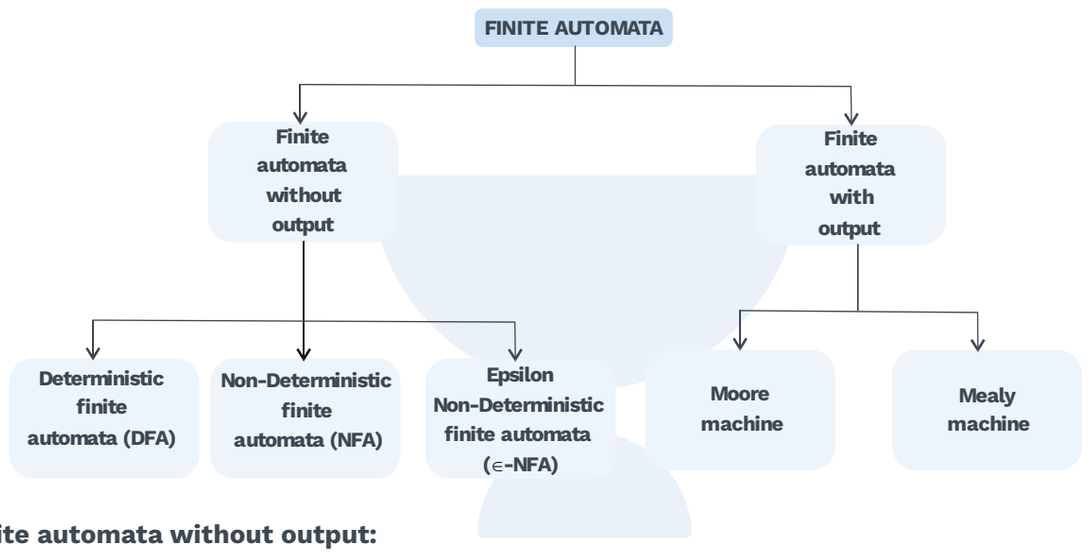
2 Finite Automata

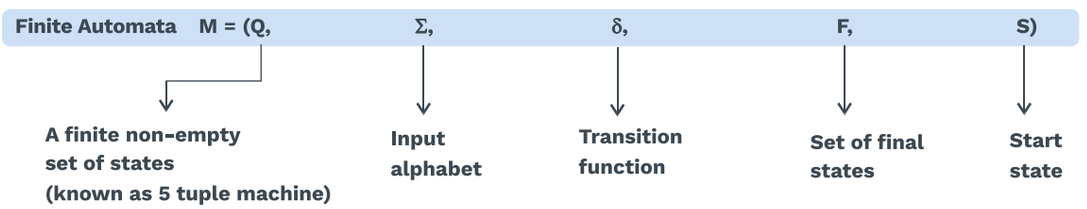
2.1 FINITE AUTOMATA

Definitions

It is a mathematical model which contains a finite number of states and transitions among states in response to inputs.



Finite automata without output:



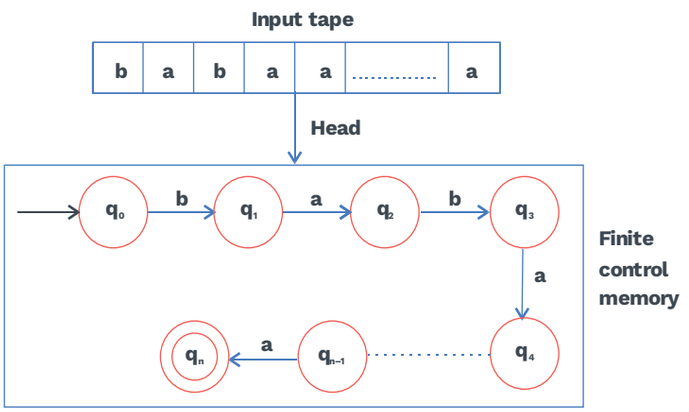


Fig. 2.1 Diagram of Finite Automata

Input tape: It consists of many cells where each cell can have only one symbol. Input tape contains a single string.

Head: It reads one symbol at a time from an input string.

Finite control memory: Memory which is having the finite number of states.

2.2 FINITE ACCEPTOR (FINITE AUTOMATA WITHOUT OUTPUT)

Deterministic finite automata (DFA)

- A DFA is set of 5 tuple and defined as:

$$M = \{Q, \Sigma, \delta, q_0, F\}$$

Where
- Q : Finite set of states
- Σ : Input alphabet
- δ : Transition function
- q_0 : Initial State
- F : A finite set of final states

Here
- δ : Transition function

$$\boxed{Q \times \Sigma \rightarrow Q}$$

↓

Next State

$\{A, B, C\} \times \{a, b\} \rightarrow$ Only deterministic output will come.

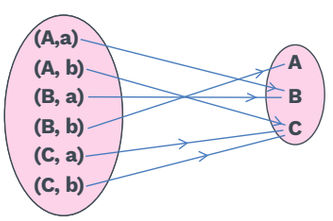
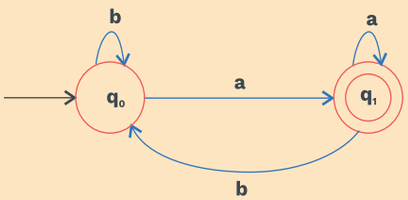


Fig. 2.2

- DFA is a finite Automata in which, from every state on every input symbol, exactly one transition should exist.
- DFA cannot use empty string transition.
- In DFA, if any string terminates in a state which is different from accepting state, then DFA rejects that string.
- All DFA are NFA.

SOLVED EXAMPLES

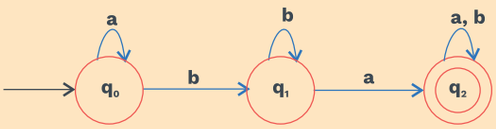
Q1



Language accepted by above Finite Automata?

Sol: L = {a, aa, aaa, aaaa, ba, bbbba, bababaaaa,.....} = Set of all strings ends with 'a' over $\sum$ = {a, b}

Q2



Language accepted by above Finite Automata?

Sol: L = {ba, bab, baa, aba, abaa, abab, ...} = Set of all strings containing "ba" as a substring over $\sum$ = {a, b}.

Note:

- **When a string is accepted by DFA?**

  Upon Scanning the string if we reach final state from initial state, then a string is accepted by DFA

- **When a language is accepted by DFA?**

  A language is accepted by DFA

i) When all strings present in the language are accepted by DFA

AND

ii) When all strings not present in the language are rejected by DFA.

Rack Your Brain

How many number of final state require to accept & in minimal finite automata?

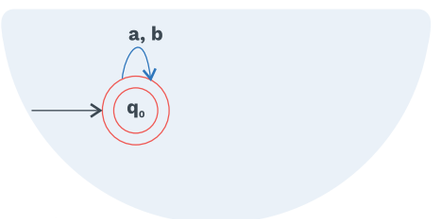
Minimal DFA: When all the states in a DFA are distinguishable, i.e. no two states in a final DFA are equivalent. The DFA is known as minimal DFA.

SOLVED EXAMPLES

Q1 **Construct minimal DFA that accepts all strings of 'a's and b's including ∈.**

Sol: L = {∈, a, b, aa, bb, ab, ba, aaa, ...}

DFA:



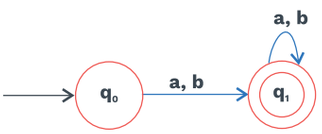
Transition table:

δ	a	b
→ q_0	q_0	q_0

Q2 **Construct minimal DFA that accepts all strings of 'a's and b's excluding ∈.**

Sol: L = {a, b, ab, ba, aa, bb, ...}

DFA:



Transition table:

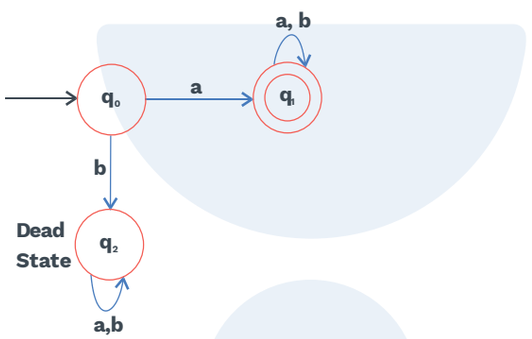
	a	b
→ q_0	q_1	q_1
q_1	q_1	q_1

Q3 **Construct minimal DFA that accepts all strings of 'a's & b's where every string "starts with a".**

Sol:

$$L = \{a, aa, aaa, aaaa, \ldots\ldots\ldots\ldots, ab, aab, aba, abb, \vdots\}$$

DFA:



- If any string starts with 'a', then it will be accepted by the DFA.

Let's consider an example.
String w = aab
$\delta(q_0, a) = q_1$
$\delta(q_1, a) = q_1$
$\delta(q_1, b) = q_1$ (accepted by DFA)

- But if the string starts with 'b', it directly goes to the dead state and get rejected by the DFA.

Let's consider an example.
String w = ba
$\delta(q_0, b) = q_2$
$\delta(q_2, a) = q_2$ (Dead state)
Thus, string ba is rejected (not accepted by DFA).

Dead state: It is a rejecting state i.e a non final state from which there is no path to the final state.
Once we reach to dead state, there is no way to reach the final state.

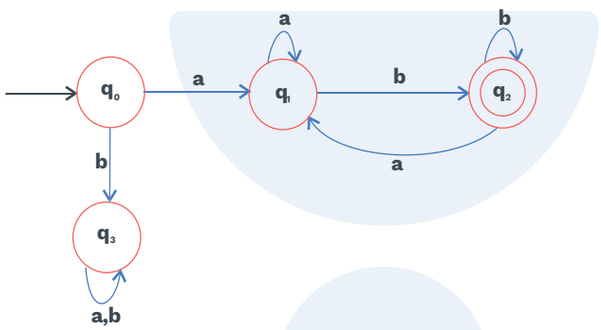
Transition table:

	a	b
→ q_0	q_1	q_2
(q_1)	q_1	q_1
q_2	q_2	q_2

Q4 **Construct a minimal DFA that accepts all the strings of a's and b's where "each string starting with a and ending with b".**

Sol: L = {ab, aab, abb, aaab, abab, aabb, ...}

DFA:



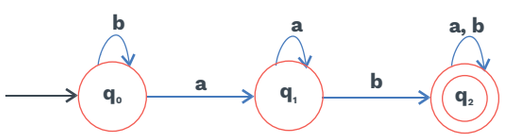
Transition table:

	a	b
→ q_0	q_1	q_3
q_1	q_1	q_2
(q_2)	q_1	q_2
q_3	q_3	q_3

Q5 **Construct a minimal DFA that accepts all the strings of a's and b's where "each string contains <u>ab</u> as substring".**

Sol: L = {ab, aba, aab, ...}

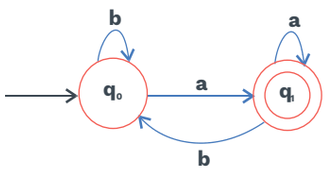
DFA:



Q6 **Construct a minimal DFA that accepts all the strings of a's and b's where "each string ends with a".**

Sol: L = {a, aa, ba, aaa, aba, ...}

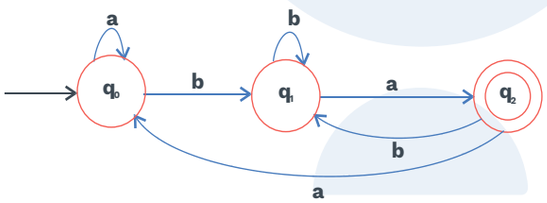
DFA:



Q7 **Construct a minimal DFA that accepts all the strings over $\sum$ = {a, b} where "each string ends with ba".**

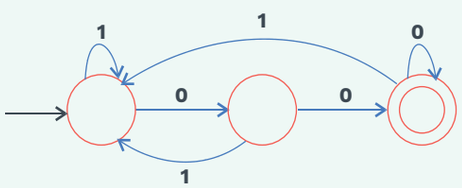
Sol: L = {ba, aba, bba, aaba, ...}

DFA:



Previous Years' Question

(GATE 2009)



The above DFA accepts the set of all strings over {0, 1} that

1) Begin either with 0 or 1 **2)** End with 0

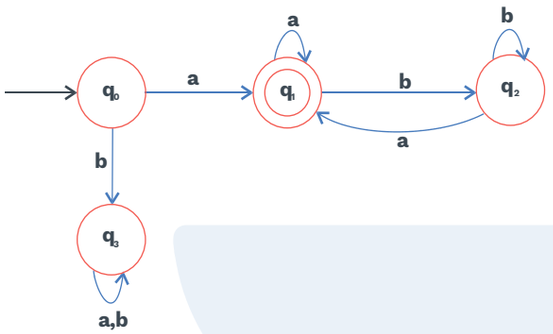
3) End with 00 **4)** Contain the substring 00

Sol: 3)

Q8 **Construct a minimal DFA that accepts all strings over $\sum$ = {a, b} where "each string starting and ending with a".**

Sol: L = {a, aa, aba, aaa, ...}

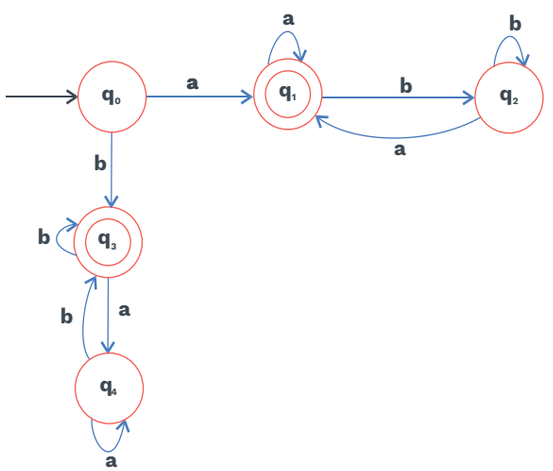
DFA:



Q9 **Construct a minimal DFA that accepts all the strings over $\sum$ = {a, b} where "each string starting and ending with same symbol".**

