Lec-5

		English	New language.
Symbol	Basic Building blocks which can be any character or token.	a b .... z A B .... Z	{0, 1} 0.
Alphabet	Finite non empty Set of symbols	{a, b, ... z}	Σ = {0, 1}
String / word	finite seq^{n} of Symbols.	boy boundary	w = {001, 100} ON OFF
Language	Set of strings		

* How many symbols will be there in a language?
Who defines it? / Who decides that?
What is the rule?

No rule there are predefine.

* How much info a symbols holds?

If we represents a by new Symbols then.
New symbol also holds the same info that a contains.

* If instead of a we declare new will there be any difference?

No, difference.

* We have taken English because it is well understood by most of the people.
* More Symbols means more possibility.
* Symbols are known as letters.
* Alphabet is a collection of letters.
* Total no of Symbols in a language is finite.
We do not have any lang^{n} which do not have finite set of Symbols.

* We can not imagine a language which do not have a ~~non~~ symbol. That's why an alphabet set is non empty. If we do not have a letter then we cannot design a word. If we cannot design a word then we cannot design a sentance. Then, every thing of these lang^n is virtual.

* Set is the unordered collection. Set cannot have duplicacy.

* We have a grammar to define spelling a word but we do not have any grammar through which we can define the correct spelling of a word.

~~H~~ere Ram is the correct spelling because of its def^n in the Oxford English dictionary.

But ∃ no dictionary which will reference as ~~absolute correct~~ that Ramay is not the correct spelling.

* When we start writing a sentance, then we use the grammar otherwise not.

* Technical L is a set of strings.
Can a L is ~~infinite~~ empty or Φ.
Answer is yes.

Lec 6

Methods to study language.

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

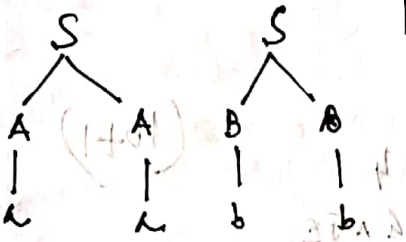
$L = \{aa, bb\}$

By G

$S \rightarrow AA / BB$

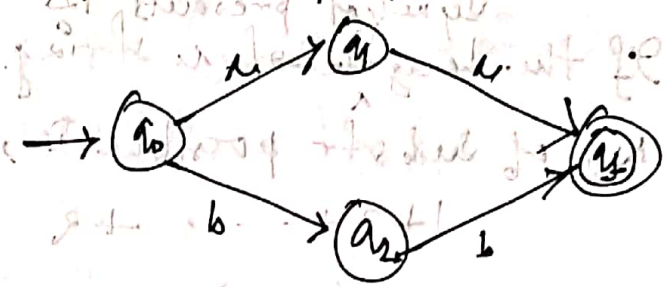
$A \rightarrow a$

$B \rightarrow b$



Generating Device

By M/C



Accepting Device.

Lec 7

Some basic operations on strings: (w)

① Length of String: $|w|$ (no of symbols that it have)
it may not be unique

② Concatenation of string: $w_1 \cdot w_2$

③ Reverse of a string: $w = 'abc'$ $w^R = 'cba'$

④ Prefix & Suffix: abaa $P(w) = \{\varepsilon, a, ab$
$S(w) = abaa, aba$
$\{\varepsilon, a, aa, baa, abaa\}$

⑤ Substring:

1. ε = String of length 0. $|\varepsilon| = 0$
2. ϕ = Empty set.
3. $|w_1| + |w_2| = |w_1 \cdot w_2|$
4. $w_1 \cdot w_2 \neq w_2 w_1$
5. $|w| = |w^R|$
6. $|w| = 2$.

prefix = 2+1
Suffix = 2+1

$w = abbcd$

abd, abbc, bcd, cb

bc ✓ Substr.

Contiguous Seq^{n} of Symbols possible.

Symbol presents in a string is unique then,

If the ~~length of~~ string is n

No of substr possible is,

$$1+2+\cdots+n+1$$

$$= \frac{n(n+1)}{2} + 1$$

$$= \frac{(n+1)}{2}(n+1)$$

	GATE			
length 0	1	2	3	4
String ε	G	GA	GAT	GATE
	A	AT	ATE	
	T	TE		
	E			

Lec 8:

Kleene Closure & + Closure. (Σ^*, Σ^+)

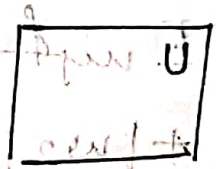
A L is a set of strings. i.e. the operations applicable for a language is now applicable for ~~set of~~ FL.

a) Complement

b) Intersection

c) Union

$L = \{\ \}$



$2^3 = 2 \times 2 \times 2 = 2 \cdot 2 \cdot 2$ (That symbol 2 is concatenated 3 times with itself).

$a^2 = a\,a$

$x^0 = 1$ but $x^0 = \varepsilon$ (zero repetitions)

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

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

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

$\Sigma^2 = \{aa, ab, ba, bb\}$

$\Sigma^k = \{w \mid |w| = k\}$

$\Sigma^\infty = \{w \mid |w| = \infty\}$

$\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots \cup \Sigma^k \cup \cdots \cup \Sigma^\infty$

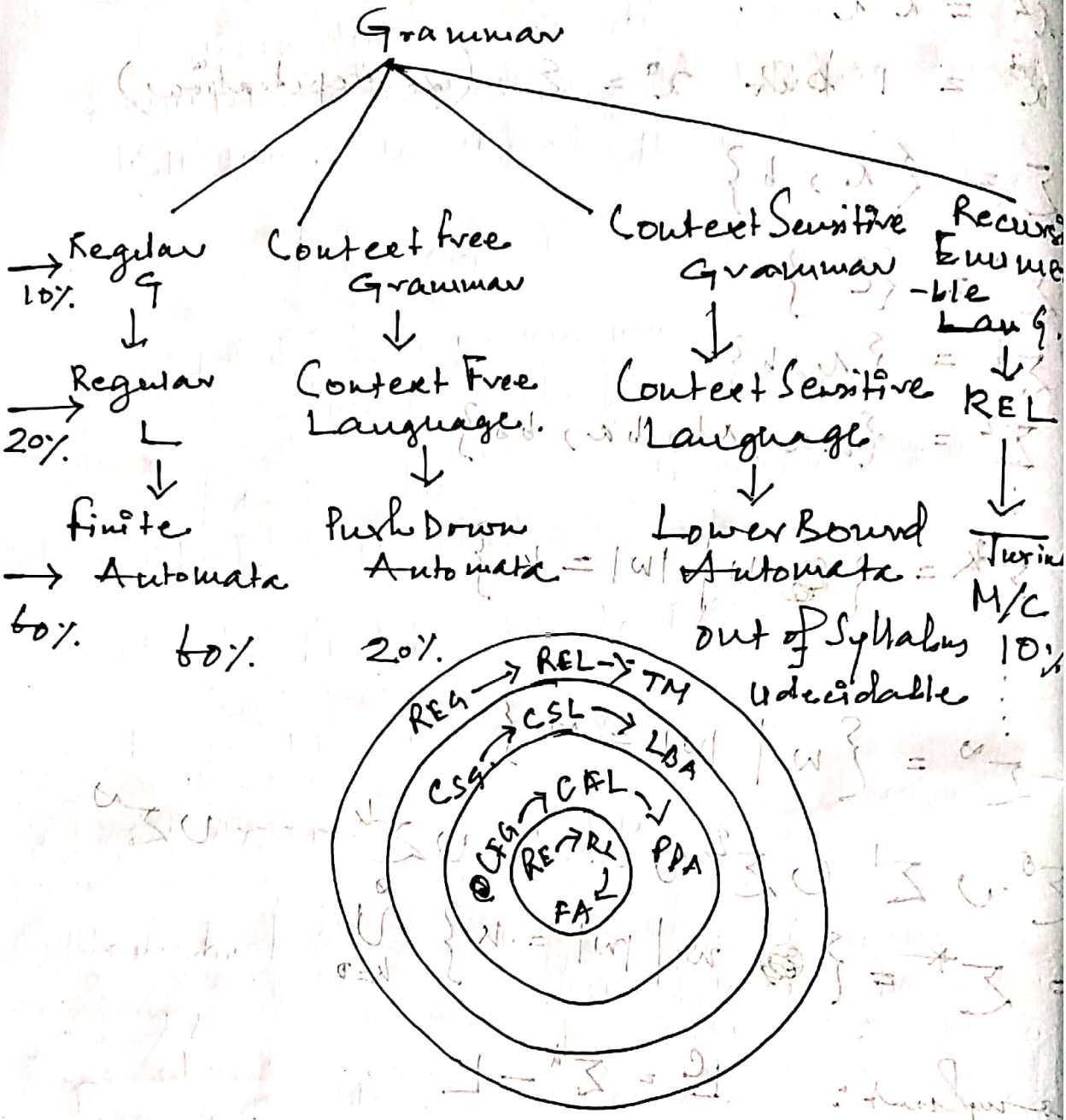
$= \Sigma^* = \{w \mid |w| = k\} \bigcup_{k=0}^{\infty}$

