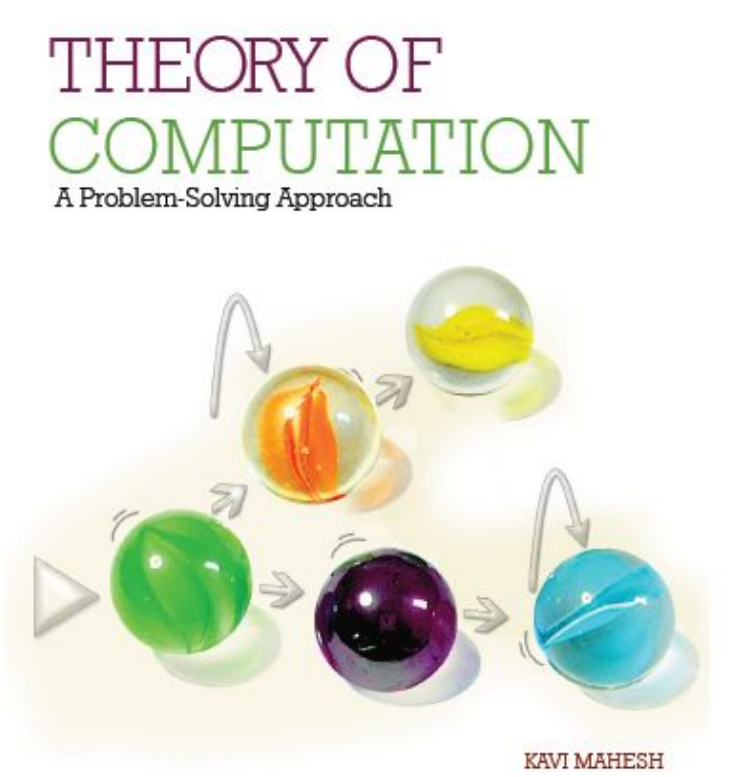


Theory of Computation: A Problem-Solving Approach

Chapter – 3: Non-Deterministic Finite Automata

Dr. Kavi Mahesh

Learning Objectives

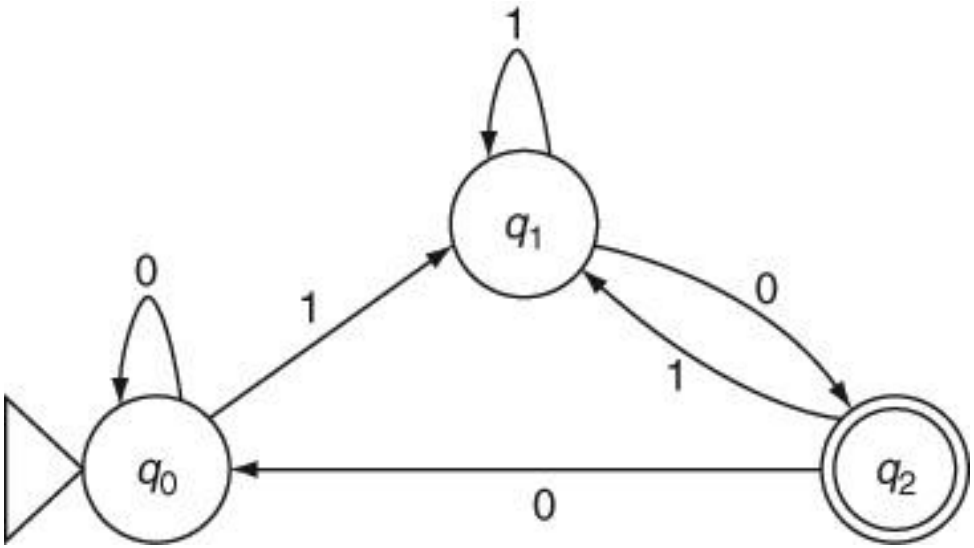
- Learn to use non-determinism as a tool in designing automata.
- Learn how to design non-deterministic automata for a given problem.
- Learn to convert a non-deterministic automaton to a deterministic one through subset construction.
- Learn the use of λ-transitions in designing non-deterministic automata.
- Learn to minimize the states in an automaton.
- Learn to compare two automata to determine their equivalence.
- Understand how finite state transducers work.

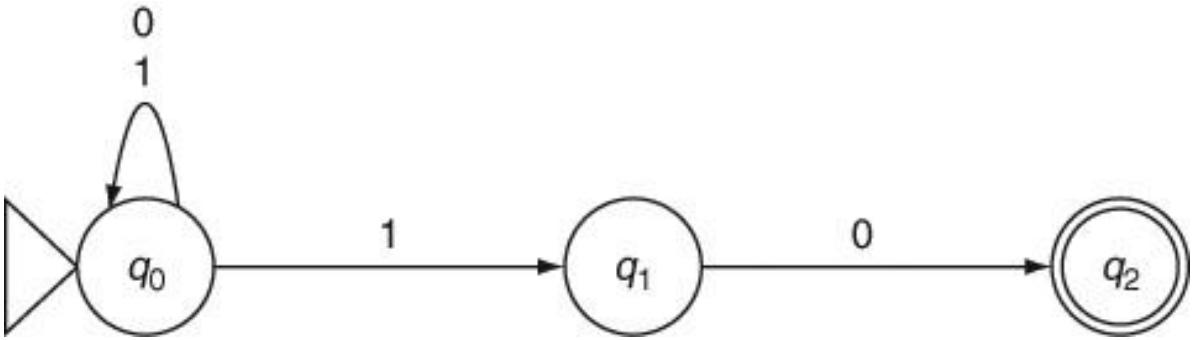
The Idea of Non-Determinism

- More than one choice for the computing machine
- More than transition for the same current state and the same input symbol
- More than one resulting state
- Machine stays simultaneously in multiple states
- Any one path leads to a final state: machine accepts input
- Two ways to understand non-deterministic behavior:
  - Multiple parallel threads of execution
  - Machine guessing the right choice each time!

