Lecture 5

Some More Examples on Designing Finite Automata

Lecture-5	PO 1 & PSO 1

5.1 Designing Finite Automata

Example 5.1: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ of length 1 i.e. $\mid w \mid = 1$ is shown in figure 5.1 where,
DFA can be described as $(\{q_0, q_1, D\}, \{a, b\}, \delta, q_0, \{q_1\})$. L ={All the strings of length 1}
$L = \{a, b\}$

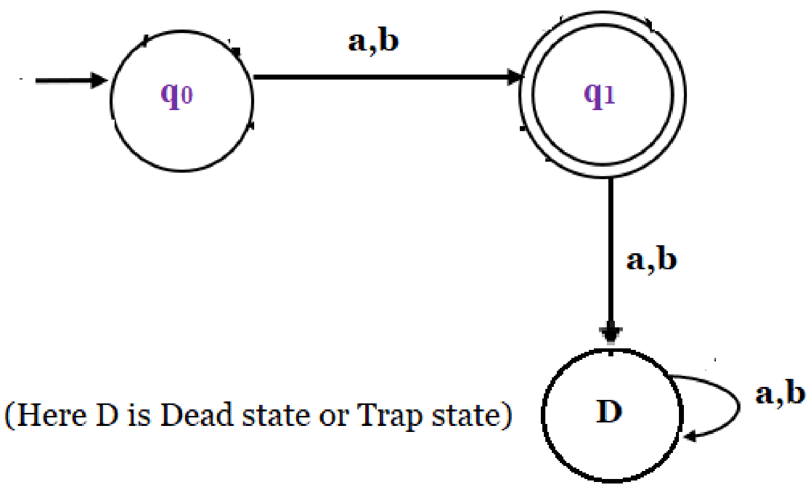

Figure 5.1: DFA that accepts set of strings of length 1.

The transition table is:

States	a	b
$\rightarrow q_0$	q_1	q_1
q_1*	D	D
D	D	D

The transition function is:

$\delta(q_0, a) = (q_1)$ $\quad \delta(q_0, b) = (q_1)$

$\delta(q_1, a) = (D)$ $\quad \delta(q_1, b) = (D)$

$\delta(D, a) = (D)$ $\quad \delta(D, b) = (D)$

Explanation:

1. We have to create DFA which will accept string of unit length on alphabet $\{a, b\}$.
2. So first thing about creating DFA which will accept of 1 length string, that is pretty simple.
3. You just take 2 states such that 1 length string can be accepted. The second state will be final state.
4. But what if we came up with a string of length 2, it should not be accepted in our DFA, right ?
5. So we have to attach one more state to take care of string of length greater than 1 called 'D'.
6. State 'D' is called dead state.

Example 5.2: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ of length 2 i.e. $\mid w \mid = 2$ is shown in figure 5.2 where,
DFA can be described as $(\{q_0, q_1, q_2, D\}, \{a, b\}, \delta, q_0, \{q_2\})$.
$L = \{\text{All the strings of length } 2\}$
$L = \{aa, ab, ba, bb\}$

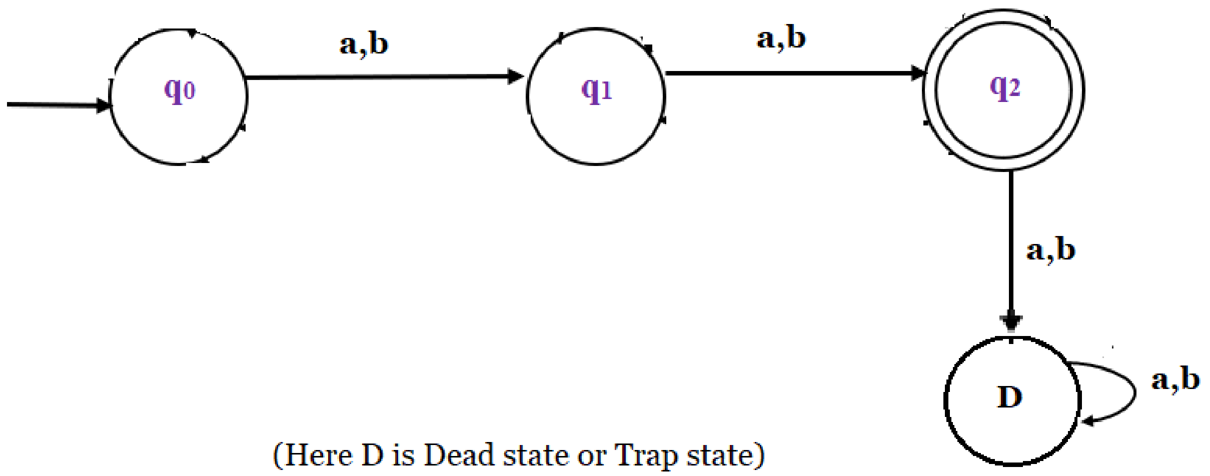

Figure 5.2: DFA that accepts set of strings of length 2.