Sol: L = {a, b, aaa, bbb, aba, bab, ...}

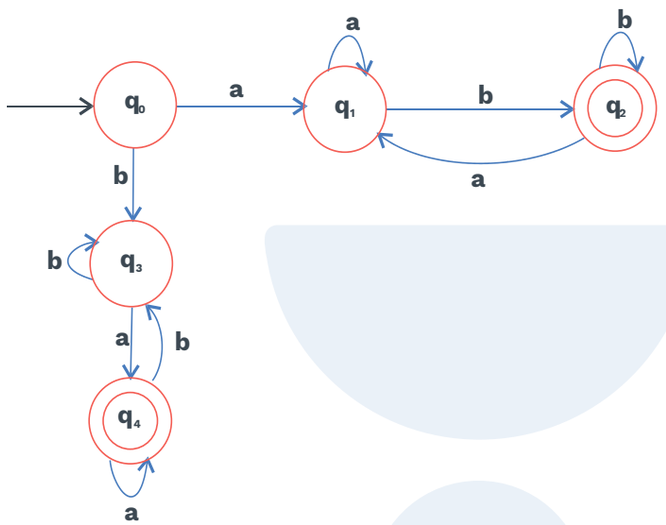
DFA:



Q10 **Construct a minimal DFA that accepts all the strings over ∑ = {a, b} where "each string starting and ending with different symbol".**

Sol: L = {ab, ba, aab, abb, baa, ...}

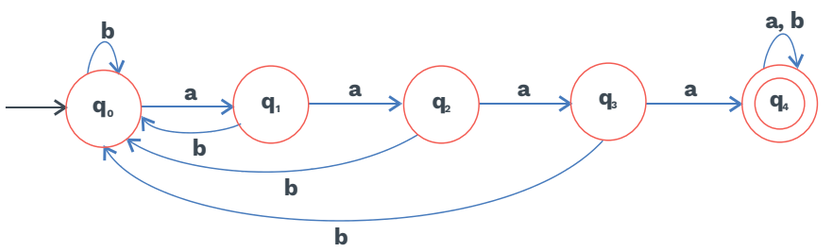
DFA:



Q11 **Construct a minimal DFA that accepts all the strings over ∑ = {a, b} where "each string contains aaaa as substrings".**

Sol: L = {aaaa, baaaa, aaaab, aaaaa, ...}

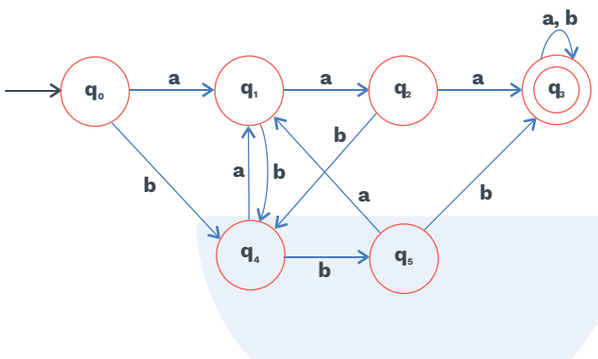
DFA:



Q12 **Construct a minimal DFA that accepts all the strings over $\sum$ = {a, b} where "each string contains aaa (or) bbb as substrings".**

Sol: L = {aaa, baaa, abbb, bbba, ...}

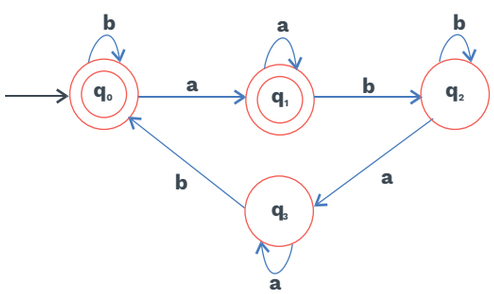
DFA:



Q13 **Construct a minimal DFA that accepts all the strings over $\sum$ = {a, b} where "number of occurrence of substring "ab" is even".**

Sol: L = $\{\in, abab, ababab, ...aa, bb, aaa, ...\}$

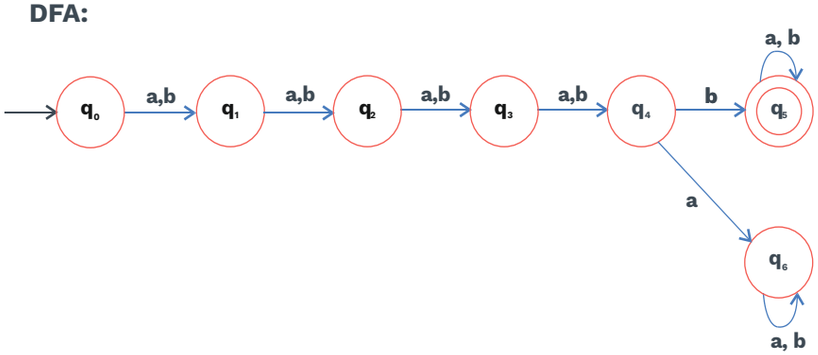
DFA:



Q14 **Construct a minimum DFA that accepts all the strings over $\sum$ = {a, b} where 5th symbol from left is 'b'.**

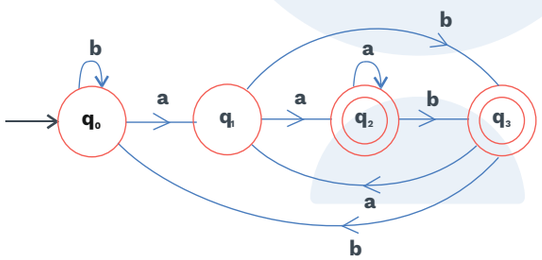
Sol: L = {aaaab, baaab, ...}

DFA:



Q15 **Construct a minimum DFA that accepts all the strings over ∑ = {a, b} where 2nd symbol from RHS is 'a'.**

Sol:



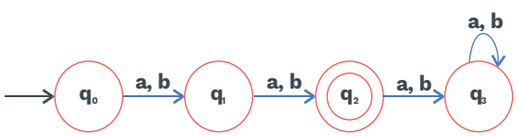
Q16 **Construct a minimum DFA that accepts all the strings over ∑ = {a, b} such that**

i) Length of string is exactly 2 i.e. |w| = 2.

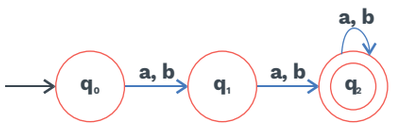
ii) Length of string is atleast 2 i.e. |w| ≥ 2

iii) Length of string is atmost 2 i.e. |w| ≤ 2

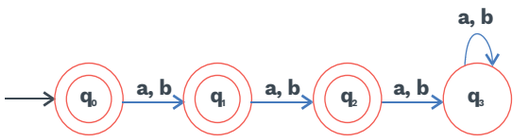
Sol: **i)** Length of the string is exactly 2 i.e. |w| = 2



ii) Length of the string is atleast 2 i.e. $|w| \geq 2$



iii) Length of the string is at most 2, i.e. $|w| \leq 2$



Note:

- If length of strings is exactly n i.e. $|w| = n$ over $\Sigma = \{a, b\}$, then number of states to construct minimum DFA = (n + 2) states.
- If length of strings is atleast n i.e. $|w| \geq n$ over $\Sigma = \{a, b\}$, then number of states to construct minimum DFA = (n + 1) states.
- If length of strings is atmost n, i.e. $|w| \leq n$ over $\Sigma = \{a, b\}$, then number of states to construct minimum DFA = (n + 2) states.

Rack Your Brain

If L is a non-empty language such that any win L has length atleast n, then any DFA accepting L must have atleast n +1 states. this statement true (or) false?

Q17 **Construct a minimal DFA which accepts all the strings over $\Sigma = \{a,b\}$ such that**

i) Length of each string is even

ii) Length of each string is odd

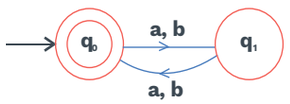
iii) Length of each string is divisible by 3

Sol: **i)** All even length strings over $\Sigma = \{a, b\}$

$|w| \bmod 2 = 0$

$L = \{\in, aa, bb, ab, \ldots\ldots\ldots\ldots\}$

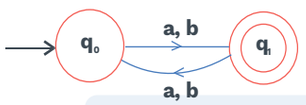
DFA:



ii) All odd length strings over $\sum = \{a, b\}$

Odd length strings meSol: |w| mod 2 = 1

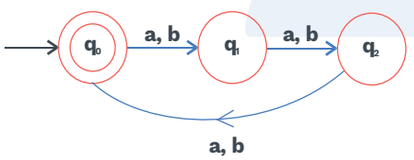
DFA:



iii) All strings whose length is divisible by 3 over $\sum = \{a, b\}$

|w| mod 3 = 0

$$L = \left\{ \begin{array}{ll} \in, & aaa, aaaaaa, ... \\ & aab \quad \vdots \\ & \vdots \quad \vdots \\ & bbb \quad \vdots \end{array} \right\}$$



Note:

If |w| mod n = m, then in minimum DFA, number of states = n and final state is q_m [th] states.

Q18 **Construct a minimal DFA that accepts all the strings over $\sum$ = {a, b} such that**

i) **$n_a(w) = 2$ i.e. number of a's in string = 2**

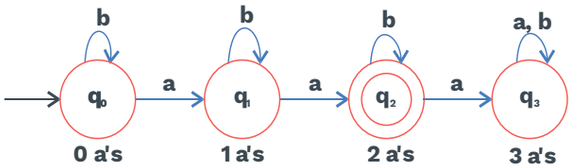
ii) **$n_a(w) \geq 2$ i.e. number of a's in string is atleast 2**

iii) **$n_a(w) \leq 2$ i.e. number of a's in string is atmost 2**

Sol:

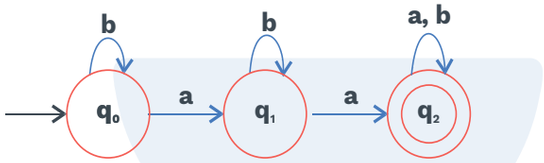
i) L = {aa, baa, aba, aab, bbaa,.....}

DFA:



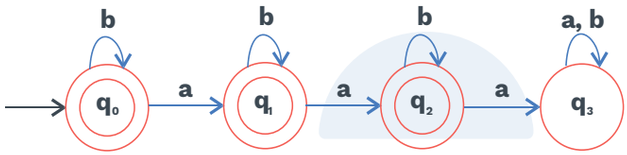
ii) L = {aa, aba, aaa, aaab,.....}

DFA:



iii) L = {∈, a, ab, aa, aba,.....}

DFA:



Note:

The minimal deterministic finite automata that accepts all the strings of a's and b's where each string contains:

- Exactly n number of a's has 'n+2' states.
- Atleast n number of a's has 'n+1' states.
- Atmost n number of a's has 'n+2' states.

Grey Matter Alert!

Number of states in minimal DFA which accepts all the strings over $\Sigma = \{a, b\}$ such that:

a) Number of a's is 'atleast m' and number of b's is 'atleast n' = (m + 1)(n + 1)

b) Number of a's is 'exactly m' and number of b's is 'exactly n' = (m + 1) (n + 1) + 1 (because of trap state, we require one more state).

c) Number of a's is 'atmost m' and number of b's is 'atmost n' = (m + 1)(n+1) + 1

Previous Years' Question

The minimum possible number of states of a deterministic finite automaton that accepts the regular language

$L = \{w_1 a w_2 \mid w_1, w_2 \in \{a, b\}^*, \mid w_1 \mid = 2, \mid w_2 \mid \geq 3\}$ is _____

Sol: Range: 8 to 8 **(GATE 2017 (Set-2))**

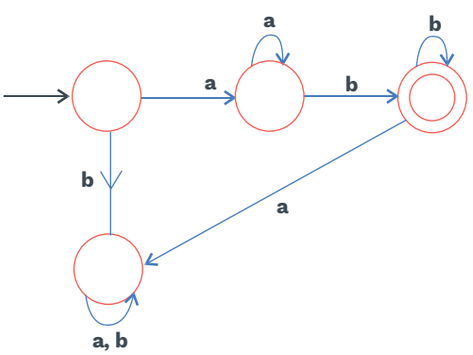
SOLVED EXAMPLES

Q19 **Construct a minimal DFA which accepts $L = \{a^n b^m \mid n, m \geq 1\}$**

Sol: $L = \{a^n b^m \mid n, m \geq 1\}$

$= \{ab, aab, abb, aabb, \ldots\ldots\ldots\ldots\}$

DFA:

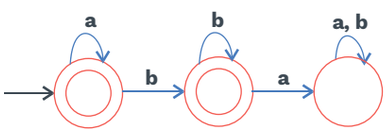


Q20 **Construct a minimal DFA which accepts L = {$a^n b^m$ | n, m ≥ 0}**

Sol: $L = \{a^n b^m \mid n, m \geq 0\}$

$= \{\in, a, b, ab, aa, bb, ...\}$

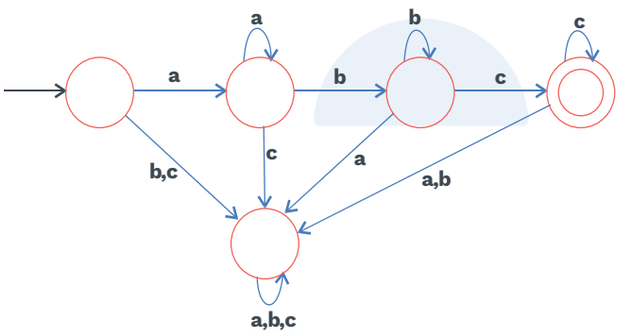
DFA:



Q21 **Construct a minimal DFA which accepts L = {$a^n b^m c^l$ | n, m, l ≥ 1}**

Sol: $L = \{a^n b^m c^l \mid n, m, l \geq 1\}$

$= \{abc, aabc, abbc, abcc, ...\}$



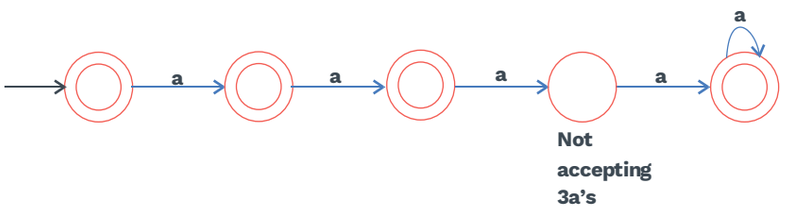
Q22 **Construct a minimal DFA over {a} :**

i) L = {a^n | n ≥ 0, n ≠ 3}

ii) L = {a^n | n ≥ 0, n ≠ 2, n ≠ 4}

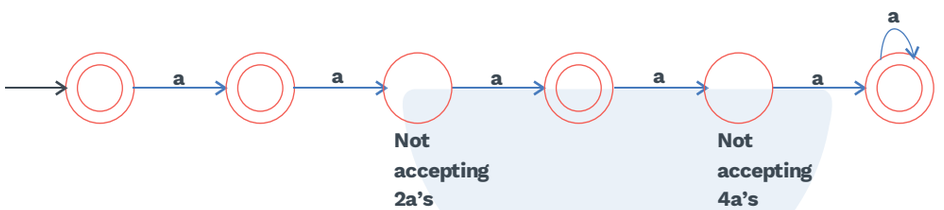
Sol: **i)** $L = \{a^n \mid n \geq 0, n \neq 3\}$

