⇒ **Language:-** A language is a set of words; i.e. finite strings of letters, symbols or tokens.

↳ The set from which these letters are taken is called 'alphabet' over which language is defined.

↳ A formal language is often defined by means of a "formal Grammer" (formation Rules).

↳ Words that belong to a formal language are sometimes called "Well-formed Formulas" (WFF).

→ A Hierarchy is defined for any language (or formal language)

Alphabet → a set of symbols

$$\boxed{\{a, b\}}$$

Sentences – are strings of symbols.

$$\boxed{a, b, aa, ab, ba, abb, \ldots}$$

Language – is a set of sentences

$$\boxed{L = \{aaa, aab, abaa, bbb\}}$$

Grammer – is a finite list of rules defining a language.

$S \rightarrow aA$	$B \rightarrow bB$
$A \rightarrow bA$	$B \rightarrow aF$
$A \rightarrow aB$	$F \rightarrow \epsilon$

Strings:-

→ **Strings** "An alphabet is a non-empty finite set of symbols. denoted by Σ.

eg. $\Sigma = \{a, b, c\}$ is an alphabet.

↳ A string is a finite sequence of symbols. (U or w)

eg. $U = abcab$ is a string on $\Sigma = \{a, b, c\}$.

The empty string (no symbol at all) denoted by λ or ϵ.

- A part of string is a substring.

bca is a substring of abcab.

Note :- A beginning of a string (up to any symbol) is a prefix & an ending is a suffix.

a b c a b — prefixes.

a b c a b — suffixes.

** "A string is a prefix & suffix of itself. λ or ϵ is a prefix & suffix of any string".

⟹ **operations on strings** :-

1.) **Finding the length** : $\Sigma = \{a, b\}$

$w = abba$

$|w| = 4$

2.) **Concatenation** :

$w = abc$, $v = ab$

$wv = \underbrace{abc}_{w}\underbrace{ad}_{v}$

* → $\lambda w = w\lambda = w$.

3) Power:- $w^0 = \lambda$ (null string)

$w^1 = w \Rightarrow w^2 = ww \Rightarrow w^3 = w.w^2 = w.w.w$

$w^n = w.w^{n-1} = w.w. \text{----} w$ (n-times)

4) Reverse:- $w^R \rightarrow$ w in reverse order

$w = abc$

$w^R \rightarrow cba$

5.) Palindrome: $|w| = |w^R|$

Word & its reversal have same value.

$w = aba$
$w^R = aba$

Even Palindrome:

i.) $w = w^R$

ii) $|w|$ is even.

Odd Palindrome:

i.) $w = w^R$

ii) $|w|$ is odd.

eg $\lambda^R = \lambda$ (null) $\rightarrow$ 0 is even palindrome.

$a^R = a$ (odd(1)) $\rightarrow$ odd Palindrome.

eg. No. of palindrome of length 8 over $\Sigma = \{0,1\}$.

$\hookrightarrow 2^4 = 16$.

** No. of Palindromes of length n over $\Sigma = k$ is

$\boxed{k^{\lceil n/2 \rceil}}$.

6.) kleen Star/kleen's closure: Σ^*

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

Σ^* = the set of all strings which can be conducted by using the symbols from Σ including λ.

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

$\hookrightarrow \Sigma^* = \{\lambda, a, b, aa, bb, ba, ab, aaa, aba, -----\}$

it is a universal language.

eg. $\Sigma = \{a\}$

$a^* = \{a^*\} = \Sigma^* = \{\lambda, a, a^2, a^3, -----\}$

7.) <u>Kleen Plus/+ve closure (Σ^+)</u> ɞ The set of all strings which can be constructed using the symbols of Σ excluding λ.

