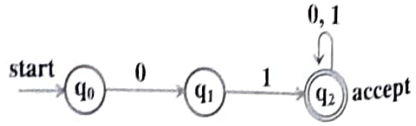
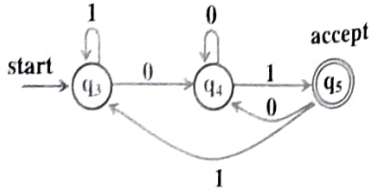
Example 18: Now, let us "Draw a DFA to accept set of all strings on the alphabet Σ = {0, 1} that either begins or ends or both with the substring 01".

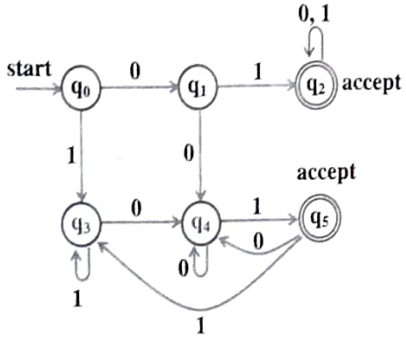
Solution: The DFA to accept strings of 0's and 1's starting with string 01 can be written as shown below (procedure remains same as Example 4):



The DFA to accept strings of 0's and 1's ending with string 01 can be written as shown below (procedure remains same as Example 5):



The two DFA can be joined to accept strings of 0's and 1's beginning with 01 and ending with 01 or both can be written as shown below:



δ	0	1
→ q_0	q_1	q_3
q_1	q_4	q_2
*q_2	q_2	q_2
q_3	q_4	q_3
q_4	q_4	q_5
*q_5	q_4	q_3

So, formally a DFA can be defined as **M = (Q, Σ, δ, q_0, F)** where

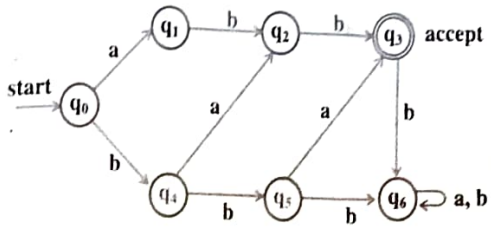
- Q = {q_0, q_1, q_2, q_3, q_4, q_5}
- Σ = {0, 1}
- δ is shown above using the transition diagram and transition table
- q_0 is the start state
- F = {q_2, q_5}

Example 19: Now, let us "Draw a DFA to accept the language L = {w: $n_a(w) \geq 1$, $n_b(w) = 2$}.

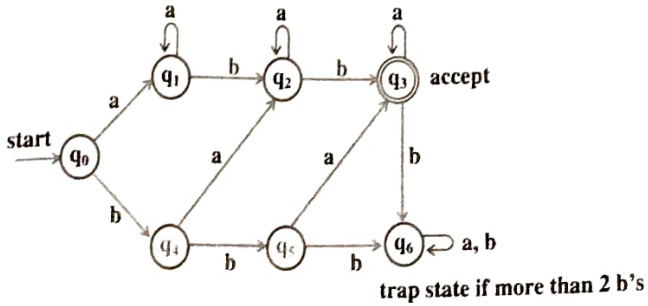
Solution: Note that the string should have exactly two *b*'s and at least one *a*. So, the minimum string that can be accepted by the DFA may be:

- abb
- bab
- bba

The skeleton DFA is shown below:



But, one or more a's can be present in the string. So, the above DFA can be written (using the same steps of designing technique) as shown below:

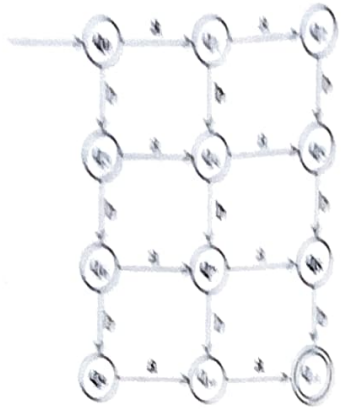


Example 20: Now, let us "Draw a DFA to accept the language L = {w: $n_a(w) = 2$, $n_b(w) >= 3$}.

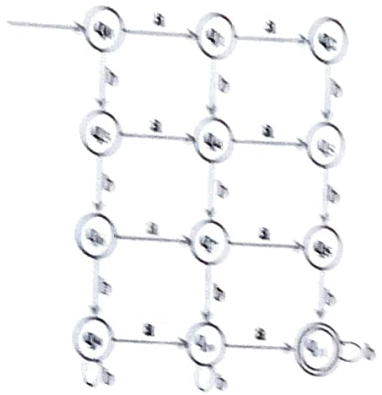
Solution: It is observed from the given language that the string that is accepted by DFA should have exactly two a's and at least three b's. So, the minimum strings that can be accepted are:

- aabbb
- abaBb
- abbab
- abbba
- baabb
- babab
- babba
- bbaab
- bbaba
- bbbaa

The skeleton DFA that accepts all the above strings is shown below:



Since, minimum three b's should be there, after three b's we can consume any number of b's. So, the above DFA can be written as shown below:



Note: From states q2, q5 and q8 do not define the transitions

1.6.2. Divisible by k Problems

In this section, let us discuss a method of constructing DFA that divide a number by k. For these types of problems, the transitions can be obtained using the following relation:

$$\delta(q_i, a) = q_j \text{ where } j = (r * i + d) \bmod k$$

- r is the radix of input. For binary r = 2
- i is the remainder obtained after dividing by k
- d represent digits. For binary d = { 0, 1}
- k is divisor

The steps to be followed to find FA using these types of problems is shown below:

Step 1: Identify the radix, input alphabets and the divider k.

Step 2: Compute the possible remainders. These remainders represent the states of DFA.

Step 3: Find the transitions using $\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$.

Step 4: Construct the DFA using the transitions obtained in step 3.

Example 1: Now, let us "Construct a DFA which accepts strings of 0's and 1's where the value of each string is represented as a binary number. Only the strings representing zero modulo five should be accepted. For example, 0000, 0101, 1010, 1111, etc. should be accepted".

Solution: The DFA can be obtained as shown below:

Step 1: Identify the radix, input alphabets and the divisor k. In this case, r = 2

$d = \{0, 1\}$, k = 5

Step 2: Compute the possible remainders: After dividing by k, the possible remainders are:

i = 0, 1, 2, 3, 4

Step 3: Compute transitions: The transitions can be computed using the following relation:

$$\delta(q_i, a) = q_j \text{ where } j = (r * i + d) \bmod k$$

with r = 2 and k = 5

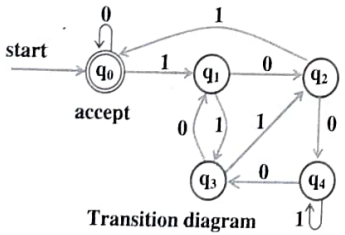
So, $j = (2i + d) \bmod 5$

remainder	d	(2* i +d) mod 5 = j	$\delta(q_i,d) = q_j$
i = 0	0	(2*0 + 0) mod 5 = 0	$\delta(q_0,0) = q_0$
	1	(2*0 + 1) mod 5 = 1	$\delta(q_0,1) = q_1$
i = 1	0	(2*1 + 0) mod 5 = 2	$\delta(q_1,0) = q_2$
	1	(2*1 + 1) mod 5 = 3	$\delta(q_1,1) = q_3$
i = 2	0	(2*2 + 0) mod 5 = 4	$\delta(q_2,0) = q_4$
	1	(2*2 + 1) mod 5 = 0	$\delta(q_2,1) = q_0$
i = 3	0	(2*3 + 0) mod 5 = 1	$\delta(q_3,0) = q_1$
	1	(2*3 + 1) mod 5 = 2	$\delta(q_3,1) = q_2$
i = 4	0	(2*4 + 0) mod 5 = 3	$\delta(q_4,0) = q_3$
	1	(2*4 + 1) mod 5 = 4	$\delta(q_4,1) = q_4$

Transitions of resulting DFA

Step 4: The DFA can be defined as **M = (Q, Σ, δ, q_0, F)** where

- **Q** = $\{q_0, q_1, q_2, q_3, q_4\}$
- **Σ** = {0, 1}
- $\mathbf{q_0}$ is the start state
- **F** = $\{q_0\}$
- δ is shown below using the transition diagram and transition table as shown below:



Transition diagram

δ	0	1
→*q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

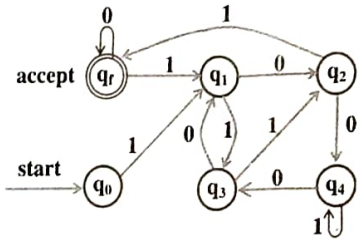
Transition Table

Note: Observe that for modulo 5, number of states of DFA will be 5.

In general, for modulo k, number of states of DFA will be k.

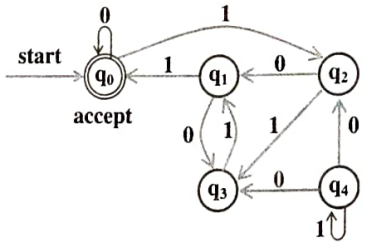
Example 2: Now, let us "obtain a DFA which accepts the set of all strings beginning with a 1 that when interpreted as a binary integer, is a multiple of 5. For example, 101, 1010, 1111 etc are multiples of 5. Note that 0101 is not beginning with 1 and it should not be accepted".

Solution: The solution to this problem is same as above problem. But, the number always should start with a 1. If a binary number starts with a 0, the number should never be accepted. So, let us rename the final state as q_f and have the new start state q_0 and from this symbol on 1, enter into state q_1. The resulting DFA is shown using the transition diagram as shown below:



Example 3: Now, let us "obtain a DFA that accepts set of all strings that, when interpreted in reverse as a binary integer, is divible by 5. Examples of strings in the language are 0, 10011, 1001100 and 0101".

Solution: The solution remains same as Example 1. But, reverse the direction of all arrow marks (except the arrow labeled with start). The resulting DFA is shown below:



Example 4: Now, let us "Draw a DFA to accept decimal strings divisible by 3".

Solution: The DFA can be obtained as shown below:

Step 1: Identify the radix, input alphabets and the divisor k: In this case, r = 10:

d = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, k = 3

Step 2: Compute the possible remainders: After dividing any decimal number by 3, results in following remainders:

i = 0, 1, 2 which implies q_0, q_1 and q_2 are the states of DFA

Step 3: Compute transitions: The transitions can be computed using the following relation: $\delta(q_i, a) = q_j$ where $\mathbf{j = (r * i + d) \bmod k}$

with r = 10 and k = 3

So, $\mathbf{j = (2i + d) \bmod 3}$

Note: For the sake of convenience, let us group the digits from 0 to 9 based on the remainders we get after dividing by 3 as shown below:

- {0, 3, 6, 9} with 0 as the remainder. So, δ from {0, 3, 6, 9} ⇒ δ from {0}
- {1, 4, 7} with 1 as the remainder. So, δ from {1, 4, 7} ⇒ δ from {1}
- {2, 5, 8} with 2 as the remainder. So, δ from {2, 5, 8} ⇒ δ from {2}