$= \{\in, a, aa, aaaa, ...\}$



ii) $L = \{a^n \mid n \geq 0, n \neq 2, n \neq 4\}$

$= \{\in, a, aaa, aaaaa, ...\}$



Rack Your Brain

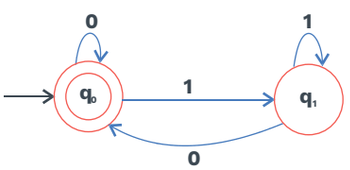
Construct the minimal DFA's for the language L = (ab | n20) u (b'a | n21}

SOLVED EXAMPLES

Q1 **Construct minimal state DFA that accept set of all binary numbers whose decimal equivalent is divisible by 2.**

Sol: Divisible by 2, so two remainders are 0, 1

DFA:



$(101)_2 = (5)_{10}$ = Not accepted by DFA:

$(110)_2 = (6)_{10}$ = Accepted by DFA

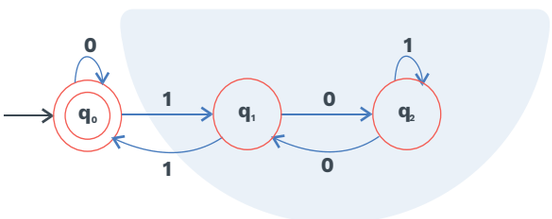
Transition table:

	0	1
→ q_0	q_0	q_1
q_1	q_0	q_1

Q2 **Construct minimal state DFA that accepts set of all binary numbers whose decimal equivalent is divisible by 3.**

Sol: Divisible by 3, so remainders are 0, 1, 2

DFA:



Transition table:

	0	1
→ q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Q3 **Construct minimal DFA that accepts set of all binary numbers whose decimal equivalent is divisible by 4.**

Sol: Divisible by 4, so remainders are 0, 1, 2, 3

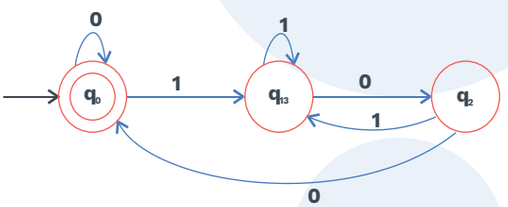
Transition table:

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

Here, we can merge q_1 and q_3 states as one state.

δ	0	1
→ q_0	q_0	q_{13}
q_{13}	q_2	q_{13}
q_2	q_0	q_{13}

DFA:



Q4 **Number of states in minimal DFA that accepts all binary strings which are congruent to 2 mod 7 i.e. binary number ≡ 2 (mod 7).**

Sol: **Transition table:**

	0	1
→ q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_5
q_3	q_6	q_0
q_4	q_1	q_2
q_5	q_3	q_4
q_6	q_5	q_6

⇒ Here two rows cannot be merged. Since all are different.

So, the minimum DFA will contain 7 states.

Q5 **Number of states in minimal DFA that accepts set of all binary strings whose decimal equivalent is divisible by 6.**

Sol: This problem belongs to below NOTE (Case 3).
Here n = 6 (even) but not in power of 2.
$n = 2^1 * 3$
So, minimal DFA contains (1 + 3) = 4 states
We can also prove it in descriptive way.
Transition table:

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

⇒ Here, we can merge states q_1 and q_4 as q_{14}.
Similarly, we can merge states q_2 and q_5 as q_{25}.
Hence, transition Table will become:

	0	1
→ q_0	q_0	q_{14}
q_{14}	q_{25}	q_3
q_{25}	q_{14}	q_{25}
q_3	q_0	q_{14}

Note:

To construct minimal DFA, which accepts all binary numbers such that Binary number ≅ r (mod n), where r is remainder.

- **Case 1**

  If n is odd, then Minimal DFA will contain 'n' states.

- **Case 2**

  If n = even & if we can express n such that $n = 2^k$, then minimal DFA will contain '(k + 1) states'.

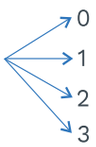
- **Case 3**

  If n = even, but $n \neq 2^k$, then if we can express n such that n = (2k) * p, where p is prime factor of n. In this case, minimal DFA contains '(k + p) states'.

SOLVED EXAMPLES

Q1 **Construct a minimal DFA that accepts all ternary numbers which are divisible by 4.**

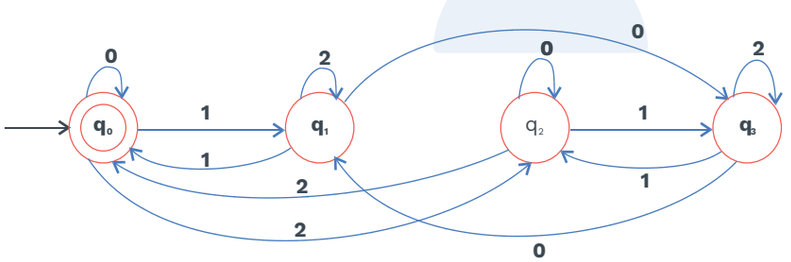
Sol: Divisible by 4, so remainders are



Input = {0, 1, 2}

Transition table:

	0	1	2
→ q_0	q_0	q_1	q_2
q_1	q_3	q_0	q_1
q_2	q_2	q_3	q_0
q_3	q_1	q_2	q_3



Operations on DFA:

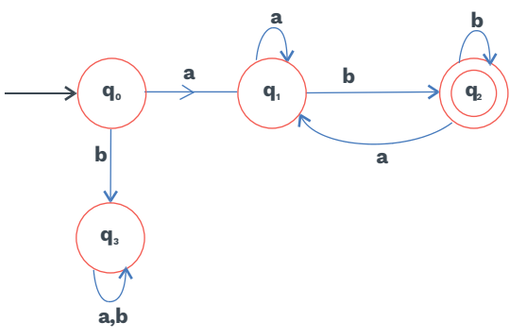
Union:

DFA which accepts all strings starting and ending with different symbols over $\Sigma = \{a, b\}$.

L_1 = {ab, aab, abb, ...}

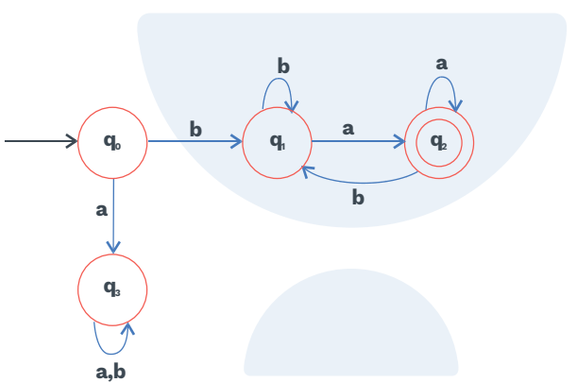
Starts with 'a' and ends with 'b'.

DFA:

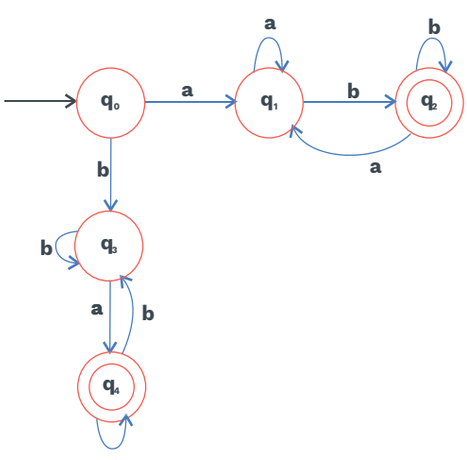


L_2 = {ba, baa, bba, ...}
= Starts with b and ends with a

DFA:



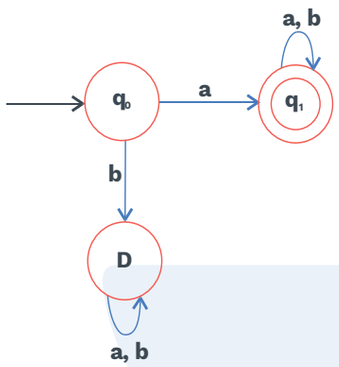
DFA for $L_1 \cup L_2$:



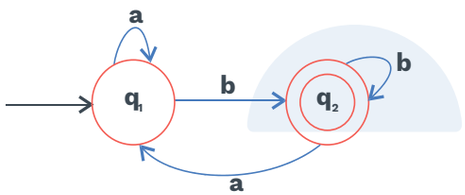
Concatenation:

DFA that accepts all strings over $\Sigma = \{a, b\}$ which starts with a and ends with b.

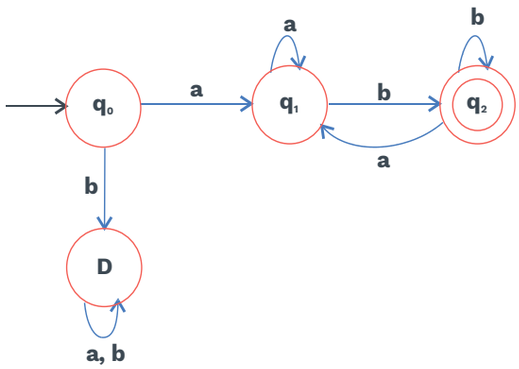
L_1 = {Starting with a}
= {a, aa, ab, aab, ...}



L_2 = {Ending with b}
= {b, ab, bb, aab, ...}



Now DFA for $L_1.L_2$:



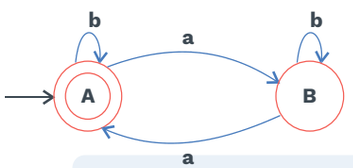
Cross product:

SOLVED EXAMPLES

Q1 **Construct a minimal DFA which accepts set of all strings over $\sum$ = {a, b} such that string of the language contain even number of a's and even number of b's.**

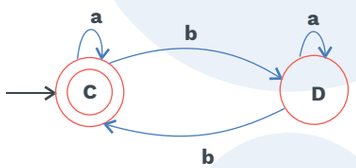
Sol: L_1 = {even number of a's} = {∈, aa, baa, ...}

DFA:



L_2 = {even number of b's} = {∈, bb, abb, ...}

DFA:



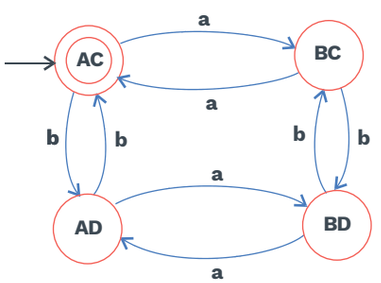
Cross product:

{A, B} × {C, D} = {AC, BC, AD, BD}

Where final state = {AC} because A and C are both final states in respective DFA for L_1 and L_2.

Initial State = {AC} because A and C are both initial states in respective DFA for L_1 and L_2.

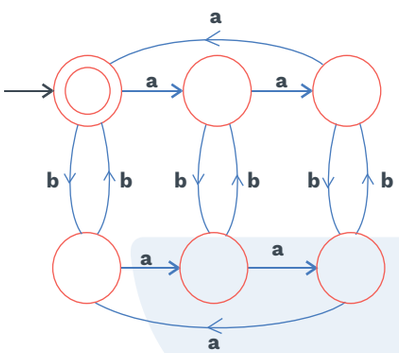
DFA for cross product:



Q2 **Construct a minimal DFA which accepts set of all strings over $\sum$ = {a, b} in which number of a's are divisible by 3 and number of b's are divisible by 2.**

Sol: $n_a(w) \cong 0 \bmod 3$ & $n_b(w) \cong 0 \bmod 2$

DFA:



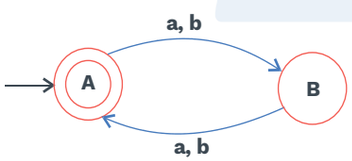
Total number of states = 3 × 2 = 6

Complementation:

eg 1: L = set of all strings over $\sum$ = {a, b} of even length

$= \{\in, ab, aa, bb, aaaa, \ldots\ldots\ldots\ldots\ldots\ldots\}$

Sol: **DFA**:



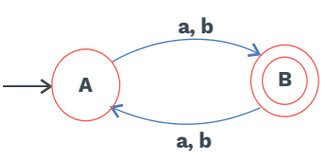
Now, L_2 = set of all strings over $\sum = \{a, b\}$ of odd length

= {a, b, aaa, aab, ...}

$$L_2 = \overline{L}_1 = \text{Complement of } L_1$$

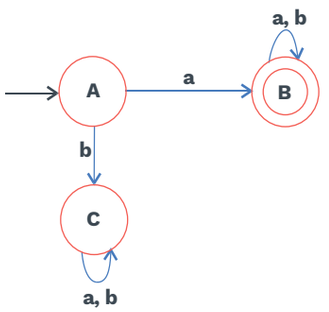
DFA for L_2:

Just complement the above DFA i.e. change the final state as Non-final state and Non-final state as Final State



eg 2: L = Set of all strings over $\sum$ = {a, b} starting with 'a'
= {a, aa, ab, ...}

Sol: **DFA**:

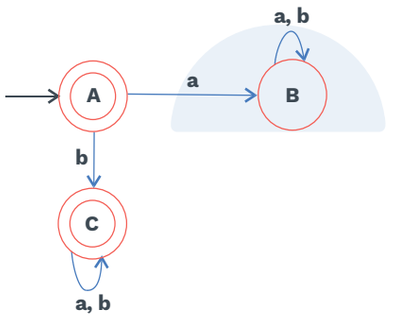


$L_2 = \overline{L}_1$ = Complement of L_1

= Set of all strings over $\sum$ = {a,b} not starting with 'a'

= {$\in$, b, ba, ...}

DFA for complement of L_1:



Note:

If a DFA is defined by (Q, Σ , δ, q_0, F) and it accepts the language L_1. Then, the DFA which accepts complement of language L_1, that is $L_2 = L_1$, will be defined as (Q, Σ , δ, q_0, Q – F).

