DFA and NFA

Assylbek Issakhov,
Ph.D., professor of School of Math and Cybernetics

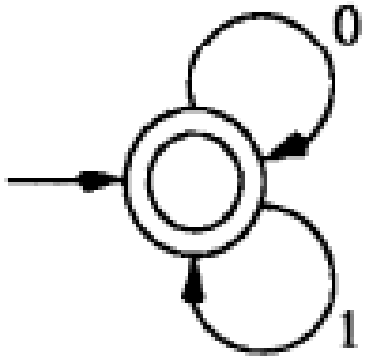
Examples of Deterministic Finite Automata

- Here we will develop some techniques to construct DFA's through examples. In each example, a language is given as $\{0,1\}$ and we show how to construct a DFA accepting the language.
- **Example 2.6.** $(0+1)^*$.
- **Solution**. The language $(0+1)^*$ contains all binary strings. So, we simply let the initial state be the final success state.

Examples of Deterministic Finite Automata

- The transition diagram of this DFA is actually the same as the labeled digraph representation for the regular expression $(0 + 1)^*$. We show it in Figure 2.7(a). (In a transition diagram, if two edges have the same starting and ending vertices, we may merge them into one edge and put both labels on the new edge. For instance, the transition diagram of Figure 2.7(a) can be simplified to that of Figure 2.7(b).)

Examples of Deterministic Finite Automata



(a)

(b)

Figure 2.7: DFA for $(0 + 1)^*$.

Examples of Deterministic Finite Automata

- **Example 2.7.** The set of all binary strings beginning with prefix 01.
- **Solution**. It is easy to find a regular expression $01(0+1)^*$ for this language. From the regular expression, we can immediately find its labeled digraph representation as shown in Figure 2.8(a). However, it is not the transition diagram of a DFA.

Examples of Deterministic Finite Automata

- Recall that in the transition diagram of a DFA $M = (Q, \Sigma, \delta, q_0, F)$, there is exactly one out-edge from state q with label a for each pair of $(q, a) \in Q \times \Sigma$. To satisfy this condition, we add a failure state q_3 and edges

$$q_0 \rightarrow^1 q_3 \text{ and } q_1 \rightarrow^0 q_3.$$

- The complete transition diagram is shown in Figure 2.8(b).

Examples of Deterministic Finite Automata

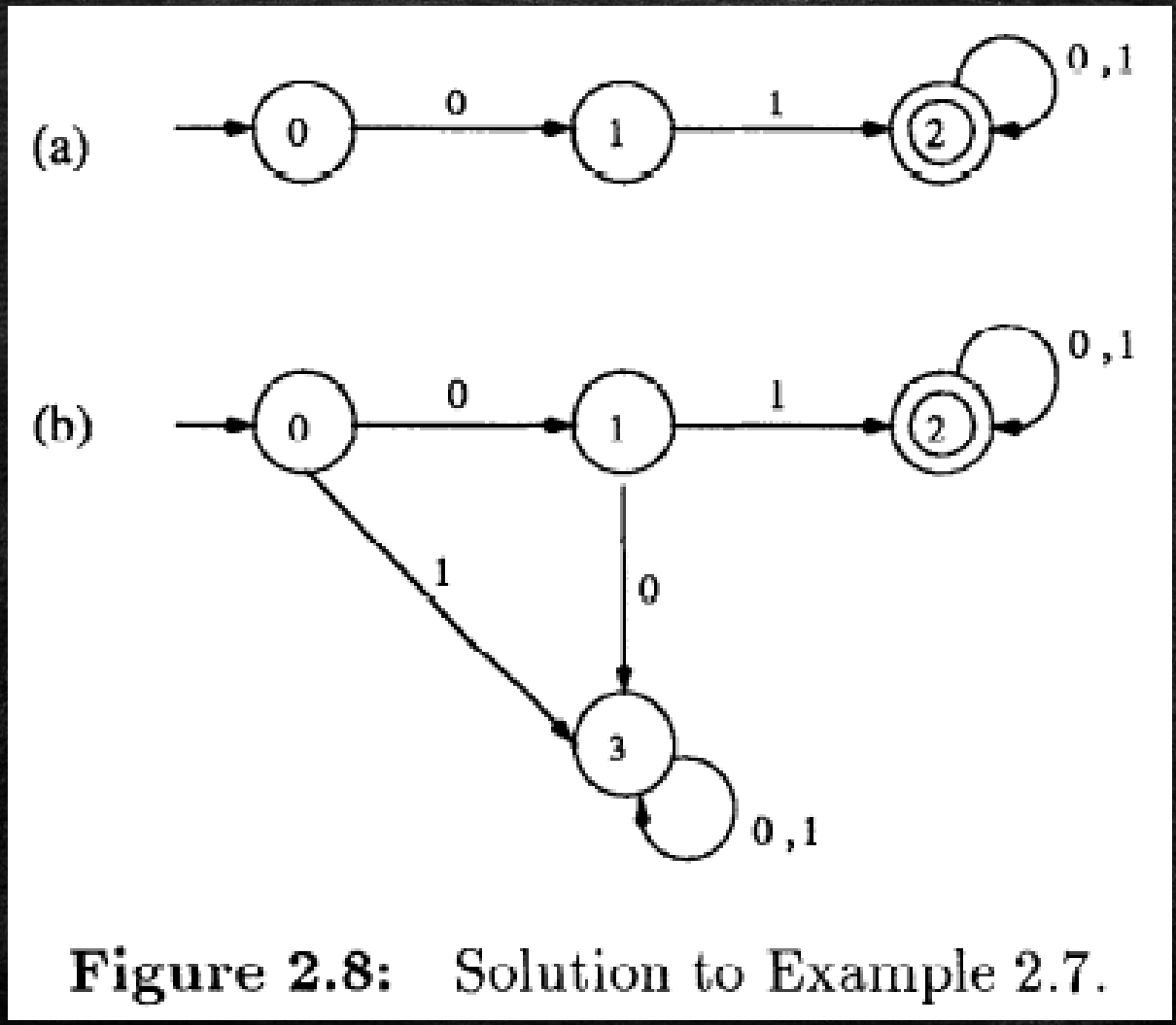


Figure 2.8: Solution to Example 2.7.

Examples of Deterministic Finite Automata

- **Example 2.8.** The set of all binary strings having a substring 00.
- **Solution**. First, let us note that the regular expression $(0+1)^*00(0+1)^*$ for this set does not indicate where the first pair 00 occurs in the string. On the other hand, the DFA must recognize the substring 00 at its first occurrence. Thus, the above regular expression is not helpful to this problem. Instead, we need to analyze the problem more carefully.

Examples of Deterministic Finite Automata

- Suppose that the string $x_1 x_2 \ldots x_n$ is stored on the tape, where each x_i denotes one bit in $\{0,1\}$. Then, we may check each of the substrings

$$x_1 x_2, x_2 x_3, \ldots, x_{n-1} x_n$$

- in turn, to see whether it is equal to 00, and accept the input string as soon as a substring 00 is found. This suggests the following way to construct the required DFA.

Examples of Deterministic Finite Automata

- *Step 1*. Build a *checker* as shown in Figure 2.9(a), with state q_2 being a success state, meaning that if two consecutive 0's are found then the input string is to be accepted. In particular, if $x_1x_2 = 00$, then the string x is accepted.
- *Step 2*. If $x_1 = 1$, then we give up on substring x_1x_2 and continue to check x_2x_3. So, we need to go back to state q_0; that is, $\delta(q_0, 1) = q_0$. This action is shown in Figure 2.9(b).

