DFA for complement of a language

- Flip final and non-final states.

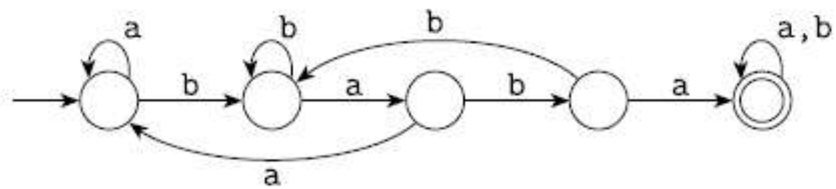
1.5 Each of the following languages is the complement of a simpler language. In each part, construct a DFA for the simpler language, then use it to give the state diagram of a DFA for the language given. In all parts, $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

[A]**a.** $\{w \mid w \text{ does not contain the substring } \mathtt{ab}\}$

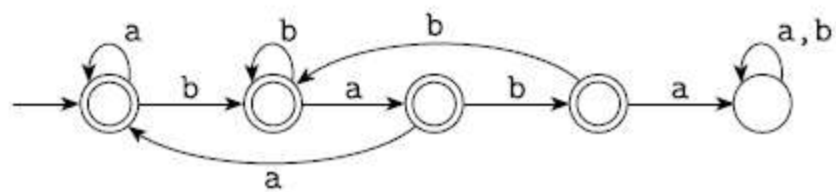
[A]**b.** $\{w \mid w \text{ does not contain the substring } \mathtt{baba}\}$

1.5 **(a)** The left-hand DFA recognizes $\{w \mid w \text{ contains } \texttt{ab}\}$. The right-hand DFA recognizes its complement, $\{w \mid w \text{ doesn't contain } \texttt{ab}\}$.

(b) This DFA recognizes $\{w \mid w \text{ contains } \texttt{baba}\}$.



This DFA recognizes $\{w \mid w \text{ does not contain } \texttt{baba}\}$.



Designing a DFA (Quick Quiz)

- How to design a DFA that accepts all binary strings representing a multiple of 5? (E.g., 101, 1111, 11001, ...)

Formally

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \cdots w_n$ be a string where each w_i is a member of the alphabet Σ. Then M ***accepts*** w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:

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

Regular language [Ref: Sipser Book]

DEFINITION 1.16

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

Set

- A set is a group of items
- One way to describe a set: list every item in the group inside { }
  - E.g., { 12, 24, 5 } is a set with three items
- When the items in the set has trend: use …
  - E.g., { 1, 2, 3, 4, … } means the set of natural numbers
- Or, state the rule
  - E.g., $\{ n \mid n = m^2$ for some positive integer $m \}$ means the set { 1, 4, 9, 16, 25, … }
- A set with no items is an empty set denoted by { } or $\emptyset$

Set

- The order of describing a set does not matter
  - $\{ 12, 24, 5 \} = \{ 5, 24, 12 \}$
- Repetition of items does not matter too
  - $\{ 5, 5, 5, 1 \} = \{ 1, 5 \}$
- Membership symbol $\in$
  - $5 \in \{ 12, 24, 5 \}$ $\quad 7 \notin \{ 12, 24, 5 \}$

- How many items are in each of the following set?
  - $\{ 3, 4, 5, \ldots, 10 \}$
  - $\{ 2, 3, 3, 4, 4, 2, 1 \}$
  - $\{ 2, \{2\}, \{\{1,2,3,4,5,6\}\} \}$
  - $\emptyset$
  - $\{\emptyset\}$

Set

Given two sets A and B

- we say $A \subseteq$ B (read as A is a subset of B) if every item in A also appears in B
  - E.g., A = the set of primes, B = the set of integers
- we say $A \subsetneq$ B (read as A is a proper subset of B) if $A \subseteq$ B but $A \neq$ B

Warning: Don't be confused with $\in$ and $\subseteq$

- Let A = { 1, 2, 3 }. Is $\emptyset \in A$? Is $\emptyset \subseteq A$?

