Chapter 2 Finite Automata

Syntax and Semantics

DFA – Deterministic FA

NFA – Nondeterministic FA

Equivalence of DFA and NFA

Syntax and Semantics -1

a statement from **Terence Parr**

- A ***language*** ***i****s a set of valid sentences. What makes a sentence valid?*
- *You can break validity down into two things: syntax and semantics.*
  - ***syntax*** *refers to grammatical structure*
  - ***semantics*** *refers to the meaning of the vocabulary symbols arranged with that structure.*

Syntax and Semantics - 2

- *Grammatical (**syntactically valid**) does not imply sensible (**semantically valid**)*
- *For example, the grammatical sentence "cows flow supremely" is grammatically ok (subject verb adverb) in English, but makes no sense.*
- *Similarly, in a programming language, your grammar (syntax rules) may allow but the language may only allow the meaningful sentence (a semantic rule).*

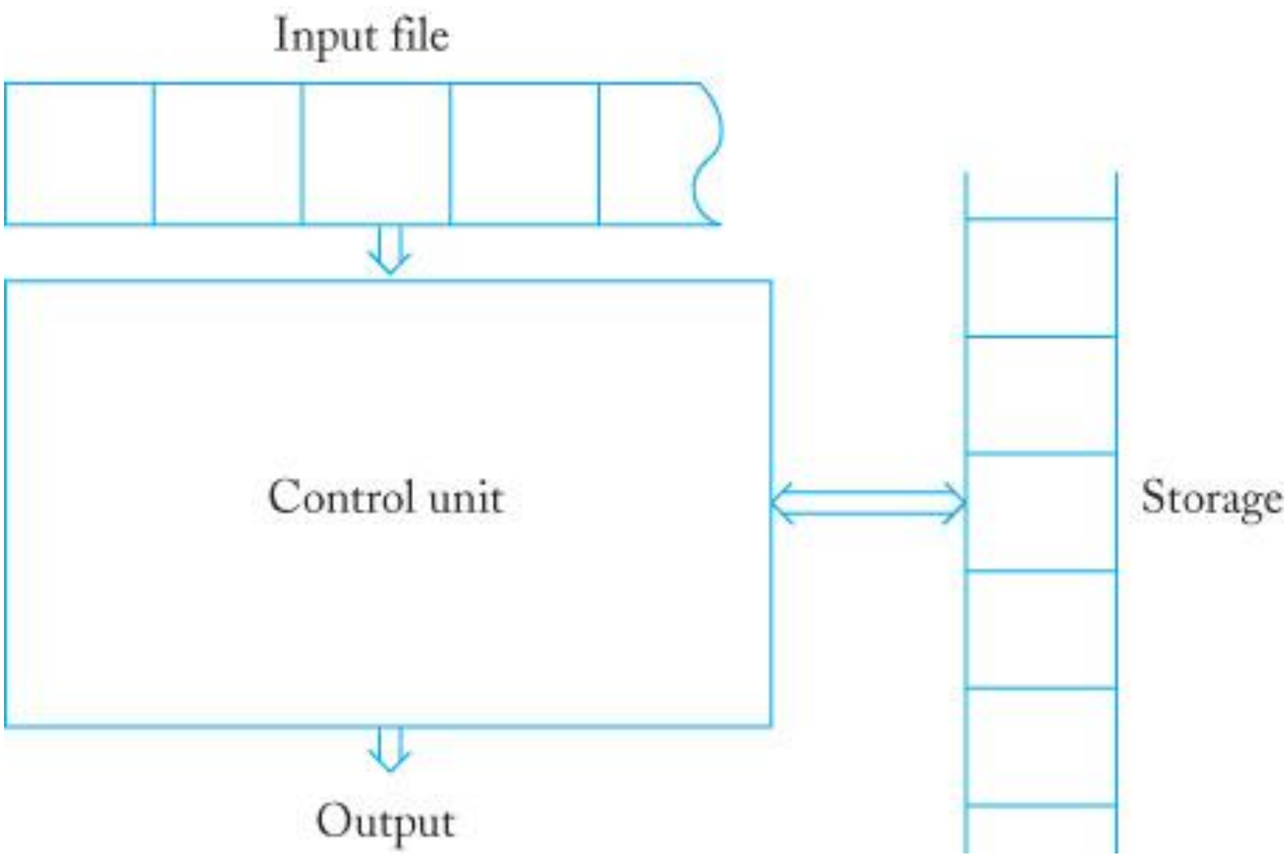
Syntax and Semantics - 3

- *When you write a grammar, you are specifying the set of syntax rules obeyed by your language.*
- *When you use this to generate a recognizer for sentences in that language. To apply semantic rules, you must add actions or semantic predicates to the grammar.*
- *The actions test the "value" of the various tokens and their relationships to determine semantic validity.*
  - *For example, if you look up a type name in a symbol table to ensure it's a type not a variable, you are applying a semantic rule.*

FA: a Representation of RL

- 4 representations of Regular Language (RL)
  - RE – regular expression
  - **FA – finite Accepters/Automata**
  - TG – a graph-based way of specifying patterns that can be represented by FA
  - RG – regular grammar

Automata: FA does not have storage



DFA and NFA

- DFA – a deterministic FA is convertible in a simple way into a program that can serve as a recognizer for RL
- NFA – a nondeterministic FA is more powerful in design stage of a recognizer
- Every NFA can be converted into a DFA (see sec. 2.3)