Complement: $L^c = \Sigma^* - L$

$\Sigma^+ = \bigcup_{k=1}^{\infty} \{w \mid |w| = k\}$

$= \Sigma^1 \cup \Sigma^2 \cup \cdots \cup \Sigma^k \cup \cdots \cup \Sigma^\infty$

LECTURE-9 Chomsky Hierchary



We mejorly concentrate on Automata/ Machine.

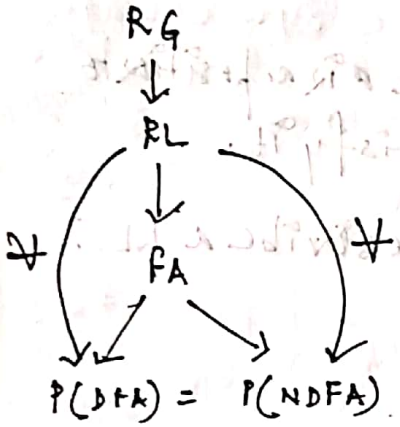
10% questions from NP - P - NP hard - NP Completeness, reducability, undecidability & decidability.

We cannot understand PDA before FA because the subject is based on hierarchy. So, before moving to PDA, we need have a full concept on FA & RL

∴ FA + Stack = PDA also FA + Something = TM.

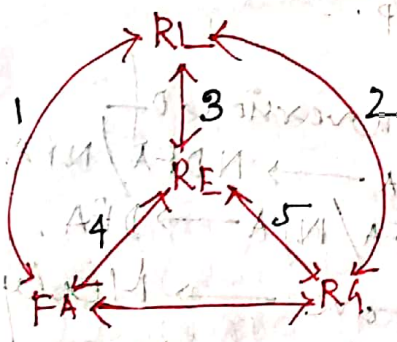
LEC: 10 Topics regarding FA, RG, RL.

Flowchart 1



1. A RG creates a RL.
2. A RL is accepted by a FA.
3. FA can be of two types.
   a. DFA.
   b. NDFA
4. Power of DFA & Power of NDFA are exactly same.
5. For any regular language, we can create a DFA then we can also create a NDFA.
6. Every DFA can be converted to NFA.
7. Every DFA is also NFA (NDFA).
8. We prefer to design NDFA / NFA because they are easy to design.

Flowchart 2



1. For what kind of languages we can design a FA.
2. We must be in a position to tell,
   a. if somebody gives us any language then we must be capable of designing a FA.
   b. if somebody gives us any FA then we must be capable of telling the RL

2. * if anybody gives us any RL then we must be in a position to tell the R.E. Grammar for it.

* if anybody gives us any RG then we must be in a position to tell that the RG will generate what type of RL.

3. RE → ∃ an expr to describe a RL.

if ∃ a RE then we must be in a position to design the RL which satisfy it.

if ∃ a RL then ..... to describe a RL.

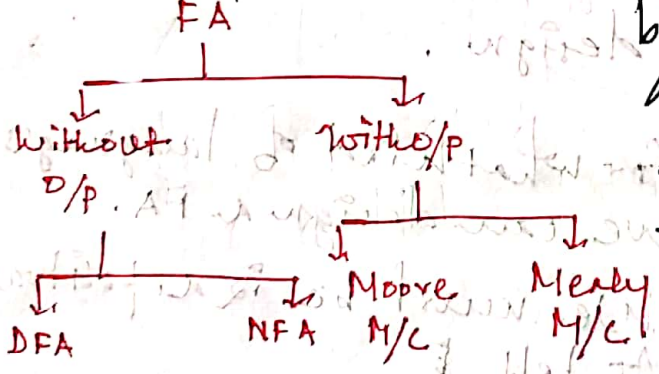
~~which~~ :

4. FA to RE & RE to FA.

5. RG to RE & RE to RG.

6. FA to RE & RG to FA.

Flowchart 3



1. if ∃ a FA then we must know that it is basically a language acceptor. Not generator. So FA is mostly with o/p.

2. Conversion of
DFA → NDFA/NFA
NDFA/NFA → DFA.

3. Moore → Mealy
Mealy → Moore.

4. Advantages & Disadvantages.

LEC 11 : FA (DFA & NDFA).

1. Mathe Def^{n} of DFA $\longrightarrow$ DFA $= \{Q, q_0, \Sigma, F, \delta\}$

$Q \longrightarrow$ Total no. of States. (No$_2$ — empty)

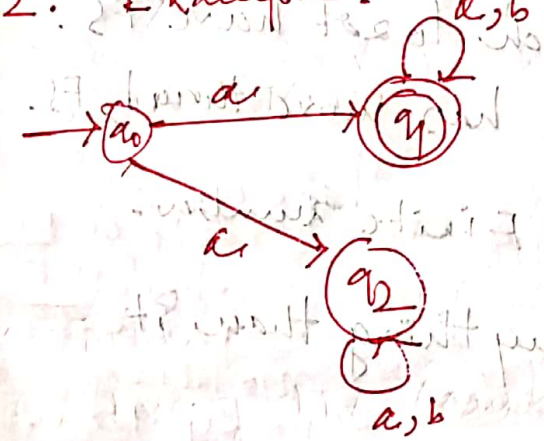
$q_0 \longrightarrow$ Initial State.

$\Sigma \longrightarrow$ Set of input Symbols / Alphabet.

$F \longrightarrow$ Set of finite final states.

$\delta \longrightarrow$ Transition Function.

