Pharos University in Alexandria
Faculty of Computer Science & Artificial Intelligence
Course Title: Theory of Computation
Code: CS 307

Theory of Computation

Lecturer: Sherine Shawky

Text Books

1. Introduction to formal languages and automata, Peter Linz, 6th edition, 2017.

Week 3

Nondeterministic Finite Accepters (NFA)

+ Equivalence of DFA and NFA

NFA

- Nondeterminism means a choice of moves for an automaton. Rather than prescribing a unique move in each situation, we allow a set of possible moves. Formally, we achieve this by defining the transition function so that its range is a set of possible states.
- A nondeterministic finite accepter or nfa is defined by the quintuple

$M = (Q, \Sigma, \delta, q0, F)$,

where Q, Σ, $q0$, F are defined as for deterministic finite accepters, but

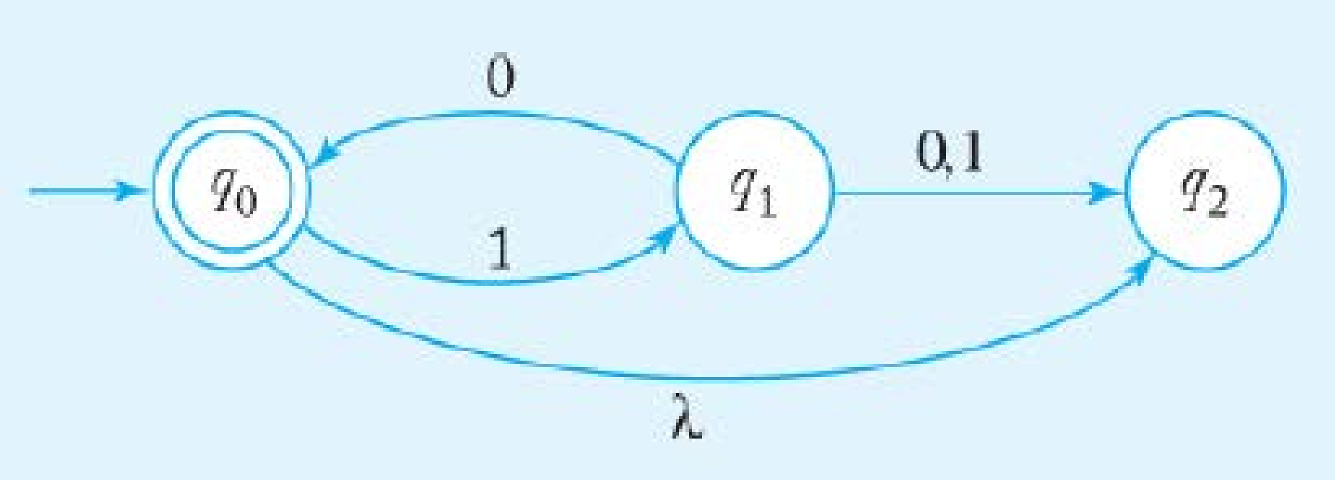
$$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q.$$

NFA

- Note that there are three major differences between this definition and the definition of a dfa. In a nondeterministic accepter, the range of δ is different, so that its value is not a single element of *Q*, but a subset of it.
- This subset defines the set of all possible states that can be reached by the transition. If, for instance, the current state is $q1$, the symbol a is read, and δ (q1, a) = {q0, q2}, then either $q0$ or $q2$ could be the next state of the nfa. Also, we allow λ as the second argument of δ.
- This means that the nfa can make a transition without consuming an input symbol.
- Although we still assume that the input mechanism can only travel to the right, it is possible that it is stationary on some moves. Finally, in an nfa, the set δ (qi, a) may be empty, meaning that there is no transition defined for this specific situation.

