Chapter 2

Automata

1. Show that there exists a deterministic finite automaton with just one state that can accept an infinite set of strings.

   Solution: The single state is the start state and the final state and has a loop back to itself labeled *a*. This DFA accepts *λ*, *a*, *aa*, *aaa*, … - an infinite set of strings.

2. Show that for the alphabet {*a*} there cannot be a deterministic finite automaton with just two states that accepts more than two strings but only a finite number of strings.

   Solution: If the alphabet has just one symbol {*a*}, then the DFA without loops can accept only two strings *λ* and *a*. If a loop is added to either state or back from the second state to the first state, the DFA accepts infinitely many strings.

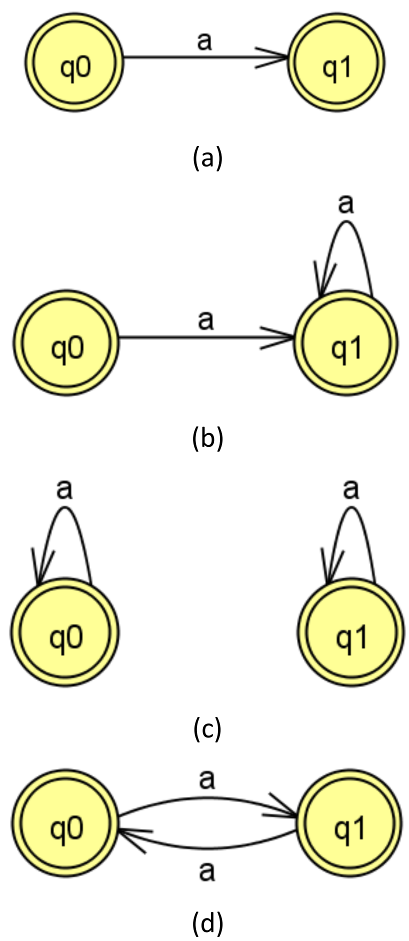


Figure (a) Accepts two strings; (b), (c), (d) accept infinite sets of strings.

   If the alphabet has more symbols, then of course the DFA can accept more strings; if the alphabet has *n* symbols, the DFA can accept *n* + 1 strings.

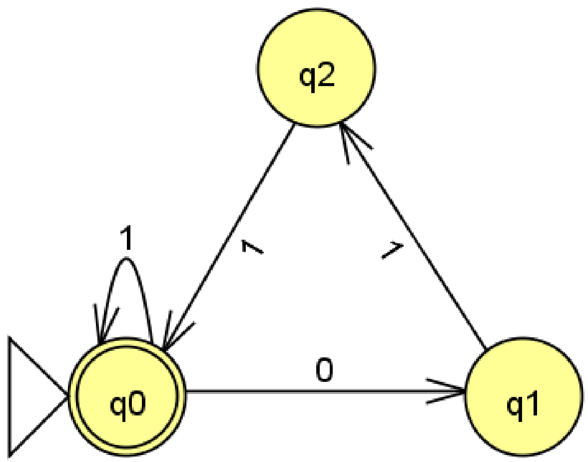
3. What is the minimum number of states in a finite automaton for implementing a soft-drink vending machine that accepts Rs. 2, 5 and 10 coins and returns correct change after delivering the drink? The price of the soft-drink is Rs. 22 and it delivers the drink as soon as the amount put in by the user reaches or exceeds Rs. 22.

Solution: Since coins of all three denominations can be inserted in any order, the maximum amount that can be inserted is 31 (i.e., 21 + 10). The automaton needs states for each of 0, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, and 31 out of which the last ten are accepting states. Thus, the minimum number of states is 30.

A. Construct a finite automaton for each of the following:

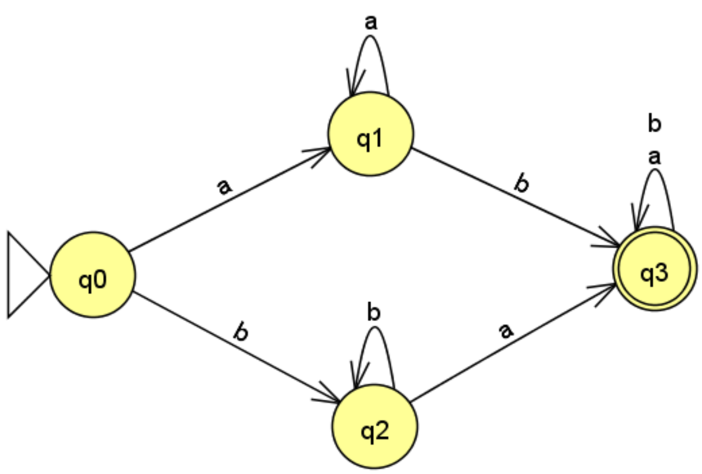
4. Binary strings in which every 0 is followed by 11.

Solution:



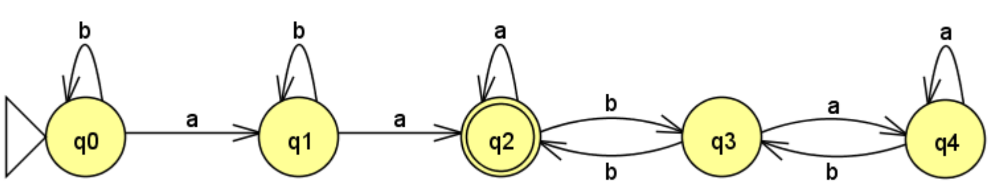
5. Strings containing at least one *a* and at least one *b*.

Solution: Either the *a* can come first or the *b*. The DFA looks for the first *a* and the first *b* in either order.



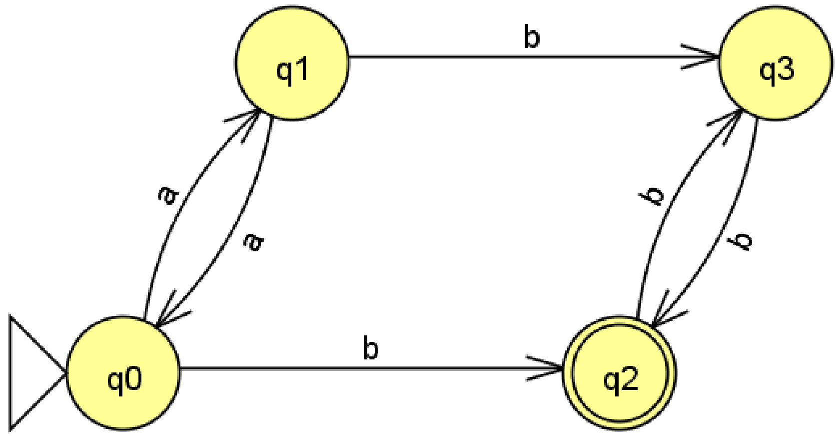
6. Strings containing at least two *a* s and ending with an even number of *b* s.

Solution: Note that strings must end with an even number of *b* s. Thus, if an *a* occurs in q_3 or q_4, two more *b* s must occur before the strings can be accepted in q_2.



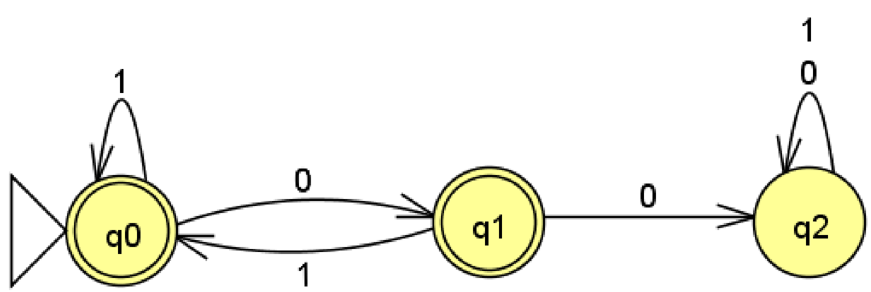
7. Some number of *a* s followed by some number of *b* s with the total length of the string being odd.