The transition table is:

States	a	b
$\rightarrow q_0$	q_1	q_1
q_1	q_2	q_2
q_2*	D	D
D	D	D

The transition function is:

$\delta(q_0, a) = (q_1)$ $\delta(q_0, b) = (q_1)$
$\delta(q_1, a) = (q_2)$ $\delta(q_1, b) = (q_2)$
$\delta(q_2, a) = (D)$ $\delta(q_2, b) = (D)$
$\delta(D, a) = (D)$ $\delta(D, b) = (D)$

Explanation:

1. We have to create DFA which will accept string of length 2 on alphabet $\{a, b\}$.
2. To take 2 length string we need 3 states.
3. The third state will be final state.
4. But what if we came up with a string of length 3, it should not be accepted in our DFA, right ?
5. So we have to attach one more state to take care of string length greater than 2 called 'D'.
6. State 'D' is called dead state.

Here we can conclude that, a finite automaton that accepts the strings of length exactly n can be constructed with $n + 2$ number of states.

Example 5.3: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ of length at most 2 i.e. $\mid w \mid \leq 2$ is shown in figure 5.3 where,
DFA can be described as $(\{q_0, q_1, q_2, D\}, \{a, b\}, \delta, q_0, \{q_0, q_1, q_2\})$.
$L = \{$All the strings of length at most 2$\}$
$L = \{\epsilon$, a, b, aa, ab, ba, bb$\}$
The transition table is:

States	a	b
$\rightarrow q_0*$	q_1	q_1
q_1*	q_2	q_2
q_2*	D	D
D	D	D

The transition function is:

$\delta(q_0, a) = (q_1)$ $\delta(q_0, b) = (q_1)$
$\delta(q_1, a) = (q_2)$ $\delta(q_1, b) = (q_2)$
$\delta(q_2, a) = (D)$ $\delta(q_2, b) = (D)$
$\delta(D, a) = (D)$ $\delta(D, b) = (D)$

Explanation:

1. We have to create DFA which will accept string of length at most 2 i.e.

of length 0, 1 and 2 on alphabet $\{a, b\}$. Hence, the language content is $= \{\epsilon, a, b, ab, ba, bb, aa\}$.

2. To accept 2 length string we need 3 states.
3. The first state accepts strings of length 0, the second state accepts the strings of length 1 and the third state accepts the strings of length 2. So all the three states will be the final states.
4. But what if we came up with a string of length 3, it should not be accepted in our DFA, right ?
5. So we have to attach one more state to take care of string length greater than 2 called 'D'.
6. State 'D' is called dead state.

Here we can conclude that, a finite automaton that accepts the strings of length at most n can be constructed with $n + 2$ number of states.

Example 5.4: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ of length at least 2 i.e. $\mid w \mid \leq 2$ is shown in figure 5.4 where,
DFA can be described as $(\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$.
$L = \{\text{All the strings of length at least 2}\}$
$L = \{aa, ab, ba, bb, aaa, \ldots, bbb, \ldots\}$

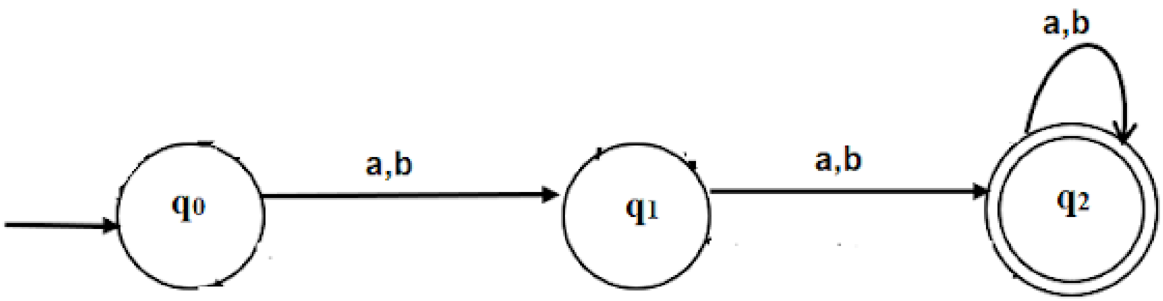

Figure 5.4: DFA that accepts set of strings of length at least 2.

The transition table is:

States	a	b
$\rightarrow q_0$	q_1	q_1
q_1	q_2	q_2
q_2*	q_2	q_2

The transition function is:
$\delta(q_0, a) = (q_1)$ $\delta(q_0, b) = (q_1)$
$\delta(q_1, a) = (q_2)$ $\delta(q_1, b) = (q_2)$
$\delta(q_2, a) = (q_2)$ $\delta(q_2, b) = (q_2)$

Explanation:

1. We have to create DFA which will accept string of length at least 2 i.e. of length ≥ 2 on alphabet $\{a, b\}$.
2. To take 2 length string we need 3 states.
3. The third state will be the final state.
4. If input comes over the third state q_2 then it will go to q_2 itself to accept strings greater than 2.
5. Now if string ¡ 2 will not reach final state, so will not be accepted.