NFA Example

- A nondeterministic automaton is shown in Figure 2.9. It is nondeterministic not only because several edges with the same label originate from one vertex, but also because it has a λ-transition. Some transitions, such as $\delta(q2, 0)$, are unspecified in the graph. This is to be interpreted as a transition to the empty set, that is, $\delta(q2, 0) = \emptyset$. The automaton accepts strings λ, 1010, and 101010, but not 110 and 10100.
- Note that for 10 there are two alternative walks, one leading to $q0$, the other to $q2$. Even though $q2$ is not a final state, the string is accepted because one walk leads to a final state.

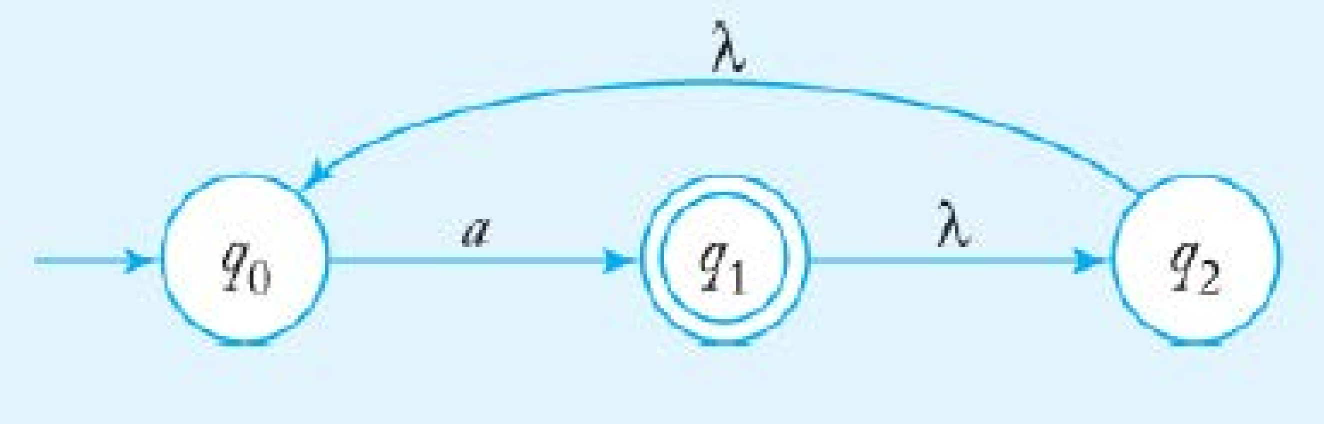


NFA Definition 2

- 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 $w \in \Sigma^*$.

NFA Example

- It has several λ-transitions and some undefined transitions such as $\delta(q2, a)$. Suppose we want to find $\delta^*(q1, a)$ and $\delta^*(q2, \lambda)$. There is a walk labeled a involving two λ-transitions from $q1$ to itself. By using some of the λ-edges twice, we see that there are also walks involving λ- transitions to $q0$ and $q2$. Thus, $\delta^*(q1, a) = \{q0, q1, q2\}$.
- Since there is a λ-edge between $q2$ and $q0$, we have immediately that $\delta^*(q2, \lambda)$ contains $q0$. Also, since any state can be reached from itself by making no move, and consequently using no input symbol, $\delta^*(q2, \lambda)$ also contains $q2$. Therefore, $\delta^*(q2, \lambda) = \{q0, q2\}$.
- Using as many λ-transitions as needed, you can also check that $\delta^*(q2, aa) = \{q0, q1, q2\}$. The definition of δ^* through labeled walks is somewhat informal, so it is useful to look at it a little more closely.