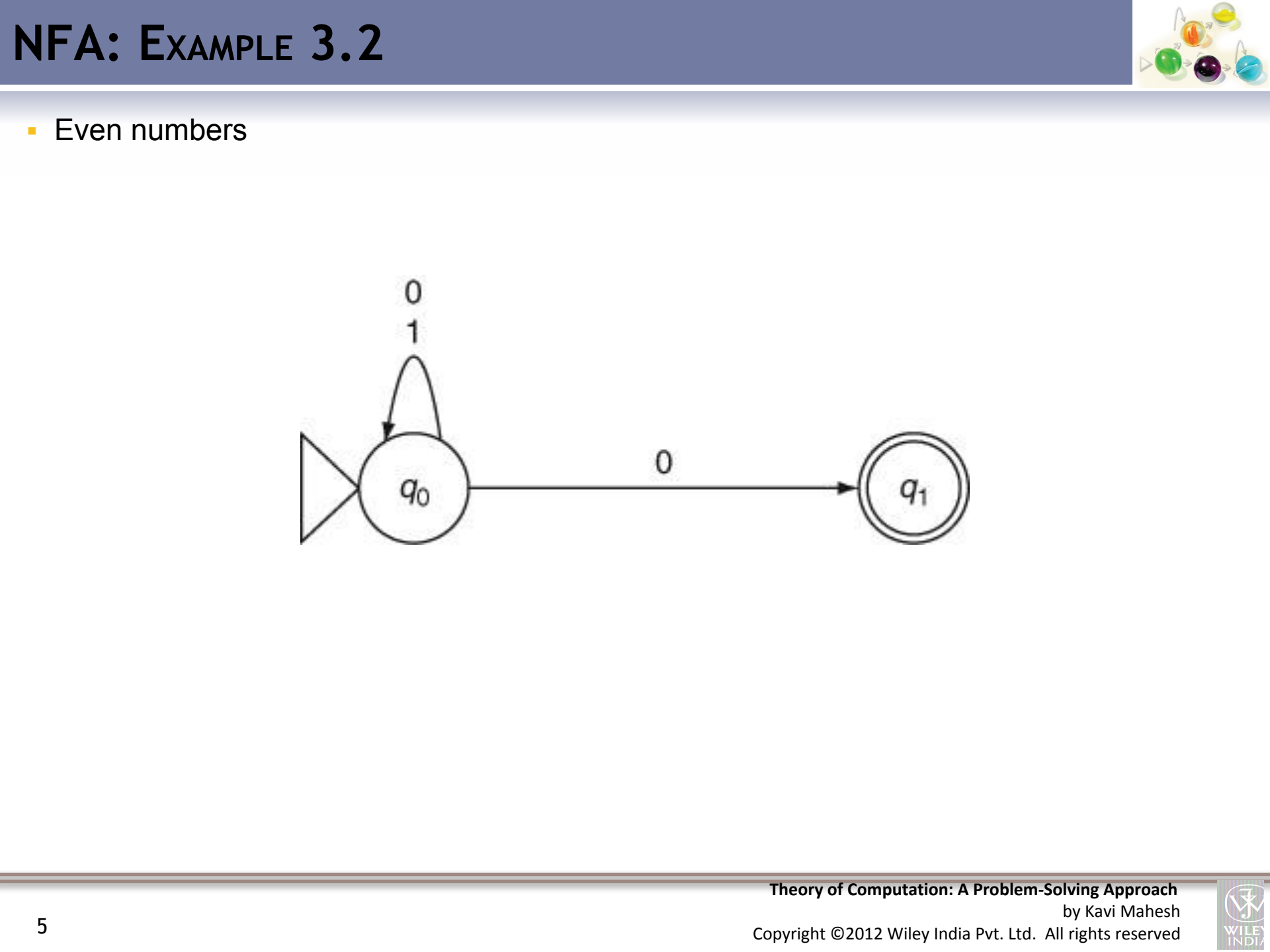
EXAMPLE 3.1: DFA AND NFA

- Strings ending with 10





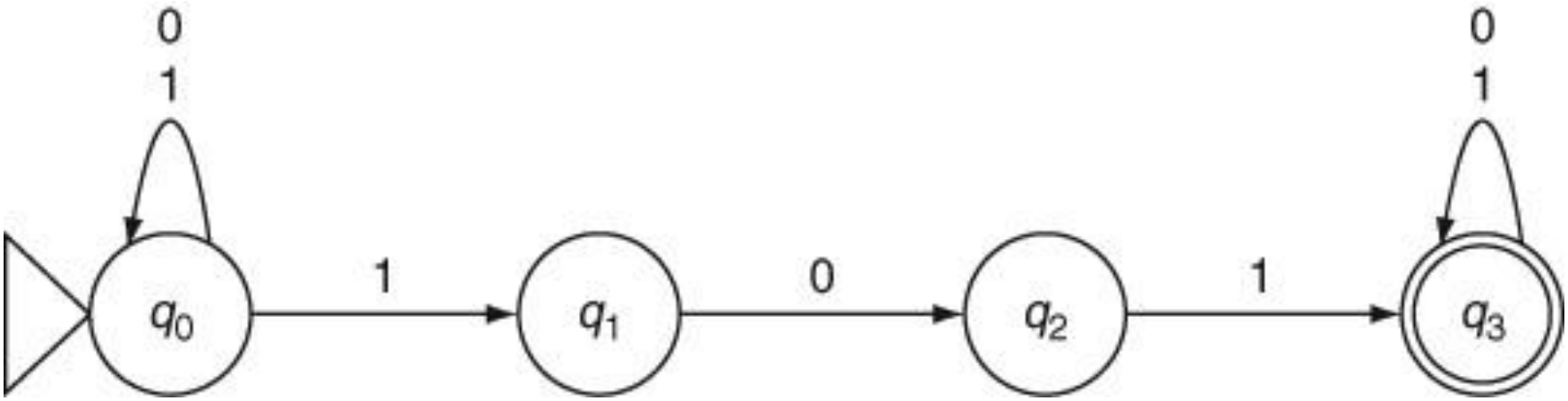
NFA: Example 3.2



- Even numbers

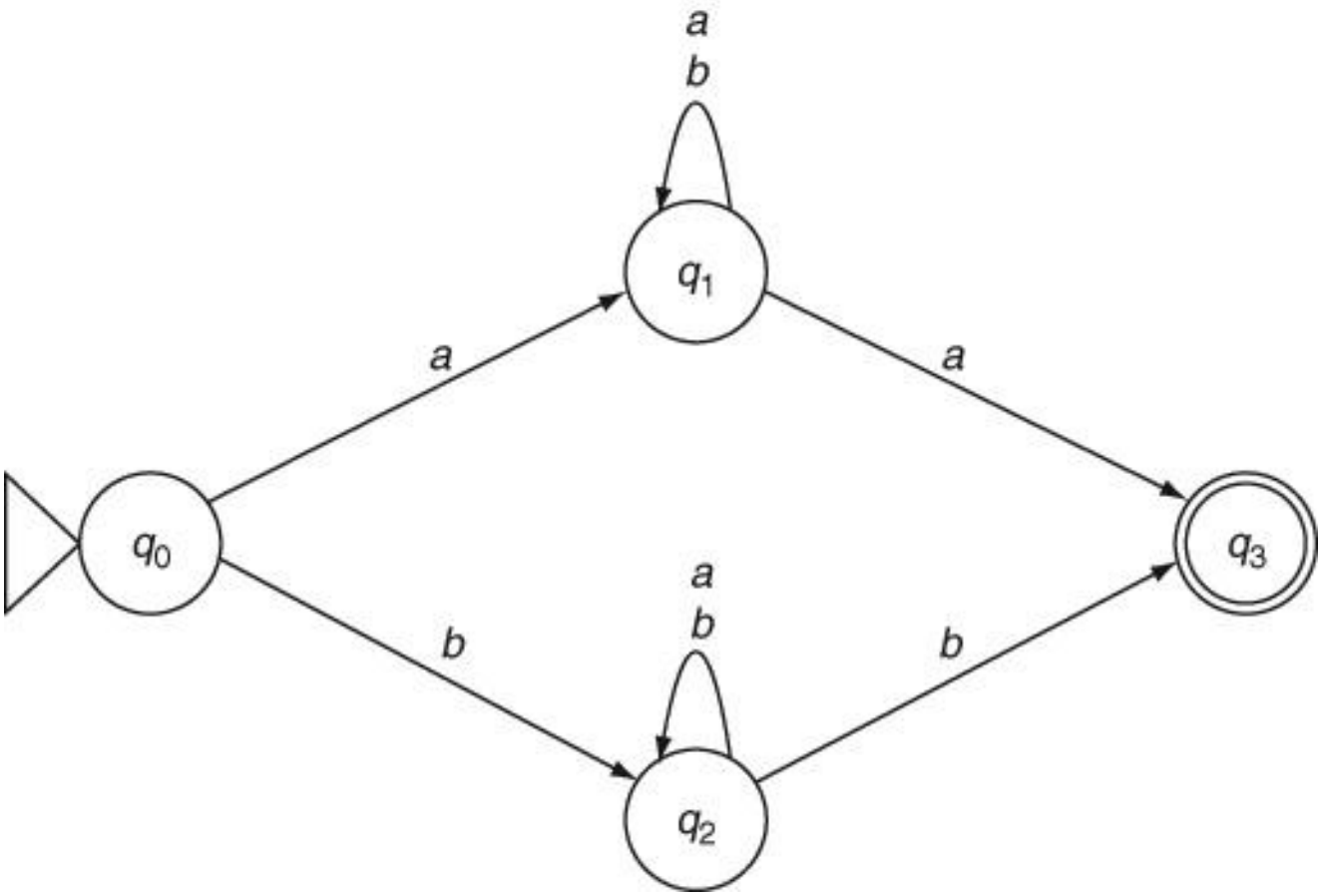
NFA: Example 3.3

- Searching for the keyword 101



NFA: Example 3.4

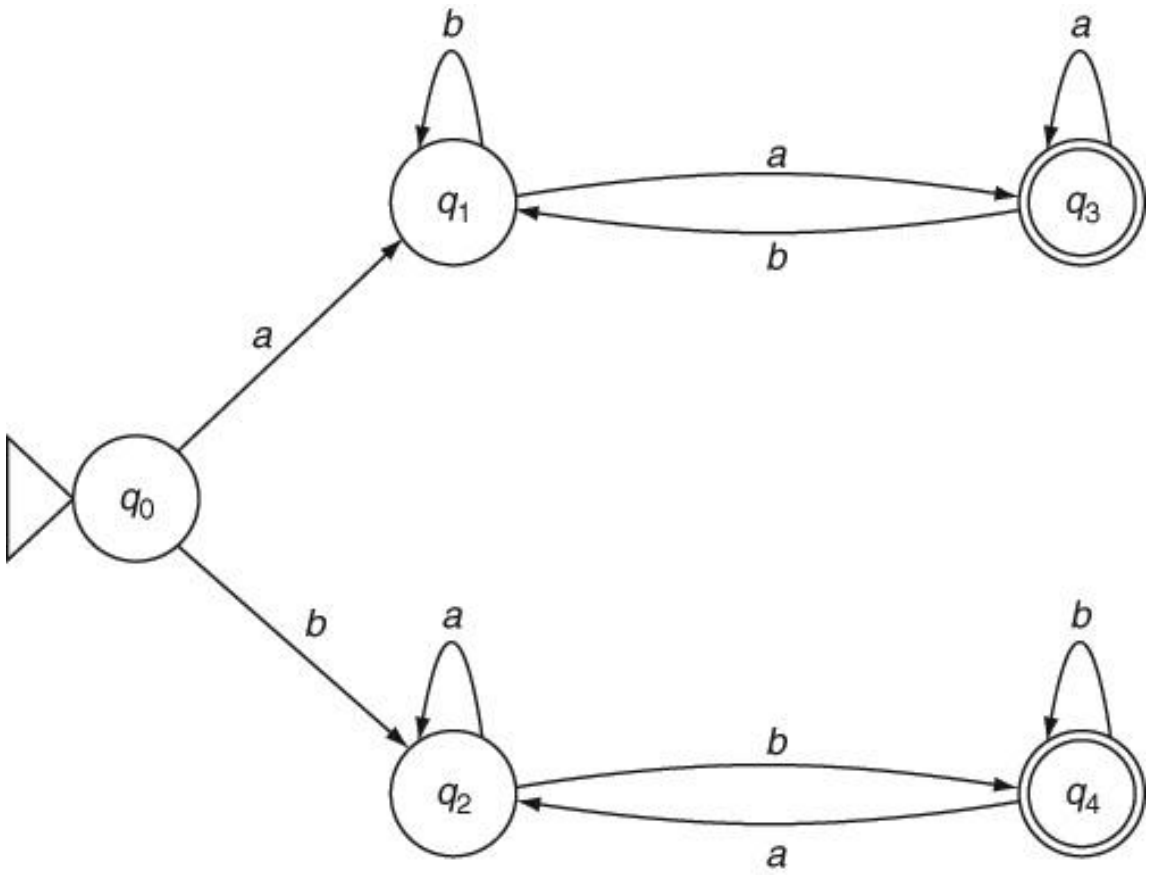
- Same first and last symbols



Theory of Computation: A Problem-Solving Approach
by Kavi Mahesh

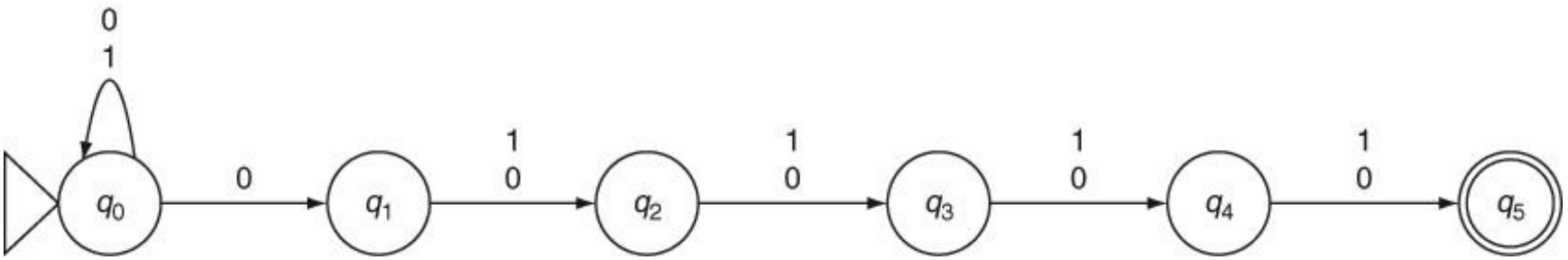
DFA: Example 3.4

- Same first and last symbols



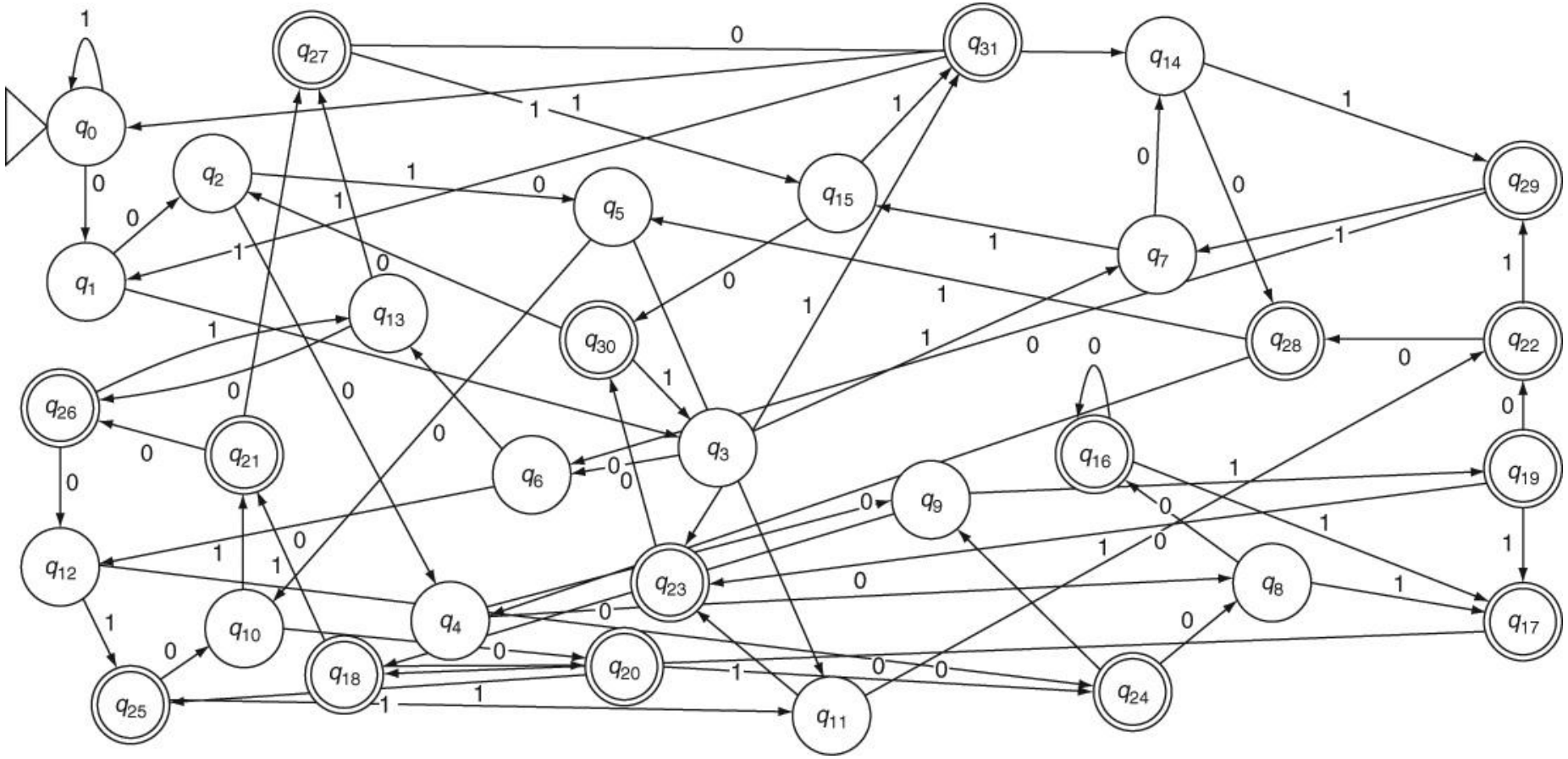
NFA: Example 3.5

- Fifth symbol from the end is a 0



DFA: Example 3.5 (also Example 2.14)

- Fifth symbol from the end is a 0



Advantages of NFA over DFA

- Non-determinism makes it easier to design and construct an automaton by ensuring that it is equally easy to handle constraints in any part of the input string without a need for backtracking in the case of incorrect decisions made by the automaton; and
