Formal languages and Automata
形式语言与自动机

Chapter2 FINITE AUTOMATA

Beijing University of Posts and Telecommunications

北京邮电大学

刘咏彬 liiuyb@bupt.edu.cn

2023.9

Finite Automaton (FA)

- Recognizer for “Regular Languages”

- A finite automaton is severely limited in its capacity to “remember” things during the computation.

- Deterministic Finite Automata (DFA)
  - The machine can exist in only one state at any given time
- Non-deterministic Finite Automata (NFA)
  - The machine can exist in multiple states at the same time

Finite State Accepter

- **finite:** a finite set of internal states and no other memory.
- **accepter:** it processes strings and either accepts or rejects them, so we can think of it as a simple pattern recognition mechanism.

Deterministic Finite Accepter - Definition 2.1

- A DFA is defined by the 5-tuple:

  $(Q, \sum, \delta, q_0, F)$

- A Deterministic Finite Accepter(DFA) consists of:

  Q : a finite set of internal states

  $\sum$: a finite set of input symbols (alphabet)

  δ : $Q \times \sum \rightarrow Q$,is a total function called the transition function

  q_0 : a initial state (start state)

  F : set of final states(accepting state)

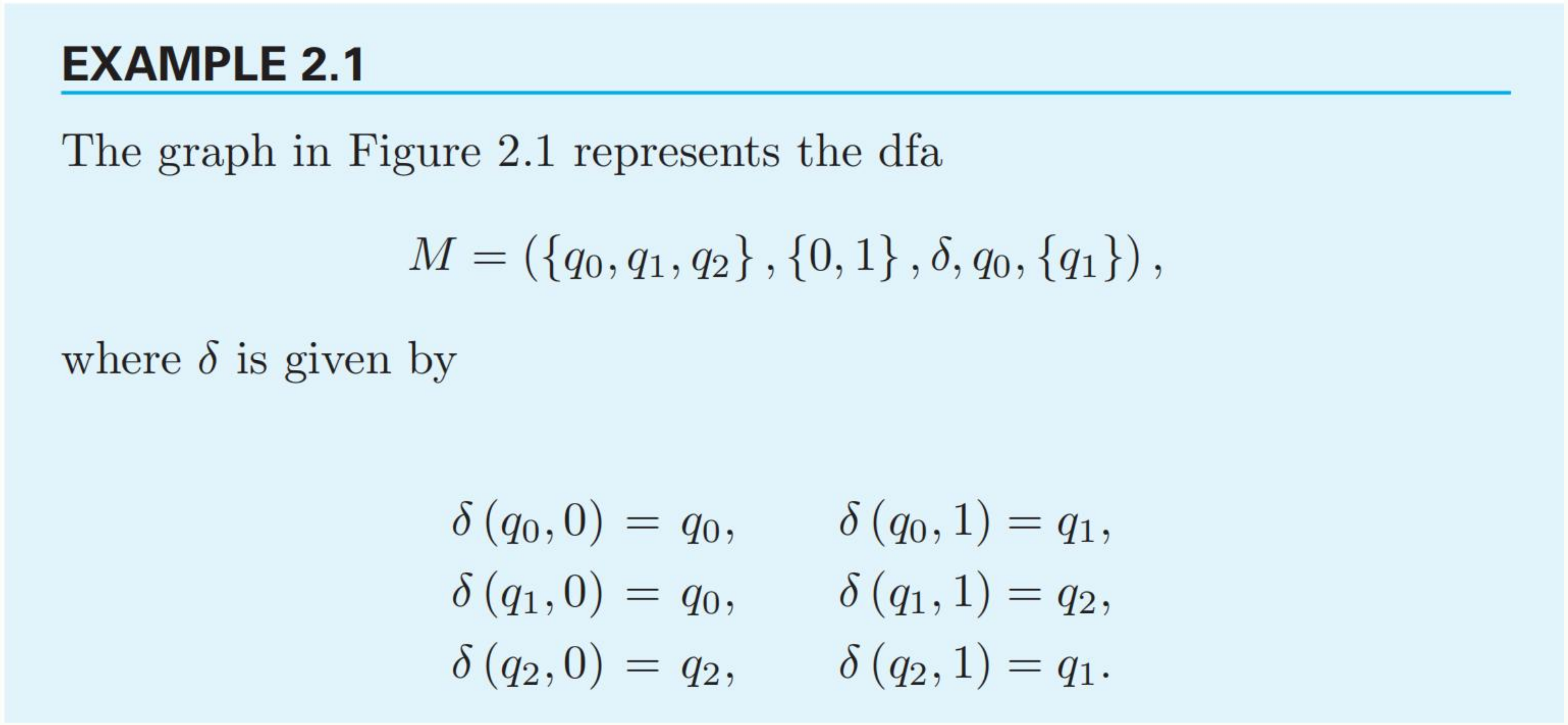
Transition function δ

- The transitions from one internal state to another are governed by the transition function δ.
- For example, if

$$\delta\ (q_0,\ a) = q_1,$$

then if the dfa is in state q_0 and the current input symbol is a, the dfa will go into state q_1.

Example 2.1



EXAMPLE 2.1

The graph in Figure 2.1 represents the dfa

$$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\}),$$

where δ is given by

$$\begin{aligned} \delta(q_0, 0) &= q_0, & \delta(q_0, 1) &= q_1, \\ \delta(q_1, 0) &= q_0, & \delta(q_1, 1) &= q_2, \\ \delta(q_2, 0) &= q_2, & \delta(q_2, 1) &= q_1. \end{aligned}$$

Transition graphs

- The **vertices** represent states and the edges represent transitions.
- The labels on the vertices are the names of the **states**,
- The labels on the edges are the current values of the **input symbol**.

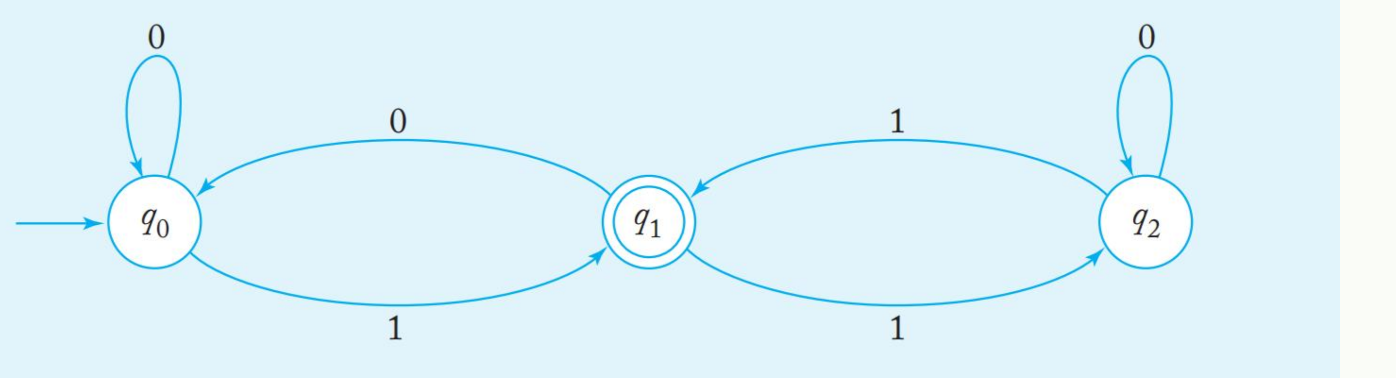
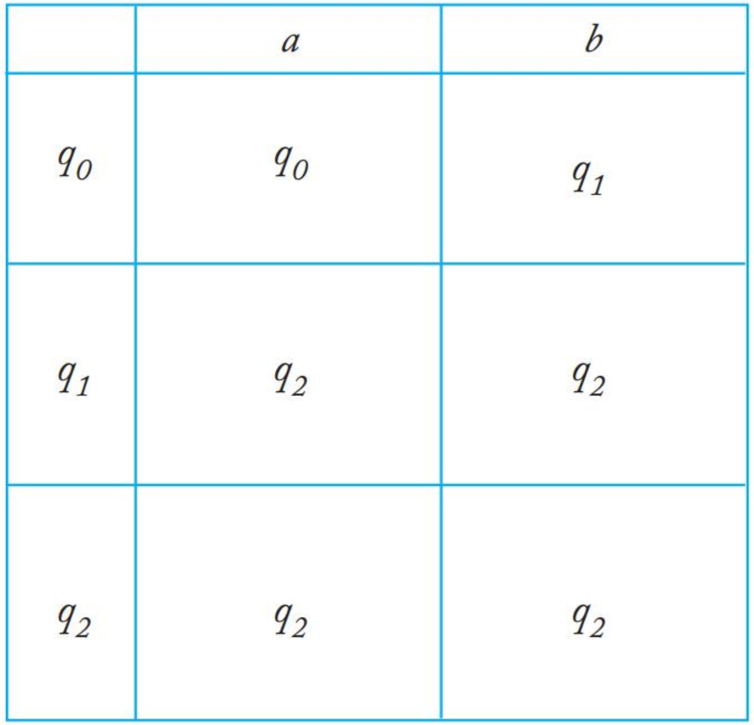