Examples of Deterministic Finite Automata

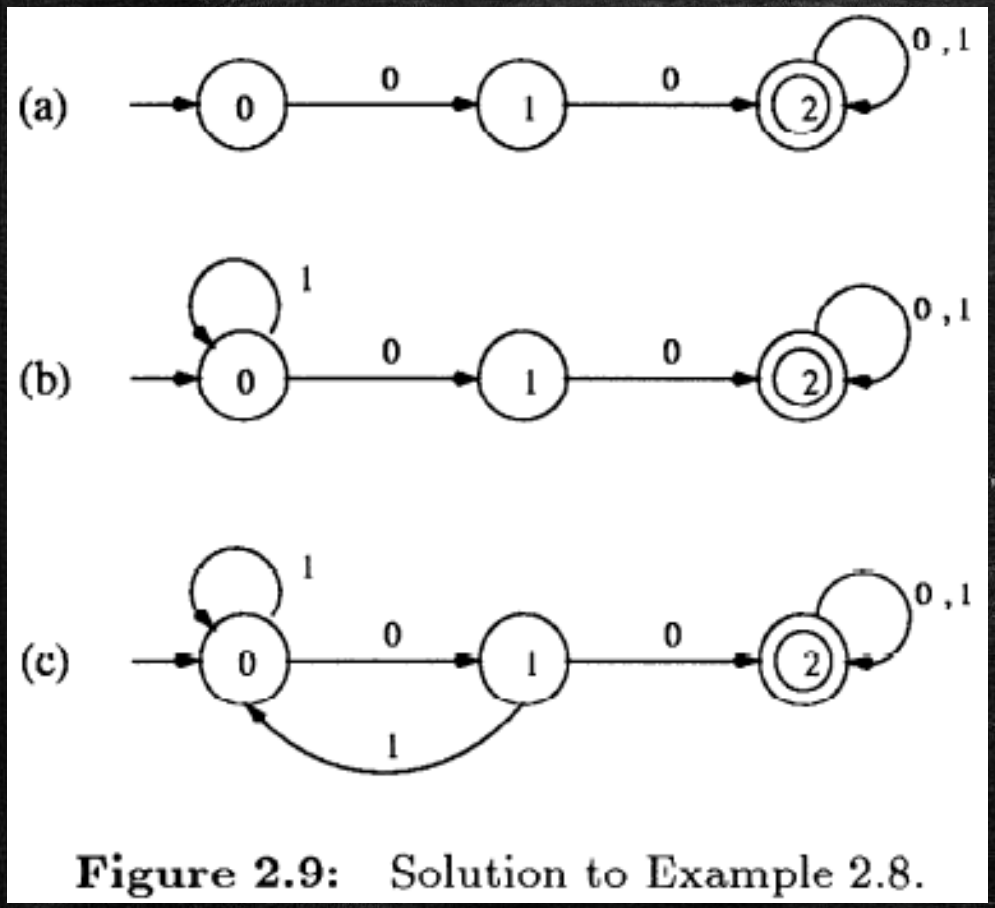


Figure 2.9: Solution to Example 2.8.

- *Step 3*. If $x_1 = 0$ and $x_2 = 1$, then neither x_1x_2 nor x_2x_3 is 00. So, we also need to go back to restart at q_0 to check x_3x_4. That is, we let $\delta(q_1, 1) = q_0$. Figure 2.9(c) shows the complete DFA.

Examples of Deterministic Finite Automata

- In general, suppose that $x_1 x_2 \ldots x_n$ is the string on the tape and we want to check

$$x_1 \ldots x_k, x_2 \ldots x_{k+1}, \ldots, x_{n-k+1} \ldots x_n$$

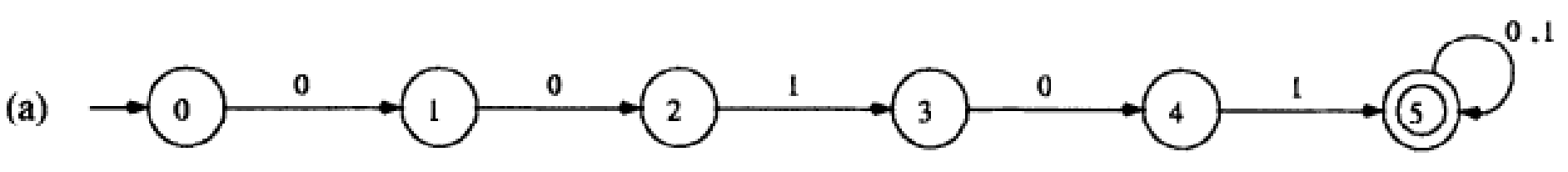
- in turn to match a substring $a_1 a_2 \ldots a_k$. Then, we set up $k+1$ states $q_0, q_1, \ldots, q_k$, with q_i standing for "found $a_1 a_2 \ldots a_i$". At state q_i, if $b = a_{i+1}$, then we define $\delta(q_i, b) = q_{i+1}$. If $b \neq a_{i+1}$, then we define $\delta(q_i, b) = q_j$, where j is the maximum index j such that

$$a_{i-j+2} a_{i-j+3} \ldots a_i b = a_1 a_2 \ldots a_j.$$

Examples of Deterministic Finite Automata

- That is, we find the longest suffix y of $a_1 a_2 \ldots a_i b$ which is a prefix of $a_1 \ldots a_k$ and go to $q_{|y|}$. The following example explains this idea more clearly.
- **Example 2.9.** The set of all binary strings having the substring 00101.
- **Solution**. Following the above idea, we first construct a checker as shown in Figure 2.10(a).

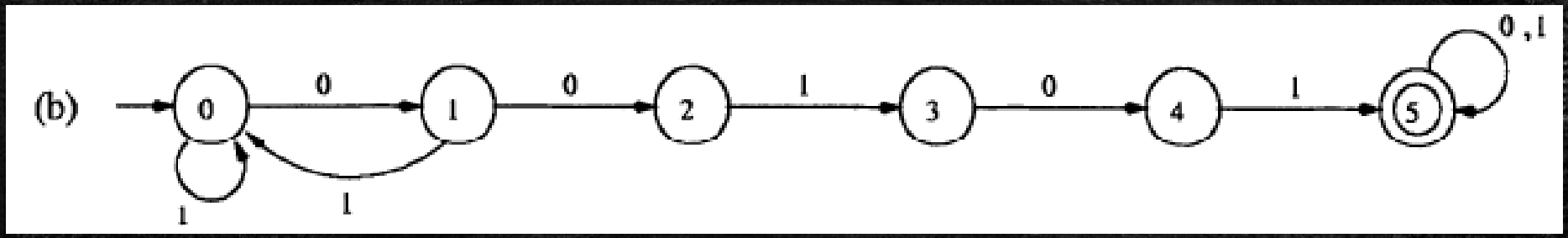
Examples of Deterministic Finite Automata



- Intuitively, for each $i = 0, 1, \ldots, 5$, state q_i means "the past i input symbols just read $a_1 \ldots a_i$" where $a_1 \ldots a_5$ is the target substring 00101. Thus, at state q_0, if we read a new symbol 1, the new string $a_1 \ldots a_i 1 = 1$ (here, $i = 0$) is not a prefix of 00101, and so we need to go back to state q_0.

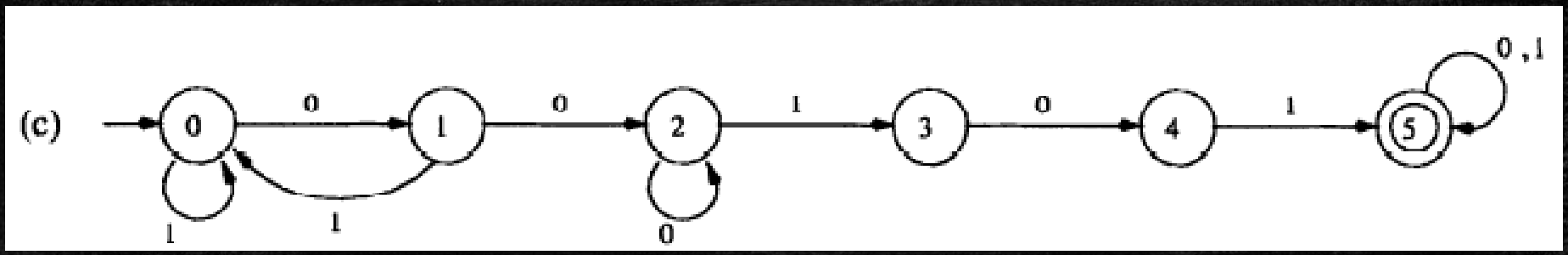
Examples of Deterministic Finite Automata

- Similarly, if a symbol 1 is read at state q_1, neither the string $a_1 1 = 11$ nor the string 1 is a prefix of 00101, and so we let $\delta(q_1, 1) = q_0$. We upgrade the DFA as shown in Figure 2.10(b).



Examples of Deterministic Finite Automata

- Now, consider $\delta(q_2, 0)$. The string $a_1 a_2 0 = 000$ is not a prefix of 00101, but the last two symbols $a_2 0 = 00$ is a prefix of 00101. That is, if the next three input symbols are $1, 0$ and 1, then we should accept the input. So, we define $\delta(q_2, 0) = q_2$ to indicate this partial success. This action is shown in Figure 2.10(c).



Examples of Deterministic Finite Automata

- Based on the same idea, we define $\delta(q_3, 1) = q_0$ (neither $a_1a_2a_3 1 = 0011$ nor any of its suffixes is a prefix of 00101), and $\delta(q_4, 0) = q_2$ (the last two bits of $a_1a_2a_3a_4 0$ are 00 and form a prefix of 00101). The complete DFA is shown in Figure 2.10(d).

