Foundations of Computing

Lecture 3

Arkady Yerukhimovich

January 21, 2025

Outline

Lecture 2 Review

- Language decided by DFA M
- Building DFAs
- Proving Correctness of DFAs

Outline

1. Lecture 2 Review
2. Regular Languages
3. Non-deterministic Finite Automata (NFA)
4. Example NFAs

From Machines to Languages

- Last lecture we saw how to build DFA M to decide a language L
- Learned to reason about machine M
- Recall that each machine M decides one language $L(M)$

From Machines to Languages

- Last lecture we saw how to build DFA M to decide a language L
- Learned to reason about machine M
- Recall that each machine M decides one language $L(M)$

Let's switch our perspective

Instead of reasoning about machines, let's focus on languages decided by those machines.

Regular Language

Definition

A language L is regular if it is decided by a DFA.

Regular Language

Definition

A language L is regular if it is decided by a DFA.

Observations:

- All languages we have seen thus far are regular
- To prove that a language is regular just need to show a DFA that decides it
- We will prove that regular languages correspond to regular expressions

Regular Language

Definition

A language L is regular if it is decided by a DFA.

Observations:

- All languages we have seen thus far are regular
- To prove that a language is regular just need to show a DFA that decides it
- We will prove that regular languages correspond to regular expressions

Something to think about

Are all languages regular?

Properties of Regular Languages

Closure under Complement

If L is a regular language, then $\overline{L}$ is also regular

$\overline{L}$ is the language that consists of all strings (in alphabet Σ) not in L.

Properties of Regular Languages

Closure under Complement

If L is a regular language, then $\overline{L}$ is also regular

$\overline{L}$ is the language that consists of all strings (in alphabet Σ) not in L.

Intuition: Swap the accept and reject states

Properties of Regular Languages

Closure under Complement

If L is a regular language, then $\overline{L}$ is also regular

Proof: Let $M = (Q, \Sigma, \delta, q, F)$ decide L

Construct $M' = (Q', \Sigma', \delta', q', F')$ that decides $\overline{L}$

Properties of Regular Languages

Closure under Complement

If L is a regular language, then $\overline{L}$ is also regular

Proof: Let $M = (Q, \Sigma, \delta, q, F)$ decide L

Construct $M' = (Q', \Sigma', \delta', q', F')$ that decides $\overline{L}$

1. $Q' = Q$

Properties of Regular Languages

Closure under Complement

If L is a regular language, then $\overline{L}$ is also regular

Proof: Let $M = (Q, \Sigma, \delta, q, F)$ decide L

Construct $M' = (Q', \Sigma', \delta', q', F')$ that decides $\overline{L}$

1. $Q' = Q$
2. $\Sigma' = \Sigma$

Properties of Regular Languages

Closure under Complement

If L is a regular language, then $\overline{L}$ is also regular

Proof: Let $M = (Q, \Sigma, \delta, q, F)$ decide L

Construct $M' = (Q', \Sigma', \delta', q', F')$ that decides $\overline{L}$

1. $Q' = Q$
2. $\Sigma' = \Sigma$
3. $\delta' = \delta$

Closure under Complement

If L is a regular language, then $\overline{L}$ is also regular

Proof: Let $M = (Q, \Sigma, \delta, q, F)$ decide L

Construct $M' = (Q', \Sigma', \delta', q', F')$ that decides $\overline{L}$

1. $Q' = Q$
2. $\Sigma' = \Sigma$
3. $\delta' = \delta$
4. $q' = q$

Properties of Regular Languages

Closure under Complement

If L is a regular language, then $\overline{L}$ is also regular

Proof: Let $M = (Q, \Sigma, \delta, q, F)$ decide L

Construct $M' = (Q', \Sigma', \delta', q', F')$ that decides $\overline{L}$

1. $Q' = Q$
2. $\Sigma' = \Sigma$
3. $\delta' = \delta$
4. $q' = q$
5. $F' = Q \setminus F$

