Date : 26/9

Computation :- eg Set of binary strings that ends with 0.

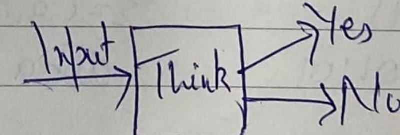
↓

Something that answers in Yes/No

eg Set of binary strings that represent a legal java code. → Yes. &- Compilers

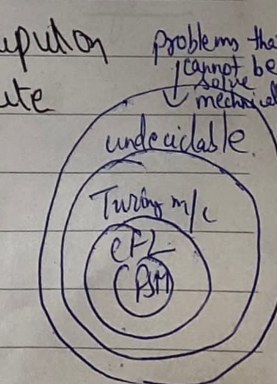
eg Set of all legal java program that will never go into an infinite loop.
↳ No

- What we can compute
- What we cannot compute
- What is model of computation
- How we can compute



Layers -

CFL
Set of strings



WONDER CEMENT

Date :

Symbols :- a-z, 0-1, ☺
A-Z
↓ ↓ ↓
alphabets letters picture

something used for representation

Alphabet (Σ) set of symbols

$\Sigma = \{1\}$ → Unary.
$\Sigma = \{0,1\}$ → Binary nos
$\Sigma = \{0-9\}$ → decimal digits

Strings :- Sequence of symbols taken from a particular alphabet.

$\Sigma = \{0,1\}$ $\Sigma = \{a,b\}$
= 0, 01101 abba

$\Sigma = \{0,1,2\}$
0110,

WONDER CEMENT

Date :

$\Sigma = \{0,1\}$ 3 digit

$\_ \_ \_ = 2^3 = 8$

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

$\underset{3}{\_} \; \underset{3}{\_} \; \underset{3}{\_} = 3^3 = 27.$

Language :- set of strings that are always defined on alphabet.

Raj → neither hindi nor eng.

$\Sigma = \{0,1\}$
L_1 = set of all binary strings of length exactly 2.

$L_1 = \{00, 01, 10, 11\}$ ⇒ finite lang.

strings which are defined on alphabet.

WONDER CEMENT

Σ^0 Σ^1 Date :

L_2: set of all strings defined over $\Sigma = \{a, b\}$ that starts with a. ⇒ Infinite

$\{a, ab, aa, aab, aaa \ldots\ldots\}$

finite → cardinality can be counted
Infinite → " cannot " "

$\{\}$

L_3: 1 is string band.

$\{1, 11, 111 \ldots\ldots\}$

Power of alphabet

Set of strings defined on the alphabet with exactly 1 length.

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

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

WONDER CEMENT

Date :

$\Sigma^2 = \{00, 01, 10, 11\}$ exactly length 2.

$\Sigma^3 = \{000, \ldots\ldots 111\}$ exactly length 3.

$\Sigma^0 = \{\Sigma\}$ → null string

$|\Sigma| = 0$

Length of string:- Counting the frequency / occurance of symbol in a string

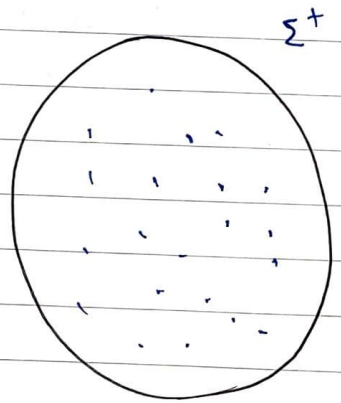
Kleene Closure

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$$

$$\{\Sigma\} \cup \{0,1\} \cup \{00,01,11,10\} \cup \cdots$$

contains all possible strings of any length including null string.
defined over an alphabet.

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



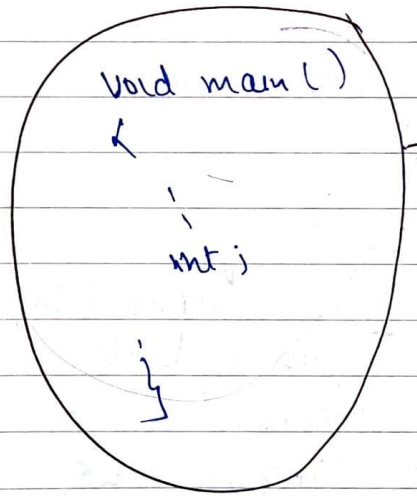
$L = \{00, 01\}$ → subset of $\{0,1\}$

* Language is always a subset of Kleene Closure

$\Sigma = \{a-z, A-Z, 0-9 \cdots\}$

↓

strings formed from alphabets



⇒ strings in TOC

ie program in C = string in TOC

Set of all valid programs is also a Language.

