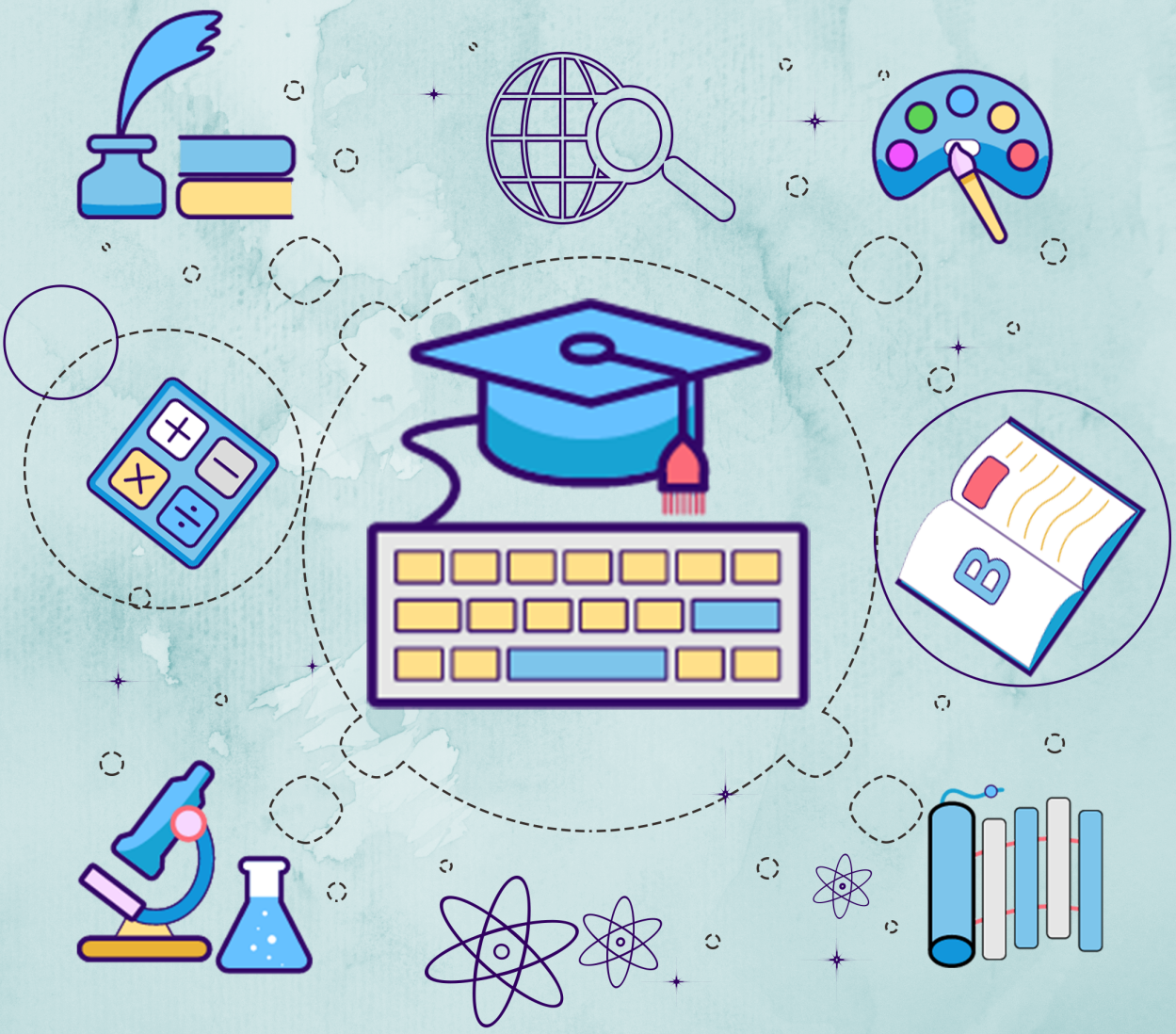
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

KeralaNotes



SYLLABUS | STUDY MATERIALS | TEXTBOOK

PDF | SOLVED QUESTION PAPERS

www.keralanotes.com

KeralaNotes

KTU STUDY MATERIALS

FORMAL LANGUAGES AND AUTOMATA THEORY

CST 301

Module 1

Related Link :

- KTU S5 STUDY MATERIALS
- KTU S5 NOTES
- KTU S5 SYLLABUS
- KTU S5 TEXTBOOK PDF
- KTU S5 PREVIOUS YEAR SOLVED QUESTION PAPER

FORMAL LANGUAGES AND AUTOMATA THEORY

Module 1

Introduction to Formal Language Theory and Regular Languages:

Introduction to formal language theory– Alphabets, Strings, Concatenation of strings, Languages.

Regular Languages - Deterministic Finite State Automata (DFA) (Proof of correctness of construction not required), Nondeterministic Finite State Automata (NFA), Equivalence of DFA and NFA, Regular Grammar (RG), Equivalence of RGs and DFA.

Prepared by:

Karishma PK

Assistant professor

Computer Science Department EKCTC

MODULE - I (Introduction to formal language theory & Regular languages)

Any task performed by calculator or a computer system or any machine is known as computation.

- **Automata** (or a Machine)

Mathematical model which perform some computation.

Some basic elements of automata are

- symbols
- Alphabets
- Strings
- Languages

Symbol

- Symbols are the basic building blocks of Automata. It can be any letter, picture or number.

  Eg:- 0, 1, 2, a, b, c etc.

Alphabets

- A finite set of symbols.
- Represented as Σ

  Eg:- $\Sigma = \{a, b, c, \ldots z\}$, set of lowercase symbols in english.

  $\Sigma = \{0, 1\}$, Binary alphabet

  $\Sigma = \{0, 1, 2 \ldots 9\}$, Decimal alphabet

  $\Sigma = \{\text{☺}, \text{☻} \ldots \text{☼}\}$

String

- Sequence of symbols taken from an alphabet (Σ)

  Eg:- a, b, aa, ab, abc etc.

Length of a string

- It is the no: of symbols present in a string.
- It is denoted by $|s|$
- Eg:- if $s = cabcad$, $|s| = 6$
- length of an empty string $|\varepsilon| = 0$

Eg:- $x = 0101$

$|x| = 4$

Empty string / Null string

- A string of length zero. (String with zero no: of symbols)
  ie, $|s| = 0$
- Denoted by ε or λ $(|\varepsilon| = 0)$

Kleen closure

- set of all strings on an alphabet (*)
- Denoted by Σ^*

Eg:- Suppose $\Sigma = \{a, b\}$

$\Sigma^0 = \{\varepsilon, \lambda\}$

$\Sigma^1 = \{a, b\}$

$\Sigma^2 = \{a, b\} \cdot \{a, b\}$

$= \{aa, ab, ba, bb\}$

$\Sigma^3 = \{aaa, aab, aba, abb, bbb, bba, baa, bab\}$

$\therefore \Sigma^* = \{\varepsilon, a, b, aa, bb, ab \ldots\}$

Σ^n → n length strings

ie, $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cdots \Sigma^n$

positive closure

set of non empty strings of an alphabet.

denoted by Σ^+

$$\Sigma^+ = \Sigma^* - \{\lambda\}$$

$$\Sigma^+ = \{a, b, aa, ab \cdots\}$$

Concatenation of strings

Join two or more strings.

- Let $x = a_1, a_2, a_3 \cdots a_n$

  $y = b_1, b_2, b_3 \cdots b_n$

  concatenation of $xy = a_1, a_2, a_3 \cdots a_n, b_1 b_2 \cdots b_n$

- Let $x = 1011$ & $y = 0110$

  $xy = 10110110$

  $yx = 01101011$

Formal Languages

Any subset of Σ^* (Kleen closure) is called a Formal language or simply language.

Σ^* having all combinations

Language

- Language can be finite or infinite
- Collection of strings which are chosen from Σ^*
- Language is the subset of Σ^*

$L \subseteq \Sigma^*$, L is a language

Let $\Sigma = \{a, b\}$, then languages can be define like

L_1 = set of all strings of length 2

= {aa, ab, ba, bb}, It is a finite set.

so L_1 is a finite Language.

L_2 = set of all strings of length 3

= {aaa, aab, abb, bbb, bba, baa, aba bab}. It is a finite set. so L_2 is a finite language.

L_3 = set of all strings ~~of length~~ where each strings starts with 'a'.

= {a, aa, ab, aaa, abb, aab, aba, abb ...}

It is an infinite set. so L_3 is an infinite language.

- Language can be Finite or Infinite.

e) $\Sigma = \{a, b\}$, no of symbols = 2

& needed length is 2

$\therefore 2^2 = 4$

If length is 3 $\Rightarrow 2^3 = 8$

ϕ / Empty language

Language consist of only empty string

ie, $\{\varepsilon\}$

PRACTICE QUESTIONS

Q) Find the number of finite 2 length string using the alphabet $\Sigma = \{a, b, c\}$?

$3^2 = 9$

Soln:- aa, ab, ac, ba, bb, bc, ca, cb, cc

2) How many strings of length n are possible from the alphabet $\Sigma = \{a, b\}$?

Soln = $\{a,b\}\{a,b\}\{a,b\}\cdots n$

$= 2 \times 2 \times \cdots n$

$= 2^n$

where are use languages, strings in practical applications ?

consider c programming.

```
Void main ()
{
  int a,b;
  ....
  ....
}
```

(In C, it is a pgm, but in FLAT it is a string)

END PG

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

  $L_1 = \{aa, ab, ba, bb\}$.

  check whether the string "aaa" is a part of this language or not?

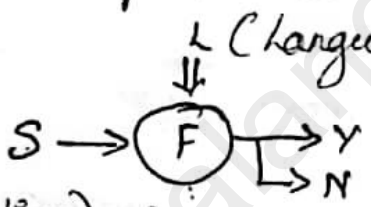
  Soln :- no, because L_1 is finite language.

- Let $L_2 = \{a, aa, aaa, ab, \ldots\}$.

  check whether the string "baba" is a part of this language or not?

  Soln :- L_2 is infinite, we should check with all strings in the language. Instead of this, we can use an easy method.

- For a given infinite language, we can create a finite representation.



(String) (Finite representation)

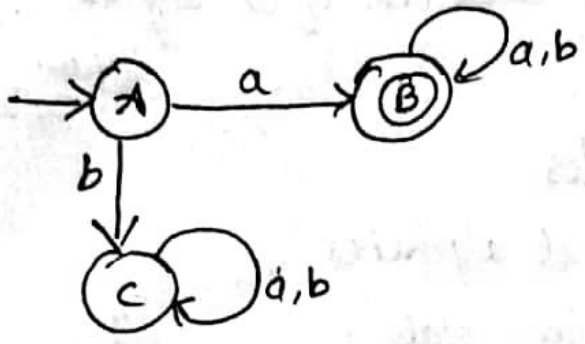
- The finite representation is called Finite Automata (FA), which is a small machine.

Finite Automata (FA)

- Finite representation
- Small machine
- An automaton with a finite no: of states is called a Finite Automata or Finite state Machine (FSM)

Let $\Sigma = \{a, b\}$. consider the language L_1 = set of all strings which starts with 'a' ?