Properties of Regular Languages

Closure under Complement

If L is a regular language, then $\overline{L}$ is also regular

Proof: Let $M = (Q, \Sigma, \delta, q, F)$ decide L

Construct $M' = (Q', \Sigma', \delta', q', F')$ that decides $\overline{L}$

1. $Q' = Q$
2. $\Sigma' = \Sigma$
3. $\delta' = \delta$
4. $q' = q$
5. $F' = Q \setminus F$

Observe:

- If $w \in L \iff w \notin \overline{L}$, then $M(w)$ stops in some $q \in F$, so $q \notin (Q \setminus F)$

Properties of Regular Languages

Closure under Complement

If L is a regular language, then $\overline{L}$ is also regular

Proof: Let $M = (Q, \Sigma, \delta, q, F)$ decide L

Construct $M' = (Q', \Sigma', \delta', q', F')$ that decides $\overline{L}$

1. $Q' = Q$
2. $\Sigma' = \Sigma$
3. $\delta' = \delta$
4. $q' = q$
5. $F' = Q \setminus F$

Observe:

- If $w \in L \iff w \notin \overline{L}$, then $M(w)$ stops in some $q \in F$, so $q \notin (Q \setminus F)$
- If $w \notin L \iff w \in \overline{L}$, then $M(w)$ stops in some $q \notin F$, so $q \in (Q \setminus F)$

Properties of Regular Languages

Closure Under Union

If L_1 and L_2 are both regular languages then $L_1 \cup L_2$ is also regular

$L_1 \cup L_2$ is the language consisting of all strings either in L_1 or L_2

Properties of Regular Languages

Closure Under Union

If L_1 and L_2 are both regular languages then $L_1 \cup L_2$ is also regular

$L_1 \cup L_2$ is the language consisting of all strings either in L_1 or L_2

Intuition: Run both machines in parallel and accept if either of them stops in an accept state

Properties of Regular Languages

Closure Under Union

If L_1 and L_2 are both regular languages then $L_1 \cup L_2$ is also regular

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize L_1, and $M_2 = ((Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize L_2

Construct $M = (Q, \Sigma, \delta, q, F)$ that recognizes $L_1 \cup L_2$

Properties of Regular Languages

Closure Under Union

If L_1 and L_2 are both regular languages then $L_1 \cup L_2$ is also regular

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize L_1, and $M_2 = ((Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize L_2

Construct $M = (Q, \Sigma, \delta, q, F)$ that recognizes $L_1 \cup L_2$

1. $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$

Properties of Regular Languages

Closure Under Union

If L_1 and L_2 are both regular languages then $L_1 \cup L_2$ is also regular

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize L_1, and $M_2 = ((Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize L_2

Construct $M = (Q, \Sigma, \delta, q, F)$ that recognizes $L_1 \cup L_2$

1. $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
2. $\Sigma = \Sigma$

Properties of Regular Languages

Closure Under Union

If L_1 and L_2 are both regular languages then $L_1 \cup L_2$ is also regular

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize L_1, and $M_2 = ((Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize L_2

Construct $M = (Q, \Sigma, \delta, q, F)$ that recognizes $L_1 \cup L_2$

1. $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
2. $\Sigma = \Sigma$
3. δ is as follows. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$

Properties of Regular Languages

Closure Under Union

If L_1 and L_2 are both regular languages then $L_1 \cup L_2$ is also regular

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize L_1, and $M_2 = ((Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize L_2

Construct $M = (Q, \Sigma, \delta, q, F)$ that recognizes $L_1 \cup L_2$

1. $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
2. $\Sigma = \Sigma$
3. δ is as follows. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$

4. $q_0 = (q_1, q_2)$

Properties of Regular Languages

Closure Under Union

If L_1 and L_2 are both regular languages then $L_1 \cup L_2$ is also regular

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize L_1, and $M_2 = ((Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize L_2

Construct $M = (Q, \Sigma, \delta, q, F)$ that recognizes $L_1 \cup L_2$

1. $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
2. $\Sigma = \Sigma$
3. δ is as follows. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$