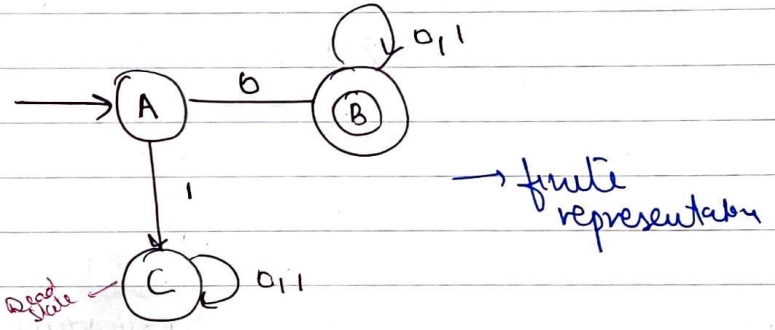
⇒ Given a lang. L & string S we are intrested weather the string belongs to lang. or not.

If we can find finite representation of this infinite set, so I will store it in the machine & then using that language we can answer Yes / No.

Machine that will answer Yes / No

finite automata.

Set of all binary strings starting with 0.



$\Sigma = \{ 0, 01, 00, 001, 010, 011 \cdots \}$

↳ infinite representation

as a Machine in a finite Yes/No can be answered but not in infinite rep.

WONDER CEMENT

Date :

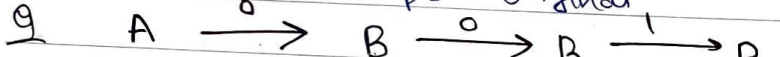
Representations

○ → state.

→○ → initial state (which will take the input)

◎ → final state / acceptance state

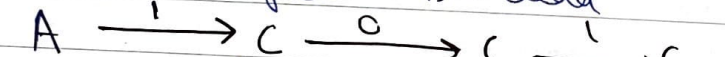
Eg "001" → accept B → final



when from initial we reach final state assuming all the states then machine says Yes I accept this input.

O/W No.

Eg "101 → reject → D → dead



here we didn't reach the final state so NO I can't accept this input

finite automata

Broadly it can be of 2 types:-

- FA without O/P Capability
  - DFA (Deterministic finite automata)
  - NFA (non deterministic automata)
  - ε-NFA (Epsilon-NFA)
- FA with O/P Capability
  - Moore Machine
  - Mealy Machine.

DFA :- A DFA is a 5 tuple collection

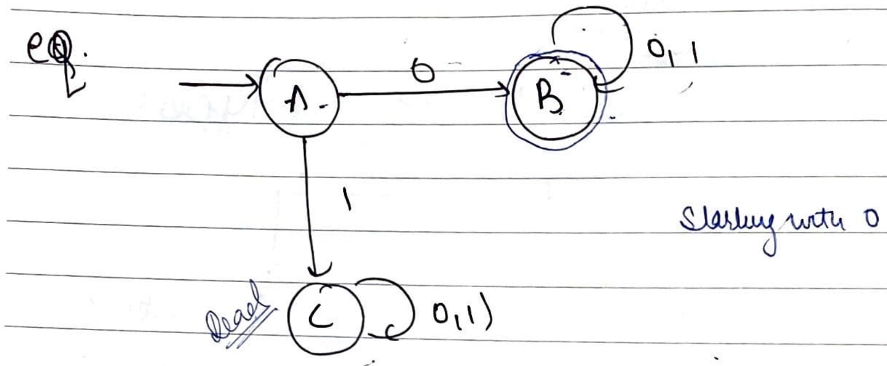
$(Q, \Sigma, \delta, q_0, F)$ → Order should be like this

tuple :- in which sequence of elements matter.

{1,2,3} → diff tuples
{3,2,1}

WONDER CEMENT

eg.



$Q \rightarrow$ finite set of states.
$\{A, B, C\}$

$\Sigma \rightarrow$ finite collection of input symbols
$\{0, 1\}$

$\delta \rightarrow$ ~~transition~~ transition ~~symbol~~ function

$q_0 \rightarrow$ $\{A\}$ initial state where input is given

$F \rightarrow$ finite set of final states.
$F \subseteq Q$

Date :

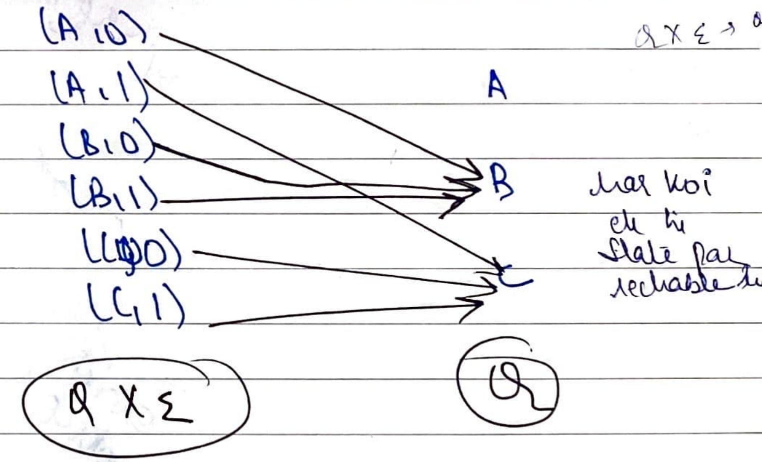
Transition function :- $A \xrightarrow{0} B$, $A \xrightarrow{1} C$ } Transitions