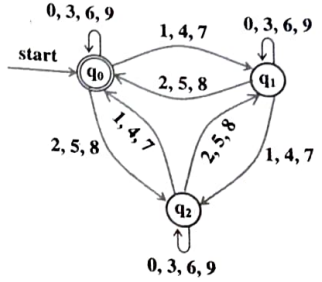
remainder	d	(2* i +d) mod 3 = j	$\delta(q_i,d) = q_j$		
i = 0	0	(10*0 + 0) mod 3 = **0**	$\delta(q_0,0) = q_0$	⇒	$\delta(q_0,\{0,3,6,9\}) = q_0$
	1	(10*0 + 1) mod 3 = **1**	$\delta(q_0,1) = q_1$	⇒	$\delta(q_0,\{1,4,7\}) = q_1$
	2	(10*0 + 2) mod 3 = **2**	$\delta(q_0,2) = q_2$	⇒	$\delta(q_0,\{2,5,8\}) = q_2$
i = 1	0	(10*1 + 0) mod 3 = **1**	$\delta(q_1,0) = q_1$	⇒	$\delta(q_1,\{0,3,6,9\}) = q_1$
	1	(10*1 + 1) mod 3 = **2**	$\delta(q_1,1) = q_2$	⇒	$\delta(q_1,\{1,4,7\}) = q_2$
	2	(10*1 + 2) mod 3 = **0**	$\delta(q_1,2) = q_0$	⇒	$\delta(q_1,\{2,5,8\}) = q_0$
i = 2	0	(10*2 + 0) mod 3 = **2**	$\delta(q_2,0) = q_2$	⇒	$\delta(q_2,\{0,3,6,9\}) = q_2$
	1	(10*2 + 1) mod 3 = **0**	$\delta(q_2,1) = q_0$	⇒	$\delta(q_2,\{1,4,7\}) = q_0$
	2	(10*2 + 2) mod 3 = **1**	$\delta(q_2,2) = q_1$	⇒	$\delta(q_2,\{2,5,8\}) = q_1$

Transitions of DFA

Step 4: The DFA can be defined as $\mathbf{M = (Q, \Sigma, \delta, q_0, F)}$ where

- $\mathbf{Q} = \{q_0, q_1, q_2\}$
- $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $\mathbf{q_0}$ is the start state

- $\mathbf{F} = \{q_0\}$
- δ is shown below using the transition diagram and transition table as shown below:



1.6.3. Modulo k Counter Problems

Example 1: Let us "Obtain a DFA to accept strings of even number of a's". The DFA can be constructed by following the steps one by one as shown below:

Solution: It is given that L = {w : w has even number of a's}.

Identify the number of states: The string w may have even number of a's or odd number of a's and results in following two cases (two states):

- **Case 0:** Strings that accepts Even number of a's is denoted by: **E**
- **Case 1:** Strings that accepts Odd number of a's is denoted by: **O**

Identify the start state and final state: Before reading any of the input symbols, number of a's will be zero which represent Even a's. So, the state E with even number of a's is the start state.

Since, it is required to accept Even number of a's the state E which accepts even number of a's is the final state.

Design: Once the start state and final states are identified the transitions can be easily obtained as shown below:

- From a state E, on reading *a* results in odd number of a's and hence change to odd state O. The transition is:

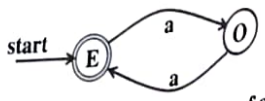
$\mathbf{\delta(E, a) = O}$

- From a state O, on reading *a* results in even number of a's and hence change to even state. The transition is:

  $\delta(O, a) = E$

So, the DFA $M = (Q, \Sigma, \delta, q_0, F)$ is defined as:

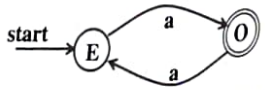
- $Q = \{E, O\}$
- $\Sigma = \{a\}$
- δ = shown in transition diagram



DFA to accept even number of a's

- $q_0 = E$ is the start state
- $F = \{E\}$

Note: The DFA to obtain odd number of a's can be obtained by making O state (odd state) as the final state and E state as the start state as shown below:



DFA to accept odd number of a's

Example 2: Let us "Obtain a DFA to accept the language $L = \{ w : |w| \bmod 3 = 0\}$ where $\Sigma = \{a\}$".

Solution: It is given that $L = \{w : |w| \bmod 3 = 0\}$ where $\Sigma = \{a\}$ } which indicates that the language consists of strings of multiples of 3 a's.

Identify the number of states: The language can be interpreted as strings of a's such that the number of a's in string is divisible by 3. Note that $|w| \bmod 3$ results in three cases:

- **Case 0:** Results in remainder 0: The state is identified as q_0. 

- **Case 1:** Results in remainder 1. The state is identified as q_1. 

- **Case 2:** Results in remainder 2. The state is identified as q_2. 

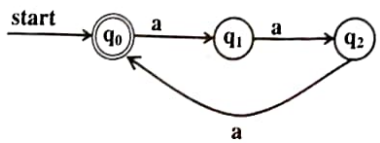
Identify the start state and final state: Before reading any inputs, number of a's will be zero. So, 0 mod 3 = 0 which results in case 0 . Hence, state q0 is start state. Since, it is required to accept string of a's such that $|w| \bmod 3 = 0$, which results in case 0, q_0 is considered as final state.

Design: Once the start state and final states are identified the transitions can be easily obtained as shown below:

- From a state with remainder 0 (case 0) denoted by q_0, on reading *a* results in remainder 1 (case 1) denoted by q_1. So, $\delta(q_0, a) = q_1$
- From a state with remainder 1 (case 1) denoted by q_1, on reading *a* results in remainder 2 (case 2) denoted by q_2. So, $\delta(q_1, a) = q_2$
- From a state with remainder 2 (case 2) denoted by q_2, on reading *a* results in remainder 0 (case 0) denoted by q_0. So, $\delta(q_2, a) = q_0$

So, the DFA $M = (Q, \Sigma, \delta, q0, F)$ is defined as:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a\}$
- δ : the transitions are shown using transition diagram as shown below:



- q_0 = start state
- $F = \{q_0\}$

The given language accepted by above DFA can also be represented as:

- $L = \{w : |w| \bmod 3 = 0\}$ where $\Sigma = \{a\}$ }

  or

- $L = \{w: n_a(w)$ are divisible by 3 where $\Sigma=\{a\}\}$

  or

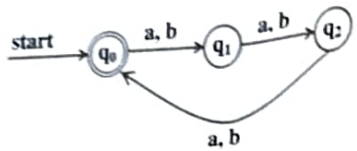
- $L = \{a^{3n} : n \geq 0\}$

Example 3: Let us "Obtain a DFA to accept the language $L = \{ w : |w| \bmod 3 = 0\}$ on $\Sigma = \{a, b\}$".

Solution: The language consists of strings of a's and b's whose length is a multiple of 3 and can be represented as:

$L = \{\varepsilon, aaa, aab, aba, abb, baa, bab, bba, bbb, \ldots\}$

The design is similar to that of the previous problem. But, add one extra label b for each edge as shown below:

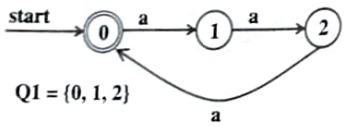


Example 4: Now, let us "Obtain a DFA to accept the following language L = {w such that

a) $|w| \bmod 3 \geq |w| \bmod 2$ where $w \in \Sigma^*$ and $\Sigma = \{a\}$

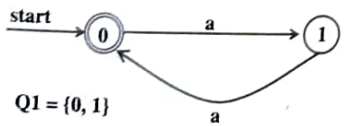
b) $|w| \bmod 3 \neq |w| \bmod 2$ where $w \in \Sigma^*$ and $\Sigma = \{a\}$

Solution: The DFA to accept a string w such that $|w| \bmod 3 = 0$ can be written as shown below (see Example 2):



M1

On similar lines, the DFA to accept a string w such that $|w| \bmod 2 = 0$ can be written as shown below (see Example 1):

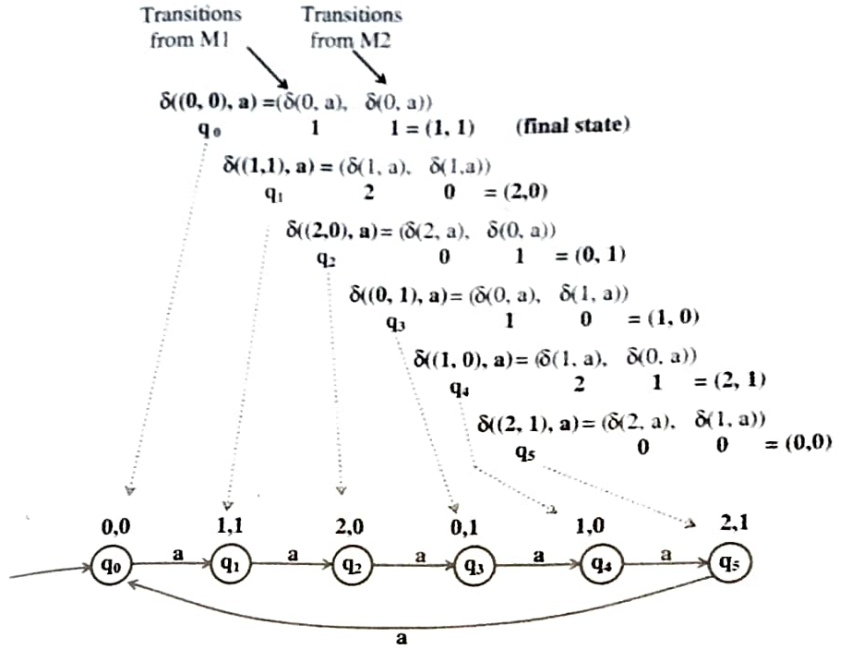


M2

Transitions: The transitions of DFA which has strings of w with $|w| \bmod 3$ and $|w| \bmod 2$ can be obtained by taking the cross product of Q1 and Q2 as shown below:

$$Q1 \times Q2 = \{ (0,0), (0,1), (1,0), (1,1), (2,0), (2,1) \}$$

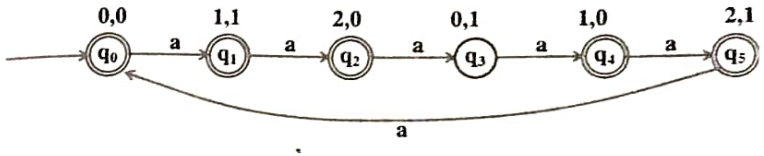
The transitions on each of the pair (x, y) can be obtained as shown below:



Case 1: To accept strings of w such that $|w| \bmod 3 \geq |w| \bmod 2$, the pairs (x, y) such that $x \geq y$ are final states. So, in the above DFA, the final states are:

$$F = \{ (0, 0), (1, 1), (2, 0), (1, 0), (2, 1) \}$$

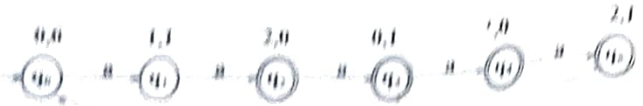
So, the DFA to accept the given language is shown below:



Case 2: To accept strings of w such that $|w| \bmod 3 \neq |w| \bmod 2$, $w \in \Sigma^*$ and $\Sigma = \{a\}$ the pairs (x, y) such that $x \neq y$ are final states. So, the final states in the DFA are:

$$F = \{ (2, 0), (0,1), (1, 0), (2, 1) \}$$

So, the DFA to accept the given language is shown below:



Example 5: Now, let us "Obtain a DFA to accept the following language $L = \{w$ such that

a) $|w|_a \bmod 3 = |w|_a \bmod 2$ where $w \in \Sigma^*$ and $\Sigma = \{a, b\}$

b) $|w|_a \bmod 3 \neq |w|_a \bmod 2$ where $w \in \Sigma^*$ and $\Sigma = \{a, b\}$

Solution: The solution is exactly similar to the above, but, the extra label b should be added along with a as shown below.

Case 1: To accept strings of w such that $|w|_a \bmod 3 = |w|_a \bmod 2$, $w \in \Sigma^*$ and $\Sigma = \{a, b\}$.

The pairs (x, y) such that $x == y$ are final states. So, in the above DFA, the final states are:

$$F = \{ (0, 0), (1, 1), (2, 0), (1, 0), (2, 1) \}$$

So, the DFA to accept the given language is shown below:

Case 2: To accept strings of w such that $|w|_a \bmod 3 \neq |w|_a \bmod 2$, $w \in \Sigma^*$ and $\Sigma = \{a, b\}$.