Solution: If the string has an odd number of *a* s, it reaches state q_1 and there must be an even number of *b* s to accept the string. Similarly, if the number of *a* s is even, in state q_0, an odd number of *b* s is needed to finally reach q_2.



8. Binary strings with no consecutive 0 s. Show the computation for the input string w_1 = 1011101 and for w_2 = 1001.

Solution:



The moment two consecutive 0 s are encountered, the DFA reaches the reject state q_2. The DFA accepts its input in both the other states. w_1 = 1011101 is accepted via

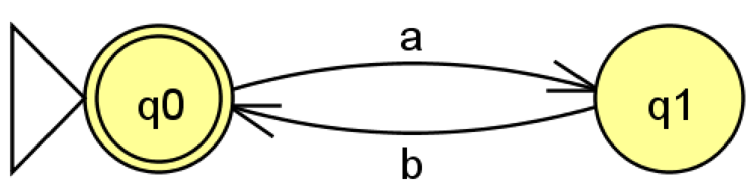
$q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_0$

w_2 = 1001 is rejected via

$q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_2$

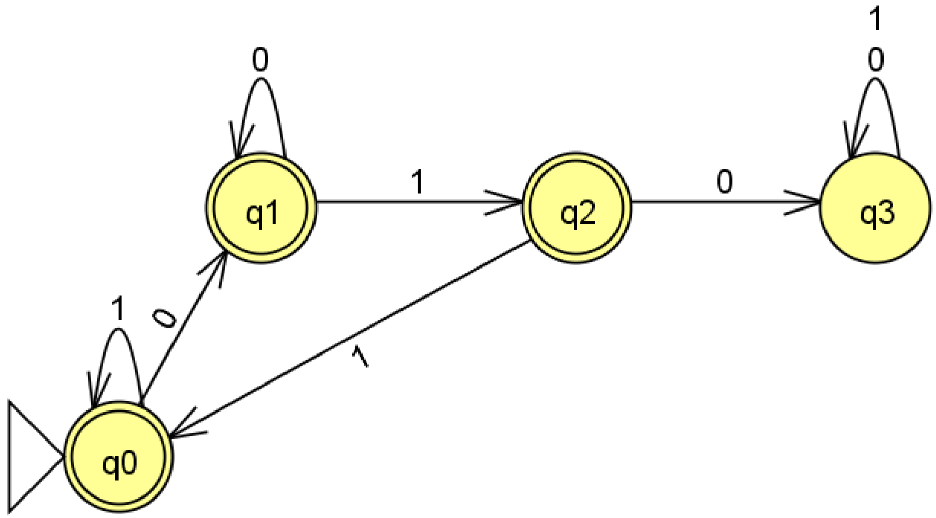
9. Strings over the alphabet {*a*, *b*} of the form $(ab)^n$, for example, *ababab*.

Solution:



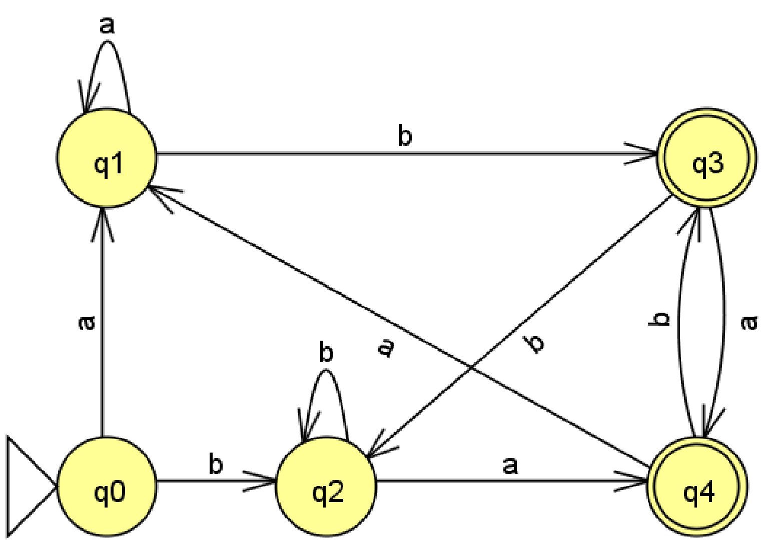
10. Strings over the binary alphabet that do not contain the sub-string 010.

Solution:



11. Strings over the alphabet {*a*, *b*} that end with either *ab* or *ba*. Show the computation for the input string *w* = *abab**aab*.

Solution:

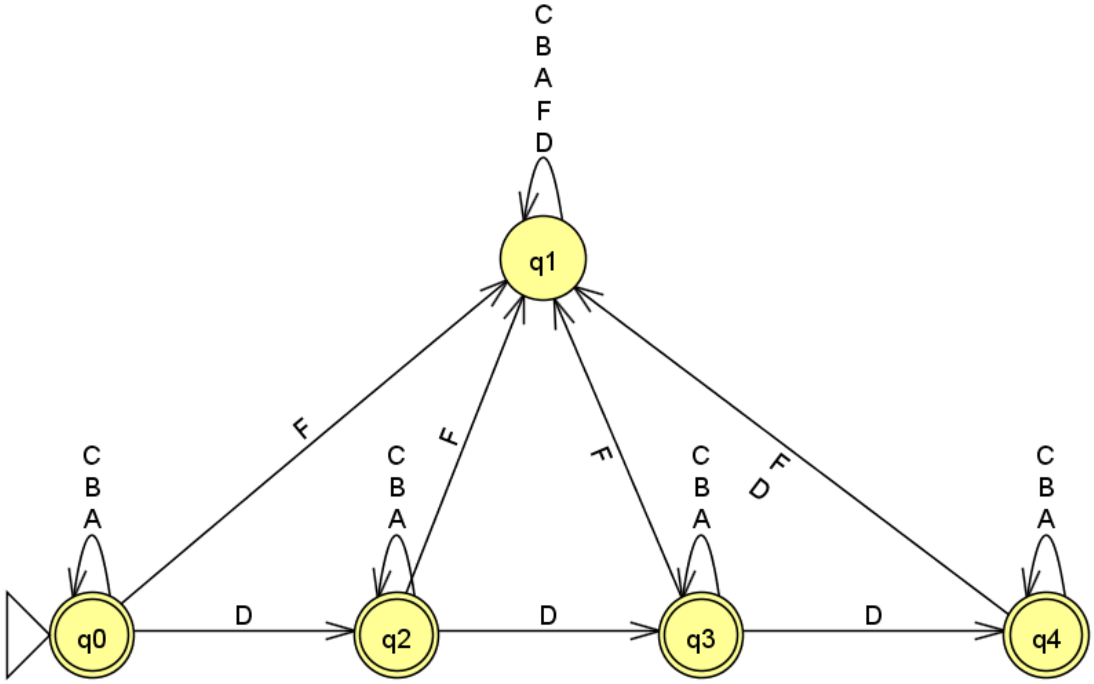


Note that state q_1 can only be reached on seeing the symbol *a* and state q_2 only be reached on seeing the symbol *b*. *w* = *ababaab* is accepted via

$q_0 \rightarrow q_1 \rightarrow q_3 \rightarrow q_4 \rightarrow q_3 \rightarrow q_4 \rightarrow q_1 \rightarrow q_3$

12. Student grades in an examination are represented with the letters {*A*, *B*, *C*, *D*, *F*}. An input string such as *ABFCAD* indicates the grades obtained by a student in six different subjects. The automaton must accept only those students who have scored at most three *D* s and no *F* s.

