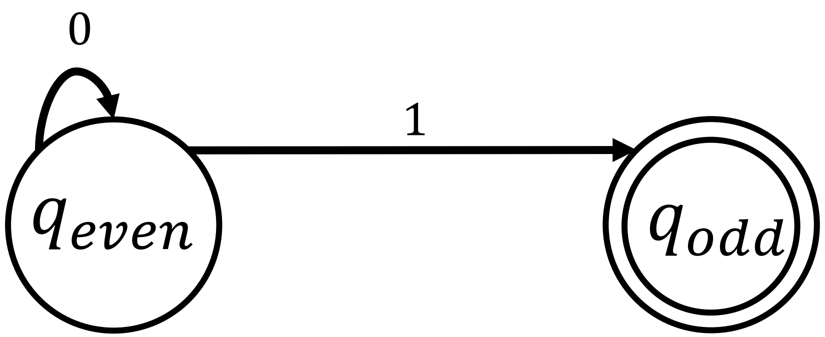


If we read the symbol 1 while we are in the q_{even} state, then we should be in the q_{odd} state. Note the 1st transition function!

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- $M_{odd} = (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}, 1) = q_{odd}$
  - $\delta(q_{even}, 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$

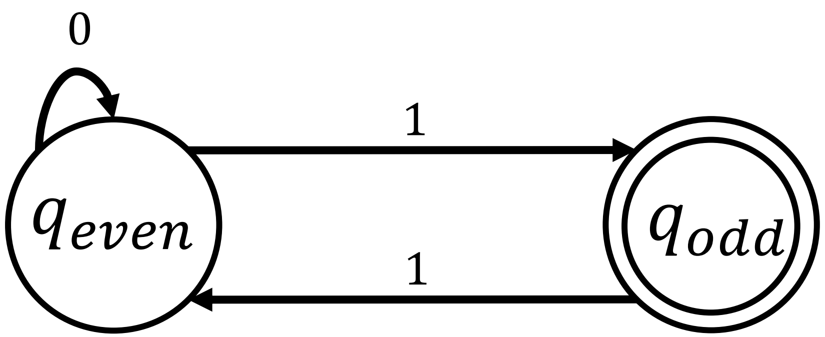


If we read the symbol 0 while we are in the q_{even} state, then we should be in the q_{even} state.
Note the 2nd transition function!

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- $M_{odd} = (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}, 1) = q_{odd}$
  - $\delta(q_{even}, 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$

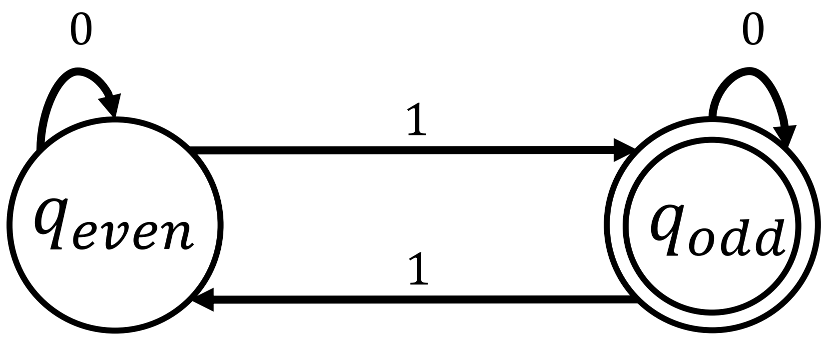


If we read the symbol 1 while we are in the q_{odd} state, then we should be in the q_{even} state.
Note the 3rd transition function!

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- $M_{odd} = (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}, 1) = q_{odd}$
  - $\delta(q_{even}, 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$

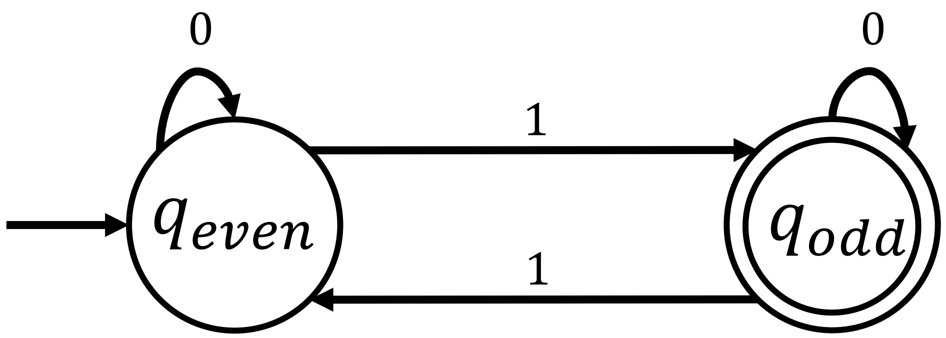


If we read the symbol 0 while we are in the q_{odd} state, then we should be in the q_{odd} state. Note the 4th transition function!