4. $q_0 = (q_1, q_2)$
5. $F = \{(r_1, r_2) | r_1 \in F_1 \text{ or } r_2 \in F_2\}$

Closure Under Intersection

If L_1 and L_2 are both regular languages then $L_1 \cap L_2$ is also regular

$L_1 \cap L_2$ is the language consisting of all strings in both L_1 and L_2

Properties of Regular Languages

Closure Under Intersection

If L_1 and L_2 are both regular languages then $L_1 \cap L_2$ is also regular

$L_1 \cap L_2$ is the language consisting of all strings in both L_1 and L_2

Intuition: Run both machines in parallel (same as for union) and accept if BOTH of them stop in an accept state

Closure Under Concatenation

If L_1 and L_2 are both regular languages then $L_1 \circ L_2$ is also regular

$L_1 \circ L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$

Outline

Nondeterminism

Deterministic Finite Automaton

- For every state q and every symbol $x \in \Sigma$, exactly one value $\delta(q, x)$ is defined
- State transitions only on an input symbol
- Execution of DFA is fully determined

Nondeterminism

Deterministic Finite Automaton

- For every state q and every symbol $x \in \Sigma$, exactly one value $\delta(q, x)$ is defined
- State transitions only on an input symbol
- Execution of DFA is fully determined

Nondeterministic Finite Automaton

- Allow multiple transitions for same state and symbol:
  e.g., $\delta(q1, 1) = \{q2, q3\}$
- Allow empty (ϵ) transitions – transitions not requiring an input

Nondeterminism

Deterministic Finite Automaton

- For every state q and every symbol $x \in \Sigma$, exactly one value $\delta(q, x)$ is defined
- State transitions only on an input symbol
- Execution of DFA is fully determined

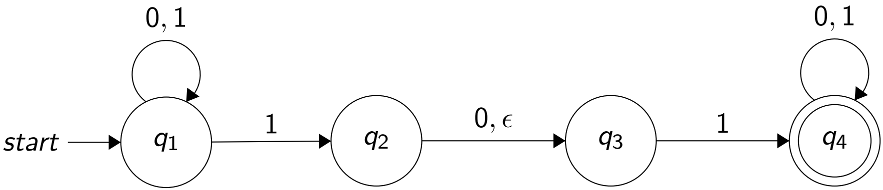
Nondeterministic Finite Automaton

- Allow multiple transitions for same state and symbol:
  e.g., $\delta(q1, 1) = \{q2, q3\}$
- Allow empty (ϵ) transitions – transitions not requiring an input

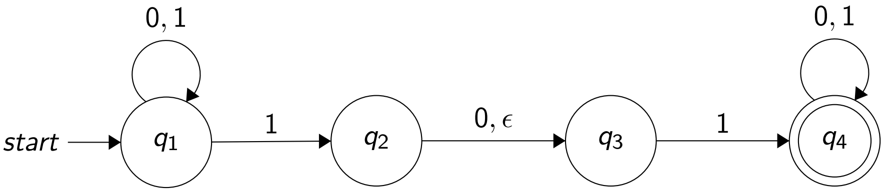
What is going on here?!?

What does non-determinism mean?

