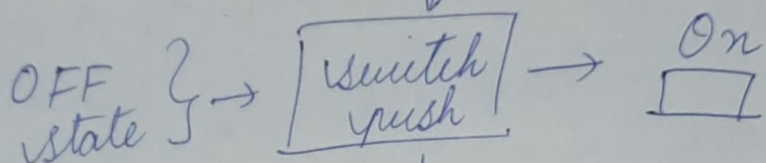
Automata - Abstract Machine.
It is has finite states - finite Automata.

Abstract Machine



Finite state Automata.

SASL

① Symbol: a, b, c . . . .
0, 1, 2 . . . . .

② Alphabet: Σ - collection of symbols
$\Sigma = \{a, b\}$ or $\Sigma = \{a, 0, 1\}$.

③ String - collection of symbols over Alphabet.
$\Sigma = \{a, b\}$
∴ string = a, ab, ba, b, bb, a . . . .

④ Language - set of all strings over alphabet
$L = \{a, b, ab, bb \ldots\}$.

⑤ Kleene Star $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots$
It of st is 0 - Null

Kleen closure

$\Sigma^+ = \Sigma^* - \{\epsilon\}$.

$\Sigma^* - \Sigma^0$.

eg: Language Accepts strings length 2 over $\{a, b\}$.

$\therefore L = \{aa, ab, ba, bb\}$ - finite

eg²: " " " ≥ 2 over " "

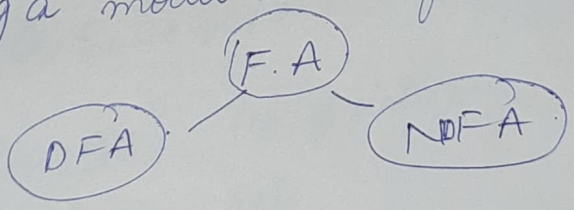
$\therefore L = \{aa, ba, bb, aaa, aba, \ldots\}$.

↓ infinite

=

Abstract Model.

Building a model with finite number of states



State: (q0)

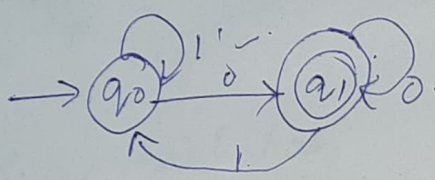
Initial state → (q0)

Transition → (q0) → (q1) edge

final state - (q1)

Finite Automata is represented with

1. Directed graph with states (Transition Diagram)
2. Transition Table



	0	1
q_0	q_1	q_0
q_1	q_1	q_0

where is q_0 when 0 input is given, its going to q_1

Finite Automata has 5 elements or tuples.

i.e $(Q, \Sigma, \delta, q_0, F)$.

$Q = \{q_0, q_1\}$ (set of states).

Σ = Input symbols

δ = Transition

q_0 = Initial state

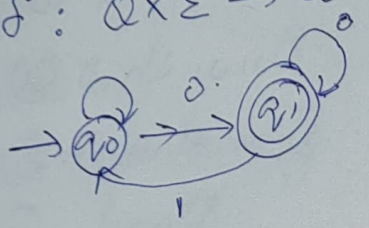
F = Final state

Only δ changes

DFA

every input ki edge nikli.

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



Every input has edge

$\therefore \{q_0, q_1\}$

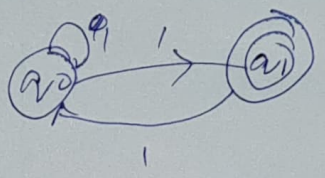
* Empty Transitions not possible
* Diffuult to construct
* Dead state
* Every state will have transition to single state

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

is the transitional funition

NDFA

$\delta : Q \times \Sigma \rightarrow 2^Q$ (power set of Q)



It may or may not have edge

$\{\{\varepsilon\}, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$

$= 2^2 = 4$

* Empty transition possible.
* Easy to construct
* No dead state
* Multiple states

$\delta : Q \times \Sigma \rightarrow P(Q)$

power set

is the transitional funition

How to construct DFA.

1. Starts with
2. Ends with.
3. Containing
4. length.
5. divisibility.

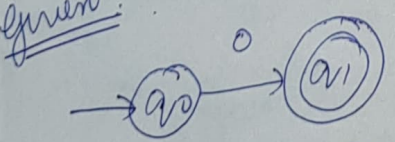
*Q) Construct a DFA which accepts all the strings over an alphabet $\Sigma = \{0,1\}$ starts with '0'.

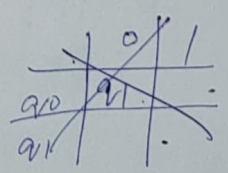
Sol:- $L = \{0, 01, 00, 010, 000, 011, 0 \ldots\}$ Infinite.

here, lt of string is 1.

∴ if lt is 1, possible states is $(n+1)$.

Given:

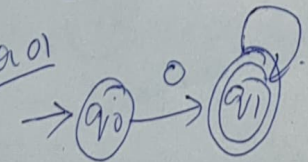




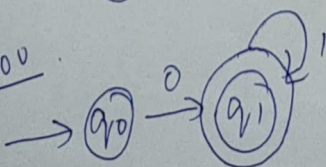
$q_0 - 0 - q_1$

$q_1 - 1 - q_1$

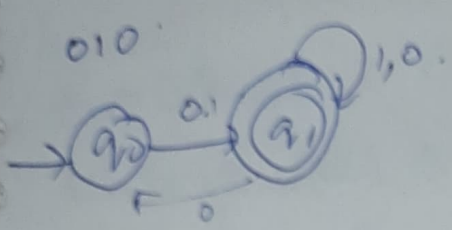
L: for 01



for 00



n+1 = (4).

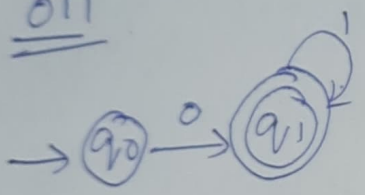


$q_0 - 0 - q_1$.
$q_1 - 1 - q_1$.
$q_0 - 1 - q_1$.
$q_1 - 0 - q_0$.

010

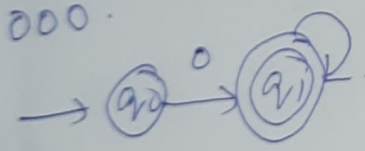
$q_0 - 0 - q_1 - 1 - q_1 - 0 - q_1$

011

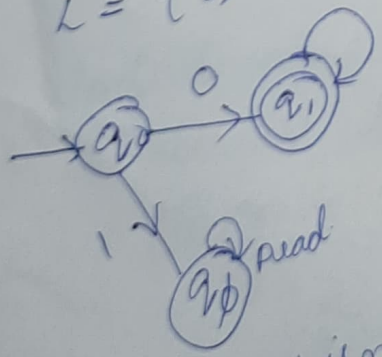


$q_0 - 0 - q_1$
1
1
✓

000.



$L = \{0, 01, 00, 010, 011, 000, 001 \ldots\}$.



here 1 is not there.

	0	1
→ q_0	q_1	q_ϕ
* q_1	q_1	q_1
q_ϕ	q_ϕ	q_ϕ

Construct a DFA which Accepts all strings over an alphabet $\Sigma = \{0,1\}$ starts with '01'

~~$L = \{$~~

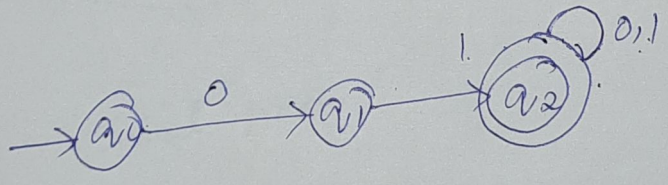
Sol: lt of string = 2 min.

$L = \{01, 010, 011, 0100, 0101, 0111 \ldots\}$ Infinite.

$\therefore (n+1) = 2+3 = 3$

$\therefore$ we get 3 states

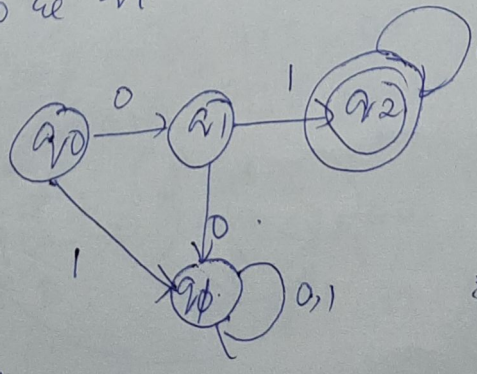
q_0 q_1 q_2



but its not complete DFA

because they dont have 0 & 1 for each state.