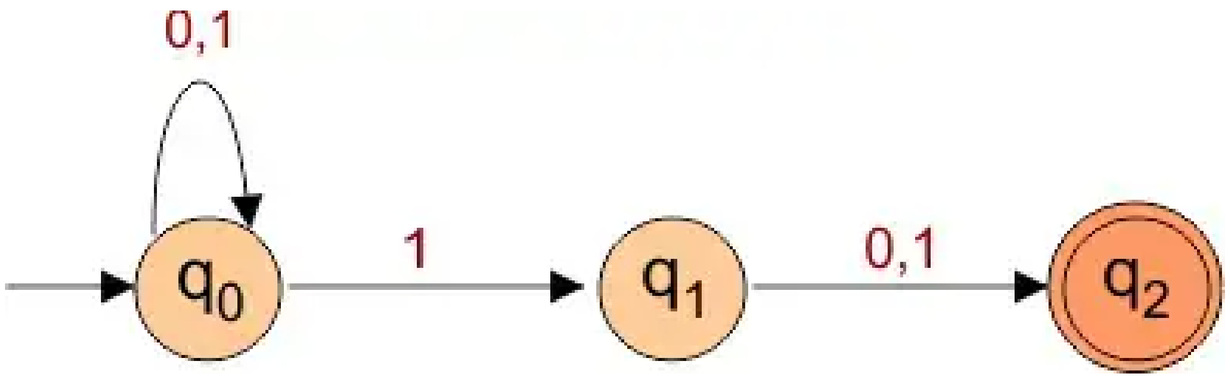


NFA Definition 3

- The language L accepted by an nfa $M = (Q, \Sigma, \delta, q0, F)$ is defined as the set of all strings accepted in the above sense.
- Formally, $L(M) = \{w \in \Sigma^* : \delta^*(q0, w) \cap F \neq \emptyset\}$.
- In words, 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.

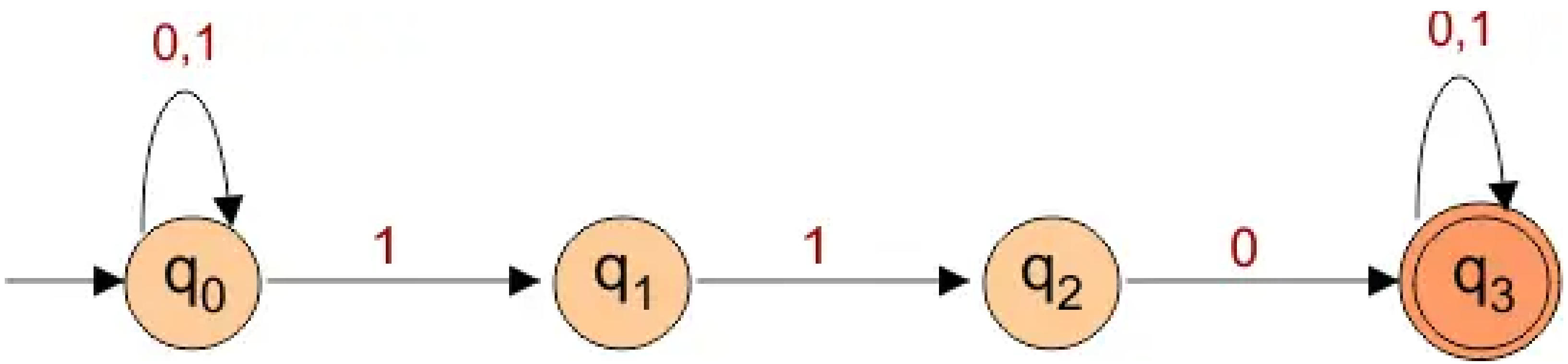
NFA Problem

- Design NFA over inputs {0,1} where second last bit is 1.



NFA Problem

- Construct an NFA with $\sum = \{0, 1\}$ in which each string must contain “double ‘1’ is followed by single ‘0’.