$L_1 = \{a, aa, ab, aaa \ldots\}$ which is infinite



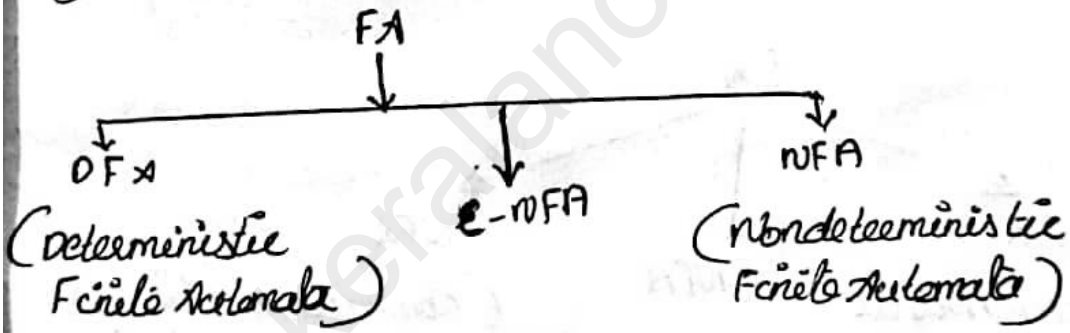
→ The circles are called states

→ A, B and C are states

→ A is initial state

→ B is final state

Types of Finite Automata



Finite Automata have 2 states

(i) Accept state (processed successfully & reach final state)

(ii) Reject state

It represented as

(i) Graphical (Transition diagrams)

(ii) Tabular (Transition Table)

(iii) Transition fn

Formal definition of FA

- A finite automaton is a collection of 5-tuples (quintuples)

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

  Q : Finite set of states

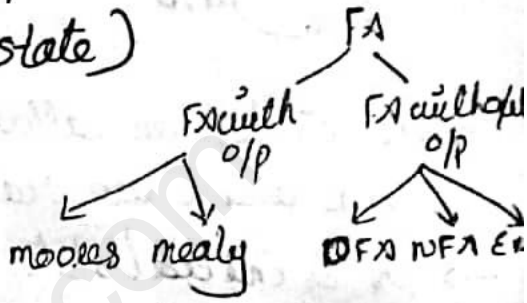
  Σ : Finite set of input alphabet

  q_0 : Initial state (start state)

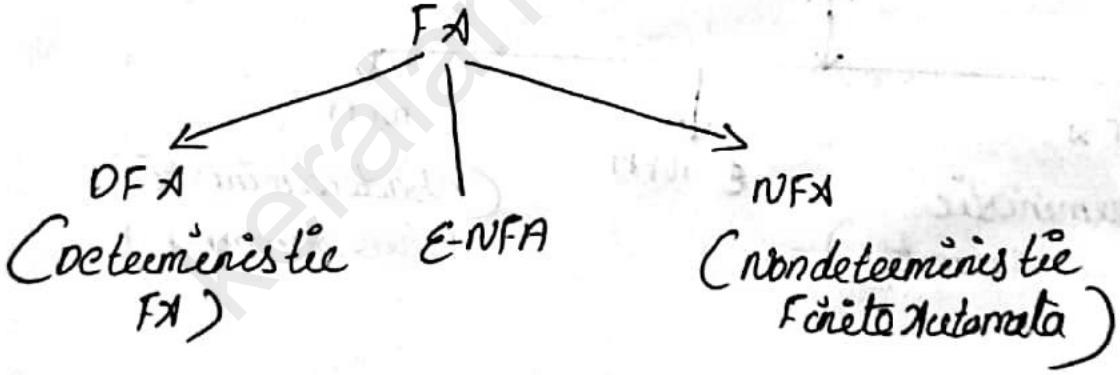
  F : Set of final states.

  δ : Transition fn.

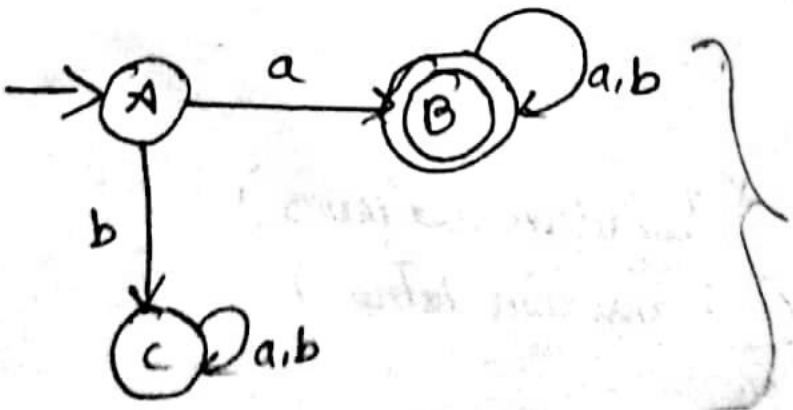
  $\delta = Q \times \Sigma \rightarrow Q$



Types of FA



Deterministic Finite Automata (DFA)



Q = set of all states (A, B, C)

Σ = input alphabet (a, b)

q_0 = start state (A)

F = set of final states (B) [More than one final state is possible]

Q, Σ, q_0 & F are common for DFA, NFA & ε-NFA. The only change in δ.

δ is a transition fn from $Q \times \Sigma \to Q$

ie, $\{A, B, C\} \times \{a, b\}$

$= (A,a)\ (A,b)\ (B,a)\ (B,b)\ (C,a)\ (C,b)$

$(A, a) \to B$
$(A, b) \to C$
$(B, a) \to B$
$(B, b) \to B$
$(C, a) \to C$
$(C, b) \to C$

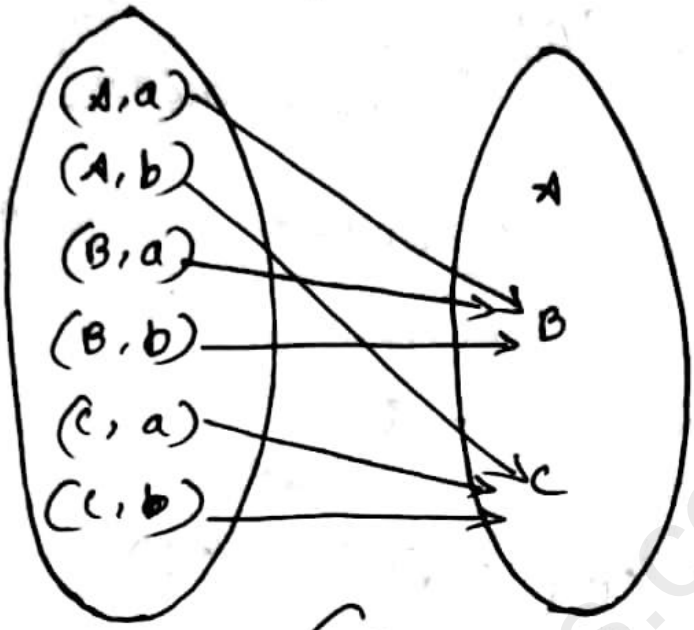
} Transition fn (ii)

States	input alphabet	
	a	b
q_0 A	B	C
q_1 B	B	B
q_2 C	C	C

} Transition Table (iii)

- There is only single resultant state ie, there is only one transition

$\delta: Q \times \Sigma \rightarrow Q$



(Transition fn representation.

- Eg:-

Transition fn

$(q_0, a) \rightarrow q_0$

$(q_0, b) \rightarrow q_1$.

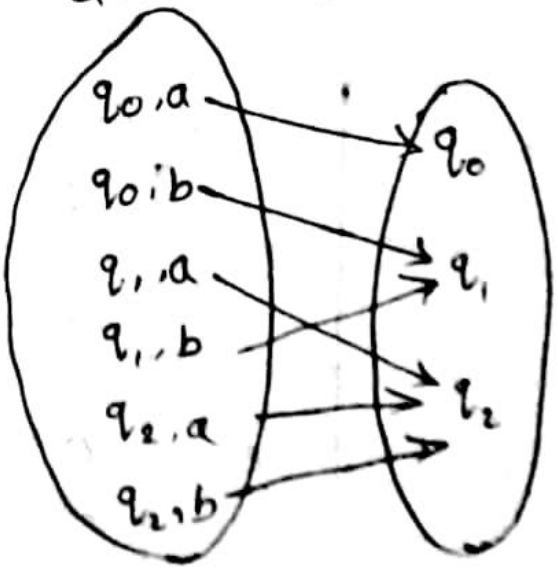
$(q_1, a) \rightarrow q_2$

$(q_1, b) \rightarrow q_1$

$(q_2, a) \rightarrow q_2$

$(q_2, b) \rightarrow q_2$

$Q \times \Sigma \rightarrow Q$



Transition table

states	input alphabet	
	a	b
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_2	q_2

NOTE

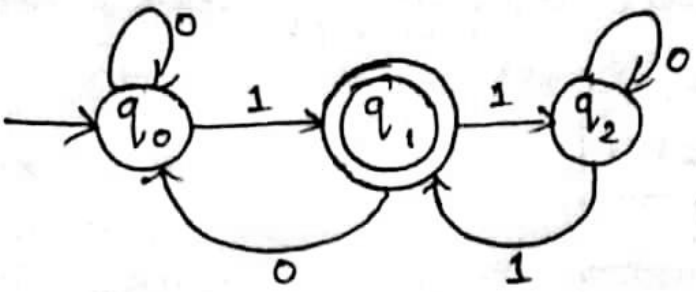
In a DFA, from each state of transition graph there must be exactly one transition for each input symbol of the alphabet.

Represent the following DFA using transition table?

$M = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$

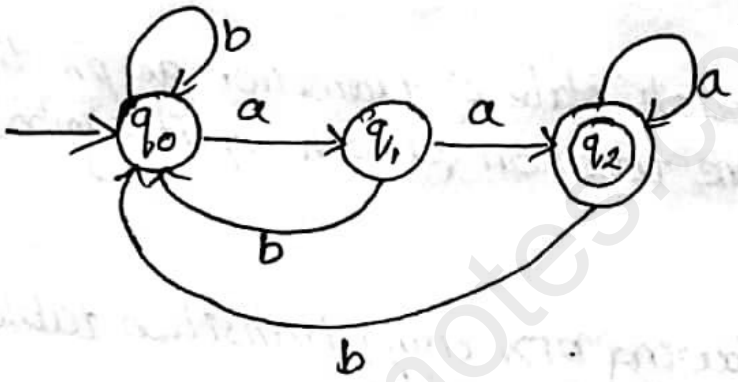
where δ is

	0	1
$\rightarrow q_0$	q_0	q_1
$+ q_1$	q_0	q_2
q_2	q_2	q_1



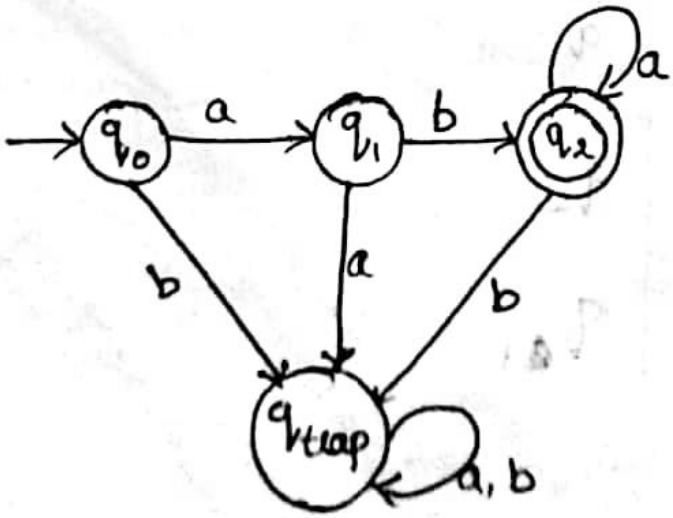
Q) Design a DFA accepting the language
$L = \{x \in \{a,b\}^* / x \text{ ends with } aa\}$!