$\therefore$ q_0 & q_1 take a dummy state.



		0	1
→	q_0	q_1	q_ϕ
	q_1	q_ϕ	q_2
*	q_2	q_2	q_2
	q_ϕ	q_ϕ	q_ϕ

$\therefore$ complete DFA.

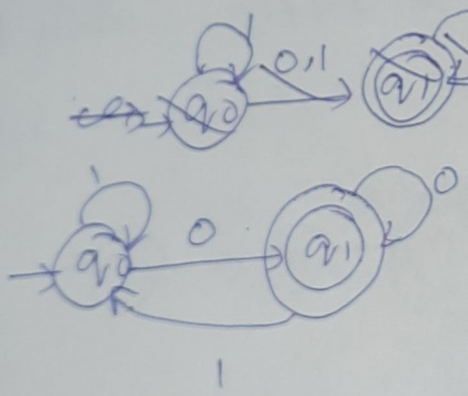
2). Construct DFA which accpt all strings over an Alphabet $\Sigma = \{0,1\}$ ends with '0'.

$L = \{0$

① Min of string = 1.

② $n+1 = 1+1 = 2$ ∴ 2 states q_0, q_1

③ $L = \{0, 10, 00, 010, 100, 110, 000, 1110, 1010, 0000, 0100, 01100 \ldots\}$ ↓ Infinite



	0	1
q_0	q_1	q_0
q_1	q_1	q_0

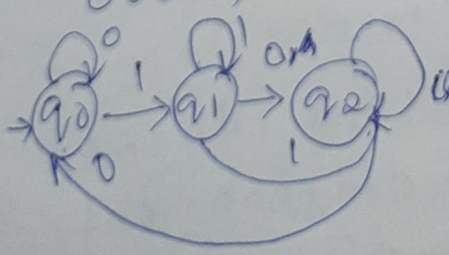
∴ Final DFA

2) ends with 10.

① Min of st = 2

② $n+1 = 2+1 = 3$, q_0, q_1, q_2

③ $L = \{10, 010, 110, 0010, 0110, 1010, 1110, 00010, 11110, 10110, 01010, 11010 \ldots\}$

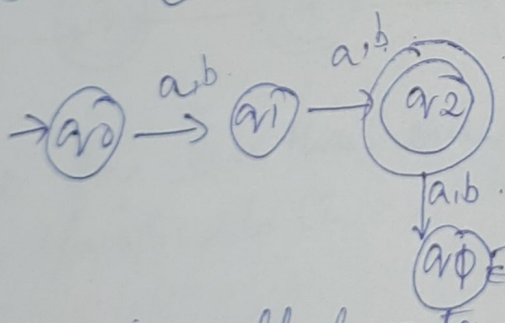


	0	1
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_0	q_1

starts with 0 aante pakka dead state
ends with 0 is not dmy state.

Construct a DFA which accepts all strings over an alphabet $\Sigma = \{a, b\}$ where <u>lt of string is 2</u>.

① lt of st = 2.
② states = n+1 = 2+1 = 3, q_0, q_1, q_2.
③ L = {aa, ab, ba, bb}.



	a	b
q_0	q_1	q_1
q_1	q_2	q_2
q_2	q_ϕ	q_ϕ
q_ϕ	q_ϕ	q_ϕ

dont give self loop to q_2 because it will increase lt of string thats why use dead state.

---

lt of st ≥ 2

① lt of st = 2
② states = n+1 = 2+1 = 3.
③ L = {aa, ab, ba, bb, aba, bba, bab, aaa, bbb, ...}

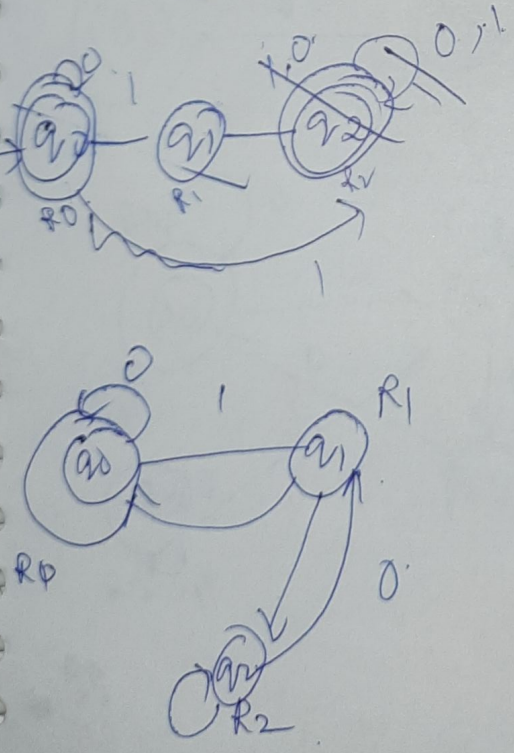
, 9, 12, 15.

④ L=

for remainder take ③ and divisible it

∴ Mg:- 0, 3, 6, 9, 12, 15
0, 0, 110, 1001,
1100, 1111 ..... }
Infinite

⑤ from remainder take states
∴ 3 states



R = div by 3.

3)2(0
0
1

3)4(1
3
1

2)2(1
2
0

2)1
0

D	B	R
0	0	0
1	01	1
2	010	2
3	11	0
4	100	1
5	101	2
6	110	

2|4
2|2-0
1-0

2|2
1-0

2|3
1-1

3

2|5
2|2-1
1-0

101

2|6
2|3-0
1-1

(2)6(3
6

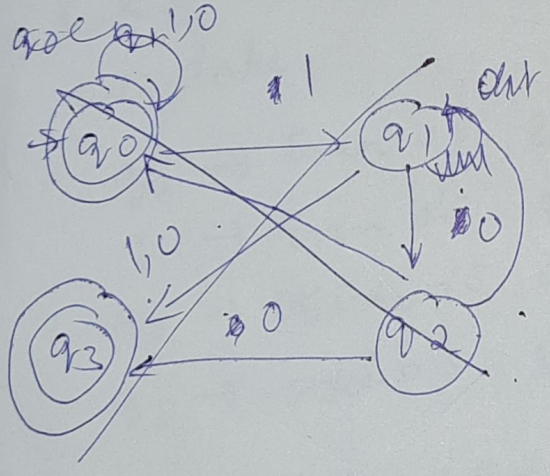
first write all decimals from 1-10 & remainders.

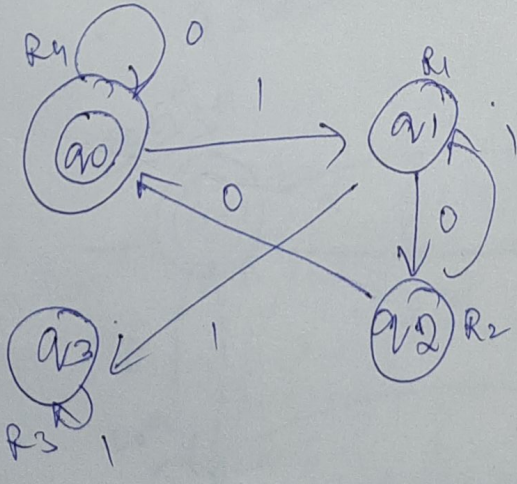
② Construct DFA which accepts all strings over an alphabet $\Sigma = \{0,1\}$ where Binary strings are divisible by 4.

① Multiples of 4 = 4, 8, 12, 16, 20 ...

100, 000, 1000, 1100, 1000

N	I/P	R
0	0	0
1	01	1
2	10	2
3	11	3
4	100	0
5	101	1
6	110	2
7	111	3
8	1000	0





3 4 11
3
1

9 8
2 4 - 0
2 2 - 0
1 - 0

Evens & odds

① Construct a DFA to accept the lang over $= \{0,1\}$

$L = \{w/w$ has even no of 0's & even no of 1's$\}$.

2 zeros, 2 ones.

$\therefore L = \{0110, 110110, 1001 \ldots\ldots\}$.