DFA

- Definition 2.1

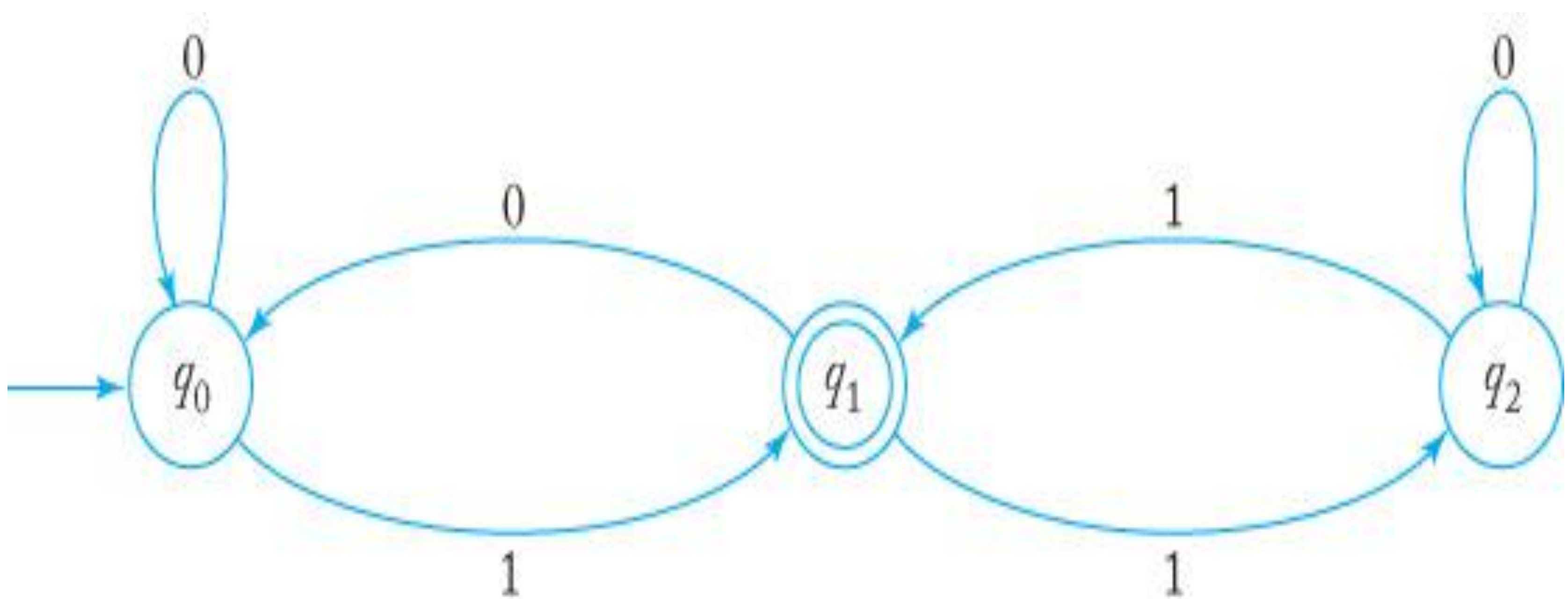
  A **deterministic finite accepter** or **dfa** is defined by the quintuple $M = (Q, \Sigma, \delta, q_0, F)$ where

  - Q is a finite set of **internal states**
  - Σ is a finite set of symbols called the **input alphabet**
  - $\delta: Q \times \Sigma \to Q$ is a total function called the **transition function**
  - $q_0 \in Q$ is the **initial state**
  - $F \subseteq Q$ is a set of **final states**

Transition Graph

- TG – a graphic notation for FA
  - **Example 2.1** and Figure 2.1 (page 39)
  - $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$
  - δ is given by
    - $\delta(q_0, 0) = q_0$, $\delta(q_0, 1) = q_1$,
    - $\delta(q_1, 0) = q_0$, $\delta(q_1, 1) = q_2$,
    - $\delta(q_2, 0) = q_2$, $\delta(q_2, 1) = q_1$,

Figure 2.1
TG for example 2.1



Languages and DFA's

- Definition 2.2 (page 40)
  - The language accepted by a dfa $M = (Q, \Sigma, \delta, q_0, F)$ is the set of all strings on alphabet accepted by M. In formal notation
    - $L(M) = \{w \in \Sigma^*: \delta^*(q_0, w) \in F\}$