Soln:- $L = \{aa, baa, bbaa, abbaa \ldots\}$



aabaa
write, T^n I^n
& T^n table

Q) Design a DFA which accepts the language
$L = \{aba^n ; n \geq 0\}$
$L = \{ab, aba, abaa, abaaa \ldots\}$



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

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

$\delta : Q \times \Sigma \rightarrow Q$

Here

$(q_0, a) \rightarrow q_1$

$(q_0, b) \rightarrow q_{trap}$

$(q_1, a) \rightarrow q_{trap}$

$(q_1, b) \rightarrow q_2$

$(q_2, a) \rightarrow q_2$

$(q_2, b) \rightarrow q_{trap}$

$(q_{trap}, a) \rightarrow q_{trap}$

$(q_{trap}, b) \rightarrow q_{trap}$

q_0 : initial state (q_0)

F : Set of final states $\{q_2\}$

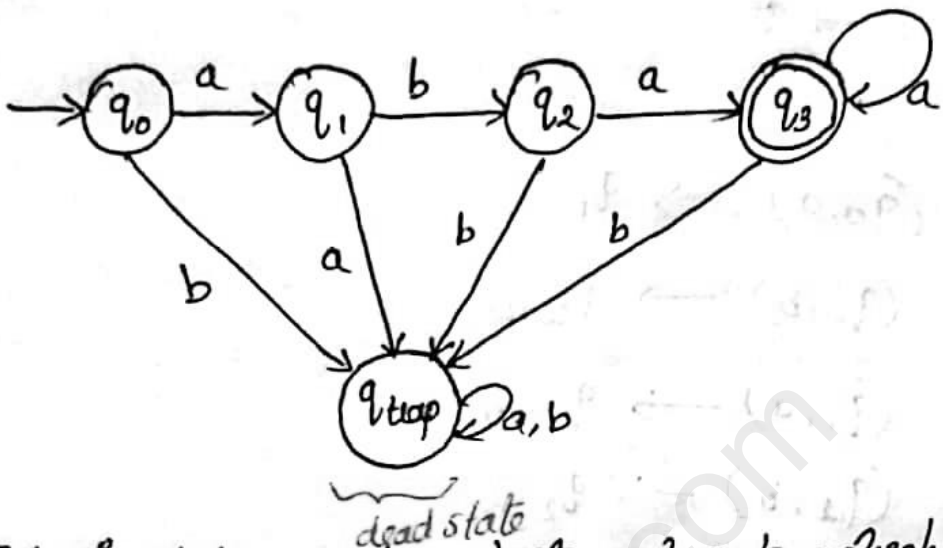
Transition table

states	i/p symbols	
	a	b
q_0	q_1	q_{trap}
q_1	q_{trap}	q_2
q_2	q_2	q_{trap}
q_{trap}	q_{trap}	q_{trap}

④ Design a DFA which accepts

$L = \{aba^n, \text{ where } n>0\}$

$L = \{aba, abaa, abaaa \cdots\}$



DFA is a deterministic finite automata which is defined as a quintuple $D = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1, q_2, q_3, q_{trap}\}$

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

$\delta : Q \times \Sigma \rightarrow Q$

$(q_0, a) \rightarrow q_1$

$(q_0, b) \rightarrow q_{trap}$

$(q_1, a) \rightarrow q_{trap}$

$(q_1, b) \rightarrow q_2$

$(q_2, a) \rightarrow q_3$

$(q_2, b) \rightarrow q_{trap}$

$(q_3, a) \rightarrow q_3$

$(q_3, b) \rightarrow q_{trap}$

$(q_{trap}, a) \rightarrow q_{trap}$

q_0 : initial state
$= q_0$

F : set of final states
$= \{q_3\}$

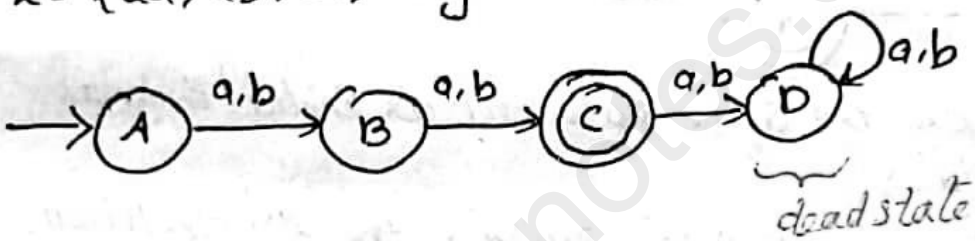
NOTE : construction of DFA

(i) Identify Σ

(ii) Identify language.

2) construct a DFA that accepts set of all strings over $\{a,b\}$ of length 2 ?

$L = \{aa, ab, ba, bb\}$



- If more than 2 length string comes, it will go to dead state D.
- Once to reach 'D', it will never go to previous state. So state D is called dead state.
- A string is said to be accepted by FA, if we are able to reach the final state starting from initial state up on scanning entire string, then the string is said to be accepted by the FA.
- If not reach to the final state, the string is said to be Rejected

String accept

- Scan the entire string and if we reach a final state from the initial state, we can say that the string is accepted. (more than one final state is possible)

Language accept

- A finite automata is said to accept a language, if all the strings in the language are accepted & all the strings not in the language are rejected.

Eg:- Let $L = \{aa, ab, ba, bb\}$.

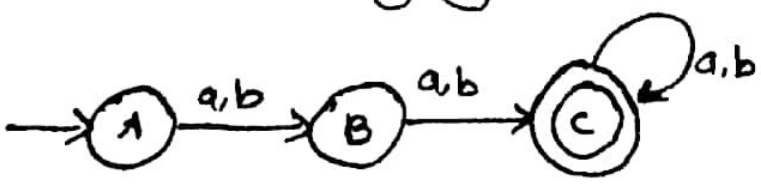
- Here the same state act as initial & final states.
- But it does not satisfies the 2nd condition, because it accept anything.

3) construct a DFA that accepts set of all strings w over $\{a,b\}$, where $|w| \geq 2$?

soln:- $\Sigma = \{a,b\}$

$L = \{aa, ab, ba, bb, aaa \cdots bbb \cdots\}$

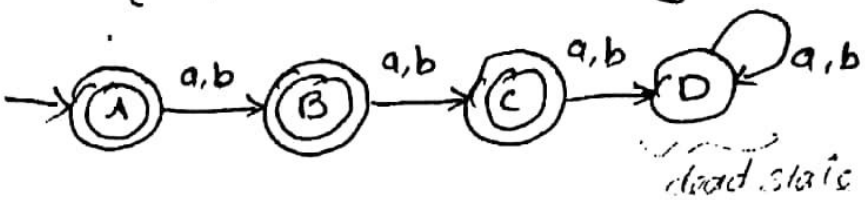
- infinite language.



Construct a DFA that accepts set of all strings w over {a,b} where $|w| \leq 2$ (almost 2)

Soln:- $\Sigma = \{a,b\}$

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



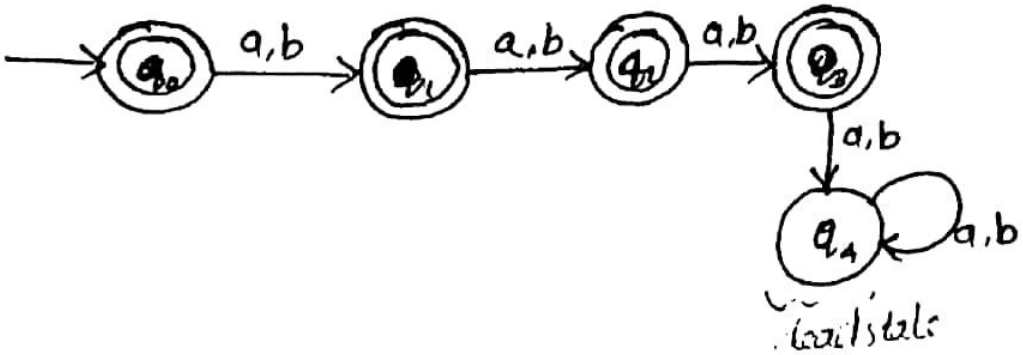
NOTE

- If $|w| = n$, then states required in minimal DFA is $n+2$
- If $|w| \geq n$, then states required in minimal DFA is $n+1$
- If $|w| \leq n$, then states required in minimal DFA is $n+2$
- There can be many possible DFAs for a pattern. A DFA with minimum no: of states is generally preferred.

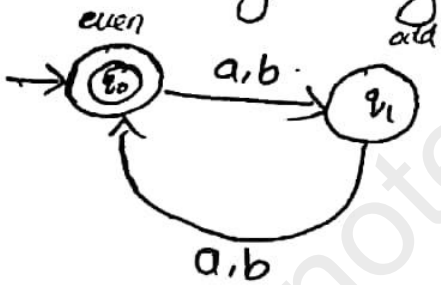
Q) Construct a DFA that accepts set of all strings w over {a,b} where $|w| \leq 3$ (almost 3)

Soln:- $\Sigma = \{a,b\}$

$L = \{\varepsilon, a, b, aa, bb, ab, ba, aaa, aba, aab, abb, bbb, bba, bab, baa\}$

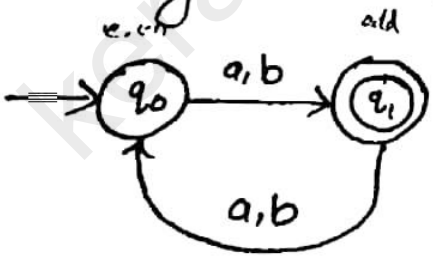


(4) Construct a minimal DFA which accepts set of all strings w over {a,b} such that |w| mod 2 = 0 (ie, even length strings) ?



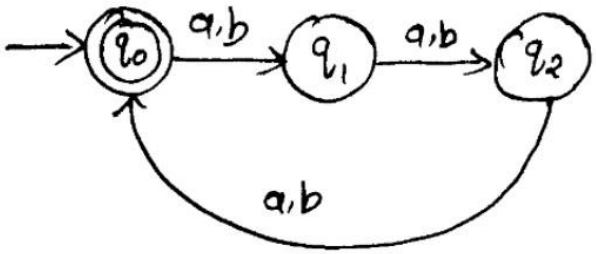
$L = \{\varepsilon, aa, ab, ba, bb, aaaa, \ldots\ldots bbbb\ldots\}$ infinite language.

(5) Construct a minimal DFA which accepts set of all strings w over {a,b} such that |w| mod 2 = 1 ?

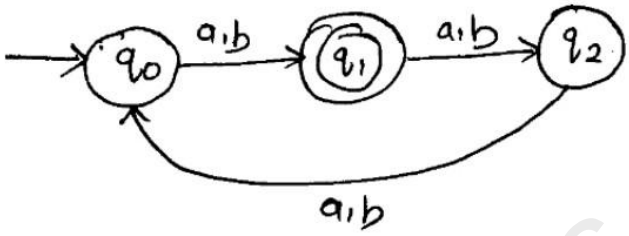


6) Construct a minimal DFA which accepts set of all strings w over {a,b} such that |w| mod 3 = 0.
(length of string is divisible by 3).

$L = \{\varepsilon, aaa, bab, \ldots bbb\ldots aaaaaa\ldots bbbbbb\ldots\}$ infinite language.



c) Construct a minimal DFA which accepts set of all strings w over {a,b} such that |w| mod 3 = 1.