Union, Intersection, Complement

Given two sets A and B

- $A \cup B$ (read as the union of A and B) is the set obtained by combining all elements of A and B in a single set
  - E.g., $A = \{ 1, 2, 4 \}$ $B = \{ 2, 5 \}$
    $A \cup B = \{ 1, 2, 4, 5 \}$
- $A \cap B$ (read as the intersection of A and B) is the set of common items of A and B
  - In the above example, $A \cap B = \{ 2 \}$
- $\overline{A}$ (read as the complement of A) is the set of items under consideration not in A

Set

- The **power set** of A is the set of all subsets of A, denoted by 2^A
  - E.g., $A = \{ 0, 1 \}$
    $2^A = \{ \{\}, \{0\}, \{1\}, \{0,1\} \}$
  - How many items in the above power set of A?
- If A has n items, how many items does its power set contain? Why?

Sequence

- A sequence of items is a list of these items in some order
- One way to describe a sequence: list the items inside ( )
  - (5, 12, 24)
- Order of items inside ( ) matters
  - $(5, 12, 24) \neq (12, 5, 24)$
- Repetition also matters
  - $(5, 12, 24) \neq (5, 12, 12, 24)$
- Finite sequences are also called tuples
  - (5, 12, 24) is a 3-tuple
  - (5, 12, 12, 24) is a 4-tuple

Sequence

Given two sets A and B

- The Cartesian product of A and B, denoted by $A \times B$, is the set of all possible 2-tuples with the first item from A and the second item from B
  - E.g., $A = \{1, 2\}$ and $B = \{x, y, z\}$

  $A \times B = \{ (1,x), (1,y), (1,z), (2,x), (2,y), (2,z) \}$

- The Cartesian product of k sets, $A_1, A_2, \ldots, A_k$, denoted by $A_1 \times A_2 \times \cdots \times A_k$, is the set of all possible k-tuples with the i^{th} item from A_i

The regular operations

DEFINITION 1.23

Let A and B be languages. We define the regular operations ***union***, ***concatenation***, and ***star*** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
- **Concatenation**: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$.
- **Star**: $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$.

- These are similar to arithmetic operations.
- Note, * is a unary operator.

THEOREM 1.25

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

- The proof is by construction.
- We build a DFA for the union from the individual DFAs.

- The idea is simple: While reading the input simultaneously follow both machines.
  - Put a finger on current state. You need two fingers. You can move these two fingers as per the respective transition function.

PROOF

Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and
M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construct M to recognize $A_1 \cup A_2$, where $M = (Q, \Sigma, \delta, q_0, F)$.

1. $Q = \{(r_1, r_2) | \, r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$.
   This set is the ***Cartesian product*** of sets Q_1 and Q_2 and is written $Q_1 \times Q_2$.
   It is the set of all pairs of states, the first from Q_1 and the second from Q_2.

2. Σ, the alphabet, is the same as in M_1 and M_2. In this theorem and in all subsequent similar theorems, we assume for simplicity that both M_1 and M_2 have the same input alphabet Σ. The theorem remains true if they have different alphabets, Σ_1 and Σ_2. We would then modify the proof to let $\Sigma = \Sigma_1 \cup \Sigma_2$.

3. δ, the transition function, is defined as follows. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let

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

Hence δ gets a state of M (which actually is a pair of states from M_1 and M_2), together with an input symbol, and returns M's next state.

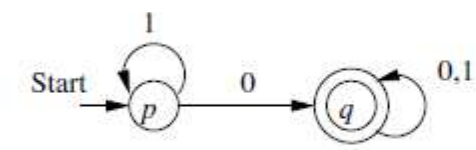
4. q_0 is the pair (q_1, q_2).