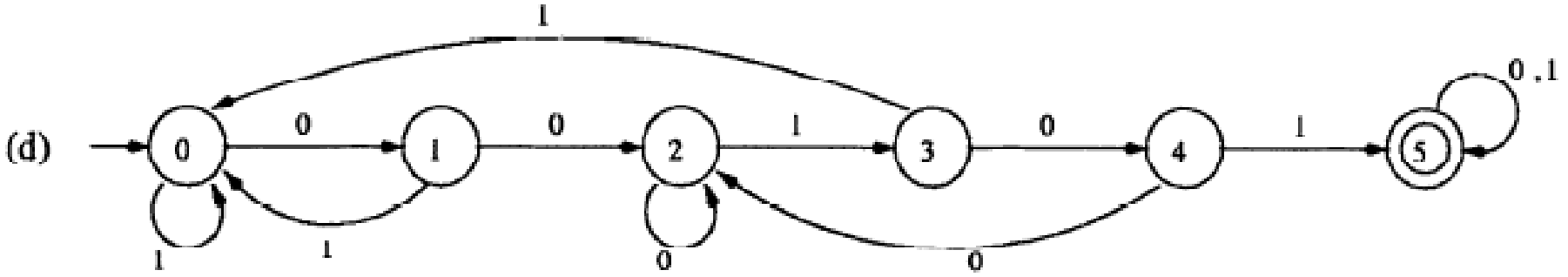


Figure 2.10: Solution to Example 2.9.

Examples of Deterministic Finite Automata

- **Example 2.10.** The set A of all binary strings ending with 01.
- **Solution**. Initially, we build a checker as shown in Figure 2.11(a). Again, states q_0 and q_1 indicate "found no prefix of 01" and "found prefix 0 of 01" respectively. Note, however, that although state q_2 is a final state, it is not a success state since a string must end at state q_2 to be accepted.

Examples of Deterministic Finite Automata

- Now, following the idea of the last example, it is easy to see that we need to define $\delta(q_0, 1) = q_0$ and $\delta(q_1, 0) = q_1$.

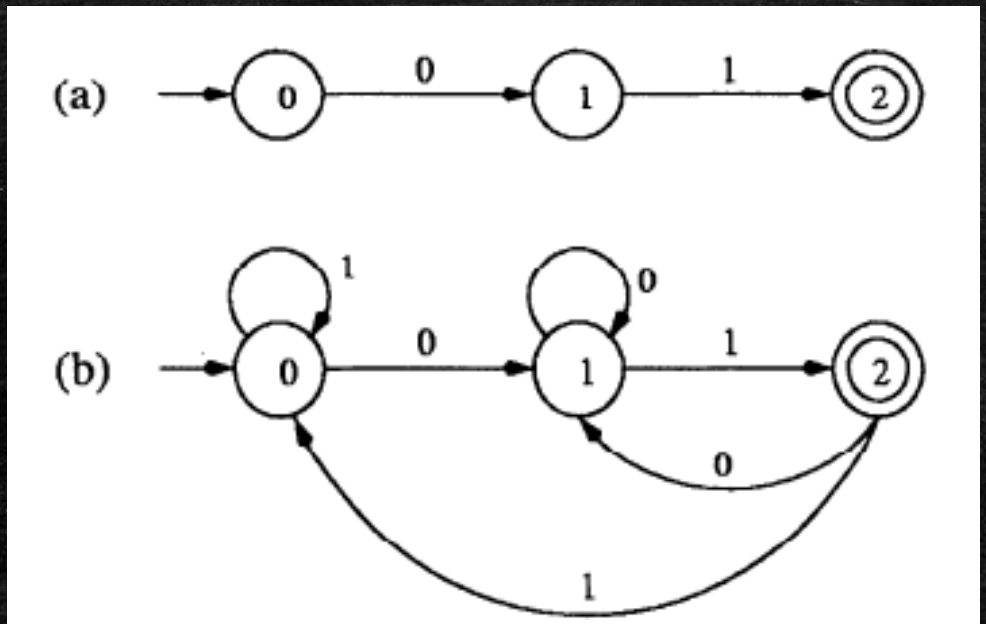


Figure 2.11: Solution to Example 2.10.

Examples of Deterministic Finite Automata

- At state q_2, if we read more input symbols, then we need to follow the same idea to set $\delta(q_2, 0) = q_1$ and $\delta(q_2, 1) = q_0$. The complete DFA is shown in Figure 2.11(b).
- **Example 2.11.** The set of all binary expansions of positive integers which are congruent to zero modulo 5.
- **Solution**. The idea of this DFA is similar to that of the last two examples.

Examples of Deterministic Finite Automata

- We need to set up five states $q_0, q_1, \ldots, q_4$, with each state q_i meaning "the prefix y of the input string read so far has the property of $y \equiv i (mod\ 5)$". That is, we need to define $\delta(q_0, x_1 x_2 \ldots x_k) = q_i$ if $x_1 x_2 \ldots x_k \equiv i (mod\ 5)$.
- How do we determine the edges between these five states from this idea? Recall that the transition function δ of a DFA has to satisfy

$$\delta(\delta(q_0, x), a) = \delta(q_0, xa)$$

Examples of Deterministic Finite Automata

- for any $x \in \{0,1\}^*$ and any $a \in \{0,1\}$. Suppose $\delta(q_0, x) = q_i$ and $\delta(q_0, xa) = q_j$. Then, we must have $x \equiv i(mod\ 5)$ and $xa \equiv j(mod\ 5)$. Thus,

$$j \equiv xa\ (mod\ 5) \equiv 2x + a\ (mod\ 5)$$
$$\equiv 2i + a\ (mod\ 5)$$

- Therefore, we need to define $\delta(q_i, a) = q_j$, where $j \equiv 2i + a\ (mod\ 5)$. For instance, $\delta(q_2, 0) = q_4$ and $\delta(q_2, 1) = q_0$. See Figure 2.12 for the other edges. In addition, state q_0 is the unique final state, since $\delta(q_0, x) = q_0$ means $x \equiv 0\ (mod\ 5)$.

Examples of Deterministic Finite Automata

- Finally, we note that a binary expansion of a positive integer always begins with the symbol 1. So, we need to add a new initial state and a failure state, as shown in Figure 2.12.

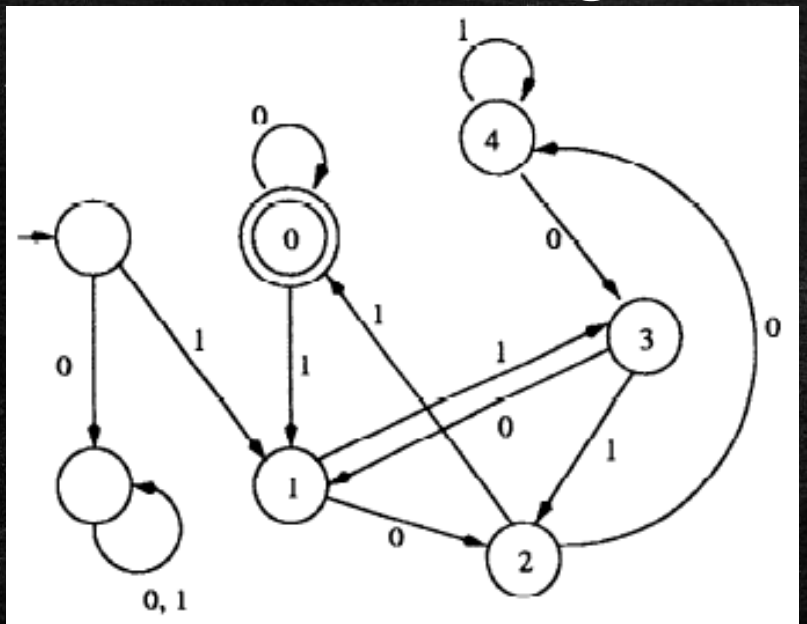


Figure 2.12: DFA for Example 2.11.

Examples of Deterministic Finite Automata

- Note that the set $A \cup \{1\}$, where A is the set defined in Example 2.10, may be regarded as the set of binary expansions of integers congruent to 1 modulo 4, with leading zeros allowed. Using the idea of the above example, we get a new DFA for set $A \cup \{1\}$ as shown in Figure 2.13. Note that the states q_0 and q_2 in this DFA can be merged into one, and the simplified DFA is just the one shown in Figure 2.11(b), except that the initial state has been changed (to state q_1 of Figure 2.11(b)).