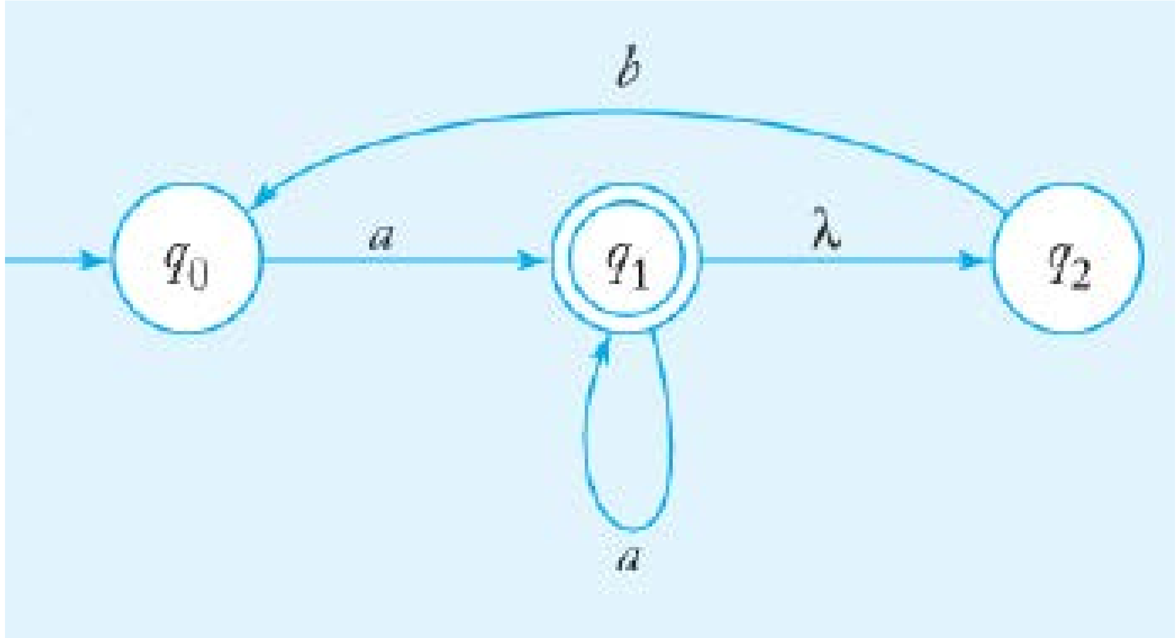


Equivalence of DFA and NFA

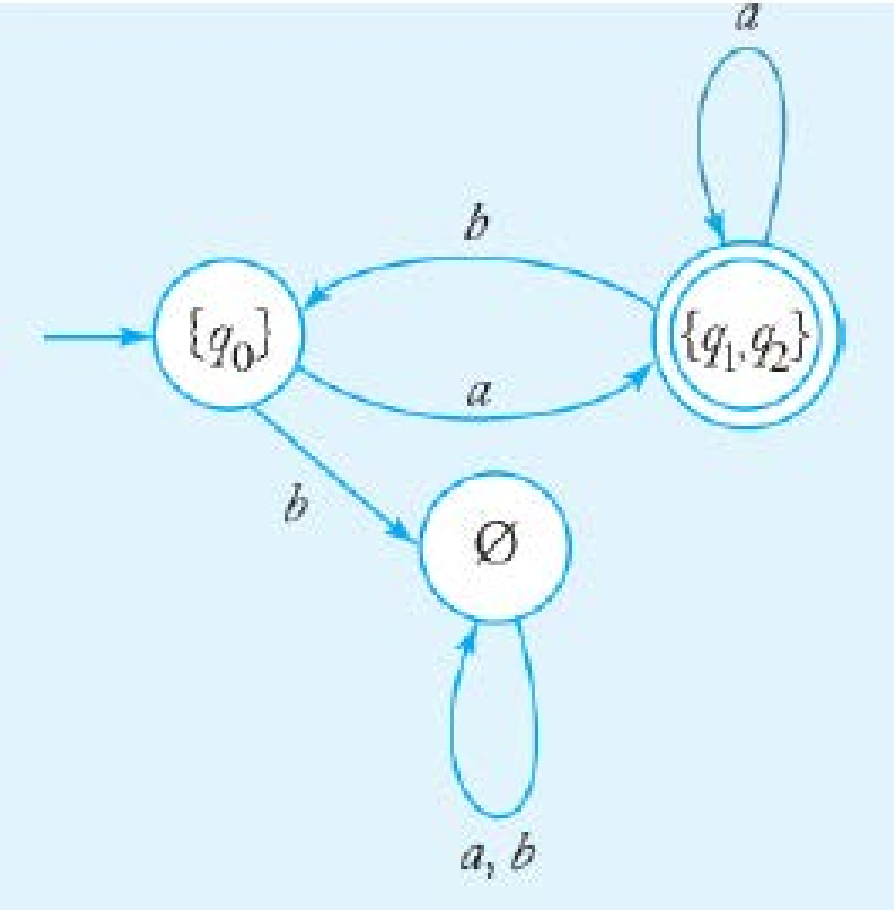
- Two finite accepters, $M1$ and $M2$, are said to be equivalent if $L(M1) = L(M2)$, that is, if they both accept the same language.