5. F is the set of pairs in which either member is an accept state of M_1 or M_2. We can write it as

$$F = \{(r_1, r_2) |\, r_1 \in F_1 \text{ or } r_2 \in F_2\}.$$

This expression is the same as $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$. (Note that it is *not* the same as $F = F_1 \times F_2$. What would that give us instead?[3])

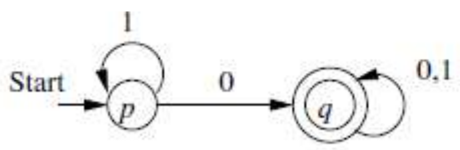
Union Example



(a)

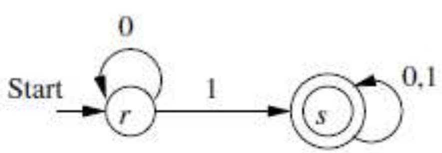
What is the language recognized by this DFA?

Union Example



(a)

What is the language recognized by this DFA?

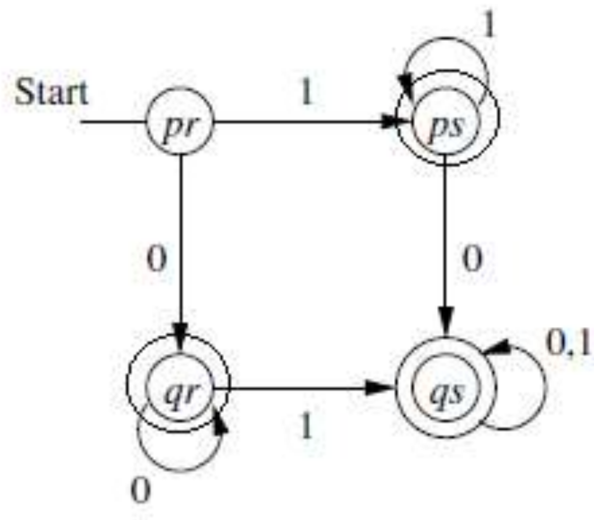


(b)

What is the language recognized by this DFA?

Find DFA for the union

Find DFA for the union

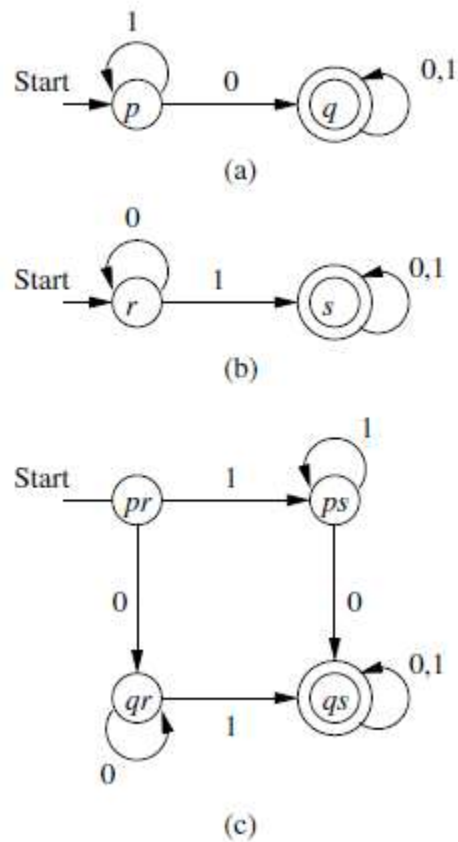


(c)

What about intersection?

- Intersection of two regular languages is also regular.
- Proof: by construction. Similar. Only final states will change.

Intersection



What else we can do with product principle?

- Set difference.
  - How?

$$A - B = A \cap \bar{B}$$

1.4 Each of the following languages is the intersection of two simpler languages. In each part, construct DFAs for the simpler languages, then combine them using the construction discussed in footnote 3 (page 46) to give the state diagram of a DFA for the language given. In all parts, $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

a. $\{w \mid w$ has at least three a's and at least two b's$\}$

[A]**b.** $\{w \mid w$ has exactly two a's and at least two b's$\}$

c. $\{w \mid w$ has an even number of a's and one or two b's$\}$

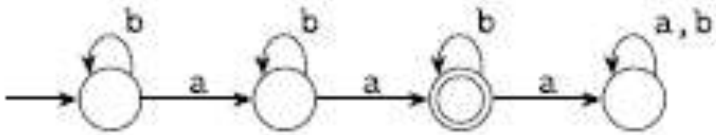
[A]**d.** $\{w \mid w$ has an even number of a's and each a is followed by at least one b$\}$