The pairs (x, y) such that $x \neq y$ are final states. So, the final states in the DFA are:

$$F = \{ (2, 0), (0, 1), (1, 0), (2, 1) \}$$

So, the DFA to accept the given language is shown below:

Example 6: Let us "Obtain a DFA to accept the language $L = \{ w \mid |w| \bmod 5 \neq 0 \}$ over $\Sigma = \{a\}$"

Solution: It is given that $L = \{ w \mid |w| \bmod 5 \neq 0 \}$ where $\Sigma = \{a\}$ which indicates that the language consists of strings of a's which are not multiples of 5 as a.

Identify the number of states: The given language can be interpreted as strings of only a's such that the number of a's in the string is not divisible by 5. Now, that

$$|w| \bmod 5$$

results in remainder 0, 1, 2, 3 and 4 which results in five cases as shown below.

- **Case 0:** results in remainder 0. The state is identified as q_0 (q_0)
- **Case 1:** results in remainder 1. The state is identified as q_1 (q_1)
- **Case 2:** results in remainder 2. The state is identified as q_2 (q_2)
- **Case 3:** results in remainder 3. The state is identified as q_3 (q_3)
- **Case 4:** results in remainder 4. The state is identified as q_4 (q_4)

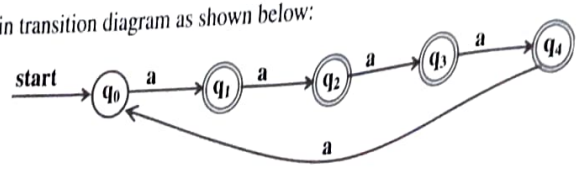
Identify the start state and final state: Before reading any of the input symbol, number of a's will be zero. So, 0 mod 5 = 0 which results in case 0. Hence, q_0 is the start state. Since it is required to accept strings of a's such that $|w| \bmod 5 \neq 0$ (i.e., remainder is not zero) while there is all other states can be the final state. So, the states q_1, q_2, q_3 and q_4 are final states.

Design: Once the start state and final states are identified, the transitions can be easily obtained as shown below:

- From a state with remainder 0 (i.e., q_0) denoted by q_0, on reading a results in remainder 1 (i.e., q_1) denoted by q_1. So, $\delta(q_0, a) = q_1$
- From a state with remainder 1 (i.e., q_1) denoted by q_1, on reading a results in remainder 2 (i.e., q_2) denoted by q_2. So, $\delta(q_1, a) = q_2$
- From a state with remainder 2 (i.e., q_2) denoted by q_2, on reading a results in remainder 3 (i.e., q_3) denoted by q_3. So, $\delta(q_2, a) = q_3$
- From a state with remainder 3 (i.e., q_3) denoted by q_3, on reading a results in remainder 4 (i.e., q_4) denoted by q_4. So, $\delta(q_3, a) = q_4$
- From a state with remainder 4 (i.e., q_4) denoted by q_4, on reading a results in remainder 0 (i.e., q_0) denoted by q_0. So, $\delta(q_4, a) = q_0$

So, the DFA $M = (Q, \Sigma, \delta, q_0, F)$ is defined as:

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{a\}$
- δ = shown in transition diagram as shown below:

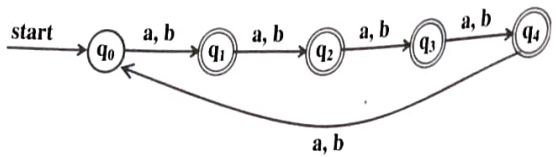


- q_0 = start state
- $F = \{q_1, q_2, q_3, q_4\}$

Example 7: Let us "Obtain a DFA to accept the language $L = \{ w : |w| \bmod 5 \neq 0 \}$ on $\Sigma = \{a,b\}$".

Solution: The design is exactly similar to the previous problem but each arrow is labeled with *a, b*. So, the DFA $M = (Q, \Sigma, \delta, q_0, F)$ is defined as:

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{a, b\}$
- δ = shown in transition diagram as shown below:



- q_0 = start state
- $F = \{q_1, q_2, q_3, q_4\}$

Example 8: Let us "Obtain a DFA to accept strings of a's and b's having even number of a's and even number of b's".

Solution: It is given that $L = \{w : w$ has even number of a's and even number of b's.$\}$

Identify the number of states: The string w made up of a's and b's may have:

- *Even number of a's* denoted by $\mathbf{E_a}$
- *Even number of b's* denoted by $\mathbf{E_b}$
- *Odd number of a's* denoted by $\mathbf{O_a}$
- *Odd number of b's* denoted by $\mathbf{O_b}$

So, even/odd a's along with even/odd b's results in following four cases (four states):

- **Case 0:** Strings having Even a's and Even b's is denoted by: q_0 — (E_a, E_b) (q_0)
- **Case 1:** Strings having Even a's, Odd b's is denoted by: q_1 — (E_a, O_b) (q_1)
- **Case 2:** Strings having Odd a's, Even b's is denoted by: q_2 — (O_a, E_b) (q_2)
- **Case 3:** Strings having Odd a's, Odd b's is denoted by: q_3 — (O_a, O_b) (q_3)

Identify the start state and final state: Before reading any of the input symbols, number of a's and number of b's will be zero which represent **E**ven a's and **E**ven b's (case 0). So, the state q_0 is the start state.

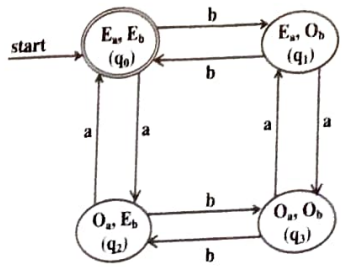
Since, it is required to accept Even a's and Even b's, the state q0 with even a's and even b's is the final state.

Design: Once the start state and final states are identified the transitions can be easily obtained as shown below:

- From a state E_a with even number of a's, on reading input symbol *a*, results in odd number of a's denoted by O_a. So, $\delta(\mathbf{E_a, a}) = \mathbf{O_a}$
- From a state O_a with odd number of a's, on reading input symbol *a*, results in even number of a's denoted by E_a. So, $\delta(\mathbf{O_a, a}) = \mathbf{E_a}$
- From a state E_b with even number of b's, on reading input symbol *b*, results in odd number of b's denoted by O_b. So, $\delta(\mathbf{E_b, b}) = \mathbf{O_b}$
- From a state O_b with odd number of b's, on reading input symbol *b*, results in even number of b's denoted by E_b. So, $\delta(\mathbf{O_b, b}) = \mathbf{E_b}$

So, the DFA $M = (Q, \Sigma, \delta, q0, F)$ is defined as:

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{a, b\}$
- δ = shown in transition diagram



DFA to accept even number of a's and even number of b's

- q_0 = start state
- F = $\{q_0\}$

Note: The language accepted by above DFA can be written as:

$$L = \{w : w \text{ has even number of a's and even number of b's.}\}$$

or

$$L = \{w : \text{Both } N_a(w) \text{ and } N_b(w) \text{ are divisible by } 2\}$$

or

$$L = \{w : \text{Both } N_a(w) \text{ and } N_b(w) \text{ are multiples of } 2\}$$

or

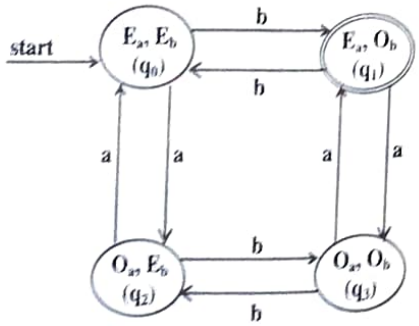
$$L = \{w \mid N_a(w) \bmod 2 = 0 \text{ and } N_b(w) \bmod 2 = 0\}$$

Note: $N_a(w)$ is the total number of a's in the string w and $N_b(w)$ is the total number of b's in the string w.

Note: The DFA to accept even number of a's and odd number of b's can be obtained by making:

E_a, O_b
(q_1)

as the only final state as shown below:

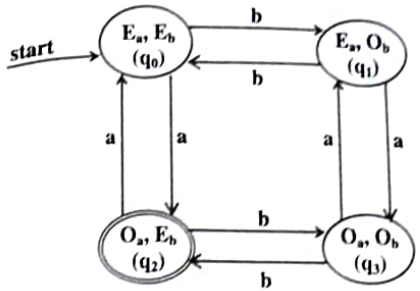


DFA to accept even number ofa's and odd number of b's

Note: The DFA to accept odd number of a's and even number of b's can be obtained by making:

O_a, E_b
(q_2)

as the only final state as shown below:

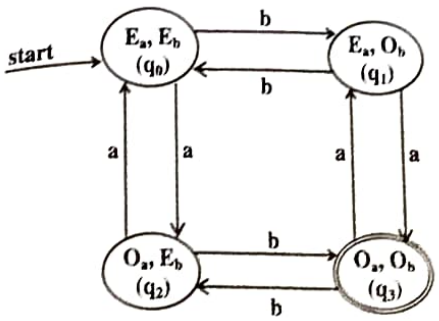


DFA to accept odd number of a's and even number of b's

Note: The DFA to accept odd number of a's and odd number of b's can be obtained by making:

O_a, O_b
(q_3)

as the only final state as shown below:



DFA to accept odd number of a's and odd number of b's

Example 9: Obtain a DFA to accept strings of a's and b's such that

$$L = \{w \mid w \in (a+b)^* \text{ such that } N_a(w) \bmod 3 = 0 \text{ and } N_b(w) \bmod 2 = 0\}$$

Solution: The $N_a(w)$ mod 3 gives the remainder after dividing number of a's by 3.

The possible remainders are $\{0, 1, 2\}$ and can be represented as:

$$\downarrow \quad \downarrow \quad \downarrow$$
$$Q_1 = \{A_0, A_1, A_2\} \qquad (1)$$

The $N_b(w)$ mod 2 gives the remainder after dividing number of b's by 2.

The possible remainders are {0, 1} and can be represented as:

$$Q_2 = \{B_0, B_1\}$$

Identify the states of DFA: Since each state of DFA should keep track of Na(w) mod 3 and Nb(w) mod 2, the possible states of the DFA can be obtained by Q1xQ2 (cross product) and can be represented as shown below:

$$Q1 \times Q2 = \{(A0, B0), (A0, B1), (A1, B0), (A1, B1), (A2, B0), (A2, B1)\}$$

where

- A0 indicates that $N_a(w) \bmod 3 = 0$
- A1 indicates that $N_a(w) \bmod 3 = 1$
- A2 indicates that $N_a(w) \bmod 3 = 2$
- B0 indicates that $N_b(w) \bmod 2 = 0$
- B1 indicates that $N_b(w) \bmod 2 = 1$

Identify the start state: Before reading any of the input symbols, number of a's and number of b's will be zero. So, Na(w) mod 3 = 0 and Nb(w) mod 2 = 0 which can be denoted by the state (A0, B0). So, (A0, B0) is the start state.

Identify the final state: Since, it is required to accept the language

$$L = \{w : N_a(w) \bmod 3 = 0 \text{ and } N_b(w) \bmod 2 = 0\}$$
$$\downarrow \qquad\qquad \downarrow$$
$$A0 \qquad\qquad B0$$

the state (A0, B0) is the final state.