Here we can conclude that, a finite automaton that accepts the strings of length at least n can be constructed with $n + 1$ number of states.

Example 5.5: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ of even length i.e. $\mid w \mid mod 2 = 0$ is shown in figure 5.5 where,
DFA can be described as $(\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_0\})$.
$L = \{\text{All the strings of even length}\}$
$L = \{\epsilon, aa, ab, ba, bb, aaab, \ldots, abbbba, \ldots\}$

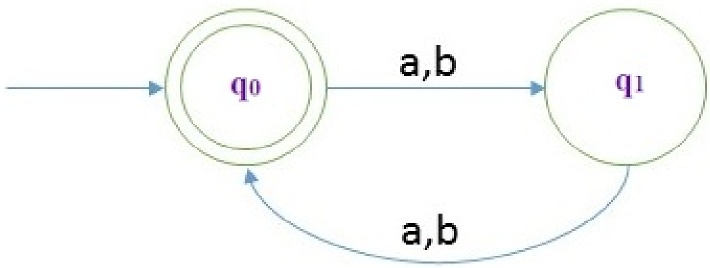

Figure 5.5: DFA that accepts set of strings of even length.

The transition table is:

States	a	b
$\rightarrow q_0*$	q_1	q_1
q_1	q_0	q_0

The transition function is:
$\delta(q_0, a) = (q_1)$ $\delta(q_0, b) = (q_1)$
$\delta(q_1, a) = (q_0)$ $\delta(q_1, b) = (q_0)$

Example 5.6: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ of length divisible by 3 i.e. $\mid w \mid \bmod 3 = 0$ is shown in figure 5.6 where,
DFA can be described as $(\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_0\})$.
$L = \{\text{All the strings of length divisible by } 3\}$
$L = \{\epsilon, aaa, aba, baa, bbb, \ldots, abbbba, \ldots\}$

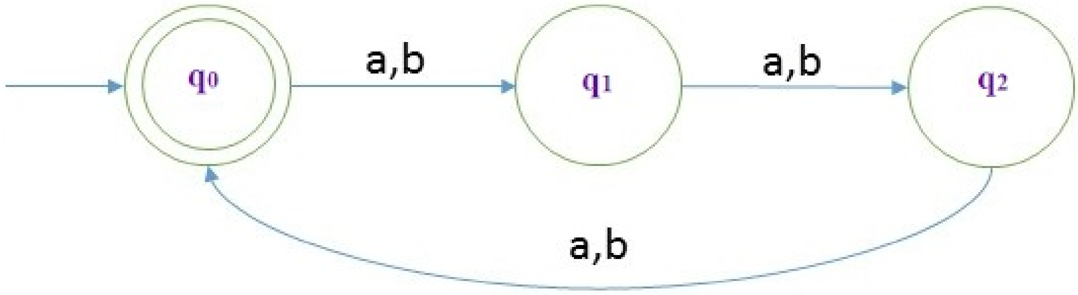

Figure 5.6: DFA that accepts set of strings of length divisible by 3 i.e. $| w | \bmod 3 = 0$.

The transition table is:

States	a	b
$\rightarrow q_0*$	q_1	q_1
q_1	q_2	q_2
q_2	q_0	q_0

The transition function is:

$\delta(q_0, a) = (q_1)$ $\quad \delta(q_0, b) = (q_1)$

$\delta(q_1, a) = (q_2)$ $\quad \delta(q_1, b) = (q_2)$

$\delta(q_2, a) = (q_0)$ $\quad \delta(q_2, b) = (q_0)$

We can design the automata for the languages $| w | \bmod 3 = 1$ and $| w | \bmod 3 = 2$ by only changing the final state in the automaton designed for $| w | \bmod 3 = 0$ and shown in Figure 5.7 and Figure 5.8 respectively.

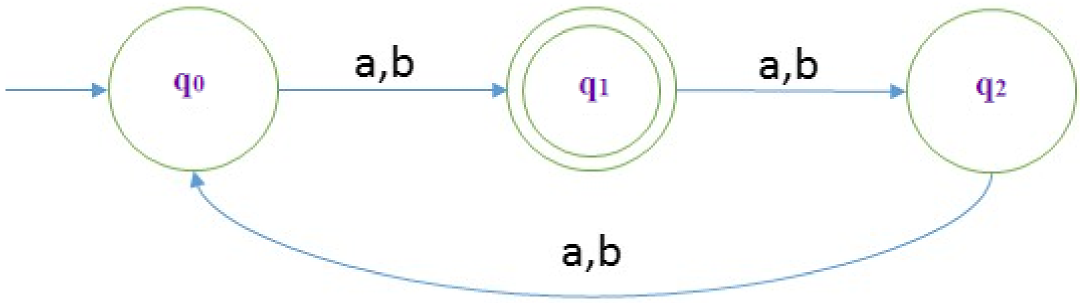

Figure 5.7: DFA that accepts set of strings where $| w | \bmod 3 = 1$.