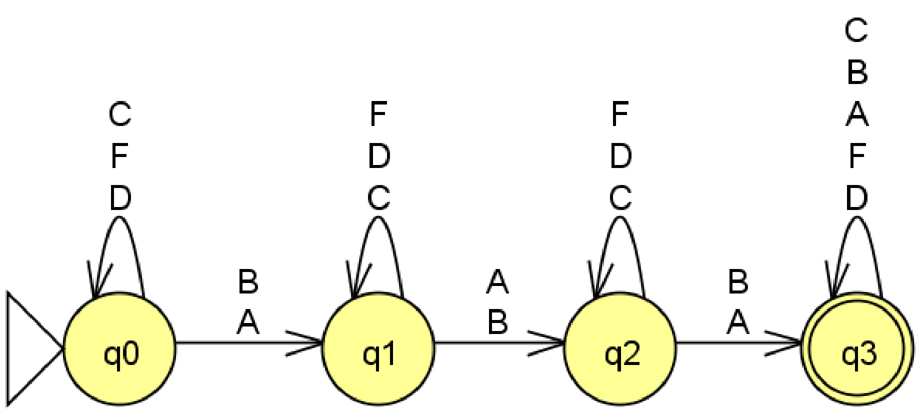
Solution:



An F grade in any state takes the DFA to the reject state q_1. A fourth D also takes the automaton to the same reject state. Everything else is accepted.

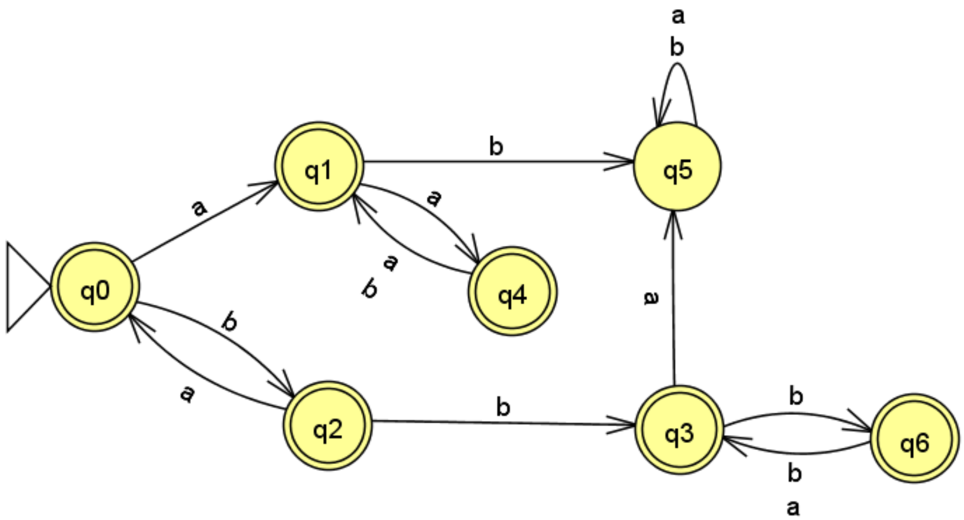
13. The automaton must accept only those students who have scored at least three *A* s or *B* s (overall).

Solution:



14. Strings over {*a*, *b*} in which either all even-numbered symbols are *a* or all odd-numbered symbols are *b*. Show the computation for the string w_1 = *aababaa* and for w_2 = *babaab*.

Solution:



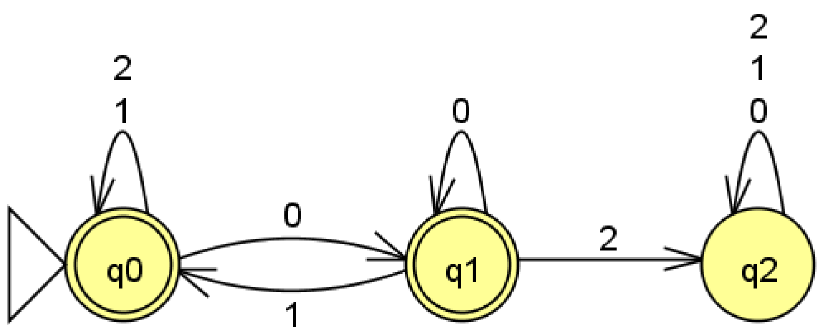
As long as the DFA is in states q_0 or q_2, it is satisfying both requirements. If it goes to q_1, then the requirement for *b* is no longer met. It is looking to satisfy the requirement for *a*. Similarly, in the q_3 branch, it is looking for *b* s in odd numbered positions.

The string w_1 = *aababaa* is accepted via $q_0 \rightarrow q_1 \rightarrow q_4 \rightarrow q_1 \rightarrow q_4 \rightarrow q_1 \rightarrow q_4 \rightarrow q_1$

and w_2 = *babaab* is rejected via $q_0 \rightarrow q_2 \rightarrow q_0 \rightarrow q_2 \rightarrow q_0 \rightarrow q_1 \rightarrow q_5$

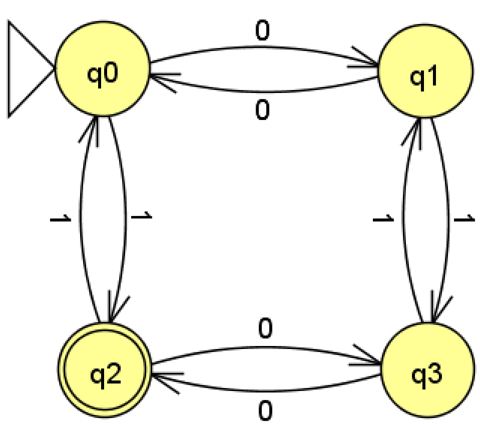
15. The automaton must reject any string over {0, 1, 2} where a 2 is immediately preceded by a 0. It should accept all other strings.

Solution:



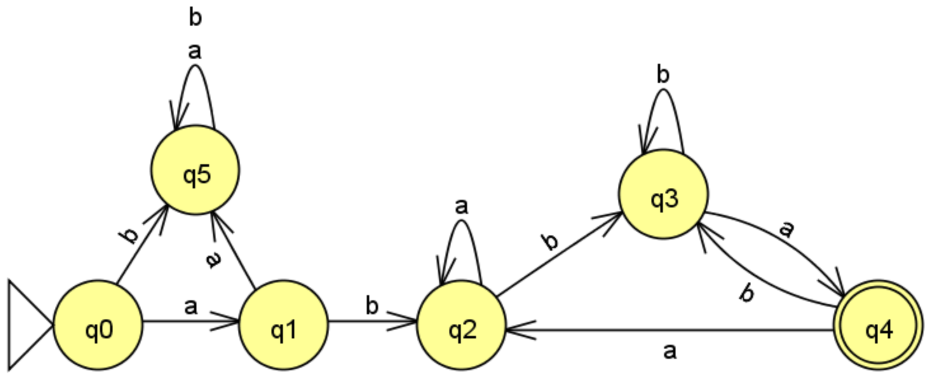
16. Binary strings with an even number of 0 s and an odd number of 1 s.

Solution:



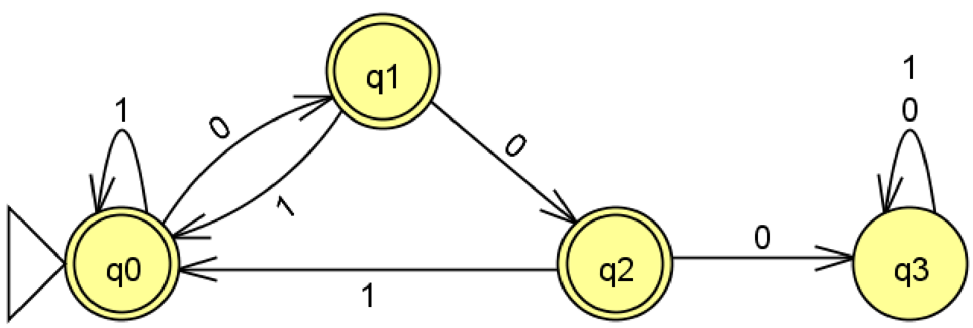
17. Strings of the form *abwba* over the alphabet {*a*, *b*}, where *w* is any string over the same alphabet.