FIGURE 2.1

Transition table

- A dfa can easily be implemented as a computer program; for example, as a simple table-lookup or as a sequence of if statements.



		a	b
开始状态	q_0	q_0	q_1
	q_1	q_2	q_2
接受状态	q_2	q_2	q_2

FIGURE 2.3

注意：在状态转移表的开始状态和结束状态上标记

What does a DFA do on reading an input string?

- Input: a string w in Σ^*
- Question: Is w acceptable by the DFA?
- Steps:
  - Start at the "start state" q_0
  - For every input symbol in the sequence w do
    - Compute the next state from the current state, given the current input symbol in w and the transition function
  - If after all symbols in w are consumed, the current state is one of the accepting states (F) then *accept w;*
  - Otherwise, *reject w.*

补充 Example

- Build a DFA for the following language:

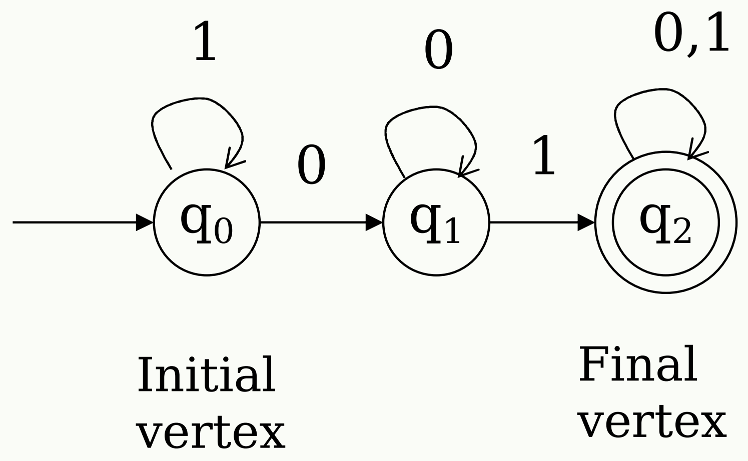
  L = {w | w is a binary string that contains 01 as a substring}

- Steps for building a DFA to recognize L:

  1. ∑ = {0,1}
  2. Decide on the states: Q
  3. Designate start state and final state(s)
  4. δ: Decide on the transitions:
     - "Final" states == same as "accepting states"
     - Other states == same as "non-accepting states"

DFA for strings containing 01

- Multiply labeled edges are shorthand for two or more distinct transitions



- What makes this DFA deterministic?

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0,1\}$
- start state = q_0
- $F = \{q_2\}$
- Transition table

symbols

states

δ	0	1
→q_0	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2

Extended transitions funtion

- $\delta^*(q,\ w)$ == *destination state* from state *q* on input string *w*

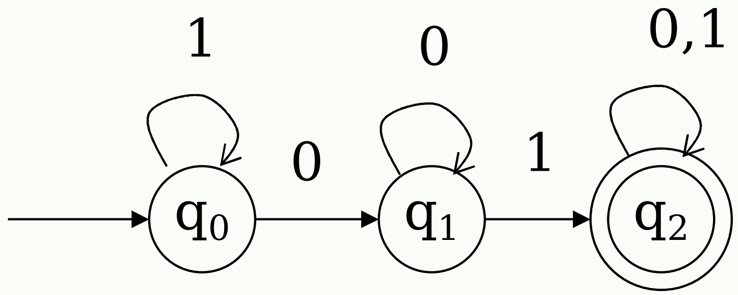
- Formally, we can define δ recursively by

$$\delta^*(q,\ \lambda) = q,$$

$$\delta^*(q,\ wa) = \delta\ (\delta^*(q,\ w),\ a),$$

  - Work out using the input sequence w=10010, a=1:

    - $\delta^*(q_0, wa) = ?$



Languages and Dfa’ s

- The language is the set of all the strings accepted by the automaton.
- **DEFINITION 2.2**

  The language accepted by a dfa M = (Q, Σ, δ, q0, F) is the set of all strings on Σ accepted by M. In formal notation,

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

- A dfa will process every string in Σ and either accept it or not accept it.
- Nonacceptance means that the dfa stops in a nonfinal state, so that

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

Trap state

- The state q_2 is a trap state.
- The automaton accepts all strings consisting of an arbitrary number of a' s, followed by a single b.

$$L = \{a^n b : n \geq 0\}.$$

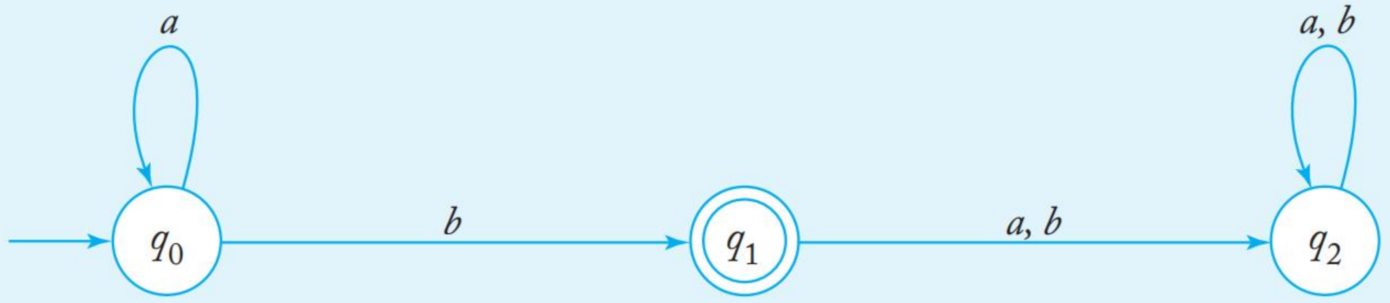
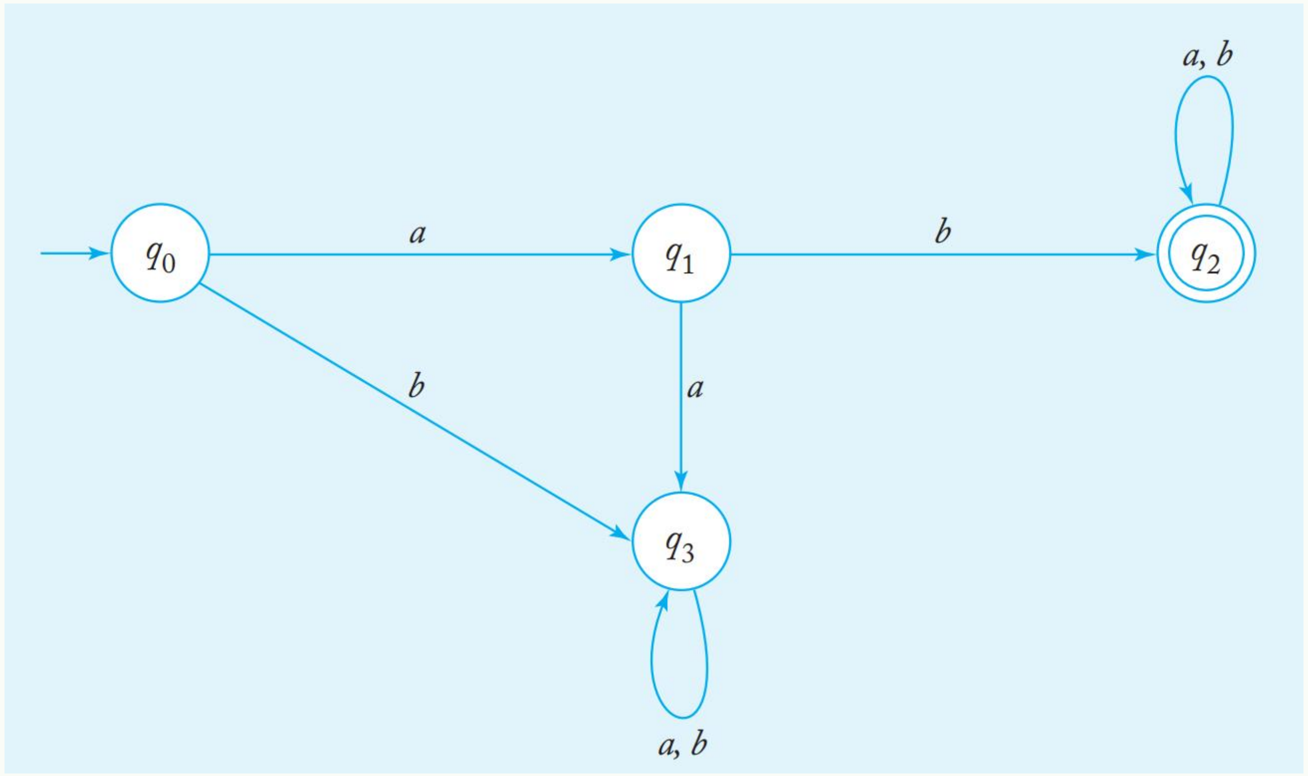