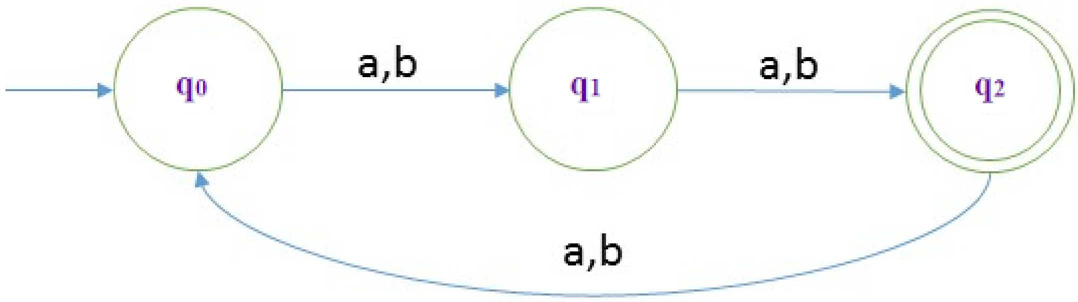

Figure 5.8: DFA that accepts set of strings where $| w | \bmod 3 = 2$.

Here we can conclude that, a finite automaton that accepts the strings of length divisible by n can be constructed with n number of states.

Example 5.7: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ starts with a 'a', is shown in figure 5.9 where,
DFA can be described as $(\{q_0, q_1, D\}, \{a, b\}, \delta, q_0, \{q_1\})$.
$L = \{$All the strings that start with a 'a'$\}$
$L = \{a, aa, ab, aaa, aab, \ldots\}$

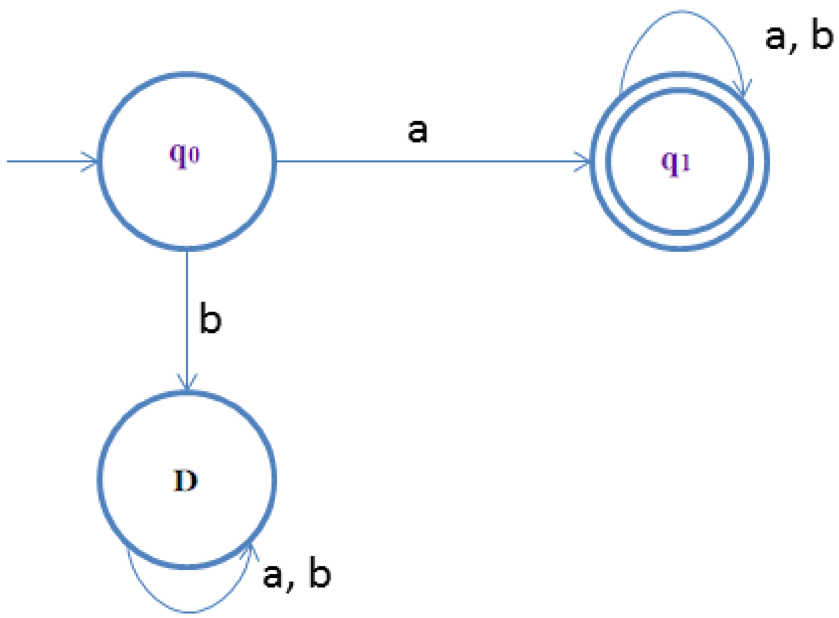

Figure 5.9: DFA that accepts set of strings that start with a 'a'.

The transition table is:

States	a	b
$\rightarrow q_0$	q_1	D
q_1*	q_1	q_1
D	D	D

The transition function is:

$\delta(q_0, a) = (q_1)$ $\quad\delta(q_0, b) = (D)$
$\delta(q_1, a) = (q_1)$ $\quad\delta(q_1, b) = (q_1)$
$\delta(D, a) = (D)$ $\quad\delta(D, b) = (D)$

Explanation:

1. We have to create DFA that accepts set of strings that start with a 'a'.
2. We know that first input must be 'a' and from start state on 'a' we should go to final state, so two states are confirmed: start state and final state.
3. And if the first input is something else than 'a' then string should not be accepted. And we don't care whatever comes on state D as state D now

will work as dead state.