- Non-determinism reduces the number of states (and transitions) in the automaton.

Eliminating Non-Determinism: NFA to DFA

Algorithm

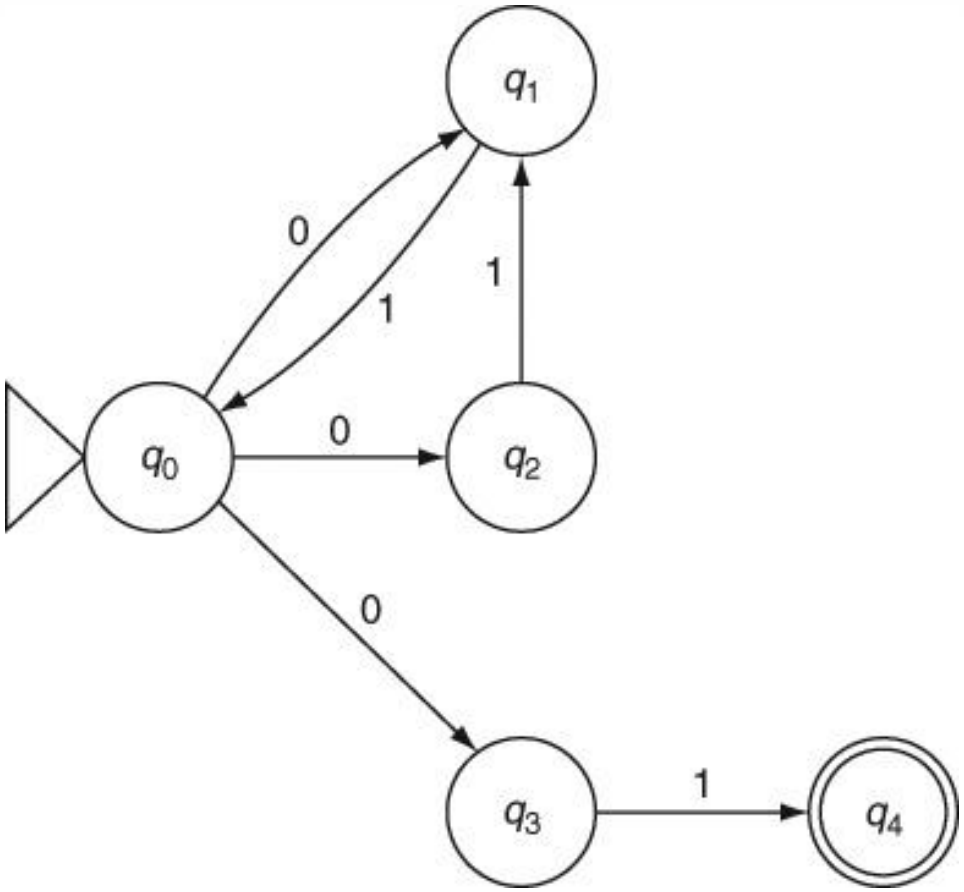
Starting from the initial state q_0 of the NFA,

- For each symbol in the alphabet, repeat
  - Determine the set of states that the NFA can reach from the current state (s) for the input symbol;

    If this is different from all the sets already enumerated,
    Create a new state Q_i in the DFA for this set.
    Add the new state to the set of states in the enumeration until no new set can e added.

- For each state Q in the DFA and for each symbol in the alphabet,
  - Add a transition to a DFA state that corresponds to any of the transitions in the NFA.