~~& &~~

we take

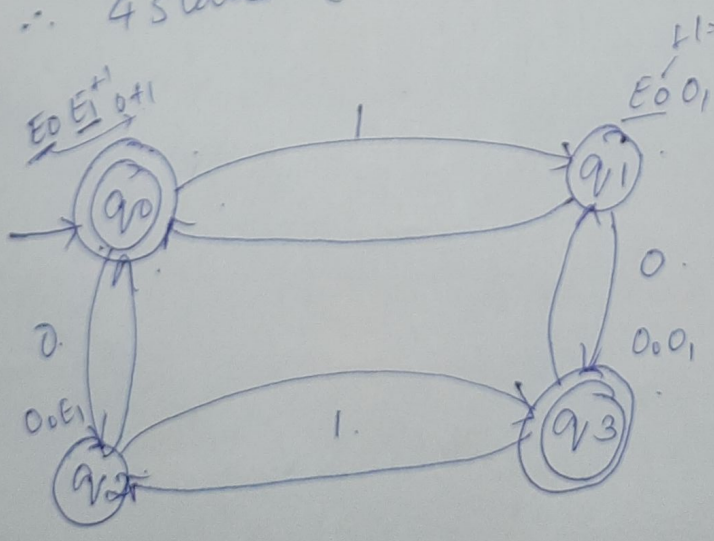
$q_0 \rightarrow$ Even 0's & Even 1's.

$q_1 \rightarrow$ Even 0's & Odd 1's.

$q_2 \rightarrow$ ~~Even~~ 0's & Even 1's.

$q_3 \rightarrow$ odd 0's & odd 1's.

$\therefore$ 4 states (Both start & have final state).



	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

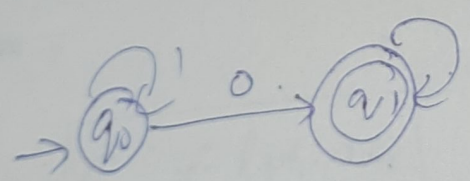
Substring

① Construct a DFA which accepts all strings over an alphabet $\Sigma = \{0,1\}$ where string contains '0' as substring.

Sol. $L = \{0, 01, 10, 00, 000, 100, 001, 101 \ldots\}$

No of st = 2

$n+1 = 2 \Rightarrow$ 2 states



1111 ✗

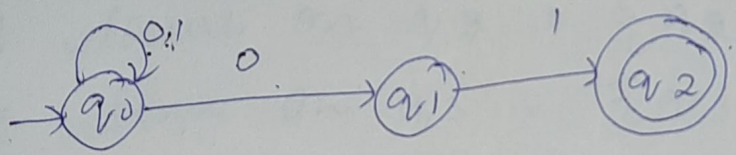
NFA

① Construct a NFA which accepts all strings over an alphabet $\Sigma = \{0,1\}$ where strings end with '01' (starting to 01)

$L = \{01, 001, 1001, 0101, 0001, 1101, \ldots\}$

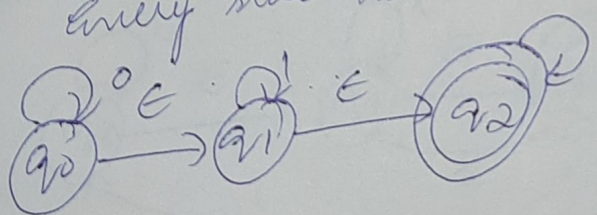
let of $s = 2$

$n+1 = 2+1 = 3$ states



ϵ Closure

Every state has its own ϵ.



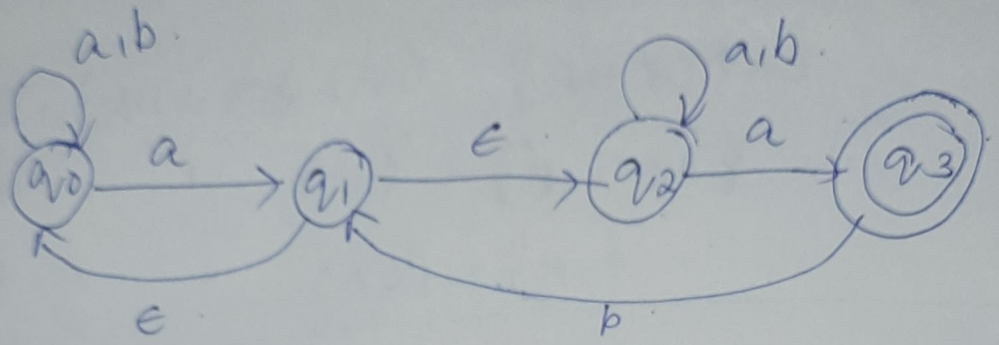
ϵ on $(q_0) = \{q_0, q_1, q_2\}$

ϵ on $(q_1) = \{q_1, q_2\}$

ϵ on $(q_2) = \{q_2\}$

We have to write for each ϵ every state

①



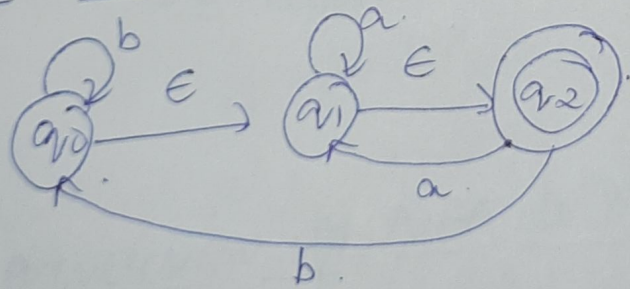
ϵ closure on q_0 = $\{q_0\}$

ϵ closure on q_1 = $\{q_1, q_2, q_0\}$.

ϵ closure on q_2 = $\{q_2\}$

ϵ closure on q_3 = $\{q_3\}$ //.

NFA with ϵ to NFA without ϵ



① ϵ - closures :-

② $\delta'(q, a)$ = ϵ - closure (δ(ϵ-closure $(q), a)$

③ Final state = $F' = F \cup \{q_0\}$ If ϵ - closure of (q_0) is having F as a memeber $= F$, Otherwise.

④ F, Q, δ } with ϵ

F', Q', δ' } without ϵ.

①

ϵ closure on $(q_0) = \{q_0, q_1, q_2\}$

" " " $(q_1) = \{q_1, q_2\}$.

ϵ " " $(q_2) = \{q_2\}$.

② $\boxed{\delta'(q,a) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q,a))}$.

$\delta'(q_0,a) = \epsilon\text{-closure}(\delta \underline{\epsilon\text{ closure}}(q_0,a))$.

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

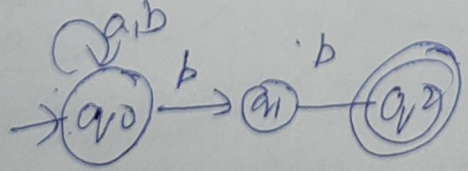
$= \epsilon$ closure $(\delta\{q_0,a\} \cup \delta\{q_1,a\} \cup \delta\{q_2,a\})$.

$= \epsilon\text{-closure}(\phi \cup q_1 \cup q_1) =$ ϵ closure(q_1)

$= \{q_1, q_2\}$.

Conversion of NFA to DFA:

$\delta: Q \times \Sigma \to 2^Q$ $\quad \delta: Q \times \Sigma \to Q$.



1 step : construct NFA transition table

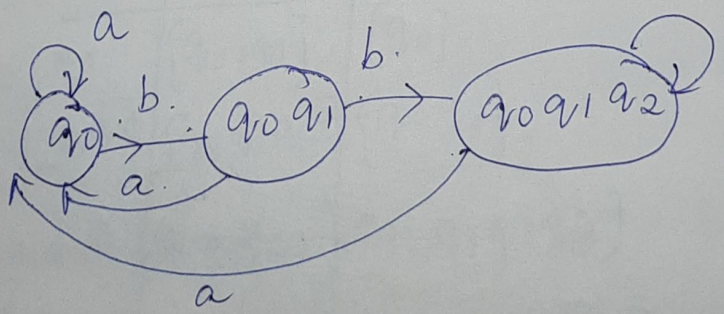
NFA table.

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

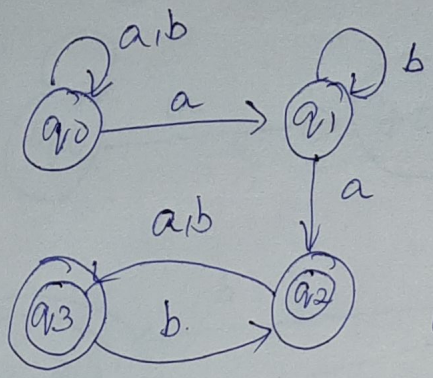
Step 2 DFA.