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- $M_{odd} = (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}, 1) = q_{odd}$
  - $\delta(q_{even}, 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$

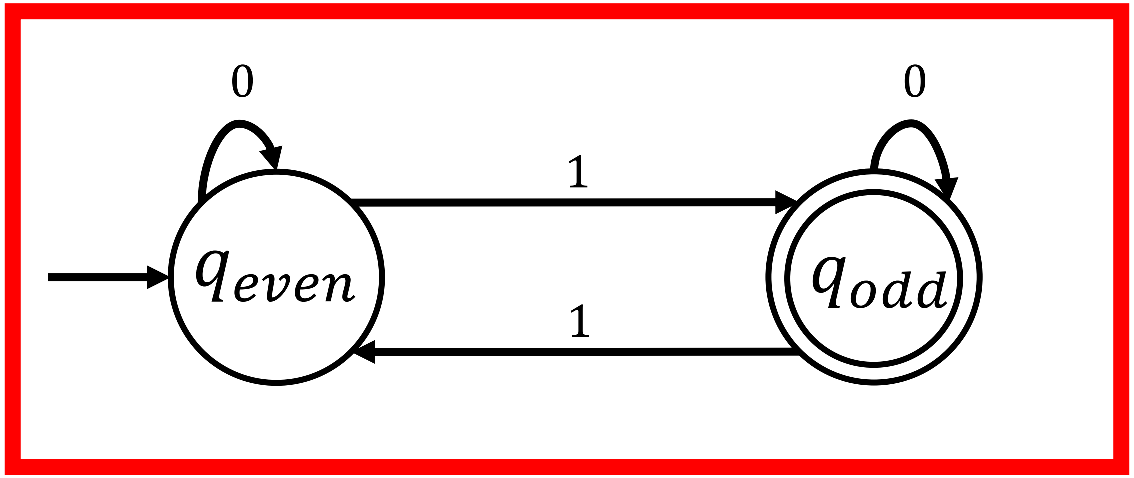


Now we need a starting state! The start state is the q_{even} state!

DFA Example 1 (Number of 1's is odd)

We want do design some DFA that recognizes binary strings (consisting of 0's and 1's) where the number of 1's **odd**.

- $M_{odd} = (Q, \Sigma, \delta, q_{even}, F)$
- $Q = \{q_{odd}, q_{even}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q_{even}, 1) = q_{odd}$
  - $\delta(q_{even}, 0) = q_{even}$
  - $\delta(q_{odd}, 1) = q_{even}$
  - $\delta(q_{odd}, 0) = q_{odd}$
- q_{even} is start state
- $F = \{q_{odd}\}$



Our DFA is complete!

DFA Example 2 (String w contains the substring “001”)

We want to design a DFA that recognizes a string of 0’s and 1’s and contains the substring “001”.

DFA Example 2 (String w contains the substring “001”)

We want to design a DFA that recognizes a string of 0’s and 1’s and contains the substring “001”.

- We know that are alphabet Σ contains the symbols 0 and 1.

DFA Example 2 (String w contains the substring “001”)

We want to design a DFA that recognizes a string of 0’s and 1’s and contains the substring “001”.

- We know that are alphabet Σ contains the symbols 0 and 1.
- We know that the most basic string that would be recognized is the string “001”. What we can do is build off this in designing our DFA.

DFA Example 2 (String w contains the substring “001”)

We want to design a DFA that recognizes a string of 0’s and 1’s and contains the substring “001”.

- We know that are alphabet Σ contains the symbols 0 and 1.
- We know that the most basic string that would be recognized is the string “001”. What we can do is build off this in designing our DFA.

Let’s draw this DFA!

DFA Example 2 (String w contains the substring "001")

We want to design a DFA that recognizes a string of 0's and 1's and contains the substring "001".

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q\,, 0) = q_0$
  - $\delta(q\,, 1) = q$
  - $\delta(q_0\,, 0) = q_{00}$
  - $\delta(q_0\,, 1) = q$
  - $\delta(q_{00}\,, 0) = q_{00}$
  - $\delta(q_{00}\,, 1) = q_{001}$
  - $\delta(q_{001}\,, 0) = q_{001}$
  - $\delta(q_{001}\,, 1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

What we can do is utilize each state to recognize each required symbol in the substring "001". This will require a total 4 states. The first state represents the first symbol in the substring has not been recognized.