Examples of Deterministic Finite Automata

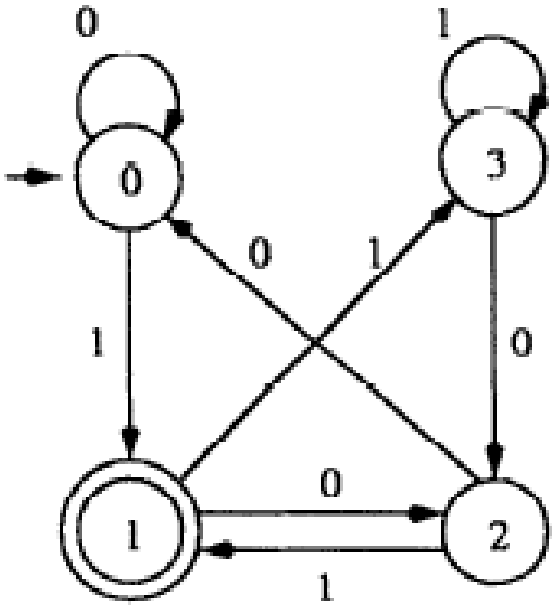


Figure 2.13: DFA for set $A \cup \{1\}$.

Examples of Deterministic Finite Automata

- **Example 2.12.** The set of all binary strings having a substring 00 or ending with 01.
- **Solution**. This language is the union of two languages $(0+1)^*00(0+1)^*$ and $(0+1)^*01$. In Examples 2.8 and 2.10, we have already constructed two DFA's for these two languages. To check whether an input string x belongs to the union of these two languages, we can run these two DFA's in parallel. For example, suppose $x = 0101$.

Examples of Deterministic Finite Automata

- In the first DFA M_1 shown in Figure 2.9(c), the computation path of x is $(q_0, q_1, q_0, q_1, q_2)$, and in the second DFA M_2 in Figure 2.11(b), the computation path is $(q_0, q_1, q_2, q_1, q_2)$. Since the second path ends at a final state, x is accepted in this parallel simulation.
- One idea on how to build a DFA for the union of these two languages is, then, to combine the two DFA's into one such that, at each step, the new DFA would keep track of the computation paths of both DFA's.

Examples of Deterministic Finite Automata

- This suggests us to consider a *product automaton* $M = M_1 \times M_2$ as follows: Assume that the first DFA is $M_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$ and the second DFA is $M_2 = (Q_2, \Sigma, \delta_2, q_0, F_2)$. (Note that Q_1 and Q_2 may have states of the same name but playing different roles in two DFA's.) Then, the state set Q of M is the cross product of the state sets of M_1 and M_2; that is, $Q = Q_1 \times Q_2$. We denote each member of Q as $[q_i, q_j]$, where $q_i \in Q_1$ and $q_j \in Q_2$. The initial state of M is $[q_0, q_0]$.

Examples of Deterministic Finite Automata

- At each state $[q_i, q_j]$ in Q, we simulate both computations of M_1 and M_2 in parallel by

$$\delta([q_i, q_j], a) = [\delta_1(q_i, a), \delta_2(q_j, a)].$$

- For instance, the computation path of $x = 0101$ in this product automaton is $[q_0, q_0]$, $[q_1, q_1]$, $[q_0, q_2]$, $[q_1, q_1]$, $[q_0, q_2]$. Furthermore, if $q_i \in F_1$ or $q_j \in F_2$, then we let $[q_i, q_j] \in F$. That is, $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

Examples of Deterministic Finite Automata

- From the above description, it is clear that M accepts the union of the DFA's M_1 and M_2. We show this product DFA M in Figure 2.14, where each vertex with the label (i, j) denotes the state $[q_i, q_j]$.
- Two facts about this DFA are worth mentioning: First, since states $[q_0, q_1]$, $[q_1, q_0]$ and $[q_1, q_2]$ are unreachable from the initial state $[q_0, q_0]$ we omitted them in Figure 2.14.

Examples of Deterministic Finite Automata

- Second, states $[q_2, q_1]$, $[q_2, q_0]$ and $[q_2, q_2]$ can be merged into a single success state, since all three states are final states and there is no way to leave these three states once we get there.

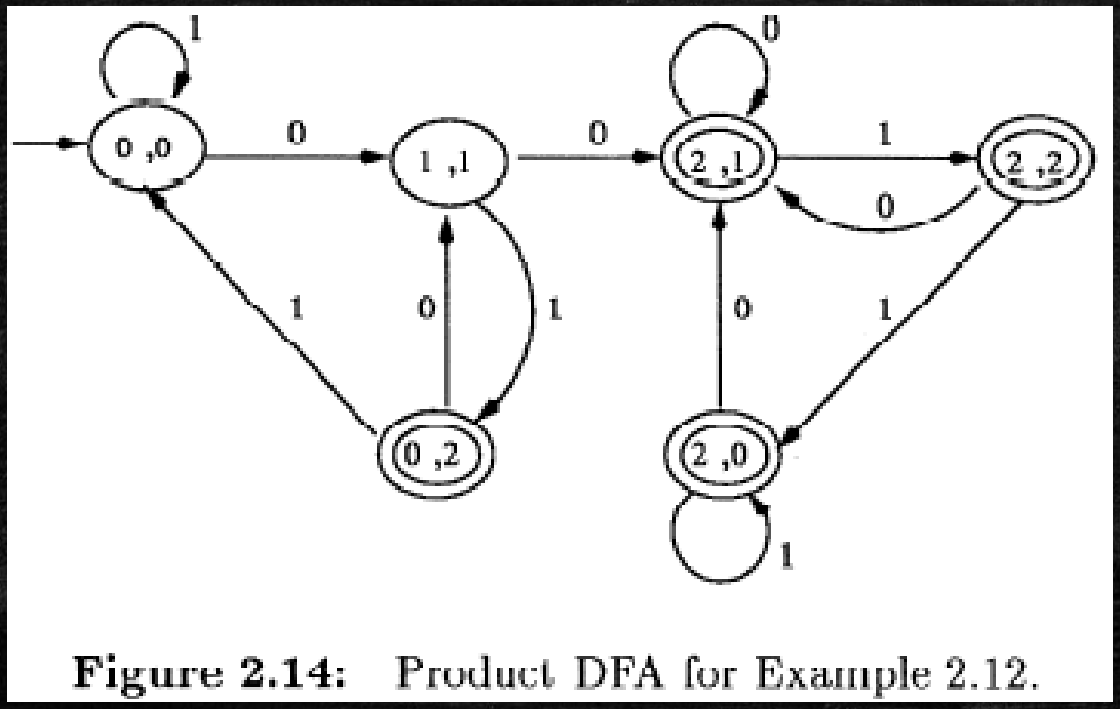


Figure 2.14: Product DFA for Example 2.12.

Examples of Deterministic Finite Automata

- **Example 2.13.** The set of all binary strings having a substring 00 and ending with 01.
- **Solution 1**. This language is the intersection of two languages $(0+1)^*00(0+1)^*$ and $(0+1)^*01$. In Example 2.12, we constructed a product DFA to accept the union of these two languages. Here, we can use the same product DFA, except that we will define the set F of final states to consist of states in which both components are final states; that is, $F = F_1 \times F_2$.

Examples of Deterministic Finite Automata

- The transition diagram of the resulting DFA is just like that of Figure 2.14, except that the final set consists of only one state $[q_2, q_2]$.

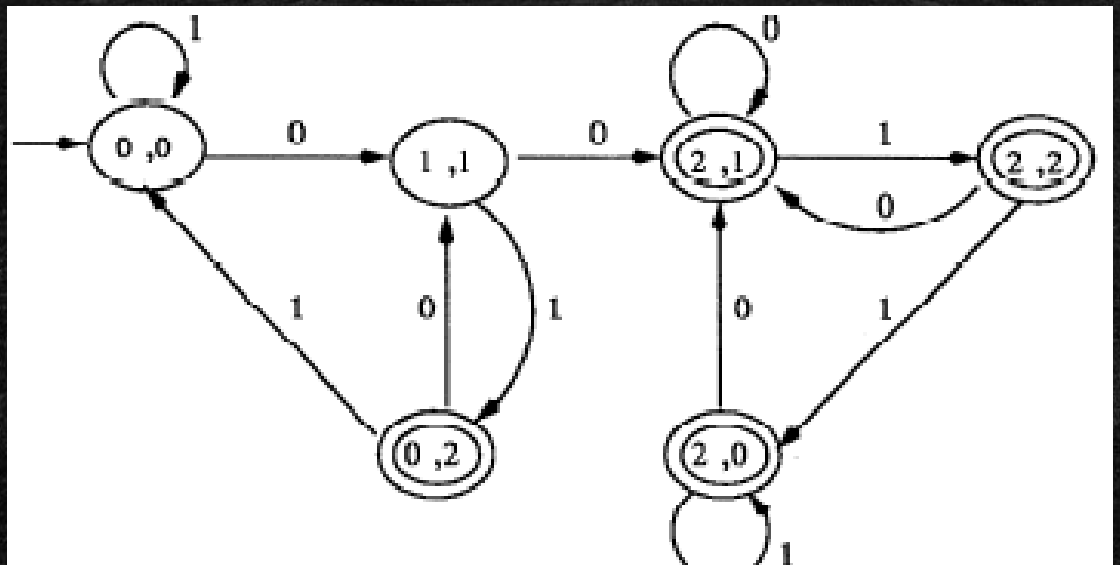


Figure 2.14: Product DFA for Example 2.12.