δ	a	b
q_0	q_0	$[q_0 q_1]$ – single state
$[q_0 q_1]$	$[q_0]$	$[q_0 q_1 q_2]$
$\{q_0 q_1 q_2\}$	$[q_0]$	$[q_0 q_1 q_2]$

wherever q_2 is there, take it as final state.



NFA.

② 

wherever q2 & q3 r final states make same an DFA too -

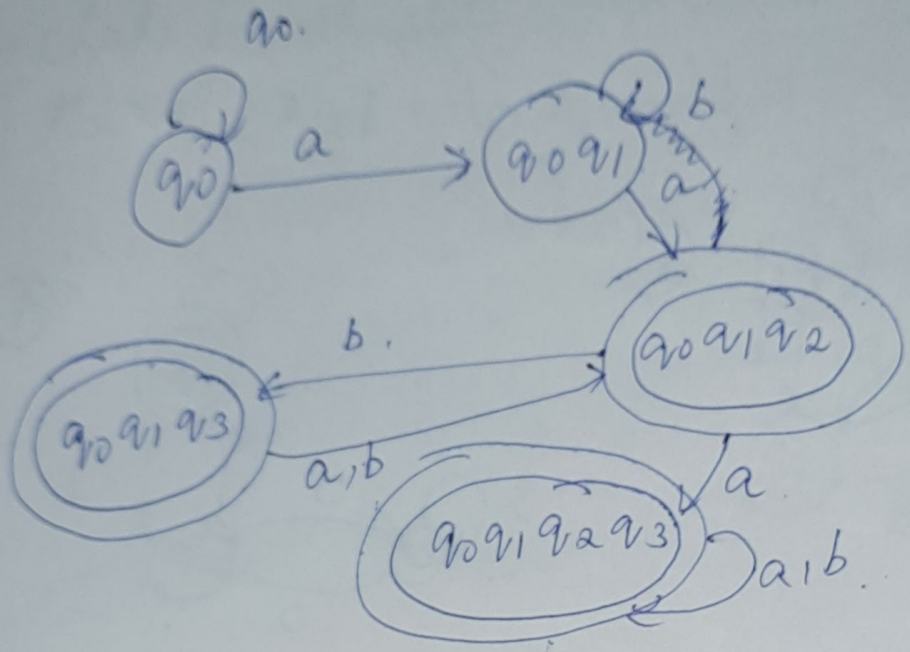
Step 1 NFA δ table

δ	a	b
q0	{q0, q1}	{q0}
q1	{q2}	{q1}
q2	{q3}	{q3}
q3	—	{q2}

Step 2 DFA table.

q0 on a =
q0 on a =

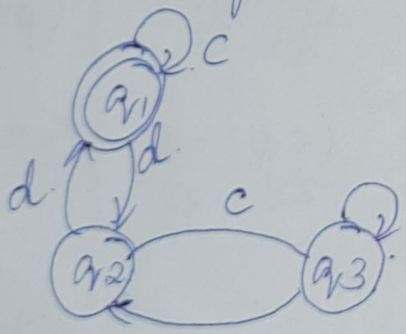
δ	a	b
q0	[q0 q1]	[q0]
[q0 q1]	[q0 q1 q2]	[q0 q1]
[q0 q1 q2]	[q0 q1 q2 q3]	[q0 q1 q3]
[q0 q1 q2 q3]	[q0 q1 q2 q3]	[q0 q1 q3 q2]
[q0 q1 q3]	[q0 q1 q2]	[q0 q1 q2]



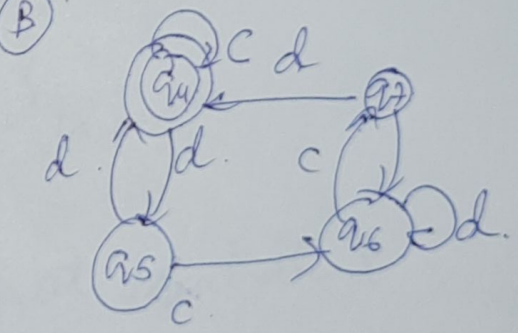
Equivalence of 2 finite Automata

Ex:

(A)



(B)

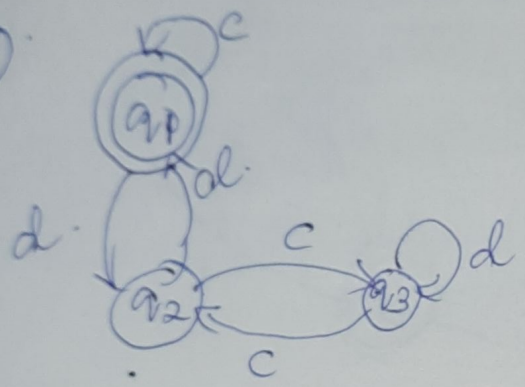


If both pair of state both r final a FS, we can say they are equivalent.

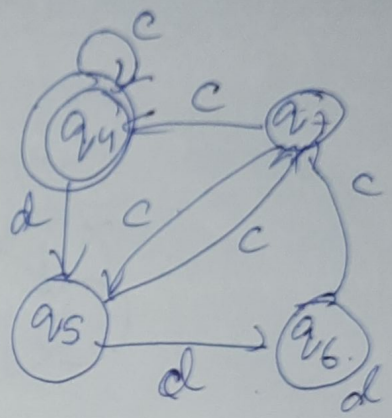
States	c	d
(q1 q4)	(q1, q4) FS FS	(q2, q5) IS IS
(q2 q5)	(q3, q6) IS IS	(q1, q4) FS FS

Check for all states
then says A & B are equivalent

(A)



(B)

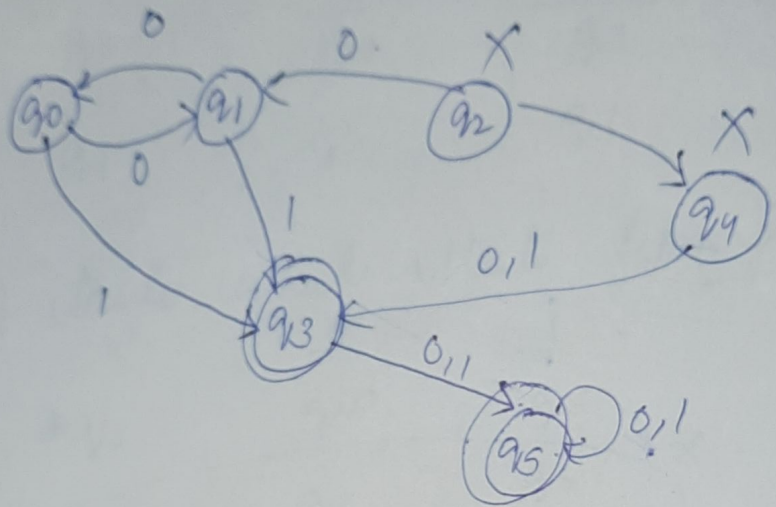


δ	c	d
(q_1, q_4)	(q_0, q_4) FS FS ✓	(q_2, q_5) IS IS ✓
(q_2, q_5)	(q_3, q_7) IS IS ✓	(q_0, q_6) FS IS ✗

∴ A & B are not equivalent

Minimization of DFA

Eg:



step 1 : Remove q_2 & q_4 in F.A ~~—~~ unreachable state.

state 2 : Draw ~~d~~ table.

States	0	1	
q_0	q_0	q_3	T_1
q_1	q_0	q_3	
q_3	q_5	q_5	T_2
q_5	q_5	q_5	

Step 3 : Div the table into

(i) Table which starts from non final states.

	0	1	set
q_0	q_1	q_3	
q_1	q_0	q_3	

(ii) table which starts from final states.

	0	1	
*q_3	q_5	q_5	
*q_5	q_5	q_5	x

Step 4 : set 1 has no similar rows, they will be same.

set 2 has similar rows, so skip.

q_5 ee replace q_5 by q_3

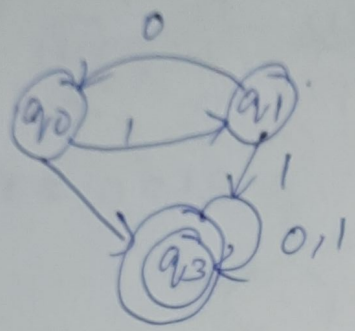
Step 5:

	0	1
q_3	q_3	q_3

Step 6 Combine set 1 ee set 2.