Problem

- Convert the nfa to an equivalent dfa.

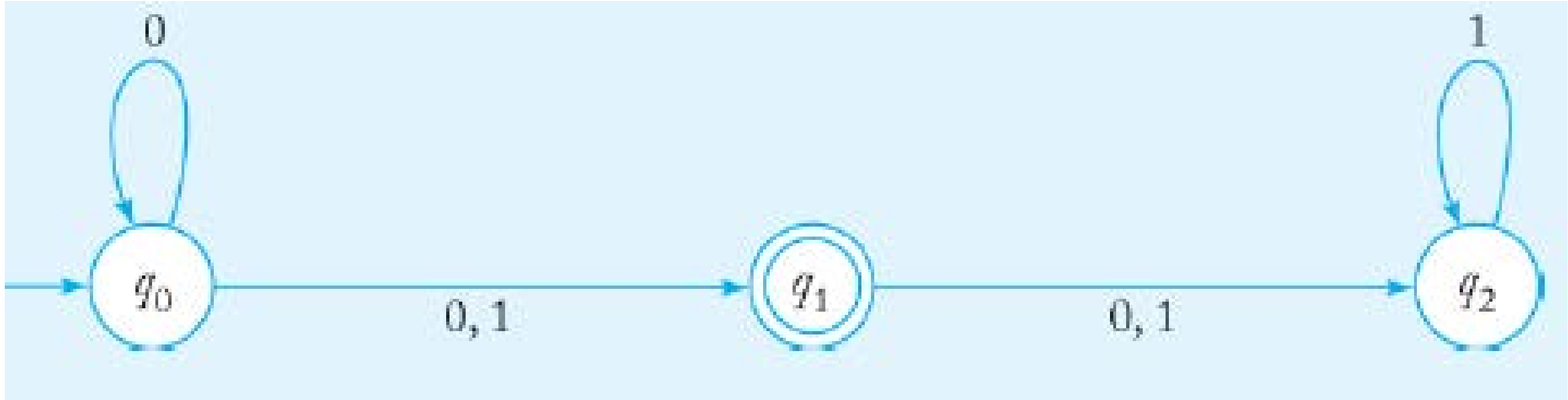


Problem’s Answer