$\Sigma^+ = \{a, b, aa, bb, aaa, ---\}$

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

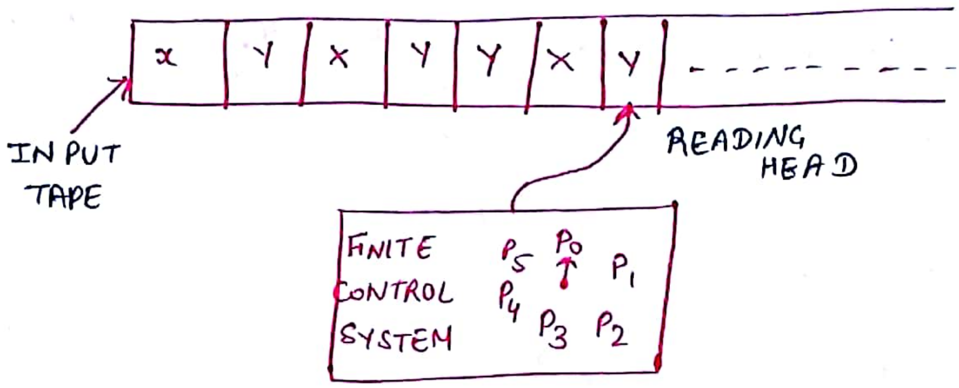
we can say,

$\Rightarrow [\Sigma^+ \cup \{\lambda\} = \Sigma^*] \Rightarrow \{\Sigma^* \cup \{\lambda\}\} = \Sigma^*$

FINITE AUTOMATA

→ Finite Automation is called "finite" because no of possible states and no. of letter in the alphabet are both finite and "automation" because the change of the state is totally governed by the input.

It is deterministic, what state is next is automatic not will-full, just as the motion of the hands of clock is automatic, while the motion of hands of a human is presumably the result of desire and thought.



Here, $P_0, P_1, P_2, P_3, P_4, P_5$ are states in Finite Control system

x and y are input Symbols.

→ At regular interval the automation reads one symbol from the input tape and then enters in a new state that depends only on the current state and the symbol just read.

→ After reading an input symbol, reading head moves one square to the right on the input tape, so that on the next move, it will read the symbol in next tape square. Repeat it again and again.

The automation then indicates approval or disapproval.

Sometimes, FA (Finite Automata) is ...

↳ If it winds up in one of a set of final states the input strings is considered to be accepted.

The language accepted by the machine is the set of strings, it accepts.

Definition :-

<u>DETERMINISTIC FINITE AUTOMATA (DFA):</u>

A deterministic Finite Automata is a quintuple

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

where,

- Q : is a non-empty finite set of states presents in finite control. $(q_0, q_1, q_2, \ldots)$
- Σ : is a non-empty finite set of input symbols which can be passed to finite state machine. $(a, b, c, \ldots)$
- q_0 : is a starting state, one of the state in Q.
- F : is a non-empty set of final states or accepting states, set of final states belongs to Q.
- δ : is a function called transition function that takes two arguments a state and a input symbol, it returns a single state. $\boxed{\delta : Q \times \Sigma \rightarrow Q.}$

Let 'q' is the state and 'a' be input symbol passed to the transition function as:

$$\delta(q, a) = q'$$

q' is output of the function.

A single state q' may be q. It can be:

$$\boxed{\delta(q,a)} \longrightarrow q' \quad (\checkmark)$$

but q' may be same as q. $(q' = q)$

$$\boxed{\delta(q,a)} \begin{cases} \rightarrow q' \\ \rightarrow q'' \end{cases} \quad (\times)$$

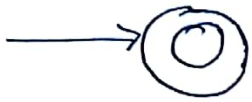
Sometimes, FA (Finite Automata) is also called as '<u>Language Recognizer</u>'.

TRANSITION DIAGRAMS:

1.) <u>Initial State:</u>

"Accepting Nothing".

ii.) <u>Final State:</u>



"accepting Everything".

2.) An alphabet Σ of possible input letters from which input strings are formed.

3.) A finite set of transitions (labelled edge) that show how to go from some states to some others, based on reading specified substrings of input letters (Null string).

Design a DFA which accepts strings that ~~Starts~~ with a over $\Sigma = \{a, b\}$.