Figure 2.2
A dfa with trap state, multiple label edge

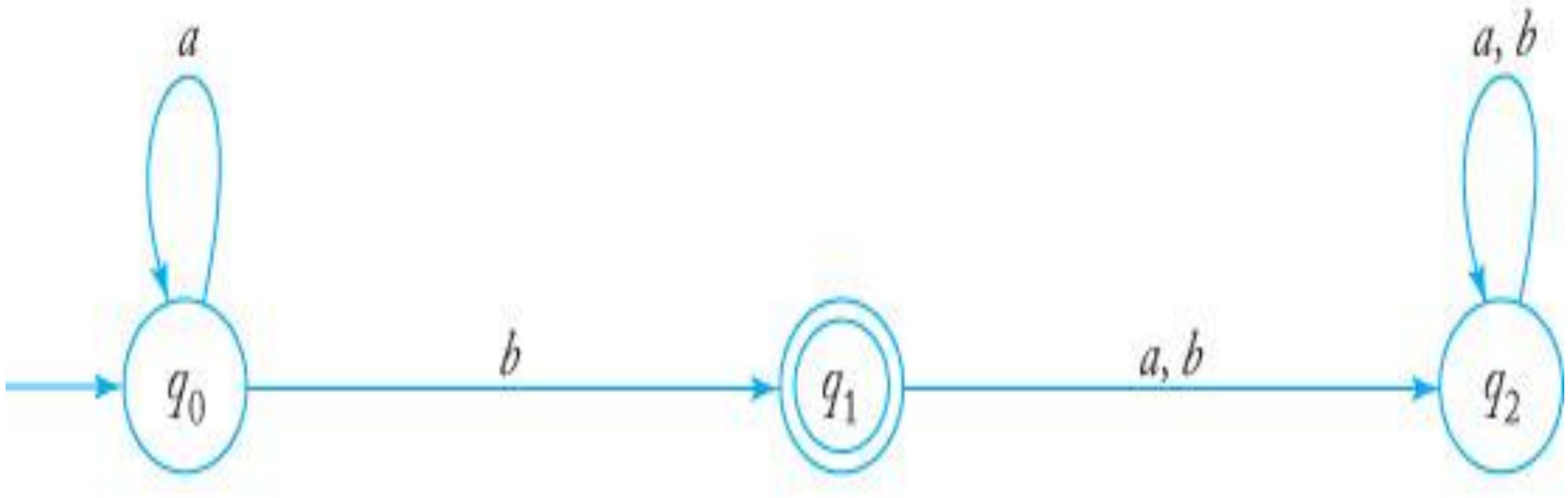
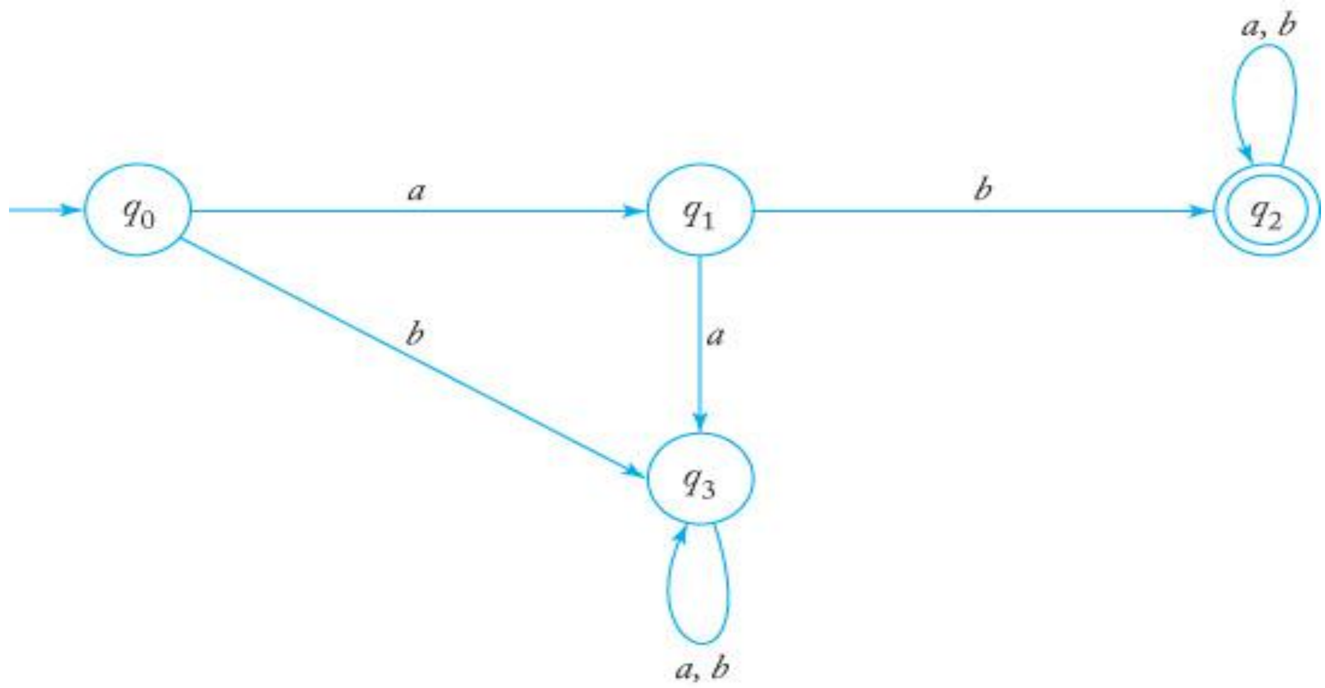


Figure 2.3
Table representation of dfa in Figure 2.2

	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

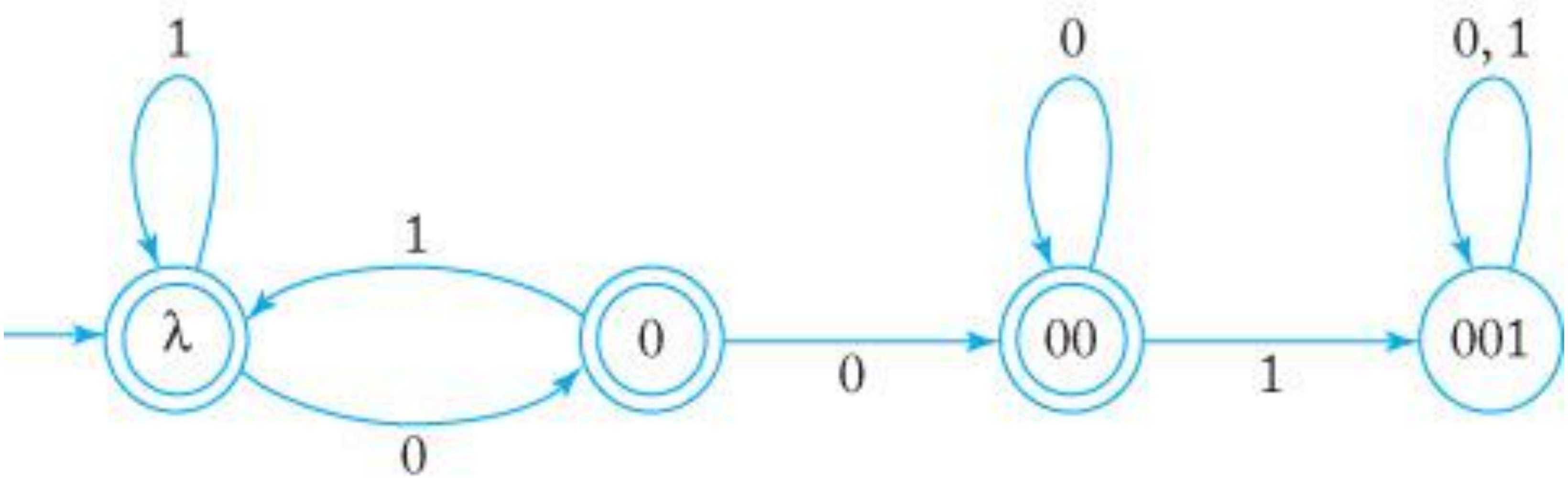
Example 2.3
dfa design technique

- Find a dfa that recognizes all strings starting with ab.