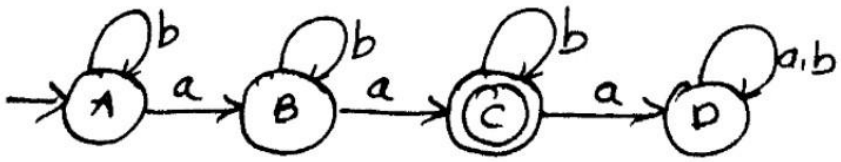


NOTE

Let the length of string |w| mod n=0, then minimal DFA contains 'n' number of state.

d) Construct a minimal DFA which accepts set of all strings w over {a,b} such that na(w) = 2. (no: of 'a' in a string should be two)

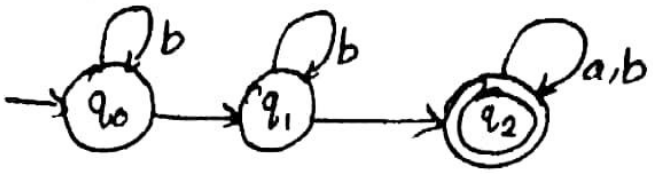
L = {aa, baa, aba, aab, bbaa .....}. infinite language



0a's 1a's 2a's 3a's

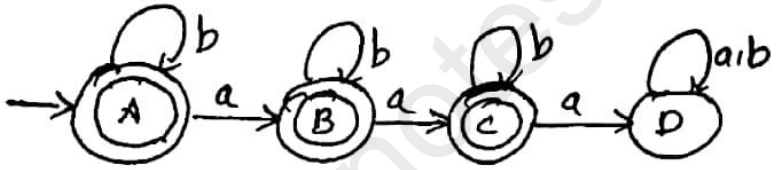
i) construct a minimal DFA which accepts set of all strings w over {a,b} such that $n_a(w) \geq 2$.
(no: of 'a' in a string should be ≥ 2)

$L = \{aa, baa, aaab, abab \ldots\}$, infinite



ii) construct a minimal DFA which accepts set of all strings w over {a,b} such that $n_a(w) \leq 2$.
(no: of 'a' in a string should be ≤ 2).

$L = \{\varepsilon, a, ab, ba, aa, baa, aba \ldots\}$ infinite



■ Extended Transition Function (δ^*)

$Q \times \Sigma^* \rightarrow Q$

- For representing string transitions.

1) Basis definition

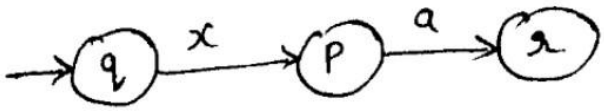
$\delta^*(q, \varepsilon) \rightarrow q$

2) Induction

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

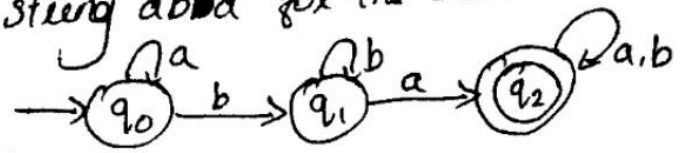
$\rightarrow \delta(p, a)$

$\rightarrow r$



$w = xa$

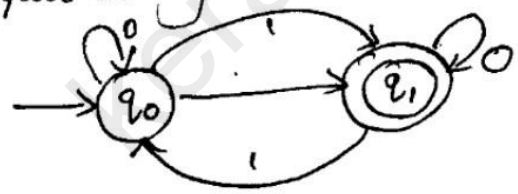
(1) Find the extended transition function for the string abba for the below DFA ?



soln:-

$$\delta^*(q_0, aba) = \delta(\delta^*(q_0, ab), a)$$
$$= \delta(\delta(\delta^*(q_0, a), b), a)$$
$$= \delta(\delta(q_0, b), a)$$
$$= \delta(q_1, a)$$
$$= q_2$$

Q) check whether the given DFA accepts the input string 01101 ?



soln:-

$$\delta^*(q_0, 01101) \rightarrow \delta(\delta^*(q_0, 0), 1101)$$
$$= \delta(\delta(\delta^*(\delta(\delta^*(q_0, 0), 1), 1), 0), 1)$$
$$= \delta(\delta(\delta(\delta(q_0, 1), 1), 0), 1)$$
$$= \delta(\delta(\delta(q_1, 1), 0), 1)$$

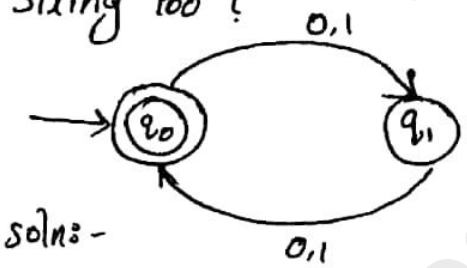
$= \delta(\delta(q_0, 0), 1)$

$= \delta(q_0, 1)$

$= q_1$ - Final state.

∴ It accept the string 01101.

ii) Check whether the given DFA accepts the string 100 ?



Soln:-

$\delta^*(q_0, 100) = \delta(\delta(\delta^*(q_0, 1), 0), 0)$

$= \delta(\delta(q_1, 0), 0)$

$= \delta(q_0, 0)$

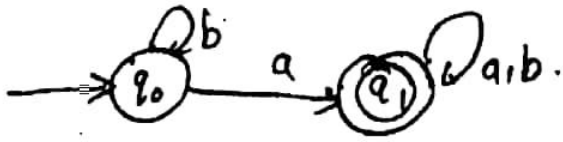
$= q_1$

Here q_1 is not a final state. Hence the given DFA does not accept the string 100.

iii) Draw DFA to accept strings of a's & b's having atleast 1 a !

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

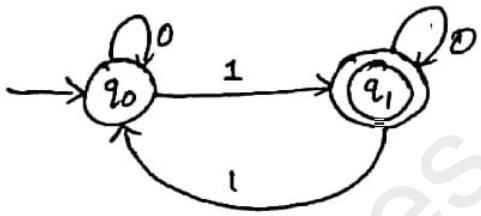
Soln:-



(i) Construct DFA for the following language
$L = \{x \mid x \in \{0,1\}^* \text{ \& } x \text{ contains odd of 1's}\}$

Soln:-

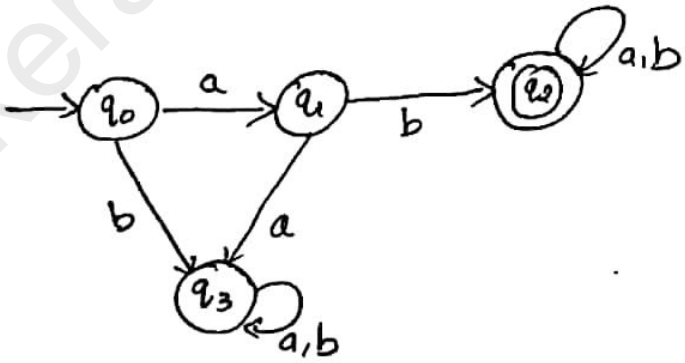
$L = \{1, 01, 100, 10011, 1001111 \ldots\}$



(ii) Draw transition diagram / graph for a given language ?

$L = \{abx \mid x \in \{a,b\}^*\}$

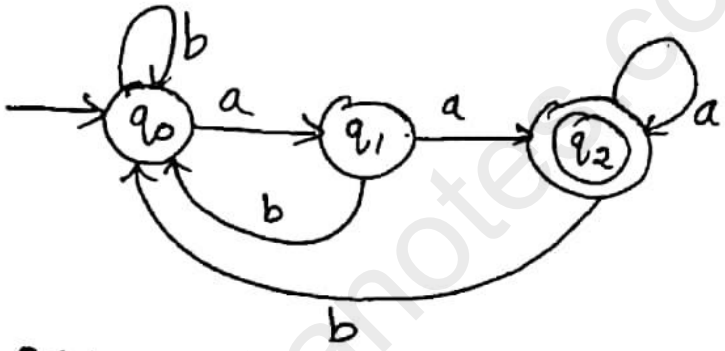
$L = \{ab, aba, abbba, ababab \ldots\}$



1) Design a DFA accepting the language of strings ending in aa?

Soln:- minimum strings aa

accepted	rejected
aa	ab
aaa	aba
baa	
abaa	
aa..aa	
bb..aa	
a..b..aa	



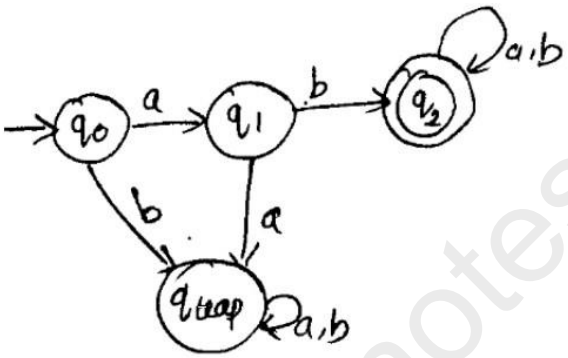
2) Find DFA recognize set of all strings $\Sigma = \{a,b\}$ starting with prefix ab.

$L = \{abx ; x \in \Sigma \{a,b\}^* \}$

Soln:- minimum → ab

possible → abaa..
abb...
abab

rejected → ba
aa
bb

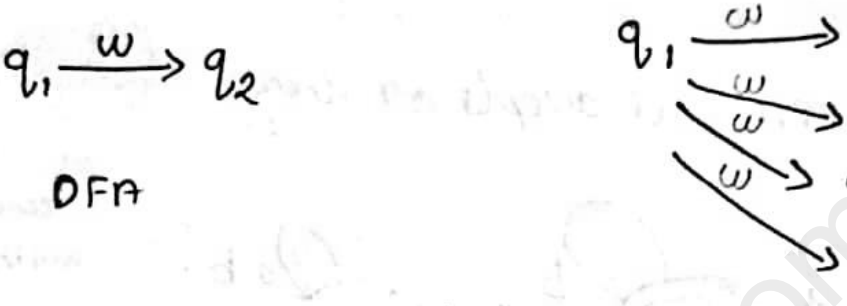


(2) Design a DFA that accepts all strings with exactly 1a's

min → a
possible : bab
b... ab....
rejected : baa

Non-deterministic Finite Automata (NFA)

- A Finite Automata is said to be non deterministic, if there is more than one possible transition from one state on the same input symbol.



- Nondeterministic machines are not real. (no real machines exist)
- If we want to do exhaustive search and backtracking, nondeterminism can be used.
- If we don't want to do any backtracking, determinism is used.
- NFA has a different transition function, rest is same as DFA.
- NFA is a collection of 5 tuple $(Q, \Sigma, \delta, q_0, F)$ where
  - Q : Finite set of states
  - Σ : Finite set of the input alphabet.
  - q_0 : Initial state
  - F : set of final states
  - δ : Transition function
    $Q \times \Sigma \longrightarrow 2^Q$

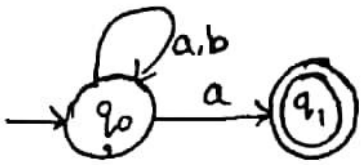
2^Q power set of Q. ie, all possible subset of Q.

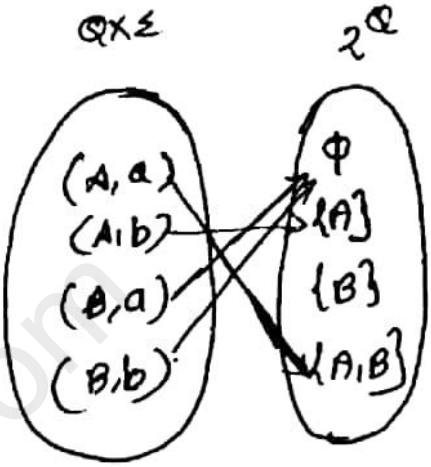
Eg:-

Construct NFA which accept set of all strings over $\Sigma = \{a, b\}$ in which every string ends with 'a'.