Examples of Deterministic Finite Automata

- **Example 2.14.** The set of all binary strings having a substring 00 but not ending with 01.
- **Solution**. This language is the difference of language $(0+1)^*00(0+1)^*$ minus language $(0+1)^*01$. Thus, we can use the same product DFA as we did in previous Examples, except that we need to choose the set of final states to consist of every state in which the first component is a final state of the first DFA and the second component is a non-final state of the second DFA;

Examples of Deterministic Finite Automata

- that is, our new final set is $F = F_1 \times (Q_2 - F_2)$. Its transition diagram is like that of Figure 2.14, with the final states $\{[q_2, q_0], [q_2, q_1]\}$.

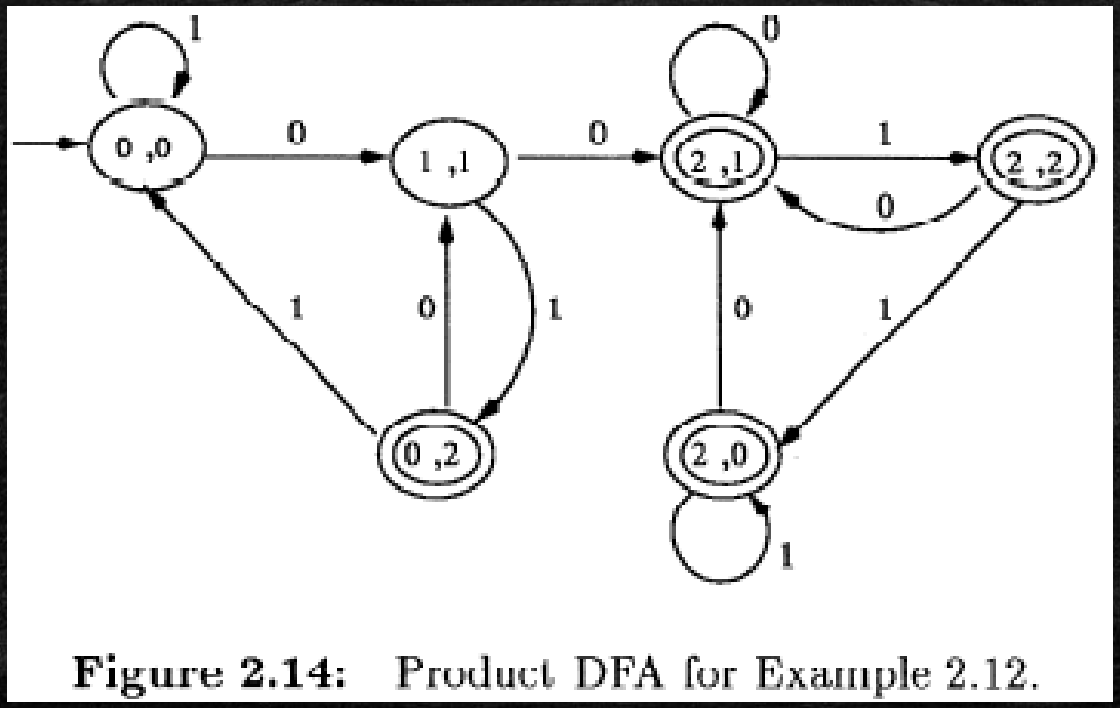


Figure 2.14: Product DFA for Example 2.12.

Examples of Deterministic Finite Automata

- A special case of subtraction is complementation: $\bar{A} = \Sigma^* - A$. There is a simpler construction in this case. Note that in DFA $M = (Q, \Sigma, \delta, q_0, F)$, for any input string x, $x \in L(M)$ if and only if $\delta(q_0, x) \in F$. Equivalently, $x \notin L(M)$ if and only if $\delta(q_0, x) \notin F$. It follows that the DFA $(Q, \Sigma, \delta, q_0, Q - F)$ accepts the complement of $L(M)$.
- **Example 2.15.** The set L of all binary strings in which every block of four consecutive symbols contains a substring 01.

Examples of Deterministic Finite Automata

- **Solution**. The condition "every block of four consecutive symbols contains a substring 01" is a *global* condition, which appears difficult to verify. By considering the complement $\overline{L}$, we turn this condition into a simpler *local* condition: $\overline{L}$ contains binary strings with a substring $0000, 1000, 1100, 1110,$ or 1111. We first construct a DFA accepting $\overline{L}$ and then change all final states into non-final states and all non-final states into final states. A solution is shown in Figure 2.16.

Examples of Deterministic Finite Automata

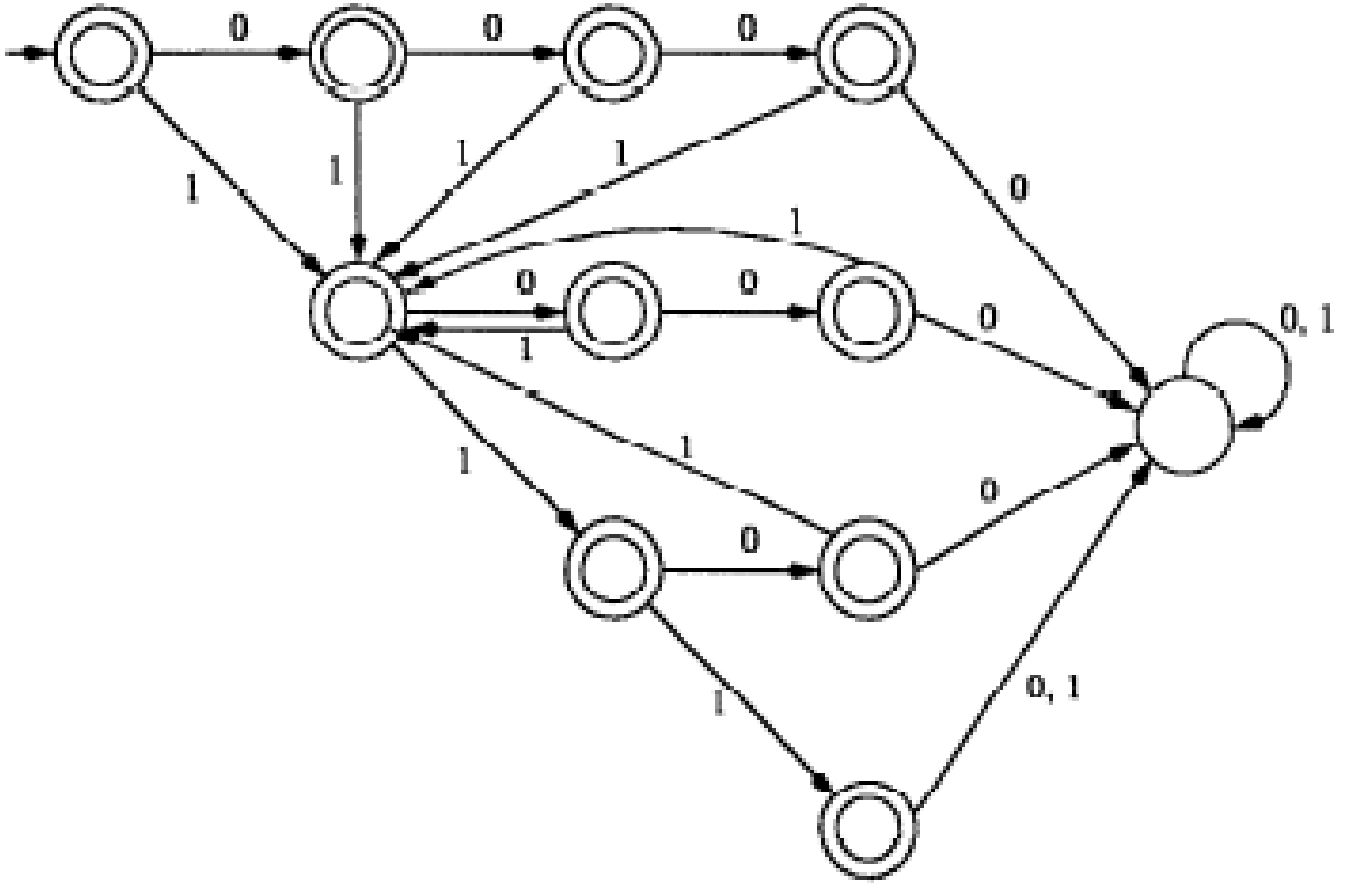


Figure 2.16: Solution to Example 2.15.