Solution:



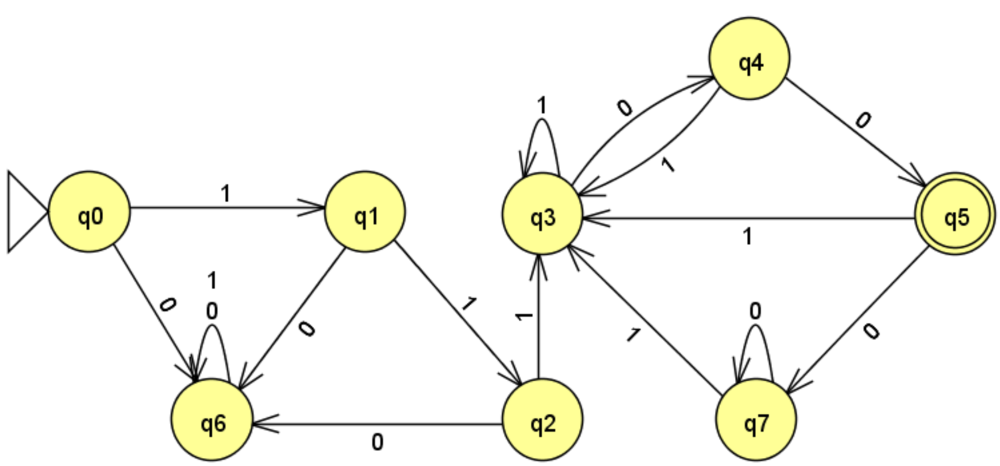
18. The set of all binary strings that do not contain three or more consecutive 0 s.

Solution:



19. The set of all binary strings that start with 111 and end with 100. Assume that each string is at least five characters long. Show the computation for the string w_1 = 111010100 and for w_2 = 11100.

Solution:

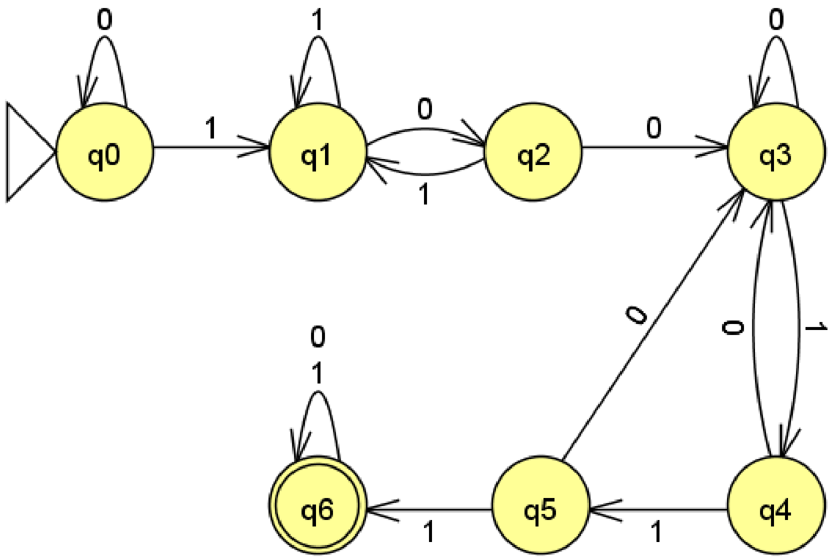


The string w_1 = 111010100 is accepted via

$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_4 \rightarrow q_3 \rightarrow q_4 \rightarrow q_5$
and w_2 = 11100 is also accepted via
$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_4 \rightarrow q_5$

20. The set of all binary strings having a sub-string 100 before a sub-string 111 (both must be present). Show the computation for the string w = 00100110111.

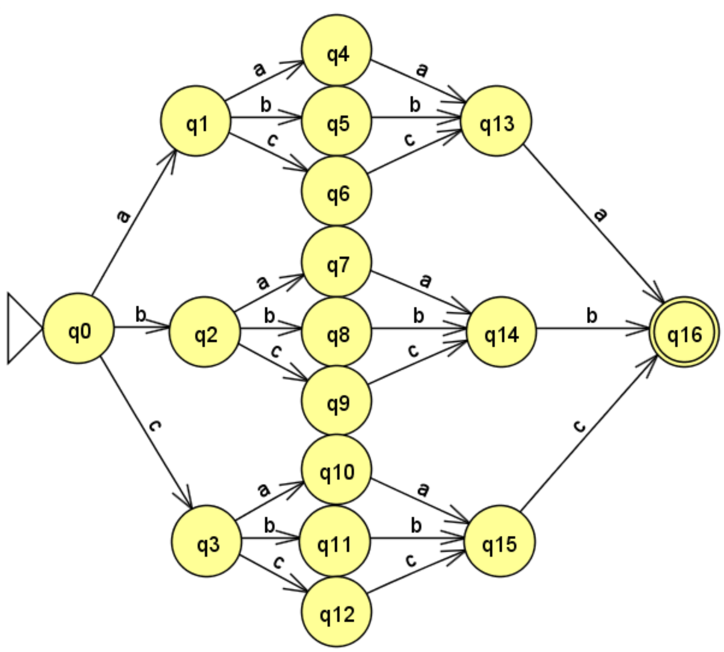
Solution:



The string w = 00100110111 is accepted via
$q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_4 \rightarrow q_5 \rightarrow q_3 \rightarrow q_4 \rightarrow q_5 \rightarrow q_6$

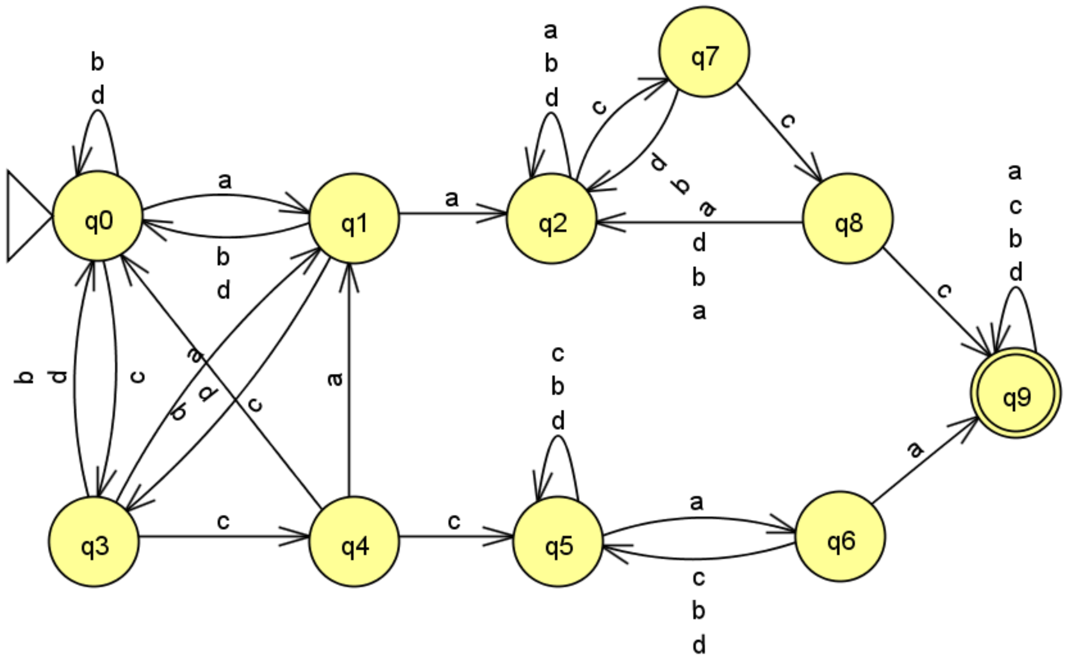
21. The set of all strings that are palindromes of length 4. The alphabet is {a, b, c}.