State	0	1
q_0	q_0	q_3
q_1	q_0	q_3
* q_3	q_3	q_0

Step 2



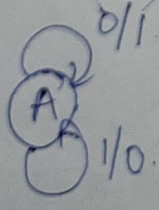
∴ we have minimized from 6 states to 3

Mealy Machines

* o/p depends on present state + i/p.
* 6 tuples $(Q, \Sigma, O, \delta, X, q_0)$
  - O → o/p
  - δ → o/p

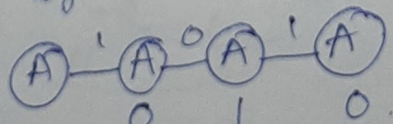
(1) Construct a mealy machine M/C that produce 1's complement of any binary input string

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



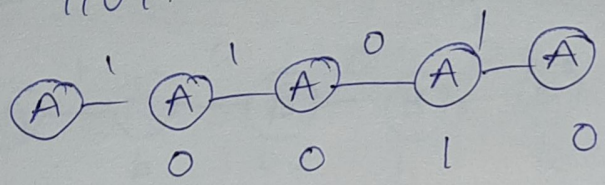
eg
01001 - i/p
1's comp.
10110 - o/p

eg2 101



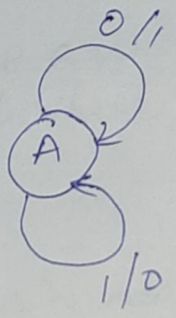
o/p = 010

Eg 1101 → i/p



o/p = 0010.

∴



---

(2) Construct a mealy m/c that prints 'a' whenever the sequence '01' is encountered in any i/p binary string.

01 01001 01
a a a

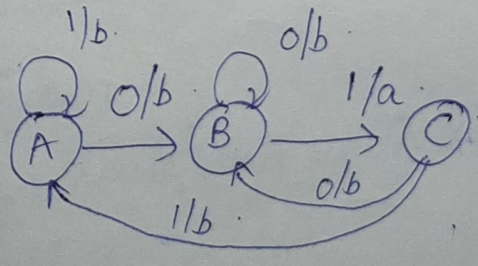
Sol i/p $\Sigma = \{0, 1\}$.

o/p $\dot{\Sigma} = \{a, b\}$

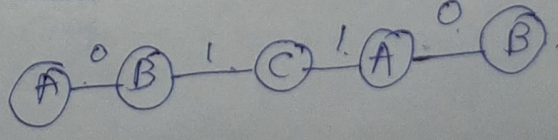
lt = 2.
states = 3.

01 - a.

reach any a print a.

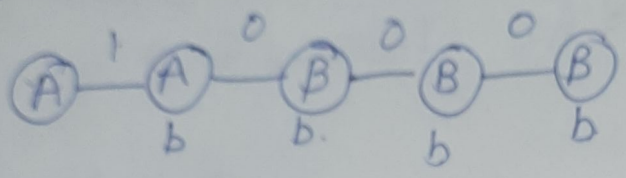


to check.



eg 01 0 as
babb.

Eg2 :- 1000



∴ 1000 = bbbb

∴ no a is encountered as 01 is not there

---

(3) Construct a mealy machine that prints 'a' whenever the sequence '01' is encountered in any i/p binary string.

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

(3) Construct a mm accepting lang consisting of strings from Σ^*, where $\Sigma = (a,b)$ & the strings should end with either aa or bb.

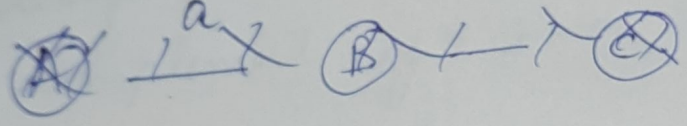
Sol I/p $\Sigma = \{a,b\}$

string should end with either aa or bb.

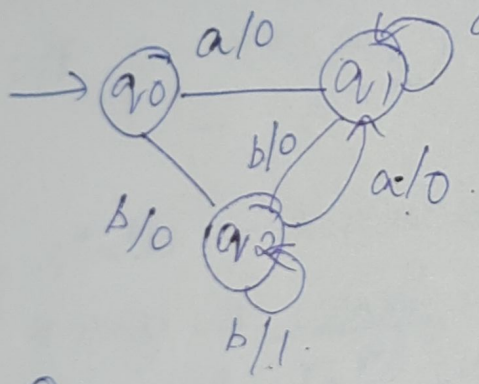
∴ O/p = O = {0,1}

* Assume that the string ends with aa or bb generates o/p = 1, else o/p = 0.

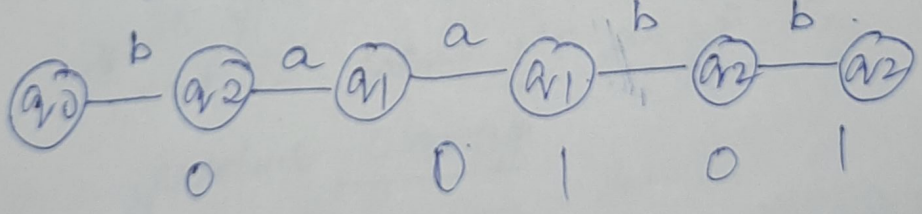
* It of $S = 2$.
* states = 3



aa
bb

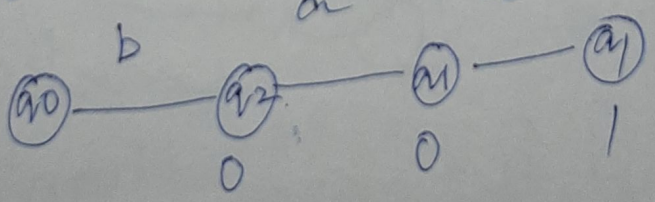


check : baabb



∴ 00101

baa = 001



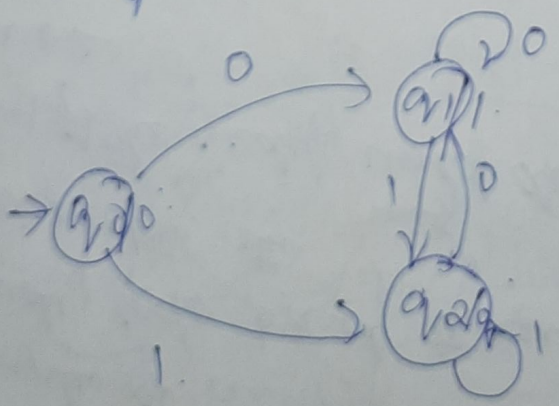
Moore Machine

* present state
* 6 tuples

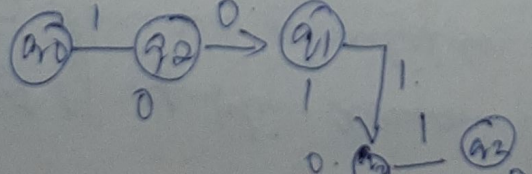
Q① Design a Moore Machine to generate 1's complement of a given no.

if i/p 0 - o/p 1.
1 - 0.