DFA Example 2 (String w contains the substring "001")

We want to design a DFA that recognizes a string of 0's and 1's and contains the substring "001".

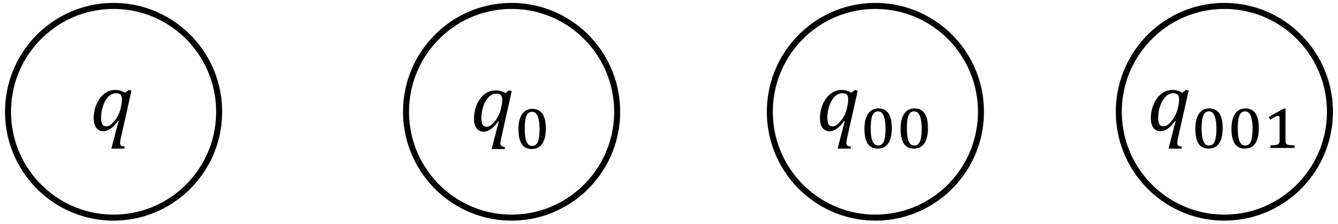
- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q\ ,0) = q_0$
  - $\delta(q\ ,1) = q$
  - $\delta(q_0\ ,0) = q_{00}$
  - $\delta(q_0\ ,1) = q$
  - $\delta(q_{00}\ ,0) = q_{00}$
  - $\delta(q_{00}\ ,1) = q_{001}$
  - $\delta(q_{001}\ ,0) = q_{001}$
  - $\delta(q_{001}\ ,1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

Let's Create the four states!

DFA Example 2 (String w contains the substring “001”)

We want to design a DFA that recognizes a string of 0’s and 1’s and contains the substring “001”.

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q\ , 0) = q_0$
  - $\delta(q\ , 1) = q$
  - $\delta(q_0\ , 0) = q_{00}$
  - $\delta(q_0\ , 1) = q$
  - $\delta(q_{00}\ , 0) = q_{00}$
  - $\delta(q_{00}\ , 1) = q_{001}$
  - $\delta(q_{001}\ , 0) = q_{001}$
  - $\delta(q_{001}\ , 1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

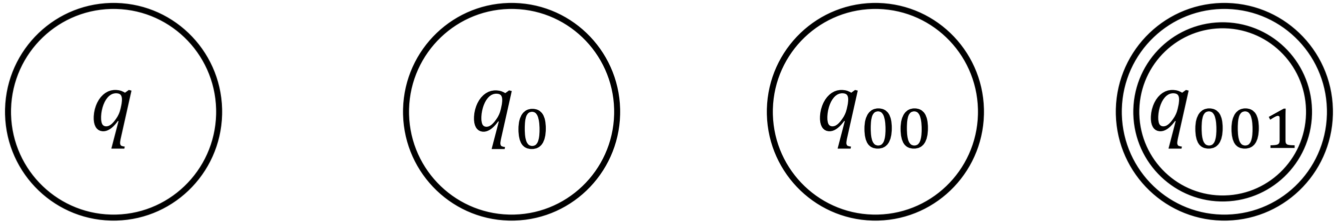


We will use the notation of each required symbol for the substring in 3 of our 4 states! Example: q_{00} means that substring 00 has been recognized.

DFA Example 2 (String w contains the substring "001")

We want to design a DFA that recognizes a string of 0's and 1's and contains the substring "001".

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q, 0) = q_0$
  - $\delta(q, 1) = q$
  - $\delta(q_0, 0) = q_{00}$
  - $\delta(q_0, 1) = q$
  - $\delta(q_{00}, 0) = q_{00}$
  - $\delta(q_{00}, 1) = q_{001}$
  - $\delta(q_{001}, 0) = q_{001}$
  - $\delta(q_{001}, 1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

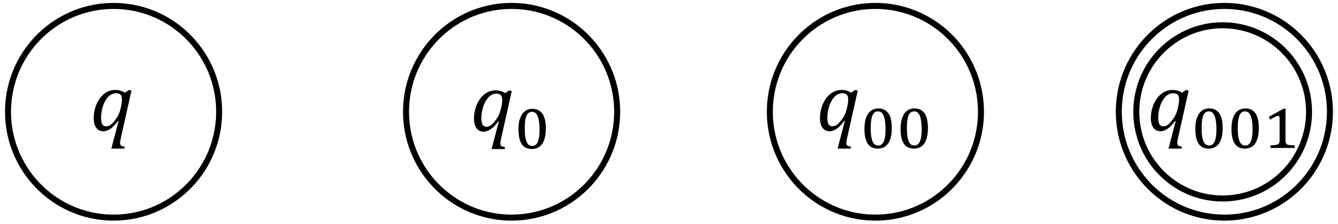


Let's update state q_{001} as our final state!