An Example NFA

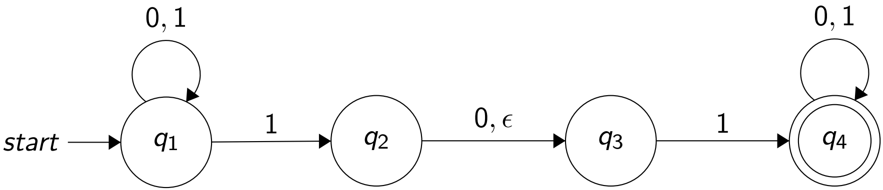


An Example NFA



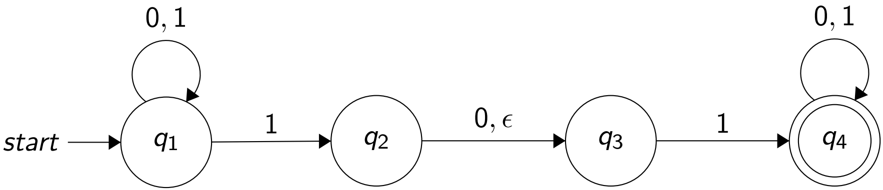
Input: 010

An Example NFA



Input: 010
Input: 010110

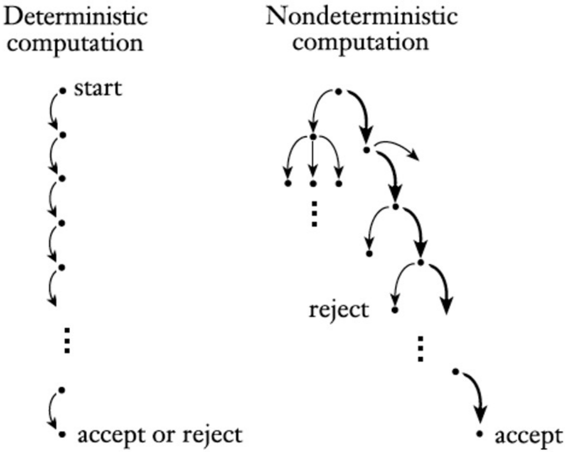
An Example NFA



Input: 010
Input: 010110
Question: What language does this recognize?

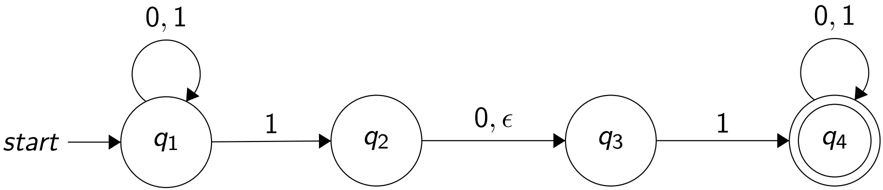
Understanding Nondeterminism

Interpretation 1: Try all paths in parallel



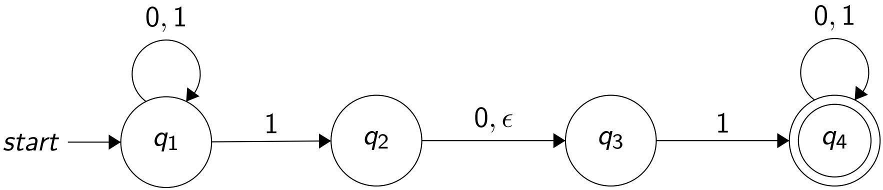
If any path leads to accept then accept

Interpretation 2: Guess and verify



Understanding Nondeterminism

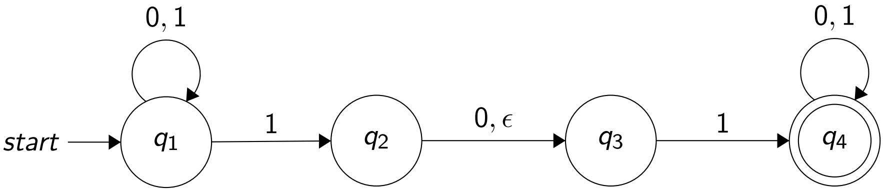
Interpretation 2: Guess and verify



- M stays in q_1 until it "guesses" next input is 11 or 101

Understanding Nondeterminism

Interpretation 2: Guess and verify



- M stays in q_1 until it "guesses" next input is 11 or 101
- Verifies that this guess was correct on path to q_4

Understanding Nondeterminism

Interpretation 3: Verifying a proof vs. finding a solution

Consider the execution of a finite automaton

Understanding Nondeterminism

Interpretation 3: Verifying a proof vs. finding a solution

Consider the execution of a finite automaton

1. DFA execution on input x:
    - A DFA must follow an exact path to an accept state
    - Input x must specify path to an accept state if $x \in L(M)$