2. Example:



1. Q = total no of States.
$= \{q_0, q_1, q_2\} = 3$.
= Finite.

2. States basically represents memory.

3. For ex: If anybody asks me which is the last symbol that we scanned while coming to q_1.
we can tell which is 'a'.

4. FA does not have any auxiliary memory. It only remembers anything by changing State.

5. q_0 = finite non-empty set of Is $= \{q_0\} = 1$

a. initial state will be 1.

b. It can be represented by ~~either~~ any symbol ~~that~~ we like.

c. Initial state can be identified by.
1 $\longrightarrow$ 1 symbol. (incoming edge)

3. Big Sigma (Σ) = Set of finite non empty Set of i/p alphabet/Symbol upon which m/c works.

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

a. Set of i/p Symbol is finite.

b. " " " " should be non empty.

c. We require atleast one symbol/alp

4. F = Set of final State

a. $\exists$ a DFA possible which do not have FS

b. $\exists$ a DFA " " has more than 1,

Here, $F = \{q_4\} = 1 =$ Finite number.

∴ if a DFA do not accept anything than it accepts empty L.

c. Initial State may also be FS.

e. All State of FA may be FS.

f. Total no. of FS. Range is,

$$0 \leq |F| \leq |Q|$$

5. δ = Transition Function.

Def? $\delta : Q \times \Sigma \to Q$

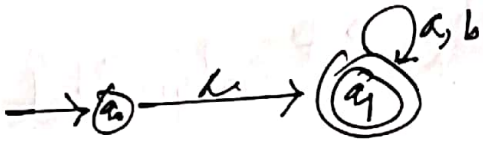
Ex: $Q_2 \{q_0\}$ State if I take i/p Symbol a (say) then I reach state q_4 which is FS also in Q.

Transition function will helps you to understand how does a m/c program. or in which fashion a m/c progress.

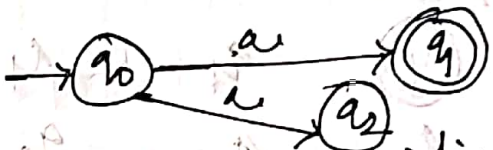
* Deterministic :

1. whether your future behaviour is predictable or not.

If I know the PS & I know the i/p then I can determine what will be the NS.



Deterministic



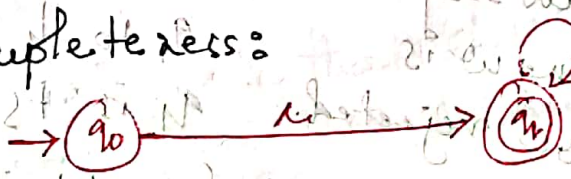
Non-Deterministic

- Every H/C that we see either h/w or S/w are D H/C
- Human are Non-Deterministic M/C.
- Non-Deterministic m/cs are easier to design compare to Deterministic M/C.
- Non-Deterministic M/C we need to draw theoritically, but there practical implementation may not be needed.

* How can we guarantee that the m/c is deterministic?

if in a state q_0 for an i/p value 'a' I only one transition.

* Incompleteness:



No. Of any state for every i/p Symbol the m/c will response. otherwise it is incomplete.

Properties of DFA : (should be)

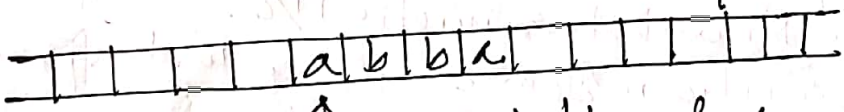
1. DFA must be always complete.
2. It must responses to a i/p Symbol.
3. Only one transition for every i/p Sym

Transition Table :

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

Block Diagram :

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

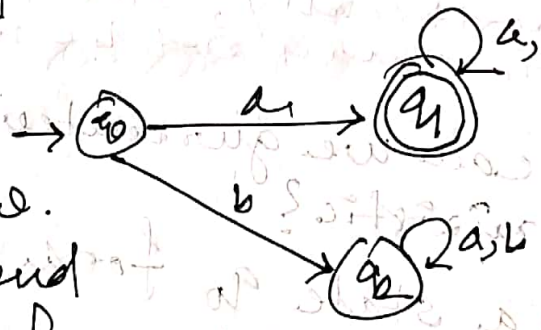


Read Header

Finite Control Unit (FCU)

Tape divided into cells each cell capable of holding one symbol at a time.

Accepting State :



After parsing a string w. and reaching at the end of the string if we reach the FS then we can say that the string w is accepted otherwise rejected.

q_0 to a q_1
q_1 to b q_1
q_1 is FS.

So, ab is accepted.

Similarly 'ba' is rejected.

LEC-12: DFA DESIGNING.

Q: Design a MDFA over $\Sigma = \{a, b\}$ s.t. every string accepted must start with a.

(i) $w = 'a'$

(ii) $w = 'ba'$

(iii) $w = 'abb'$

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

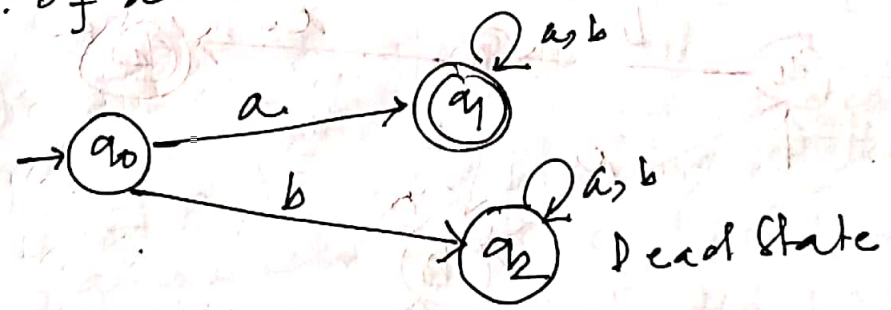
* MDFA → Minimal Deterministic Finite Automata.
* By minimal we can understand that the m/c has minimal no. of states.
* MDFA is a separate topic which we discuss later.
* All the DFA that we are going to design contains minimal no of states.

Case 1: $w = 'a'$ Every state that we are accepting is started with a.

Then the language will be,

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

No. of words infinite ∴ L is infinite.



Dead State:

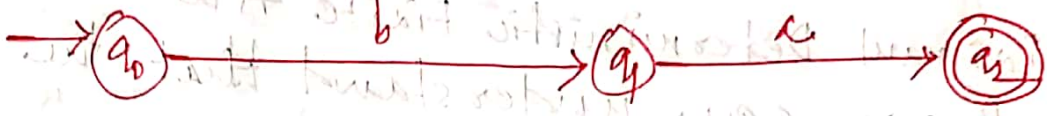
Dead State is that state if we reach that state then we cannot reach from that state to FS.

Case II:

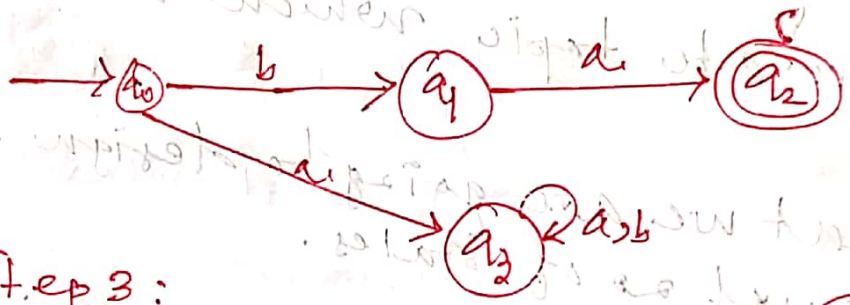
Any string should be accepted if the string is started with 'ba'

$$L = \{ ba, bab, baa, \ldots \}$$

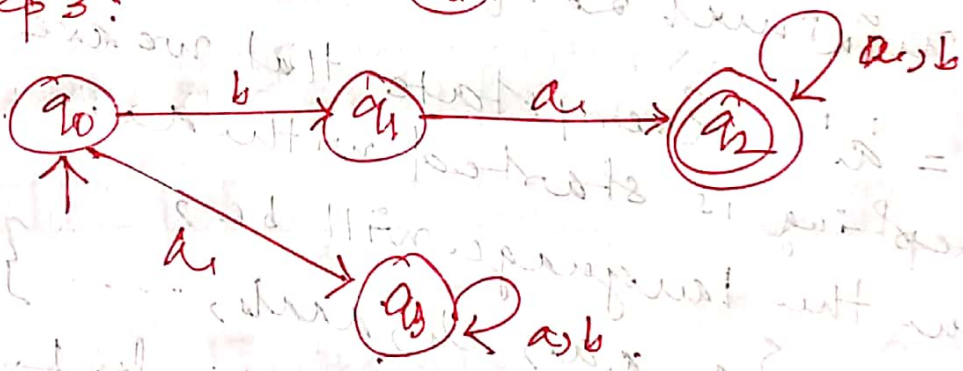
Step 1:



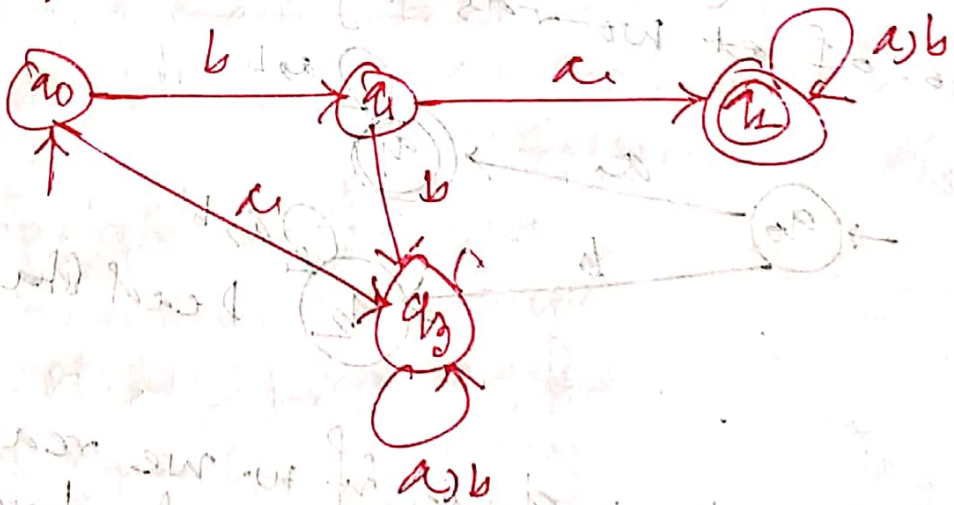
Step 2:



Step 3:



Step 4:

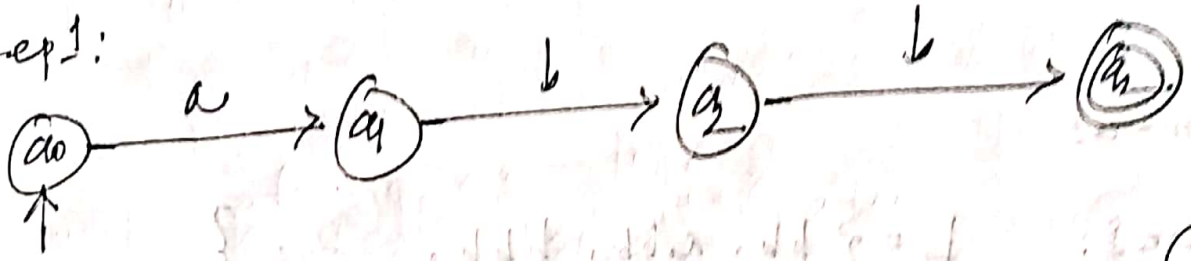


Case III:

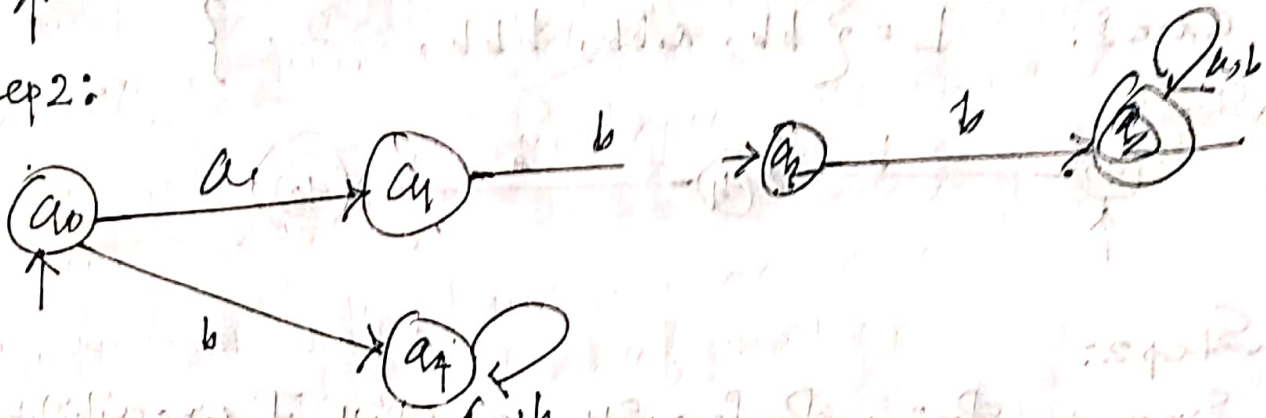
Any string should be accepted if the string is started with 'abb'

$L = \{abb, abba, abbb, abbaa, abbba, \ldots\}$

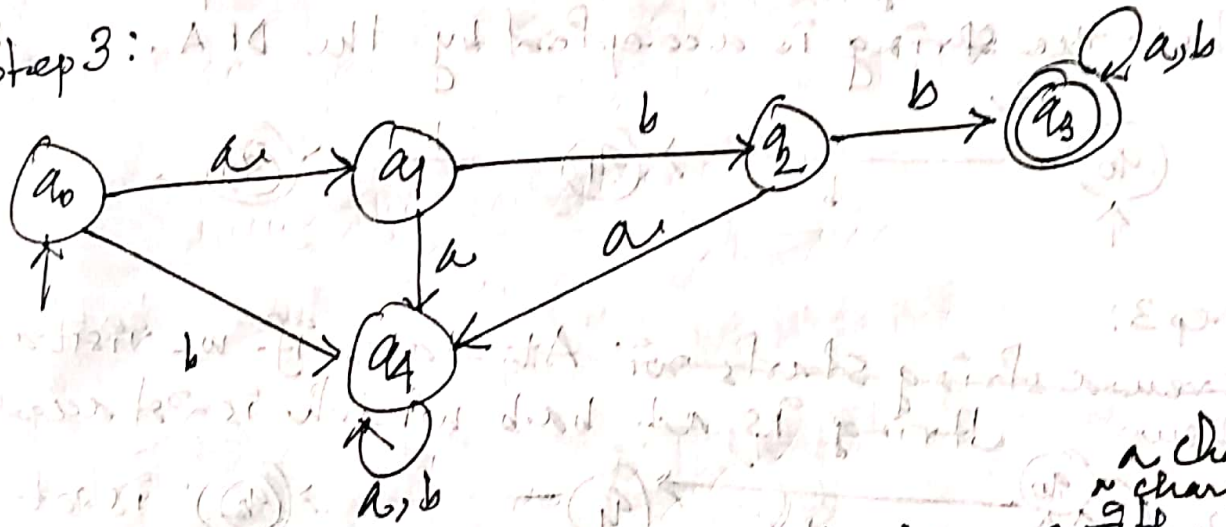
Step 1:



Step 2:



Step 3:



Note: if $\exists$ a DFA and starting with 'x' a char & chars. So, and ~~a string~~ there how many state will be there ≥ 2 the MDFA?

