CS 4410

Automata, Computability, and Formal Language

Dr. Xuejun Liang

Chapter 2

Finite Automata

1. Deterministic Finite Accepters
    - Deterministic Accepters and Transition Graphs
    - Languages and Dfas
    - Regular Language
2. Nondeterministic Finite Accepters
    - Definition of a Nondeterministic Accepter
    - Why Nondeterministic
3. Equivalence of Deterministic and Nondeterministic Finite Accepters

Learning Objectives

At the conclusion of the chapter, the student will be able to:

- Describe the components of a deterministic finite accepter (dfa)
- State whether an input string is accepted by a dfa
- Describe the language accepted by a dfa
- Construct a dfa to accept a specific language
- Show that a particular language is regular
- Describe the differences between deterministic and nondeterministic finite automata (nfa)
- State whether an input string is accepted by a nfa
- Construct a nfa to accept a specific language
- Transform an arbitrary nfa to an equivalent dfa

Nondeterministic Finite Accepters

An automaton is nondeterministic if it has a choice of actions for given conditions

Definition 2.4

A nondeterministic finite accepter or nfa is defined by the quintuple

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

where Q, Σ, q_0, and F are as for deterministic finite accepter, but

$$\delta: Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$$

Basic differences between deterministic and nondeterministic finite automata:

- In an nfa, a (state, symbol) combination may lead to several states simultaneously
- If a transition is labeled with the empty string as its input symbol, the nfa may change states without consuming input
- An nfa may have undefined transitions

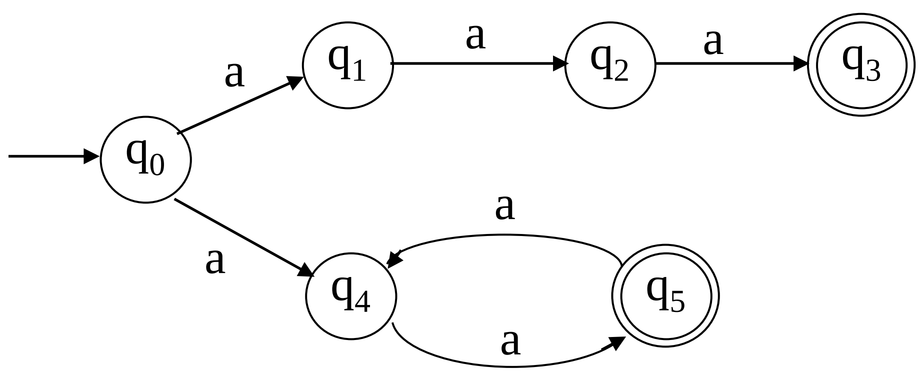
Nondeterministic Finite Accepters

Transition Graph of an nfa $M = (Q, \Sigma, \delta, q_0, F)$

Vertex labeled with q_i: state $q_i \in Q$,

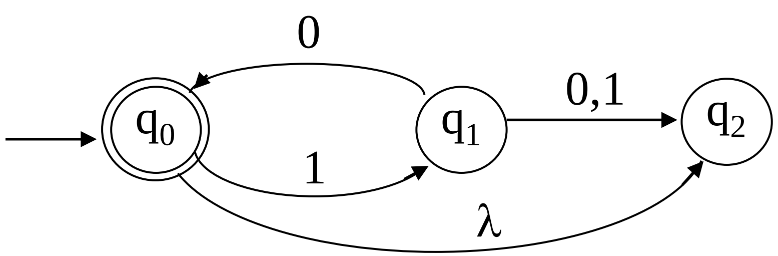
Edge from q_i to q_j labeled with a: $q_j \in \delta(q_i, a)$

Example 2.7: An nfa is shown as below in Figure 2.8



Nondeterministic Finite Accepters

Example 2.8   An nfa is shown in Figure 2.9 as below



The extended transition function $\delta^* : Q \times \Sigma^* \rightarrow 2^Q$ could be defined by

$$\delta^*(q,\lambda) = \{q\} \cup \delta(q,\lambda)$$

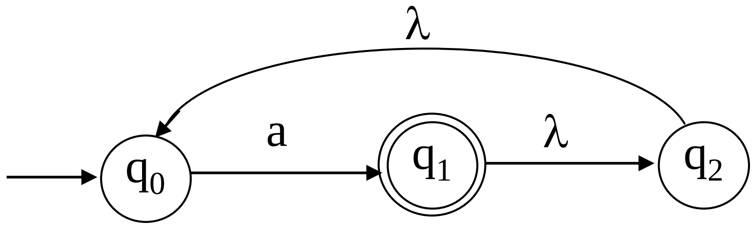
$$\delta^*(q,wa) = \cup\{\delta(p,a) : p \in \delta^*(q,w)\}$$

Definition 2.5 (This could a theorem if the above definition is used)

For an nfa, the extended transition function is defined so that $\delta^*(q_i,w)$ contains q_j if and only if there is a walk in the transition graph from q_i to q_j labeled w. This holds for all q_i, $q_j \in Q$ and $w \in \Sigma^*$.