DFA $\longrightarrow$ (Q, Σ, δ, q_0, F)

Complement

(Q, Σ, δ, q_0, Q–F)

Previous Years' Question

Let L be the language represented by the regular expression $\Sigma^*0011\Sigma^*$ where $\Sigma = \{0,1\}$

What is the minimum number of states in a DFA that recognizes $\overline{L}$ (complement of L)? **(GATE 2015 SET 3)**

1) 4 **2) 5**
3) 6 **4) 8**

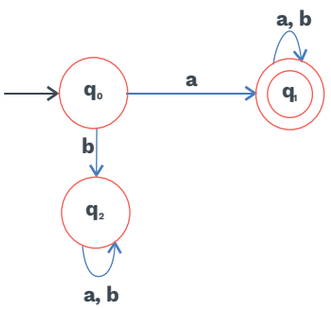
Sol: 2)

Reversal:
Reversing a language meSol: reverse all strings present in the language.
Steps:
a) Draw the state as it is.
b) Make final state as initial state and initial state as final state.
c) Reverse the edges.
d) Remove all the unreachable state and its transition from initial state.

Note:
Not every reversal of DFA Result into DFA.

DFA for the set of string over $\Sigma = \{a,b\}$ which start with 'a'.

L_1 = {a, aa, ab, aab,.....}
DFA:



Now, Let L_2 = Reversal of L_1 = {a, aa, ba, baa, ...}

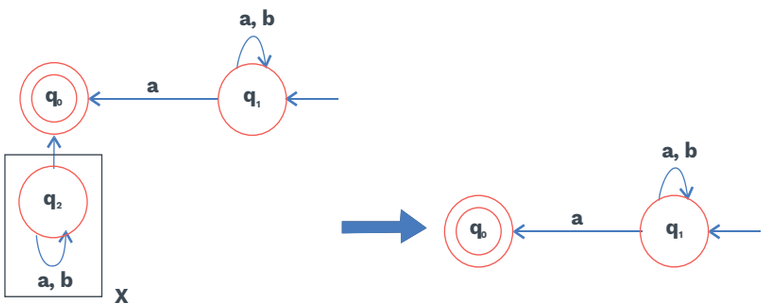


Fig 2.3

Previous Years' Question

Consider the following language: **(GATE-2020)**

L = {x ∈ {a, b}*| number of a's in x divisible by 2 but not divisible by 3}

The minimum number of states in DFA that accepts L is _____.

Sol: Range = 6 to 6

Q1 **Counting DFA:**

Number of 2-state DFA with designated initial state that can be constructed over $\Sigma = \{a, b\}$.

Sol:

- Final State can be : Zero state i.e. None of them final
  : One state i.e. either q_0 final state or q_1 final state
  : Two states i.e. q_0 and q_1 both final state

So, Total $2^2 = 4$ possibility.

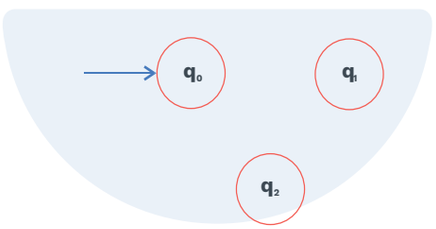
δ	a	b
→ q_0	q_0/q_1	q_0/q_1
q_1	q_0/q_1	q_0/q_1

2^4 possibility
(i.e. either go to the state q_0 or q_1)

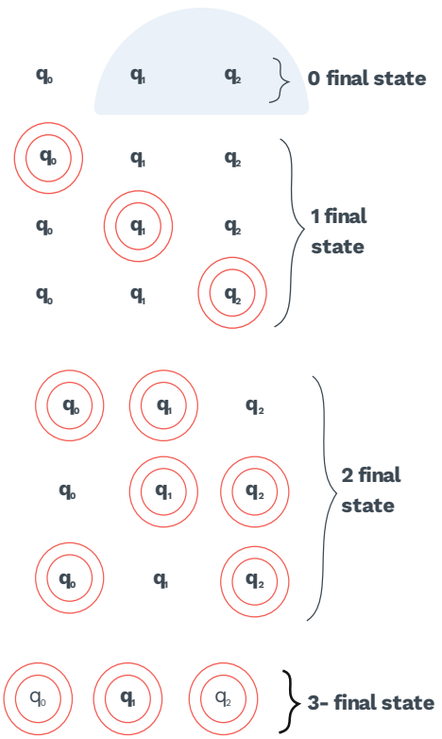
- q_0 on 'a' can either go to the state q_0 or q_1. q_0 on 'b' can either go to the state q_0 or q_1.
- Similarly, q_1 on 'a' can either go to the state q_0 or q_1. q_1 on 'b' can either go to the state q_0 or q_1.
  Total number of ways to construct 2-state DFA with designated initial state over $\sum = \{a, b\}$ = $2^2 * 2^4 = 2^6 = 64$

Q2 **Number of 3-state DFA with designated initial state that can be constructed over $\sum = \{a, b\}$.**

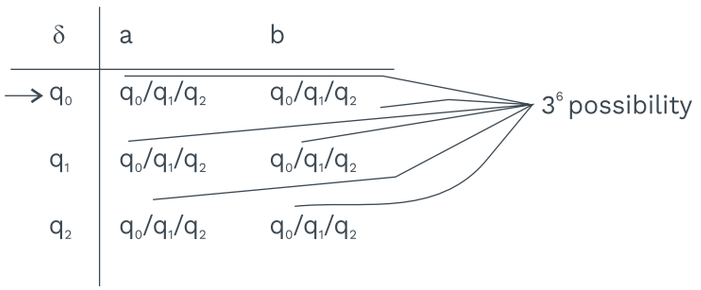
Sol:



- Final state can be:



Total possibility = 2^3 = 8 possibility



Total number of ways to construct 3-state DFA with a designated initial state over $\sum = \{a, b\}$ = $2^3 * 3^6 = 8 * 9^3 = 5832$

Note:
Number of n-state DFA with a designated initial state over alphabet containing m symbols.

Given: $|\sum| = m$

|Q| = number of states = n

- Number of ways, we can construct final states = 2^n

δ	1	2	3.....m
1	n	n	n .....n
2	n	n	n .....n
3	n	n	n .....n
.	.	.	. .....
.	.	.	. .....
.	.	.	. .....
n	n	n	n .....n

Total cells in $\delta = n \times m$
$= m \times n$
and each cell can be filled in 'n ways'
So, Total possibility = $n^{m \times n}$

$$Q \times \sum \rightarrow 2^Q$$

Total number of ways to construct n-state DFA with a designated initial state over

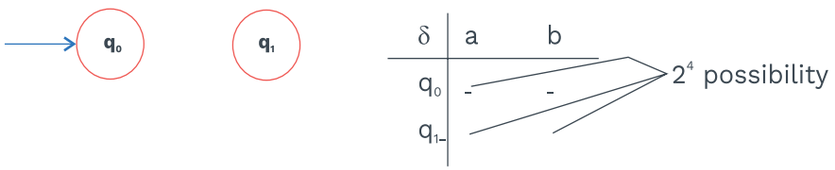
$\sum$ = {1, 2, ...m} containing m symbols

= $2^n \times n^{m \times n}$

Q3 **Number of 2-state DFA's with a designated initial state accepting ϕ over $\Sigma = \{a, b\}$**

Sol:

Zero final state



Since neither q_0 nor q_1 is final state. So, it accepts ϕ in 16 ways.

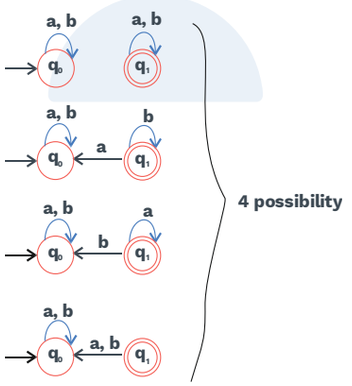
Case 2: One final states

1) 

It accepts $\in$. Thus, this cannot be the required case.

2) 

It can be the desired case but not all transition on it is desirable.



Case 3: Two final states

It cannot be the described case as it is accepting $\in$.

$\therefore$ Overall total number of 2-state DFAs with designated initial state accepting ϕ.

Over $\Sigma = \{a, b\}$ = (16 + 4) = 20

Q4 **Number of 2-state DFA with a designated initial state accepting $\sum^*$ (universal language) over $\sum = \{a, b\}$.**

Sol:

Case 1: Zero final state

It is not a favourable case as it accepts ϕ. It cannot accept $\sum^*$.

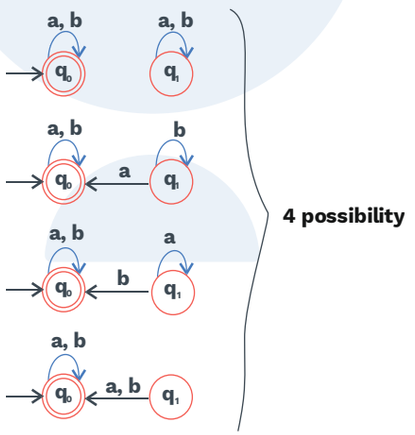
Case 2: One final state

1) 

It will not accept $\in$ which is part of $\sum^*$. Thus, it cannot accept $\sum^*$. Not favourable case.

2) 

It can be favourable case.



4 possibility

Case 3: 2 Final state

δ	a	b
→ q_0	q_0/q_1	q_0/q_1
q_1	q_0/q_1	q_0/q_1

2^4 possibility

When both q_0 and q_1 states are final states, then it will accept $\sum^*$ in 16 possible ways.

$\therefore$ Overall, total number of 2–state DFA with designated initial state accepting $\sum^*$ over $\sum = \{a, b\}$ = 4 + 16 = 20

Minimization of DFA:

Definitions

Minimization of DFA meSol: trSol:forming a given DFA into an equivalent DFA that has a minimum number of states.

i) Remove all the unreachable states (i.e. states which are not reachable from initial state).

ii) Separate all final states in one set and all non-final states in another set.

iii) Two states p and q of a DFA are called Indistinguishable if $\delta^*(p, w) \in$ Final state

$\Rightarrow \delta^*(q, w) \in$ Final State

and $\delta^*(p, w) \notin$ Final State $\Rightarrow \delta^*(q, w) \notin$ Final State

for all $w \in \Sigma^*$

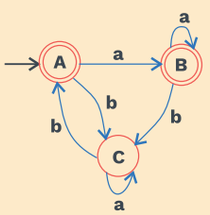
iv) On the other hand, if there exists some string $w \in \Sigma^*$ such that $\delta^*(p, w) \in$ Final State and $\delta^*(q, w) \notin$ Final State or vice-versa, then the states p and q are said to be distinguishable by a string 'w'.

v) Identify distinguishable states and use the logic known as seperating the states. It meSol: in each step if the two states in a set are distinguishable seperate them.

vi) Stop the algorithm once find no change in partition.

SOLVED EXAMPLES

Q1 **Minimize the following DFA:**



Sol:

δ	a	b
→ A	B	C
B	B	C
C	C	A

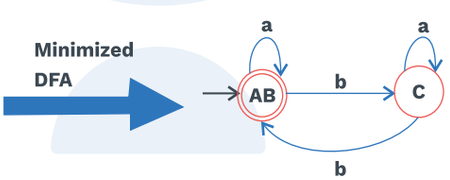
State equivalence method:

Follow steps i) and step ii), separate all final states in one set and all non final states in another set.

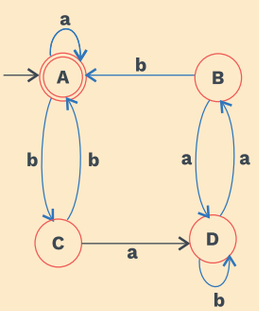
$\pi_0 = \{(A,B),(C)\}$ [0 equivalence class]

Let group 1 i.e. g_1 = (A, B) and
group 2 i.e. g_2 = (C)
Then, A on 'a' goes to the group g_1
And B on 'a' goes to the group g_1.
δ (A, a) = B (belongs to g_1)
δ (B, a) = B (belongs to g_1)
A on 'b' goes to the group g_2
B on 'b' goes to the group g_2
δ (A, b) = C (belongs to g_2)
δ (B, b) = C (belongs to g_2)
So, A and B will be in same group(set).
$\pi_1 = \pi_0$ [1 equivalence class]

δ	a	b
→ (AB)	AB	C
C	C	AB



Q2 **Minimize the following DFA:**



Sol:

δ	a	b
→ A	A	C
B	D	A
C	D	A
D	B	D

State equivalence method:

For State B and C:

$\pi_0 = \{(A), (B, C, D)\}$

δ	a	b
B	g_3	g_1
C	g_3	g_1

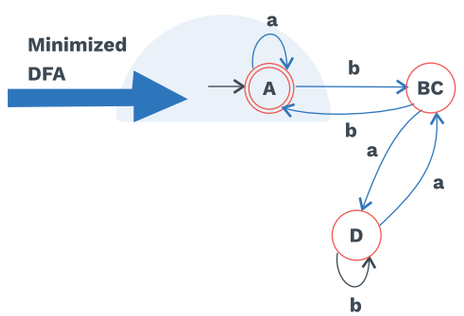
So, $\boxed{B = C}$

$\pi_1 = \begin{Bmatrix} (A), (B, C), (D) \\ g_1 \quad g_2 \quad g_3 \end{Bmatrix}$

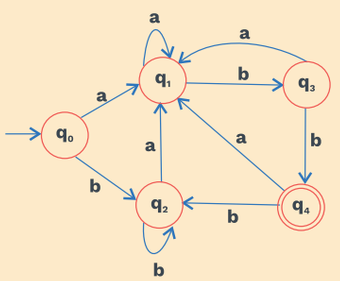
$\pi_2 = \pi_1$

δ	a	b
→ A	A	BC
BC	D	A
D	BC	D

Minimized DFA



Q3 **Minimize the following DFA:**



Sol:

δ	a	b
→ q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
(q_4)	q_1	q_2

By state equivalence method:

$$\pi_0 = \{(q_0, q_1, q_2, q_3), (q_4)\}$$
$$(g_1) \qquad (g_2)$$

For state q_0, q_1, q_2 and q_3

δ	a	b
q_0	g_1	g_1
q_1	g_1	g_1
q_2	g_1	g_1
q_3	g_1	g_2

$$\pi_1 = \{(q_0, q_1, q_2), (q_3), (q_4)\}$$
$$(g_1) \qquad (g_2) \quad (g_3)$$

For State q_0 , q_1 and q_2

δ	a	b
q_0	g_1	g_1
q_1	g_1	g_2
q_2	g_1	g_1

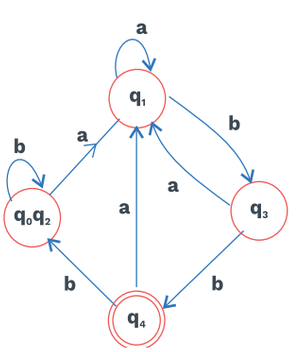
$$\pi_2 = \{(q_0, q_2), (q_1), (q_3), (q_4)\}$$

So, $q_0 = q_2$

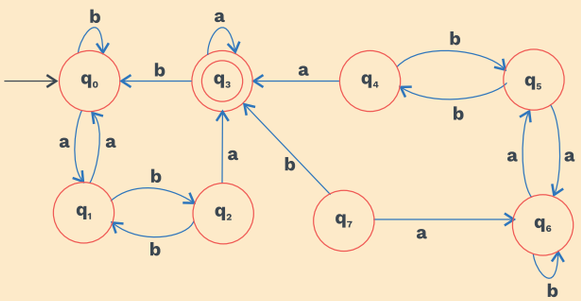
$\pi_3 = \pi_2$

δ	a	b
→ q_0q_2	q_1	q_0q_2
q_1	q_1	q_3
q_3	q_1	q_4
(q_4)	q_1	q_0q_2

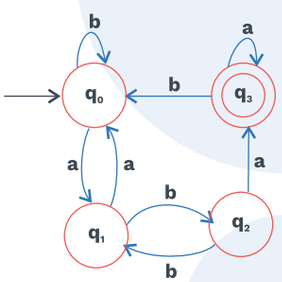
Minimized DFA



Q4 **Minimize the following DFA:**



Sol: Remove states q_4, q_5, q_6, q_7 as these are unreachable states.



δ	a	b
→ q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
(*q_3)	q_3	q_0

State equivalence method:

$$\Pi_0 = \begin{Bmatrix} (q_0, q_1, q_2), (q_3) \\ g_1 \qquad g_2 \end{Bmatrix}$$

For state q_0, q_1 and q_2

δ	a	b
q_0	g_1	g_1
q_1	g_1	g_1
q_2	g_2	g_1

$$\Pi_1 = \left\{ \begin{matrix} (q_0, q_1), (q_2), (q_3) \\ g_1 \quad g_2 \quad g_3 \end{matrix} \right\}$$

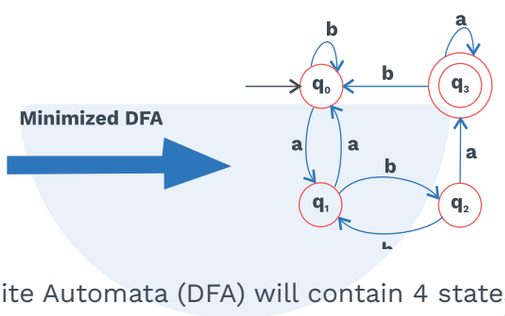
For state q_0 and q_1

δ	a	b
q_0	g_1	g_1
q_1	g_1	g_2

$$\Pi_2 = \{(q_0), (q_1), (q_2), (q_3)\} \Rightarrow \boxed{q_0 \neq q_1}$$

$$\Pi_3 = \Pi_2$$

δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
(q_3)	q_3	q_0



Minimized Deterministic Finite Automata (DFA) will contain 4 states.

Rack Your Brain

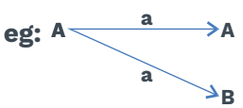
Given a regular language L, will its minimal DFA unique?

Non-Deterministic finite automata (NFA)

- In NFA, there can be 0 or 1 or more than one transition for any input symbol from any state.
- A NFA is defined as:

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

where δ: Transition function



i.e. state A on seeing input a, goes to more than one state which is A and B.
Let $Q = \{A, B\}$ and $\Sigma = \{a, b\}$

$$Q \times \Sigma \rightarrow 2^Q$$

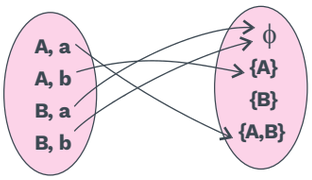


Fig. 2.4

Note:

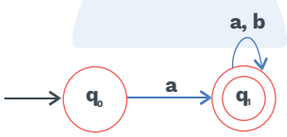
- In NFA, the transitions are not uniquely determined by their input symbol.
- In NFA, for any input symbol on a state can have many possible next states.
- NFA is easier to construct than DFA.

SOLVED EXAMPLES

Q1 **Construct NFA that accepts all strings of a's and b's where each string starts with 'a'.**

Sol: L = {a, aa, ab, ...}

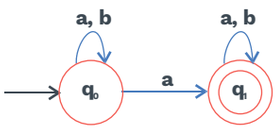
NFA:



Q2 **Construct NFA that accepts all strings of a's and b's where each string contains 'a'.**

Sol: L = {a, aa, ba, ab, ...}

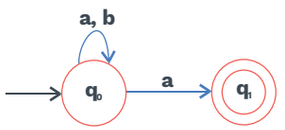
NFA:



Q3 **Construct NFA that accepts all strings of a's and b's where each string ending with 'a'.**

Sol: L = {a, aa, ba, aba, ...}

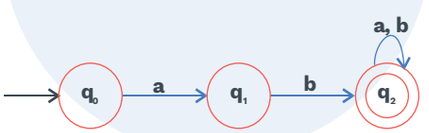
NFA:



Q4 **Construct NFA that accepts all strings of a's and b's where each string starts with 'ab'.**

Sol: L = {ab, aba, abb, ...}

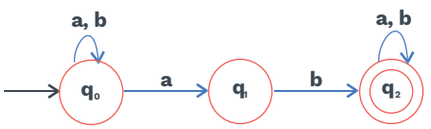
NFA:



Q5 **Construct NFA that accepts all strings of a's and b's where each string contains 'ab' as a substring.**

Sol: L={ab, aab, bab , aba ,.....}

NFA:



Q6 **Construct NFA that accepts all strings of a's and b's where each string starts with aabb.**

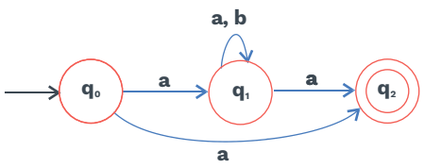
Sol: L = {aabb, aabba, aabbb, ...}

NFA:

Q7 **Construct NFA that accepts all strings of a's and b's where each string "starts and ends with a".**

Sol: L = {a, aa, aba, ...}

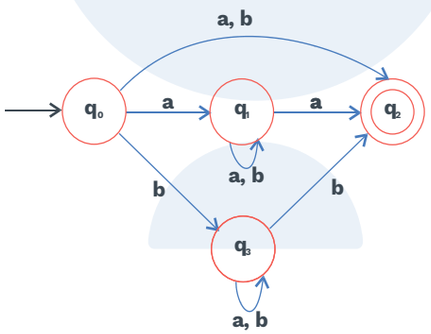
NFA:



Q8 **Construct NFA that accept all strings of a's and b's where each string "starting and ending with same symbol".**

Sol: L = {a, b, aa, bb, aba, bab, ...}

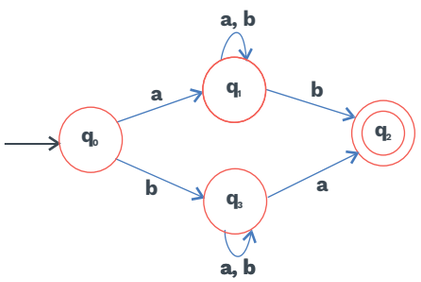
NFA:



Q9 **Construct NFA that accept all strings of a's and b's where each string "starting and ending with different symbol".**

Sol: L = {ab, ba, aab, abb, ...}

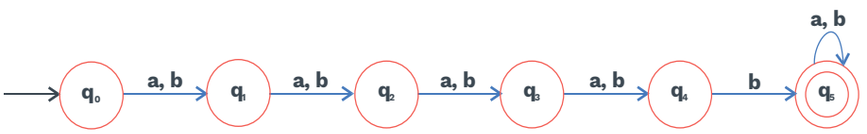
NFA:



Q10 **Construct NFA that accept all strings of a's and b's where 5th symbol is 'b' while reading the string from left to right.**

Sol: L = {aaaab, baaab, ...}

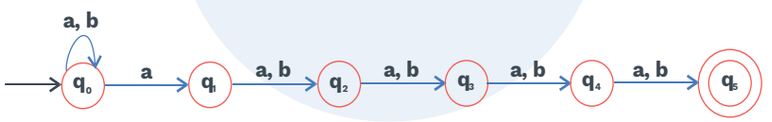
NFA:



Q11 **Construct NFA that accept all strings of a's and b's where 5th symbol is 'a' while reading string from right to left.**

Sol: L = {aaaaa, abbbb, ababb, baabab,.....}

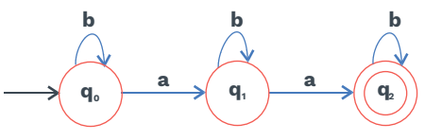
NFA:



Q12 **Construct NFA that accept all strings of a's and b's where each string contains exactly two a's.**

Sol: L = {aa, baa, aba, ...}

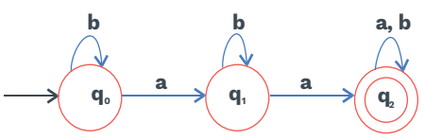
NFA:



Q13 **Construct NFA that accept all strings of a's and b's where each string contains atleast two a's.**

Sol: L = {aa, aaa, aab, baa, baaa,.....}

NFA:



Q14 Construct NFA that accept all strings of a's and b's where the length of string is exactly 4.

Sol: $L = \begin{Bmatrix} aaaa, aabb, aaba, abaa, \ldots\ldots\ldots\ldots \\ baaa, bbbb, \ldots\ldots\ldots\ldots\ldots\ldots \end{Bmatrix}$

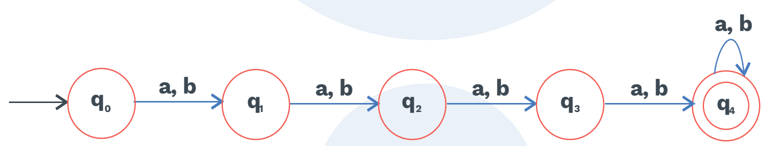
NFA:

Q15 Construct NFA that accept all strings of a's and b's where the length of string is atleast 4.

Sol: Length of string is atleast 4 meSol: $|w| \geq 4$.

$L = \begin{Bmatrix} aaaa, aaaab, \ldots\ldots\ldots\ldots\ldots\ldots \\ baaa, bbbbb, \ldots\ldots\ldots\ldots\ldots\ldots \end{Bmatrix}$

NFA:

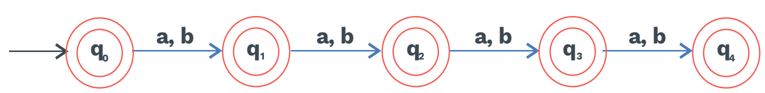


Q16 Construct NFA that accept all strings of a's and b's where the length of string is atmost 4.

Sol: Length of string is atmost 4 meSol: $|w| \leq 4$.

$L = \{\epsilon, a, b, aa, ab, ba, bb, \ldots\ldots\ldots\ldots\}$

NFA:



Note:

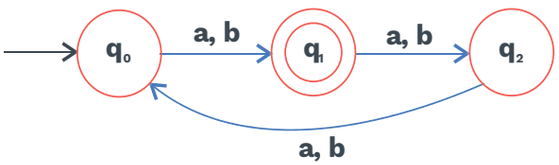
If the given NFA accept set of all the strings over $\Sigma = \{a, b\}$, where

Case I: Length of the string is exactly n
Case II: Length of the string is atleast n
Case III: Length of the string is atmost n
In all these cases, minimum number of states in NFA = (n+1) states

Q17 **Construct NFA that accept all strings of a's and b's such that |w| = 1 mod 3.**

Sol: L = {a, b, aaaa, abab, ...}

NFA:



Rack Your Brain

Draw NFA with 3 states that accepts the language L = {a" In 21}{ba Imzo,k 20}

Previous Years' Question

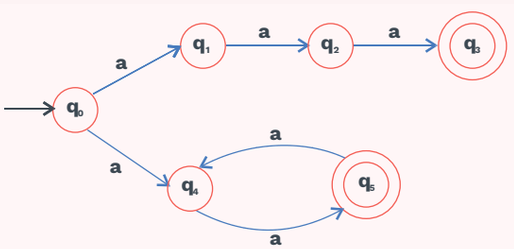
Let N be an NFA with n states. Let K be the number of states of minimal DFA which is equivalent to N. Which of the following is necessarily True? **(GATE 2018)**

1) $K \geq 2^n$ **2)** $K \geq n$

3) $K \leq n^2$ **4)** $K \leq 2^n$

Sol: 4)

Rack Your Brain



Let L be the language accepted by the above NFA. Construct an NFA that accepts LU {a").

2.2 COMPARISON OF DFA AND NFA

	Deterministic Finite Automata	Non-Deterministic Finite Automata
1) Definition	DFA is a type of finite Automata where for any input on a state, only one path is possible.	NFA is a type of Finite Automata where for any input on a state, many path can be possible.
2) Next State	In DFA, next possible state is unique.	In NFA, for any state and input pair, many next state can be possible.
3) Minimization	Minimization of DFA is possible	Minimization alogrithm is not applicable for NFA.
4) Construction	DFA is complex to construct and every DFA is a NFA.	NFA is easy to construct than DFA.

Table 2.1

Conversion from NFA to DFA:

Procedure:

Let $N = (Q, \Sigma, \delta, q_0, F) \rightarrow$ NFA

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

i) There is no change in the initial state i.e. $\boxed{q_0' = q_0}$

ii) Start the construction of δ' with initial state and continue it for every new state which will come under input column.
Terminate this process when no more new state appears in the input column.

iii) Every set which contains final state of NFA as subset, will be the final state for corresponding DFA.