4. Once we reach final state we accept everything that is why we have a loop.

Example 5.8: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ which contains 'a', is shown in figure 5.10 where,
DFA can be described as $(\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_1\})$.
$L = \{$All the strings which contains 'a'$\}$
$L = \{a, aa, ab, ba, aaa, aab, abb, baa, bab, bba, \ldots\}$

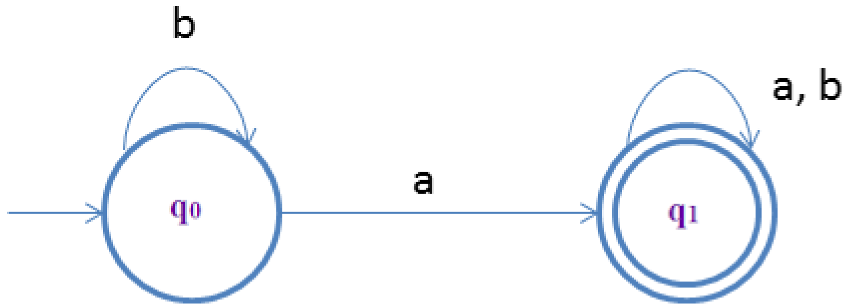

Figure 5.10: DFA that accepts set of strings which contains 'a'.

The transition table is:

States	a	b
$\rightarrow q_0$	q_1	q_0
q_1*	q_1	q_1

The transition function is:
$\delta(q_0, a) = (q_1)$ $\quad \delta(q_0, b) = (q_0)$
$\delta(q_1, a) = (q_1)$ $\quad \delta(q_1, b) = (q_1)$

Explanation:

1. We have to create DFA that accepts set of strings which contains 'a'.
2. We know that first input must be 'a' and from start state on 'a' we should go to final state, so two states are confirmed: start state and final state.
3. And if the first input is 'b' then we will loop it to the start state itself.
4. Once we reach final state we accept everything that is why we have a loop.

Example 5.9: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ which ends with 'a', is shown in figure 5.11 where,
DFA can be described as $(\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_1\})$.
$L = \{$All the strings which ends with 'a'$\}$
$L = \{aa, aba, abba, baba, \ldots, aaabbba, \ldots\}$

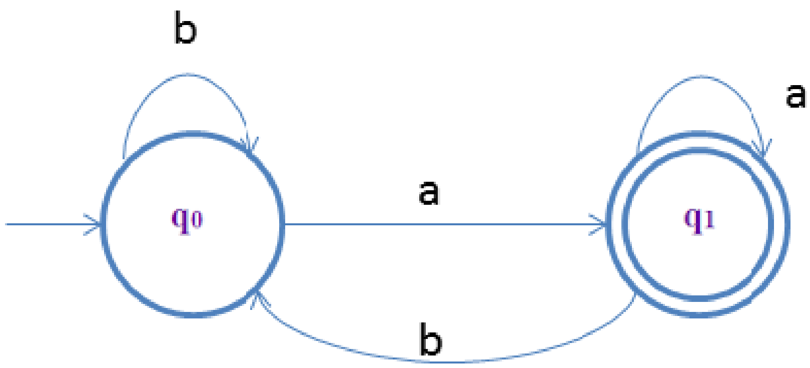

Figure 5.11: DFA that accepts set of strings which ends with 'a'.

The transition table is:

States	a	b
$\rightarrow q_0$	q_1	q_0
q_1*	q_1	q_0

The transition function is:

$\delta(q_0, a) = (q_1)$ $\quad$ $\delta(q_0, b) = (q_0)$

$\delta(q_1, a) = (q_1)$ $\quad$ $\delta(q_1, b) = (q_0)$

Explanation:

1. We have to create DFA that accepts set of strings which ends with 'a'.
2. We will first start accepting 'b' on start state itself and then on 'a' we will go to final state.
3. And on final state we do not care for 'a''s so we have to put a self-loop.
4. On final state if 'b' comes then we will redirect it start state so that any string which does not end with 'a' should not get accepted.

Example 5.10: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ starts with "ab", is shown in figure 5.12 where,
DFA can be described as $(\{q_0, q_1, q_2, D\}, \{a, b\}, \delta, q_0, \{q_2\})$.
$L = \{$All the strings that start with "ab"$\}$
$L = \{ab, aba, abba, \ldots, abaabbb, \ldots\}$
The transition table is:

States	a	b
$\rightarrow q_0$	q_1	D
q_1	D	q_2
q_2*	q_2	q_2
D	D	D

The transition function is:

$\delta(q_0, a) = (q_1)$ $\quad$ $\delta(q_0, b) = (D)$

$\delta(q_1, a) = (D)$ $\quad$ $\delta(q_1, b) = (q_2)$

$\delta(q_2, a) = (q_2)$ $\quad$ $\delta(q_2, b) = (q_2)$

$\delta(D, a) = (D)$ $\quad$ $\delta(D, b) = (D)$

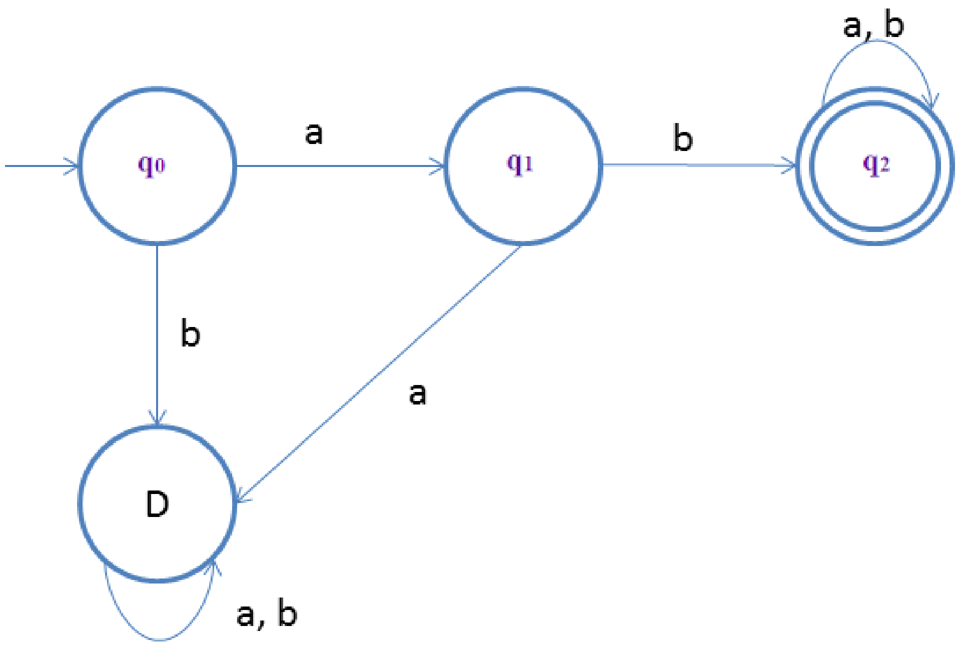

Figure 5.12: DFA that accepts set of strings that starts with “ab”.

Explanation:

1. We have to create DFA that accepts set of strings which starts with \\ab”.
2. First we will make DFA for accepting the smallest string that is \\ab” and after that whatever comes we don’t care as the condition is satisfied.
3. In DFA we have to take care of all the input alphabets at every state.
4. So we have to take care of input symbol ‘b' on state q_0, that is we have to reject it cause first ‘a' should come then ‘b'.
5. So we will make one more state D to take care of ‘b' from state q_0.
6. On state q_1 if ‘a' comes then we have to reject it, so we will direct this input to state D.
7. On state D for input ‘a' and ‘b' we do not care so we will make one self-loop.

Example 5.11: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ which contains “ab” as a substring, is shown in figure 5.13 where,
DFA can be described as $(\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$.
$L = \{$All the strings which contains “ab” as a substring $\}$
$L = \{ab, aba, abba, abaabbb, bbabaabb, \ldots\}$

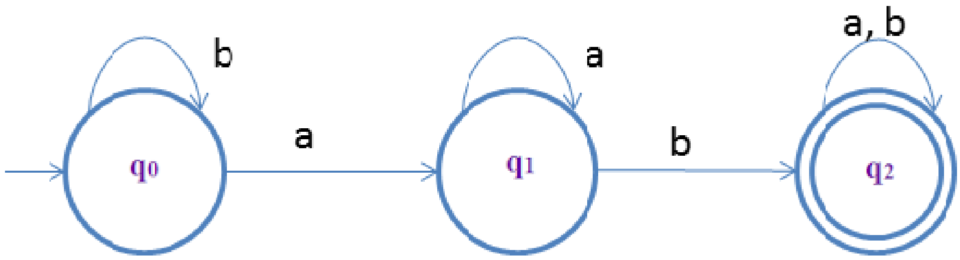

Figure 5.13: DFA that accepts set of strings which contains "ab" as a substring.

The transition table is:

States	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2*	q_2	q_2

The transition function is:

$\delta(q_0, a) = (q_1)$ $\quad \delta(q_0, b) = (q_0)$

$\delta(q_1, a) = (q_1)$ $\quad \delta(q_1, b) = (q_2)$

$\delta(q_2, a) = (q_2)$ $\quad \delta(q_1, b) = (q_2)$

Explanation:

1. We have to create DFA that accepts set of strings which contains "ab" as a substring.
2. First we will make DFA for accepting the smallest string that is "ab" and after that whatever comes at final state we don't care as the condition is satisfied.
3. In DFA we have to take care of all the input alphabets at every state.
4. So we have to take care of input symbol 'b' on state q_0, that is we made self-loop on start state.
5. On state q_1 if 'a' comes then we will accept it as repetition of 'a' and this 'a' will not ruin anything. Because, we want "ab" in the entire string to be present anywhere.
6. On state q_2 for input 'a' and 'b' we do not care so we will make one self-loop for 'a', 'b'.