Soln:-

$L = \{a, aa, ba, \ldots\}$





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

$\delta(q_0, a) \rightarrow \{q_0, q_1\}$

$\delta(q_0, b) \Rightarrow \{q_0\}$

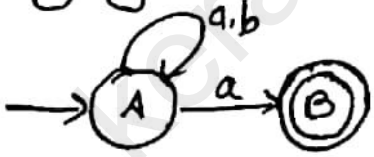
$\delta(q_1, a) \rightarrow \Phi$

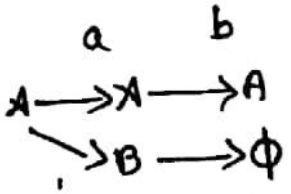
$\delta(q_1, b) \rightarrow \Phi$

$Q = 2$

$\therefore 2^Q = 4$

- Lets take another string 'ab', which is not in our language ?





Here A & Φ are not final states. so the string 'ab' is rejected.

now we can define the transition fn of NFA.

- There is no dead state in NFA

NOTE

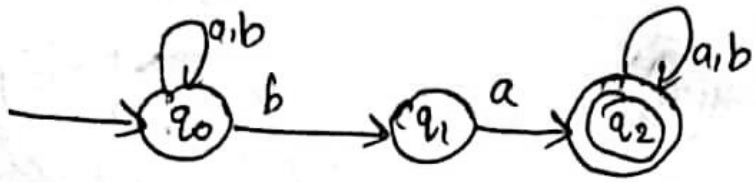
NFA :- $\delta : Q \times \Sigma \to 2^Q$

DFA :- $\delta : Q \times \Sigma \to Q$

Here Q is a part of 2^Q. So we can say that

Every DFA is an NFA, but the reverse is not true

Eg:-



$Q = 3$

$2^Q = 2^3 = 8$

$\delta(q_0, a) = \{q_0\}$

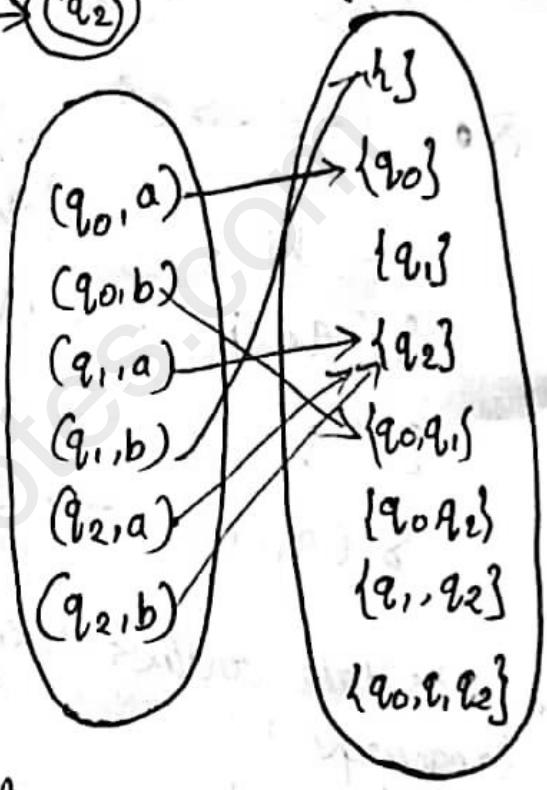
$\delta(q_0, b) = \{q_0, q_1\}$

$\delta(q_1, a) = \{q_2\}$

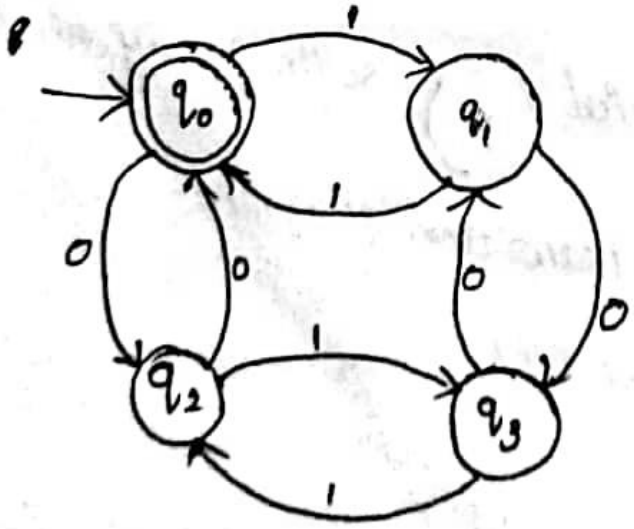
$\delta(q_1, b) = \phi$

$\delta(q_2, a) = \{q_2\}$

$\delta(q_2, b) = \{q_2\}$



Q) Design a DFA which accepts even no: of zero's & even no: of ones?



possible

11, 1111, 00, 0000, 1100, 0011, 1010, 0101

not possible

10, 01, 0, 1, 101, 010, 100, 001

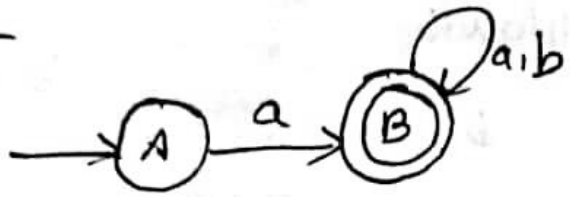
Q) construct NFA which accepts set of all strings over $\Sigma = \{a, b\}$

(i) starts with a

(ii) starts with b.

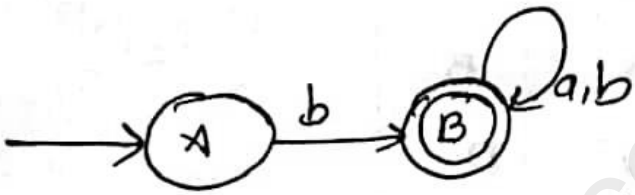
soln:-

(i)



no dead state

(ii)

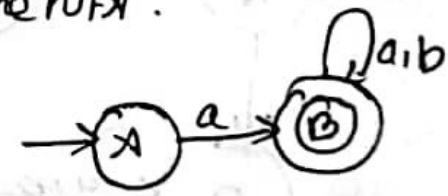


CONVERSION OF NFA TO DFA / Equivalence of DFA & NFA

- Here we use subset construction method

STEP 1

Draw state transition table for the NFA.



state	i/p alphabets	
	a	b
A	B	ϕ
B	B	B

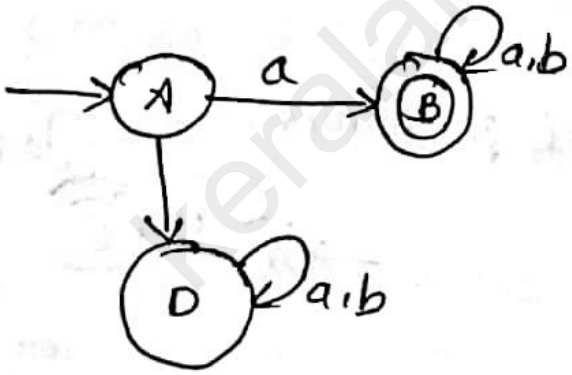
8. STEP 2

Draw state transition table for DFA from the above table.

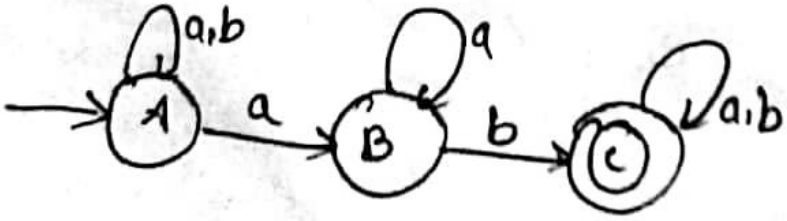
States	i/p alphabets	
	a	b
A	B	D
B	B	B
D	D	D

STEP - 3

Draw state transition diagram for DFA



9) Convert NFA to DFA ?



Soln:-

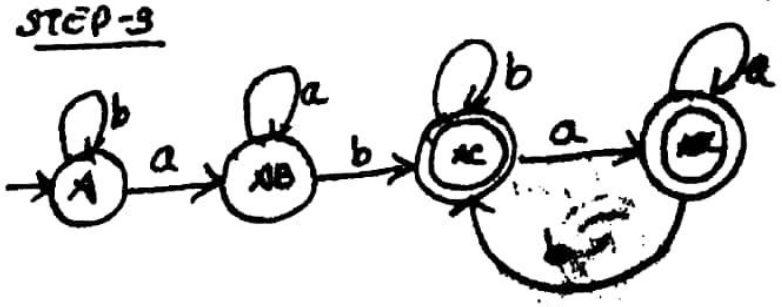
STEP-1

States	i/p a	b
A	{A,B}	{A}
B	{B}	{C}
C	{C}	{C}

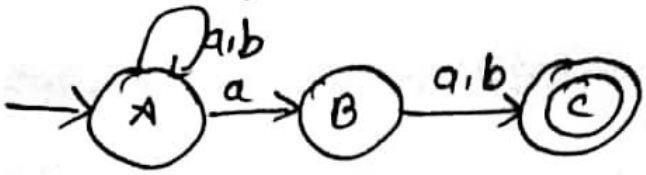
STEP-2

States	i/p alphabets a	b
A	{A,B}	{A}
AB	{A,B}	{A,C}
AC	{A,B,C}	{A,C}
ABC	{A,B,C}	{A,C}

STEP-3



Q) Find the equivalent DFA for the following NFA!



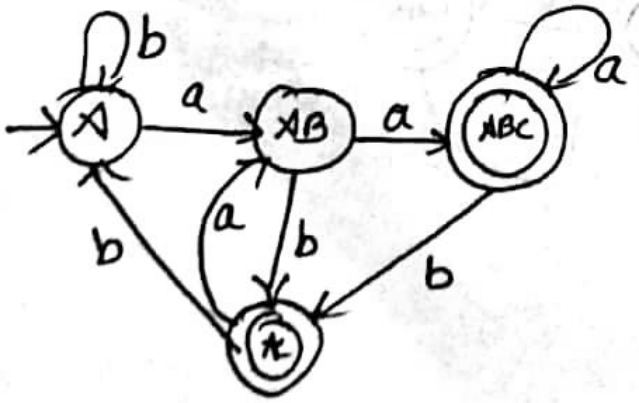
Soln:

STEP 1

States	i/p alphabets	
	a	b
A	{A,B}	{A}
B	{C}	{C}
C	ϕ { }	ϕ { }

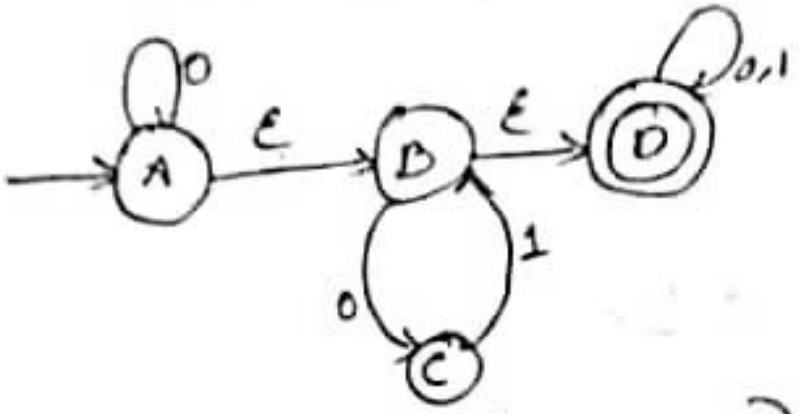
STEP-2

States	i/p	
	a	b
A	{A,B}	{A}
AB	{A,B,C}	{A,C}
ABC	{A,B,C}	{A,C}
AC	{AB}	{A}



ε NFA / NFA with ε transition or ε moves

- ε is an empty string



- It can be defined as $(Q, \epsilon, \delta, q_0, F)$
- $\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$

  From a particular state on seeing an input symbol or with out seeing something (ε), a transition occurs.
  It is also called ε-closure

ε-closure

- It is a finite collection tuple. $N = (Q, \Sigma, \delta, q_0, F)$ where

  Q :- Finite set of states

  Σ :- Finite set of input alphabets

  δ :- Transition fn.