FIGURE 2.2

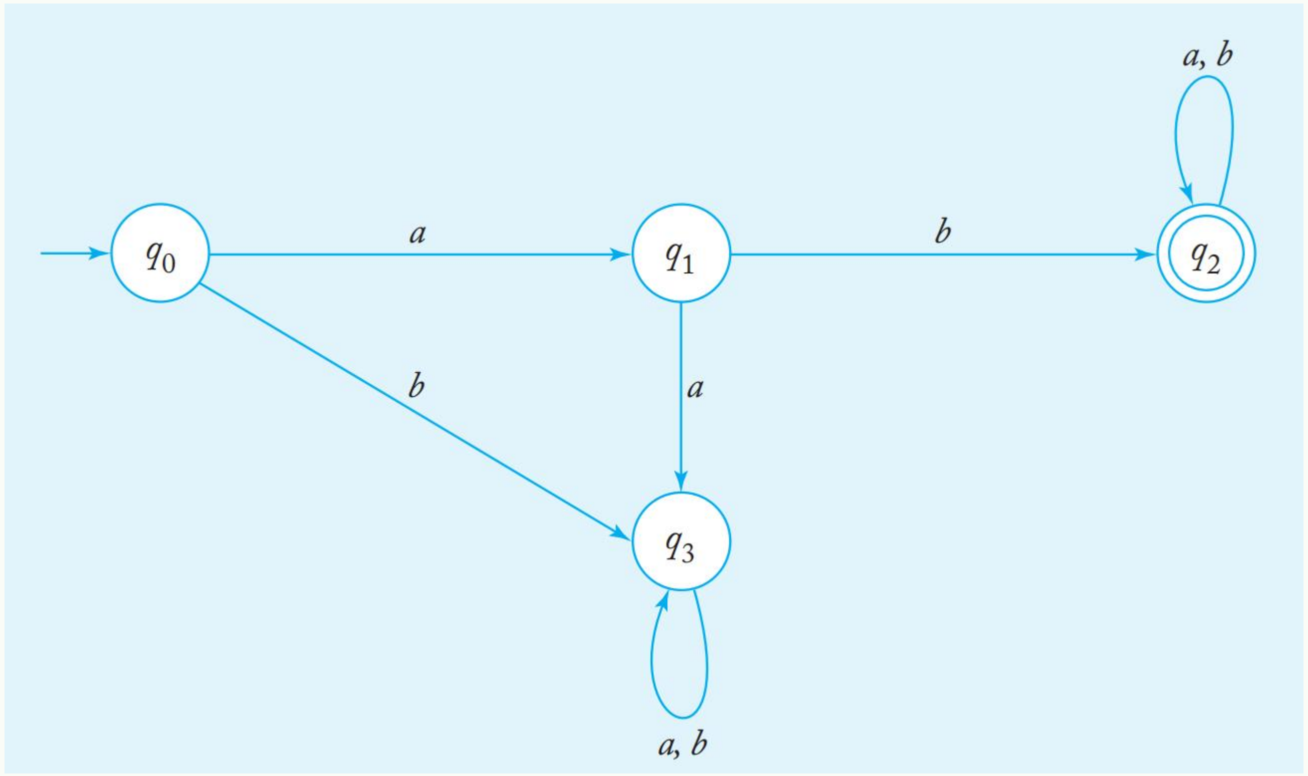
EXAMPLE 2.3

- Find a deterministic finite accepter that recognizes the set of all strings on Σ = {a, b} starting with the prefix ab.



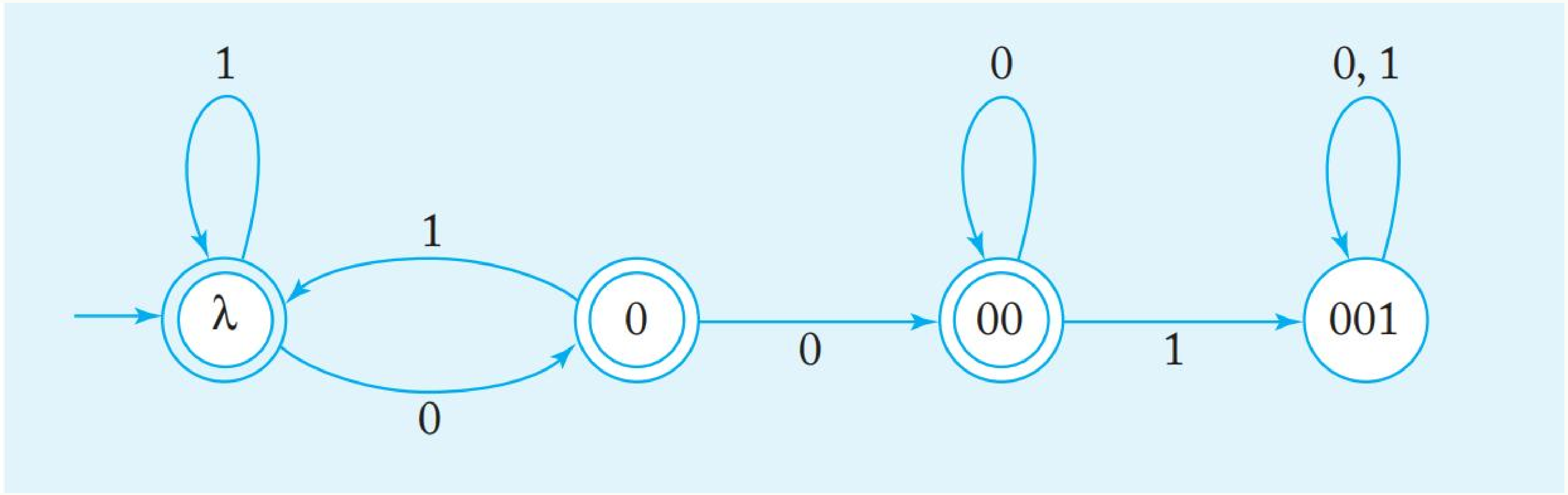
EXAMPLE 2.3

- Find a deterministic finite accepter that recognizes the set of all strings on Σ = {a, b} starting with the prefix ab.



EXAMPLE 2.4

- Find a dfa that accepts all the strings on {0, 1}, except those containing the substring 001.