Example 2.4

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

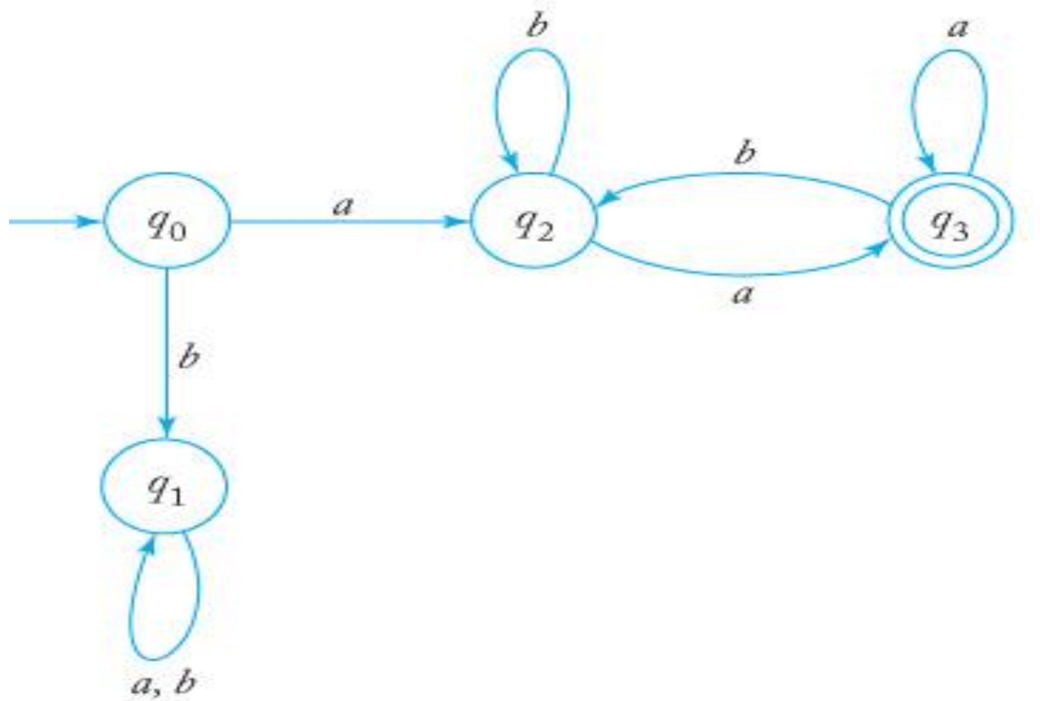


Regular Languages

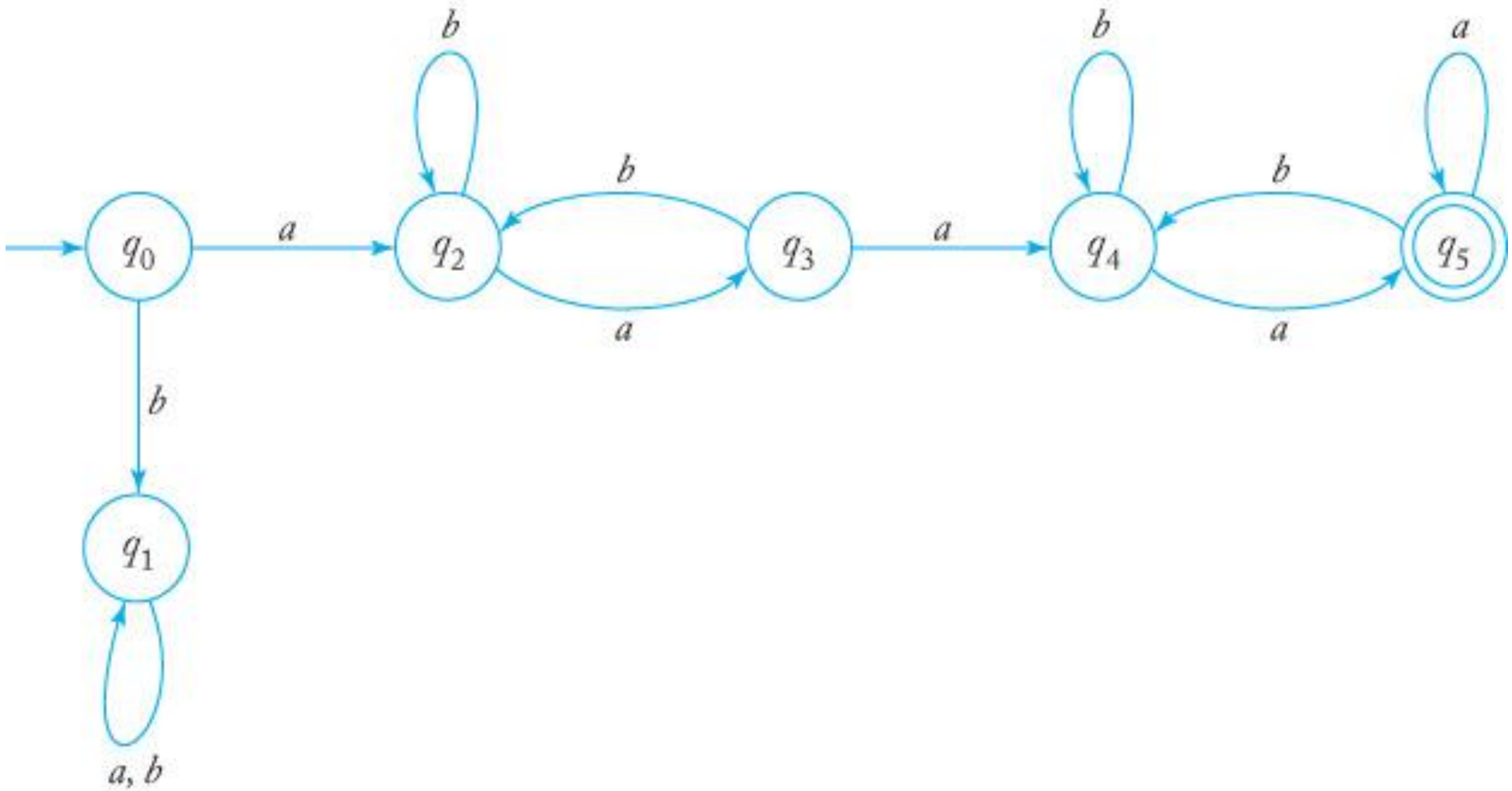
- The family of languages that is accepted by DFA is quite limited
- Definition 2.3
  - **A language L is called regular** if and only if there exists some dfa M such that
    - L = L(M)
- In class exercise to show a language is regular (see example 2.5)