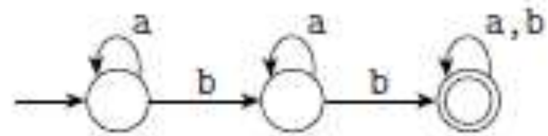
e. $\{w \mid w$ starts with an a and has at most one b$\}$

f. $\{w \mid w$ has an odd number of a's and ends with a b$\}$

g. $\{w \mid w$ has even length and an odd number of a's$\}$

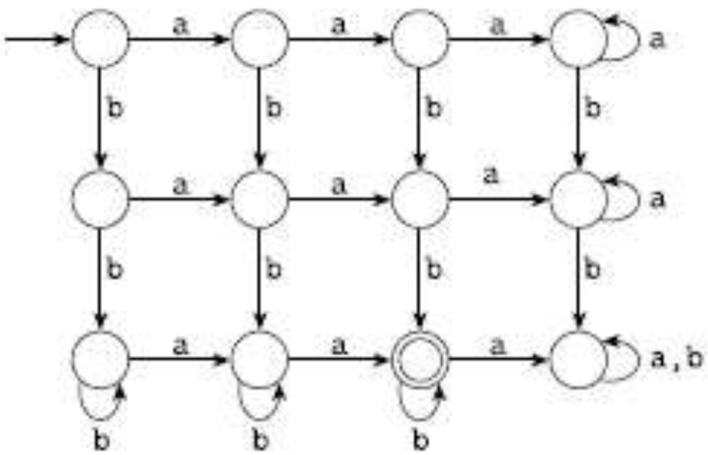
1.4 (b) The following are DFAs for the two languages $\{w \mid w \text{ has exactly two a's}\}$ and $\{w \mid w \text{ has at least two b's}\}$.





- Now find product machine.

Combining them using the intersection construction gives the following DFA.



- This can be minimized. {Some states are redundant}.

NONDETERMINISM

- Useful concept, has great impact on ToC/algorithms.
- DFA is deterministic: every step of a computation follows in a unique way from the preceding step.
  - When the machine is in a given state, and upon reading the next input symbol, we know deterministically what would be the next state.
  - Only one next state.
  - No choice !!

NONDETERMINISM

- In a nondeterministic machine, several choices may exist for the next state at any point.
- Nondeterminism is a generalization of determinism.

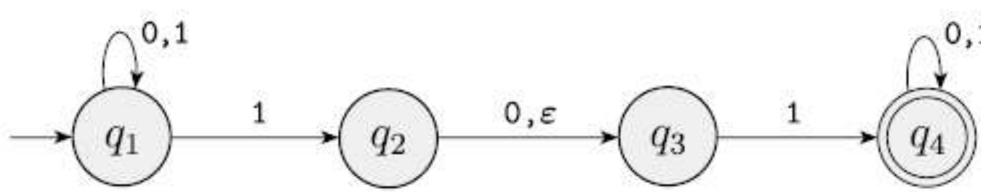


FIGURE 1.27
The nondeterministic finite automaton N_1

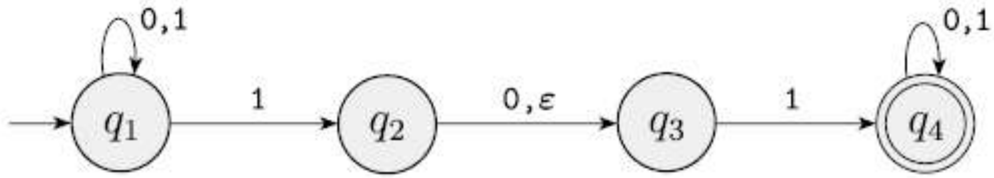


FIGURE 1.27
The nondeterministic finite automaton N_1

- More than one arrow from from q_1 on symbol 1.
- No arrow at all from q_3 on 0.
- There is Ɛ over an arrow !

How does an NFA compute?