Note:

DFA obtained from NFA need not be a minimal DFA.

Q1 **Construct an equivalent DFA for the NFA that accepts all strings of a's and b's where each strnig "starts with a".**

Sol: $L_1 = \{a, aa, ab, ...\}$

NFA:

Transition table for NFA:

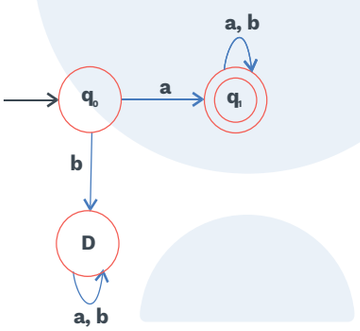
	a	b
→ q_0	q_1	ϕ
(*) q_1	q_1	q_1

Corresponding transition Table for DFA obtained from above table:

	a	b
→ q_0	q_1	D
(*) q_1	q_1	q_1
D	D	D

where D = Dead state (Introduced in DFA)

DFA:



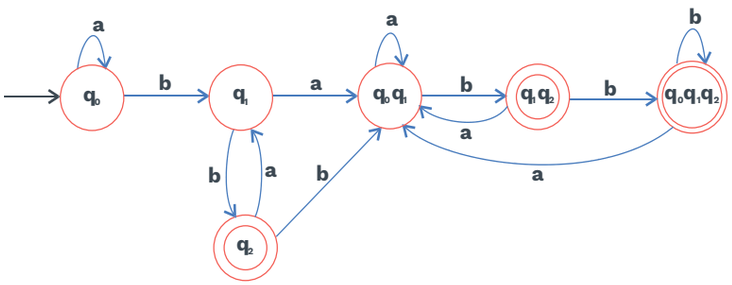
Q2 **Construct an equivalent DFA for the following NFA:**

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

Sol:

δ	a	b
→ q_0	$\{q_0\}$	$\{q_1\}$
q_1	$\{q_0q_1\}$	$\{q_2\}$
q_0q_1	$\{q_0q_1\}$	$\{q_1q_2\}$
(*) q_2	$\{q_1\}$	$\{q_0q_1\}$
(*) q_1q_2	$\{q_0q_1\}$	$\{q_0q_1q_2\}$
(*) $q_0q_1q_2$	$\{q_0q_1\}$	$\{q_0q_1q_2\}$

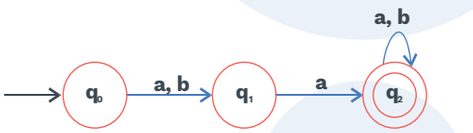
DFA diagram:



Q3 **Construct an equivalent DFA for the NFA that accepts all strings of a's and b's where 2nd symbol is 'a' while reading the string from left to right.**

Sol: $L = \{aa, ba, bab, aab, ...\}$

NFA:



Transition table for NFA:

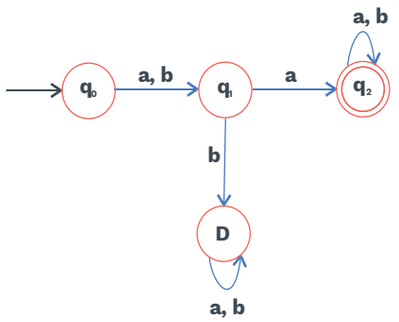
δ	a	b
→ q_0	q_1	q_1
q_1	q_2	ϕ
(*q_2)	q_2	q_2

Corresponding transition Table for DFA obtained from above NFA table:

δ	a	b
→ q_0	q_1	q_1
q_1	q_2	D
(*q_2)	q_2	q_2
D	D	D

Here 'D' is dead state.

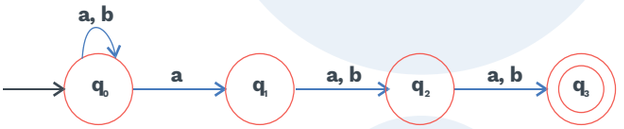
DFA diagram:



Q4 **Construct an equivalent DFA for the NFA that accepts all strings of a's and b's where 3rd symbol is 'a' while reading the string from right to left.**

Sol: $L = \{aaaa, aaab, baab, babb, ...\}$

NFA:

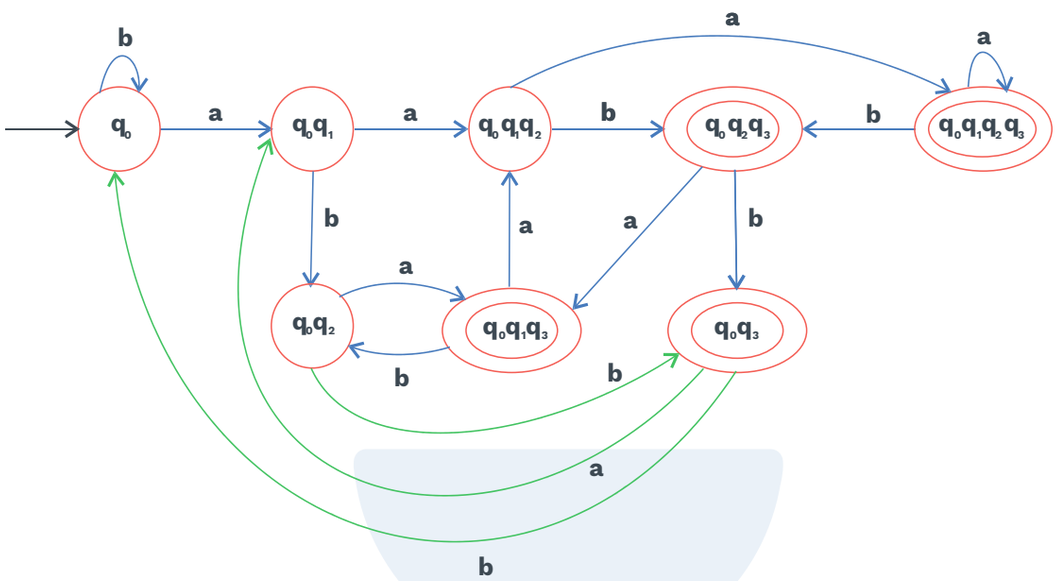


Transition table:

δ	a	b
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\{q_3\}$
(*) q_3	ϕ	ϕ

Corresponding transition table for DFA obtained from above table:

δ	**a**	**b**
→ q_0	$\{q_0q_1\}$	$\{q_0\}$
q_0q_1	$\{q_0q_1q_2\}$	$\{q_0q_2\}$
$q_0q_1q_2$	$\{q_0q_1q_2q_3\}$	$\{q_0q_2q_3\}$
q_0q_2	$\{q_0q_1q_3\}$	$\{q_0q_3\}$
(*) $q_0q_1q_2q_3$	$\{q_0q_1q_2q_3\}$	$\{q_0q_2q_3\}$
(*) $q_0q_2q_3$	$\{q_0q_1q_3\}$	$\{q_0q_3\}$
(*) $q_0q_1q_3$	$\{q_0q_1q_2\}$	$\{q_0q_2\}$
(*) q_0q_3	$\{q_0q_1\}$	$\{q_0\}$



Q5 **Construct the minimal DFA that accept all the strings of 0's and 1's where each string do not contain "101" as a substring.**

Sol: Since constructing NFA is easy, first Construct NFA where each string contain "101" as a substring

NFA:

	0	1
→ q_0	$\{q_0\}$	$\{q_0q_1\}$
q_1	$\{q_2\}$	ϕ
q_2	ϕ	$\{q_3\}$
(*) q_3	$\{q_3\}$	$\{q_3\}$

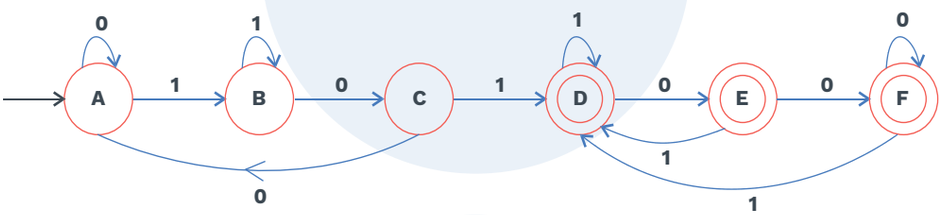
Conversion from NFA to DFA:

	0	1
→ q_0	$\{q_0\}$	$\{q_0q_1\}$
q_0q_1	$\{q_0q_2\}$	$\{q_0q_1\}$
q_0q_2	$\{q_0\}$	$\{q_0q_1q_3\}$
$q_0q_1q_3$	$\{q_0q_2q_3\}$	$\{q_0q_1q_3\}$
$q_0q_2q_3$	$\{q_0q_3\}$	$\{q_0q_1q_3\}$
q_0q_3	$\{q_0q_3\}$	$\{q_0q_1q_3\}$

Can replace states as

	0	1
→ A	{A}	{B}
B	{C}	{B}
C	{A}	{D}
D	{E}	{D}
E	{F}	{D}
F	{F}	{D}

DFA:



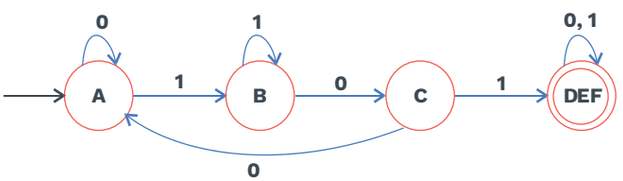
Minimal DFA:

$$\Pi_0 = \{(A,B,C),(D,E,F)\}$$

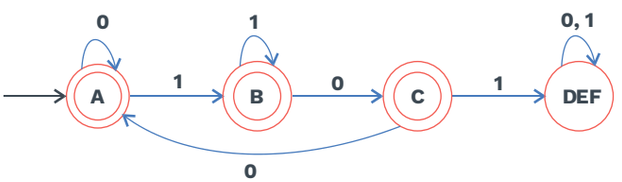
$$\Pi_1 = \{(A,B),(C),(D,E,F)\}$$

$$\Pi_2 = \{(A),(B),(C),(D,E,F)\}$$

$$\Pi_3 = \Pi_2$$



Now, complement the above DFA to get the minimal DFA that accepts all the strings over $\Sigma = \{0,1\}$ where each string "do not contain "101" as a substring."



2.3 EPSILON NON-DETERMINISTIC FINTE AUTOMATA (∈-NFA)

Specification of epsilon NFA:

- Epsilon- NFA $(\in -\text{NFA})$ is the NFA which contains $\in$ - moves/Null moves.
- This automation allows $\in$ as possible transition.
- $\in$-NFA is defined as 5-tuple:

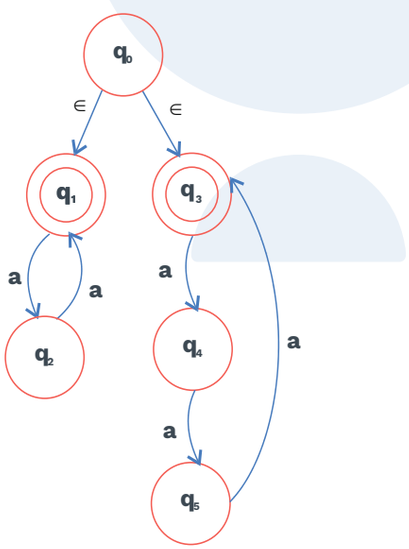
$(Q, \Sigma, \delta, q_0, F)$

Where transition function:

δ: $\boxed{Q \times \Sigma \cup \{\in\} \to 2^Q}$

Q: finite set of states
Σ: finite set of input symbols
q_0: Initial state
F: set of final states

Example: $\{a^n \mid n \text{ is even or divisible by 3}\}$

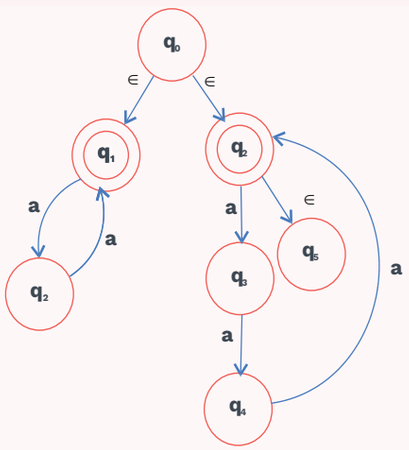


∈-closure

Definitions

The $\in$-closure of the state q, is the set that contains q along with all states which can be reached by only getting any number of $\in$ from state q.

Rack Your Brain



∈–closure (q_0) = ?
∈–closure (q_2) = ?

Conversion from ∈-NFA to NFA

- Why need ∈–NFA, when already NFA is present?

NFA: Easier to construct
∈–NFA: Even more easier to construct.

Procedure:

Let $N_1 = (Q', \Sigma', \delta', q_0', F')$ is the ∈–NFA.
$N_2 = (Q, \Sigma, \delta, q_0, F)$ is the equivalent NFA.

i) Initial state:

$$q_0 = q_0'$$

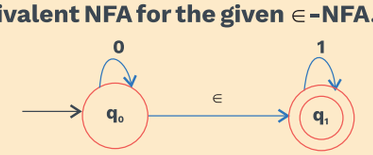
ii) δ Construction:

$$\forall_{i \in \Sigma}\ \forall_{q \in Q}\ \delta(q, i) = \in\text{-closure}\ [\delta'\ (\in\text{-closure}(q), i)]$$

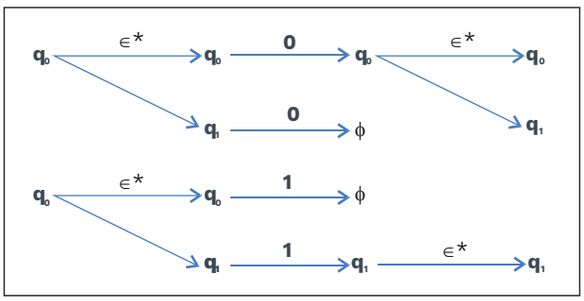
iii) Final state: Find ∈-closure of each state, if it contains any of the final state of N_1, then those state(s) will be the final state(s) in equivalent NFA N_2.