Example 2.5
Show that the language L is regular

- Show that L = {awa: w ∈ {a, b}* } is regular



Example 2.6
If L is regular, show L^2 is regular too



Nondeterministic FA

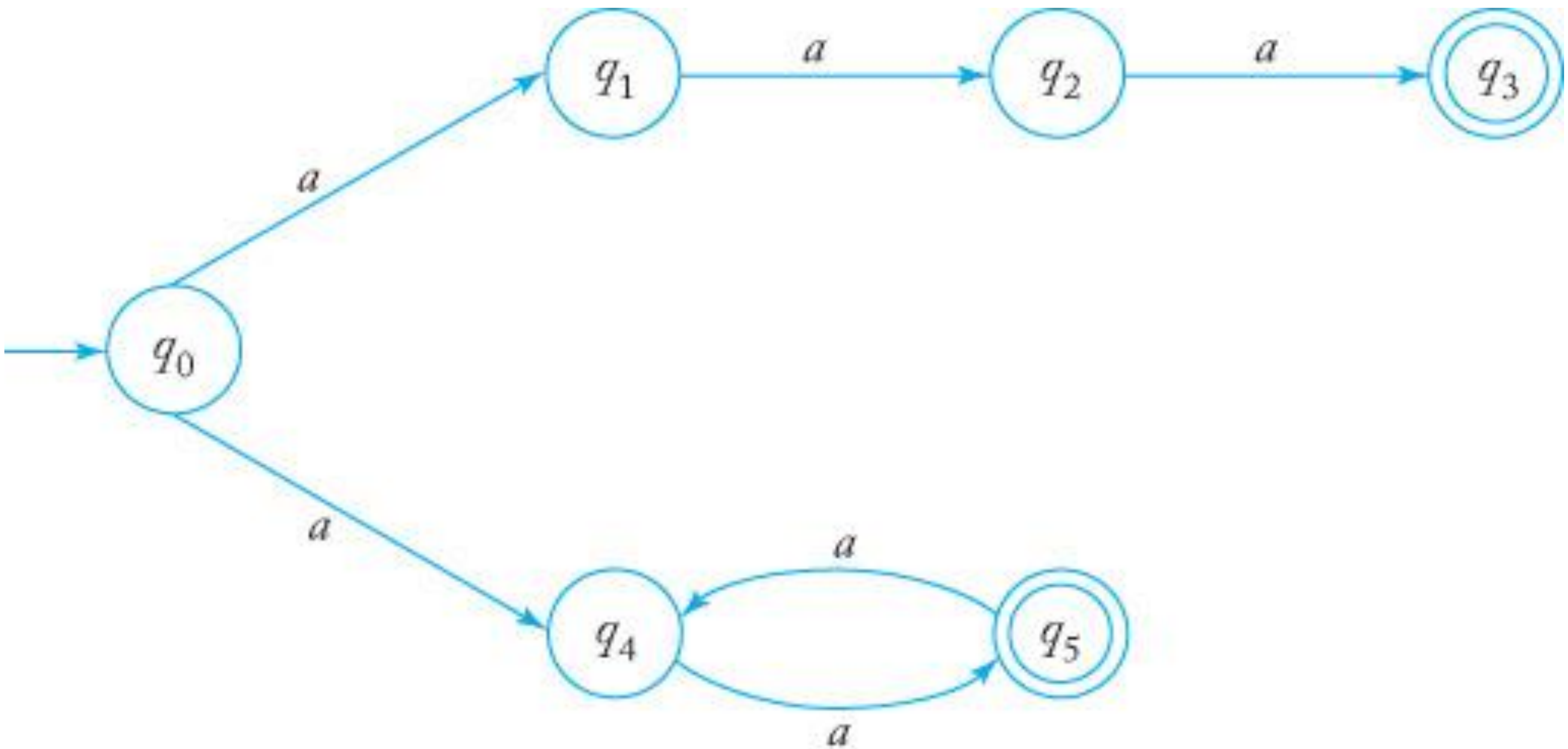
- Nondeterminism is an effective mechanism for describing some complicated languages concisely
- Definition 2.4 NFA (page 49)
  - NFA is defined by the quintuple $M = (Q, \Sigma, \delta, q_0, F)$
  - Almost same like DFA with the only difference in transition function
  - $\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$

NFA and its TG notation

- In class exercise – show the definition difference of DFA and NFA
  - 2.2 #2 (page 55)
- TG – Transition graph
- Fig 2.8 and Fig 2.9 (page 50 - 51)