Example 5.12: Construction of a DFA, that accepts set of strings over $\Sigma = \{a, b\}$ which ends with "ab", is shown in figure 5.14 where,
DFA can be described as $(\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$.
$L = \{$All the strings which ends with "ab"$\}$
$L = \{ab, abab, abbab, abaabbab, bbabaabab, \ldots\}$

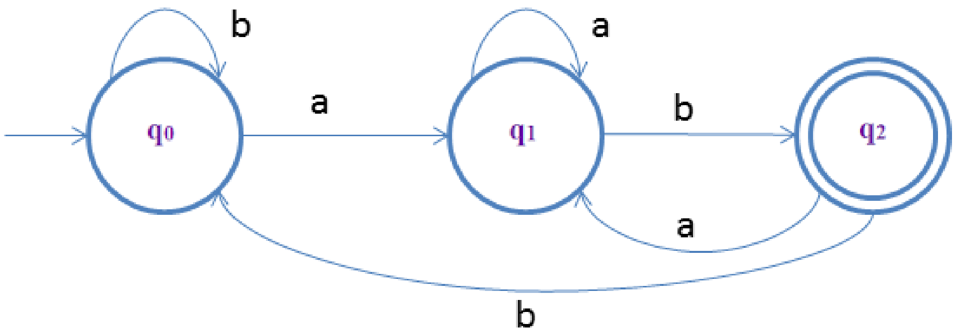

Figure 5.14: DFA that accepts set of strings which ends with “ab”.

The transition table is:

States	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2*	q_1	q_0

The transition function is:

$\delta(q_0, a) = (q_1)$ $\quad$ $\delta(q_0, b) =$

$\delta(q_1, a) = (q_1)$ $\quad$ $\delta(q_1, b) =$

$\delta(q_2, a) = (q_1)$ $\quad$ $\delta(q_2, b) =$

Explanation:

1. We have to create DFA that accepts set of strings which ends with “ab”.
2. First we will make DFA for accepting the smallest string that is “ab”.
3. In DFA we have to take care of all the input alphabets at every state.
4. So we have to take care of input symbol ‘b’ on state q_0, that is we made self-loop on start state.
5. On state q_1 if ‘a’ comes then we will accept it as repetition of ‘a’ and that ‘a’ will not ruin anything. Because, we want “ab” in the end.
6. On State q_2 if ‘b’ comes then that will be a problem as we only want “ab” in the end not “bb”, so we will direct ‘b’ to state q_0.
7. If ‘a’ comes on state q_2 then we will direct it to state q_1 and if one ‘b’ comes then we will be good by getting “ab” in the end.

Example 5.13: Construction of a DFA, that accepts set of strings over $\Sigma = \{0, 1\}$ which when interpreted as binary number is divisible by ‘2’, is shown in figure 5.15. For example, 110 in binary is equivalent to 6 in decimal and 6 is divisible by 2. For this automaton,

DFA can be described as $(\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_0\})$.

$L = \{$All the strings which interpreted as binary number is divisible by ‘2’$\}$

$L = \{\epsilon, 0, 00, 10, 100, 110, \ldots\}$

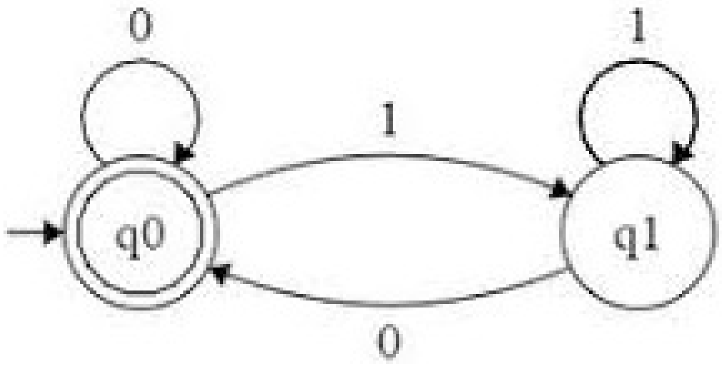

Figure 5.15: DFA that accepts set of strings which when interpreted as binary number is divisible by '2'.

The transition table is:

States	0	1
$\rightarrow q_0*$	q_0	q_1
q_1	q_0	q_1

The transition function is:
$\delta(q_0, 0) = (q_0)$ $\quad$ $\delta(q_0, 1) = (q_1)$
$\delta(q_1, 0) = (q_0)$ $\quad$ $\delta(q_1, 1) = (q_1)$

Short Trick:

1. We have to create DFA that accepts set of strings which when interpreted as binary number is divisible by '2'.
2. First write the input alphabets, example 0, 1.
3. If there will n states,
4. Then start writing states, as for $n = 2$: q_0 under 0, q_1 under 1.
5. Continue the process as, q_0 under 0 and q_1 under 1.