DFA Example 2 (String w contains the substring “001”)

We want to design a DFA that recognizes a string of 0’s and 1’s and contains the substring “001”.

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q\ ,0) = q_0$
  - $\delta(q\ ,1) = q$
  - $\delta(q_0\ ,0) = q_{00}$
  - $\delta(q_0\ ,1) = q$
  - $\delta(q_{00}\ ,0) = q_{00}$
  - $\delta(q_{00}\ ,1) = q_{001}$
  - $\delta(q_{001}\ ,0) = q_{001}$
  - $\delta(q_{001}\ ,1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

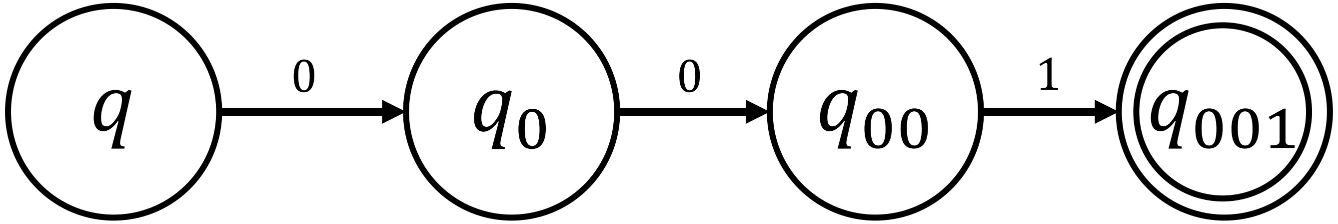


Let’s add the transitions!

DFA Example 2 (String w contains the substring “001”)

We want to design a DFA that recognizes a string of 0’s and 1’s and contains the substring “001”.

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q\,, 0) = q_0$
  - $\delta(q\,, 1) = q$
  - $\delta(q_0\,, 0) = q_{00}$
  - $\delta(q_0\,, 1) = q$
  - $\delta(q_{00}\,, 0) = q_{00}$
  - $\delta(q_{00}\,, 1) = q_{001}$
  - $\delta(q_{001}\,, 0) = q_{001}$
  - $\delta(q_{001}\,, 1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

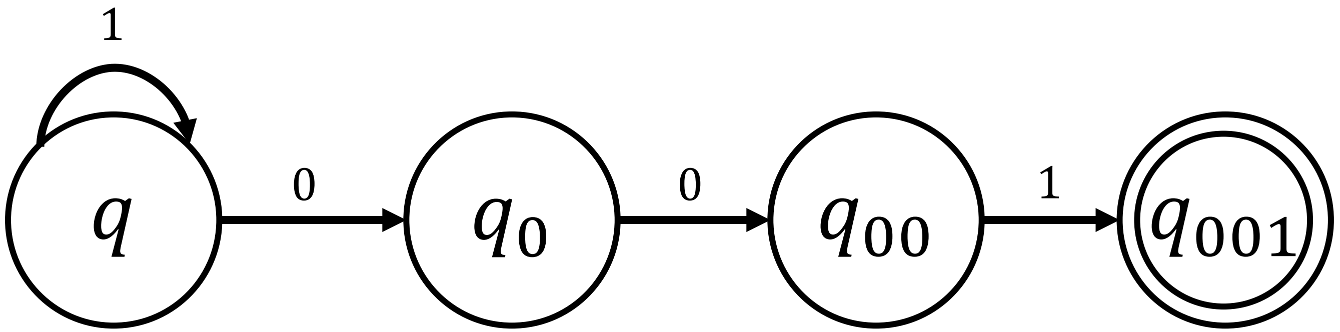


We can first start with the transitions that allows us to reach our basic string of “001” that was stated previously.

DFA Example 2 (String w contains the substring "001")

We want to design a DFA that recognizes a string of 0's and 1's and contains the substring "001".