$2+2$

if $\exists$ a string $\neq 0$ with length 2 then, $\exists$ 2+2 states.

Q: Design a DFA over $\Sigma = \{a, b\}$ s.t. every string accepted must ends with a substring w.

a. $w =$ "bb"

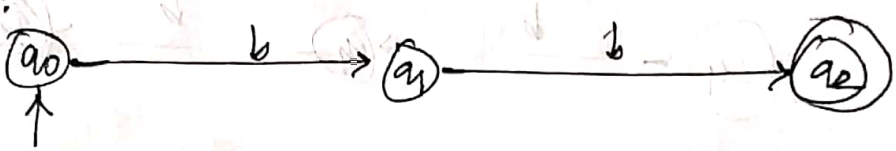
b. $w =$ "ab"

c. $w =$ "bab"

Answer:

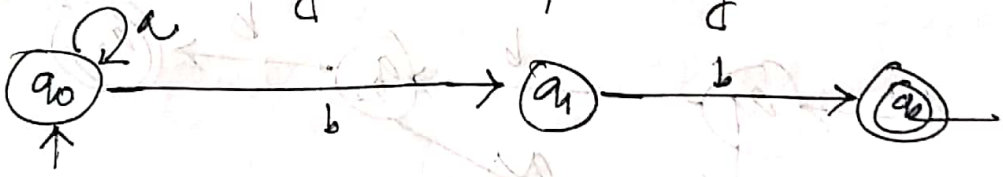
Case I: $L = \{bb, abb, bbb, \ldots\}$

Step 1:



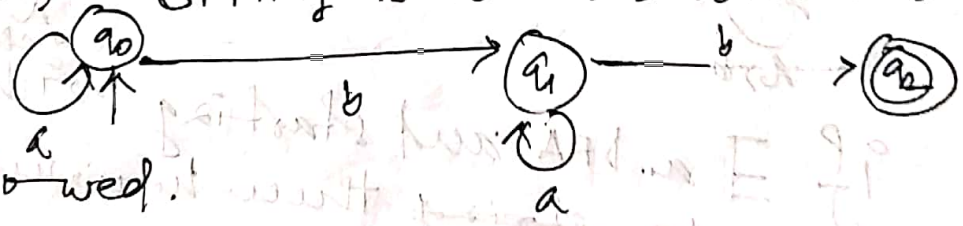
Step 2:

Even a string starts with a split if possibility that the string is accepted by the DFA.



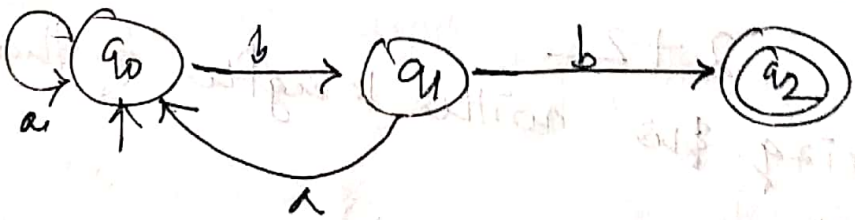
Step 3:

~~Even a string starts with~~ At state if we visit then string is ~~at~~ bab which is ~~not~~ accepted by



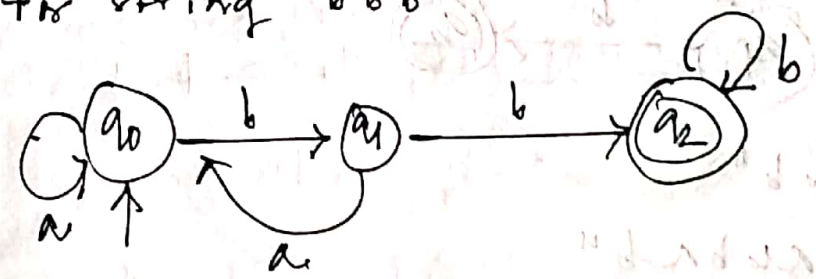
is not allowed.

Now, what about, babb?

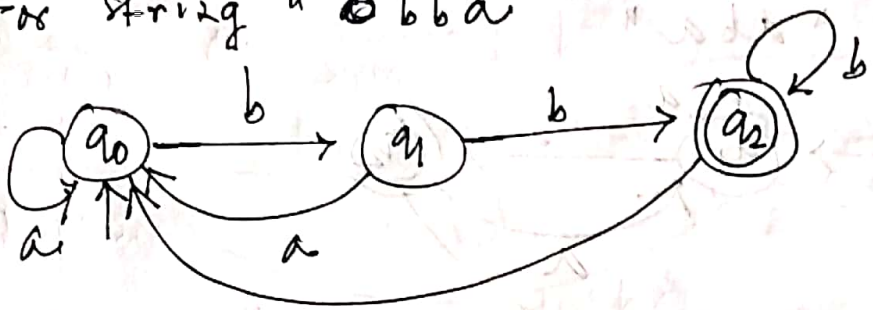


Step 4:

For string "bbb"



For string "bbba"

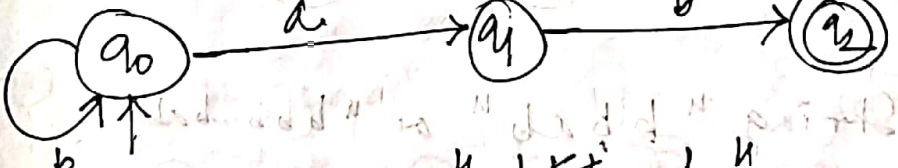


For string "bba" & "babb" there is a possibility of dead state.

Case II: • $L = \{ab, bab, aiab, baiab, \cdots\}$

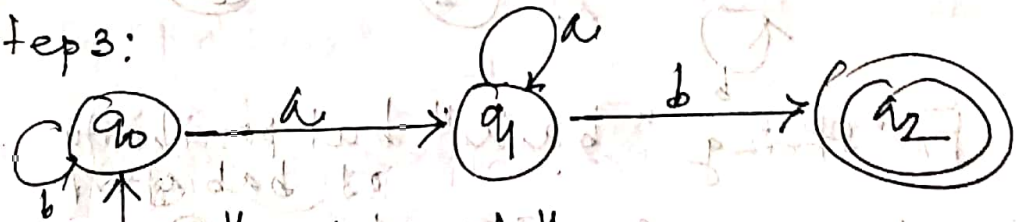
Step 1:

Step 2:



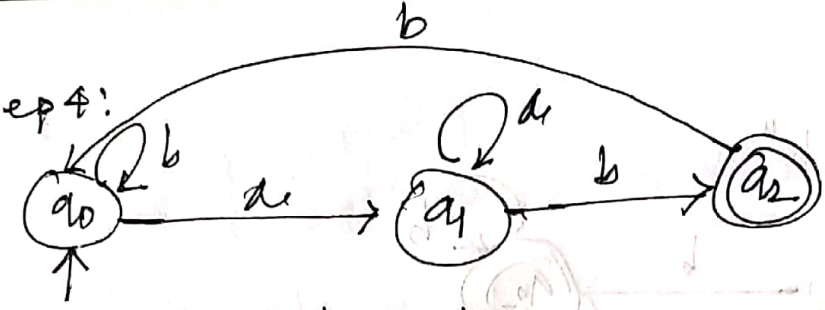
For string "babab".