NFA to DFA: Example 3.6



NFA to DFA: Example 3.6

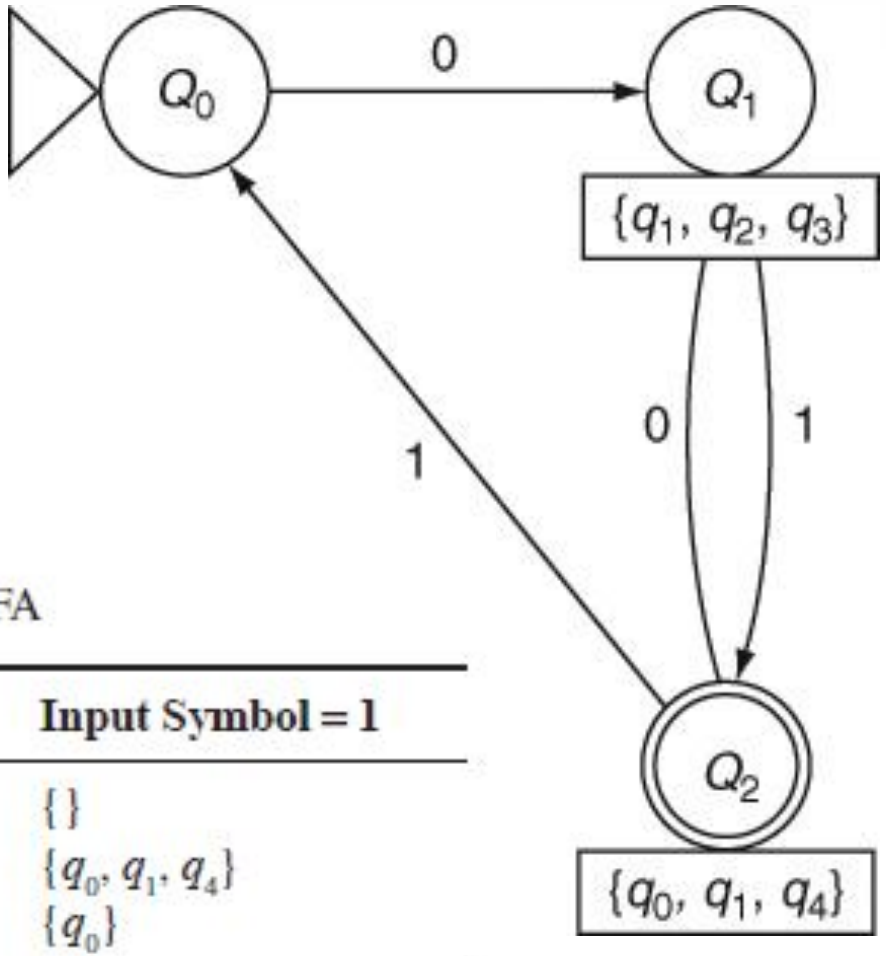
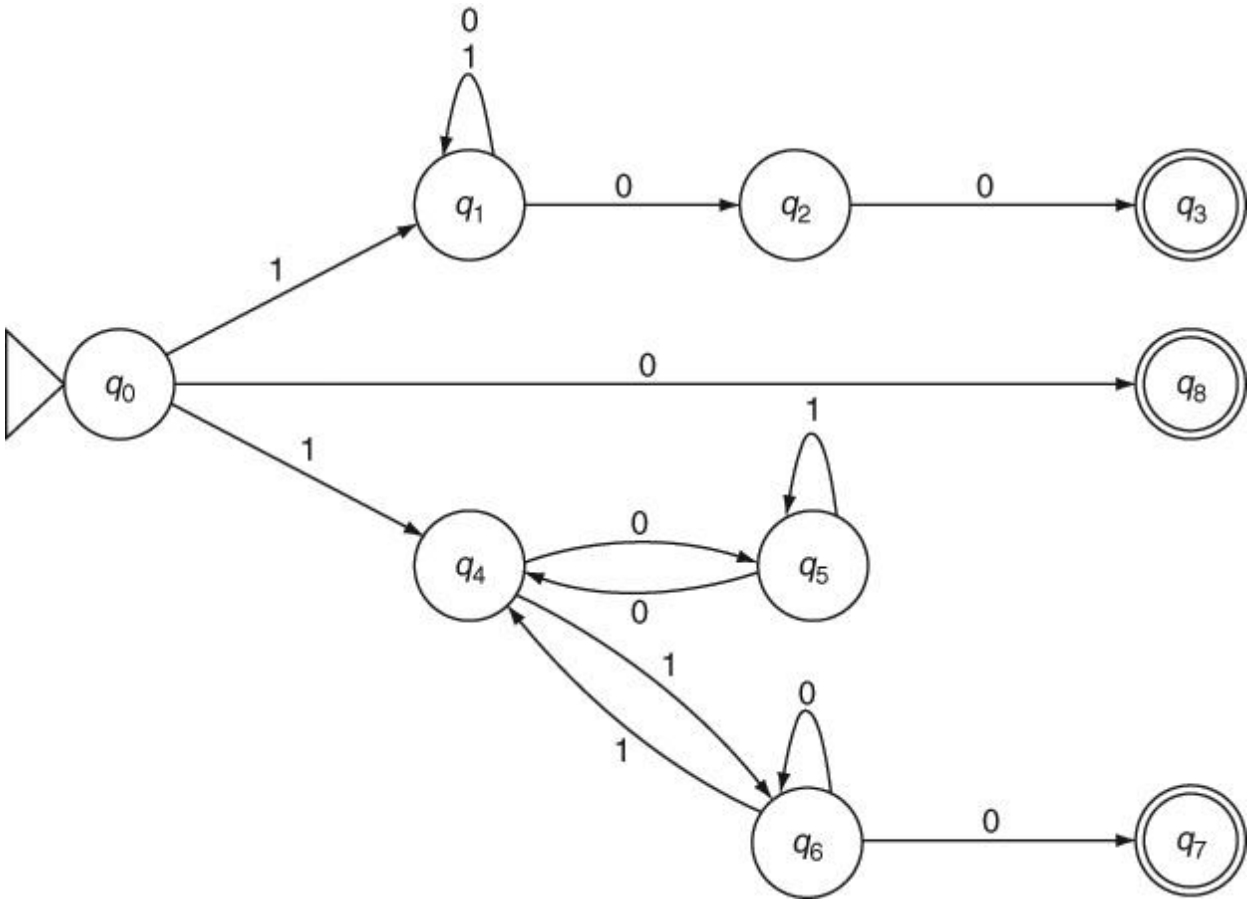


TABLE 3.1 Example of Converting NFA to DFA

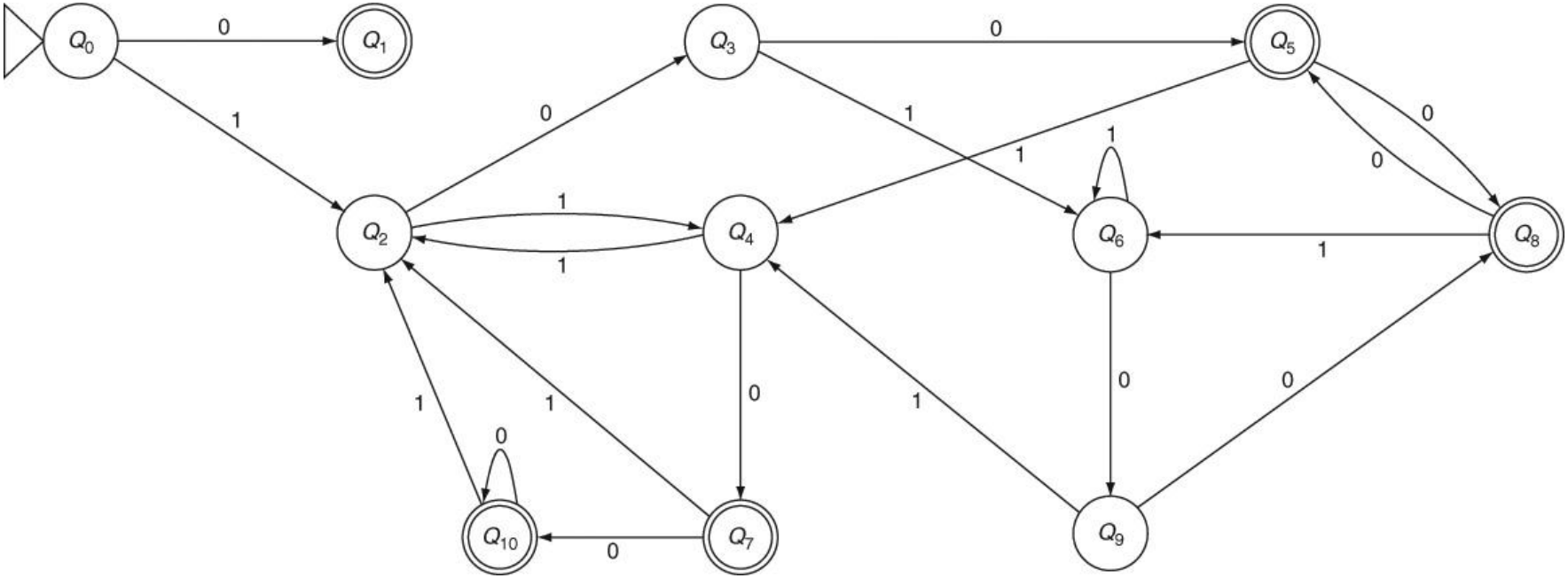
DFA State	Input Symbol = 0	Input Symbol = 1
$\rightarrow Q_0 = \{q_0\}$	$\{q_1, q_2, q_3\}$	$\{\}$
$Q_1 = \{q_1, q_2, q_3\}$	$\{\}$	$\{q_0, q_1, q_4\}$
$* Q_2 = \{q_0, q_1, q_4\}$	$\{q_1, q_2, q_3\}$	$\{q_0\}$

NFA TO DFA: EXAMPLE 3.7

- Binary numbers divisible by either 4 or 6



RESULTING DFA: EXAMPLE 3.7



Subset Construction: Example 3.7

TABLE 3.2 Converting NFA to DFA for Divisibility by 4 or 6

DFA State	Input Symbol = 0	Input Symbol = 1
$\rightarrow Q_0 = \{q_0\}$	$\{q_8\} = Q_1$	$\{q_1, q_4\} = Q_2$
$* Q_1 = \{q_8\}$	$\{\}$	$\{\}$
$Q_2 = \{q_1, q_4\}$	$\{q_1, q_2, q_5\} = Q_3$	$\{q_1, q_6\} = Q_4$
$Q_3 = \{q_1, q_2, q_5\}$	$\{q_1, q_2, q_3, q_4\} = Q_5$	$\{q_1, q_5\} = Q_6$
$Q_4 = \{q_1, q_6\}$	$\{q_1, q_2, q_6, q_7\} = Q_7$	$\{q_1, q_4\} = Q_2$
$* Q_5 = \{q_1, q_2, q_3, q_4\}$	$\{q_1, q_2, q_3, q_5\} = Q_8$	$\{q_1, q_6\} = Q_4$
$Q_6 = \{q_1, q_5\}$	$\{q_1, q_2, q_4\} = Q_9$	$\{q_1, q_5\} = Q_6$
$* Q_7 = \{q_1, q_2, q_6, q_7\}$	$\{q_1, q_2, q_3, q_6, q_7\} = Q_{10}$	$\{q_1, q_4\} = Q_2$
$* Q_8 = \{q_1, q_2, q_3, q_5\}$	$\{q_1, q_2, q_3, q_4\} = Q_5$	$\{q_1, q_5\} = Q_6$
$Q_9 = \{q_1, q_2, q_4\}$	$\{q_1, q_2, q_3, q_5\} = Q_8$	$\{q_1, q_6\} = Q_4$
$* Q_{10} = \{q_1, q_2, q_3, q_6, q_7\}$	$\{q_1, q_2, q_3, q_6, q_7\} = Q_{10}$	$\{q_1, q_4\} = Q_2$