Design: Once the start state and final states are identified, the transitions for the input symbol *a* can be obtained as shown below:

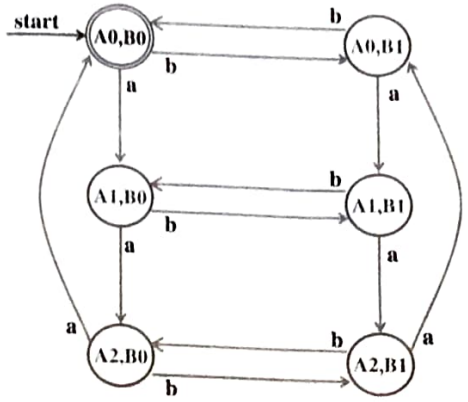
- From state A0, on reading input symbol *a*, the machine should change the state to A1. So, $\delta(A_0, a) = A_1$
- From state A1, on reading input symbol *a*, the machine should change the state to A2. So, $\delta(A_1, a) = A_2$
- From state A2, on reading input symbol *a*, the machine should change the state to A0. So, $\delta(A_2, a) = A_0$

Similarly, the transitions for the input symbol *b* can be obtained as shown below:

- From state B0, on reading input symbol *b*, the machine should change the state to B1. So, $\delta(B_0, b) = B_1$
- From state B1, on reading input symbol *b*, the machine should change the state to B0. So, $\delta(B_1, b) = B_0$

So, the DFA M = (Q, Σ, δ, q_0, F) is defined as:

- Q = {(A0,B0), (A0, B1), (A1, B0), (A1, B1), (A2,B0), (A2, B1)}
- Σ = {a, b}
- q_0 = {(A0, B0) is the start state
- F = {(A0, B0)}
- δ = shown in transition diagram



Note: By changing the final states of DFA various languages can be accepted by DFAas shown below:

- To accept the language $L = \{w \mid w \in (a+b)^* \; N_a(w) \bmod 3 = 1 \text{ and } N_b(w) \bmod 2 = 0\}$
  $\downarrow$ A1, $\downarrow$ B0

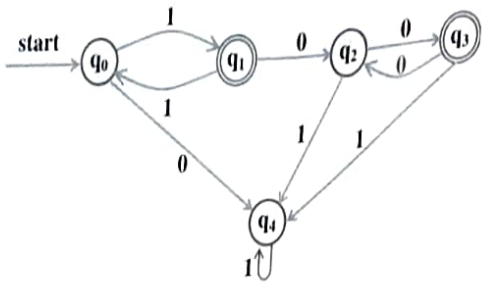
  make the state (A1, B0) as the final state.
- To accept the language $L = \{w \mid w \in (a+b)^* \; N_a(w) \bmod 3 = 2 \text{ and } N_b(w) \bmod 2 = 0\}$
  $\downarrow$ A2, $\downarrow$ B0

  make the state (A2, B0) as the final state.
- To accept the language $L = \{w \mid w \in (a+b)^* \; N_a(w) \bmod 3 = 2 \text{ and } N_b(w) \bmod 2 = 1\}$
  $\downarrow$ A2, $\downarrow$ B1

  make the state (A2, B1) as the final state and so on.

Example 10: Now, let us "Draw a DFA to accept the language: L = {w: w has odd number of 1's and followed by even number of 0's) Completely define DFA and transition function:

Solution: The DFA to accept strings of 0's and 1's such that the string has odd number of 1's and followed by even number of 0's is shown below:



Example 11: Now, let us "Obtain a DFA to accept strings of a's and b's such that the number of a's is divisible by 5 and number of b's is divisible by 3".

Note: Observe that the problem is similar to that the previous problem with more number of states.

Solution: The given language can be interpreted as shown below:

$$L = \{w \mid w \in (a+b)^* \; N_a(w) \bmod 5 = 0 \text{ and } N_b(w) \bmod 3 = 0\}$$

The $N_a(w)$ mod 5 gives the remainder after dividing number of a's by 5.

The possible remainders are { 0, 1, 2, 3, 4} and can be represented as:

$$\downarrow \; \downarrow \; \downarrow \; \downarrow \; \downarrow$$

$$Q_1 = \{A0, A1, A2, A3, A4\} \quad (1)$$

The $N_b(w)$ mod 3 gives the remainder after dividing number of b's by 3.

The possible remainders are {0, 1, 2} and can be represented as:

$$\downarrow \; \downarrow \; \downarrow$$

$$Q_2 = \{ B0, B1, B2 \} \quad (2)$$

Identify the states of DFA: Since each state of DFA should keep track of Na(w) mod 5 and Nb(w) mod 3, the possible states of the DFA can be obtained by Q1xQ2 (cross product) and can be represented as shown below:

$$Q1 \times Q2 = \{(A0, B0), (A0, B1), (A0, B2),$$
$$(A1, B0), (A1, B1), (A1, B2),$$
$$(A2, B0), (A2, B1), (A2, B2),$$
$$(A3, B0), (A3, B1), (A3, B2),$$
$$(A4, B0), (A4, B1), (A4, B2)$$
$$\}$$

where

- A0 indicates that $N_a(w) \bmod 5 = 0$
- A1 indicates that $N_a(w) \bmod 5 = 1$
- A2 indicates that $N_a(w) \bmod 5 = 2$
- A3 indicates that $N_a(w) \bmod 5 = 3$
- A4 indicates that $N_a(w) \bmod 5 = 4$
- B0 indicates that $N_b(w) \bmod 3 = 0$
- B1 indicates that $N_b(w) \bmod 3 = 1$
- B2 indicates that $N_b(w) \bmod 3 = 2$

Identify the start state: Before reading any of the input symbols, number of a's and number of b's will be zero. So, $N_a(w) \bmod 5 = 0$ and $N_b(w) \bmod 3 = 0$ which can be denoted by the state (A0, B0). So, (A0, B0) is the start state.

Identify the final state: Since, it is required to accept the language

$$L = \{w : N_a(w) \bmod 5 = 0 \text{ and } N_b(w) \bmod 3 = 0\}$$
$$\downarrow \qquad\qquad \downarrow$$
$$\mathbf{A0} \qquad\qquad \mathbf{B0}$$

the state (A0, B0) is the final state.

Design: Once the start state and final states are identified, the transitions for the input symbol *a* can be obtained as shown below:

- From state A0, on reading input symbol *a*, the machine should change the state to A1. So, $\delta(\mathbf{A_0, a}) = \mathbf{A_1}$
- From state A1, on reading input symbol *a*, the machine should change the state to A2. So, $\delta(\mathbf{A_1, a}) = \mathbf{A_2}$
- From state A2, on reading input symbol *a*, the machine should change the state to A3. So, $\delta(\mathbf{A_2, a}) = \mathbf{A_3}$
- From state A3, on reading input symbol *a*, the machine should change the state to A4. So, $\delta(\mathbf{A_3, a}) = \mathbf{A_4}$
- From state A4, on reading input symbol *a*, the machine should change the state to A0. So, $\delta(\mathbf{A_4, a}) = \mathbf{A_0}$

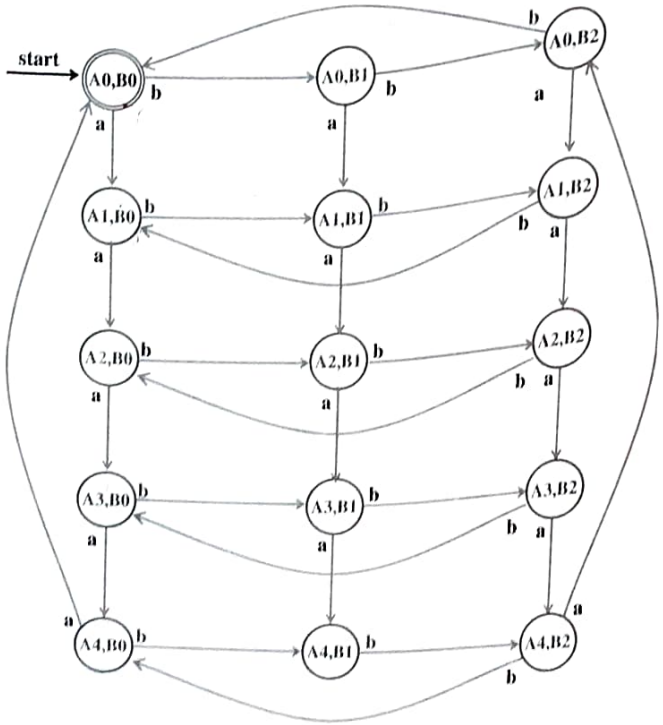
Similarly, the transitions for the input symbol *b* can be obtained as shown below:

- From state B0, on reading input symbol *b*, the machine should change the state to B1. So, $\delta(\mathbf{B_0, b}) = \mathbf{B_1}$
- From state B1, on reading input symbol *b*, the machine should change the state to B2. So, $\delta(\mathbf{B_1, b}) = \mathbf{B_2}$

- From state B2, on reading input symbol *b*, the machine should change the state to B_0. $\delta(B_2, b) = B_0$

So, the DFA M = (Q, Σ, δ, q0, F) is defined as:

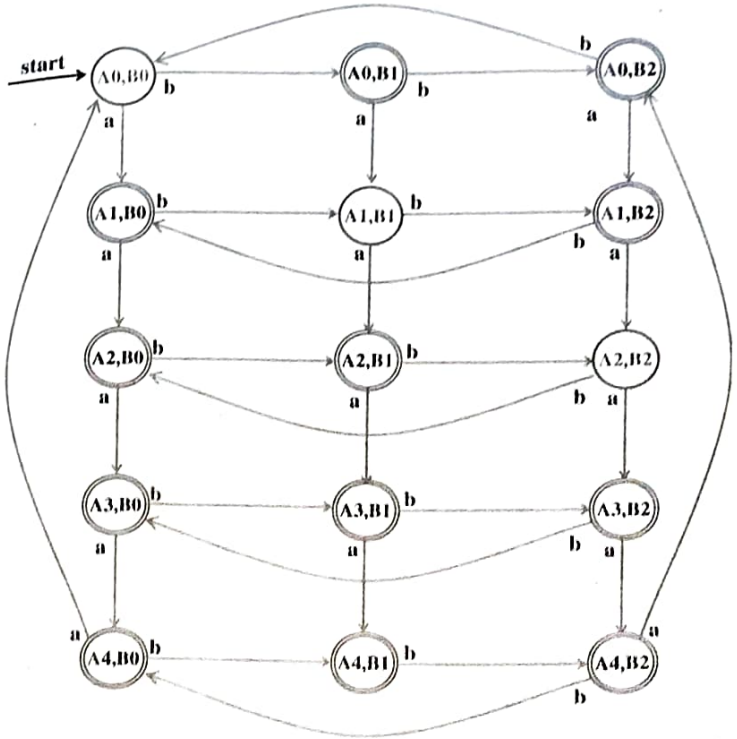
- Q = {(A0, B0), (A0, B1), (A0, B2),
  (A1, B0), (A1, B1), (A1, B2),
  (A2, B0), (A2, B1), (A2, B2),
  (A3, B0), (A3, B1), (A3, B2),
  (A4, B0), (A4, B1), (A4, B2) }
- Σ = {a, b}
- q_0 = (A0, B0) is the start state
- F = {(A0, B0)}
- δ = shown below using the transition diagram



Now, let us "Construct a DFA to accept the following language":