transition → moving from one state to another state &

a function which defines all the transition is a transition function

$Q \times \Sigma \rightarrow Q$

$Q \times \Sigma$
$\{A, B, C\}$ $\{0, 1\}$ → from diagram.
Cartesian Product



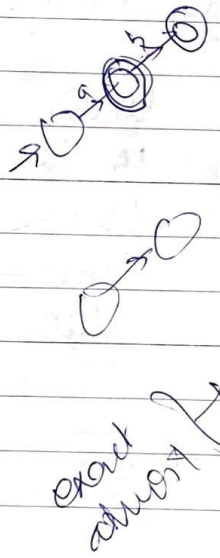
$f: Q \times \Sigma \rightarrow Q$

$f = Q \times \Sigma \rightarrow Q$

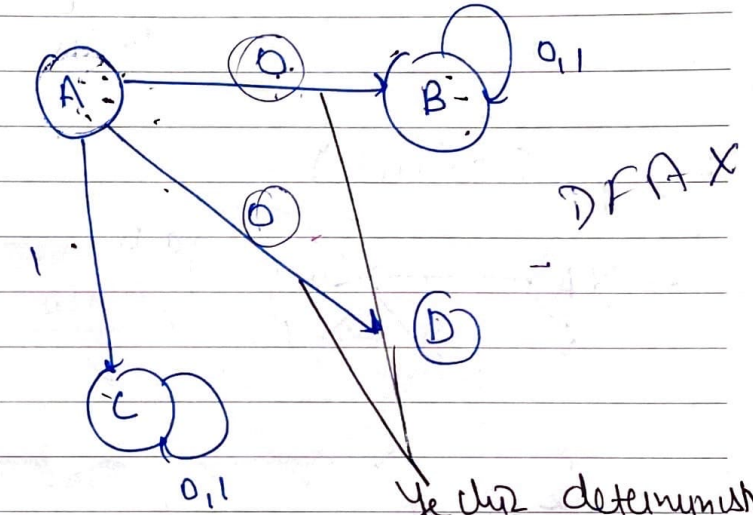
Yaha k ek combo k liye ek hi O/P milega

Date :

To accept 1 char :- 2 states req
2 char :- 3
3 char 4
4 char 5 + 1dead



hai state par jitni & mani state ho ek time par ek hi state possible hogi. (in DFA)



DFA X

NFA

DFA

Ye chiz deterministic nahi hai because yaha decision karna padega ki kaha jae

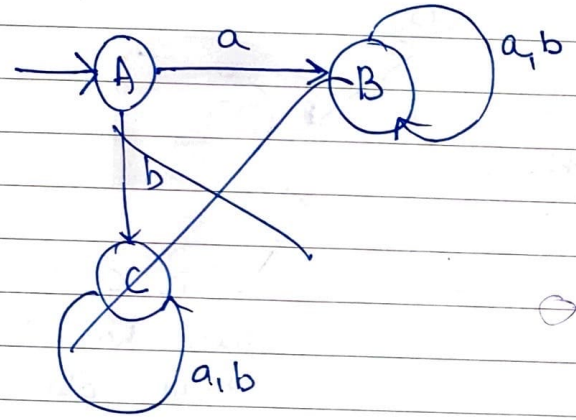
DFA

⇒ Design a DFA over the alphabet $\Sigma = \{a, b\}$ that will accept all strings w such that $|w| = 2$

Yes/No

length is 2.

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



~~$\Sigma = \{ab, a, b, \in\}$~~

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

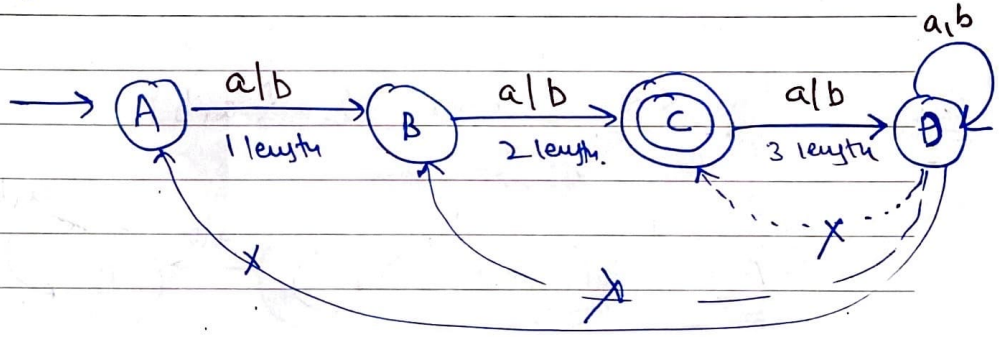
↳ finite set

⇓

so we can compute everything

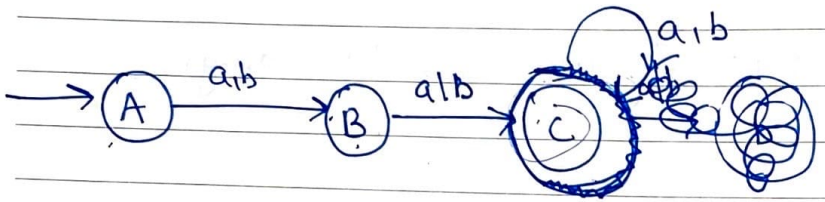
sabse phle initial state to chaye to agar hum A par khade hai means 0 length ki string.