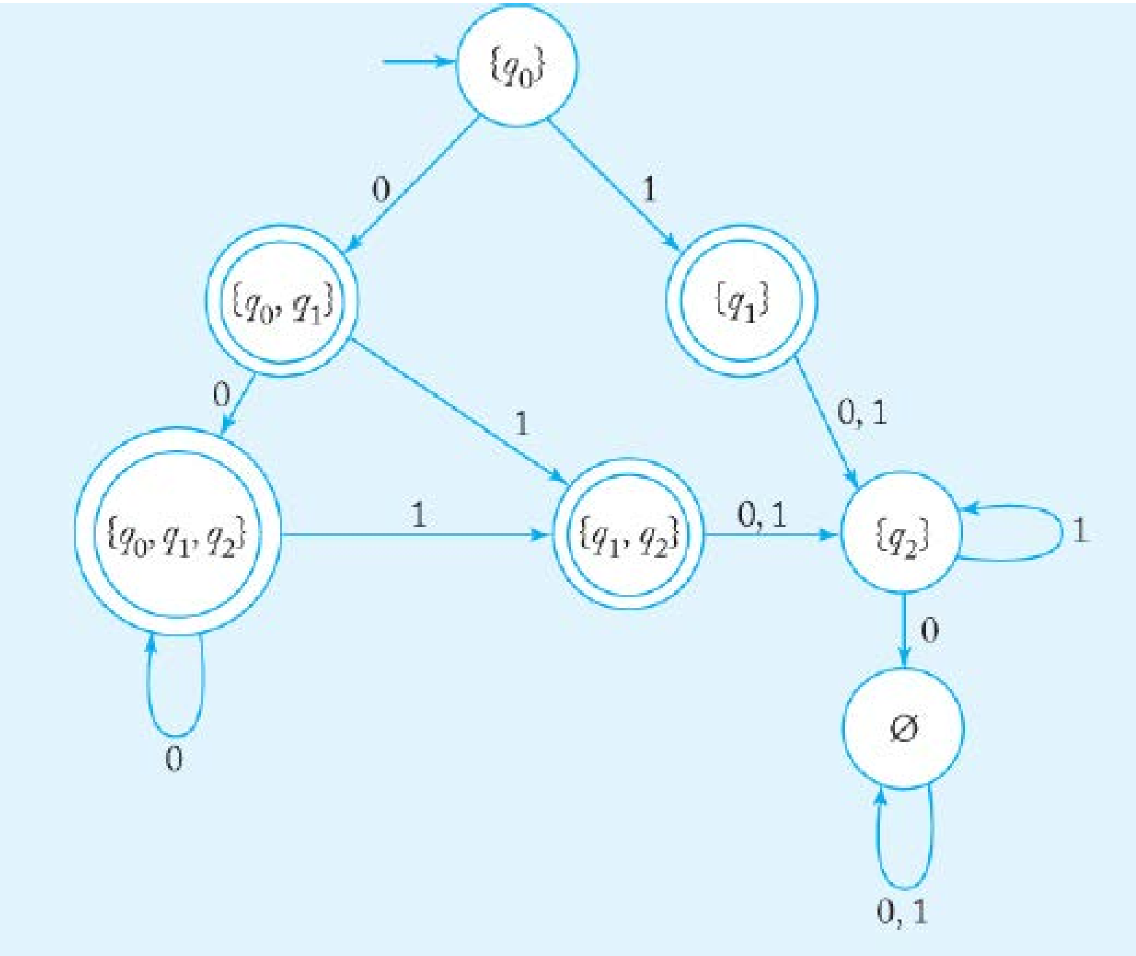


Problem

- Convert the nfa to an equivalent dfa.