SOLVED EXAMPLES

Q1 **Construct an equivalent NFA for the given ∈-NFA.**



Sol: $\in$–closure $(q_0) = \{q_0, q_1\}$
$\in$–closure $(q_1) = \{q_1\}$

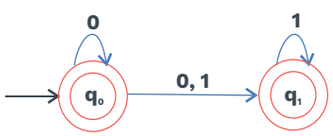


$\in$-NFA to NFA

Transition table:

δ	0	1
$\rightarrow q_0$	$\{q_0 q_1\}$	$\{q_1\}$
q_1	ϕ	$\{q_1\}$

NFA:

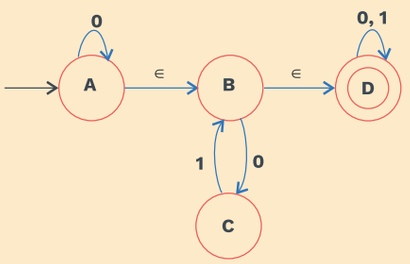


Final states in the NFA: Find the $\in$-closure of each state, if it contains any of the final state, then include this state as part of final states of equivalent NFA.
That's why here q_0 and q_1 both are final states.

Note:

In the conversion of $\in$-NFA to NFA, number of final states may increase.

Q2 **Construct an equivalent NFA for the given ∈-NFA.**



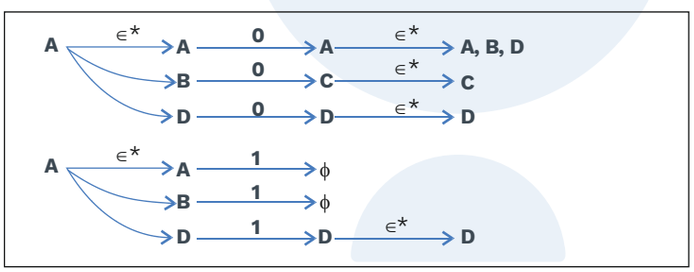
Sol:

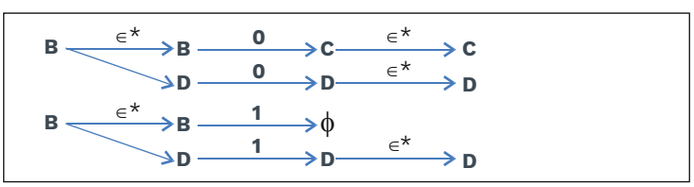
∈-closure A) = {A, B, D}

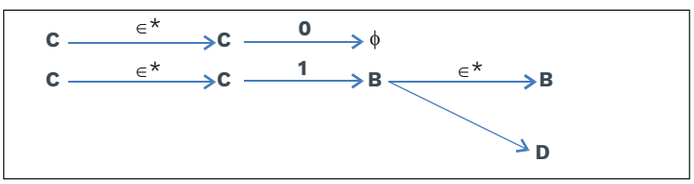
∈-closure B) = {B, D}

∈-closure C) = {C}

∈-closure D) = {D}





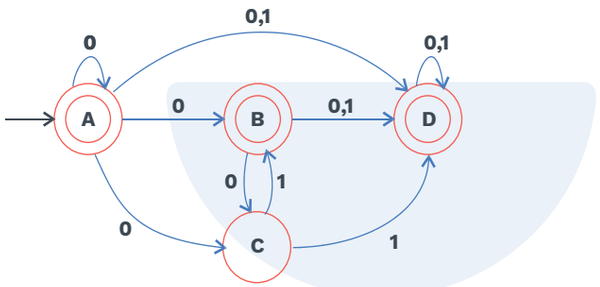


∈-NFA to NFA:

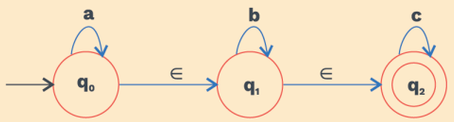
Transition table:

δ	0	1
→ (A)	{A, B, C, D}	{D}
(B)	{C, D}	{D}
C	ϕ	{B,D}
(D)	{D}	{D}

NFA:



Q3 **Construct an equivalent NFA for the given ∈-NFA.**

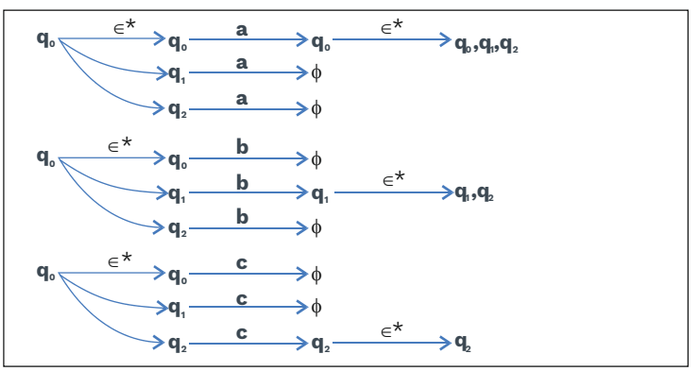


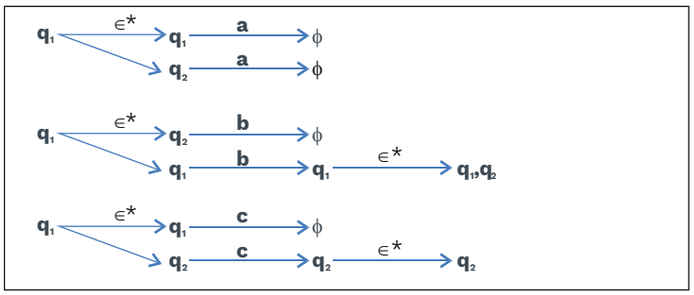
Sol:

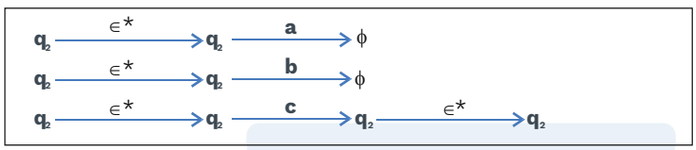
∈-closure (q_0) = {q_0, q_1, q_2}

∈-closure (q_1) = {q_1, q_2}

∈-closure (q_2) = {q_2}





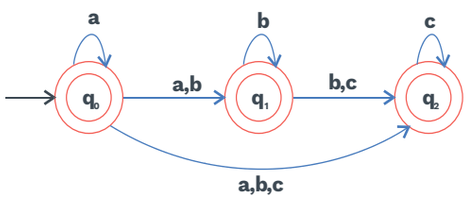


∈-NFA to NFA

Transition table:

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

NFA:



Rack Your Brain

Is the language preserved in all the steps while eliminating e-transition from a NFA?

Note:

Conversion from ∈–NFA to NFA shows that ∈–NFA and NFA are equivalent in power.

Previous Years' Question

Let δ denote the transition function and $\hat{\delta}$ denotes the extended transition function of the $\in$-NFA whose transition table is given below: **(GATE-2017 Set-2)**

δ	$\in$	a	b
$\rightarrow q_0$	$\{q_2\}$	$\{q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_0\}$	ϕ	ϕ
q_3	ϕ	ϕ	$\{q_2\}$

Then $\hat{\delta}(q_2, aba)$ is

1) ϕ **2)** $\{q_0, q_1, q_3\}$

3) $\{q_0, q_1, q_2\}$ **4)** $\{q_0, q_2, q_3\}$

Sol: 3)

Conversion from epislon-NFA to DFA:

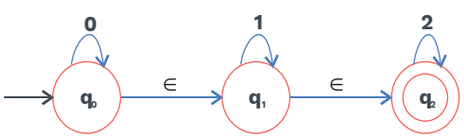
Let $N' = (Q', \sum', \delta', q_0', F')$ is the $\in$-NFA.

$M = (Q, \sum, \delta, q_0, F)$ is the equivalent DFA

i) Initial State:

$q_0 = \in$–closure$\left(q_0'\right)$

eg:



$\in$–closure $(q_0) = \{q_0, q_1, q_2\}$

ii) δ Construction:

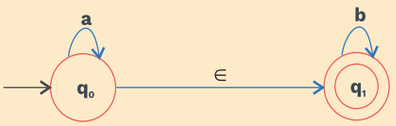
- Start Constructing δ with initial state and continue for every new state that appears under the input column.
- Terminate this process when no new state appears.

iii) Final state:

All states in equivalent DFA which contain final state of $\in$-NFA as a subset, is going to be the final state of DFA.

SOLVED EXAMPLES

Q1 **Construct an equivalent DFA for given $\in$-NFA.**



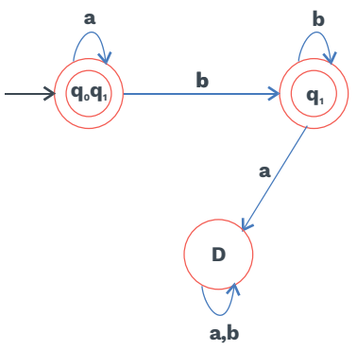
Sol: $\in$ – Closure $(q_0) = \{q_0, q_1\}$ = Initial state of DFA

Transition Table for DFA:

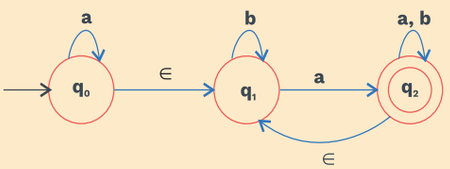
δ	a	b
$\rightarrow q_0q_1$	$\{q_0q_1\}$	$\{q_1\}$
q_1	{D}	$\{q_1\}$
D	{D}	{D}

Every state which contains q_1 will be the final state in DFA.

DFA:



Q2 **Construct an equivalent DFA for given ∈-NFA.**



Sol: ∈–closure (q_0) = {q_0, q_1}

We can construct the transition table for DFA as:

δ	a	b
→ q_0q_1	{$q_0q_1q_2$}	{q_1}
$q_0q_1q_2$	{$q_0q_1q_2$}	{q_1q_2}
q_1	{q_1q_2}	{q_1}
q_1q_2	{q_1q_2}	{q_1q_2}

Initial state

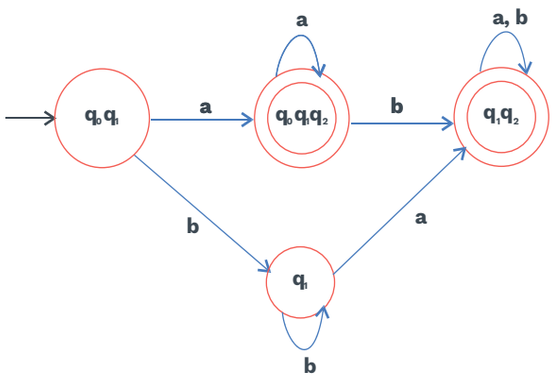
$q_0q_1 \xrightarrow{a} q_0,q_1, q_2$
$q_0q_1 \xrightarrow{b} q_1$

$q_0q_1q_2 \xrightarrow{a} q_0,q_1, q_2$
$q_0q_1q_2 \xrightarrow{b} q_1,q_2$

$q_1 \xrightarrow{a} q_1,q_2$
$q_1 \xrightarrow{b} q_1$

$q_1q_2 \xrightarrow{a} q_1,q_2$
$q_1q_2 \xrightarrow{b} q_1,q_2$

DFA:



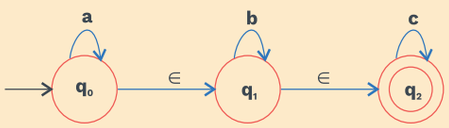
Here, $q_0q_1q_2$ and q_1q_2 are final states in DFA as it contains q_2 which is final states for given $\in$-NFA.

Note:

In the conversion from $\in$ –NFA to DFA:

- There might be change in initial state.
- There might be change in final state.
- There might be change in total number of states.

Q3 **Construct an equivalent DFA for given $\in$-NFA.**



Sol: $\in$–closure $(q_0) = \{q_0, q_1, q_2\}$

We can construct the transition table for DFA as:

δ	**a**	**b**	**c**
$\rightarrow q_0q_1q_2$	$\{q_0q_1q_2\}$	$\{q_1q_2\}$	$\{q_2\}$
q_1q_2	$\{D\}$	$\{q_1q_2\}$	$\{q_2\}$
q_2	$\{D\}$	$\{D\}$	$\{q_2\}$
D	$\{D\}$	$\{D\}$	$\{D\}$

Initial state

$q_0q_1q_2 \xrightarrow{a} q_0q_1q_2$

$q_0q_1q_2 \xrightarrow{b} q_1q_2$

$q_0q_1q_2 \xrightarrow{c} q_2$

$q_1q_2 \xrightarrow{a}$ Dead State 'D'

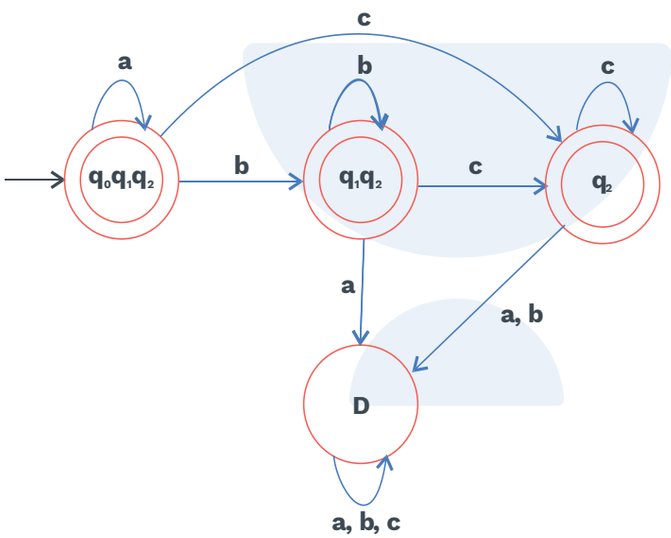
$q_1q_2 \xrightarrow{b} q_1,q_2$

$q_1q_2 \xrightarrow{c} q_2$

$q_2 \xrightarrow{a}$ Dead State 'D'
$q_2 \xrightarrow{b}$ Dead State 'D'
$q_2 \xrightarrow{c} q_2$

D $\xrightarrow{a}$ Dead State 'D'
D $\xrightarrow{b}$ Dead State 'D'
D $\xrightarrow{c}$ Dead State 'D'

DFA:

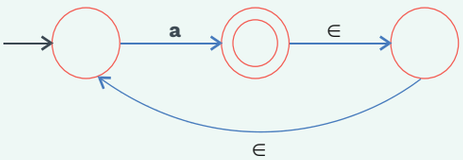


Note:

The DFA obtained from ∈ –NFA need not be minimal.