∴ 3 states r required.
* one is start state
* second state is taking 0's as i/p & produce 1 as o/p.
* 3rd is taking 1's as i/p & produce 0 as o/p



Eg. 1011 ⇒ 0100



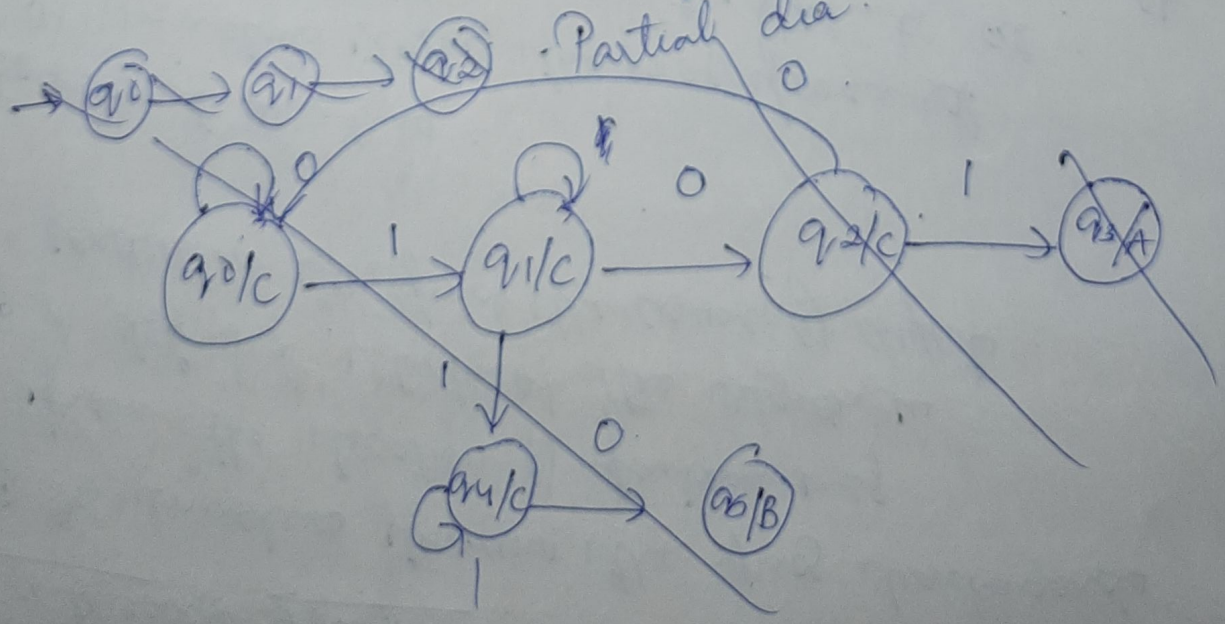
① Design a moore m/c for a binary i/p seq such that if it has substring 101, the m/c o/p A, if the i/p has substring 110, its o/p is B otherwise it o/p's C

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

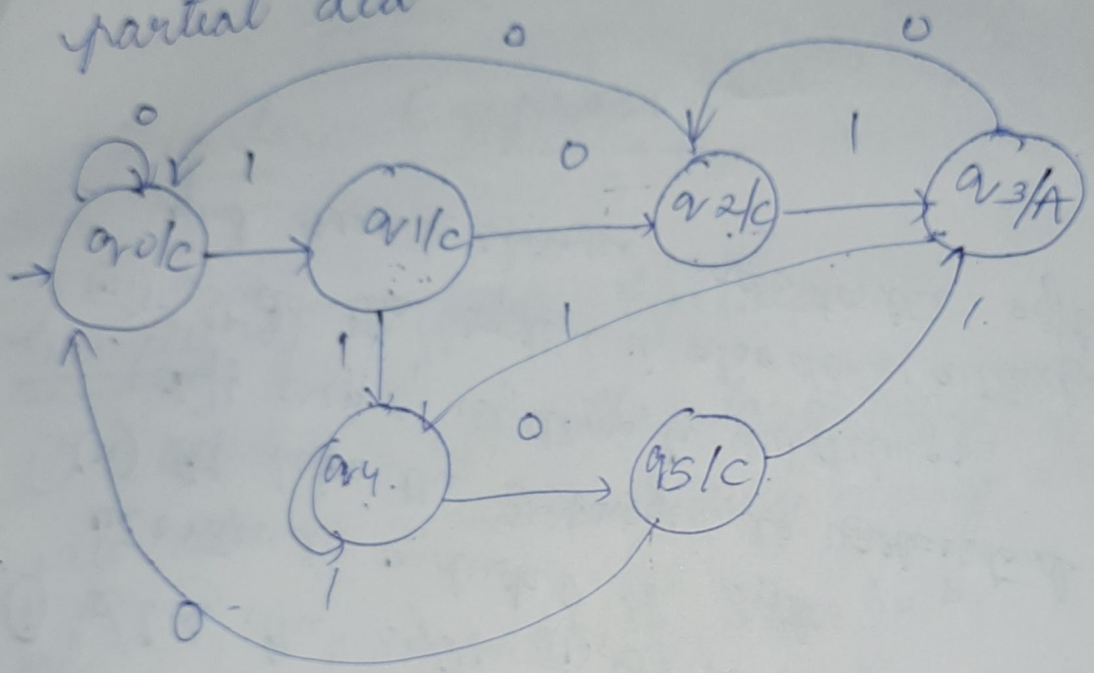
i/p	o/p
101	A
110	B
---	C

for designing we check 3 cond...

* if we get a 101 - o/p is A.
* if we get 110 - o/p is B.
* for other strings - o/p is C.



partial dia



More to come in Moore Machines.

Unit 2

Regular Expressions

Examples of Regular Expressions

① Write R.E for the lang accepting all strings containing any no of a's & b's

Sol: $L = \epsilon, a, b, aa, bb, aba, bab \ldots$

$$\boxed{R.E = (a+b)^*}$$

* This will give the set as $L = \{\epsilon, a, aa, b, bb, ab, ba, ba \ldots$ any combination of a & b.
* The $(a+b)^*$ shows any combination with a & b even a null string.

② Write R.E for the lang accepting all strings which are starting with 1 & ending with 0, over $\Sigma = \{0, 1\}$

Sol * The first symbol is 1.
* The last " " 0.

$$\therefore \boxed{RE = 1(0+1)^*0}$$

, inbetween.

$L = 1101$
$10, 1000$

(3) Write R.E for the lang starting with a but not having consecutive b.

Sol $L = \{a, aba, aab, abaa, aaa, abab \ldots\}$

$R.E = \{a + ab\}^*$

(4) Write the RE for the lang over $\Sigma = \{0\}$ having even # of string.

Sol $L = \{\epsilon, 00, 0000, 000000 \ldots\}$ * contain ϵ

$R.E = (00)^*$

(5) Write R.E for the lang L over $\Sigma = \{0, 1\}$ such that all the strings do not contain the substring 01.

Sol $L = \{\epsilon, 0, 1, 00, 10, 100, \ldots\}$

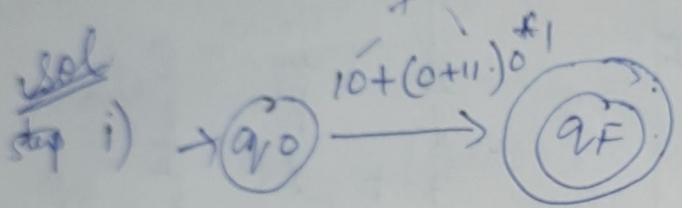
$R.E = (1^*0^*)$

Conversion of R.E to F.A

(1) Design a FA from RE $10+(0+11)0^*1$.

Sol

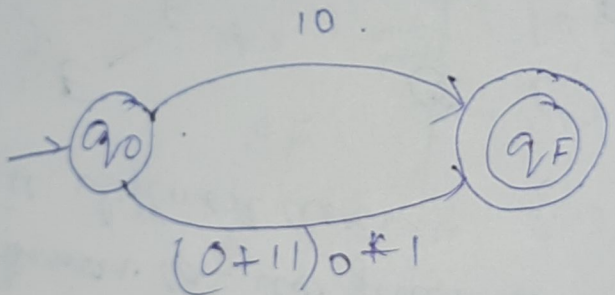
Step 1)



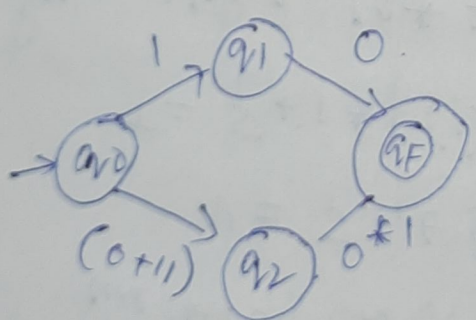
RE
↓
NFA-ϵ
↓
NFA
↓ DR
DFA

0* = self loop
0+ = 00

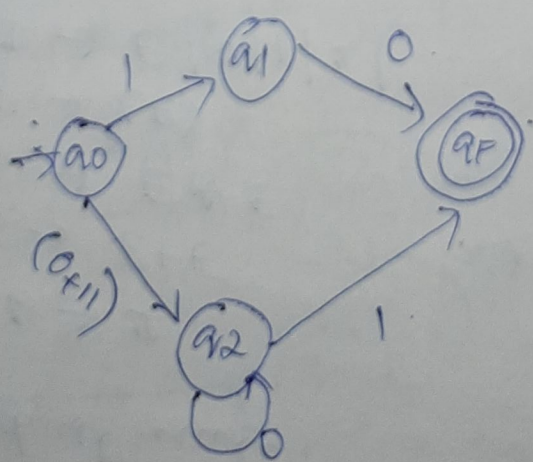
Step 2)