<u>Step 1:</u> A DFA is a quintuple:

$\Rightarrow (Q, q_0, \Sigma, F, \delta)$ where, $Q = \{q_1, q_0\}$

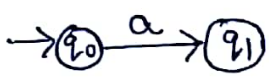
$q_0 = q_0$

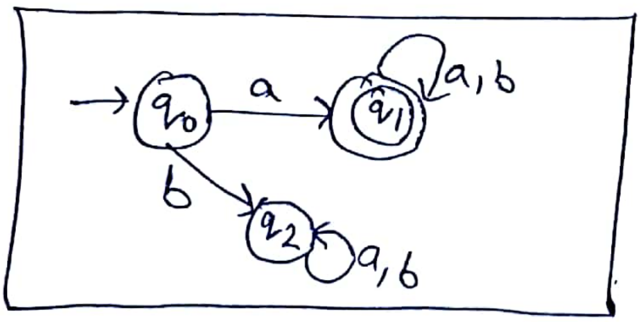
$F = q_1$

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

$L = \{a, aa, ab, aba, \ldots\}$

<u>Step 2:</u>





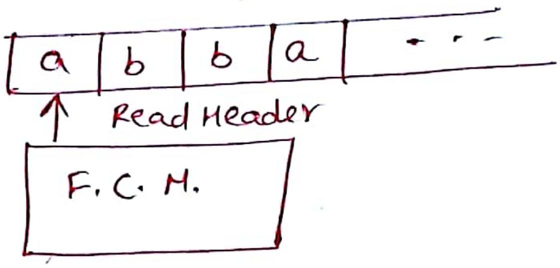
<u>TRANSITION DIAGRAM</u>

with ω.

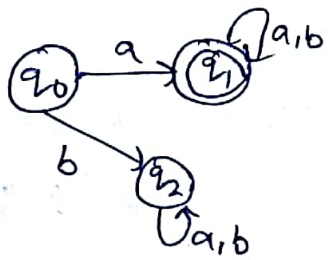
Step 3: Transition Diagram: we have $\Sigma = \{a, b\}$

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

Step 4: A string ex. 'abba' is accepted by this DFA or Not:



Using δ :-



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

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

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

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

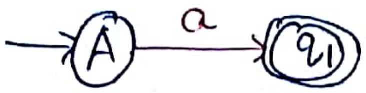
So, final transition state is 'q_1' & 'q_1' in transition diagram is Final State. Hence, the string 'abba' is accepted by DFA.

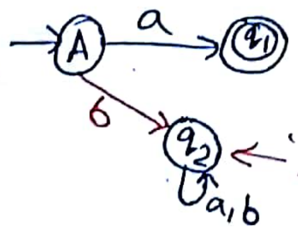
Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must start with w.

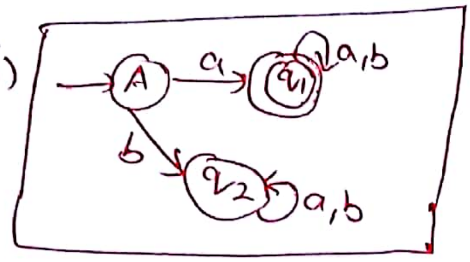
1.) $w = $ 'a'

~~Step~~ Sol^n Possible Language;

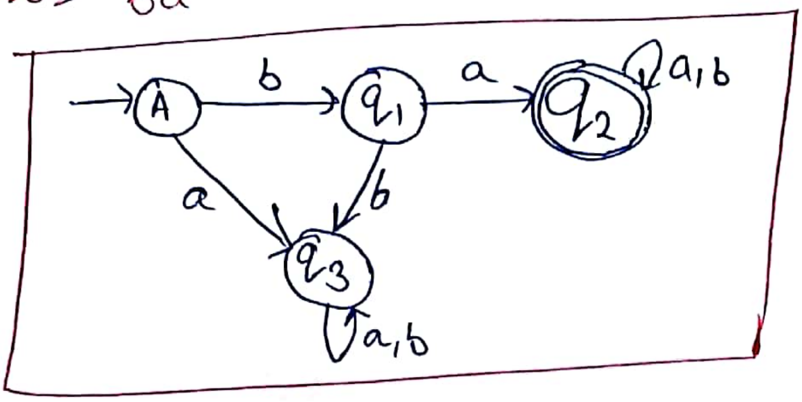
$L = \{a, aa, ab, aaa, \text{------}\}$

i.) 