Solution:



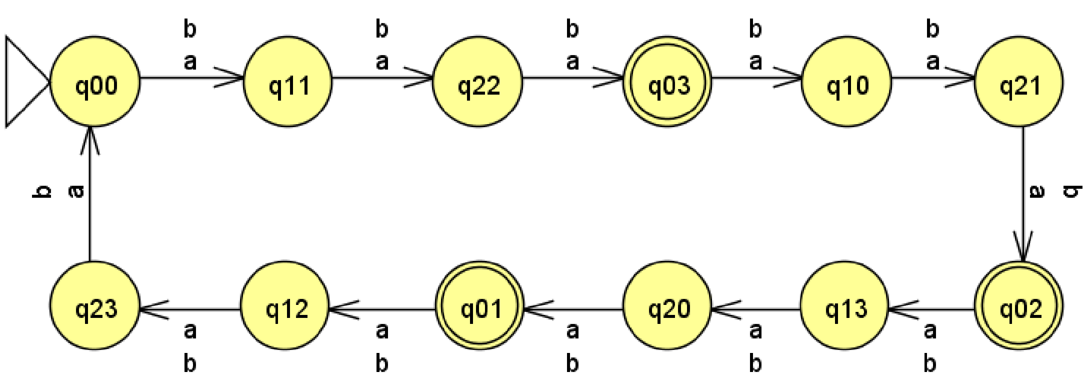
22. A *run* in a string is a sub-string made up of a single symbol. For example, *aabbbcd* has a run of *a* s of length 2 and a run of *b* s of length 3. The automaton must accept all strings over the alphabet {*a*, *b*, *c*, *d*} with at least one run of *a* s of length 2 and a run of *c* s of length 3.

Solution: Note that in acceptable strings, either the run of *a* s comes first (q_0-q_1-q_2) and then the run of *c* s (q_2-q_7-q_8-q_9) or the run of *c* s first (q_0-q_3-q_4-q_5) and then the run of *a* s (q_5-q_6-q_9). The DFA must accept both types of inputs.



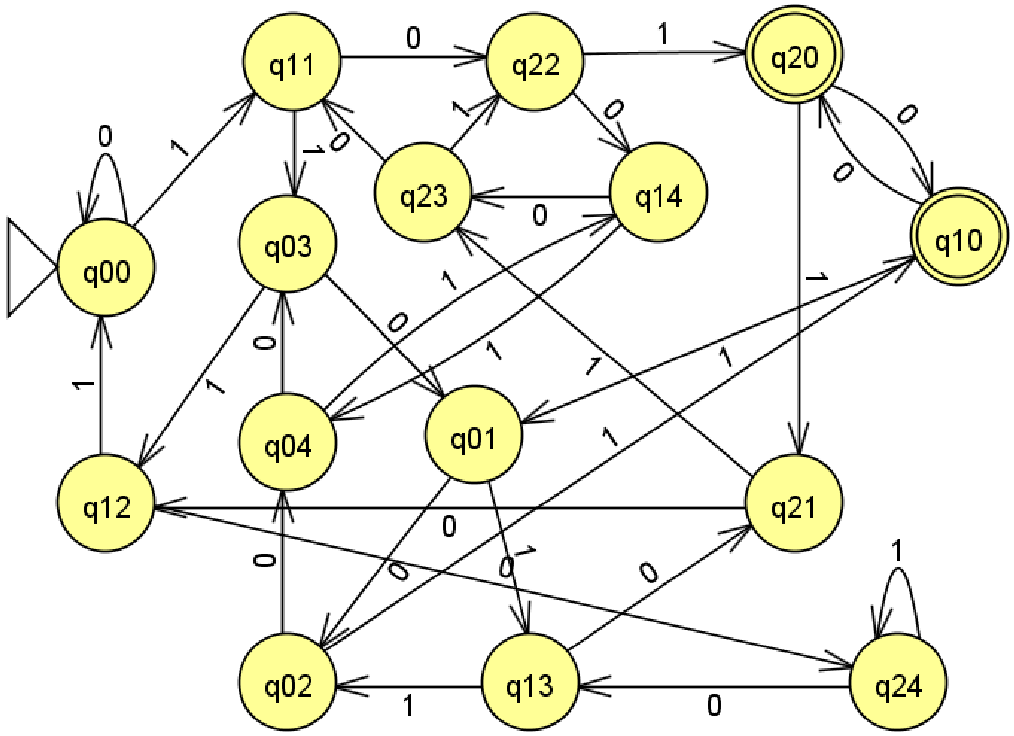
23. Strings over {*a*, *b*} such that the length of the string is a multiple of 3 but not a multiple of 4.

Solution: Note that states are labeled with two-digits, the first digit denoting the value of the remainder when divided by 3 (i.e., mod 3) and the second digit denoting the value of the remainder when divided by 4 (i.e., mod 4).



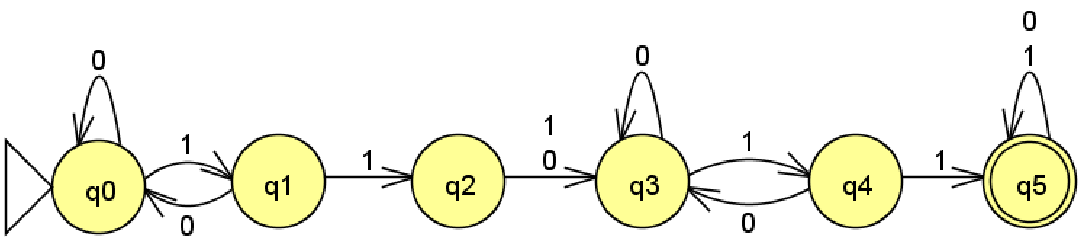
24. Binary strings representing positive integers divisible by 5 but not divisible by 3.

Solution:



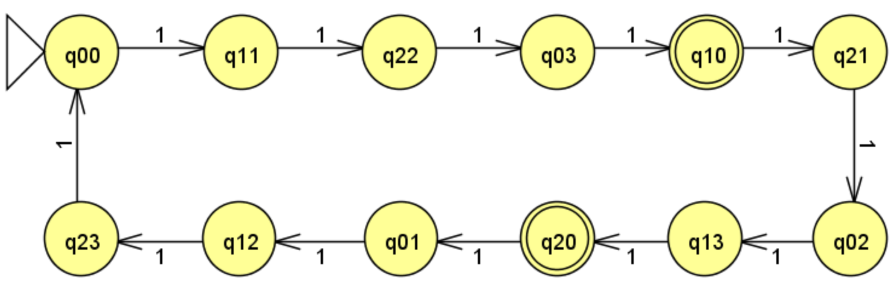
25. Binary strings with at least two occurrences of at least two consecutive 1 s, the two occurrences not being adjacent (i.e., 011011 and 011111 are acceptable but 01111 is not).

Solution:



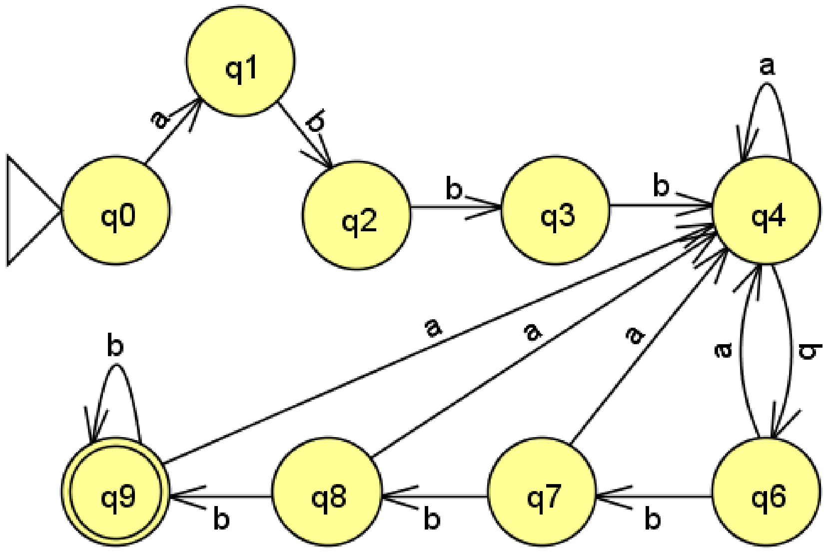
26. Consider the unary number system with the alphabet {1} where a number *n* is represented by a string of *n* 1 s, for example, 4 is 1111 and 7 is 1111111. Construct a finite automaton that accepts all unary numbers that are divisible by 4 but not divisible by 3.

Solution: Note that states are labeled with two-digits, the first digit denoting the value of the remainder when divided by 3 (i.e., mod 3) and the second digit denoting the value of the remainder when divided by 4 (i.e., mod 4).