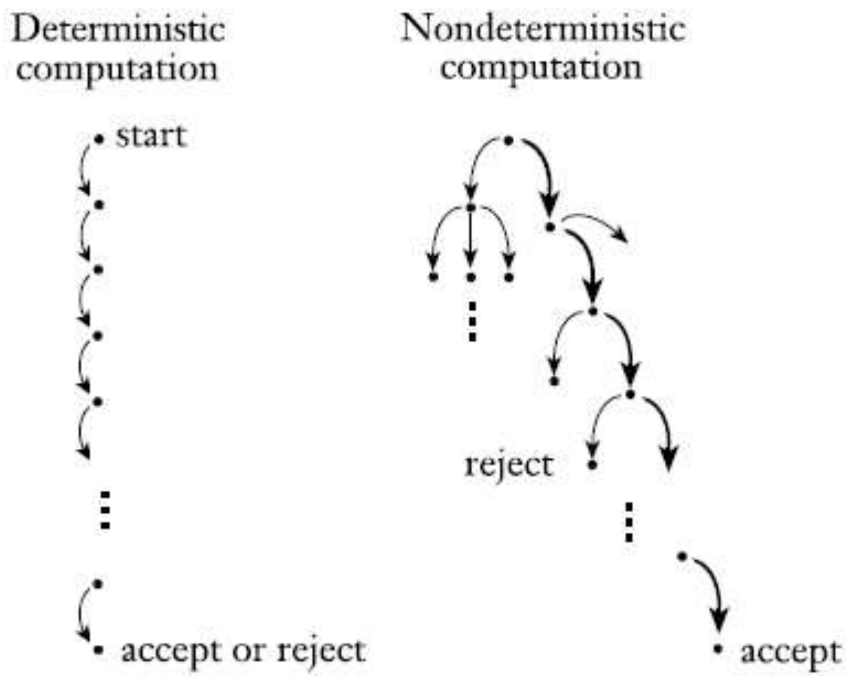


FIGURE 1.28
Deterministic and nondeterministic computations with an accepting branch