Regular Languages

- Every finite automaton accepts some language.
- If we consider all possible finite automata, we get a set of languages associated with them. We will call such a set of languages a **family**.
- **DEFINITION 2.3**

  A language L is called regular if and only if there exists some deterministic finite accepter M such that

  $$L = L(M).$$

EXAMPLE 2.5

- Show that the language

$$L = \{awa : w \in \{a, b\}*\}$$

is regular.

To show that a language is regular, all we have to do is find a dfa for it.

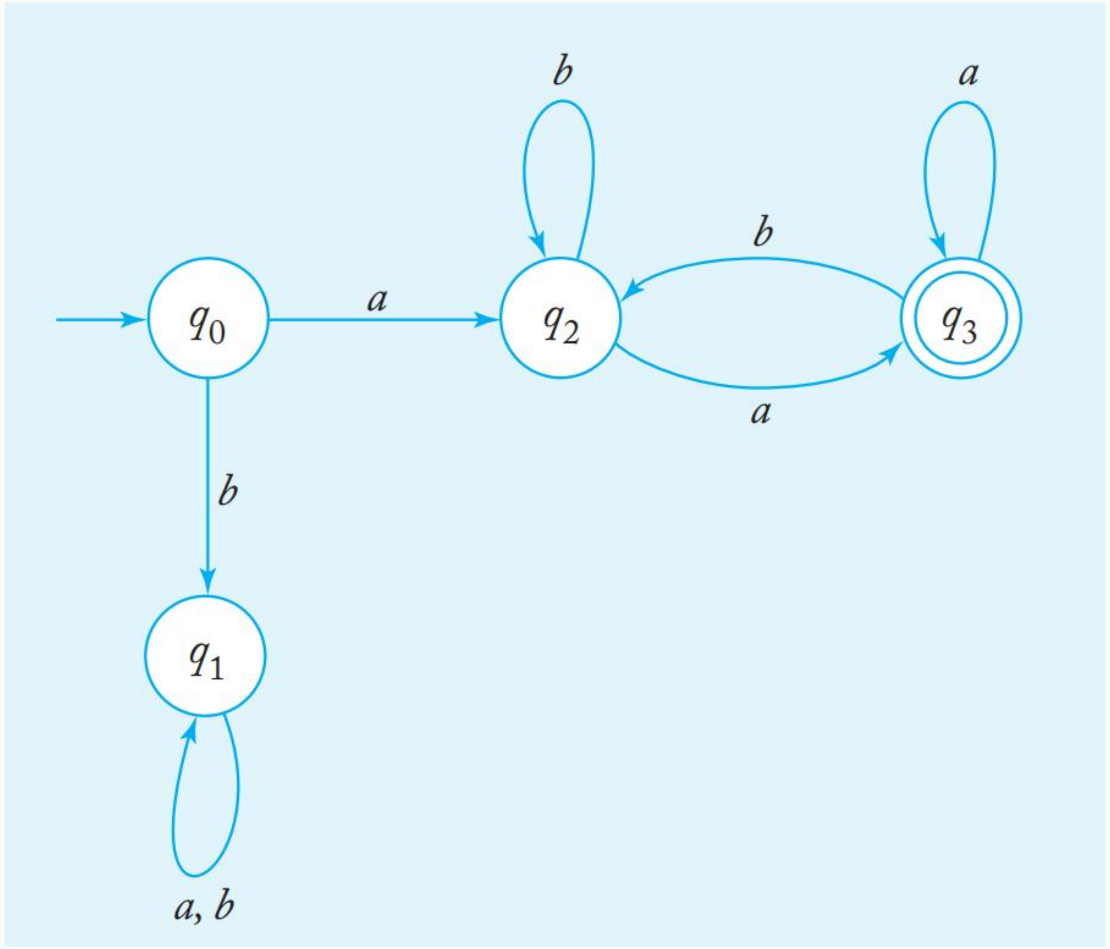
EXAMPLE 2.5

- Show that the language

$$L = \{awa : w \in \{a, b\}^*\}$$

is regular.

To show that a language is regular, all we have to do is find a dfa for it.