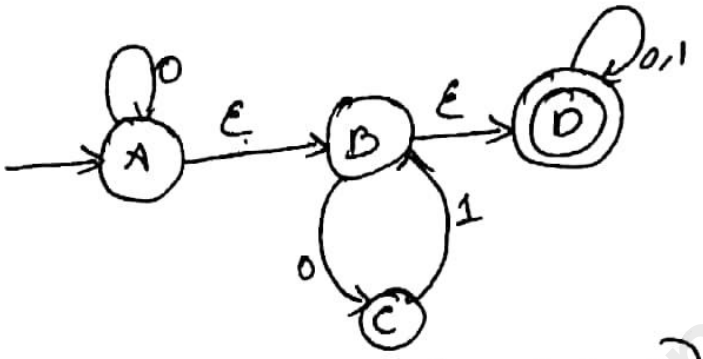
  $Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$

  q_0 :- Initial state

  F :- set of final states

ε NFA / NFA with ε transition or moves

- ε is an empty string



- It can be defined as $(Q, \Sigma, \delta, q_0, F)$
- $\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$

  From a particular state on seeing an input symbol or with out seeing something (ϵ), a transition occurs. It is also called ε-closure

- It is a five tuple (collection), $N = (Q, \Sigma, \delta, q_0, F)$ where

  Q :- Finite set of states

  Σ :- Finite set of input alphabets

  δ :- Transition fn.

  $Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$

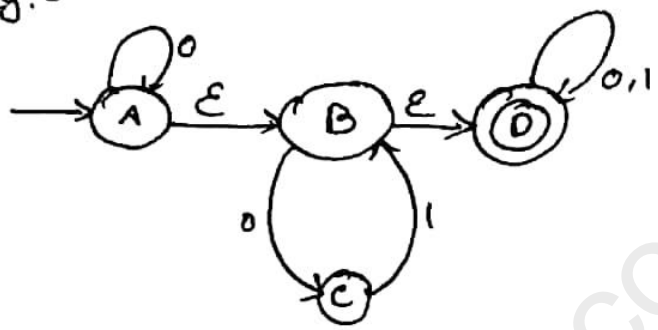
  q_0 :- Initial state

  F :- set of final states

ε-closure (ε^*)

From a particular state on seeing an input symbol or without seeing something ((ϵ) or (λ)), a transition occurs is called ε^* / ε-closure

Eg:-

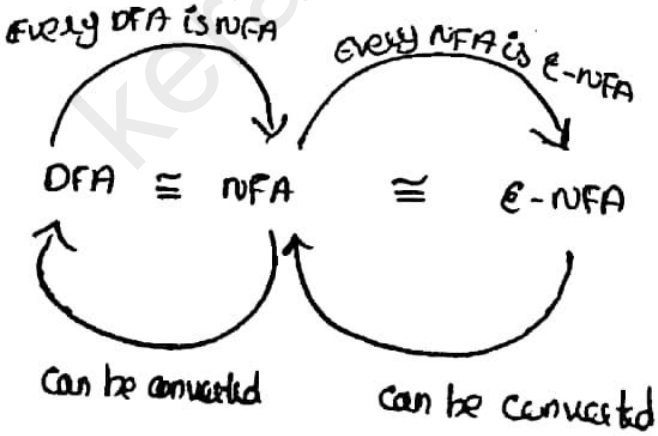


ε-closure (A) = {A, B, D}

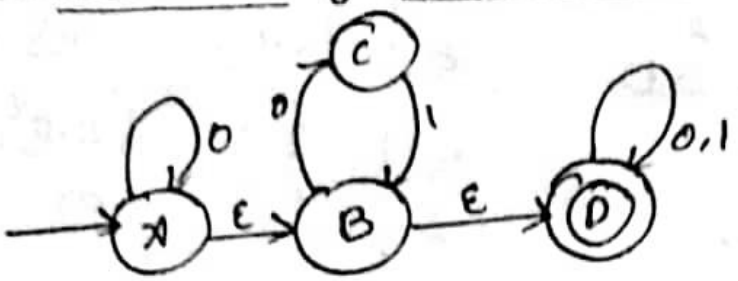
ε-closure (B) = {B, D}

ε-closure (C) = {C}

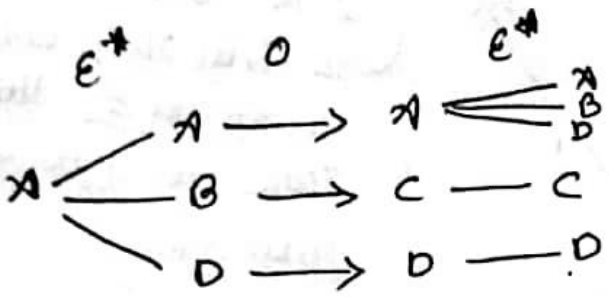
ε-closure (D) = {D}



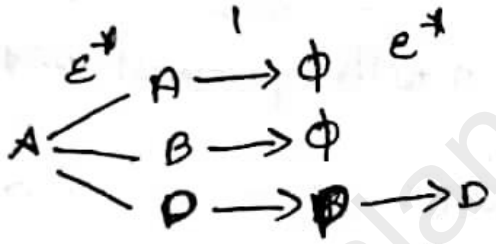
Conversion of ε-NFA to NFA / Equivalence of NFA & ε NFA



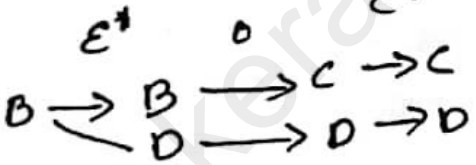
ε-closure (δ(ε-closure(A), 0))



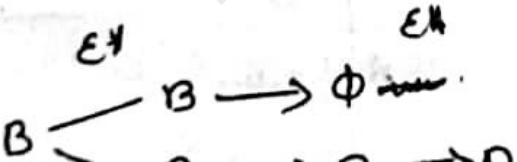
	0	1
A	{A,B,C,D}	



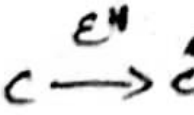
	0	1
A	{A,B,C,D}	{D}

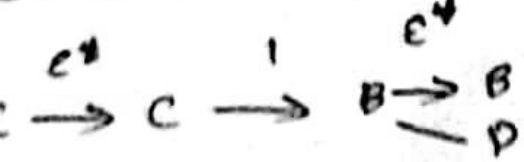


	0	1
A	{A,B,C,D}	{D}
B	{C,D}	



	0	1
A	{A,B,C,D}	{D}
B	{C,D}	{D}



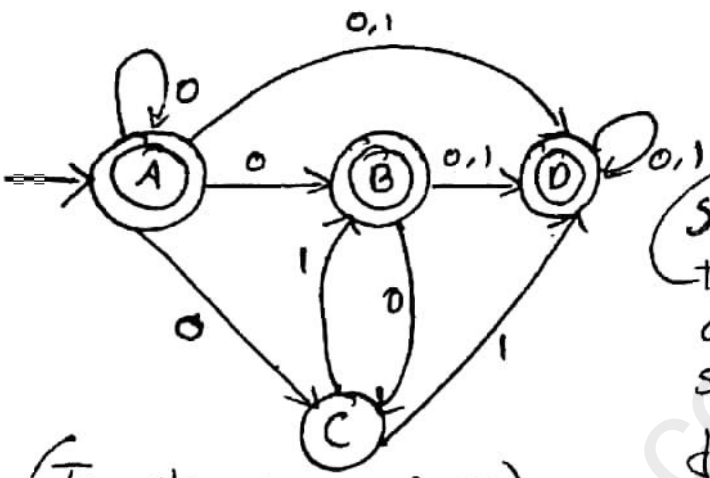


	0	1
A	{A,B,C,D}	{D}
B	{C,D}	{D}
C	ϕ	{D}

- $D \xrightarrow{\varepsilon^*} D \xrightarrow{0} D \xrightarrow{\varepsilon^*} D$
- $D \xrightarrow{\varepsilon^*} D \xrightarrow{1} D \xrightarrow{\varepsilon^*} D$

	0	1
A	{A, B, C, D}	{D}
B	{C, D}	{D}
C	{ }	{B, D}
D	{D}	{D}

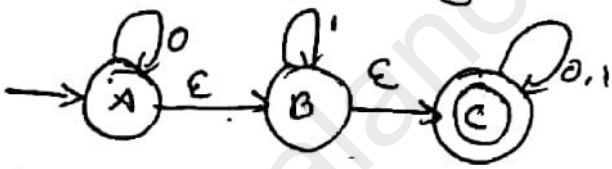
(NFA table)



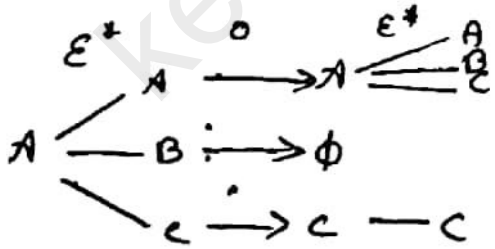
(Transition diagram of NFA)

(Some state can reach to final state only on seeing ε, that state should be a final state)

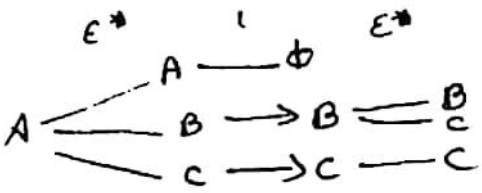
Q) Convert the following ε-NFA to its equivalent NFA ?