Step 3:



For string "aaaaab"

Step 4:

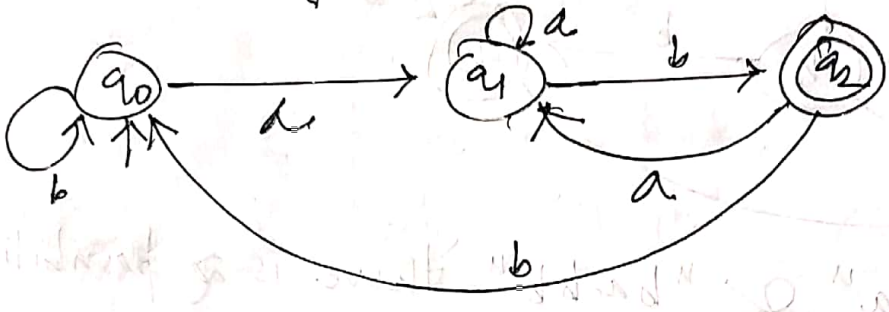


For string "abb"

For string "abbabb"

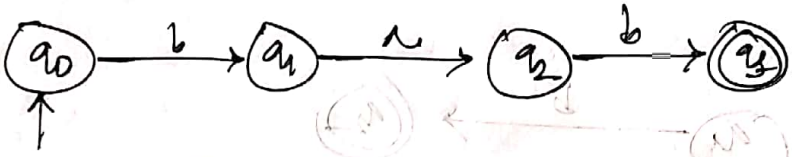
Step 5:

For string "aba"

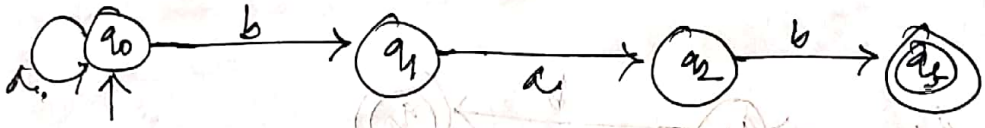


Case III: $L = \{bab, abab, bbab, babab, \ldots\}$

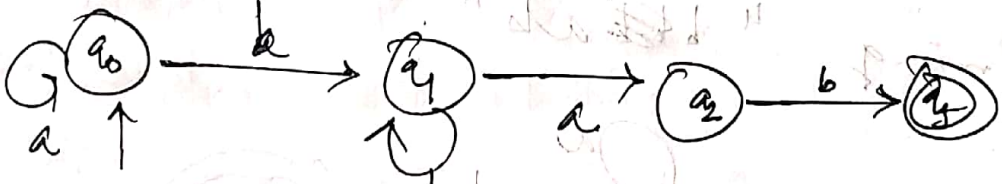
Step 1:



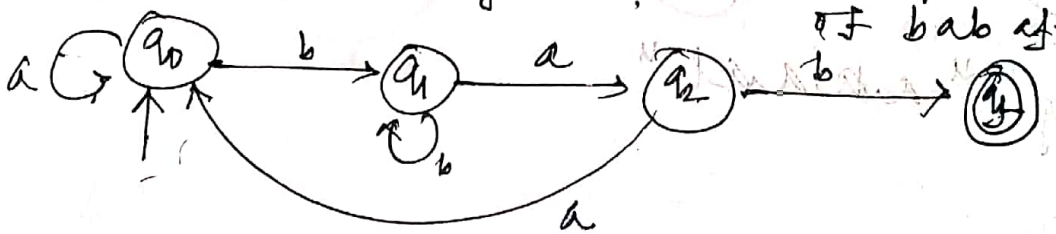
Step 2: For string "abab" or "aa...bab"



Step 3: For string "bbab" or "bbb...bab"

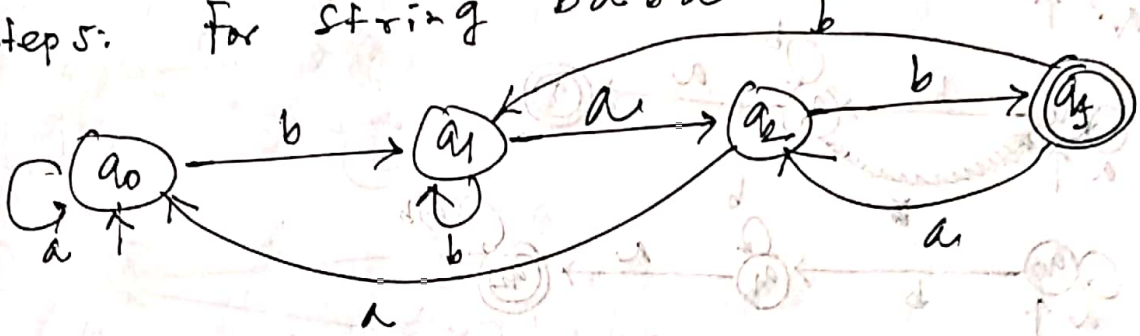


Step 3: For string "baba" but possibility of bab after



Step 4: For string baba

Step 5: For string "baba"



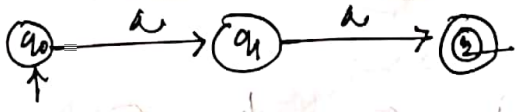
baba bab

Q: 3

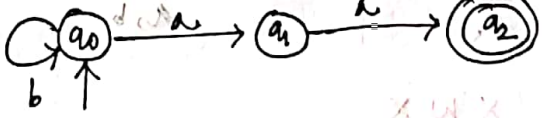
Design a DFA over $\Sigma = \{a, b\}$ s.t. every string accepted must contain a substring aa.

(i) aa
(ii) baa
(iii) abb

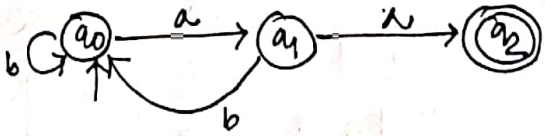
$L = \{aa, baab, \ldots\}$

① 

② For "bb....baa"



③ For "abaa"

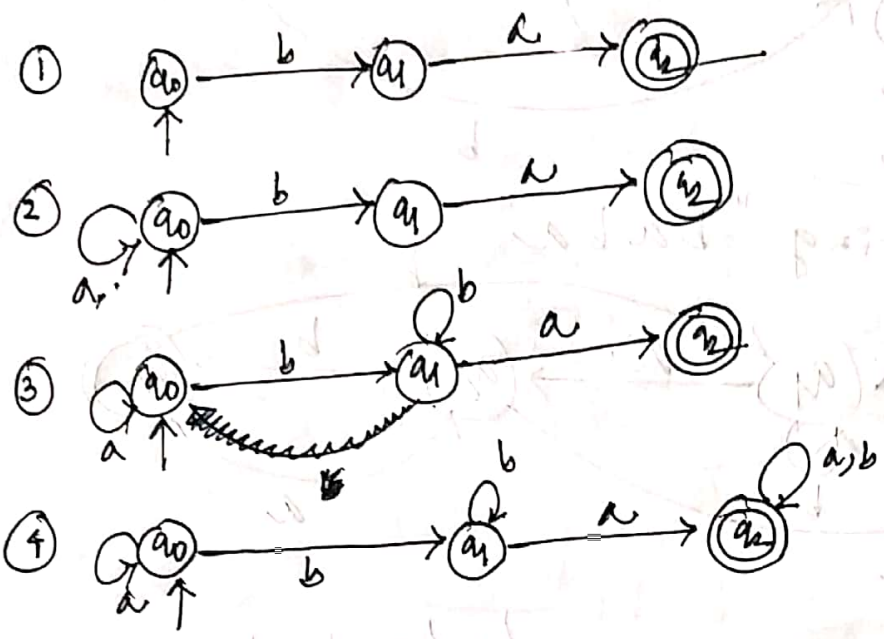


④ For "aabb" & "aaa"

Note:
All the strings are responded to both the i/p symbols $\{a, b\}$.