Meanings of NFA States: Example 3.7

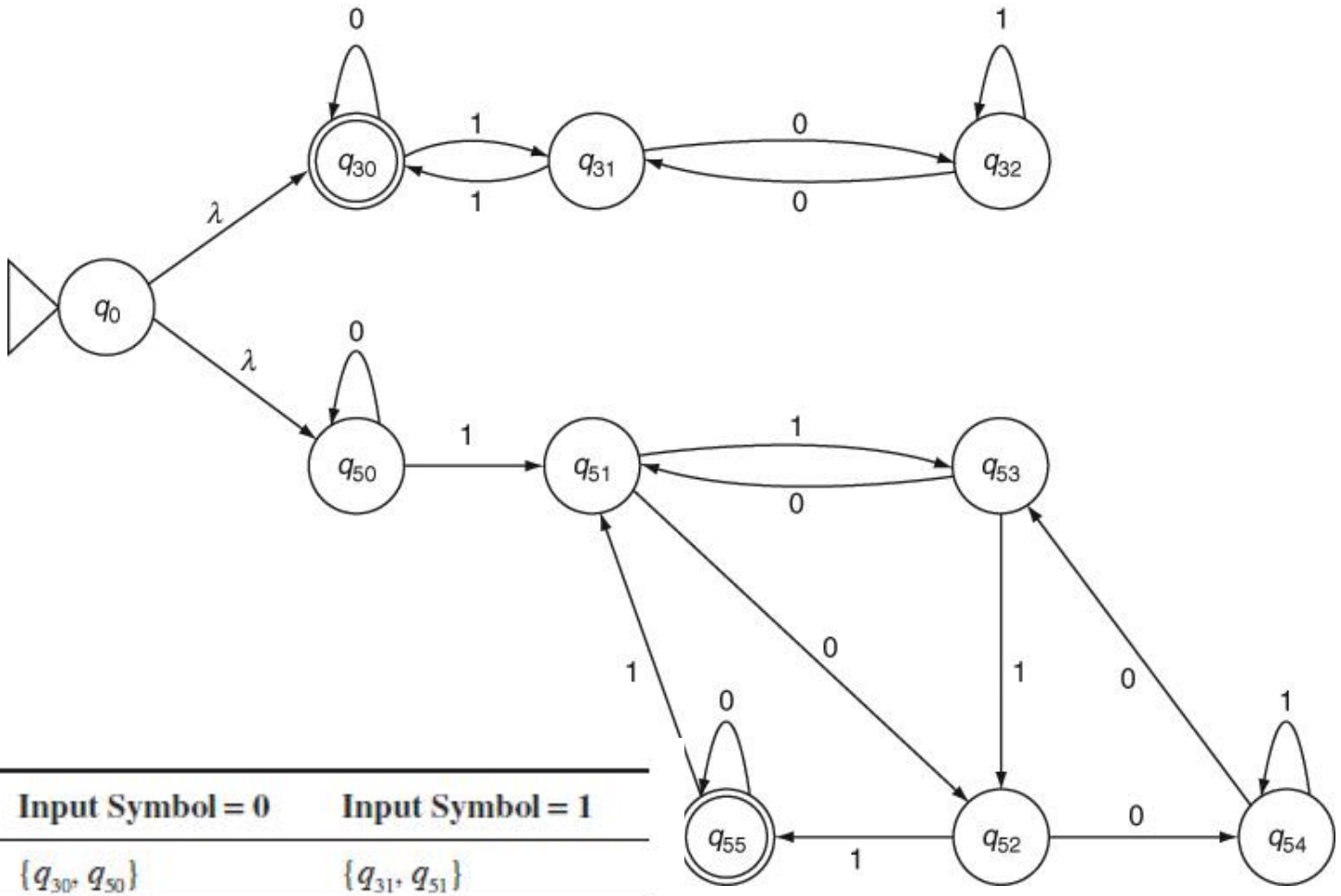
- q_0 is the start state.
- q_1 is the state where all symbols are consumed before looking for an ending with 00.
- q_2 is where one 0 has been seen, that is, the number so far is an even number.
- q_3 is the final state where the input is divisible by 4.
- q_4 is the state where mod 3 of the number so far is 1.
- q_5 is the state where mod 3 of the number so far is 2.
- q_6 is the state where the number so far is divisible by 3.
- q_7 is the final state where the number is divisible by 6.
- q_8 is the special final state for accepting the number 0.

Meanings of DFA States: Example 3.7

- Q_1 is the special final state for accepting the number 0.
- Q_5 is the state where the number is divisible by 4 but not by 6 (mod 6 is 4).
- Q_7 is the state where the number is not divisible by 4 but is divisible by 6.
- Q_8 is the state where the number is divisible by 4 but not by 6 (mod 6 is 2).
- Q_{10} is the state where the number is divisible by both 4 and 6.
- The number so far is odd in states Q_2, Q_4 and Q_6
- The number so far is even but not divisible by either 4 or 6 in Q_3 and Q_9

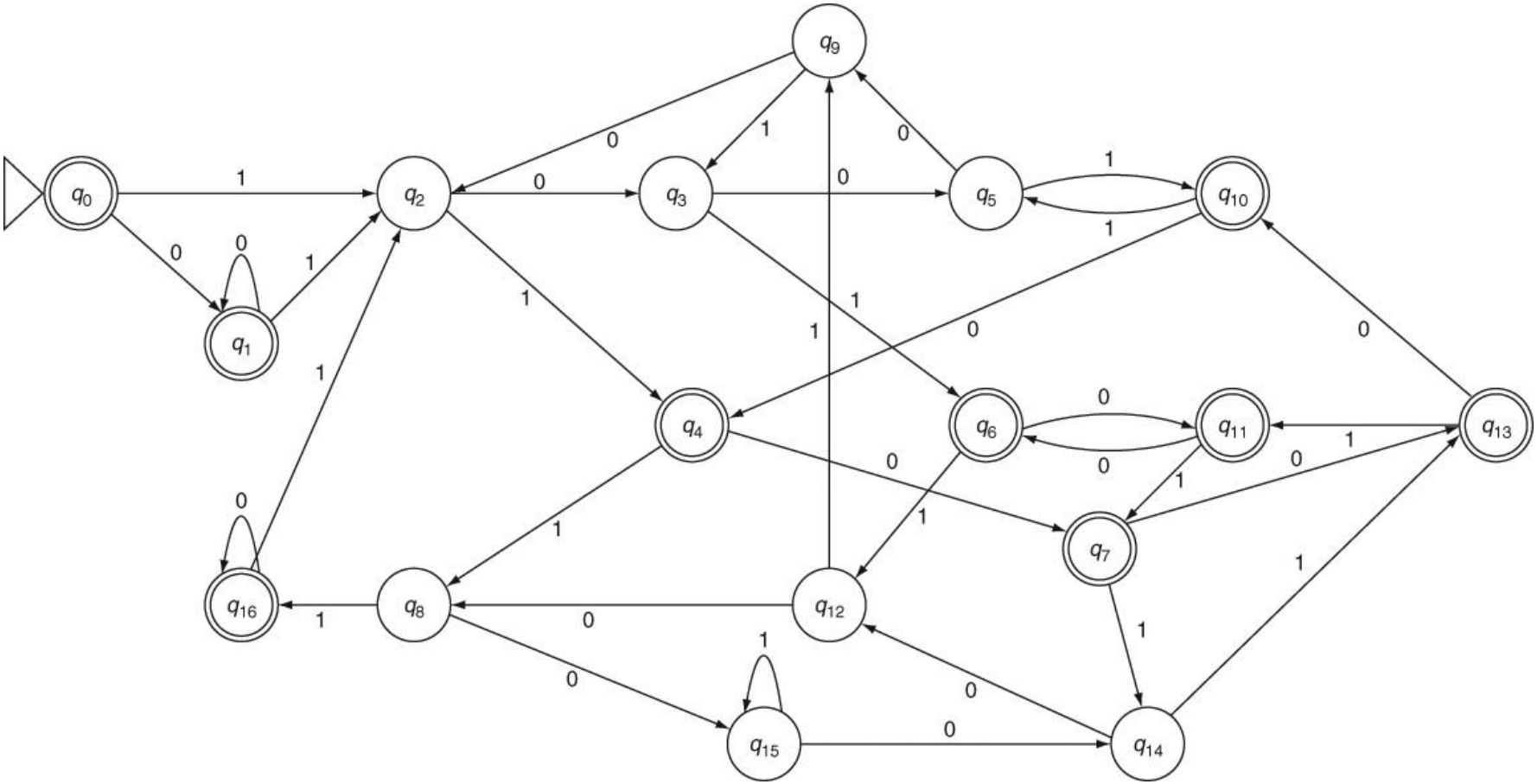
Jumping States without Input: λ-Transitions

- Example 3.8: Numbers divisible by either 3 or 5
- NFA:



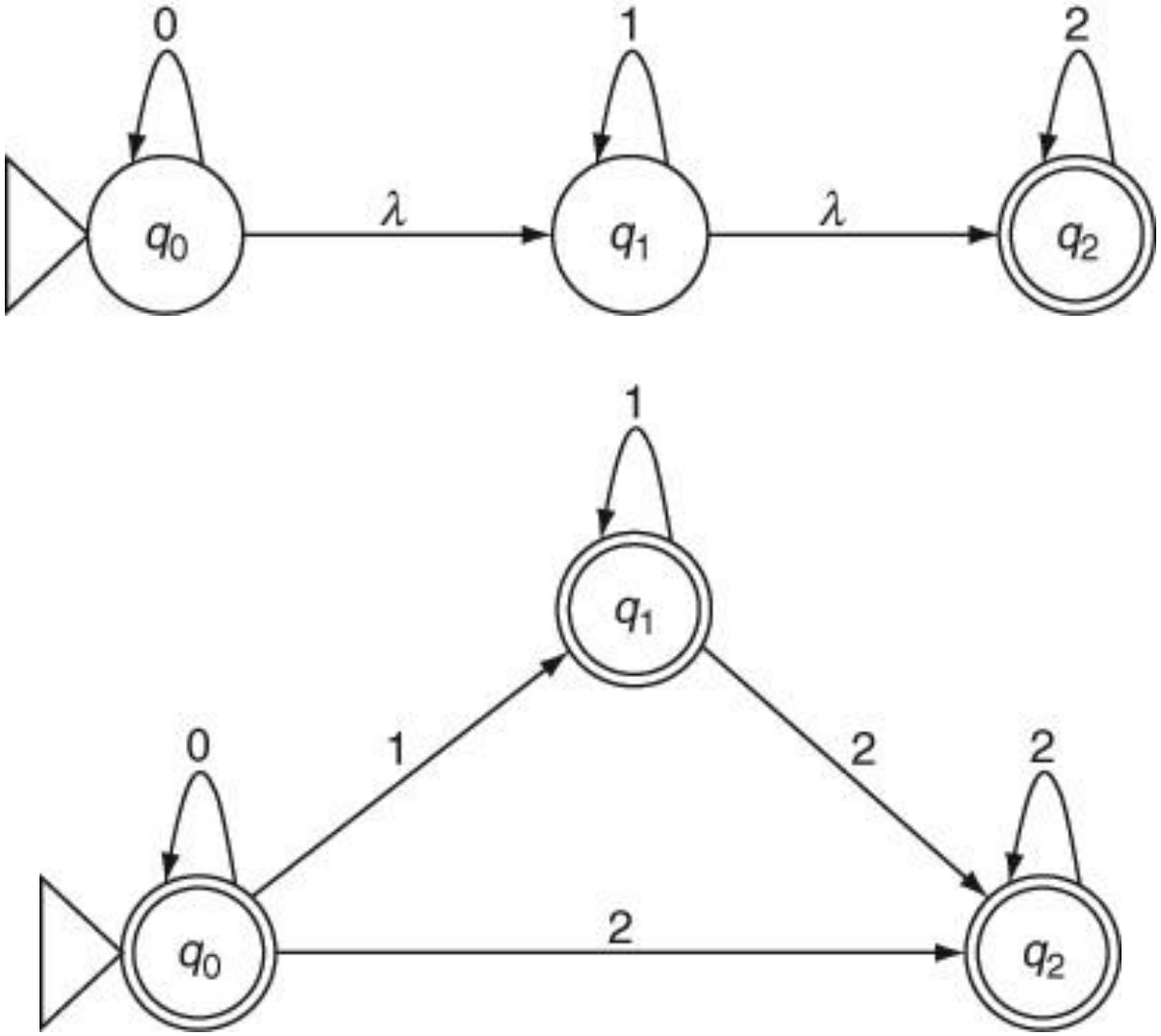
DFA State	Input Symbol = 0	Input Symbol = 1
$Q_0 = \{q_0\}$	$\{q_{30}, q_{50}\}$	$\{q_{31}, q_{51}\}$

RESULTING DFA: EXAMPLE 3.8



NFA TO DFA: EXAMPLE 3.9

- 0 s, 1 s and 2 s in that order



Mark & Reduce: Method for Minimizing DFA

Algorithm

First deleted any state that is unreachable from the start state;

- For each pair of states where one is a final state and the other is non-final,
  mark them as distinguishable;
- For each pair of states q_i and $q_{j,}$
  For each symbol in the alphabet,
    If qi takes the automaton to q_m and q_j to q_n and
    If q_m and q_n are already marked as distinguishable,
    Then mark q_i and q_j as distinguishable;
- Repeat the above until no more pairs can be marked;
- All the pairs of states that are not marked are indistinguishable;
- Collapse indistinguishable pairs to single states and merge their transitions.

MINIMIZING DFA: EXAMPLE 3.10

- NFA and DFA for multiple keyword search: 010 or 000

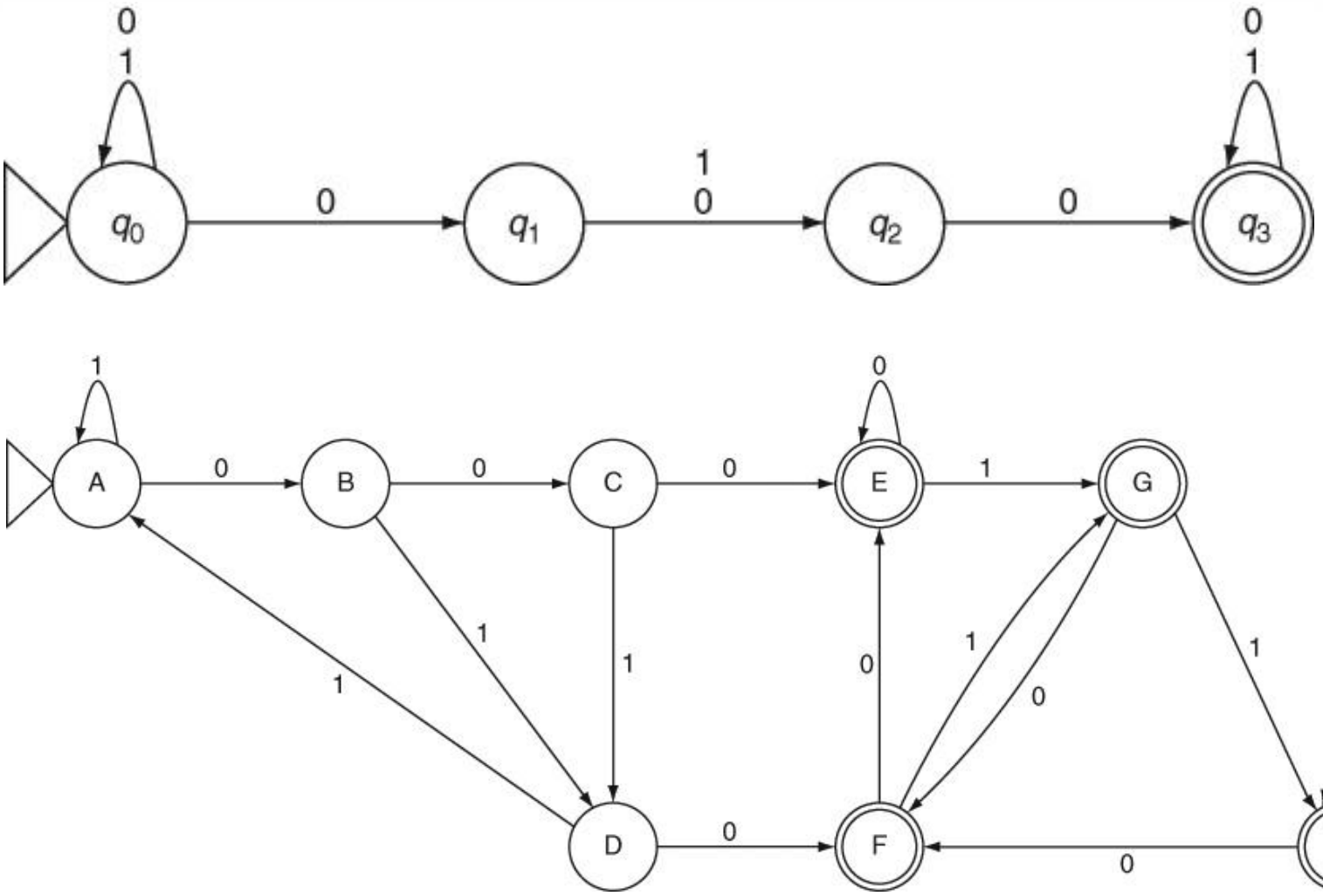
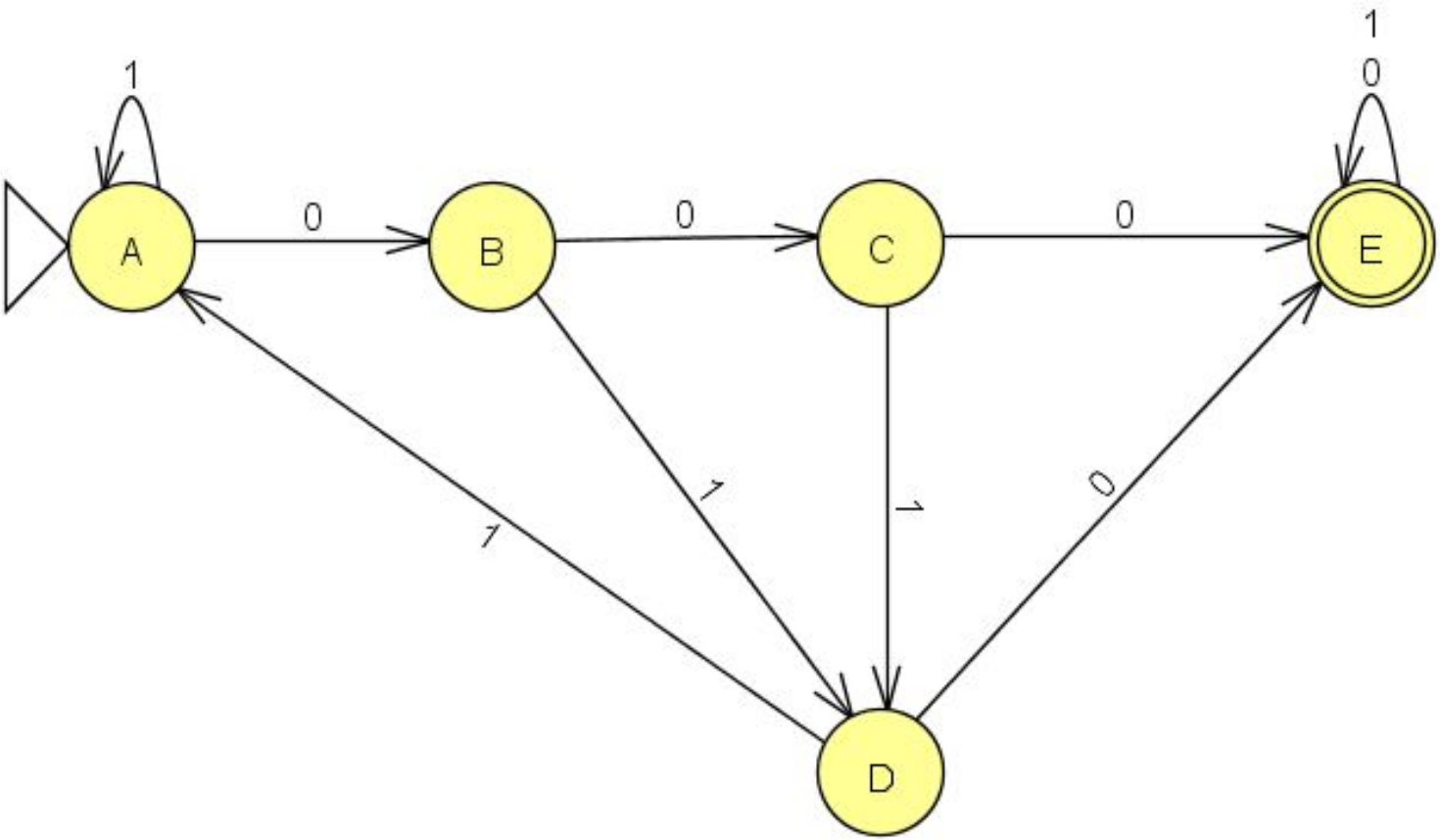