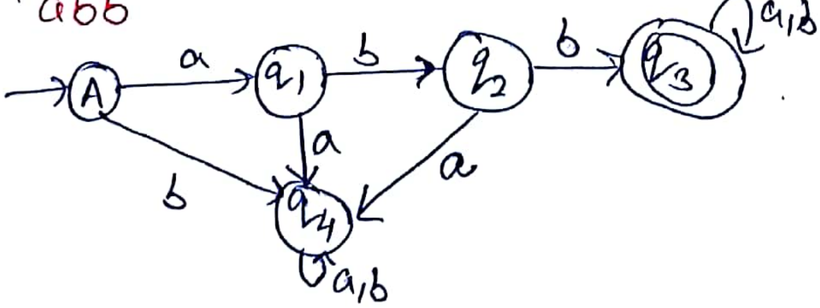
ii.) 

iii) 

II) $w = $ 'ba'



III $w = $ 'abb'



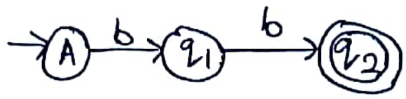
* **NOTE:-** $(n-2)$ states in MDFA.

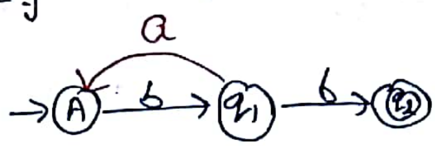
start and ends with same Symbol.

Design a MDFA over $\Sigma=\{a,b\}$ such that every string accepted must ends with a substring w.

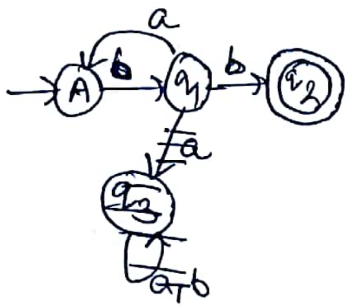
i.) w='bb'

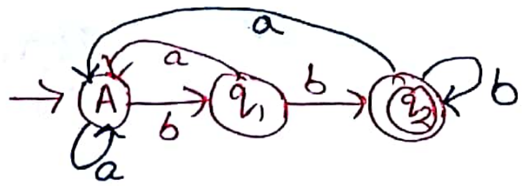
L={ bb, abb, aabb, bbb, ----}

1.) 

3.) 

No Need of dead state.

2.) 

4.) 

PRACTICE PROBLEMS

TRY IT YOURSELF!

① Design a DFA over $\Sigma=\{a,b\}$ such that every string accepted must start with a substring w.

i.) w = ba   ii) w = abb

② Design a DFA over $\Sigma=\{a,b\}$ Such that every string accepted must ending with a substring w.

i.) w = bb   ii) w = ab   iii) w = bab

③ Design a DFA over $\Sigma=\{a,b\}$ such that every string accepted contains a substring w.

i.) w = aa   ii) w = ba   iii) w = bb

1.) Design a DFA over $\Sigma=\{a,b\}$ such that every string accepted must starts and ends with 'a'.

5.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must start and ends with same symbol.

6) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must start with 'a' ending with 'b' and vice-versa.

7.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must start with 'aa' or 'bb'.

8.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must ends with 'aa' or 'bb'.

9.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must contains a substring 'aa' or 'bb'.

10.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must

(a) $|w| = 2$ b) $|w| \geq 2$ c.) $|w| \leq 2$

d.) $|w|_a = 2$ e.) $|w|_a \geq 2$ f.) $|w|_b \leq 2$.

11.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must

a.) $|w| = 2 \pmod 3$ b.) $|w| = 3 \pmod 4$

c.) $|w| = 1 \pmod 5$ d.) $|w|_b = 2 \pmod 3$

e.) $|w|_a = 3 \pmod 4$ f.) $|w|_b = 1 \pmod 5$