EXAMPLE 2.6

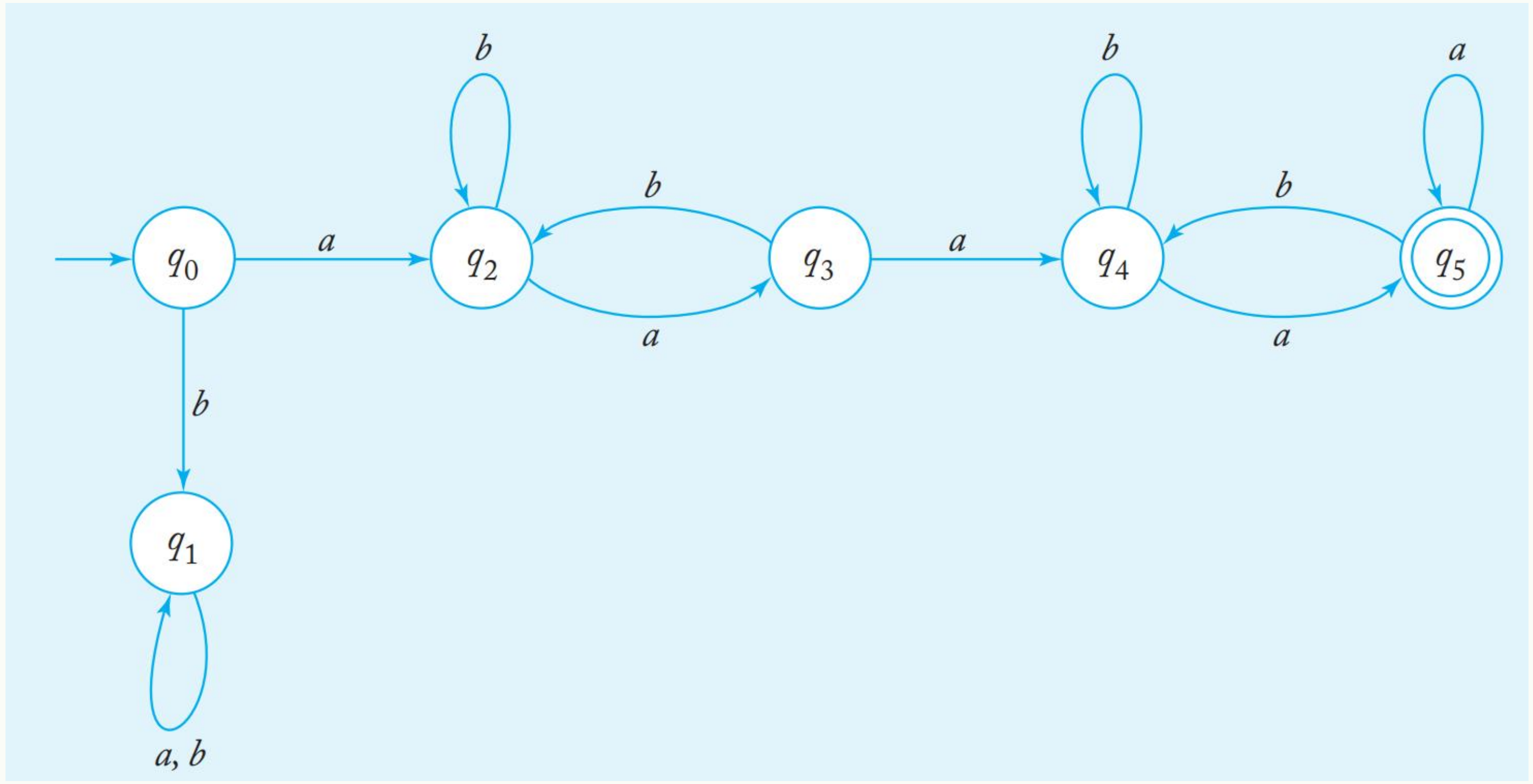
- Let L be the language in Example 2.5. Show that L^2 is regular. We can write an explicit expression for L^2, namely,

$$L^2 = \{aw_1aaw_2a : w_1, w_2 \in \{a, b\}^*\}$$

EXAMPLE 2.6

- Let L be the language in Example 2.5. Show that L^2 is regular. We can write an explicit expression for L^2, namely,

$$L^2 = \{aw_1aaw_2a : w_1, w_2 \in \{a, b\}^*\}$$



END