TABLE FILLING: EXAMPLE 3.10

TABLE 3.3 Marking Distinguishable States for Minimizing a DFA

B	on 0 (B×C)					
C	on 0	on 0				
D	on 0	on 0	on 1 (D×A)			
E	fnf	fnf	fnf	fnf		
F	fnf	fnf	fnf	fnf	?	
G	fnf	fnf	fnf	fnf	?	?
H	fnf	fnf	fnf	fnf	?	?	?
States	**A**	**B**	**C**	**D**	**E**	**F**	**G**

Minimized DFA: Example 3.10



EXAMPLE 3.11

- DFA for minimization

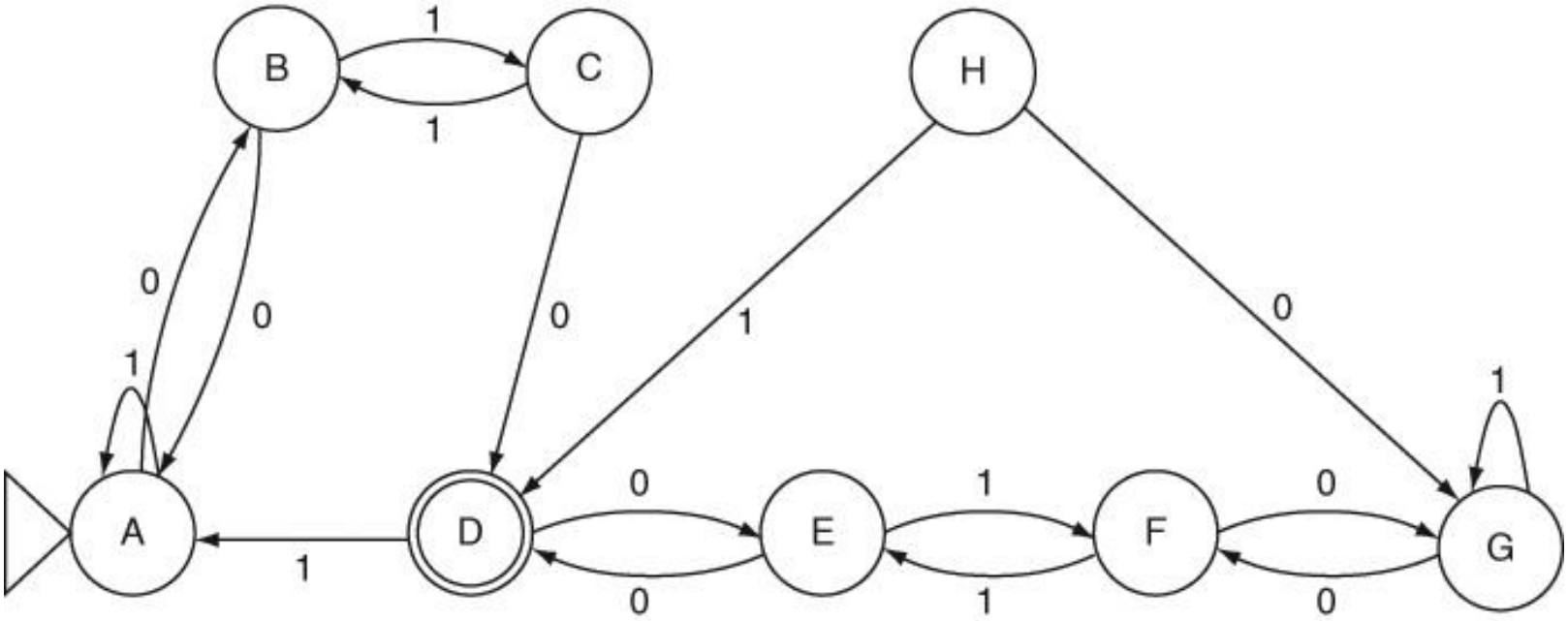
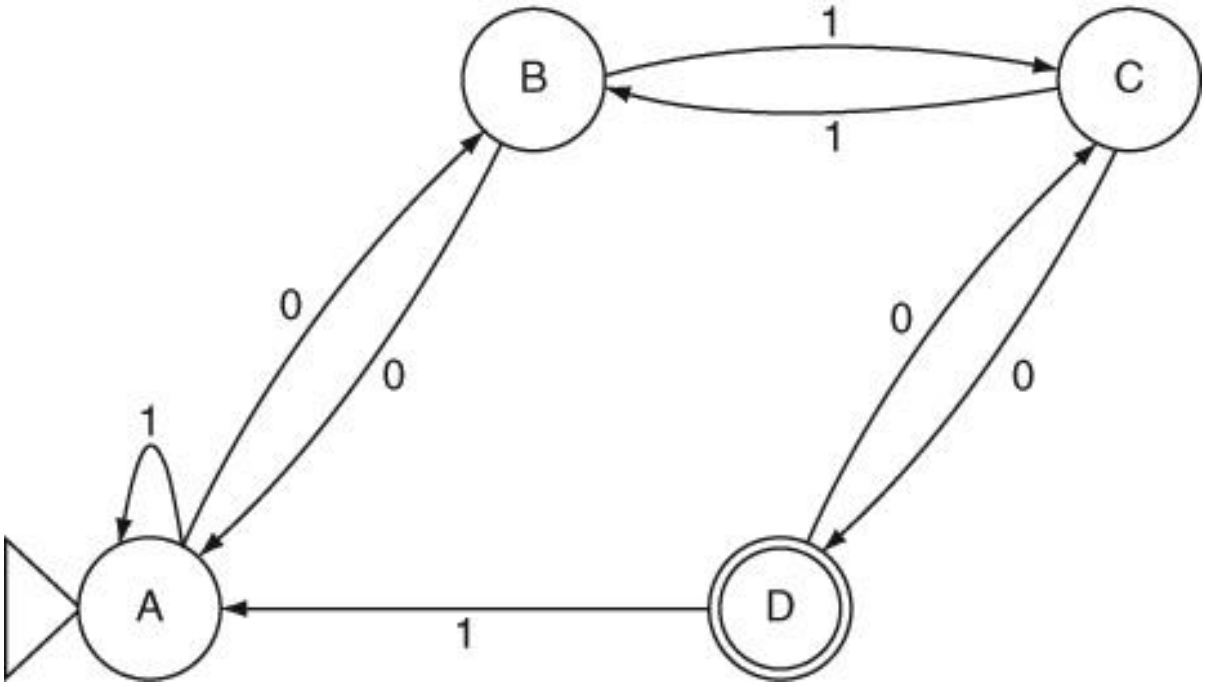


TABLE FILLING: EXAMPLE 3.11

TABLE 3.4 Marking Distinguishable States for Minimizing a DFA

B	on 1 (A×C)					
C	on 0	on 0				
D	fnf	fnf	fnf			
E	on 0	on 0	?	fnf		
F	on 1 (A×E)	?	on 0	fnf	on 0	
G	?	on 1 (C×G)	on 0	fnf	on 0	on 1 (E×G)
H						
States	**A**	**B**	**C**	**D**	**E**	**F**	**G**