Previous Years' Question

What is the complement of the language accepted by the NFA shown below? Assume $\Sigma = \{a\}$ and $\in$ is the empty string. **(GATE-2012)**



1) ϕ **2)** $\{\in\}$

3) a^* **4)** $\{a, \in\}$

Sol: 2)

Note:

- Conversion from $\in$–NFA to DFA shows that $\in$–NFA and DFA are equivalent in power.
- $\in$–NFA, NFA and DFA all are equivalent in power.

2.4 FINITE TRSol:DUCER

Definitions

Finite state trSol:ducer is a finite state automaton which produces output on reading any input.

- A finite state trSol:ducer (FST) can be thought of as trSol:lator or relator between strings in a set.
- It is very useful in parsing.

Moore machines

Definitions

Moore Machines are finite state machines with output value associated with states.

It can be defined as –

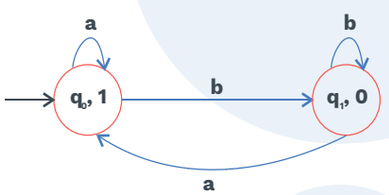
$(Q, \Sigma, \delta, q_0, \Delta, \lambda)$ where

Q: Finite set of states
Σ: Input alphabets
δ: Transition function

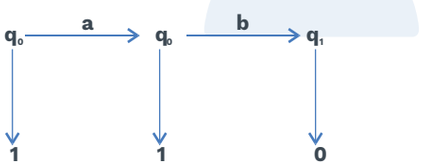
$$\boxed{Q \times \Sigma \rightarrow Q}$$

q_0 Initial state
Δ: output alphabets
λ: output function which maps $Q \rightarrow \Delta$

eg:



Input: ab



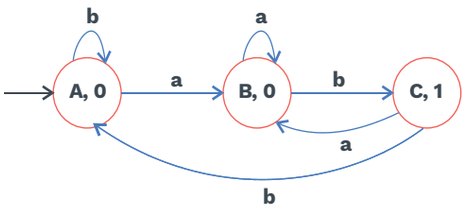
Note:
In Moore machines if input is of 'n bits' then output will be of 'n+1 bits'.

SOLVED EXAMPLES

Q1 **Construct a moore machine that takes set of all strings over {a,b} as input and prints '1' as output for every occurrence of 'ab' as a substring.**

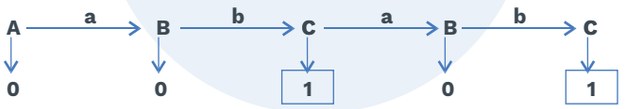
Sol: Whenever see 'ab' print '1' as output

$\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$



Take example and verify it:
Input1: ab

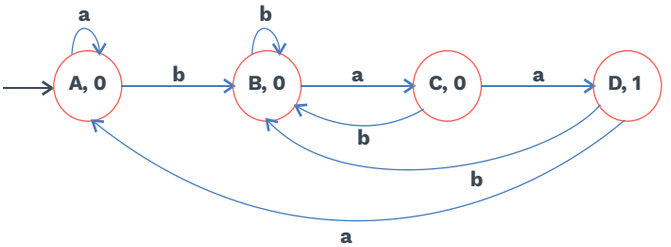
Input2: abab



Q2 **Construct a Moore machine that takes set of all strings over {a,b} and count number of occurrences of substring 'baa'.**

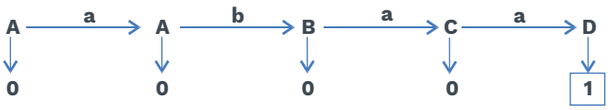
Sol: $\Sigma = \{a, b\}$

Take $\Delta = \{0, 1\}$



Take example and verify it

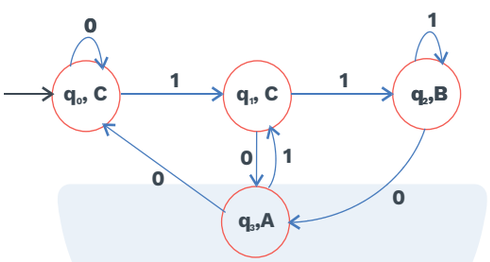
Input: abaa



Q3 **Construct a Moore machine that takes set of strings over Σ = {0, 1} and produces 'A' as output if input ends with '10' or produces 'B' as output if input ends with '11' otherwise produces 'C' as output.**

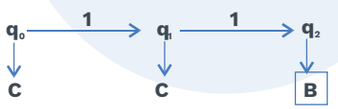
Sol: $\Sigma = \{0, 1\}$

$\Delta = \{A, B, C\}$



Take example and verify it:

Input 1: 11



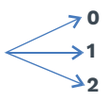
Input 2: 1 1 0

Q4 **Construct a moore machine that takes binary number as input and produces 'residue module 3' as output.**

Sol: $\Sigma = \{0, 1\}$

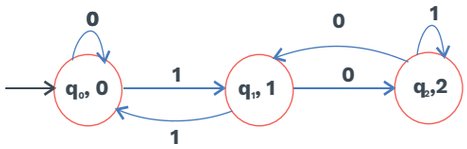
$\Delta = \{0, 1, 2\}$

Remainder: Binary number %3



δ	0	1	Δ
→ q_0	$\{q_0\}$	$\{q_1\}$	0
q_1	$\{q_2\}$	$\{q_0\}$	1
q_2	$\{q_1\}$	$\{q_2\}$	2

Represents states equal to the number of remainders when any binary number divisible by 3



Rack Your Brain

Construct a moore machine that takes base 4 numbers as input and produces 'reakdus modulo 6 output?

Mealy machine:

Definitions

Mealy machines are finite state machines, where output depends upon the present input symbol and present state of the machine.

- It can be defined as: $(Q, \Sigma, \delta, q_0, \Delta, \lambda)$, where

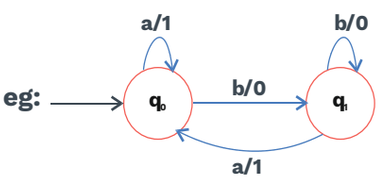
Q: Finite set of states
Σ: Input alphabets
δ: Transition function

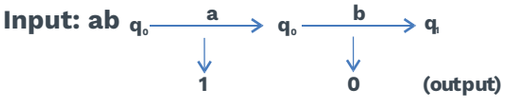
$$Q \times \Sigma \rightarrow Q$$

q_0: Initial state

Δ: Output alphabet

λ: Output function $Q \times \Sigma \rightarrow \Delta$





Note:
In Mealy machine, if input is of 'n bits' then output will be of 'n bits'.

SOLVED EXAMPLES

Q 1 **Construct a mealy machine that takes binary number as input and produces 2's complement of that number as output. Assume the string is read from LSB to MSB and end carry is discarded.**

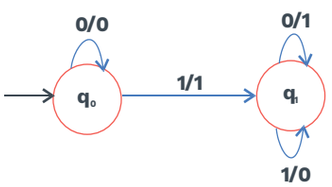
Sol: $\Sigma = \{0, 1\}$

$\Delta = \{0, 1\}$

To calculate: 2's complement of binary number 1011
1's complement of this number = 0100

2's complement of this number = $\begin{array}{r} 0\ 1\ 0\ 0 \\ +\ 1 \\ \hline 0\ 1\ 0\ 1 \end{array}$

Observation:
From LSB whenever you see 0, leave it as it is, whenever you see 1st 1, leave as it is. After this, whatever digits come, just complement it, will give 2's complement



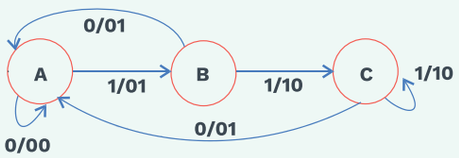
Rack Your Brain

Construct a mealy machine that takes binary umber as input and produces 1's complement of that number as output

Previous Years' Question

The finite state machine described by the following state diagram with A as starting state, where an arc label is x/y and x stands for 1-bit input and y stands for 2-bit output

(GATE-CS-2002)



1) Outputs the sum of the present and the previous bits of the input.
2) Outputs 01 whenever the input sequence contains 11
3) Outputs 00 whenever the input sequence contains 10
4) None of the above

Sol: 1)

Rack Your Brain

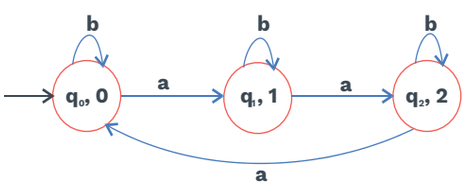
Construct a mealy machine that takes binary umber as input and produces output = $(\text{input})_2 \bmod 3$.

Conversion of Moore machine to Mealy machine

SOLVED EXAMPLES

Q1 **Convert a Moore machine to Mealy machine which count the number of a's divisible by 3.**

Sol: We have to construct a Moore machine which counts the number of a's is divisible by 3.

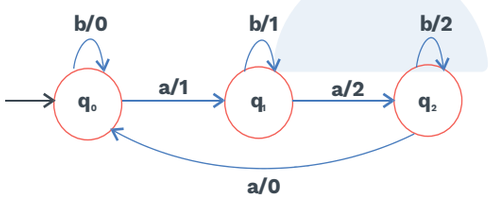


	a	b	Δ
→ q_0	$\{q_1\}$	$\{q_0\}$	{0}
q_1	$\{q_2\}$	$\{q_1\}$	{1}
q_2	$\{q_0\}$	$\{q_2\}$	{2}

When convert moore to mealy, then q_0 state on input a goes to q_1 states and output associated with state q_1 is 1 so, it will go onto the transition in equivalent mealy machine. Apply similar concepts for every state.

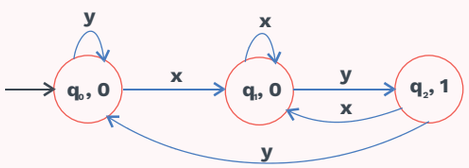
Transition: Diagram for mealy machine:

	a	b
→ q_0	$\{q_1,1\}$	$\{q_0,0\}$
q_1	$\{q_2,2\}$	$\{q_1,1\}$
q_2	$\{q_0,0\}$	$\{q_2,2\}$



Rack Your Brain

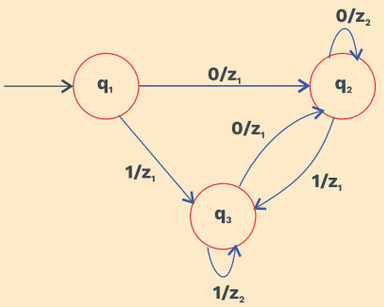
Convert the following given Moore machine to Mealy machine:



Conversion from mealy to moore machine:

SOLVED EXAMPLES

Q1 **Convert the following given Mealy machine to Moore machine:**

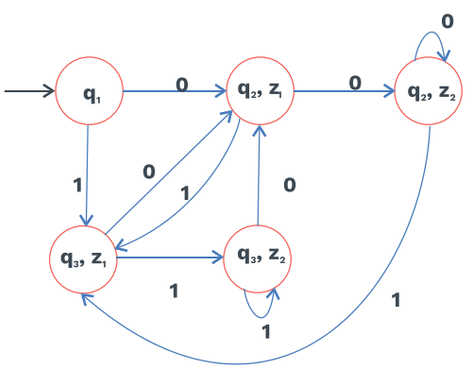


Sol:

i) State q_1 on getting input 0 goes to state q_2 and the output present in this transition gets associated with states q_2 in the corresponding equivalent moore machine.

ii) State q_1 on getting input 1 goes to state q_3 and the output present in this transition i.e. 'z_1' gets associated with state q_3 in the corresponding equivalent moore machine.

iii) We will follow this above procedure for other states as well as the new states we are getting.

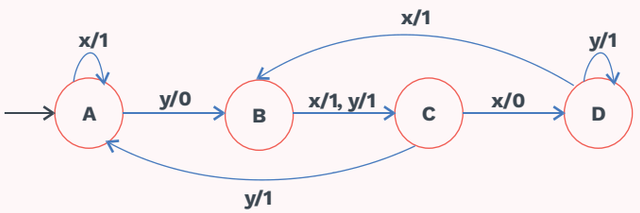


Note:

- In the conversion from Moore to Mealy machine, the number of states remain same.
- In the conversion from Mealy to Moore machine, the number of states may increase.

Rack Your Brain

Convert the following given Mealy machine to equivalent Moore machine.



Chapter Summary

- **Finite Automata:** Finite automata is a mathematical model which involves states and transitions among states in response to inputs.
- **Types of Finite Automata:** Finite automata without output and Finite automata with output.
- **Classification of Finite Automata without output:** DFA, NFA, $\in$–NFA.
- **Classification of Finite Automata with output:** Moore machine, Mealy machine.
- **DFA:** A DFA has a finite set of states and a finite set of input symbols. DFA is defined as quintuple (Q, $\sum$, δ, q_0, F) where $\delta : Q \times \sum \to Q$
- **Minimization of DFA:** TrSol:forming a given DFA into an equivalent DFA having the minimum number of states.
- **NFA:** A NFA is defined as Quintuple (Q, $\sum$, δ, q_0, F) where $\delta : Q \times \sum \to 2^Q$
- **Comparison of DFA and NFA:** Mainly the NFA differs from the DFA in that the NFA can have any number of transitions (including zero) to the next states from a given state on a given input symbol.
- **Conversion from NFA to DFA:** It is possible to convert any NFA to DFA that accepts the same language. All DFAs are NFA.
- DFA obtained from NFA need not be a minimal DFA.
- $\in$**-Transition:** It is possible to extend the NFA by allowing transitions on an empty input (i.e. no input symbol at all).
- $\in$**-NFA:** NFA in which transition is also defined for an empty input is known as $\in$-NFA.
- Conversion from $\in$-NFA to DFA as well as NFA is possible.
- **Finite State TrSol:ducer:** A finite state automaton which produces output on reading any input.
- **Moore Machines:** Finite state machine with output value associated with states. It is defined as (Q, $\sum$, δ, q_0, Δ, λ) where output function $\lambda : Q \to \Delta$.
- **Mealy Machines:** Finite state machines where output is associated with the transition.
  It is defined as (Q, $\sum$, δ, q_0, Δ, λ) where output function $\lambda : Q \times \sum \to \Delta$.