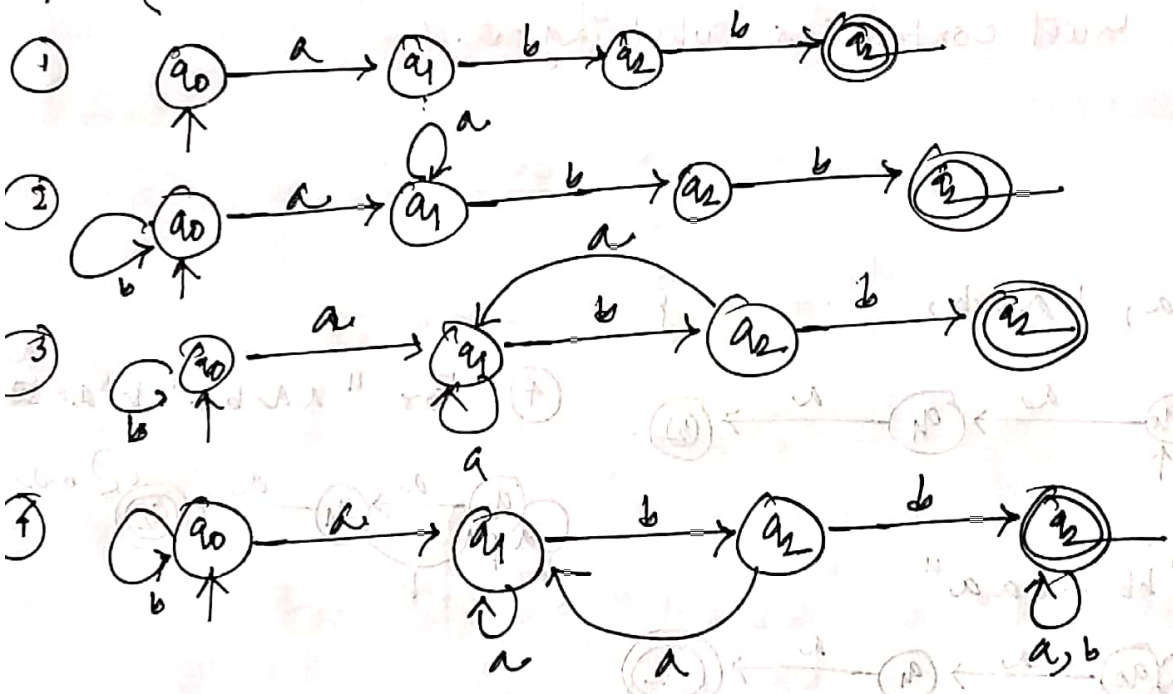
Case II :

$L = \{ ba, bab, aba, abab, \ldots \}$



Case III:

$L = \{ abb, babb, aabb, abbb, abba \ldots \}$



Note :

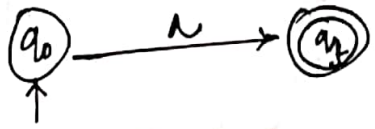
	wx	xw	xwx
$\|w\|=2$	2+2	2+1	2+1

Q: 4

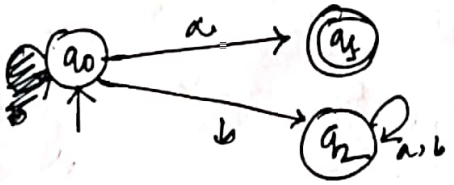
Design a DFA over $\Sigma = \{a, b\}$ s.t. every string accepted must start & ends with a.

Answer: $L = \{a, \lambda a, \lambda x a, \ldots\}$

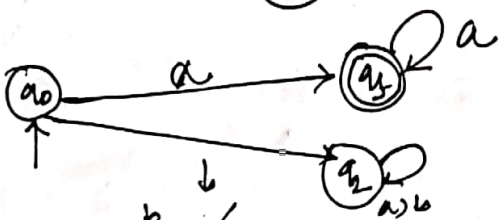
(1)



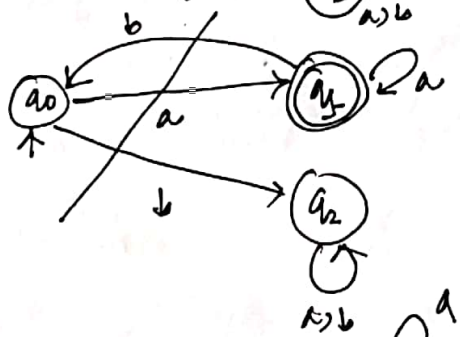
(2)



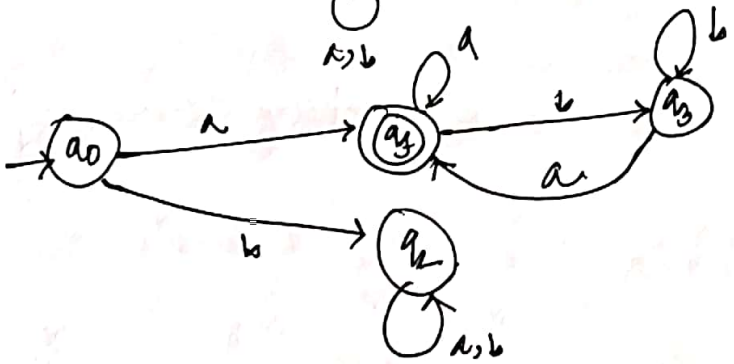
(3)



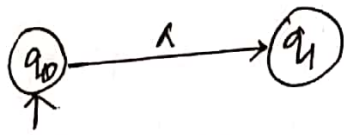
(4)



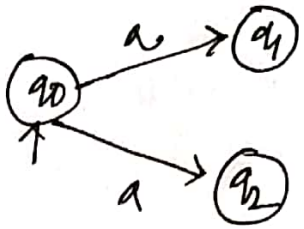
"$\lambda b \lambda$" or "λ bb"



Non-Deterministic FA : (NDFA/NFA)



$\delta(q_0, a) = q_1$



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

Deterministic ND

Non-Deterministic : Example LIFT, Ceiling FAN. Human being.

* Can we want a m/c to be non-deterministic?

No

Why? because if we are making a m/c then we are expecting the m/c to respond as we command.

for example if we are in a lift and we press button 4 then we ~~is~~ want that lift to go to 4th floor not in 3rd or any other floor.

* Every h/w & s/w is made of with deterministic m/c.

* Deterministic m/c is very difficult to implement but implementation is possible.

Why we are using NDFA/NFA?

because,

(1) It is easy to implement theoritically.

(2) It is simple to design.

(3) NDFA/NFA ⟶ DFA ⟶ MDFA

Def of NFA :

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

$Q \rightarrow$ Set of non empty finite Set

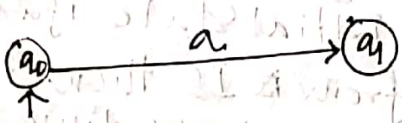
$\Sigma \rightarrow$ Non-empty set of finite state.