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q, 0) = q_0$
  - $\delta(q, 1) = q$
  - $\delta(q_0, 0) = q_{00}$
  - $\delta(q_0, 1) = q$
  - $\delta(q_{00}, 0) = q_{00}$
  - $\delta(q_{00}, 1) = q_{001}$
  - $\delta(q_{001}, 0) = q_{001}$
  - $\delta(q_{001}, 1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

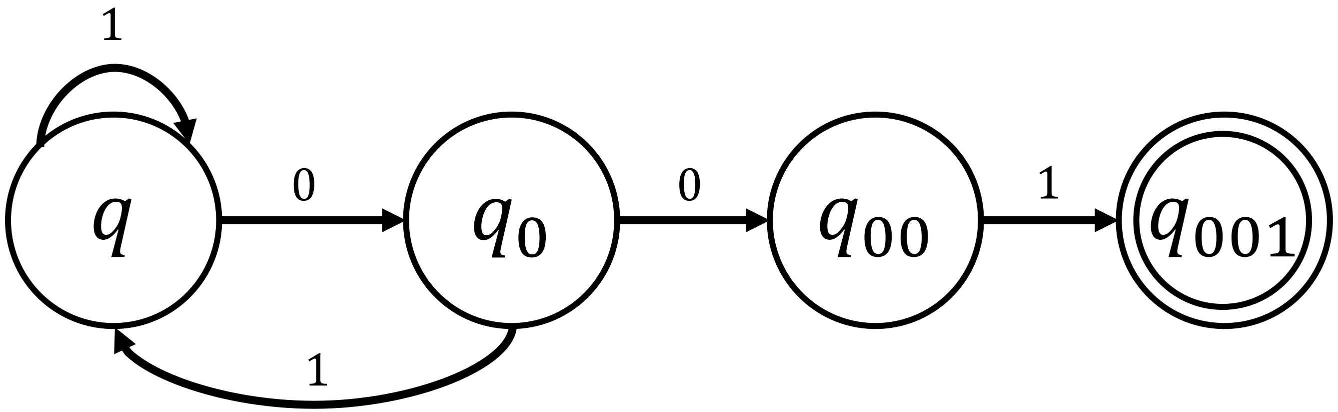


Let's start with state q. We already have the transition if symbol 0 is read. However, what happens if symbol 1 is read? There are strings such as "1001", "111001", "1111111111111111111111111001", etc… that can be accepted since they contain the required substring. That means we need a self cycle in state q when the symbol 1 is read.

DFA Example 2 (String w contains the substring "001")

We want to design a DFA that recognizes a string of 0's and 1's and contains the substring "001".

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q\,,0) = q_0$
  - $\delta(q\,,1) = q$
  - $\delta(q_0\,,0) = q_{00}$
  - $\delta(q_0\,,1) = q$
  - $\delta(q_{00}\,,0) = q_{00}$
  - $\delta(q_{00}\,,1) = q_{001}$
  - $\delta(q_{001}\,,0) = q_{001}$
  - $\delta(q_{001}\,,1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

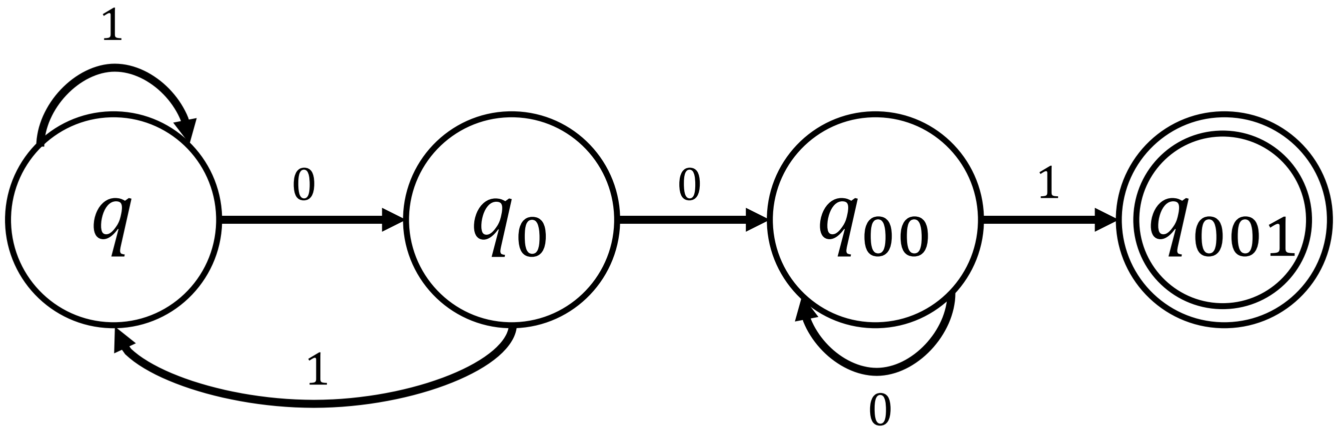


Let's now move to state q_0. We already have the transition if symbol 0 is read. However, what happens if symbol 1 is read? If a 1 is read that breaks the potential substring "001" and would require us to start all over! For example, 01001 would be accepted. What we can do is create a transition that takes us from q_0 to q when a 1 is read.