Examples of Deterministic Finite Automata

- **Theorem 2.16 .** The class of languages accepted by DFA's is closed under union, intersection, subtraction, and complementation.

Nondeterministic Finite Automata

- A *nondeterministic finite automaton* (NFA) $M = (Q, \Sigma, \delta, q_0, F)$ is defined in the same way as a DFA except that multiple-state transitions and ε-transitions are allowed.
- What is a multiple-state transition? It means that at each move, there could be more than one next state. That is, for any state q and any input symbol a, the value of $\delta(q, a)$ is a subset of Q, where $\delta(q, a) = \{p_1, p_2, \ldots, p_k\}$ means that the next state from state q, after reading a, can be any one of p_1, p_2, … or p_k.

Nondeterministic Finite Automata

- A special case is that $\delta(q, a)$ could be the empty set $\emptyset$. This means that the machine has no next state and hangs, and the input is rejected regardless of what remaining input symbols are. It is the equivalent of going to a failure state in a DFA.
- In addition to multiple-state transitions, we also allow ε-transitions (or, ε-moves) in an NFA. An ε-transition is a move in which the tape head does not do anything (it neither reads nor moves), but the state can be changed.

Nondeterministic Finite Automata

- In other words, we allow a transition like $\delta(q, a) = \{p_1, p_2, \ldots, p_k\}$, which means that state q can be changed to any one of p_1, p_2, ... or p_k, without reading a symbol from the input.
- Therefore, the transition function δ is, formally, a function of the form

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

- where 2^Q denotes the collection of all subsets of Q.

Nondeterministic Finite Automata

- The following is an example of a transition function of an NFA M_1:

$\delta(q, a)$	0	1	ε
q_0	$\emptyset$	$\{q_0, q_1\}$	$\{q_1\}$
q_1	$\{q_2\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_2\}$	$\emptyset$	$\{q_1\}$

- NFA's, like DFA's, can also be represented by transition diagrams. In the transition diagram, we still use a vertex to represent a state and a labeled edge to represent a move, except that we allow multiple edges from one vertex to other vertices with the same label.

Nondeterministic Finite Automata

- That is, if $\delta(q, a) = \{p_1, p_2, \ldots, p_k\}$, then we draw k edges from state q to each of $p_1, p_2, \ldots, p_k$, and each with a label a. For instance, the transition diagram of the transition function of M_1 above is as shown in Figure 2.17. We also made $F = \{q_2\}$.
- On an input x, an NFA may have more than one computation path. For instance, for NFA M_1 shown in Figure 2.17, the input 01 has the following three computation paths:

Nondeterministic Finite Automata

$$q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{0} q_2 \xrightarrow{\varepsilon} q_1 \xrightarrow{1} q_1,$$

$$q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{0} q_2 \xrightarrow{\varepsilon} q_1 \xrightarrow{1} q_2,$$

$$q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{0} q_2 \xrightarrow{\varepsilon} q_1 \xrightarrow{1} q_2 \xrightarrow{\varepsilon} q_1.$$

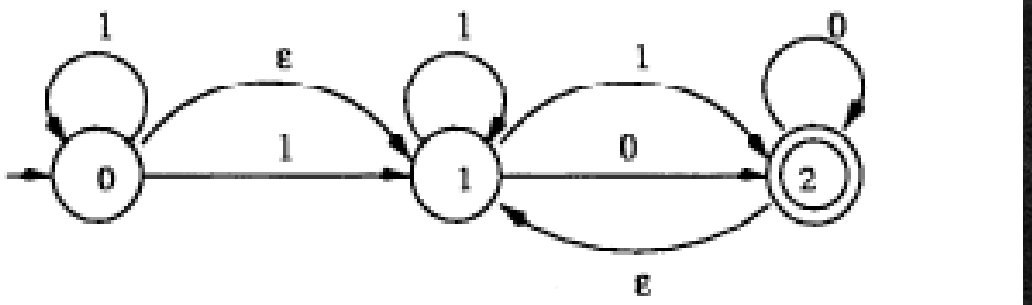


Figure 2.17: The transition diagram of an NFA.

Nondeterministic Finite Automata

- Note that in the last two paths, after the NFA reads all input symbols 01, it can choose to halt in state q_2 or to use the ε-move to move to state q_1. In general, the computation paths for an input x form a computation tree since they all start with the same state q_0 and then branch out to different states. Figure 2.18 shows the computation tree of the NFA M_1 on input 01. (In Figure 2.18, we added an edge $q_2 \xrightarrow{\varepsilon} q_2$ to indicate that the second path ends at q_2.)

Nondeterministic Finite Automata

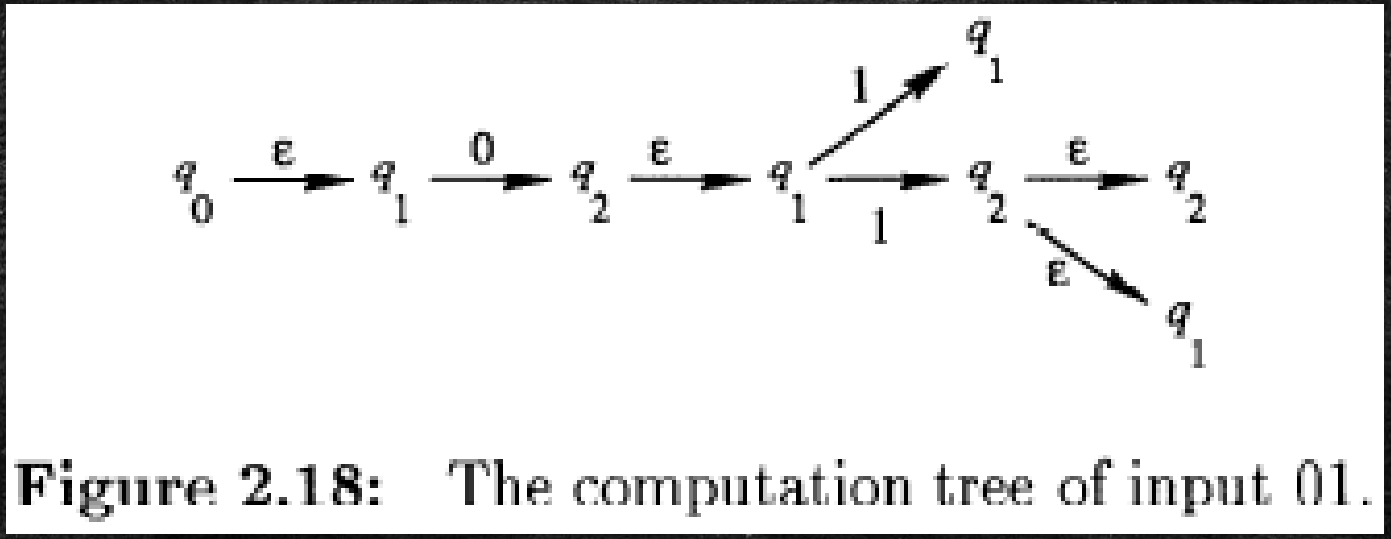


Figure 2.18: The computation tree of input 01.

- Some of these computation paths lead to final states and some do not. How do we define the notion of an NFA accepting an input in such a situation?

Nondeterministic Finite Automata

- The answer is that the NFA accepts an input x if at least one computation path on x leads to a final state. For instance, in the above example, since the second computation path ends at state $q_2 \in F$ after the NFA reads the input 01, the string 01 is accepted by the NFA M_1.
- To formally define the notion of an NFA M accepting an input x, we need to define the notion of ε-closure.