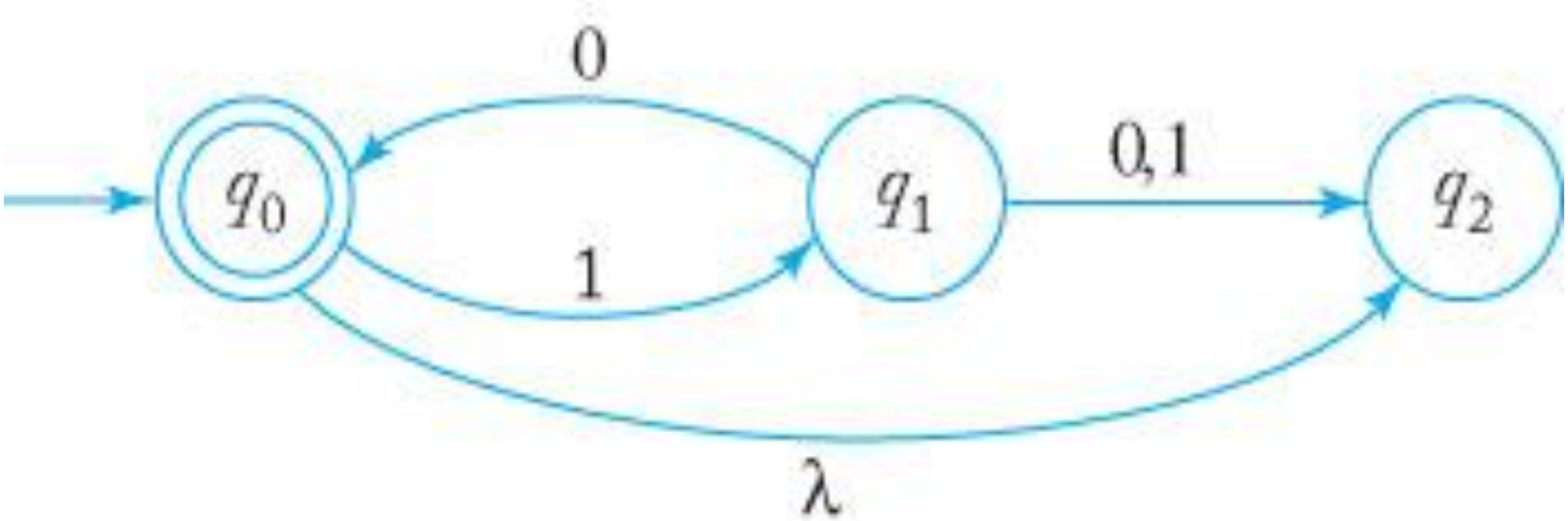
Example 2.7, Fig. 2.8

nfa with 2 transitions labeled the same



Example 2.8, Fig. 2.9

nfa: unspecified transition (to empty set) and λ transition



An Application of NFA

- Example 1: NFA that is designed to accept only strings in the form of a signed or unsigned decimal number
  - Accept strings:
    - -15.
    - 75.38
    - +.oo2
  - Reject strings:
    - 3..14
    - +17-56

Examples 2.9 extended notation *

- Extended transition function notation * to allow you describe several applications of transition function in one compact expression
- Example 2.9 (page 51-52)
  - Write out the extended transition function for a new word, say aaa

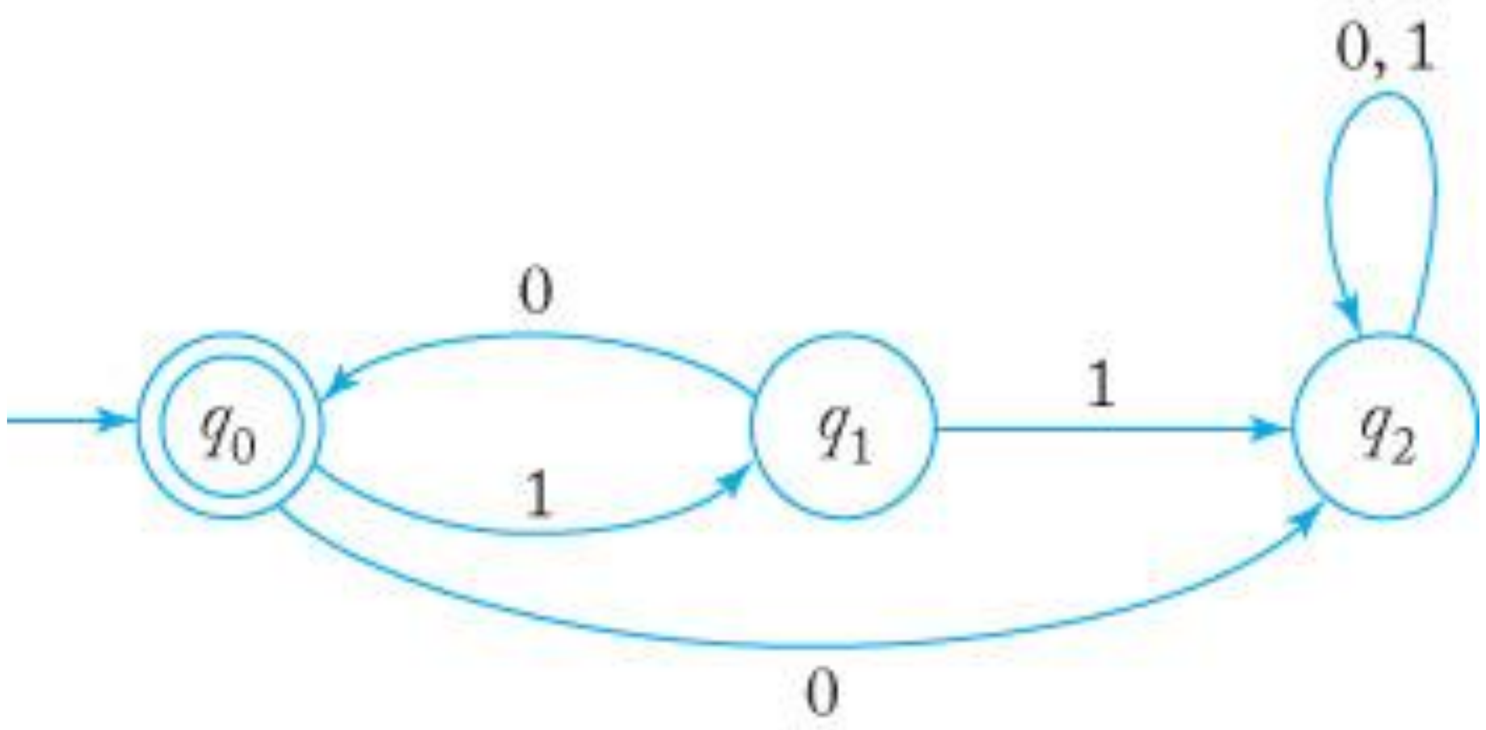
Languages and NFA's

- Definition 2.6 (page 53)
  - The language accepted by an **nfa** M = (…) is defined as the set of all strings accepted by M. formally,
  - $L(M) = \{w \in \Sigma^*: \delta^*(q_0, w) \cap F \neq \emptyset \}$
  - The language consists of all strings w for which there is a walk labeled w from the initial vertex of the transition graph to some final vertex.
  - What is complement of L(M)?