B par doosra character ayega.

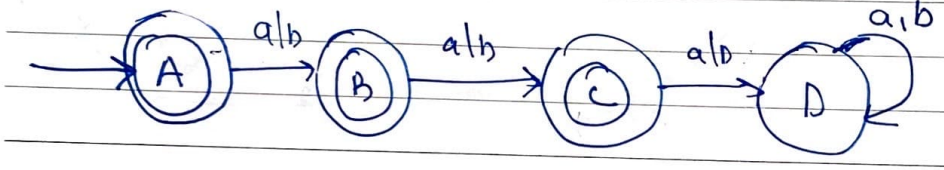


agar hum str/ C tak banaege to uske baad agar koi state aati hai to wo kabhi wapis nai a pae. Uske self loop

Q Design a DFA over some alphabet $\Sigma = \{a, b\}$ & define all strings such that $|w| \geq 2 = \{2, 3, 4 \cdots\}$



Q $|w| \leq 2$ over $\Sigma = \{a, b\}$



$\{ \varepsilon, a, b, aa, ab, ba, bb \}$
↳ finite

(A) → This will accept 0 length string

⇒ NOTE

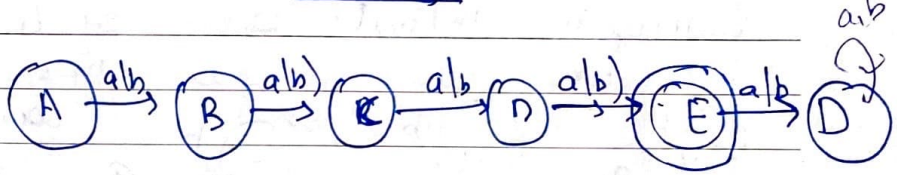
agar initial state ko hi final state bana rahe ho ie it will accept ε

NOTE

* atleast me kabhi bhi deadstate nahi ayegi.

atmost & equal to me deadstate atti hai.

Q → find the no of states in alphabet $\{a, b\}$ with exactly length 4.



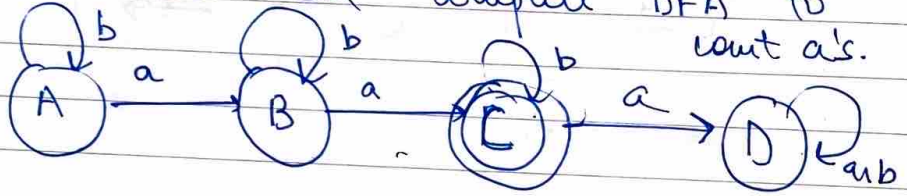
5 + (1) ⇒ dead

= 6 states.

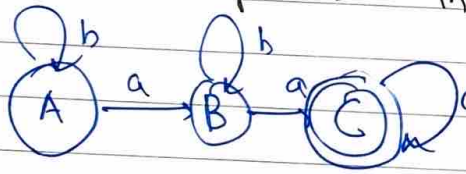
Q Design a DFA over the alphabet {a,b} that will accept all those strings w such that no of a's in w is 2. exactly

Σ = { aa, aab, ~~baab~~ baa, aba ... ... abba ... }

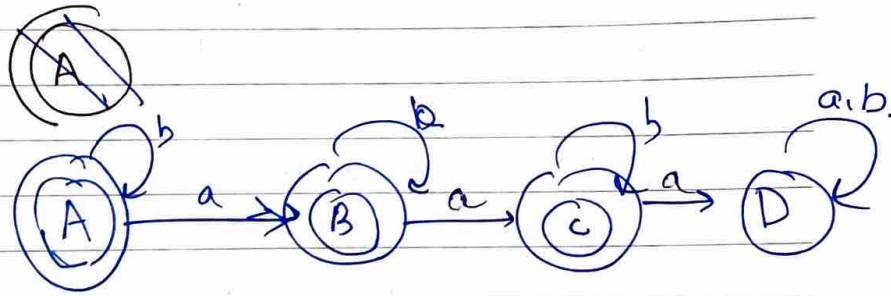
Language is infinite because of b. So we can compute DFA to count a's.