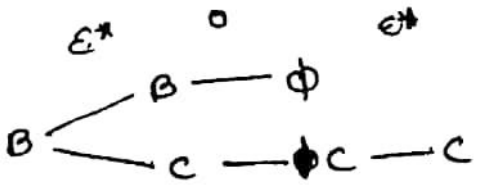
Soln:-



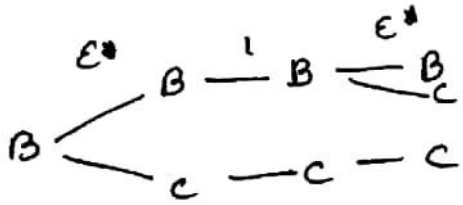
	0	1
A	{A, B, C}	



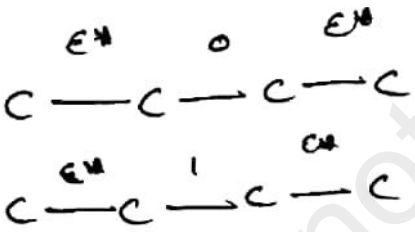
	0	1
A	{A, B, C}	{B, C}



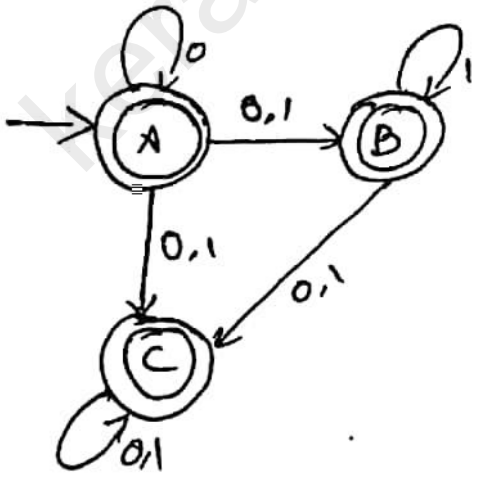
	0	1
A	{A, B, C}	{B, C}
B	{C}	



	0	1
A	{A, B, C}	{B, C}
B	{C}	{B, C}

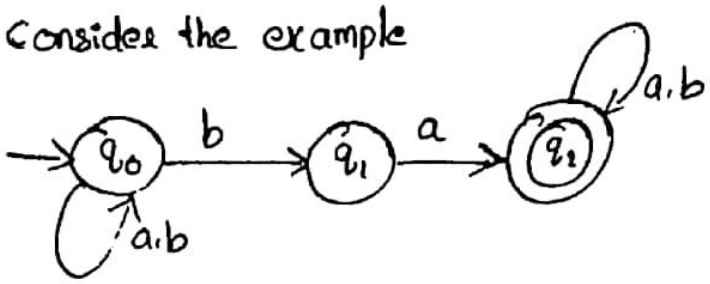


	0	1
A	{A, B, C}	{B, C}
B	{C}	{B, C}
C	{C}	{C}



Extended transition fn of NFA

Consider the example



$L = \{ xbay : x, y \in \Sigma^* \}$

~~Lets~~ consider aba string

$\delta^*(q_0, aba)$

$\delta(q_0, a) = \{q_0\}$

$\delta(q_0, b) = \{q_0, q_1\}$

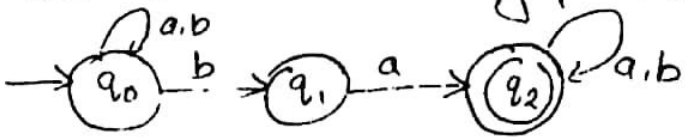
$\delta(\{q_0, q_1\}, a) = \{q_0, q_2\}$

Definition

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and $x \in \Sigma^*$, then x is said to be accepted by N iff $\delta^*(q_0, x) \cap F \neq \phi$. The set of all strings accepted by N is called the language of N, & is denoted by $L(N)$

$L(N) = \{ x : x \in \Sigma^* \text{ and } \delta^*(q_0 x) \cap F \neq \phi$

- Let take babaa string for the below figure.



$\delta(q_0, b) = \{q_0, q_1\}$

$\delta(\{q_0, q_1\}, a) = \delta((q_0, a) \cup (q_1, a))$

$= \{q_0, q_2\}$

$\delta(\{q_0, q_2\}, b) = \delta((q_0, b) \cup (q_2, b))$

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

$\delta(\{q_0, q_1, q_2\}, a) = \delta((q_0, a) \cup (q_1, a) \cup (q_2, a))$

$= \{q_0, q_2\}$

$\delta(\{q_0, q_2\}, a) = \delta((q_0, a) \cup (q_2, a))$

$= \{q_0, q_2\}$

$\delta^*(q_0, x) \wedge F$

$= \{q_0, q_2\} \wedge \{q_2\}$

$= \{q_2\} \neq \phi$

Regular Language

Let A be a DFA $A = \{Q, \Sigma, \delta, q_0, F\}$ then the language of A is the set of strings w that take the start state q_0 to one of the accepting states

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

If there is such a DFA for the language L then we can say L is a Regular language

GRAMMAR

- A grammar describes how to form strings from a language's alphabet that are valid according to the language's syntax.
- A grammar is usually thought of as a language generator

Formal definition

A grammar is a 4-tuple such that $G = (V, T, P, S)$ where.

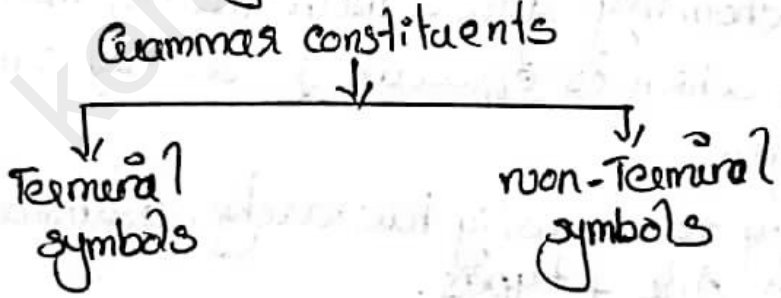
V = Finite non empty set of non-terminals symbols (variables)

T = Finite set of terminal symbols.

P = Finite non empty set of production rules

S = start symbol.

- A Grammar is mainly composed of 2 basic elements.



- Terminal symbols are denoted by using small case letters such as a, b, c etc.
- non-Terminal symbols are denoted by using capital letters such as A, B, C. It is also called variables

eg:- Consider a grammar $G = (V, T, P, S)$ where

$S \rightarrow aSbS, S \rightarrow bSaS, S \rightarrow \varepsilon$.

$V = \{S\}$

$T = \{a, b\}$

$P = \{S \rightarrow aSbS, S \rightarrow bSaS, S \rightarrow \varepsilon\}$

$S = \{S\}$

eg:- $G = (V, T, P, S)$ where

$V = \{S, A, B\}$

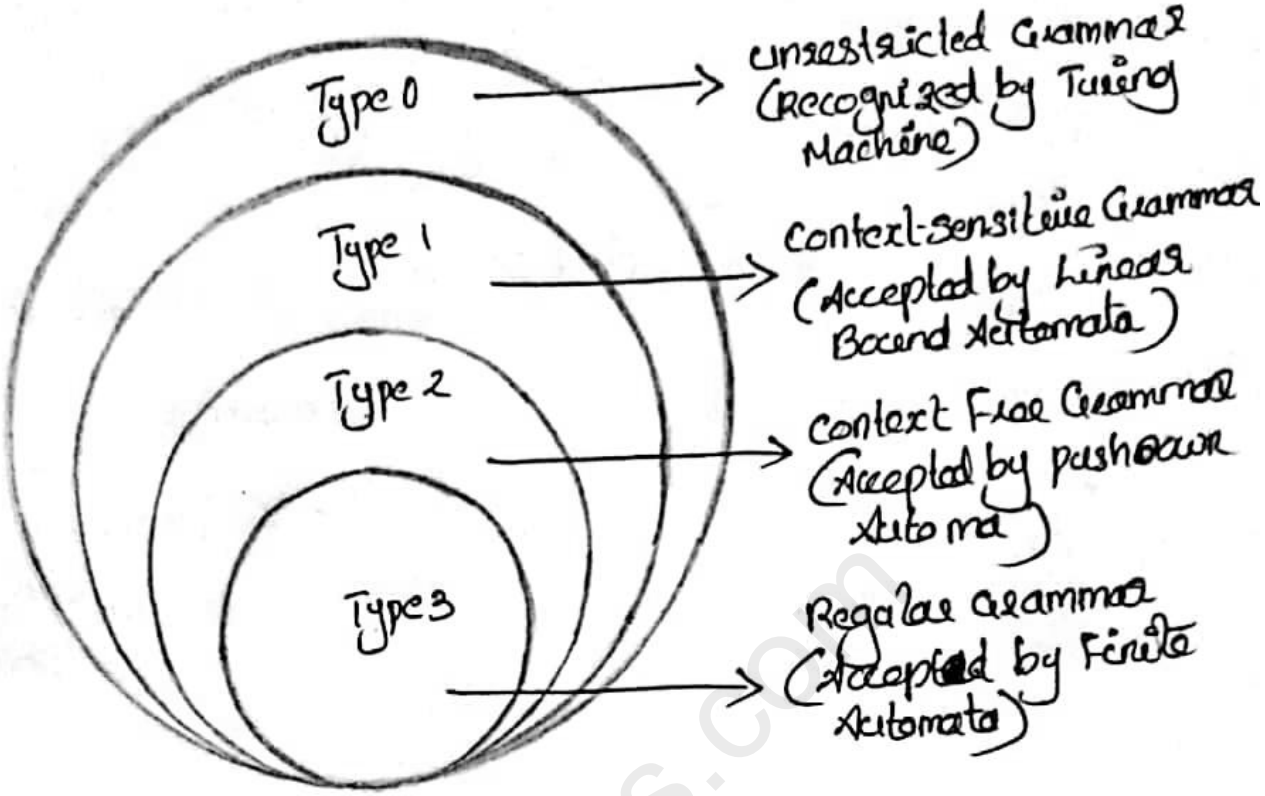
$T = \{a, b\}$

$P = \{S \rightarrow ABa, A \rightarrow BB, B \rightarrow ab, AA \rightarrow b\}$

$S = \{S\}$

Chomsky Hierarchy

- Noam chomsky gave a mathematical model of grammar in BNG which is effective for writing computer languages.
- According to chomsky hierarchy, grammars are divided into 4 types.
  - (a) Type 0 known as unrestricted grammar
  - (b) Type 1 known as context-sensitive grammar
  - (c) Type 2 known as context-free grammar
  - (d) Type 3 known as Regular grammar.



REGULAR GRAMMAR

- Regular grammars generate regular languages.
- These languages are exactly all languages that can be accepted by a finite state automaton.
- Type 3 is most restricted form of grammar.

Formal definition

A regular grammar is defined as $G = (V, T, P, S)$ where

V = Finite non empty sets of non terminal symbols (variables)

T = Finite set of terminals

P = Finite non empty set of production rules in the form $A \rightarrow aB$, and $A \rightarrow a$. where A & B are the non terminals & 'a' is the terminal.

S = start symbol.

Eg:- $G = (V, T, P, S)$ with $V = (S, B)$, $T = \{0, 1\}$ &

P is given by $S \rightarrow 0S \mid 1B$

$B \rightarrow \varepsilon$

Derivation of string 001 ?

Soln :-

$S \rightarrow 0\underline{S}$

$\rightarrow 00\underline{S}$

$\rightarrow 001\underline{B}$

$\rightarrow 001\underline{\varepsilon}$

$\rightarrow \underline{001}$

From the above example we can understand that the production is represented as

LHS $\rightarrow$ RHS

(Single non Terminal) (ε | string & Terminal, non terminal)

$A \rightarrow aBC$
$B \rightarrow \varepsilon$
$C \rightarrow b$

} Grammar

LHS ⟶ RHS

↓

(Single non Terminal) ⟶ ε
Terminal
Terminal followed by non Terminal
non Terminal followed by Terminal

⟶ Then we can say the grammar is Regular Grammar.

ie,

$A \longrightarrow \varepsilon$
$B \longrightarrow a$
$C \longrightarrow aA$
$D \longrightarrow Ab$

} Regular Grammars

Eg:- consider the grammar

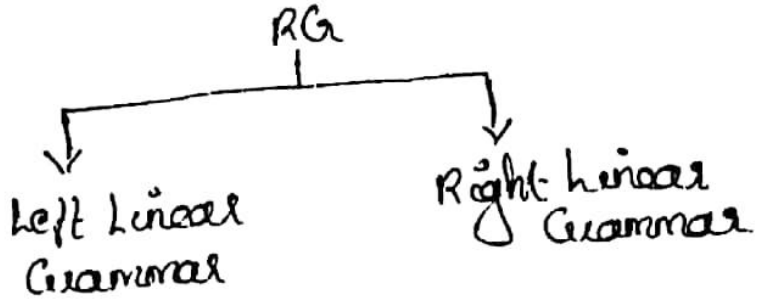
$A \rightarrow aBb$. Is it regular grammar or not?

soln:-

not a regular grammar, because a terminal is followed by non terminal & this non terminal is by Terminal.

⇒ The Regular Grammar (RG) is devided into two types

(a) Left Linear Grammar
(b) Right Linear Grammar



Left Linear Regular Grammar

In this type of regular grammar, all the nonterminals at the leftmost place.