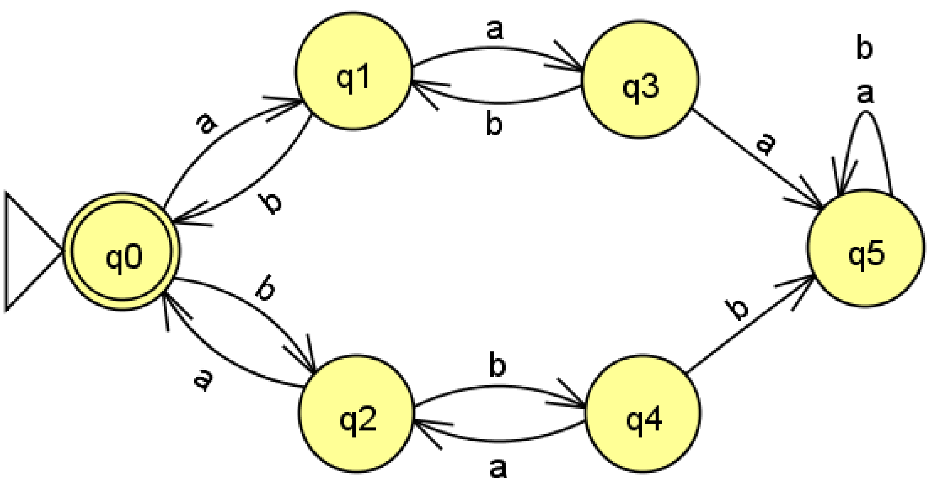
27. Strings of the form ab^3wb^4 where the alphabet is {*a*, *b*} and *w* is any string over the same alphabet.

Solution:



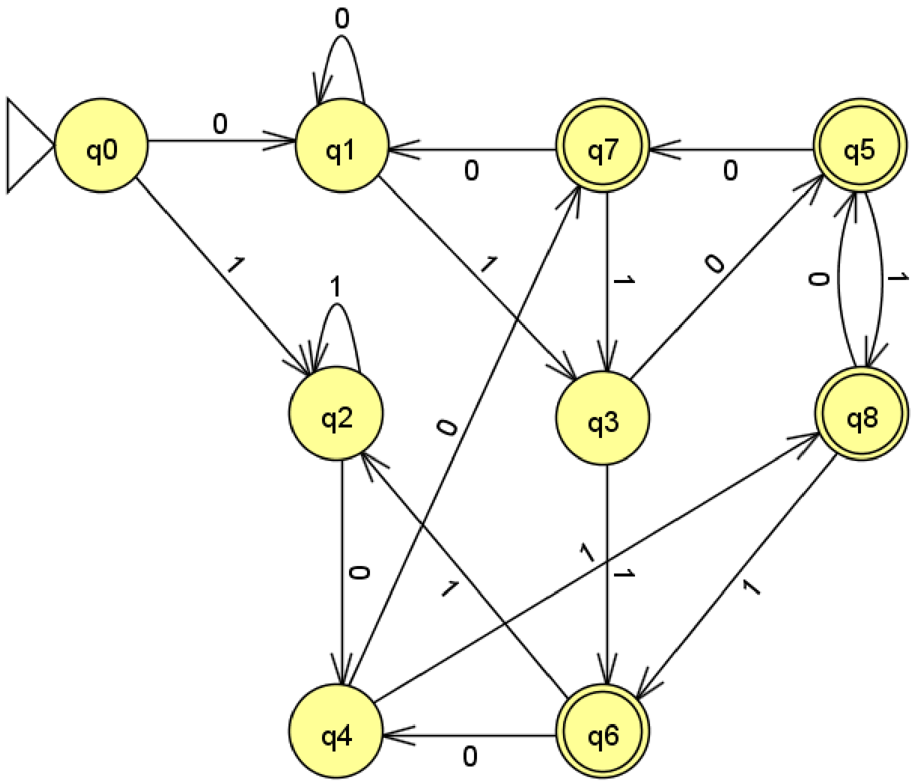
28. All strings over {*a*, *b*} where the total number of *a* s is equal to the total number of *b* s. Further, at any point in the string, the number of *b* s seen thus far cannot exceed the number of *a* s seen by more than 2; similarly, the number of *a* s seen thus far cannot exceed the number of *b* s seen by more than 2. For example, *aababb* is acceptable but *aabaabbb* is to be rejected.

Solution: See also Example 2.12 and Fig~~.~~ure ~~-~~2.13.



29. All binary strings where the third and second symbols from the end are not the same.

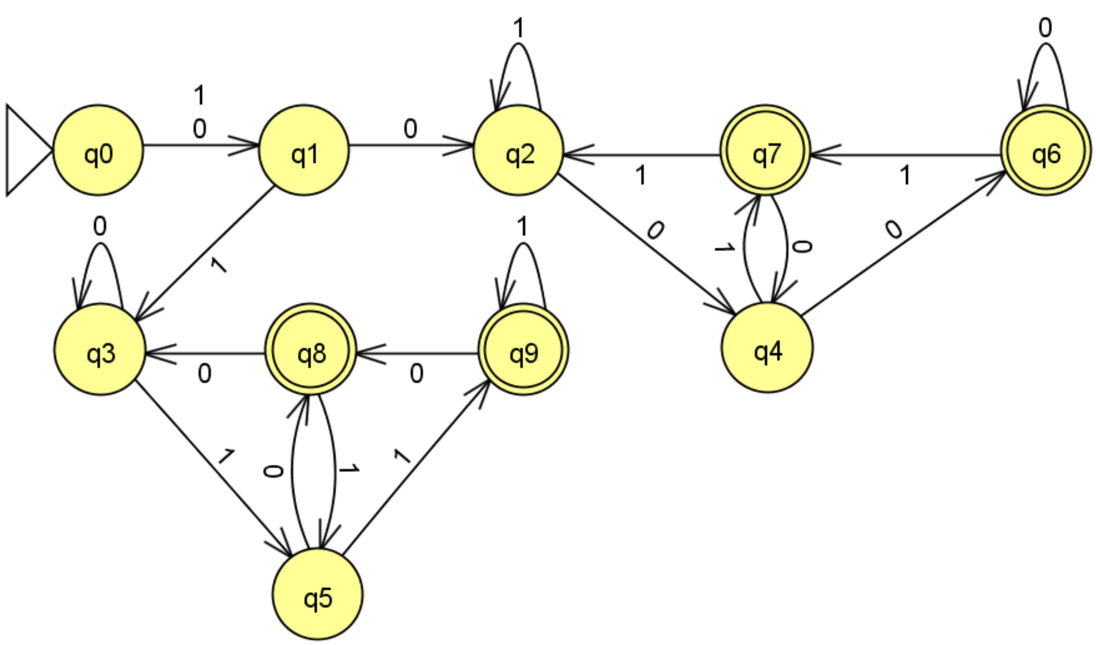
Solution: Note that it is rather difficult to design this DFA or to understand how it works. Observe that q_1 and q_4 are only reached on processing a 0; q_2 and q_3 only on processing a 1. Similarly, q_3 and q_4 are reached on processing a potential second symbol from the end; q_1 and q_2 are reached on processing a potential third symbol from the end.



Load the file provided in JFLAP to try out various input strings. The DFA itself was obtained by first constructing a nondeterministic finite automaton (see Chapter 3) and then automatically converting it to a DFA.

30. All binary strings where the second symbol is the same as the penultimate (i.e., the second from the end) symbol.

Solution: Note: As in Exercise 29, this DFA was obtained by automatically converting from a nondeterministic finite automaton.