Q no of a's in w is ≥2



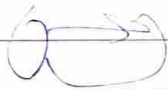
Que $n_a(w) \leq 2$

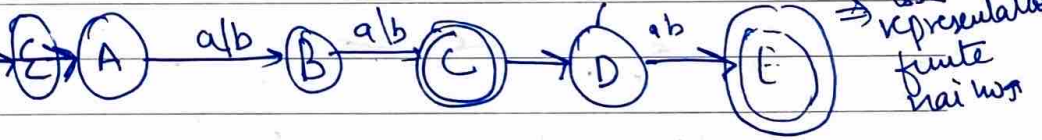


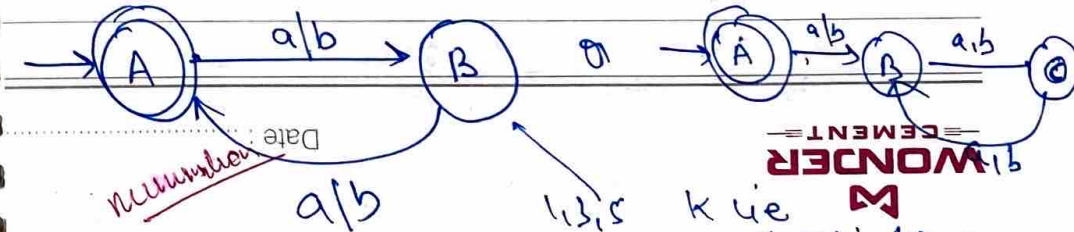
Q Design a DFA over Σ = {a,b} such that |w| mod 2 = 0

0, 2, 4, 8, 16 .....

{ aa, ab, ba, bb, aaaa, aabb ... }

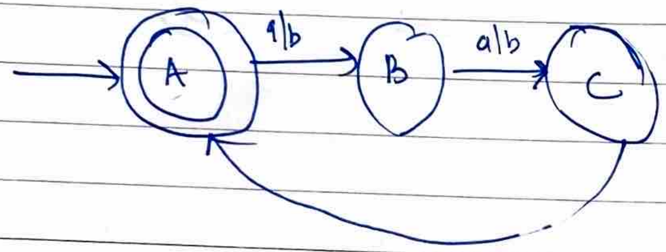






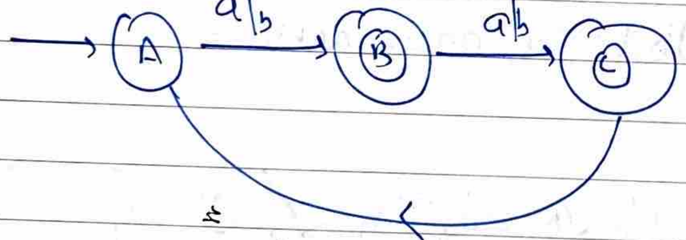
✓ Design a DFA over an alphabet {a,b} which will accept |w| mod 3 = 0

0, 3, 6, 9 ....
↓
(0, 1, 2) ⇒ remainder



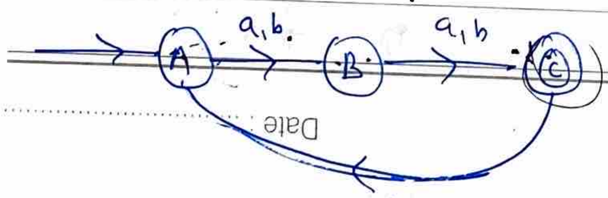
✓ |w| mod 3 ≠ 0

(not equal or complement kame
K the initial → final &
final → initial)



✓ |w| mod 3 > 1

0, 1, (2) > 1
0, 1, (2)
↓
final

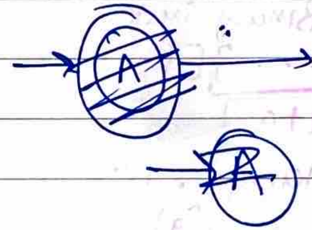


~~last 3~~ → 1

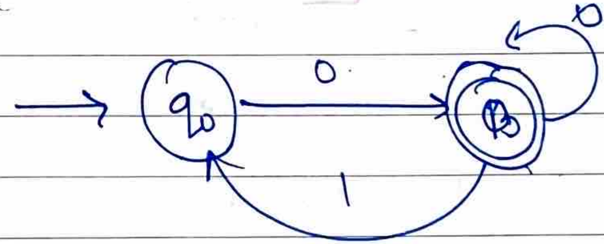
Q4 Design a DFA over the alphabet = {0,1} that will accept all those strings which when interpreted as binary no. then their decimal equivalent should be divisible by 2.

{0,1}   01   21

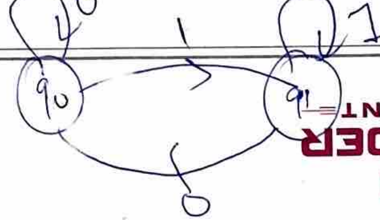
0, 2, 4.
{zero, 10, 100 ....



: divisible by 2 when last bit is 0.



	0	1
0	r0	r1
1	r0	r1



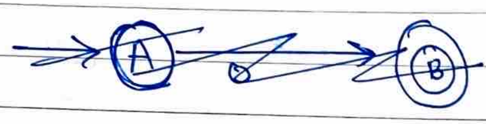
Q: All ternary no. representation in binary divisible by 2.

ternary → 0, 1, 2 $( )_3$

000
001

$(12)_3 = 5$

$(11)_3 = 4$



⇒ Deadlock represents rejection ie. permanent rejection

Number System

1 → 1

14 → 1 × base + 4
= 1 × 10 + 4
= 14.