Understanding Nondeterminism

Interpretation 3: Verifying a proof vs. finding a solution

Consider the execution of a finite automaton

1. DFA execution on input x:
   - A DFA must follow an exact path to an accept state
   - Input x must specify path to an accept state if $x \in L(M)$
2. NFA execution on input x

Understanding Nondeterminism

Interpretation 3: Verifying a proof vs. finding a solution

Consider the execution of a finite automaton

1. DFA execution on input x:
   - A DFA must follow an exact path to an accept state
   - Input x must specify path to an accept state if $x \in L(M)$
2. NFA execution on input x
   - Input x alone does not necessarily take you to an accept state

Understanding Nondeterminism

Interpretation 3: Verifying a proof vs. finding a solution

Consider the execution of a finite automaton

1. DFA execution on input x:
   - A DFA must follow an exact path to an accept state
   - Input x must specify path to an accept state if $x \in L(M)$
2. NFA execution on input x
   - Input x alone does not necessarily take you to an accept state
   - Need to somehow choose which edge to take whenever there is a choice

Understanding Nondeterminism

Interpretation 3: Verifying a proof vs. finding a solution

Consider the execution of a finite automaton

1. DFA execution on input x:
   - A DFA must follow an exact path to an accept state
   - Input x must specify path to an accept state if $x \in L(M)$
2. NFA execution on input x
   - Input x alone does not necessarily take you to an accept state
   - Need to somehow choose which edge to take whenever there is a choice
   - Can view this sequence of nondeterministic choices as a "witness" w that allows you to verify that $x \in L(M)$

Important

For any $x \notin L$, there must be no path to an accepting state – no possible "witness" works

Nondeterministic Finite Automaton – Formal Definition

Nondeterministic Finite Automaton (NFA)

An NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:

- Q is a finite set of states
- Σ is a finite input alphabet
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to P(Q)$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

Recall:
$P(Q)$ is the power set of Q, i.e., the set of all subsets of Q

Nondeterministic Finite Automaton – Formal Definition

Nondeterministic Finite Automaton (NFA)

An NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:

- Q is a finite set of states
- Σ is a finite input alphabet
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow P(Q)$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

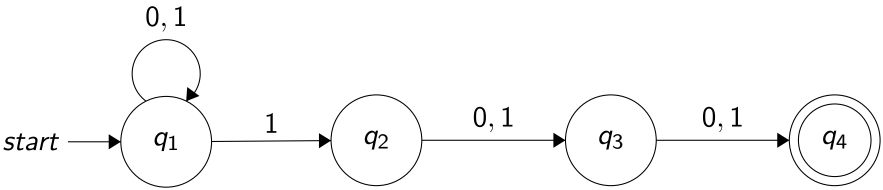
Recall:
$P(Q)$ is the power set of Q, i.e., the set of all subsets of Q

Changes:

1. Transition function allows empty symbol (ϵ)
2. Output of transition function is a set of states $\in P(Q)$, not a single state in Q

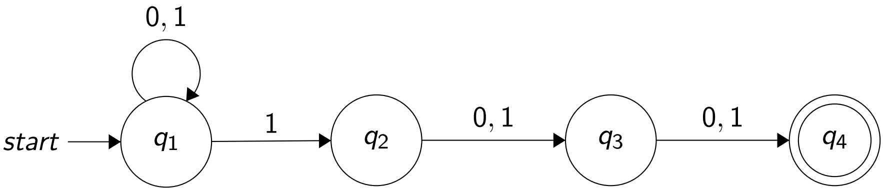
Outline

NFA Example 1



Question: What is $L(M)$?

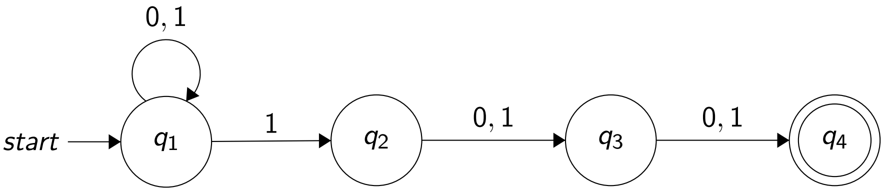
NFA Example 1



Question: What is $L(M)$?