Nondeterministic Finite Automata

- The ε-closure of a subset $A \subseteq Q$ is the set of states that can be reached from a state q in A by ε-moves (including the move from q to q). That is,

$$\varepsilon\text{-closure}(A) =$$
$$\{p \in Q \mid p \in A \text{ or } \exists q_0, q_1, \ldots, q_m (q_0 \in A, q_m = p \text{ and } q_{i+1} \in \delta(q_i, \varepsilon), i = 0, 1, \ldots, m-1)\}$$

Nondeterministic Finite Automata

- For instance, in the above example,

$$\varepsilon\text{-closure}(\{q_0\}) = \{q_0, q_1\}$$

$$\varepsilon\text{-closure}(\{q_2\}) = \{q_1, q_2\}$$

- Now, we extend the transition function δ to the domain $2^Q \times (\Sigma \cup \{\varepsilon\})$ by

$$\delta(A, a) =$$

$$\varepsilon\text{-}closure\left(\bigcup_{q \in \varepsilon\text{–closure}(A)} \delta(q, a)\right)$$

Nondeterministic Finite Automata

- and then further extend it to the domain $2^Q \times \Sigma^*$ as follows

$$\delta(A, \varepsilon) = \varepsilon\text{-}closure(A),$$

$$\delta(A, xa) = \delta(\delta(A, x), a), \text{ if } x \in \Sigma^* \text{ and } a \in \Sigma.$$

- For instance, in the above example, we have $\delta(\{q_0\}, 0) = \{q_1, q_2\}$ and $\delta(\{q_1, q_2\}, 1) = \{q_1, q_2\}$. Therefore, $\delta(\{q_0\}, 01) = \{q_1, q_2\}$. Note that $\delta(\{q_0\}, x)$ is the set of all leaves in all computation paths of x.

Nondeterministic Finite Automata

- We can now formally define that an NFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts input x if $\delta(\{q_0\}, x) \cap F \neq \emptyset$. For an NFA M, we let $L(M)$ denote the set of all strings accepted by M; that is

$$L(M) = \{x \in \Sigma^* \mid \delta(\{q_0\}, x) \cap F \neq \emptyset\}$$

- It is often easier to design NFA's than DFA's.

Nondeterministic Finite Automata

- **Example 2.17.** Find an NFA that accepts the set of binary strings having a substring 010.
- **Solution**. We show the NFA M in Figure 2.19. Note that it is just the checker for substring 010 plus two loops from state q_0 to q_0 with labels 0 and 1. These two loops allow the machine to wait until it successfully checks the substring 010. With these two *waiting* loops, we do not need to define $\delta(q_1, 0) = q_1$ and $\delta(q_2, 1) = q_0$, as we did in a DFA. Instead, we simply let $\delta(q_1, 0) = \delta(q_2, 1) = \emptyset$.

Nondeterministic Finite Automata

- We also show, in Figure 2.20, the *computation tree* of M on input $x = 1001010$. Note that there are two occurrences of 010 in x: and so there are two different accepting paths for x.

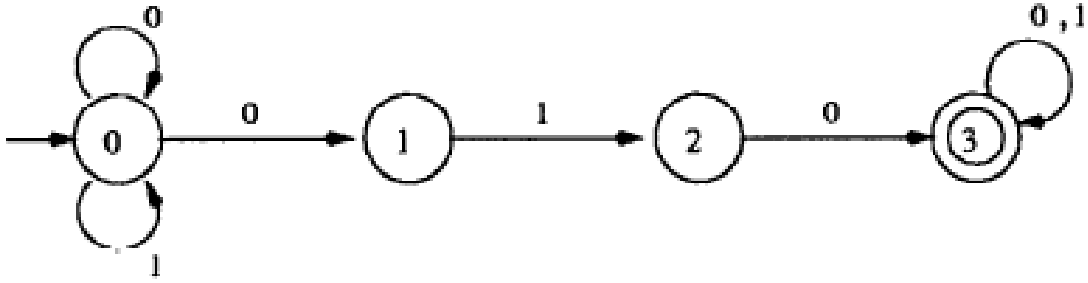


Figure 2.19: NFA for Example 2.17.

Nondeterministic Finite Automata

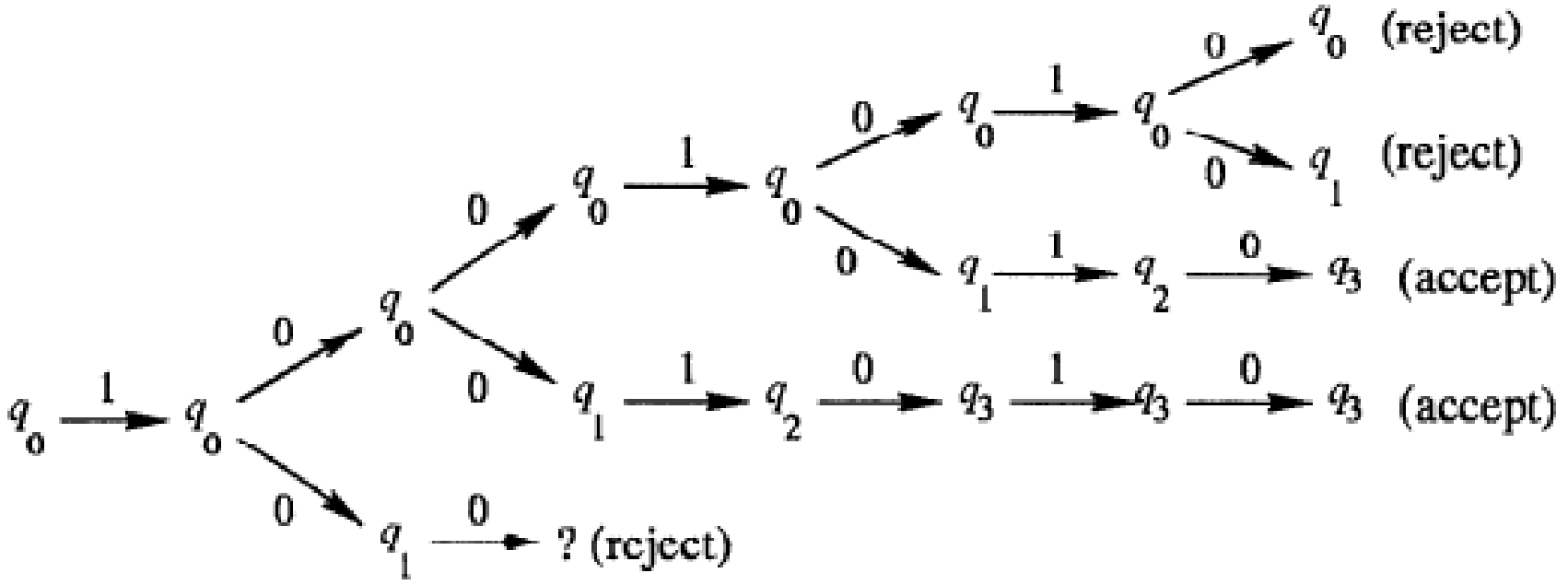


Figure 2.20: The computation tree of input 1001010.

Nondeterministic Finite Automata

- **Example 2.18.** Find an NFA that accepts the set of binary strings beginning with 010 or ending with 110.
- **Solution**. The set of binary strings beginning with 010 is accepted by the NFA of Figure 2.21(a). The set of binary strings ending with 110 is accepted by the NFA of Figure 2.21(b). Note that the final state q_3 has no outlet; that is, $\delta(q_3, 0) = \delta(q_3, 1) = \emptyset$. So, if a computation path of a string x reaches state q_3 before x is completely read, it is a rejecting path.

Nondeterministic Finite Automata

- Now, we combine these two NFA's by adding an initial state and two ε-edges to the two old initial states to form the final NFA, as shown in Figure 2.21(c).