Nondeterministic Finite Accepters

Example 2.9 Consider an nfa in Figure 2.10, we have



$$\delta^*(q_1, a) = \{q_0, q_1, q_2\}$$

$$\delta^*(q_2, \lambda) = \{q_0, q_2\}$$

$$\delta^*(q_2, aa) = \{q_0, q_1, q_2\}$$

$$\delta^*(q, \lambda) = \{q\} \cup \delta(q, \lambda)$$

$$\delta^*(q, wa) = \cup\{\delta(p, a) : p \in \delta^*(q, w)\}$$

Nondeterministic Finite Accepters

Definition 2.6

The language accepted by an nfa M = (Q, Σ, δ, q_0, F) is the set of all strings on Σ accepted by M. In formal notation

$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \cap F \neq \varnothing\}.$$

Example 2.10 What is the language accepted by the nfa in Example 2.8

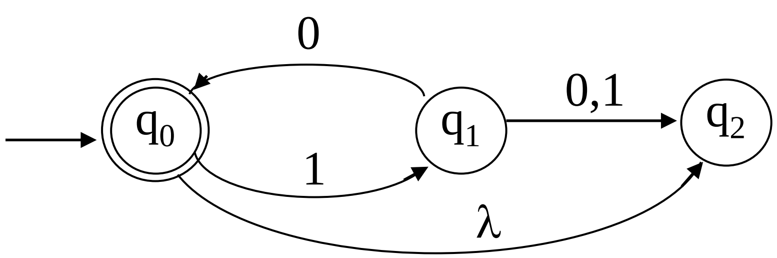


Figure 2.9

Why Nondeterminism?

Equivalence of Deterministic and Nondeterministic Finite Accepters

Definition 2.7

Two finite accepters M_1 and M_2 are said to be equivalent if

$$L(M_1)=L(M_2)$$

That is, if they accept the same language.