Example 5.14: Construction of a DFA, that accepts set of strings over $\Sigma = \{0, 1\}$ which when interpreted as binary number is divisible by '3', is shown in figure 5.16. For example, 110 in binary is equivalent to 6 in decimal and 6 is divisible by 3. For this automaton,
DFA can be described as $(\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_0\})$.
$L = \{$All the strings which when interpreted as binary number is divisible by '3'$\}$
$L = \{\epsilon, 0, 00, 11, 110, 1001, \ldots\}$

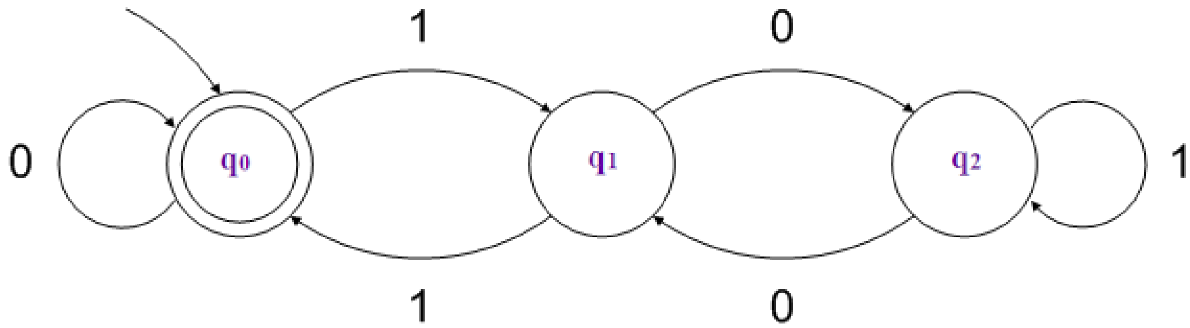

Figure 5.16: DFA that accepts set of strings which when interpreted as binary number is divisible by '3'.

The transition table is:

States	0	1
$\rightarrow q_0*$	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

The transition function is:

$\delta(q_0, 0) = (q_0)$ $\quad$ $\delta(q_0, 1) = (q_1)$

$\delta(q_1, 0) = (q_2)$ $\quad$ $\delta(q_1, 1) = (q_0)$

$\delta(q_2, 0) = (q_1)$ $\quad$ $\delta(q_2, 1) = (q_2)$

Short Trick:

1. We have to create DFA that accepts set of strings which when interpreted as binary number is divisible by '3'.
2. First write the input alphabets, example 0, 1.
3. If there will n states (n is the number of reminders),
4. Then start writing states, as for $n = 3$: q_0 under 0, q_1 under 1.
5. Continue the process as, q_2 under 0 and q_0 under 1.
6. q_1 under 0 and q_2 under 1.

The DFA above accepts empty string as a "number" divisible by 3. This can easily be fixed by adding one more intermediate state in front (S), which is shown in figure 5.17.

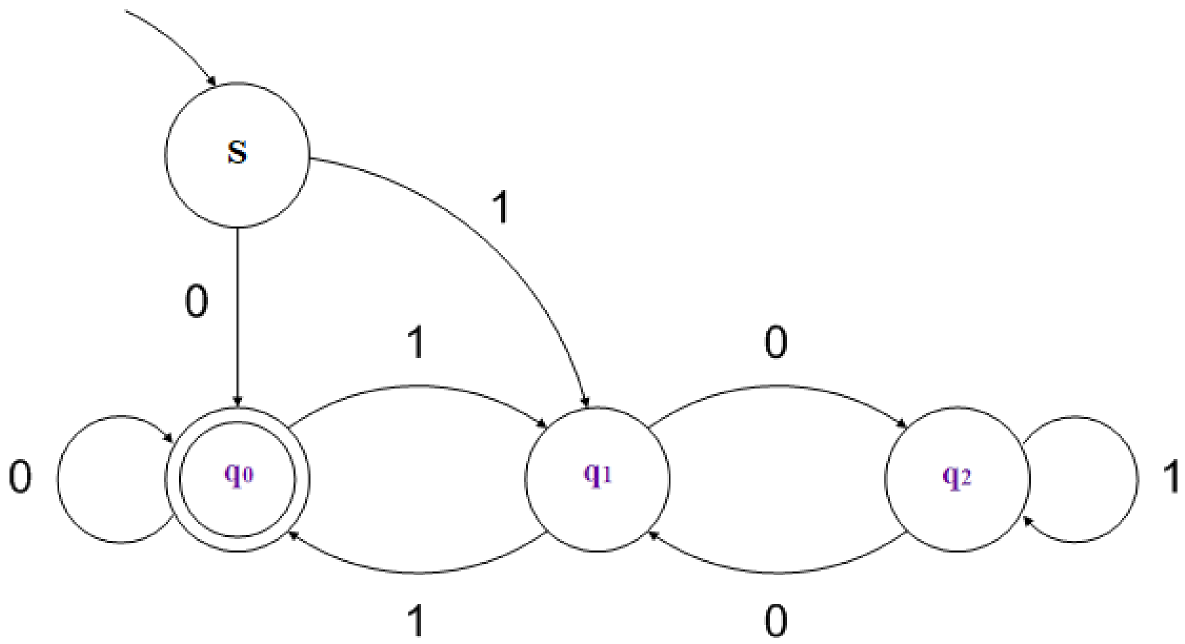

Figure 5.17: DFA that accepts set of strings which when interpreted as binary number is divisible by ‘3’ but not accepting empty string.