Answer: Strings in $\{0, 1\}^*$ with a 1 as third from the end

NFA Example 1

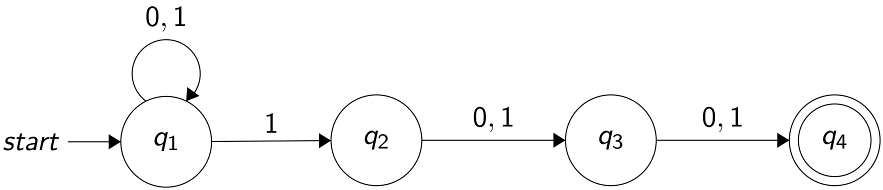


Question: What is $L(M)$?
Answer: Strings in $\{0,1\}^*$ with a 1 as third from the end

How does it work?

- M waits in q_1 until it "guesses" that it is 3 symbols from the end

NFA Example 1

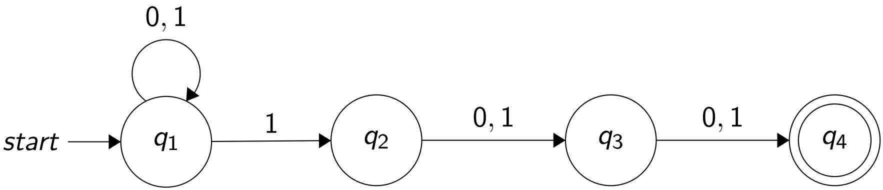


Question: What is $L(M)$?
Answer: Strings in $\{0, 1\}^*$ with a 1 as third from the end

How does it work?

- M waits in q_1 until it "guesses" that it is 3 symbols from the end
- Uses the rest of the states to verify that 1 is third from the end

NFA Example 1



Question: What is $L(M)$?
Answer: Strings in $\{0, 1\}^*$ with a 1 as third from the end

How does it work?

- M waits in q_1 until it "guesses" that it is 3 symbols from the end
- Uses the rest of the states to verify that 1 is third from the end
- DFA doing the same thing would have to track the last three bits seen – requires 8 states

Example 2 – OR statement

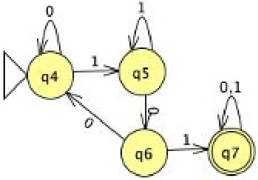
$L = \{x | x \in \{0, 1\}^*$ and x contains

1. the substring 101, or
2. the substring 010$\}$

Example 2 – OR statement

$L = \{x | x \in \{0, 1\}^*$ and x contains

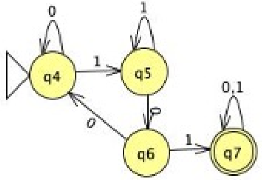
1. the substring 101, or
2. the substring 010$\}$

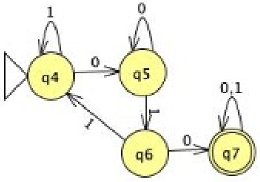


Example 2 – OR statement

$L = \{x | x \in \{0, 1\}^*$ and x contains

1. the substring 101, or
2. the substring 010$\}$

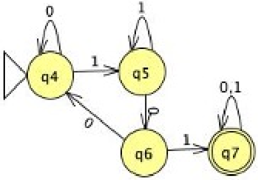




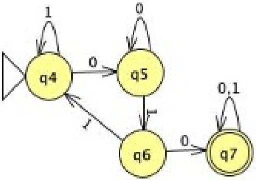
Example 2 – OR statement

$L = \{x | x \in \{0,1\}^* \text{ and } x \text{ contains}$

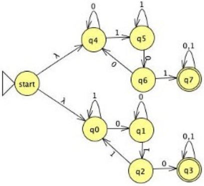
1. the substring 101, or
2. the substring 010$\}$



DFA for prop. (1)



DFA for prop. (2)



NFA for L