$L = \{w : n_a(w) \bmod 5 \neq n_b(w) \bmod 3\}$

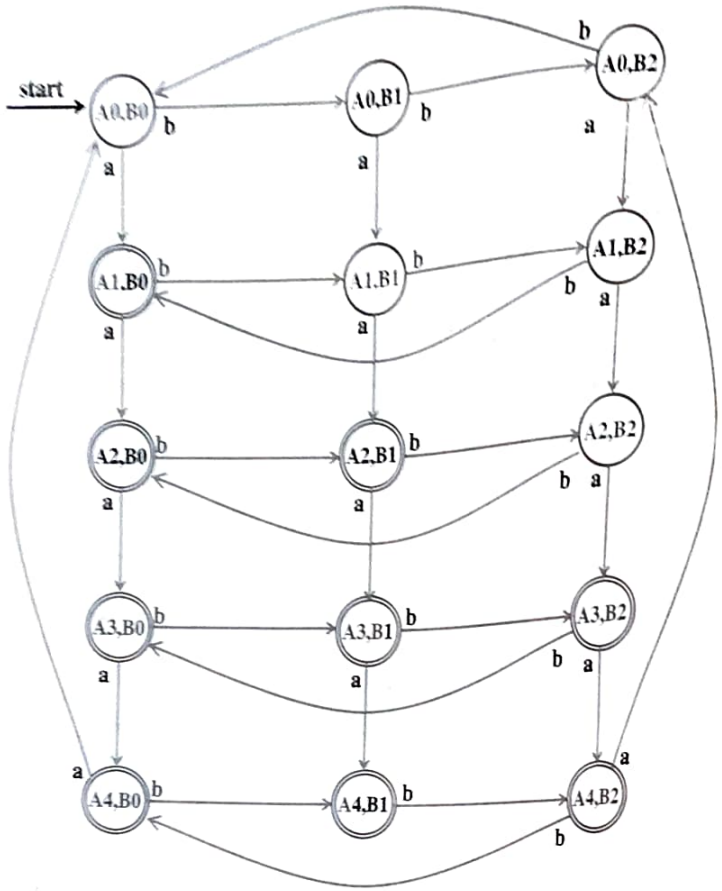
Note: The design is exactly same as the above problem. Except the states (A0, B0), (A1, B1) and (A2, B2), the rest of the states are final states. The DFA to accept the given language is shown below:



Now, let us "Construct a DFA to accept the following language":

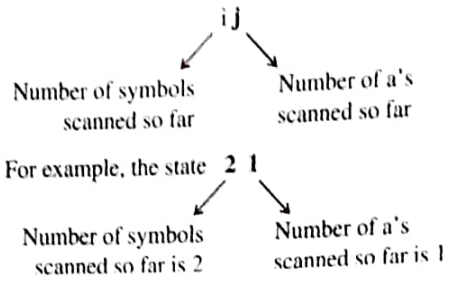
$L = \{w : n_a(w) \bmod 5 > n_b(w) \bmod 3\}$

Note: The design is exactly similar to the above problem. But, in all possible states (A_i, B_j), index i should be greater than index j. So, the final DFA to accept the above language is shown below:



Example 12: Now, let us "Obtain a DFA to accepts strings of a's and b's such that each block of 5 consecutive symbols have at least two a's".

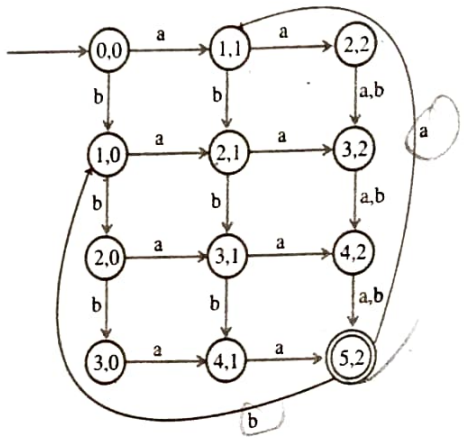
Solution: This DFA should keep track of number of symbols scanned for and number of a's. Hence, each state of DFA is represented as shown below:



The transition can be obtained using the following relationships:

- The state 00 is the start state of DFA.
- If $i \leq 4$ and $j \leq 1$ then $\delta(ij, a) = \delta(i+1\ \ j+1)$.
- If $i \leq 4$ and $j = 2$ then $\delta(ij, x) = \delta(i+1\ \ j)$ where x can be either a or b.
- 52 indicates that current block has 5 symbols and has at least 2 a's. So, the string has to be accepted and 52 is the final state.
- 50 and 51 states indicate that the block has 5 symbols and at least 2 a's are not present. So, the string has to be rejected and represent the trap state.
- $\delta(52, a) = 11$ indicate that the beginning of the next block has scanned one symbol and has one a.
- $\delta(52, b) = 10$ indicate that the beginning of the next block has scanned one symbol and has no a.

The complete DFA is shown below:



Example 13: Now, let us "Prove that $\delta^*(q, xa) = \delta(\delta^*(q, x), a)$" where x is a string and a is the current input symbol.

Proof: It is required to prove that $\delta^*(q, xa) = \delta(\delta^*(q, x), a)$. From this statement it is observed that:

$w = xa$

where

- a is the last symbol of w
- x is the remaining string of w
- q is the current state of the machine

From state q on input string w let us assume that the machine enters into state p. This is given by the transition:

$$\delta^*(q, w) = p \quad (1)$$

Since w = xa, the above transition can be written as shown below:

$$\delta^*(q, xa) = p \quad (2)$$

Since w = xa, first, we should find the transition on string x and then on symbol a.

On string x: From state q on input string x, let the machine enters into state r. This is denoted by:

$$\delta^*(q, x) = r \quad (3)$$

On symbol a: From state r on input symbol a, the machine should enter into state p. This is denoted by:

$$\delta(r, a) = p$$

Replacing the state r in above transition using equation (3), we have:

$$\delta(\delta^*(q, x), a) = p \quad (4)$$

Comparing equation (2) and (4) observe that R.H.S's are equal. Therefore, L.H.S.'s are also equal. So,

$$\delta^*(q, xa) = \delta(\delta^*(q, x), a)$$

$$\text{i.e., } \delta^*(q, w) = \delta^*(q, xa) = \delta(\delta^*(q, x), a)$$

Example 14: Show that $\delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$ for any state w and strings x and y.

Note: The symbol δ^* can also be denoted by the symbol $\hat{\delta}$.

Solution: The statement to be proved is:

$$\delta^*(q, xy) = \delta^*(\delta^*(q, x), y) \quad (1)$$

Let us proceed by induction on the length of y.

Basis: Let $\delta(q, x) = p$ (2)

If $y = \varepsilon$, L.H.S. of Eq. (1) $= \delta^*(q, xy)$

$= \delta^*(q, x)$ [By replacing y by ε]

$= p$ [from Eq. (2)]

Now, consider R.H.S. of Eq. (1) $= \delta^*(\delta^*(q, x), y)$

$= \delta^*(p, y)$ [By relation (2)]

$= \delta^*(p, \varepsilon)$ [Replacing y by ε]

$= p$ [According to the property: $\delta(q, \varepsilon) = q$]

Since, L.H.S. is same as R.H.S the relation (1) holds good.

Induction: Let us split the string y into za i.e., **y = za** (3)

a is last symbol of y

Consider R.H.S. of relation (1) $= \delta^*(\delta^*(q, x), y)$

$= \delta^*(\delta^*(q, x), za)$ Substituting relation (3)

$= \delta(\delta^*(\delta^*(q, x), z), a)$ Treat $\delta^*(q,x)$ as a state

$= \delta(\delta^*(q, xz), a)$ Inductive hypothesis

$= \delta(\delta^*(q, xz), a)$ Inductive hypothesis

$= \delta^*(q, xza)$ By definition

$= \delta^*(q, xy)$ Using relation (3)

= L.H.S. of relation (1)

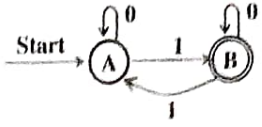
Hence, the proof.

Example 15: Consider the DFA with following transition table:

	δ	0	1
→	A	A	B
	*B	B	A

Informally describe the language accepted by this DFA and prove by induction on the length of an input string, that your description is correct.

Solution: The transition diagram corresponding to the transition table is shown below:



State A: The machine in state A can consume any number of 0's. But, the number of consumed at state A is 0 which is EVEN. So, from EVEN state on consuming a 1, the machine enters into ODD state called B.

State B: In ODD state B, the machine can consume any number of 0's. But, if the input symbol is 1, the machine goes to state A which consumes odd number of 1's.

Informal description of language: So, the language accepted by the machine is "Strings of 0 and 1's with odd number of 1's".

Proof: It is required to show by induction that $\delta^*(A, w) = A$ if and only if w has even number of 1's.

Basis: $|w| = 0$. Since w is an empty string, surely has an even number of 1's, namely zero 1's, and so $\delta^*(A, w) = A$.

Induction: Let us consider a string shorter than w. So, let $w = za$

where a is either 0 or 1.

Case 1: Let a = 0. So, if w has an odd number of 1's, z also has odd number of 1's.

By the inductive hypothesis, $\delta^*(A, z) = B$. The transitions of the DFA tell us

$$\delta^*(A, w) = B$$

If w has an even number of 1's, then z also has even number of 1's. By the inductive hypothesis, $\delta^*(A, z) = A$. The transitions of the DFA tell us

$$\delta^*(A, w) = A$$

Thus, in this case, $\delta^*(A, w) = A$ if and only if w has an even number of 1's.

$\delta^*(A, w) = B$ if and only if w has an odd number of 1's.

Case 2: Let a = 1. So, if w has an even number of 1's, then z has an odd number of 1's. By the inductive hypothesis, $\delta^*(A, z) = B$. The transitions of the DFA tell us

$$\delta^*(A, w) = A$$

If w has an odd number of 1's, then z has an even number of 1's. By the inductive hypothesis, $\delta^*(A, z) = A$. The transitions of the DFA tell us

$$\delta^*(A, w) = B$$

Thus, in this case as well, $\delta^*(A, w) = A$ if and only if w has an even number of 1's

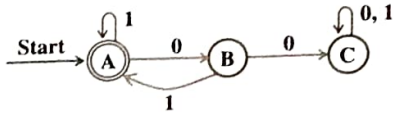
$\delta^*(A, w) = B$ if and only if w has odd number of 1's.

Example 16: Consider the DFA with following transition table:

δ	0	1
→*A	B	A
*B	C	A
C	C	C

Informally describe the language accepted by this DFA and prove by induction on the length of an input string, that your description is correct.

Solution: The transition diagram corresponding to the transition table is shown below:



Observe the following facts from the above transition diagram:

- A string w having either ε or ends in 1 but does not having the substring 00 is consumed at state A
- A string w ends in 0 and does not have the substring 00 is consumed at state B
- A string w that contains the substring 00 is consumed at state C

Informal description of the language: Since A is the final state, the language accepted by DFA is "Strings of 0's and 1's having ε or ending with 1 but does not have a substring 00".

Proof: To start with machine will be in start state A.

Basis: If $w = \varepsilon$, the machine will be in state A which is the final state. Thus, empty string is accepted by DFA.

Induction: If w has only 1's the machine will be in state A. So, w1 does not contain the substring 00 and hence the string is accepted by DFA.

If w ends with 0, the machine goes to state B and hence string ends with one 0. But, on 1 the machine enters into state A and thus the substring 00 is not accepted.

1.7. Applications of Finite Automata

Now, let us see "Why to study finite automata? or What are applications of finite automata?"

Some of the applications where automata play an important role are shown below:

- Design of digital circuits: The FA is used during designing and checking the behavior of the digital circuits using software. The FA is very useful in hardware design such as circuit verification, in design of automatic traffic signals etc.
- Compiler construction: Used in the design of *lexical analyzer* (the first phase of compiler design) which breaks the input text into various units such as identifiers, keywords, punctuation etc.
- String matching: In designing a software for identifying the words, phrases and other patterns in large bodies of text (such as collection of web pages).
- String processing: To write software for processing the natural language (Ex: Speech processing). Large natural vocabularies can be described which includes the applications such as spelling checkers and advisers, multi-language dictionaries, indenting the documents etc.
- Software design: In building the software to verify the systems having finite number of states (for example, communication protocols in computer networks).
- Other applications: The FA are used in variety of applications in Artificial intelligence and knowledge engineering, in game theory and games, computer graphics, linguistics, etc.