$q_0 \rightarrow$ only initial state

$F \rightarrow$ Set of finial state, $[0 \leq |F| \leq |Q|]$

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

PowerSet is a all possible subset of a set



$P\{q_0, q_1\} \rightarrow \phi, \{q_0\}, \{q_1\}, \{q_0, q_1\}$

$Q \rightarrow \{q_0, q_1\}$; $\Sigma = \{a\}$; $q_0 \rightarrow \{q_0\}$; $F \rightarrow \{\phi\}$

At state q_0 if I give a then NS may be q_0

" " q_0 " I " a " " " " q_1

" " q_0 " I " a " " " " q_0 & q_1 both

" " q_0 " I " a " " " " ϕ

Deterministic

① For one char Say 'a' ∃ only one edge from PS to NS.

② No concept of dead state

③ TF: $\delta : Q \times \Sigma \rightarrow Q$

④ ϕ Set never generates

Non-Deterministic

① For one char Say 'a' ∃ more than one edge or no edge from PS to NS.

② concept of dead state is not present.

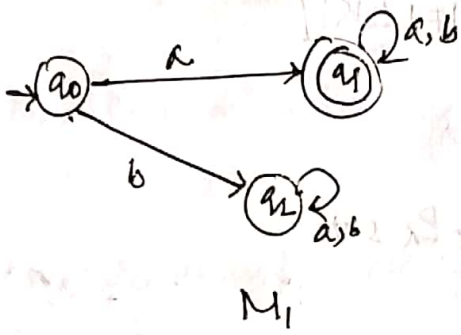
③ TF: $\delta : Q \times \Sigma \rightarrow 2^Q$

④ ϕ Set may generate.

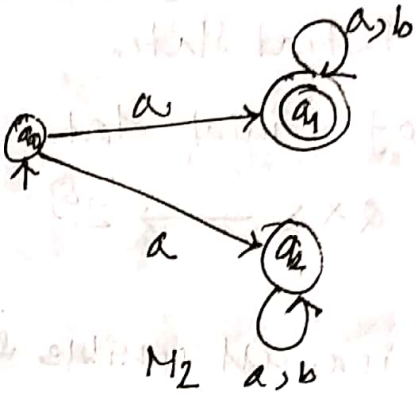
Acceptance by a NFA:

A string 'w' is said to be accepted by a NFA if ∃ at least one transition path on which we start at initial state q_0 ends at FS.

$$\delta^*(q_0, w) = F$$



M_1



M_2

After giving a string from initial state if at many times do not reach the FS from Q IS then it is okay. But, if we reach the FS at one time then we can say that the string is accepted by the NFA

Ex: 'ba' the string is not accepted by M_1

because,

At q_0 if I give b then we have to go to q_2

" q_2 " I " a " " " " " q_2

∴ Ultimate state is q_2 hence the string is not accepted.

~~But it~~ And as because M_1 is a DFA it is possible.

But at M_2 which is a NFA,

At q_0 if I give 'a' NS is q_0 i.e. FS

" " " " " q_2 i.e. Non-FS.

So, ∃ atleast one transition path on which we start at IS & ends at FS.

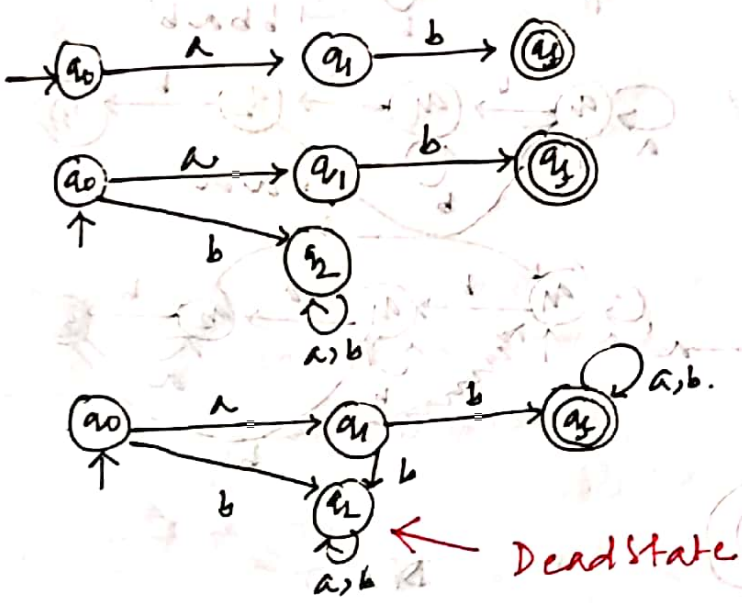
'ba' is not the member of the M_2 S_0, $\exists$ no chance of its acceptance.

'a,b' is the member of M_2 S_0, $\exists$ atleast one chance (transition) that it will be accepted by the NFA/M_2.

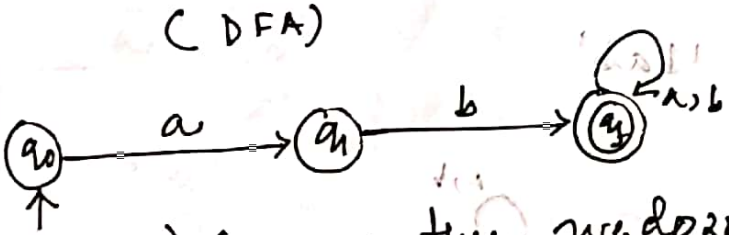
Problems:

* Design a NFA over an alphabet $\Sigma = \{a, b\}$ s.t. every string accepted must start with,

(i) ab. x



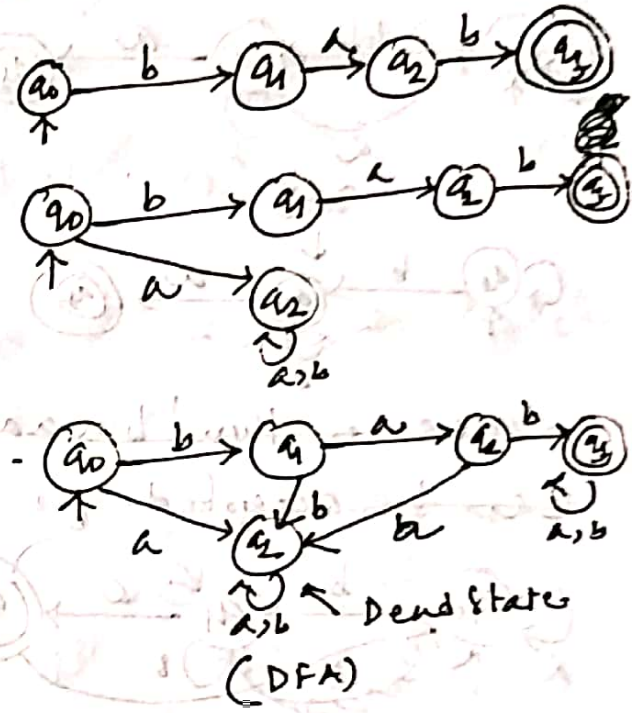
(DFA)



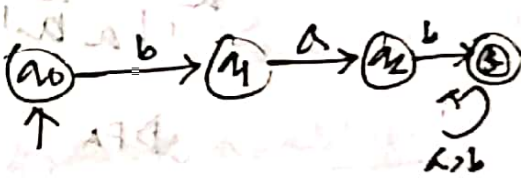
if at q_0 'b' comes then we do not need to respond.

(NFA)

(ii) bab x



(DFA)



(NFA)

Note:

	if $\|w\| = 2$.
$\{w\}^* \times 0$	DFA: 2 + 2
	NFA: 2 + 1 (without dead state)