Base 10

* In Binary System
(101 . . .) [a]
[2x + a]

* In ternary system
[012012 . . .] [a]
[3x + a]

143 → 14 × 10 + 3 = 143

Binary

1

10 → 1 × 2 + 0
= 2.

101 → 2 × 2 + 1
= 5

(101 . . . .) [a]
x

x is the value of binary no.

[2x + a] → This is for binary no's.

formal ternary no. [3x + a]
octal no. [8x + a]

Date :

Date :

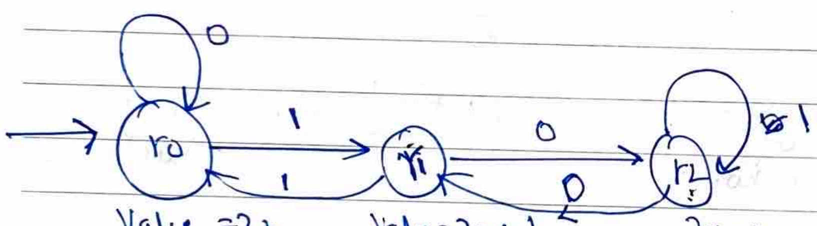
Q. Design a DFA over the alphabet {0,1} that will accept all binary no.'s who's decimal equivalent is divisible by 3. $= 3x$

{0,1}

	0	1
r_0	r_0	r_1
r_1	r_2	r_0
r_2	r_1	r_2

3 states.

1 represent 0 remainder r_0
2 " 1 remaint r_1
3 " 2 rem r_2



Value $= 3x$ Value $= 3x+1$ $3x+2$

$\boxed{3x}\boxed{1}$ $3x \times 2 + 1 = \underline{6x+1}$ → r0 se r1 par we get remaind = 1 r1 pe move karo.

r0 me kitne bhi zero aiye r0 hi rahega.

$\boxed{}$ 0

$3x+1$

$= (3x+1)2 + 0$ → b10 2 binary no

$= \underline{6x + 2}$ ↓ rem

	0	1
r_0	r_0	r_1
r_1	r_2	r_0
r_2	r_1	r_2

0 → r_2
(0 se us state par move karenge jaha rem = 2)

	0	1
r_0	r_0	r_1
r_1	r_2	r_0
r_2	r_1	r_2

$(3x+1) \times 2 + 1$

$\underline{3x+3}$ → rem = 0 ⟶ move to r_0

$\boxed{}$ 1

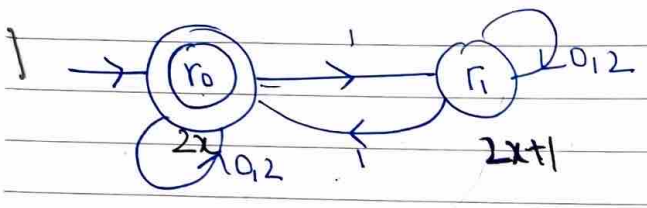
$3x+2$

$(3x+2)2 + 1$

$\boxed{3x+5}$ → Rem = 2 move to r_2

Date :

Q→ all ternary no. divisible by 2



	0	1	2
r0	r0	r1	r0
r1	r1	r0	r1

$(2x)_3 + 0$ → $6x$ → r_0 (rem = 0 when divided by 2)

$(2x)_3 + 1$ → $6x+1$ → r_1 (rem = 1 when divided by 2)

$(2x)_3 + 2$ → $6x+2$ → r_0 (when divided by 2)

$(2x+1)_3 + 0$ → $6x+3$ → r_1 (rem = 1)

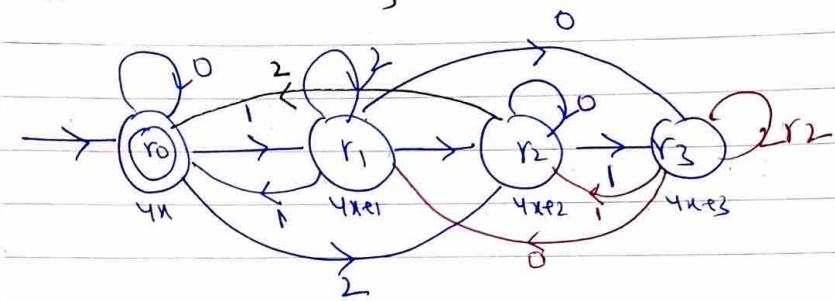
$(2x+1)_3 + 1$ → $6x+4$ → r_0 (divisible by 2)