Exercises

1. What is DFA? Explain with an example.
2. When we say that a language is accepted by the machine? Explain with example.
3. When a given language is not accepted by DFA? Explain with example.
4. How DFA's can be represented? Explain with example.
5. What is a transition diagram/graph?
6. Obtain a DFA to accept strings of a's and b's starting with the string ab.
7. Draw a DFA to accept string of 0's and 1's ending with the string 011.
8. Obtain a DFA to accept strings of a's and b's having a sub string aa.
9. Obtain a DFA to accept strings of a's and b's except those containing the substring aab.
10. Obtain DFAs to accept strings of a's and b's having exactly one a, atleast one a, not more than three a's.
11. Obtain a DFA to accept the language L = {awa | w ∈ (a+b)* }.
12. Obtain a DFA to accept even number of a's, odd number of a's.
13. Obtain a DFA to accept strings of a's and b's having even number of a's and b's.
14. Obtain a DFA to accept odd number of a's and even number of b's.
15. Obtain a DFA to accept even number of a's and odd number of b's.

16. Obtain a DFA to accept strings of a's and b's having odd number of a's and b's.
17. Obtain a DFA to accept strings of 0's, 1's and 2's beginning with a '0' followed by odd number of 1's and ending with a '2'.
18. Obtain a DFA to accept binary odd numbers.
19. Obtain a DFA to accept strings of a's and b's starting with at least two a's and ending with at least two b's.
20. Obtain a DFA to accept strings of a's and b's with at most two consecutive b's.
21. Obtain a DFA to accept the language L = { w : |w| mod 3 = 0 on Σ = {a, b}.
22. Obtain a DFA to accept the language L = { w : |w| mod 5 ≠ 0 on Σ = {a, b}.
23. What is a regular language?
24. What are the applications of finite automaton?

1.8. Disadvantages of DFA

Now, let us see "What are the various disadvantages of DFA?" The various disadvantages of DFA are listed below:

- Constructing a DFA is difficult
- The DFA cannot guess about its input
- The DFA is not very powerful
- At any point of time, the DFA is in only state. So, a DFA does not have the power to be in several states at once

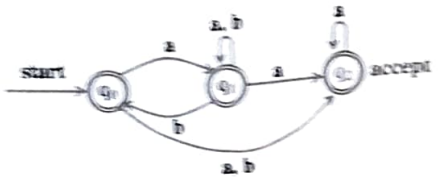
1.9. Why NFA?

Computers are completely deterministic machines (DFA). The state of the computer can be predicted from the input and initial state. We cannot find a computer which is non-deterministic. In such case, the question is "Why non-deterministic Finite Automata?" All the disadvantages of DFA mentioned earlier can be overcome using Non-Deterministic Finite Automata (in short NFA or NDFA). The various advantages of NFA are:

- Very easy to construct

- A "non-deterministic" finite automaton has the ability to guess something about its input
- A "non-deterministic" finite automaton is more powerful than DFA
- It has the power to be in several states at once
- An NFA is an efficient mechanism to describe some complicated languages concisely

Before defining NFA, let us consider the following transition diagram:



Let us see "What is the specialty of the above finite automaton?" Observe the following facts:

- From a given state there is possibility of zero, one or more edges leaving that state after consuming the input symbol.

Example 1: From state q_2, there are zero edges (indicates no edges) for the input symbol b.

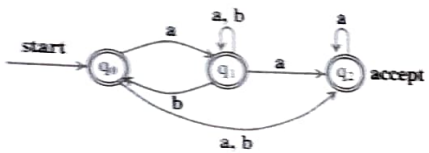
Example 2: From state q_2, there is one edge (q_2, q_2) labeled a.

Example 3: From state q_0, there are two edges (q_0, q_1) and (q_0, q_2) labeled a.

Note: In the FA shown above, there can be zero, one or more transitions on an input symbol. Such machines are called non-deterministic finite state machines or non-deterministic finite automata.

1.10. Non-Deterministic Finite Automaton

Before worrying about the definition, let us consider the following pictorial representation of NFA.



From the above figure, observe following components of NFA:

- **States:** The NFA has three states q_0, q_1 and q_2 and can be represented as

$$Q = \{q_0, q_1, q_2\}$$

Note: The power set of Q is denoted by 2^Q which is set of subsets of set Q. This is denoted as shown below:

$$2^Q = \{\ \{\}, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\ \}$$

- **Input alphabets:** Each edge is labeled with a or b and represent the input alphabets which can be denoted as:

$$\Sigma = \{a, b\}$$

- **Transitions:** Transition is nothing but change of state after consuming an input symbol. If there is a change of state from q_i to q_j on an input symbol a, then we write

$$\delta(q_i, a) = q_j$$

If there is a change of state from q_i to q_j and q_i to q_k on an input symbol a, then we write

$$\delta(q_i, a) = \{q_j, q_k\}$$

The transition diagram of above NFA is shown below:

Current state	Input	Next state	Representations
q_0	a	q_1, q_2	$\delta(q_0, a) = \{q_1, q_2\}$
q_0	b	q_2	$\delta(q_0, b) = q_2$
q_1	a	q_1, q_2	$\delta(q_1, a) = \{q_1, q_2\}$
q_1	b	q_0, q_2	$\delta(q_1, b) = \{q_0, q_1\}$
q_2	a	q_2	$\delta(q_2, a) = q_2$
q_2	b	q_1	$\delta(q_2, b) = q_1$

δ: **Q x Σ to 2^Q** (power set)

Now, let us see "What is a transition function?" The transition function δ is defined as:

$$\delta: Q \times \Sigma \text{ to } 2^Q$$

which is read as "δ is a transition function which maps Q X Σ to 2^Q". For example, the change of state from state q on input symbol a to states $p_1, p_2, \ldots, p_n$ is denoted by

$$\delta(q, a) = \{ p_1, p_2, \ldots, p_n \}$$

where

- δ: is a function called transition function
- q: is the first parameter representing the current state of the machine
- a: is the second parameter representing the current input symbol read
- $p_1, p_2, \ldots, p_n$ represent possible states of machine

Note: In other words, the transition function δ accepts two parameters namely state q and input symbol a as the parameters and returns set of states the machine enters into.

- **Start state (q_0):** q_0 with the label start is treated as the start state.
- **Final state (q_2):** q_2 with two circles is treated as the final or accept state.

Note: From this discussion, it is observed that the NFA has five components:

(states, input alphabets, transitions, start state, final states)

↓ ↓ ↓ ↓ ↓

Q Σ δ q_0 F

With this concept, now let us see "What is a non-deterministic finite automaton (NFA)?"

❖ **Definition:** The non-deterministic finite automaton in short NFA is 5-tuple or quintuple indicating five components:

$$M = (Q, \Sigma, \delta, q_0, F)$$

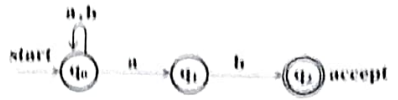
where

- M is the name of the machine. It can also be called by any name.
- Q is non-empty, finite set of states.
- Σ is non-empty, finite set of input alphabets.
- δ: Q X Σ to 2^Q i.e. δ is transition function which is a mapping from Q X Σ to 2^Q. Based on the current state and input symbol, the machine enters into one or more states.

- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is set of accepting or final states

Suppose δ(q, a) is in P where P is in 2^Q and a is in Σ. This indicates that there is a transition from state q on an input symbol a to set of states P. Note: In an NFA there can be zero, one or more transitions on an input symbol.

For example, an NFA to accept strings of a's and b's ending with *ab* is shown below



The above NFA can be specified formally as M = (Q, Σ, δ, q_0, F) where

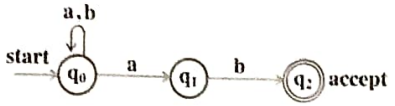
- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- q_0 is the start state
- $F = \{q_2\}$
- δ is shown below using the transition table:

δ	a	b
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
*q_2	ϕ	ϕ

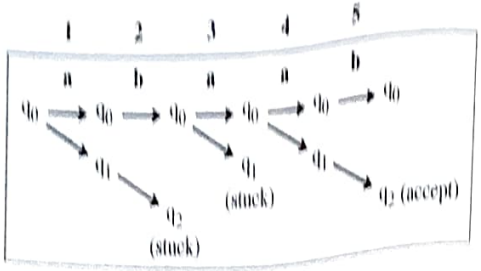
Note: In the transition table of NFA, each entry in the table should be denoted by a set. Also, when there is no transition at all from a given state on an input symbol, make an entry ϕ indicating an empty set.

1.10.1. Moves made by NFA

Now, let us see "What are the states an NFA is in during the processing of input sequence abaab and abb?".



Solution: The states an NFA is in during the processing of input sequence *abaab* is shown below:



The machine always starts from initial state q_0. The various actions performed by NFA for the string *abaab* is shown below:

1) From q_0 on reading *a*, the NFA may go to $\mathbf{q_0}$ **or** $\mathbf{q_1}$

2) q_0 on *b* NFA goes to state $\mathbf{q_0}$

   q_1 on *b* NFA goes to state $\mathbf{q_2}$

3) q_0 on *a* NFA goes to $\mathbf{q_0}$ **or** $\mathbf{q_1}$

   Note: q_2 on *a* there is no transition and hence NFA is stuck.

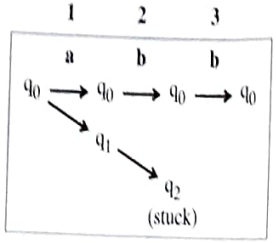
4) q_0 on *a* NFA goes to $\mathbf{q_0}$ **or** $\mathbf{q_1}$

   Note: q_1 on *a* there is no transition and hence NFA is stuck.

5) q_0 on *b* NFA goes to $\mathbf{q_0}$. At the end, the NFA is in q_0 (non-final state). So, this path is not chosen.

   But, q_1 on *b* NFA goes to q_2. So, at the end, NFA is in q_2 which is a final state and hence the input string **abaab** is accepted by the machine.

The states an NFA is in during the processing of input sequence *abb* is shown below:



The machine always starts from initial state q_0. The various actions performed by NFA for the string abaab is shown below:

1) From $\mathbf{q_0}$ on reading *a*, the NFA may go to $\mathbf{q_0}$ or $\mathbf{q_1}$

2) $\mathbf{q_0}$ on *b* NFA goes to state $\mathbf{q_0}$

   $\mathbf{q_1}$ on *b* NFA goes to state $\mathbf{q_2}$

3) $\mathbf{q_0}$ on *b* NFA goes to state $\mathbf{q_0}$

   Note: q_2 on *b* there is no transition and hence NFA is stuck.

After the string *abb*, the machine is in state q_0 which is non-final state. So the string *abb* is rejected by the machine.

1.10.2. Extended Transition Function of NFA to Strings

Note: The transition $\delta(\mathbf{q, a}) = \mathbf{P}$ accepts two parameters namely state *q* and input symbol *a* as parameters and returns a set of states P where $P = \{p_1, p_2, p_3, \ldots, p_n\}$ which represent the possible states of the NFA at a given instance. But, if there is a change of state from *q* to set of states P where $P\{p_1, p_2, p_3, \ldots, p_n\}$ on input string *w*, then we use extended transition function denoted by δ^*.

Note: We can also use $\hat{\delta}$ in place of δ^*. But, in our book let us use δ^* to denote extended transition. Now, let see "What is extended transition function δ^* for NFA?".

❖ **Definition:** The extended transition function δ^* describes what happens to a state of machine when the input is a string (sequence of symbols). Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA. The extended transition function δ^*: $\mathbf{Q \times \Sigma^* \text{ to } 2^Q}$ is defined recursively as shown below:

- **Basis:** $\delta^*(q, \varepsilon) = \{q\}$. This indicates that if the machine is in state q and read no input, then the machine is still in state q.
- **Induction:** Let $w = xa$ where *a* is the last symbol of *w* and *x* is the remaining string of *w*. Let *q* is the current state and *x* is the string to be processed and after consuming the string *x*, let the state of the machine is $\{p_1, p_2, p_3, \ldots, p_m\}$.