DFA Example 2 (String w contains the substring "001")

We want to design a DFA that recognizes a string of 0's and 1's and contains the substring "001".

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q\ ,0) = q_0$
  - $\delta(q\ ,1) = q$
  - $\delta(q_0\ ,0) = q_{00}$
  - $\delta(q_0\ ,1) = q$
  - $\delta(q_{00}\ ,0) = q_{00}$
  - $\delta(q_{00}\ ,1) = q_{001}$
  - $\delta(q_{001}\ ,0) = q_{001}$
  - $\delta(q_{001}\ ,1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

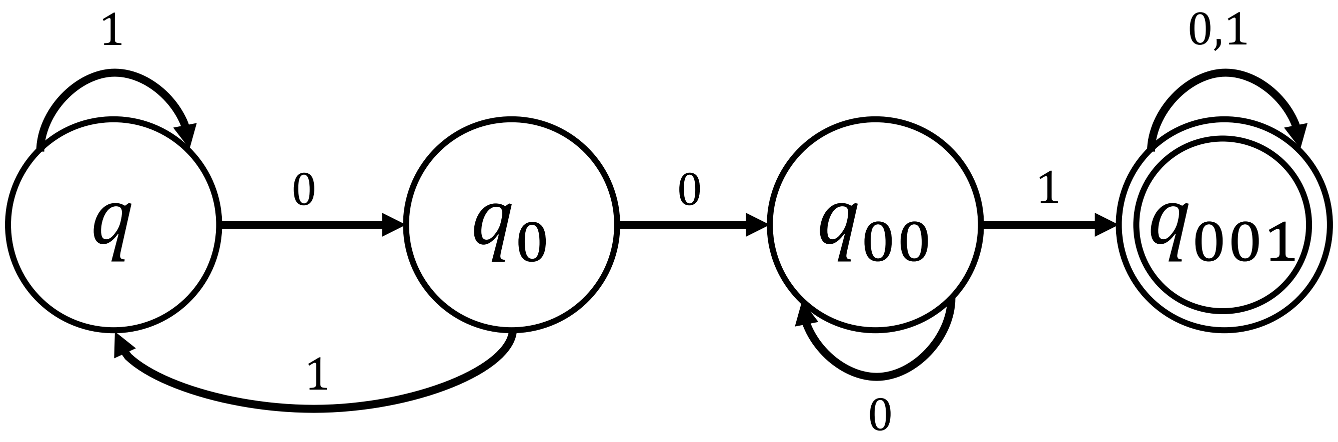


Let's now move to state q_{00}. We already have the transition if symbol 1 is read. However, what happens if symbol 0 is read? You might think intuitively, we would have to start over, but that is not the case! No matter how many 0's are read, we still have ideally the first two symbols of our substring. Example, "0000000001" would be accepted. That means we need a self cycle in state q_{00} when the symbol 0 is read.

DFA Example 2 (String w contains the substring “001”)

We want to design a DFA that recognizes a string of 0’s and 1’s and contains the substring “001”.

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q\,, 0) = q_0$
  - $\delta(q\,, 1) = q$
  - $\delta(q_0\,, 0) = q_{00}$
  - $\delta(q_0\,, 1) = q$
  - $\delta(q_{00}\,, 0) = q_{00}$
  - $\delta(q_{00}\,, 1) = q_{001}$
  - $\delta(q_{001}\,, 0) = q_{001}$
  - $\delta(q_{001}\,, 1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