$(2x+1)_3 + 2$ → $6x+3+2$ → $6x+5$ → r_1

$6x+3$ → which is divisible by 2

Q→ all ternary no. divisible by 4

r_0, r_1, r_2, r_3 =



$(4x)_3 + 0$ → $12x$ → r_0

$(4x)_3 + 1$ → $12x+1$ → r_1

$(4x)_3 + 2$ → $12x+2$ → r_2

$(4x)_3 + 3$ $(4x)_3 + 4$

$12x+3$ $12x+4$

r_3 r_0

$(4x+1)_3 + 0$ → $12x+3$ → r_3

$(4x+1)_3 + 1$ → $12x+4$ → r_0

$(4x+1)_3 + 2$ → $12x+5$ → r_1

	0	1	2
0	r0	r1	r2
1	r3	r0	r1
2	6x+3	6x+4	6x+5
	r1	r2	r3

$(4x+2)_3 + 0$ → $12x+6$ → r_2

$(4x+2)_3 + 1$ → $12x+7$ → r_3

$(4x+2)_3 + 2$ → $12x+6+2$ → $12x+8$ → r_0

$(4x+3)_3 + 0$ → $12x+9$ → r_1

$(4x+3)_3 + 1$ → $12x+10$ → r_2

$(4x+3)_3 + 2$ → $12x+11$ → r_3

WONDER CEMENT

Date :

SHORT CUT

✓ ternary system = {0,1,2}

✓ divisible by (4) → remainder {0,1,2,3}
r_0 r_1 r_2 r_3

	0	1	2
r_0	r_0	r_1	r_2
r_1	r_3	r_0	r_1
r_2	r_2	r_3	r_0
r_3	r_1	r_2	r_3

→ write the counting 0 to r_i in the order.

divisible by 5 → remainder {0,1,2,3,4}
r_0 r_1 r_2 r_3 r_4
ternary system

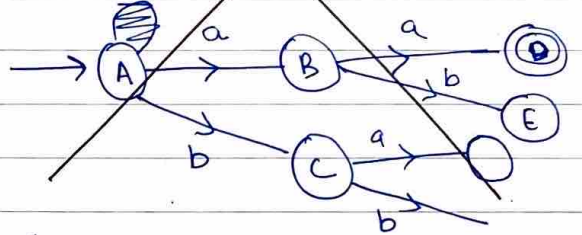
	0	1	2
r_0	r_0	r_1	r_2
r_1	r_3	r_4	r_0
r_2	r_1	r_2	r_3
r_3	r_4	r_0	r_1
r_4	r_2	r_3	r_4

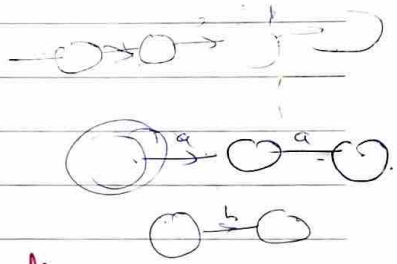
WONDER CEMENT

Date :

$n_a(w)$: x
$n_b(w)$: y
(x mod 2, y mod 2)

Q. DFA that accept all strings w over $\Sigma = \{a,b\}$ such that
n_a mod 2 = 0
&
n_b mod 2 = 0





* To read a → 3 states
To read b → 2 states

$3 \times 2 \rightarrow 6$ states + 1 (dead)

* ⇒ To read $n_a(w) = x$
$n_b(w) = y$

$n_a(w)$: x
$n_b(w)$: x
(x+1)(y+1)

$$(x+1)(y+1) + 1 \text{ states}$$

dead

na

a) $n_a(w) = 2$ & $n_b(w) = 2$ → exact / exactly → exactly ↓ dead state

$(2+1)(2+1)+1$

$= 3 \cdot 3 + 1 = 10$ states

b) $n_a(w) = 2$ & $n_b(w) \geq 2$ → exactly ↓ dead state

(=) ↓ exactly dead state required

$= 10$

c) $n_b(w) \geq 2$ & $n_a(w) \geq 2$ → atleast ↓ no DS

$= 9$ ⇒ no dead states req.

$(3)(3) = 9$ → We require dead states for atmost & exactly

exactly } atmost } → deadstate
atleast → no dead

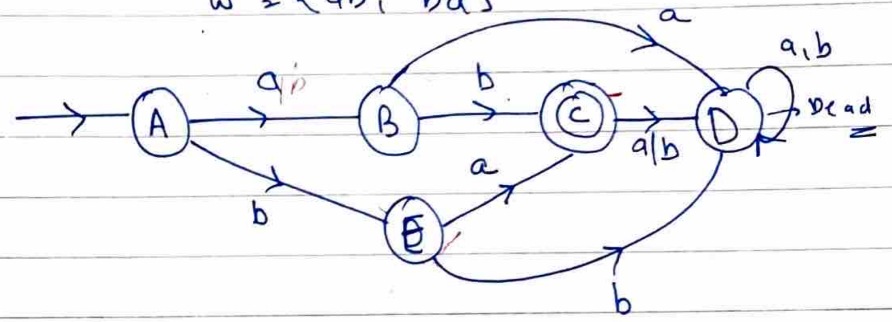
exactly & } atmost } → deadstate
atleast } → no deadstate