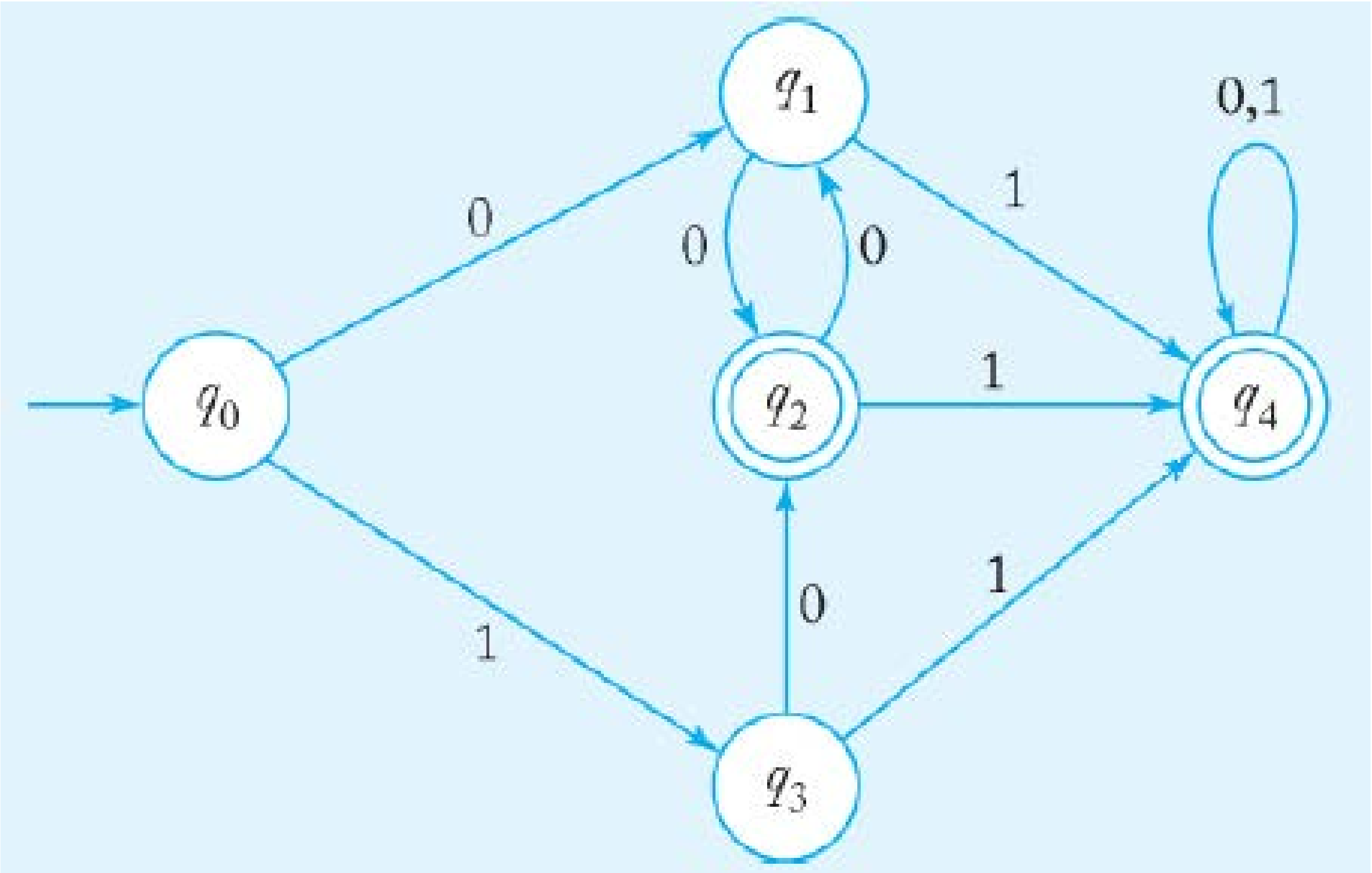
Problem’s Answer



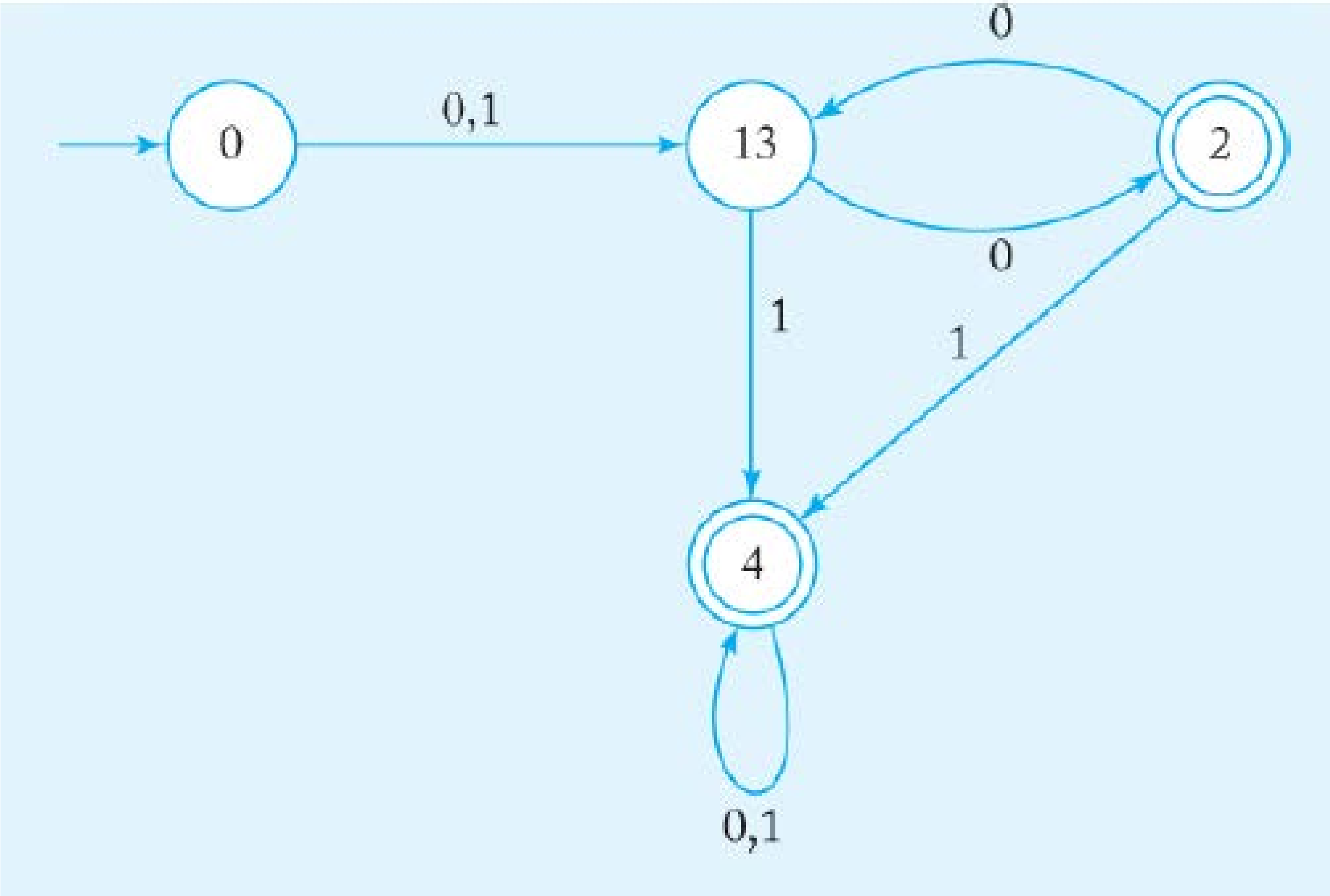
Minimize DFA

- Two states p and q of a dfa are called indistinguishable if $\delta^*(p, w) \in F$ implies $\delta^*(q, w) \in F$,
- And $\delta^*(p, w) \notin F$ implies $\delta^*(q, w) \notin F$, for all $w \in \Sigma^*$. If, on the other hand, there exists some string $w \in \Sigma^*$ such that $\delta^*(p, w) \in F$ and $\delta^*(q, w) \notin F$, or vice versa, then the states p and q are said to be distinguishable by a string w.
- Clearly, two states are either indistinguishable or distinguishable. Indistinguishability has the properties of an equivalence relation: If p and q are indistinguishable and if q and r are also indistinguishable, then so are p and r, and all three states are indistinguishable.
- One method for reducing the states of a dfa is based on finding and combining indistinguishable states. We first describe a method for finding pairs of distinguishable states.

Minimize DFA Problem



Minimize DFA Answer



Quiz

- Design DFA machine that accepts only strings that end with “11” over inputs {0,1}.