Equivalence of DFA and NFA

- Definition 2.7 (page 56)
  - Two finite accepters M_1 and M_2 are said to be equivalent if $L(M_1) = L(M_2)$, that is, if they both accept the same language
- Example 2.11
  - $L = \{(10)^n : n \geq 0\}$ (page 57)
  - There are many dfa or nfa for a given language
  - Fig 2.11 dfa for L
  - Can you draw nfa for L?

Example 2.11, Fig 2.11

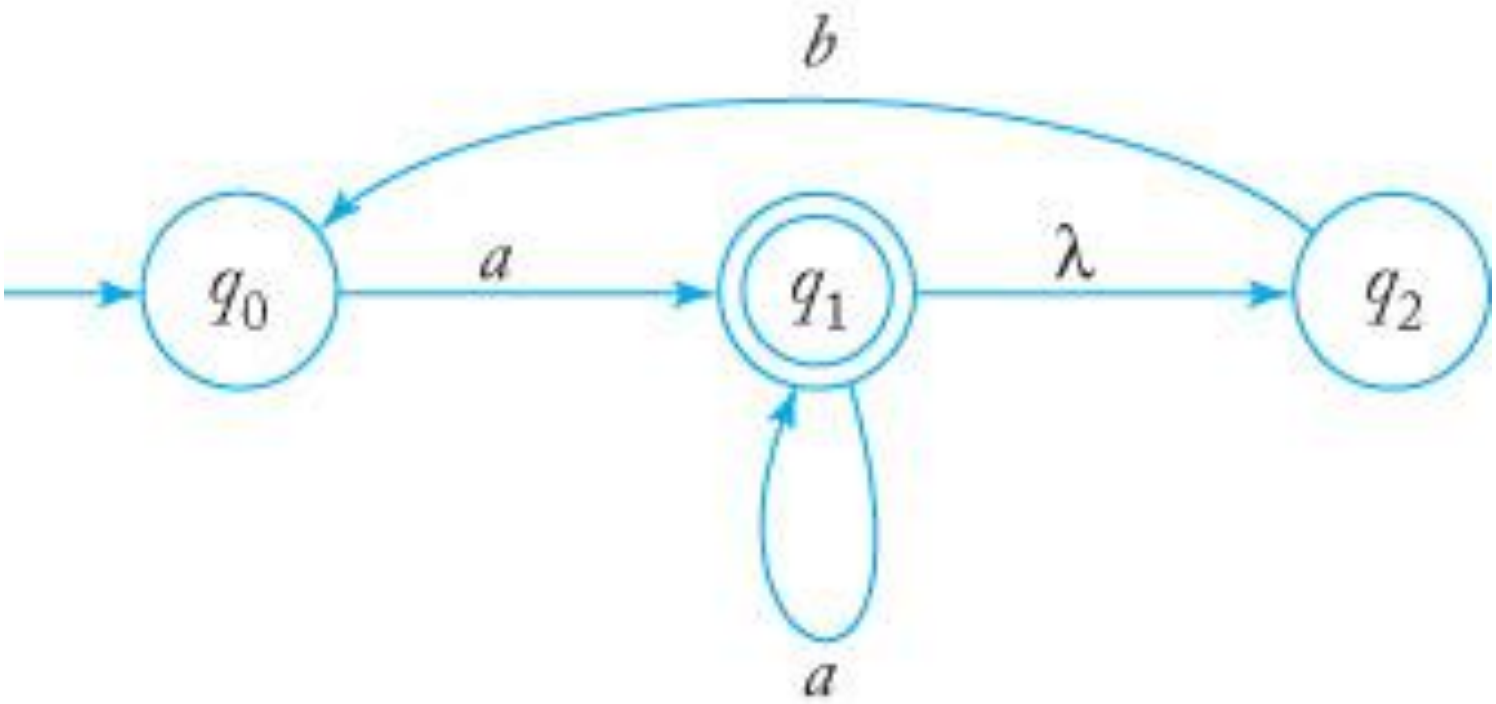


Review of Power Set

- For $A = \{0, 1\}$, $2^A =?$
- For $B = \{0, 1, 2\}$, $2^B =?$
- For $C = \{q_0, q_1, q_2\}$, $2^C =?$