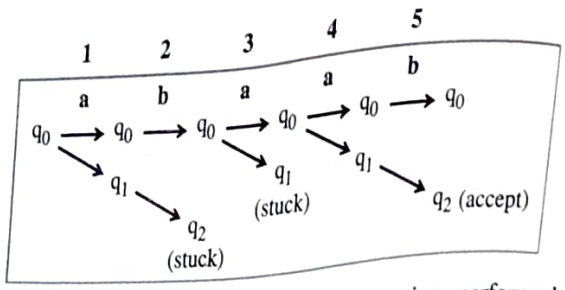
  i.e. $\delta^*(q, x) = \{p_1, p_2, p_3, \ldots, p_m\}$

  Let the transition from $\{p_1, p_2, p_3, \ldots, p_m\}$ on input symbol *a* is:

  $\delta(\{p_1, p_2, p_3, \ldots, p_m\}, a) = \{r_1, r_2, r_3, \ldots, r_k\}$

  i.e. $\bigcup_{i=1}^{m} \delta(p, a) = \{r_1, r_2, r_3, \ldots, r_k\}$

Then $\delta^*(q, w) = \delta^*(q, xa) = \{r_1, r_2, r_3, \ldots, r_k\}$.



The machine always starts from initial state q_0. The various actions performed by NFA for the string *abaab* is shown below:

1) From q_0 on reading *a*, the NFA may go to $\mathbf{q_0}$ **or** $\mathbf{q_1}$

2) q_0 on *b* NFA goes to state $\mathbf{q_0}$

   q_1 on *b* NFA goes to state $\mathbf{q_2}$

3) q_0 on *a* NFA goes to $\mathbf{q_0}$ **or** $\mathbf{q_1}$

   Note: q_2 on *a* there is no transition and hence NFA is stuck.

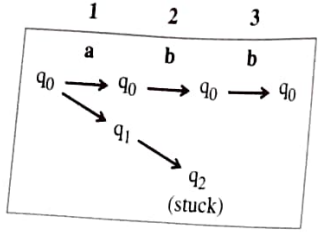
4) q_0 on *a* NFA goes to $\mathbf{q_0}$ **or** $\mathbf{q_1}$

   Note: q_1 on *a* there is no transition and hence NFA is stuck.

5) q_0 on *b* NFA goes to $\mathbf{q_0}$. At the end, the NFA is in q_0 (non-final state). So, this path is not chosen.

   But, q_1 on *b* NFA goes to q_2. So, at the end, NFA is in q_2 which is a final state and hence the input string **abaab** is accepted by the machine.

The states an NFA is in during the processing of input sequence *abb* is shown below:



The machine always starts from initial state q_0. The various actions performed by NFA for the string abaab is shown below:

1) From $\mathbf{q_0}$ on reading *a*, the NFA may go to $\mathbf{q_0}$ or $\mathbf{q_1}$

2) $\mathbf{q_0}$ on *b* NFA goes to state $\mathbf{q_0}$

   $\mathbf{q_1}$ on *b* NFA goes to state $\mathbf{q_2}$

3) $\mathbf{q_0}$ on *b* NFA goes to state $\mathbf{q_0}$

   Note: q_2 on *b* there is no transition and hence NFA is stuck.

After the string *abb*, the machine is in state q_0 which is non-final state. So the string *abb* is rejected by the machine.

1.10.2. Extended Transition Function of NFA to Strings

Note: The transition $\delta(\mathbf{q, a}) = \mathbf{P}$ accepts two parameters namely state *q* and input symbol *a* as parameters and returns a set of states P where $P = \{p_1, p_2, p_3, \ldots, p_n\}$ which represent the possible states of the NFA at a given instance. But, if there is a change of state from *q* to set of states P where $P\{p_1, p_2, p_3, \ldots, p_n\}$ on input string *w*, then we use extended transition function denoted by δ^*.

Note: We can also use $\hat{\delta}$ in place of δ^*. But, in our book let us use δ^* to denote extended transition. Now, let see "What is extended transition function δ^* for NFA?".

❖ **Definition:** The extended transition function δ^* describes what happens to a state of machine when the input is a string (sequence of symbols). Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA. The extended transition function δ^*: $\mathbf{Q \times \Sigma^* \text{ to } 2^Q}$ is defined recursively as shown below:

- **Basis:** $\delta^*(q, \varepsilon) = \{q\}$. This indicates that if the machine is in state q and read no input, then the machine is still in state q.
- **Induction:** Let $w = xa$ where *a* is the last symbol of *w* and *x* is the remaining string of *w*. Let *q* is the current state and *x* is the string to be processed and after consuming the string *x*, let the state of the machine is $\{p_1, p_2, p_3, \ldots, p_m\}$.

  i.e. $\delta^*(q, x) = \{p_1, p_2, p_3, \ldots, p_m\}$

  Let the transition from $\{p_1, p_2, p_3, \ldots, p_m\}$ on input symbol *a* is:

  $\delta(\{p_1, p_2, p_3, \ldots, p_m\}, a) = \{ r_1, r_2, r_3, \ldots, r_k \}$

  i.e. $\bigcup_{i=1}^{m} \delta(p, a) = \{r_1, r_2, r_3, \ldots, r_k\}$

Then $\delta^*(q, w) = \delta^*(q, xa) = \{r_1, r_2, r_3, \ldots, r_k\}$.

Note: Thus, various properties of extended transition functions for an NFA can be:

- $\delta^*(q, \varepsilon) = \{q\}$
- $\delta^*(q, w) = \delta^*(q, xa) = \{\hat{\delta}(\{\delta^*(q, x)\}, a\}$ where $w = xa$
- $\delta^*(q, w) = \delta^*(q, ax) = \{\delta^*(\{\delta(q, x)\}, x\}$ where $w = ax$

For example, change of state from state q on input string w to set of states $\{r_1, r_2, r_3, \ldots, r_k\}$ denoted by

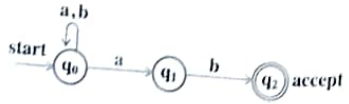
$$\delta^*(q, w) = \{r_1, r_2, r_3, \ldots, r_k\}$$

where

- δ^*: is a function called as extended transition function
- q: is the first parameter representing the current state of the machine
- w: is the second parameter representing the current input string being read
- $\{r_1, r_2, r_3, \ldots, r_k\}$: represent the possible states of the machine which is returned by the transition function

Note: The transition function δ^* accepts two parameters namely state q and input string w as the parameters and returns set of states of the machine.

Now, let us see "What are the moves made by the following NFA while processing the string *abaab* using the extended transition function?".



Solution: The moves made by the DFA for the input string *abaab* using δ^* is obtained starting from ε and taking prefix of *abaab* in increasing size as shown below:

- For prefix ε: $\delta^*(q_0, \varepsilon) = \{q_0\}$ ..... (1)
- For prefix a:
$$\delta^*(q_0, a) = \hat{\delta}(\delta^*(q_0, \varepsilon), a\} \quad \text{Substituting (3)}$$
$$= \delta(q_0, a)$$
$$= \{q_0, q_1\} \quad \text{..... (2)}$$

- For prefix ab:
$$\delta^*(q_0, ab) = \delta(\delta^*(q_0, a), b\} \quad \text{Substituting (2)}$$
$$= \delta(\{q_0, q_1\}, b)$$
$$= \delta(q_0, b) \cup \delta(q_1, b)$$
$$= \{q_0\} \cup \{q_2\} \phi$$
$$= \{q_0, q_2\} \quad \text{..... (3)}$$
- For prefix aba:
$$\delta^*(q_0, aba) = \delta(\delta^*(q_0, ab), a\} \quad \text{Substituting (3)}$$
$$= \delta(\{q_0, q_2\}, a)$$
$$= \delta(q_0, a) \cup \delta(q_2, a)$$
$$= \{q_0, q_1\} \cup \phi$$
$$= \{q_0, q_1\} \quad \text{..... (4)}$$
- For prefix abaa:
$$\delta^*(q_0, abaa) = \delta(\delta^*(q_0, aba), a\} \quad \text{Substituting (3)}$$
$$= \delta(\{q_0, q_1\}, a)$$
$$= \delta(q_0, a) \cup \delta(q_1, a)$$
$$= \{q_0, q_1\} \cup \phi$$
$$= \{q_0, q_1\} \quad \text{..... (5)}$$
- For prefix abaab: $\delta^*(q_0, abaab) = \delta(\delta^*(q_0, abaa), b\}$ Substituting (4)
$$= \delta(\{q_0, q_1\}, b)$$
$$= \delta(q_0, b) \cup \delta(q_1, b)$$
$$= \{q_0\} \cup \{q_2\}$$
$$= \{q_0, q_1\}$$

Note: After the string abaab, the states of NFA = $\{q_0, q_2\}$. Since the possible states of NFA after consuming abaab has one final state, the string abaab is accepting by the machine.

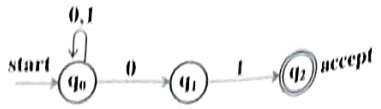
1.10.3. Language Accepted by a NFA

Now, let us "Define the language accepted by NFA". The language accepted by NFA is formally be defined as follows:

❖ **Definition:** Let $M \equiv (Q, \Sigma, \delta, q_0, F)$ be an NFA. A string w is accepted by the machine M, if it takes the initial state q_0 to final state, i.e., $\delta^*(q_0, w)$ is in F. Thus, the languages accepted by NFA represented as L(M) can be formally written as:

$$L(M) = \{w \mid w \in \Sigma^* \text{ and } \delta^*(q_0, w) \text{ is in } F\}$$

Now, let us "Prove formally that the following NFA accepts the language $L = \{w: w \text{ ends in } 01\}$".



Proof: It is given that the above NFA accepts the language:

$$L = \{w: w \text{ ends in } 01\}$$

The statement can be proved by mutual induction on the length of string w accepted by the three states shown in three cases as shown below:

- Case 1: $\delta^*(q_0, w) = q_0$ for every w
- Case 2: $\delta^*(q_0, w) = q_1$ for every w ending in *0*
- Case 3: $\delta^*(q_0, w) = q_2$ for every w ending in *01*

Basis: Consider a string $w = \varepsilon$ where $|w| = 0$.

Case 1: $\delta(q_0, \varepsilon) = q_0$ by definition. Hence case 1 is proved.

Case 2: $\delta(q_0, \varepsilon) = q_0$ by definition. Here, ε do not end with *0* and hence $\delta(q_0, \varepsilon)$ does not contain q_1. Hence case 2 is proved.

Case 3: $\delta(q_0, \varepsilon) = q_0$ by definition. Here, ε do not end with *01* and hence $\delta(q_0, \varepsilon)$ does not contain q_2. Hence case 3 is proved.

Induction hypotheses: Let $w = xa$ where a is either 0 or 1 and assume the three statements case 1 through case 2 holds good for x.

Let $|x| = n$. So, $|w| = n+1$

We have to prove that the three cases mentioned above are true for $n + 1$.

Proof: The proof for all the three cases can be obtained as shown below:

Case 1: $\delta^*(q_0, w) = q_0$. This statement is true. Because, on any of the input symbols 0 and 1, the NFA may stay in q_0. Hence, case 1 is proved.

Case 2: Assume w ends in 0, i.e., $a = 0$. We proved that $\delta^*(q_0, w) = q_0$. So, for the string x can write $\delta^*(q_0, x) = q_0$. Since there is a transition from q_0 to q_1 on 0, we conclude that $\delta^*(q_0, x) = q_1$. Hence, case 2 is proved.

Case 3: Assume w ends in 01. Let $w = xa$ where $a = 1$ and x is a string such that $\delta^*(q_0, x) = q_1$ where the string ends with 0. Since there is a transition from q_1 to q_2 on 1, we conclude that $\delta^*(q_0, w) = q_2$. Hence, case 3 is proved.