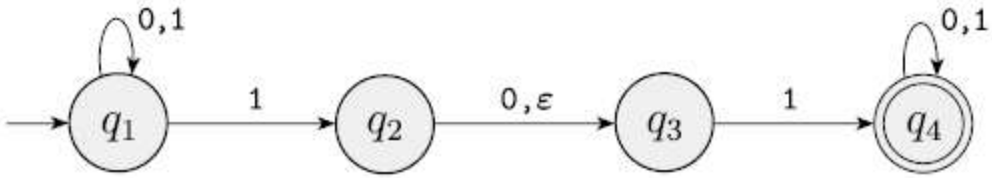


FIGURE 1.27
The nondeterministic finite automaton N_1

On input **010110**

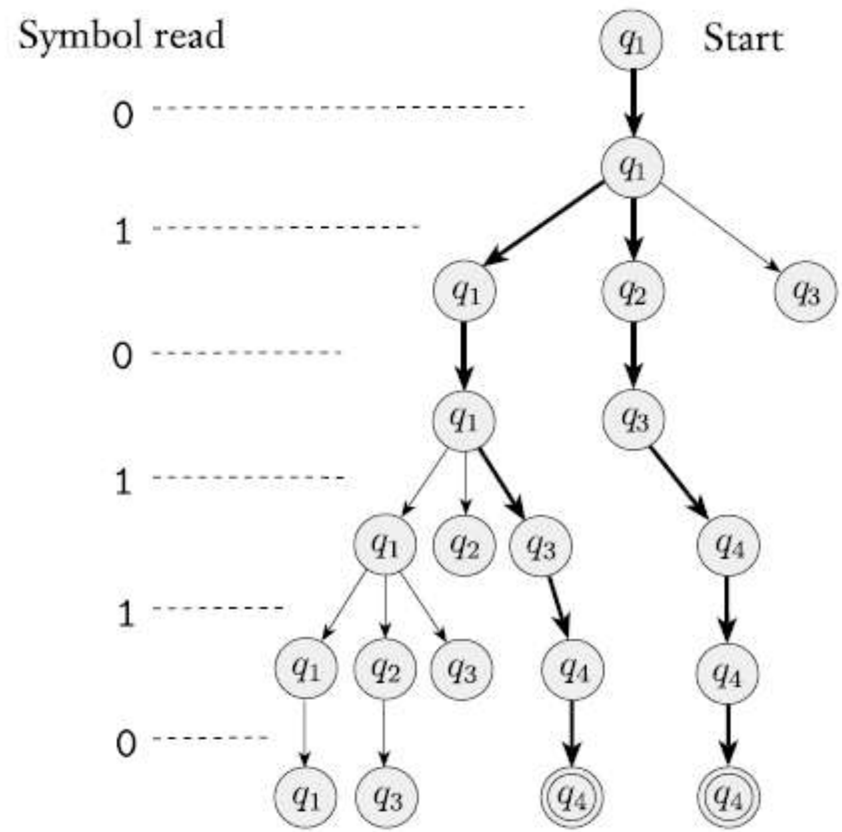


FIGURE 1.29
The computation of N_1 on input 010110

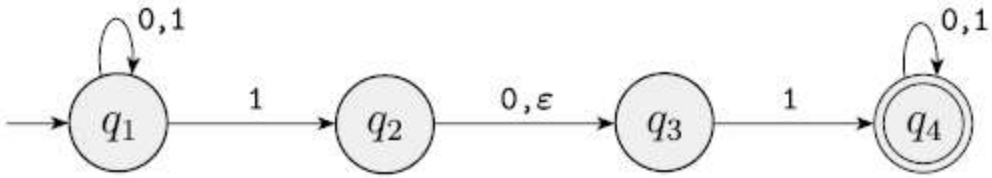


FIGURE 1.27
The nondeterministic finite automaton N_1

- What is the language accepted by this NFA?

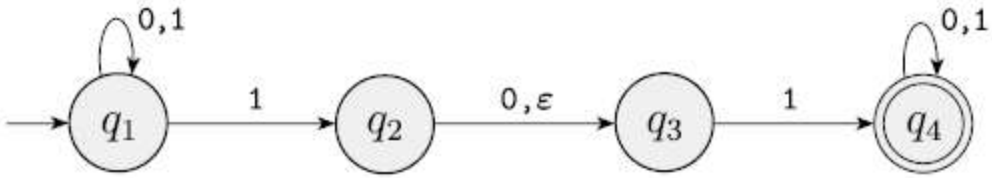


FIGURE **1.27**
The nondeterministic finite automaton N_1

- It accepts all strings that contain either 101 or 11 as a substring.

- Constructing NFAs is sometimes easier than constructing DFAs.
  - Later we see that every NFA can be converted into an equivalent DFA.

EXAMPLE 1.30

Let A be the language consisting of all strings over $\{0,1\}$ containing a 1 in the third position from the end (e.g., 000100 is in A but 0011 is not). The following four-state NFA N_2 recognizes A.

- Building DFA for this is possible, but difficult.
- Try this.

But NFA is easy to build.

EXAMPLE 1.30

Let A be the language consisting of all strings over $\{0,1\}$ containing a 1 in the third position from the end (e.g., 000100 is in A but 0011 is not). The following four-state NFA N_2 recognizes A.