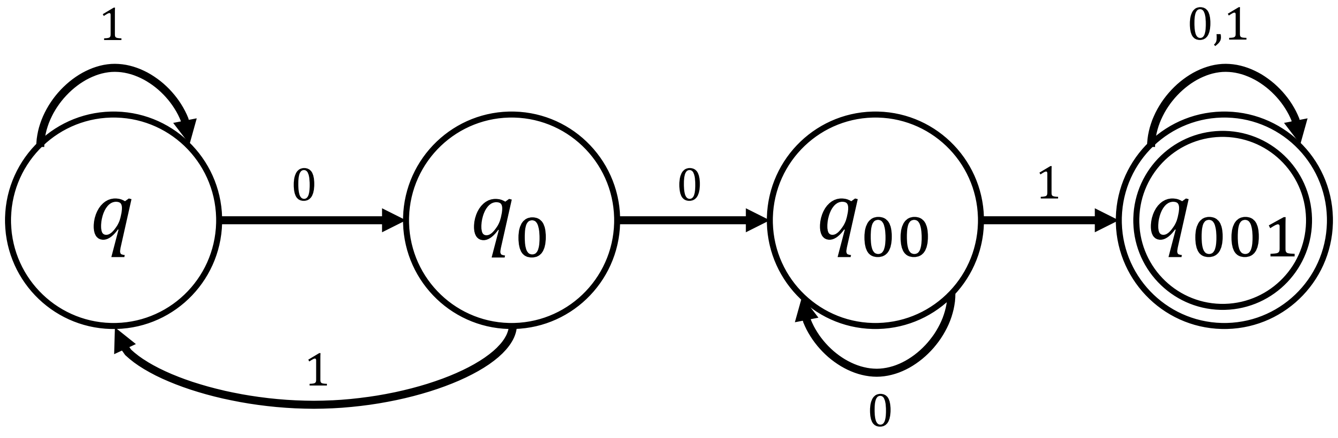


Let’s now move to state q_{001}. At this point in the processing, we have already read the substring “001”, which means no matter what symbols are read between 0 and 1, the string would be accepted. That means we just need a self cycle in state q_{001} for the symbols 0 and 1.

DFA Example 2 (String w contains the substring "001")

We want to design a DFA that recognizes a string of 0's and 1's and contains the substring "001".

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q\,,0) = q_0$
  - $\delta(q\,,1) = q$
  - $\delta(q_0\,,0) = q_{00}$
  - $\delta(q_0\,,1) = q$
  - $\delta(q_{00}\,,0) = q_{00}$
  - $\delta(q_{00}\,,1) = q_{001}$
  - $\delta(q_{001}\,,0) = q_{001}$
  - $\delta(q_{001}\,,1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

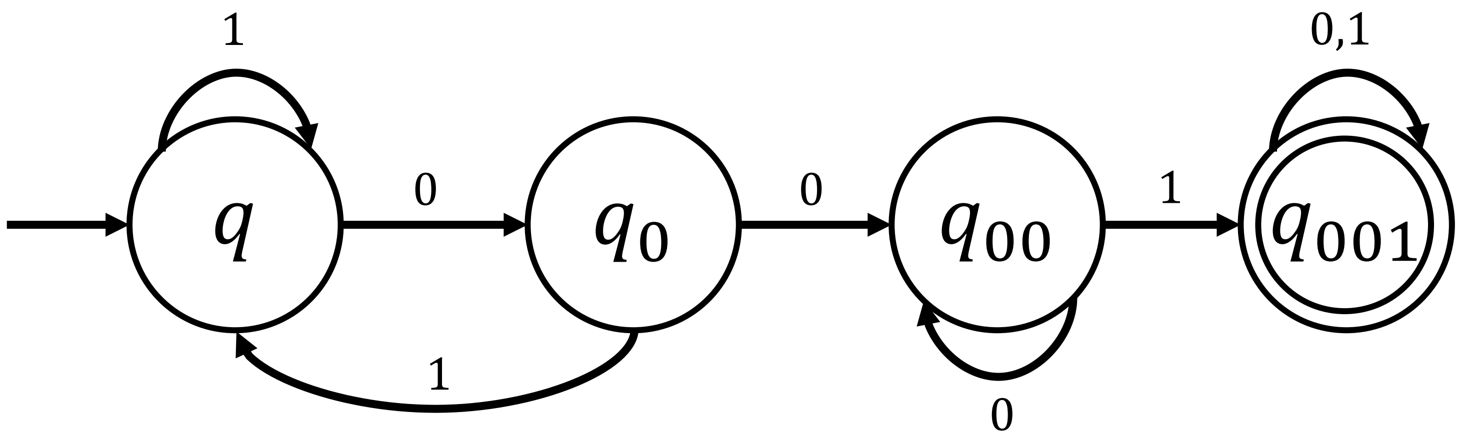


Let's now move to state q_{001}. At this point in the processing, we have already read the substring "001", which means no matter what symbols are read between 0 and 1, the string would be accepted. That means we just need a self cycle in state q_{001} for the symbols 0 and 1.

DFA Example 2 (String w contains the substring "001")

We want to design a DFA that recognizes a string of 0's and 1's and contains the substring "001".