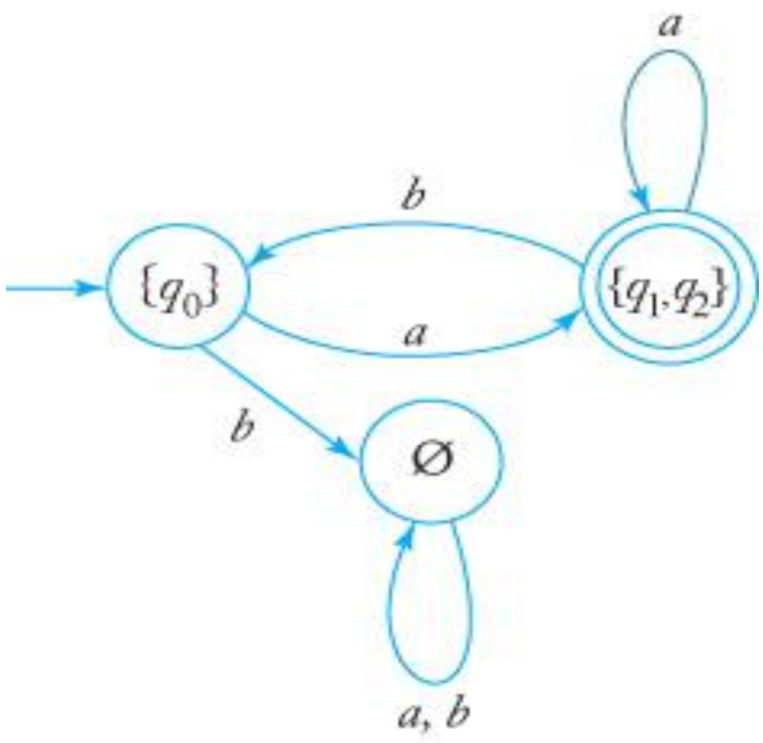
Example 2.12, Fig. 2.12

Convert the nfa to an equivalent dfa



Example 2.12, Fig 2.13

The resulted dfa



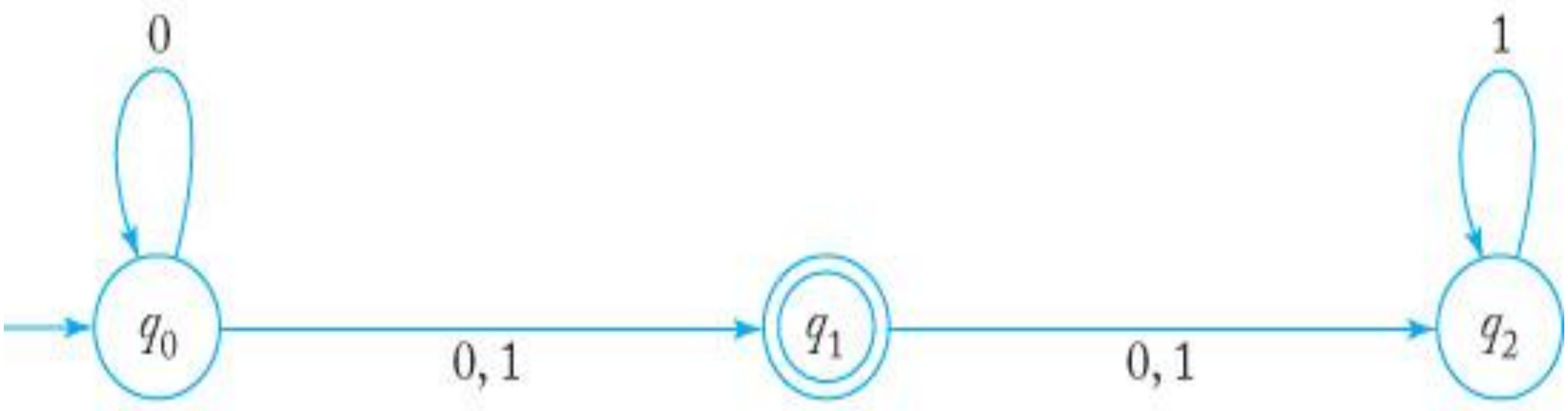
Procedure: nfa-to-dfa

- Theorem 2.2 (page 59)
  - Let L be the language accepted by a nfa M_N. Then there exists a dfa M_D such that
    - $L = L(M_D)$
  - Proof of Theorem 2.2 is a constructive proof which shows the procedure of transforming nfa to dfa step by step
  - The key of the procedure is to use the power set of Q_N to be Q_D in construction of the new dfa M_D

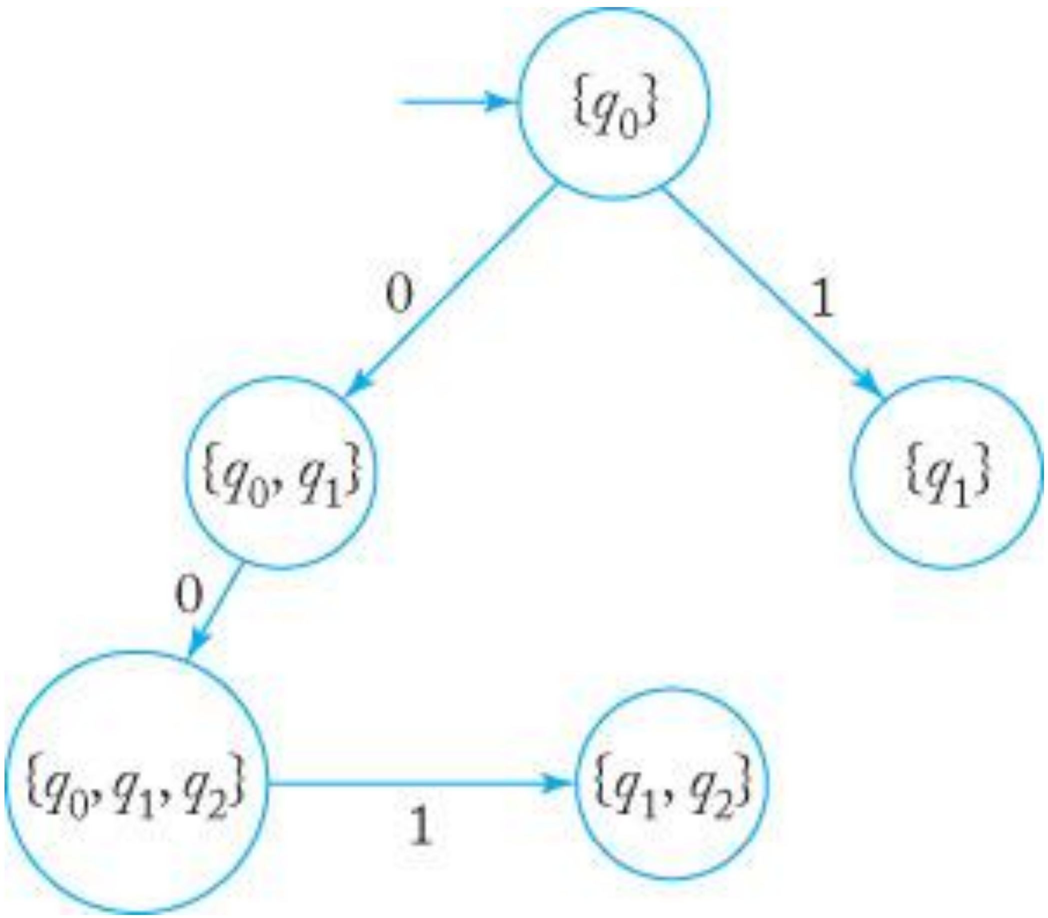
Another example of nfa-to-dfa

- Example 2.13 (page 60 – 61)
  - If you are still not sure about how the procedure nfa-to-dfa works, check out this example
  - It is important to know that every language accepted by an nfa is regular
  - What is a regular language?

Example 2.13, Fig. 2.14



Example 2.13, Fig. 2.15 applying procedure nfa-to-dfa



Example 2.13, Fig. 2.16 complete solution