Example 2.11

The dfa is equivalent to the nfa in Example 2.8

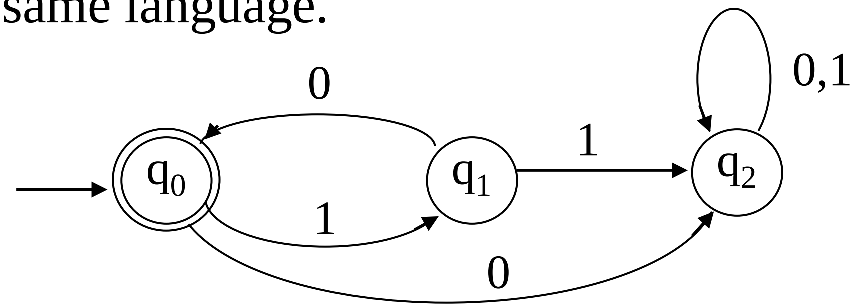


Figure 2.11

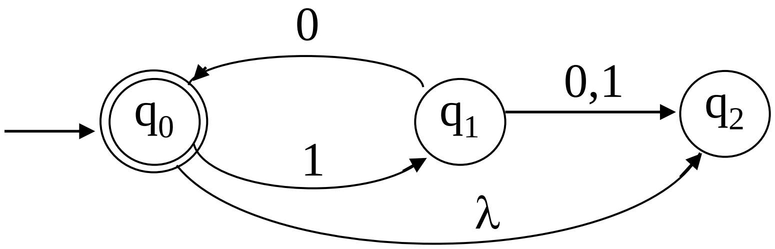


Figure 2.9

Example 2.12

Convert the nfa to an equivalent dfa

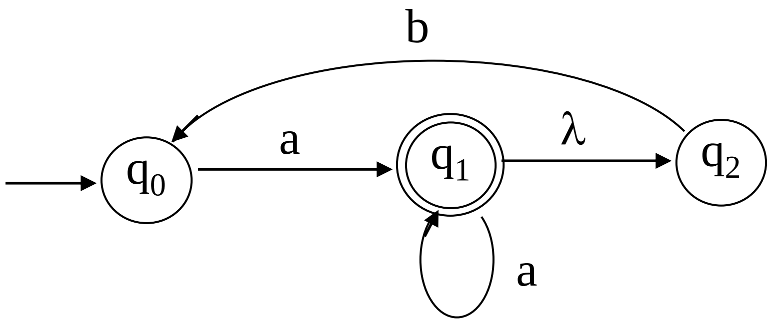


Figure 2.12

DFA state	a	b
$\{q_0\}$	$\{q_1, q_2\}$	Φ
$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_0\}$
Φ	Φ	Φ

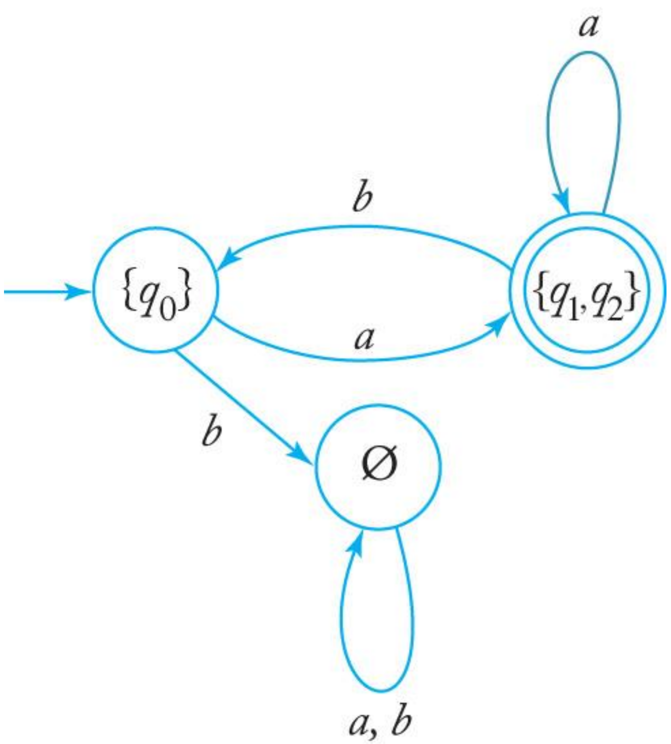


Figure 2.13

Theorem 2.2 Let L be the language accepted by an nfa $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$. Then there exists a dfa $M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ such that $L = L(M_D)$.

Procedure: nfa-to-dfa

1. Create a graph G_D with vertex $\{q_0\}$ as the initial vertex
2. Repeat until no more edges are missing
   a. Take any vertex $\{q_i, q_j, .., q_k\}$ of G_D that has no outgoing edge for some $a \in \Sigma$.
   b. Compute $\{q_l, q_m, \ldots, q_n\} = \delta^*(q_i, a) \cup \delta^*(q_j, a) \cup \ldots \cup \delta^*(q_k, a)$
   c. Create a vertex for G_D labeled $\{q_l, q_m, \ldots, q_n\}$ if it does not already exist.
   d. Add to G_D an edge from $\{q_i, q_j, .., q_k\}$ to $\{q_l, q_m, \ldots, q_n\}$ and label it with a
3. Every state of G_D whose label contains any $q_f \in F_N$ is identified as a final vertex.
4. If M_N accepts λ, the vertex $\{q_0\}$ in G_D is also made a final vertex.

Property: Every language accepted by an nfa is regular

Example 2.13

Convert the nfa to an equivalent dfa

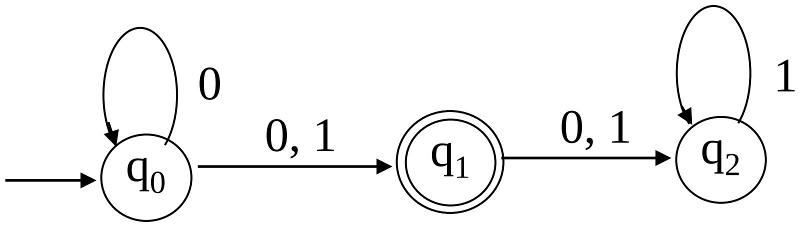


Figure 2.14

DFA state	0	1
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
$\{q_1\}$	$\{q_2\}$	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_2\}$	Φ	$\{q_2\}$
Φ	Φ	Φ