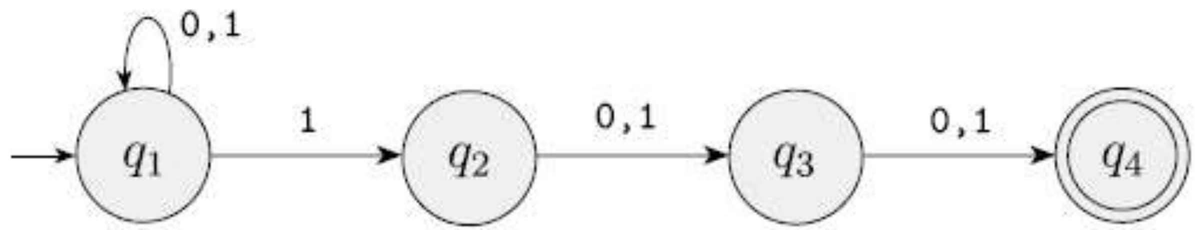


FIGURE 1.31
The NFA N_2 recognizing A

DFA for A

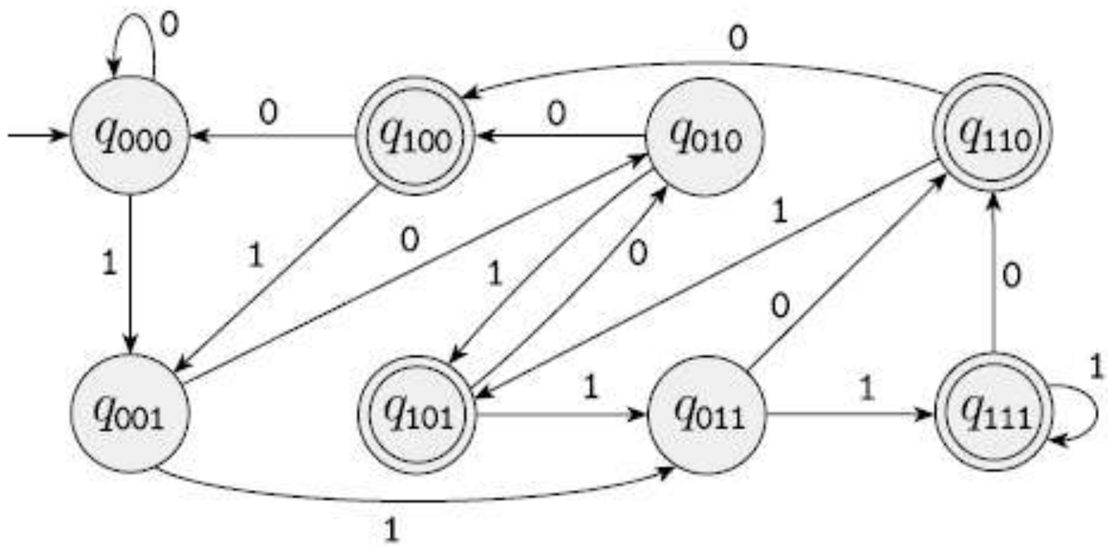


FIGURE **1.32**
A DFA recognizing A

- See number of states and complexity !

Formal definition of NFA

We use Σ_ε to mean $\Sigma \cup \{\varepsilon\}$

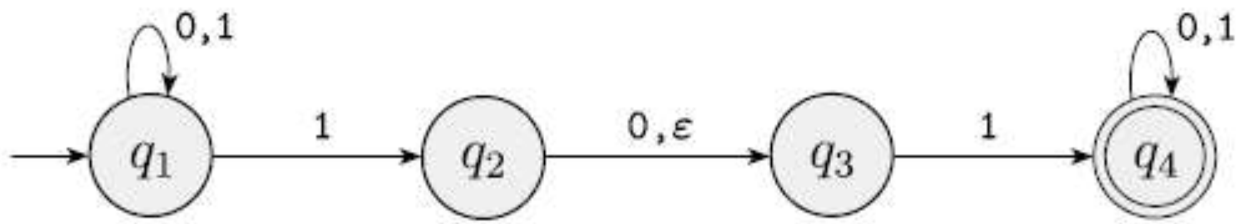
DEFINITION 1.37

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. $\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

EXAMPLE 1.38

Recall the NFA N_1:



The formal description of N_1 is $(Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3, q_4\}$,
2. $\Sigma = \{0,1\}$,
3. δ is given as

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$,

4. q_1 is the start state, and
5. $F = \{q_4\}$.

The formal definition of computation for an NFA is similar to that for a DFA. Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w a string over the alphabet Σ. Then we say that N ***accepts*** w if we can write w as $w = y_1 y_2 \cdots y_m$, where each y_i is a member of Σ_ε and a sequence of states $r_0, r_1, \ldots, r_m$ exists in Q with three conditions:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, \ldots, m-1$, and
3. $r_m \in F$.