Eg:- $D \rightarrow Ab$

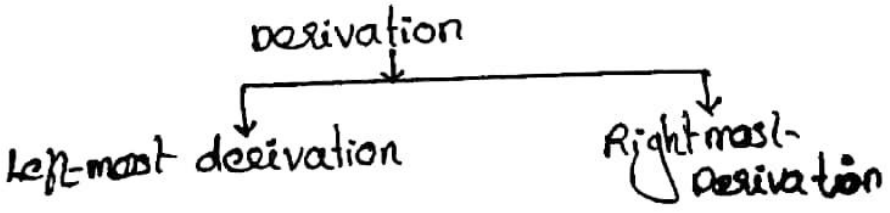
Right Linear Regular Grammar

In this type of regular grammar, all the nonterminals on the right-hand side exist at the right-most place.

Eg:- $C \rightarrow aA$

Derivations

- The process of deriving a string is called as derivation.
- The geometrical representation of a derivation is called as a parse tree or derivation tree



Leftmost Derivation

It is the process of deriving a string by expanding the left most non terminal at each step.

Right most Derivation

It is the process of deriving a string by expanding the right most non terminal at each step.

5) Find the leftmost derivation of the following Grammar?

$S \rightarrow AB \mid \epsilon$

$A \rightarrow aB$

$B \rightarrow Sb$

derive the string abb?

Soln:-

$S \rightarrow \underline{A}B$
$\rightarrow a\underline{B}B$
$\rightarrow aS\underline{b}B$
$\rightarrow a\underline{\epsilon}bB$
$\rightarrow ab\underline{B}$
$\rightarrow abS\underline{b}$
$\rightarrow ab\underline{\epsilon}b$
$\rightarrow abb$

i) Find right most derivation of the following grammar?

$S \to AB \mid \varepsilon$

$A \to aB$

$B \to Sb$

Derive the string abb?

Soln:-

$S \to AB$

$\to AS\underline{b}$

$\to A\underline{\varepsilon}b$

$\to \underline{A}b$

$\to aB\underline{b}$

$\to aS\underline{b}b$

$\to a\underline{\varepsilon}bb$

$\to \underline{abb}$

Conversion of Regular Grammar into Finite Automata

Here we want to convert $G = (V, T, P, S)$, i.e, four tuples into 5 tuples ie, $M = (Q, \Sigma, \delta, q_0, F)$

$G = (V, T, P, S) \Rightarrow M = (Q, \Sigma, \delta, q_0, F)$

$Q = V$

$\Sigma = T$

$q_0 = S$

$\delta = P$

- From this production we can create final states. suppose if a production is the form $A \rightarrow a$ then we can create a new state with transition from A to new state with label a and this new state becomes unique final states.

- Suppose a production are in the form $A_i \rightarrow aA_j$, the variables in the production are A_i and A_j becomes the states in Finite Automata 'a' is the terminal symbol, the terminal symbol is the input symbol.
  On state A_i with input symbol 'a' we are moving to state A_j

  ie, $A_i \rightarrow aA_j$

  $\delta(A_i, a) = A_j$

(9) Convert the regular grammar into finite automata?

$S \rightarrow 0A \mid 1A$

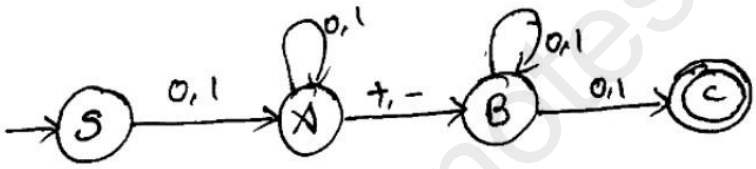
$A \rightarrow 0A \mid 1A \mid +B \mid -B$

$B \rightarrow 0B \mid 1B \mid 0 \mid 1$

Soln:-

$S \to 0A$, $\delta(S,0) \to A$
$S \to 1A$, $\delta(S,1) \to A$
$A \to 0A$, $\delta(A,0) \to A$
$A \to 1A$, $\delta(A,1) \to A$
$A \to +B$, $\delta(A,+) \to B$
$A \to -B$, $\delta(A,-) \to B$
$B \to 0B$, $\delta(B,0) \to B$
$B \to 1B$, $\delta(B,1) \to B$
$B \to 0$, $\delta(B,0) \to C$ } Final state
$B \to 1$, $\delta(B,1) \to C$ } Final state

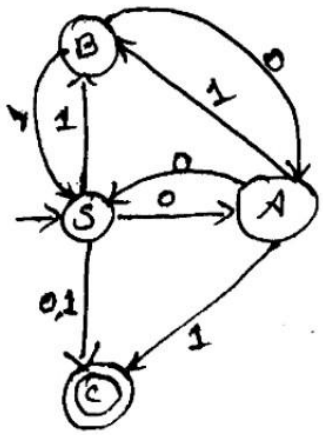
Here $Q = \{S, A, B, C\}$ & $T = \{0, 1, +, -\}$



Q) Convert the following grammar into Finite Automata?

$S \to 0A \mid 1B \mid 0 \mid 1$
$A \to 0S \mid 1B \mid 1$
$B \to 0A \mid 1S$

Soln:



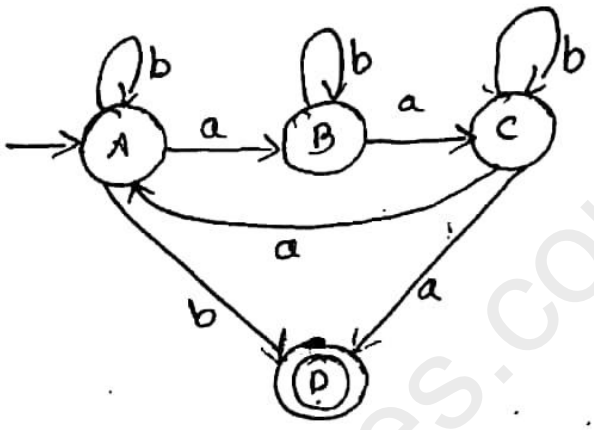
4) Convert Regular Grammar into Finite Automata?

$A \rightarrow aB \mid bA \mid b$

$B \rightarrow aC \mid bB$

$C \rightarrow aA \mid bC \mid a$

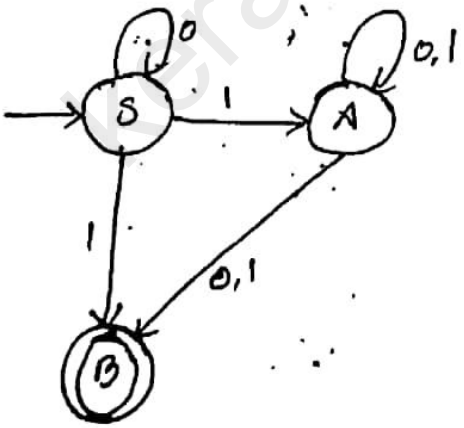
Soln:-



5) Give the automaton for the following grammar?

$S \rightarrow 0S \mid 1A \mid 1$

$A \rightarrow 0A \mid 1A \mid 0 \mid 1$



) Convert the Regular Grammar into Finite Automata?

$S \rightarrow a \mid aA \mid bB \mid \varepsilon$

$A \rightarrow aA \mid aS$

$B \rightarrow cS \mid \varepsilon$

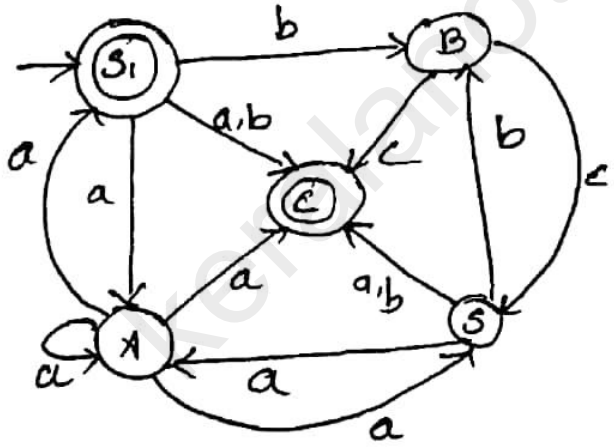
soln: -

$S_1 \rightarrow a \mid aA \mid bB \mid b \mid \varepsilon$

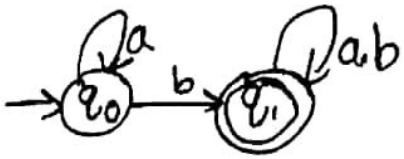
$S \rightarrow a \mid aA \mid bB \mid b$

$A \rightarrow aA \mid aS \mid a$

$B \rightarrow cS \mid c$



Q) convert the finite automata into regular grammar?



Soln:-

$M(Q, \epsilon, \delta, q_0, F) \Rightarrow G(V, T, P, S)$

$\delta(q_0, a) \rightarrow q_0$

$\delta(q_0, b) \rightarrow q_1$

$\delta(q_1, a) \rightarrow q_1$

$\delta(q_1, b) \rightarrow q_1$

$\Rightarrow q_0 \rightarrow aq_0$

$q_0 \rightarrow bq_1$

$q_0 \rightarrow b$

$q_1 \rightarrow aq_1$

$q_1 \rightarrow a$

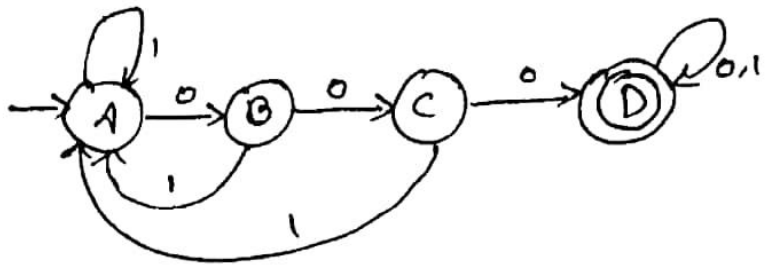
$q_1 \rightarrow bq_1$

$q_1 \rightarrow b$

$\therefore q_0 \rightarrow aq_0 \mid bq_1 \mid b$

$q_1 \rightarrow aq_1 \mid a \mid bq_1 \mid b$

Q) Convert the Finite Automata into regular grammar?



$M(Q, \Sigma, \delta, q_0, F) \Rightarrow G(V, T, P, S)$

$\delta(A,0) \rightarrow B$

$\delta(A,1) \rightarrow A$

$\delta(B,0) \rightarrow C$

$\delta(B,1) \rightarrow A$

$\delta(C,0) \rightarrow D$

$\delta(C,1) \rightarrow D$

$\delta(D,0) \rightarrow D$

$\delta(D,D) \rightarrow D$

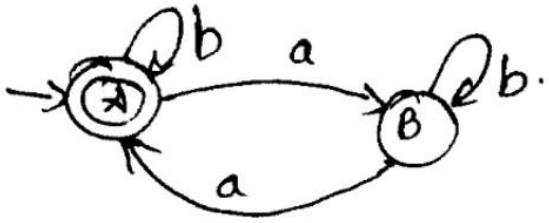
$\Rightarrow A \rightarrow 0B \mid 1A$

$B \rightarrow 0C \mid 1A$

$C \rightarrow 0D \mid 0 \mid 1A$

$D \rightarrow 0D \mid 0 \mid 1D \mid 1$

Convert the given finite Automata to Regular Grammar?



Soln:-

$A \rightarrow bA \mid aB \mid b \mid \varepsilon$

$B \rightarrow aA \mid bB \mid a$

(put an epsilon (ε) on 'A' because it's the starting & final state)