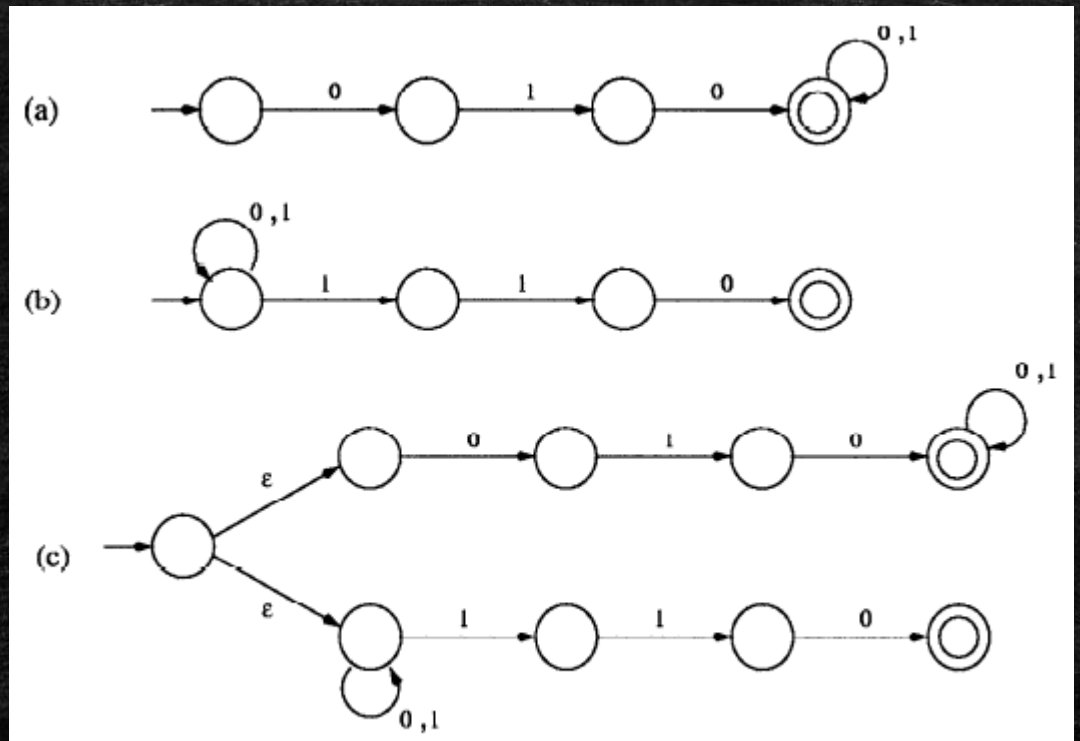


Figure 2.21: Solution to Example 2.18.

Nondeterministic Finite Automata

- **Example 2.19.** Find an NFA that accepts the set of binary strings with at least two occurrences of substring 01 and ends with 11.
- **Solution**. We use the ε-moves to connect three simple NFA's into one, as shown in Figure 2.22. Note that there is a special case where the second occurrence of the substring 01 runs into the suffix 11. It presents no problem for the design of the NFA; we simply add one extra ε-edge.

Nondeterministic Finite Automata

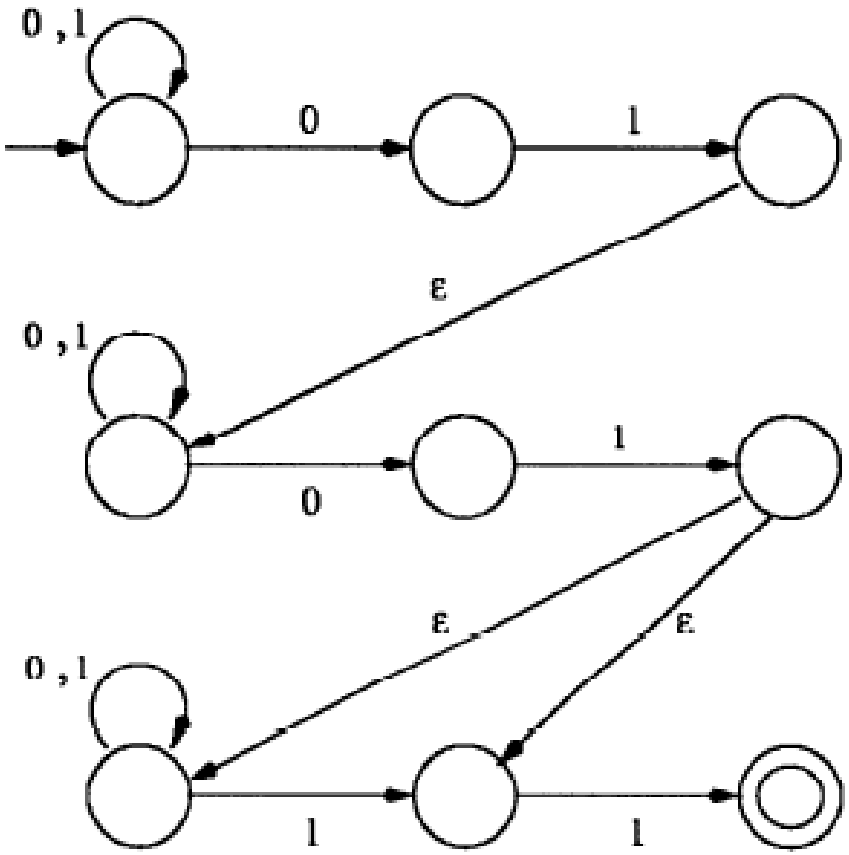
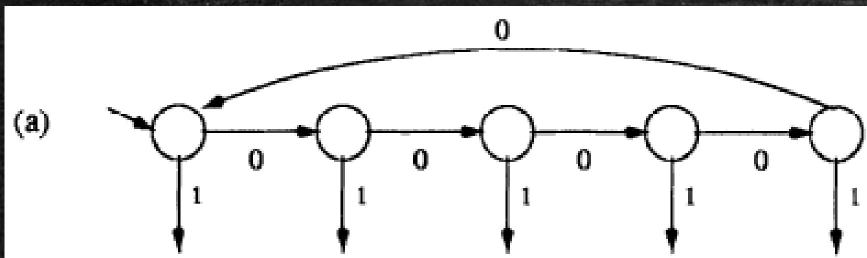


Figure 2.22: The Solution to Example 2.19.

Nondeterministic Finite Automata

- **Example 2.20.** Find an NFA that accepts the set $\{0^n 1 0^m\}$.
- **Solution.** We apply the idea of product automata to construct the required NFA. First, we construct a simple NFA M_1 to separate the input strings into five groups according to the value of $n \ mod \ 5$ (see Figure 2.23(a)). Next, for each group, we construct a copy of M_1 to find the value $m \ mod \ 5$. Then, we assign final states to each copy of M_1 accordingly. Figure 2.23(b) shows the complete DFA.

Nondeterministic Finite Automata



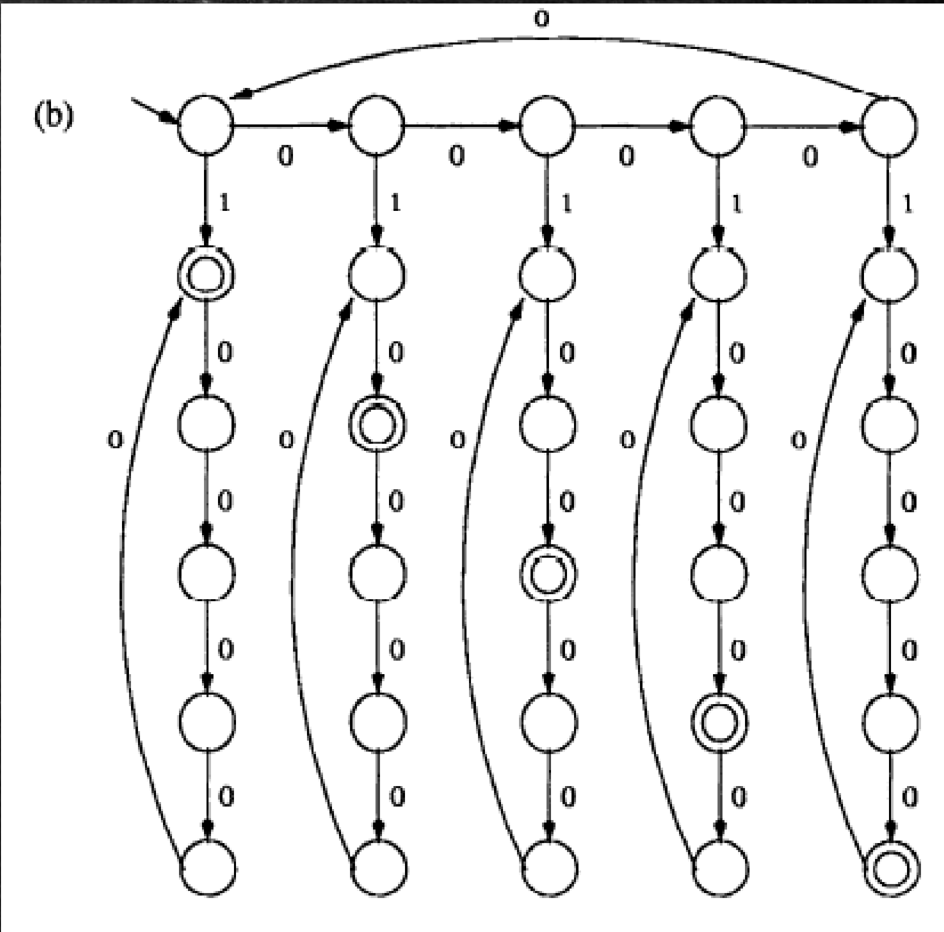


Figure 2.23: Solution to Example 2.20.