Note: Thus, the different properties of the transition function with respect to NFA are:

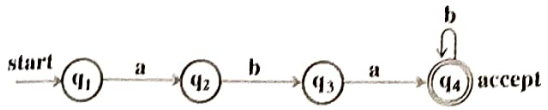
$$\delta(q, \varepsilon) = \delta^*(q, \varepsilon) = q$$
$$\delta(q, wa) = \delta(\delta^*(q, w), a) = P_j$$
$$\delta(q, aw) = \delta^*(\delta(q, a), w) = P_q$$

where q is in Q, a is im Σ, w is in Σ^* and P_j and P_q are the set of states which are reachable from q.

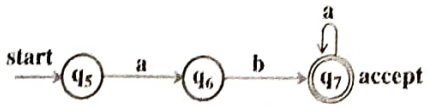
Example 1: Obtain an NFA to accept the following language:

$$L = \{w \mid w \in abab^n \text{ or } aba^n \text{ where } n \geq 0\}$$

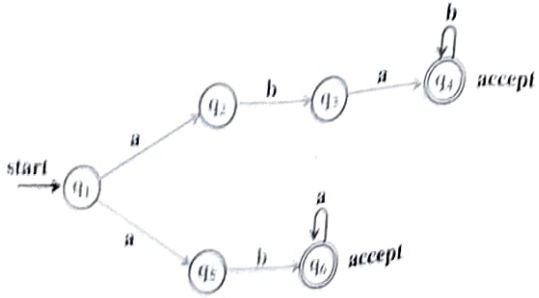
Solution: The machine to accept $abab^n$ where $n \geq 0$ is shown below:



The machine to accept aba^n where $n \geq 0$ is shown below:

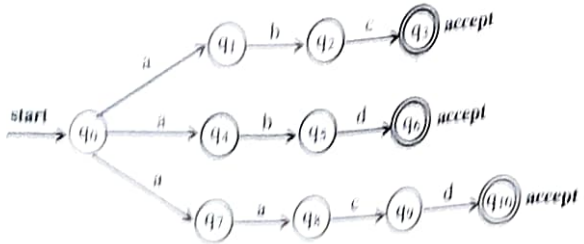


Since both the machines accept *a* as the first input symbol, the states q_1 and q_5 can be merged into a single state and the machine to accept either $abab^n$ or aba^n where $n \geq 0$ is shown below:



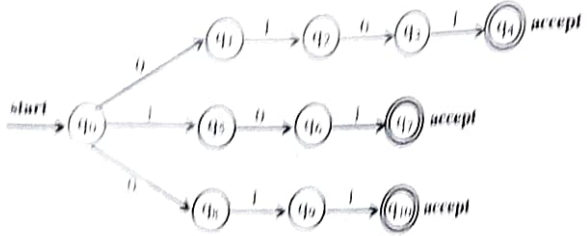
Example 1: Design an NFA to recognize the following set of strings abc, abd and aacd.

Solution: An NFA to recognize the following set of strings abc, abd and aacd is shown below.



Example 2: Obtain an NFA to recognize the following set of strings 0101, 101 and 011.

Solution: An NFA to recognize the following set of strings 0101, 101 and 011 is shown below.



1.11. Conversion from NFA to DFA

Digital computers are deterministic machines. Given the input, the state of the machine is predictable. Sometimes, constructing DFA is difficult compared to NFA. So, given any problem we construct a NFA. This is an efficient mechanism to describe some complicated languages concisely. Practically, non-deterministic machines will not exist. So, we convert an NFA in to a DFA. Now, let us see "What are the methods using which an NFA can be converted into a DFA?". The NFA can be converted into DFA using two methods:

- Subset construction
- Lazy evaluation

1.11.1. Conversion from NFA to DFA (Subset Construct Method)

Now, let us "Describe the subset construction procedure to convert an NFA into a DFA"

Procedure NFA_DFA: Given an NFA $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ which accepts the language $L(M_N)$, we can find an equivalent DFA $M_D = (Q_D, \Sigma, \delta_D, q_0, F_D)$ such that $L(M_D) = L(M_N)$.

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, $\{q_0\}$ is the start state of DFA.

Step 2: Identify the alphabets of DFA: The input alphabets of DFA are the input alphabets of NFA. So, $\Sigma = \{a, b\}$.

Step 3: Identify Q_D which are the states of DFA: The set of subsets of Q_N will be the states of DFA Q_D. So, if Q_N has n states then Q_D will have 2^n states. For example,

Let $Q_N = \{q_0, q_1, q_2\}$ then $|Q_N| = 3$

$Q_D = \{\ \}, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\ \}$

1 2 3 4 5 6 7 8

So, $|Q_D| = 8$

Note: In this example, the number of states of NFA = 3 and hence number of states of DFA = 8. If n is number of states of NFA the number of states of DFA will be 2^n.

In general, $\{q_i, q_j, \ldots, q_k\}$ is considered as a state in Q_D.

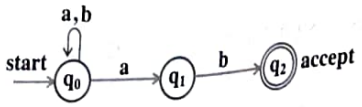
Step 4: Identify the final states of DFA: If $\{q_i, q_j, \ldots, q_k\}$ is a state in Q_D, then $\{q_i, q_j, \ldots, q_k\}$ will be final state of DFA provided one of $q_i, q_j, \ldots, q_k$ is the final state of NFA.

Step 5: Identify the transitions (i.e., δ_D) of DFA: For each state $\{q_i, q_j, \ldots, q_k\}$ in Q_D and for each input symbol a in Σ, the transition can be obtained as shown below:

$$\delta_D(\{q_i, q_j, \ldots, q_k\}, a) = \delta_N(q_i, a) \cup \delta_N(q_j, a) \cup \ldots \delta_N(q_k, a)$$

Thus, DFA can be obtained using subset construction method.

Example: Now, let us "Obtain the DFA for the following NFA using subset construction method".



Solution: The transition table for the above DFA can be written as shown below:

δ	a	b
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
*q_2	ϕ	ϕ

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, $\{q_0\}$ is the start state of DFA.

Step 2: Identify the alphabets of DFA: The input alphabets of NFA are the input alphabets of DFA. So, $\Sigma = \{a, b\}$.

Step 3: Identify Q_D which are the states of DFA: Here, $QN = \{q_0, q_1, q_2\}$. Its subsets are the states of DFA. So, states of DFA are:

$$Q_D = \{ \phi, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\} \}$$

Step 4: Identify the final states of DFA: Since q_2 is the final state of NFA in the above set, wherever q_2 is present as an element, the corresponding set is the final state of DFA. So

$$F_D = \{ \{q_2\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\} \}$$

Step 5: Identify the transitions (i.e., δ_D) of DFA: Obtain the transitions for each of the states of Q_D obtained in step 3 as shown below:

For state ϕ:

Input symbol = *a*

$\delta_D(\phi, a) = \phi$

Input symbol = *b*

$\delta_D(\phi, b) = \phi$

For state $\{q_0\}$:

Input symbol = *a*

$\delta_D(\{q_0\}, a) = \{q_0, q_1\}$

Input symbol = *b*

$\delta_D(\{q_0\}, b) = \{q_0\}$

For state $\{q_1\}$:

Input symbol = *a*

$\delta_D(\{q_1\}, a) = \phi$

Input symbol = *b*

$\delta_D(\{q_1\}, b) = q_2$

For state $\{q_2\}$:

Input symbol = *a*

$\delta_D(\{q_2\}, a) = \phi$

Input symbol = *b*

$\delta_D(\{q_2\}, b) = \phi$

For state $\{q_0, q_1\}$:

Input symbol = *a*

$$\begin{aligned}\delta_D(\{q_0, q_1\}, a) &= \delta_N(\{q_0, q_1\}, a) \\ &= \delta_N(q_0, a) \cup \delta_N(q_1, a) \\ &= \{q_0, q_1\} \cup \phi \\ &= \{q_0, q_1\}\end{aligned}$$

Input symbol = *b*

$$\begin{aligned}\delta_D(\{q_0, q_1\}, b) &= \delta_N(\{q_0, q_1\}, b) \\ &= \delta_N(q_0, b) \cup \delta_N(q_1, b) \\ &= \{q_0\} \cup \{q_2\} \\ &= \{q_0, q_2\}\end{aligned}$$

For state $\{q_0, q_2\}$:

Input symbol = *a*

$$\begin{aligned}\delta_D(\{q_0, q_2\}, a) &= \delta_N(\{q_0, q_2\}, a) \\ &= \delta_N(q_0, a) \cup \delta_N(q_2, a) \\ &= \{q_0, q_1\} \cup \phi \\ &= \{q_0, q_1\}\end{aligned}$$

Input symbol = *b*

$\delta_D(\{q_0, q_2\}, b) = \delta_N(\{q_0, q_2\}, b)$

$= \delta_N(q_0, b) \cup \delta_N(q_2, b)$

$= \{q_0\} \cup \phi$

$= \{q_0\}$

For state $\{q_1, q_2\}$:

Input symbol = *a*

$\delta_D(\{q_1, q_2\}, a) = \delta_N(\{q_1, q_2\}, a)$

$= \delta_N(q_1, a) \cup \delta_N(q_2, a)$

$= \phi \cup \phi$

$= \phi$

Input symbol = *b*

$\delta_D(\{q_1, q_2\}, b) = \delta_N(\{q_1, q_2\}, b)$

$= \delta_N(q_1, b) \cup \delta_N(q_2, b)$

$= \{q_2\} \cup \phi$

$= \{q_2\}$

For state $\{q_0, q_1, q_2\}$:

Input symbol = *a*

$\delta_D(\{q_0, q_1, q_2\}, a) = \delta_N(\{q_0, q_1, q_2\}, a)$

$= \delta_N(q_0, a) \cup \delta_N(q_1, a) \cup \delta_N(q_2, a)$

$= \{q_0, q_1\} \cup \phi \cup \phi$

$= \{q_0, q_1\}$

Input symbol = *b*

$\delta_D(\{q_0, q_1, q_2\}, b) = \delta_N(\{q_0, q_1, q_2\}, b)$

$= \delta_N(q_0, b) \cup \delta_N(q_1, b) \cup \delta_N(q_2, b)$

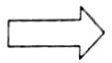
$= \{q_0\} \cup \{q_2\} \cup \phi$

$= \{q_0, q_2\}$

Now, all the above transitions can be represented using transition table as shown below:

δ	a	b
φ	φ	φ
→ $\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
$\{q_1\}$	φ	$\{q_2\}$
*$\{q_2\}$	φ	φ
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
*$\{q_0,q_2\}$	$\{q_0,q_1\}$	$\{q_0\}$
*$\{q_1,q_2\}$	φ	$\{q_2\}$
*$\{q_0,q_1,q_2\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$

By renaming the states of DFA as A, B, C, D, E, F, G, H



δ	a	b
A	A	A
→ B	E	B
C	A	D
*D	A	A
E	E	F
*F	E	B
*G	A	D
*H	E	F

Note: Even though we have 8 states, observe from the table that B is the start state. The states reachable from B are B, E, F. The rest of the states are not reachable and hence can be eliminated.

1.11.2. Disadvantage of Subset Construction Method

Now, let us see "What are the disadvantages of subset construction method?". Observe that from the above table that there are 2^n states and from each state we have the transition from input symbol in Σ and hence the time complexity to convert an NFA to DFA is $\Sigma * 2^n$. Since the time complexity is exponential, the procedure takes very long time to construct the table. This exponential time complexity can be avoided using another technique called "Lazy evaluation" on the subsets.

1.11.3. Conversion from NFA to DFA (Lazy Evaluation Method)

Now, let us "Describe the lazy evaluation method to convert NFA into a DFA". The lazy evaluation method of obtaining a DFA from NFA is shown below:

Procedure NFA_DFA: Given an NFA $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ which accepts the language L(MN), we can find an equivalent DFA $M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ such that $L(M_D) = L(M_N)$.

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, $\{q_0\}$ is the start state of DFA.