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q, 0) = q_0$
  - $\delta(q, 1) = q$
  - $\delta(q_0, 0) = q_{00}$
  - $\delta(q_0, 1) = q$
  - $\delta(q_{00}, 0) = q_{00}$
  - $\delta(q_{00}, 1) = q_{001}$
  - $\delta(q_{001}, 0) = q_{001}$
  - $\delta(q_{001}, 1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$

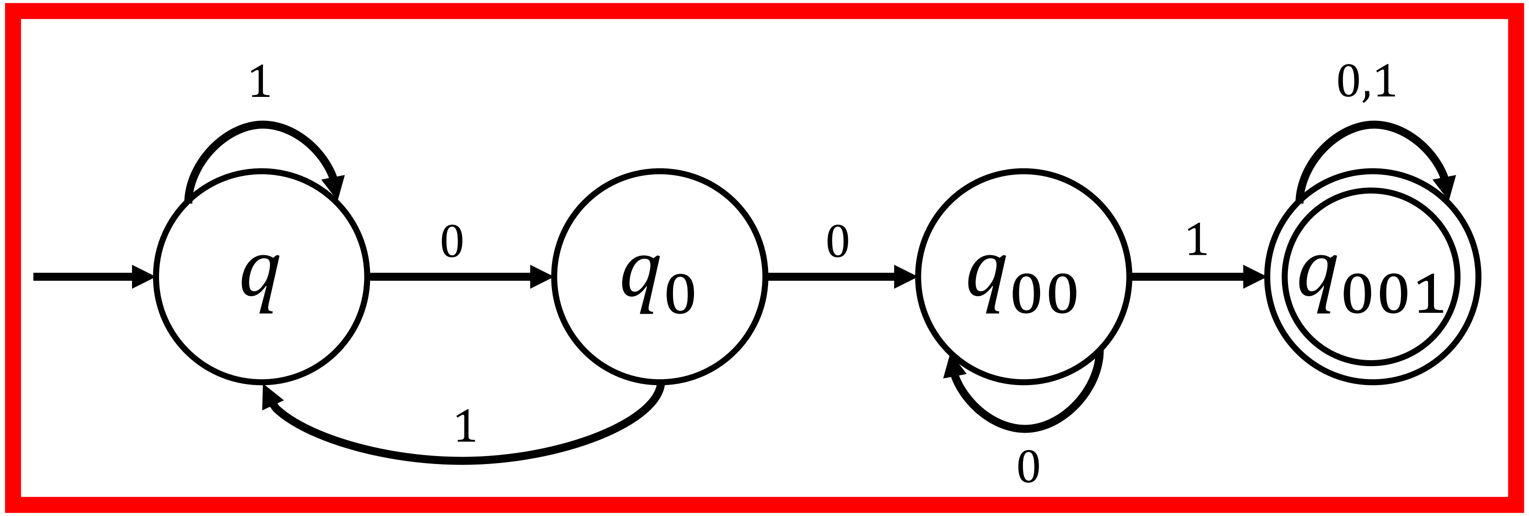


Now we need the starting state!
The start state is the q state!

DFA Example 2 (String w contains the substring “001”)

We want to design a DFA that recognizes a string of 0’s and 1’s and contains the substring “001”.

- $M_2 = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q, q_0, q_{00}, q_{001}\}$
- $\Sigma = \{0,1\}$
- δ (transition functions)
  - $\delta(q\,,0) = q_0$
  - $\delta(q\,,1) = q$
  - $\delta(q_0\,,0) = q_{00}$
  - $\delta(q_0\,,1) = q$
  - $\delta(q_{00}\,,0) = q_{00}$
  - $\delta(q_{00}\,,1) = q_{001}$
  - $\delta(q_{001}\,,0) = q_{001}$
  - $\delta(q_{001}\,,1) = q_{001}$
- q is start state
- $F = \{q_{001}\}$



Our DFA is complete!

Let’s practice designing more DFAs on the board!

Extra Examples

The Regular Operations

Let *A* and *B* be languages. We define **union**, **concatenation** and **star** (or **Kleene Closure**) as:

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$$

$$A \circ B \text{ or } AB = \{xy \mid x \in A \text{ and } y \in B\}$$

$$A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$$

Regular Languages: Union Closure

- We want to prove that the class of regular languages is **closed** under the regular operations – that performing those operations on regular languages results in regular languages.
- Let's start with union – and for that, let's go to the board…

Back from the Board

- We just proved that regular languages are closed under union.
- It was awful.
- There's got to be an easier way to do this…
- …and there is, but it's going to require some carefully crafted nonsense.

UCF

Acknowledgement

Some Notes and content come from Dr. Gerber, Dr. Hughes, and Mr. Guha's COT4210 class and the Sipser Textbook, *Introduction to the Theory of Computation*, 3rd ed., 2013