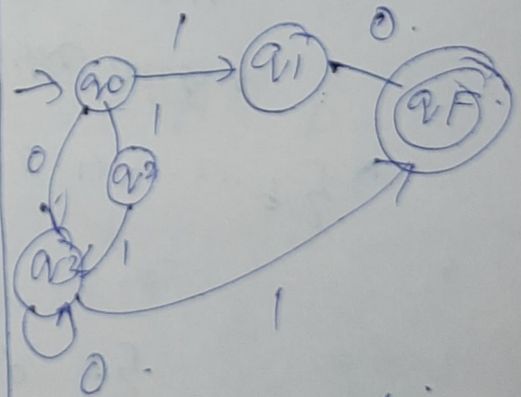
Step 3



Step 4



Step 6



This is NFA of dia

Now we got NFA without ε Now

convert NFA ≃ DFA (separate states bec in DFA)

∴ First write TT for NFA

state	0	1
q_0	q_3	$\{q_1, q_2\}$
q_1	q_F	ϕ
q_2	ϕ	q_3
q_3	q_3	q_F
q_4	ϕ	ϕ
q_F	ϕ	ϕ

=

The Equivalent DFA

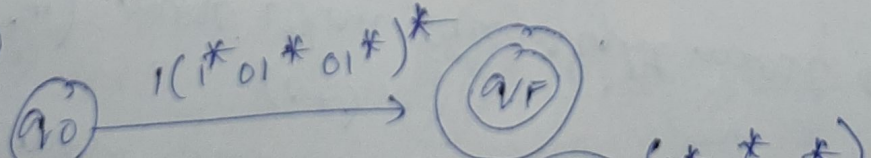
state	0	1
$[q_0]$	$[q_3]$	$[q_1 q_2]$
$[q_1]$	$[q_F]$	$\{\phi$
$[q_2]$	$\{\phi$	$[q_3]$
$[q_3]$	$[q_3]$	$[q_F]$
$[q_1, q_2]$	$[q_F]$	$[q_F]$
$[q_F]$	ϕ	ϕ

② Design NFA from a given R.E

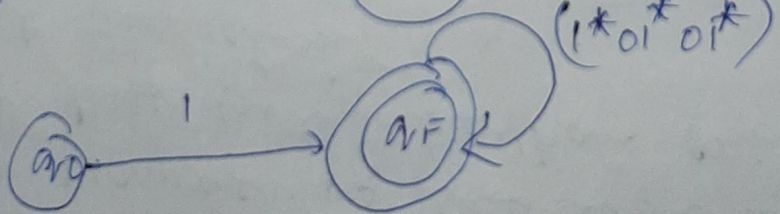
$1(1^*01^*01^*)^*$

( )* means loop.

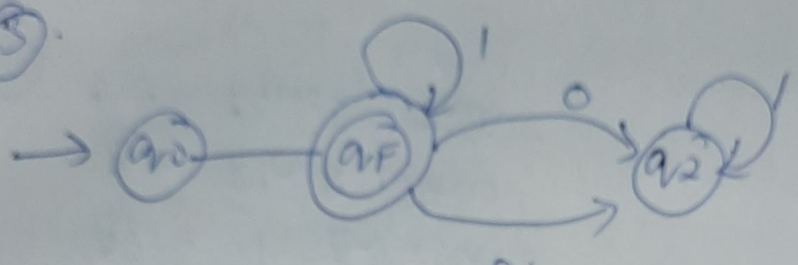
①



②



(5)



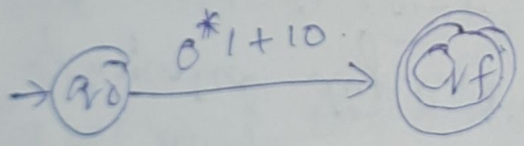
= NFA

E →

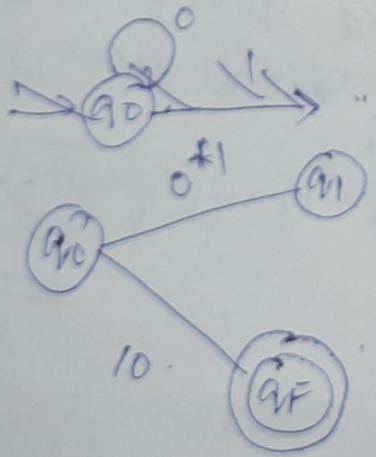
(3) Construct a finite automata for R.E

0^*1+10

(1)

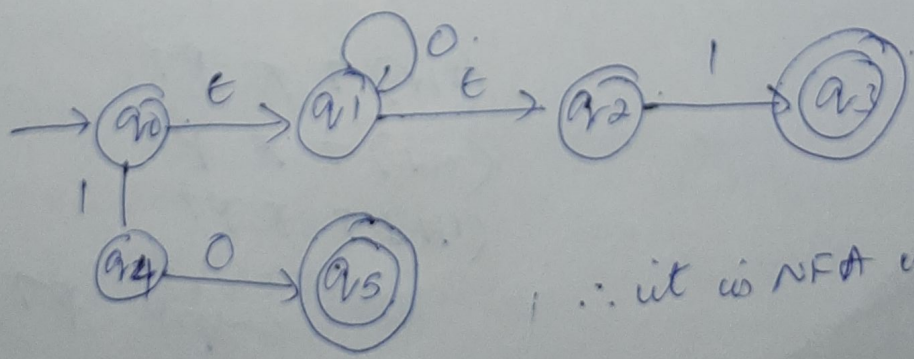


(2)



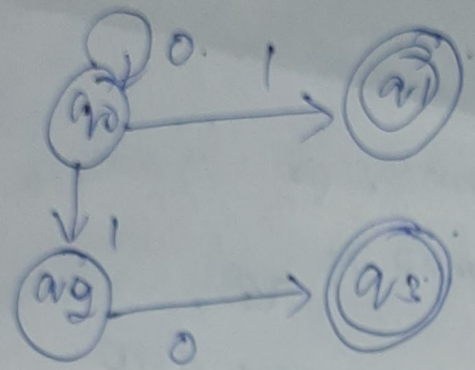
ε was without doing anything more to state

(3)



∴ it is NFA with ε.

(4)



∴ without ε.

Closure Properties of Regular Lang-

Or = union
and = intersection

De morgan's law

$$L \cap M = (L^c \cup M^c)^c$$

for properties refer pdf

Pumping Lemma for Regular Lang

* To show lan r not regular

Eg.

Prove that $L = \{a^i b^i / i \geq 0\}$ is not regular.

Sol

① At first, we assume that L is regular & n is the no of states.

② Let $\omega = a^n b^n$, Thus $|\omega| = 2n \geq n$.

③ By pumping lemma, let $\omega = xyz$, where $|xy| \leq n$.

④ Let, $x = a^p$

$y = a^q$

$z = a^r b^n$

where $p+q+r = n$,

$p \neq 0, q \neq 0, r \neq 0$

∴ Thus ...

(5) let $k=2$

Then $xy^2z = a^1 a^{2q} a^r b^n$.

(6) No of a's $= (p+2q+r)$

$= (p+2q+r) + q = n+q$.

(7) Hence, $xy^2z = a^{n+q} \cdot b^n$. Since $q \neq 0$.

xy^2z is not in the form of $a^n b^n$.

(8). Thus xy^2z is not L.

Hence L is not regular.

FLAT OBJ.

1. Complementation (CFL ✗)
2. CFL (Push ↓A)
3. NPDA (wcw')
4. Even & palindromes ($S \to 0S0 | 1S1 | 0 | 1$)
5. Ambiguous (left most)
6. Recursive (TM stops)
7. PCP ✗ RE
8. RE ✓ (TM)
9. Recursive (diag)
10. Unit Production ($A \to B$)

11. Regular ($L_1 \cup L_2$)
12. Ambiguous
13. P = set of Production Rules.
14. $A \to c$; A is non-terminal.
    $0^n 1^n 2^n$ is context free lang.
15. PCP can be solved by TM.
16. Direction : q_0 , q_1
17. Halt ties -

htm ✓ = universal turning machine

tM = Recursive Enumerable

④ Prove the Existance of Languages which are not recursively enumerable

Sol

* There are non recursively enumerable languages.
* proof by a diagonalization argument.
* All possible turing machines & accept languages over Σ

Proof

$M_1, M_2, M_3, M_4, M_5 \ldots\ldots$ (M_i) $\ldots\ldots$

$L(M_1), L(M_2), L(M_3) \ldots$ All the recursively enumerable lang which are over Σ

* We will define a lang L_d and ensure that $L_d \neq L(M_i)$ for every i.
* $\therefore$ L_d is not recursively enumerable over Σ we would have ensured that $L_d \neq L(M_i)$.