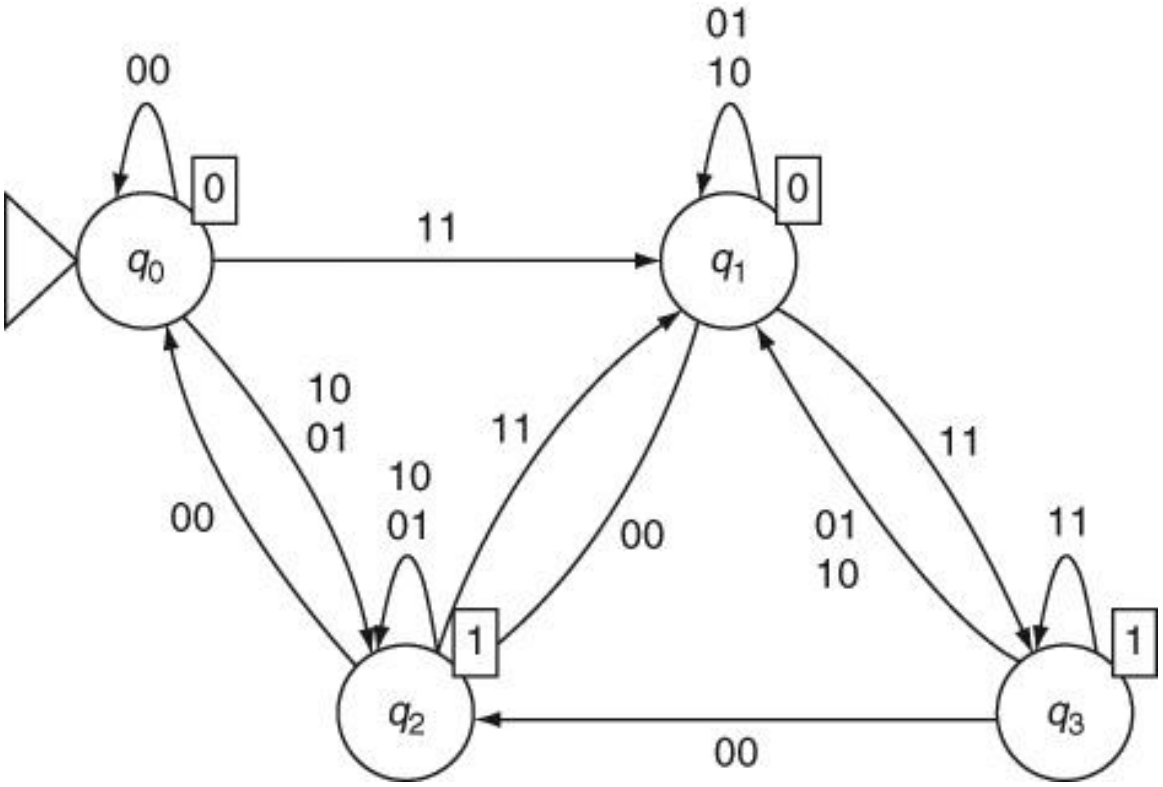
MINIMIZED DFA: EXAMPLE 3.11



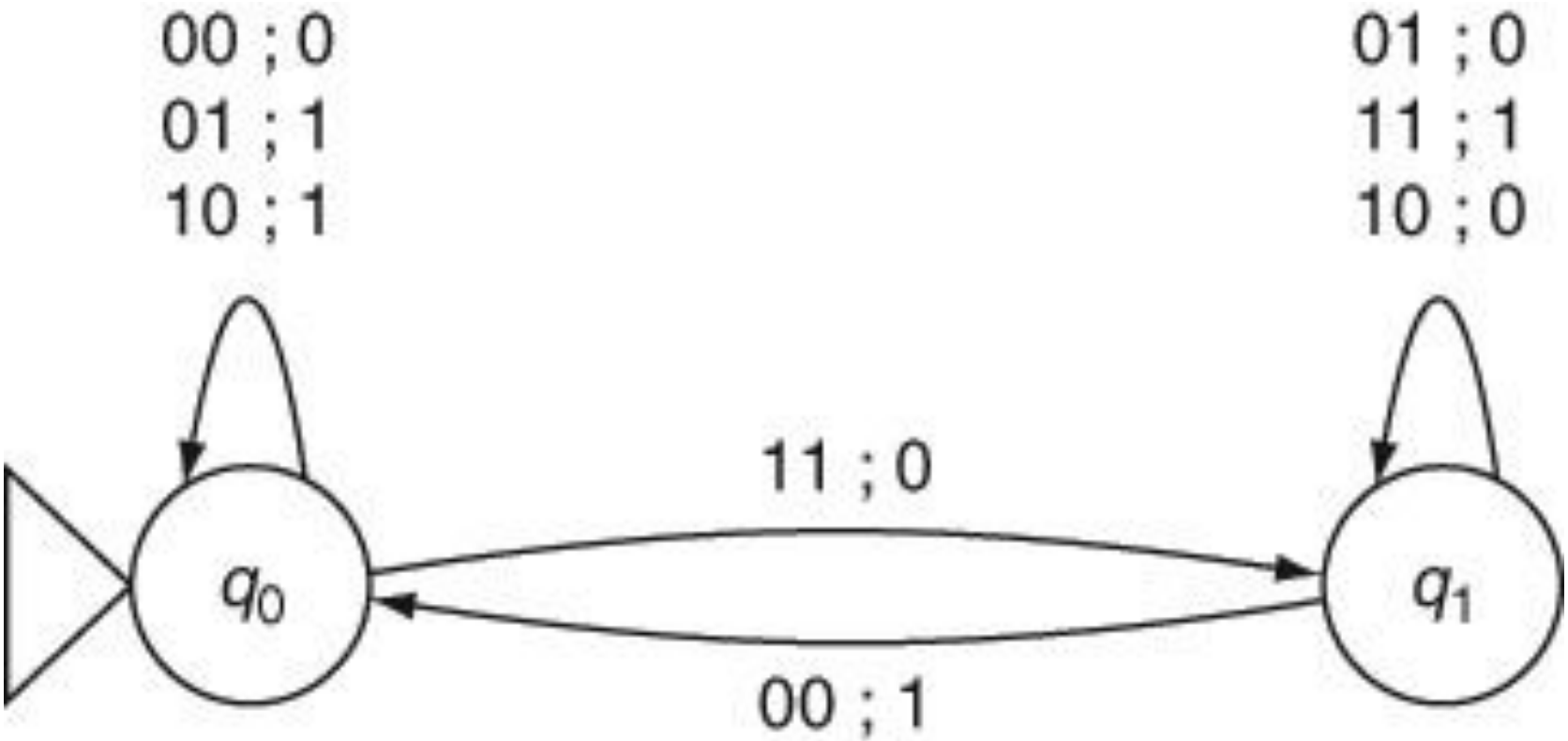
Finite State Transducers

- Produce output strings
- Output symbol defined per state: Moore machine
- Output symbol defined per transition: Mealy machine

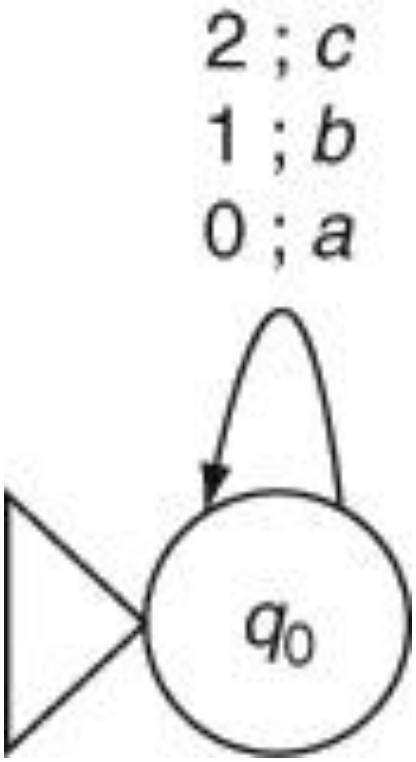
Moore Machine Binary Adder: Example 3.12



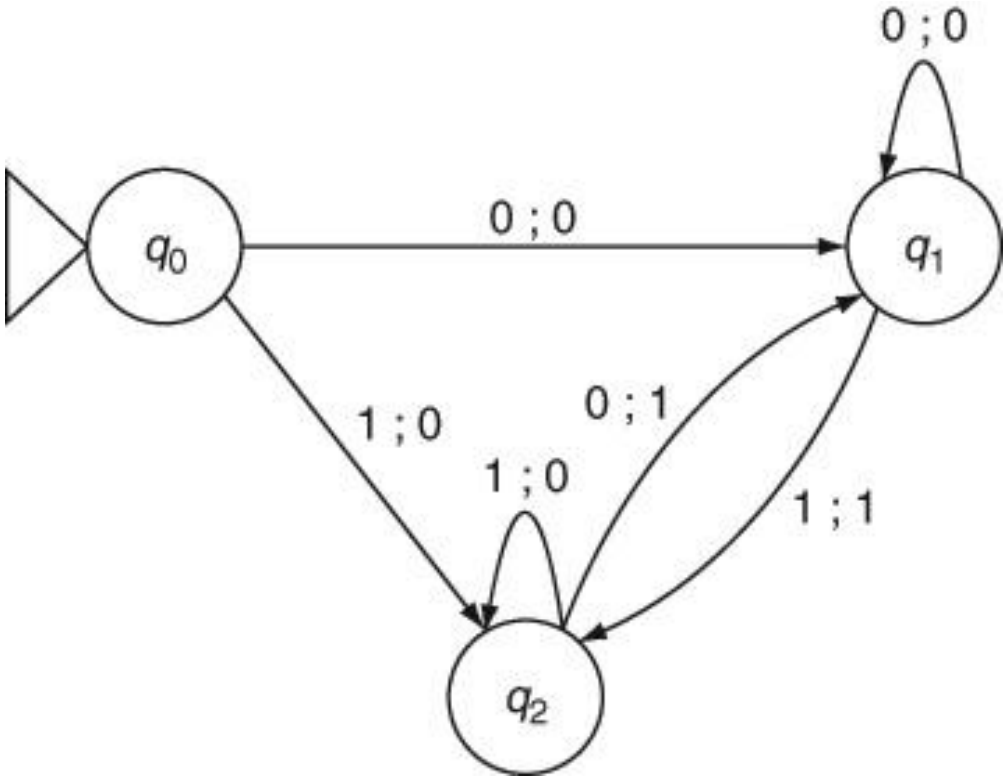
Mealy Machine Binary Adder: Example 3.13



Mealy Machine Translator: Example 3.14



Mealy Machine Edge Detector: Example 3.15



THEOREMS

- **Theorem 1.** *Equivalence of NFA and* DFA: For every nondeterministic finite automaton M_n, there is an equivalent deterministic finite automaton M_d whose language is the same as that of the nondeterministic automaton, that is, M_n.language = M_d.language.

- **Theorem 2:** *DFA* minimization: The minimal DFA obtained by marking distinguishable states and collapsing indistinguishable states is equivalent to the original DFA (i.e., it accepts the same language).

KEY IDEAS

- A non-deterministic finite automaton can have more than one transition from the same state for the same input symbol.
- It can guess the right choice for accepting an input string. Equivalently, it can try out all the choices in parallel to see if any one leads to acceptance of the input string.
- Non-deterministic automata are often easier for us to design.
- Non-deterministic automata can also change states without consuming any input symbol. NFA with such λ-transitions make it further easy for us to design them for a given problem. They also help us combine two or more automata to solve a more complex problem.
- Non-deterministic finite automata are strictly equivalent to deterministic ones. They have the same computing power as deterministic ones, in terms of the range of problems they can solve (or the range of input strings they can handle).

KEY IDEAS (CONTD..)

- The method of subset construction is used to convert an NFA to an equivalent DFA.
- A DFA obtained from an equivalent NFA usually has more states than the NFA. In the worst case, the number of states in the resulting DFA can be exponential in the number of NFA states.
- A DFA can be minimized by identifying and merging indistinguishable states. Indistinguishable states denote the same equivalence class of (sub)strings and can be merged into a single state. The minimal number of states in an equivalent DFA is the total number of equivalence classes of strings. This number must be finite for a regular language according to the Myhill–Nerode theorem (Theorem 10 in Appendix B).
- Book-keeping in minimizing a DFA is made easy by applying a table-filling technique to mark all distinguishable pairs of states.
- Unlike finite automata which merely accept or reject an input string, finite-state transducers such as Moore and Mealy Machines produce an output while processing the input string.

End of Chapter 3