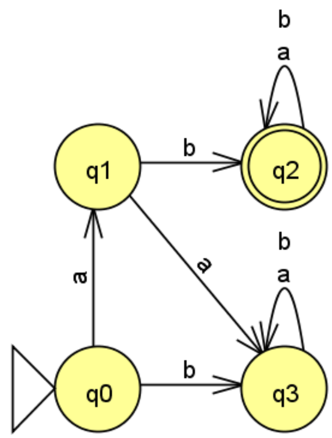


B. For the following problems, a transition table is specified:

31. Describe the language (i.e., set of all strings) accepted by the following automaton:

State	Input = ***a***	Input = ***b***
$\rightarrow q_0$	q_1	q_3
q_1	q_3	q_2
$^*q_2$	q_2	q_2
q_3	q_3	q_3

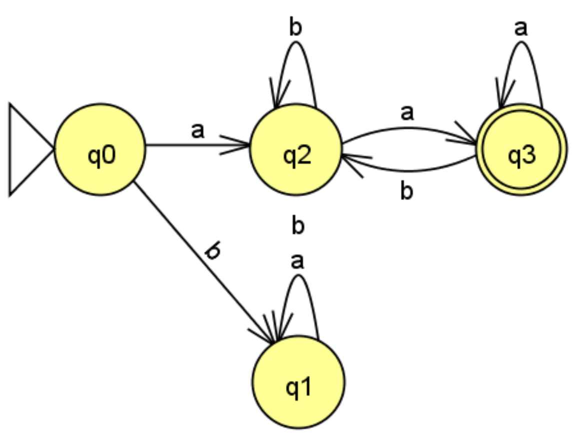
Solution: The transition table is converted to a diagram for convenience. State q_3 is a reject state. Therefore the language accepted is the set of strings beginning with *ab*.



32. Describe the language (i.e., set of all strings) accepted by the following automaton:

State	Input = ***a***	Input = ***b***
$\rightarrow q_0$	q_2	q_1
q_1	q_1	q_1
q_2	q_3	q_2
$^*q_3$	q_3	q_2

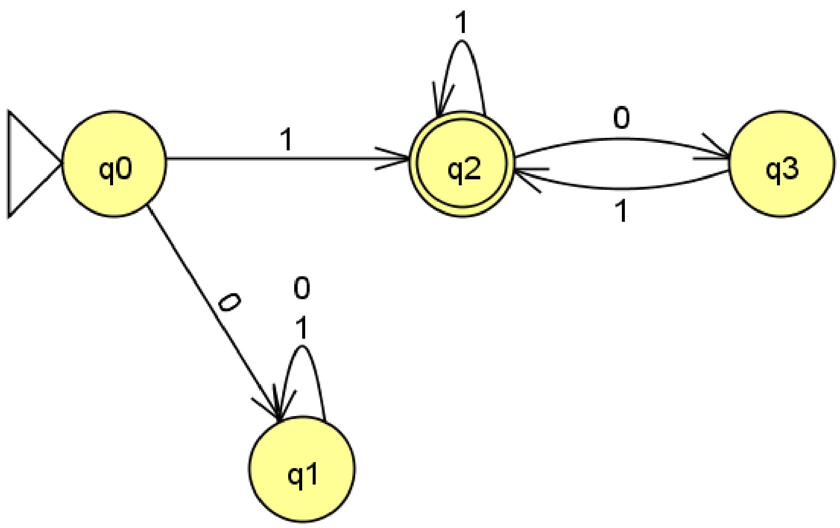
Solution: State q_1 is a reject state. The DFA accepts the set of all strings that begin with *a* and end with *a*.



C. Debug and fix the following defective finite automata:

33. The automaton in Figure 2.6, Example 2.5.

Solution: The defect with the automaton shown in Fig. 2.6 lies in the meaning of the state q_0. How is it both the start state and the state for even numbers? This is why that automaton wrongly rejects the strings 1001. The corrected finite automaton is as follows (with q_3 as the new state for even numbers):



34. The automaton shown in Figure 2.16 for accepting all binary strings with alternating 0 s and 1 s:

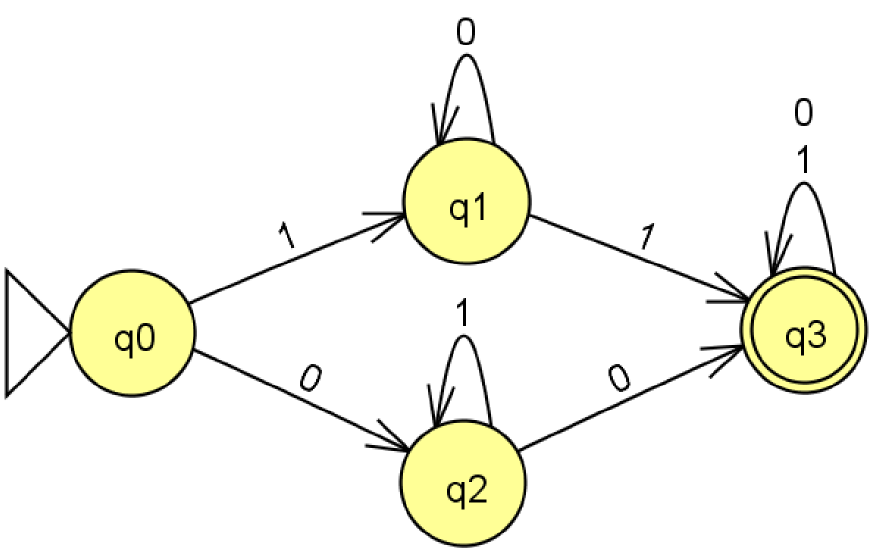
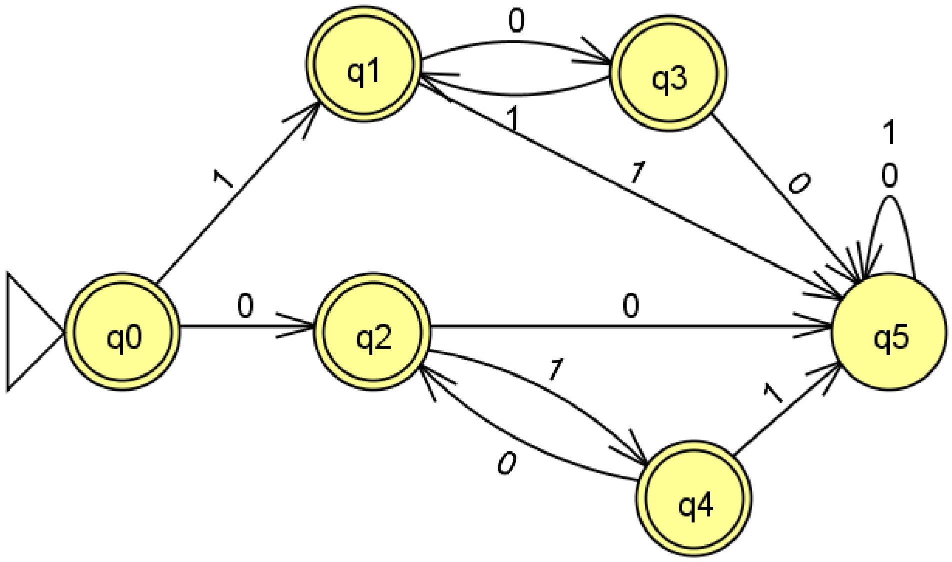
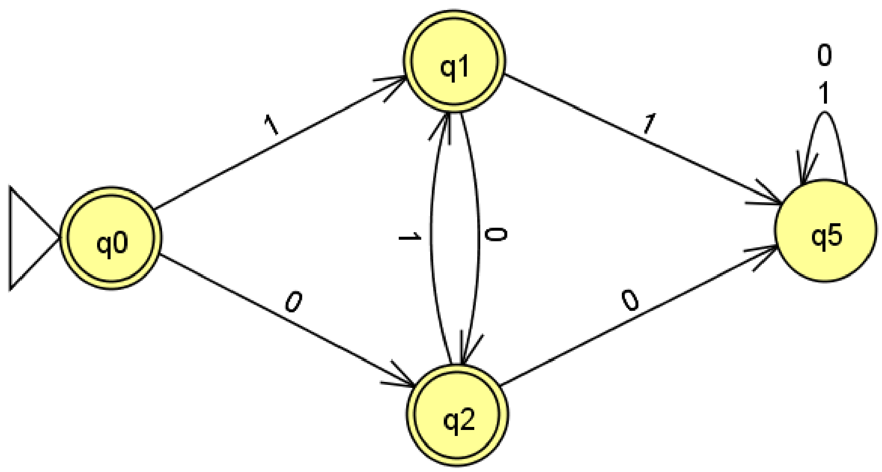