Δ = every TM can be encoded & rep by a finite string

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

A set of quintuples can be represented by finite string.

Defination of L_d

$L_d = \{x \in \{0,1\}^* \mid$ The TM whose code is x does not accept x.

Claim

* L_d is not RE.
* Suppose it is, then by defn, there is a T.M to accept L_d.
* Let M be the ith TM is not our enumeration.
* Let x be the ith binary string in our enumeration.

$\because$ There $x \in L_d \Rightarrow M$ does not accept $x \Rightarrow L(M) \neq L_d$.

Or $x \in L_d$

$\Rightarrow M$ would accept the $x \Rightarrow L(M) \notin L_d$

$\therefore$ L_d is not R.E

L_d defn

$L_d \neq L(M_1)$
$L_d \neq L(M_2)$
$L_d \neq L(M_3)$
⋮

---

(5)(1) Components of Turing Machine

* Turing Machine dev - 1936.
* TM accepts Recursive Enumerable lang.
* Generated by a grammer
* There are 7 comp

i.e $(Q, T, \Sigma, \delta, q_0, B, F)$

Q = finite set of states
T = Tape alphabet
Σ = I/P alphabet
δ = Transition func
$\delta: Q \times X \to \{$ leftshift, Rightshift $\}$.
q_0 = Initial states
B = Blank states
F = Final states

(5) Ambiguous Grammer

* A grammer is said to be ambiguous
(i) if there exists more than one left most derivation

(OR)

ii) more than one right most derivation.

(OR)

(iii) more than one Parse tree for given I/P string.

* G (V, T, P, S) is said to be ambiguous if & only there exists a string in T^* that has more than 1 parse tree.

where,

V = finite set of variables.
T = finite set of terminals.
P - finite set of production $A \rightarrow \alpha$, where A is variable & $\alpha \in (V \cup T)^*$
S - It is designated variable called the start symbol.

Ex: Let us consider a grammar with product rule.

$E \to I$

$E \to E+E$

$E \to E*E$

$E \to (E)$

$I \to E \mid 0 \mid 1 \mid 2 \mid \ldots 9$

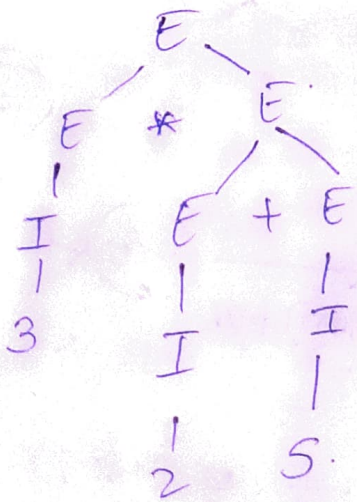
soln. we k.t $G = \{V, T, P, S\}$

$V = \{I, E\}$

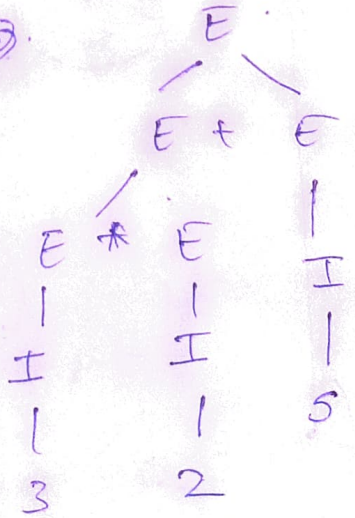
$T = \{+, *, (), E, 0, 1, 2 \ldots 9\}$

For the string "3*2+5" the above grammar can be generated two parse tree leftmost derivn.

(1)



(2)



Since there are 2 parse tree for a single string, the grammar is ambiguous.

(3)

Chomsky Normal Form

A CFG is in CNF if all production rules satisfy one of the following conditions

① A terminal symbol generating ∈

eg : $A \to \epsilon$

② A non terminal generating 2 non terminals

eg : $S \to AB$

③ A non terminal generating a terminal

eg : $S \to a$

For eg :

* $G_1 = \{ S \to AB$ (CNF from 2 rule)
  $S \to C$ ✓ CNF
  $A \to a$ ✓ CNF
  $B \to b$ ✓ CNF $\}$ 3 rule.

∴ G_1 satisfy the rule for CNF so it is in CNF.

* $G_2 = \{ S \to aA$ ✗ does not CNF
  $A \to a$ ✓ CNF.
  $B \to c$ ✓ CNF

∴ G_2 does not satisfy the rules so not in CNF.

(9) Instantaneous Description for PDA

* The Instantaneous description is called as an informal notation we explain how a push down Automata (PDA) computes the given i/p string
* Then it makes decision that the given string is accepted or rejected.
* The PDA involves both state & content of the stack.
* Stack is one of the most imp parts of PDA at any time.
* The factors of PDA notation by triple (q, w, y)

where,

q = current state.
w = remaining i/p alphabet.
y = current contents of PDA stack.

⊢ sign is called "turnstile notation" & represents one or more.

⊢* sign represents a sequence of moves.

$\delta [q_0, c, Z_0] \rightarrow [q_1, Z_0]$ gives

$P_{12} : [q_0, Z_0, q_0] \rightarrow c[q_1, Z_0, q_0]$

$P_{13} : [q_0, Z_0, q_1] \rightarrow c[q_1, Z_0, q_1]$

$P_{14} : [q_0, Z_0, q_2] \rightarrow c[q_1, Z_0, q_2]$

Similarly, we can find for

$\delta [q_0, c, a] = [q_1, a]$

$\delta [q_0, c, b] = [q_1, b]$

$\delta [q_1, a, a] = [q_1, \in]$ gives

$P_{15} : [q_1, a, q_1] \rightarrow a$

$\delta [q_1, b, b] = [q_1, \in]$ gives

$P_{16} : [q_1, b, q_1] \rightarrow b$

$\delta [q_1, \in, Z_0] = [q_2, Z_0]$ gives

$P_{17} : [q_1, Z_0, q_2] \rightarrow \in$.

Q70. Convert the following grammar to a PDA that accepts the language by empty stack.

$S \rightarrow 0S1|A$

$A \rightarrow 1A0 \mid S \mid \in$.

Answer : **(Dec.-18(R16), Q6(a) | May-17(R13), Q6(b))**

Given grammar (G) is,

$S \rightarrow 0S1|A$

$A \rightarrow 1A0|S|\in$

Let PDA be = $\{\Sigma, T, \delta, S, F)$

Where,

$Q = \{p, q\}$

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

$\uparrow = \{S, A, 0, 1\}$

$S = \{p\}$

$F = \{q\}$

δ is given as follows,

$R_1 : \delta(P, \in, \in) = \{(q, S)\}$

$R_2 : \delta(q, \in, S) = \{(q, 0S1)\}$

$R_3 : \delta(q, \in, S) = \{(q, A)\}$

$R_4 : \delta(q, \in, A) = \{(q, 1A0)\}$

$R_5 : \delta(q, \in, A) = \{(q, S)\}$

$R_6 : \delta(q, \in, A) = \{(q, \in)\}$

$R_7 : \delta(q, 0, 0) = \{(q, \in)\}$

$R_8 : \delta(q, 1, 1) = \{(q, \in)\}$

Let us apply this on string 0110 and check its acceptance on given grammar.

State	Input	Stack	Rule Used
p	011001	E	
q	011001	S	R_1
q	011001	0S1	R_2
q	11001	S1	R_7
q	11001	A1	R_3
q	11001	1A01	R_4
q	1001	A01	R_8
q	1001	1A001	R_4
q	001	A001	R_8
q	001	$\in$001	R_6
q	01	01	R_7
q	1	1	R_7
q	E	E	R_8 (Accepted)

∴ The constructed PDA accepts the language generated by the given grammar.

Q71. Construct PDA from the following CFG.

S → aAA

A → aS | bS | a.

(Model Paper-III, Q7(b) | Dec.