Figure 2.16

Solution: There can never be a loop in this automaton. Every symbol is significant. The automaton cannot simple loop back to the same state. For example, this automaton accepts 01110 although it does not have alternating 0 s and 1 s. The corrected finite automaton is as follows:



This automaton goes to the reject state q5 the moment either 00 or 11 occurs in the input. In fact, it can be further simplified to obtain a smaller automaton as follows:



35. The automaton shown in Figure 2.17 for accepting strings over {*a*, *b*} that either begin and end with an *a* and contain an even number of *b* s in between or begin and end with a *b* and contain an even number of *a* s in the middle:

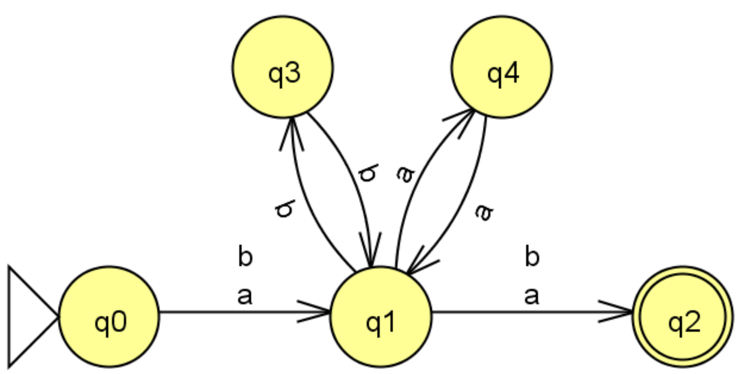
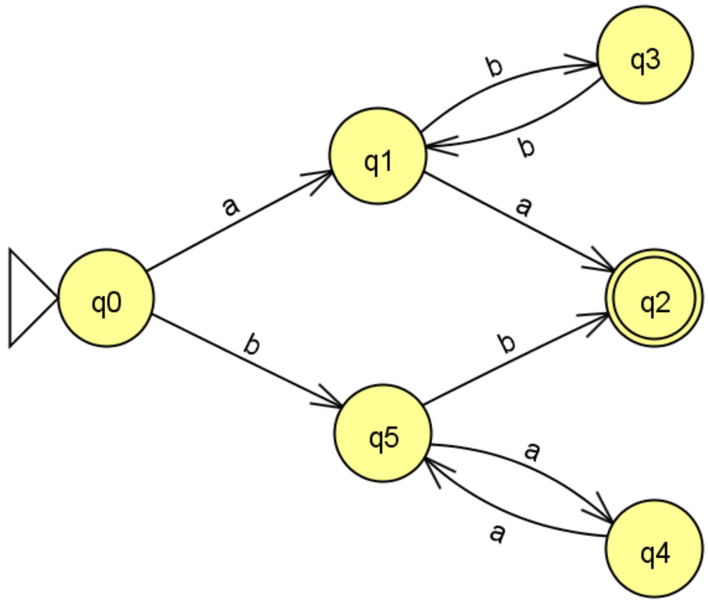


Figure 2.17

Solution: The automaton needs two separate branches that cannot be combined. The corrected one is as follows:



36. The automaton shown in Figure 2.18 for all binary strings representing positive numbers divisible by 7:

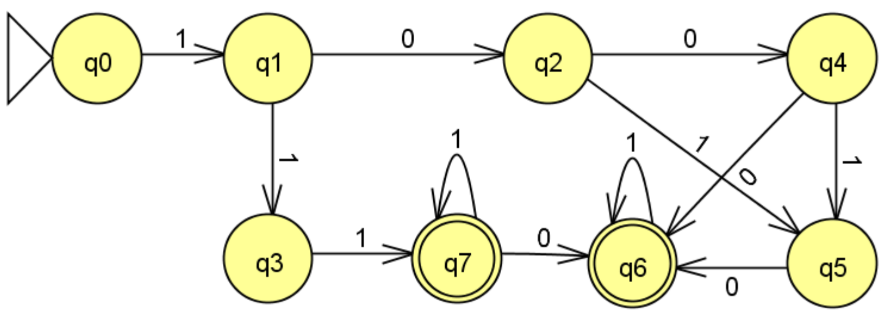
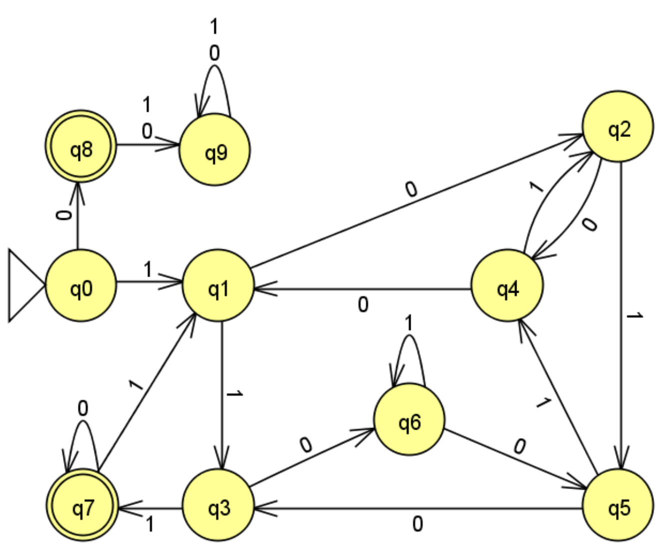


Figure 2.18

Solution: The correct automaton is as follows:



D. Explain why we cannot construct a finite automaton for the following:

37. In coding theory, the *weight* of a binary string is the number of 1 s in the string. Consider the set of all binary strings whose weight is less than half the length of the string.

    Answer: Since strings can be arbitrarily long and since 0 s and 1 s can appear in any order, the DFA will have to count the number of 1 s and match the count to the length of the string (i.e., the count of the total number of symbols in the input). Finite automata are unable to count with their finite number of states and no other memory unit. A new state is needed for each new value of the weight. Therefore it is not possible to construct a finite automaton for this example. (See Chapter 6 for a way of proving this formally.)

38. Unequal numbers of *a* s and *b* s, in any order, for the alphabet {*a*, *b*}.

    Answer: Since strings can be arbitrarily long and since *a* s and *b* s can appear in any order, the DFA will have to count the number of *a* s and match it to the count of the number of *b* s. Finite automata are unable to count with their finite number of states and no other memory unit. Therefore it is not possible to construct a finite automaton for this example. (See Chapter 6 for a way of proving this formally.)

39. Palindromes (i.e., strings that read the same forward or backward, for example, *malayalam*) of arbitrary length.

    Answer: Once again, since the palindromes can be arbitrarily long, the automaton will need arbitrary amounts of memory to remember the first half of the palindromes so that it can be matched to the second half. It is unable to do this with its finite number of states and no other memory unit. Therefore it is not possible to construct a finite automaton for this example. (See Chapter 6 for a way of proving this formally.)

40. Some number of *a* s (however large) followed by double the number of *b* s.

    Answer: The finite automaton has to count the number of *a* s, remember the count and double it to match to the number of *b* s. Finite automata are unable to count with their finite number of states and no other memory unit. A new state is needed for every possible value of the count. Therefore it is not possible to construct a finite automaton for this example. (See Chapter 6 for a way of proving this formally.)

41. Formulate a set of *mantras* to decide whether a finite automaton can be constructed to accept a given set of strings. *Hint:* Generalize from exercises 37-40.

    Answer: Possible *mantras*:

1. Is there a limit on the length of strings to be accepted?
2. Is there a limit on the size of what needs to be remembered by the automaton?
3. Does it involve counting?
4. Does it involve two or more parts of the input string, each part being of unlimited length?
5. Is it sufficient to remember only one or a few attributes of the input symbol? Is there a finite number of possible values for the attributes (e.g., even or odd, divisible by 3, etc.)?
6. Is the problem similar to one of the problems for which we already know that it is not possible to construct a finite automaton?