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

$(1+1)(1+1)+1 = 4+1 = 5$

Q → Design $n_a(w) = 1$ & $n_b(w) = 1$

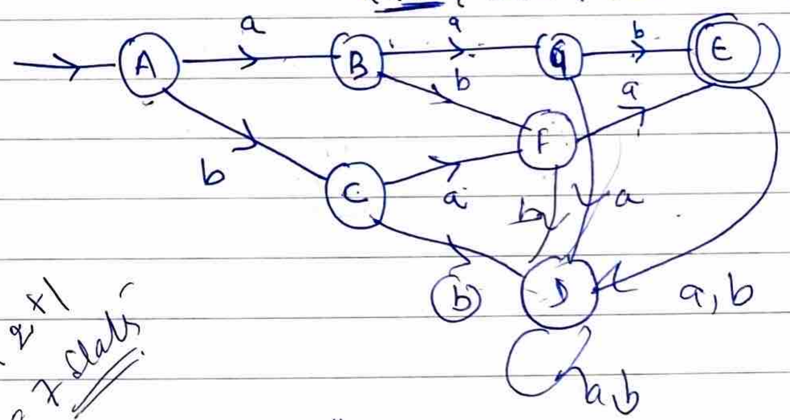
$w = \{ab, ba\}$



Q → $n_a(w) = 2$ & $n_b(w) = 1$

$\{aab, aba, baa\}$

$(2+1)(1+1)+1$
$3 \times 2 + 1$
$= 7$



$3 \times 2 + 1$
$= 7$ states

$3 \times 2 + 1$

WONDER CEMENT

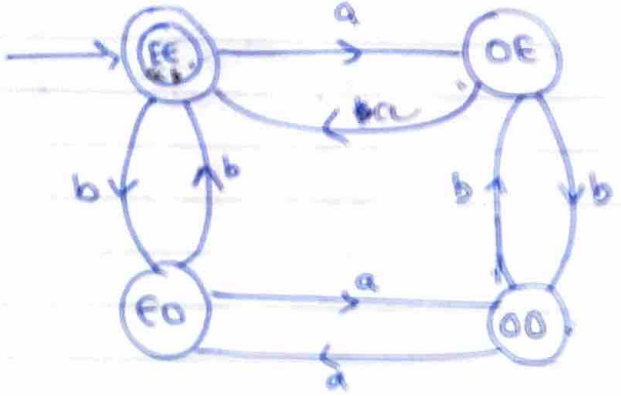
Q. Design a DFA

$n_a \bmod 2 = 0$ &

$n_b \bmod 2 = 0$

a	b
E	E
E	O
O	E
O	O

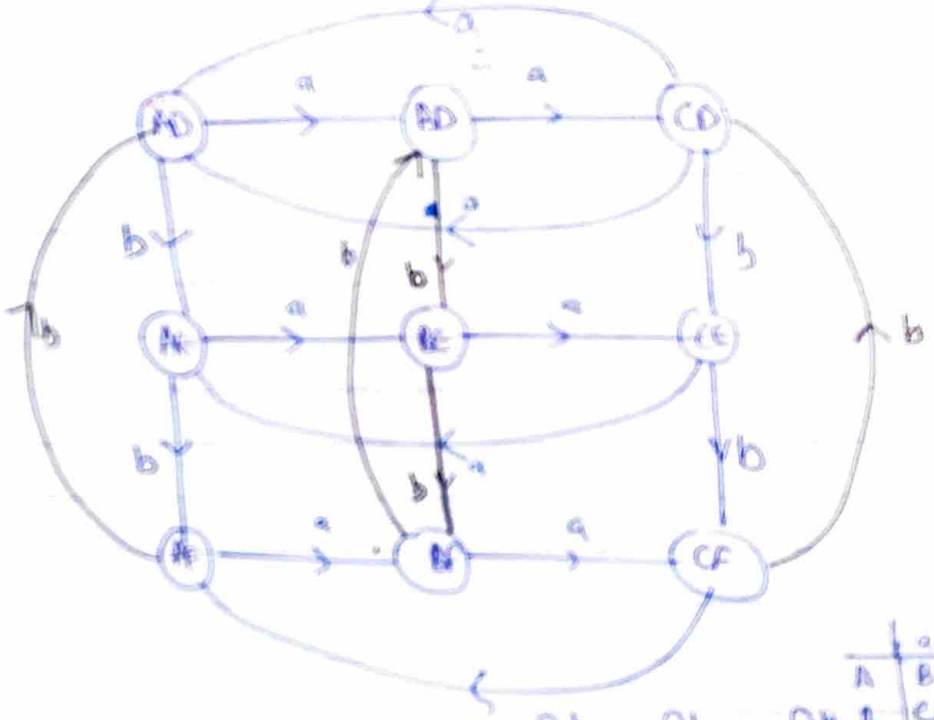
O → odd
E → even



E E
a b
again when a appears
to E → O
O E
a
E E

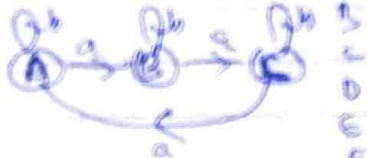
WONDER CEMENT

Design a DFA that will accept all strings w over alphabet $\Sigma = \{a, b\}$ such that $n_a \bmod 3 = 0$ & $n_b \bmod 3 = 0$ (0, 1, 2)



	a	b
A	B	A
B	C	B
C	A	C
D	E	F
E	F	D
F	D	E

$n_a \bmod 3 =$



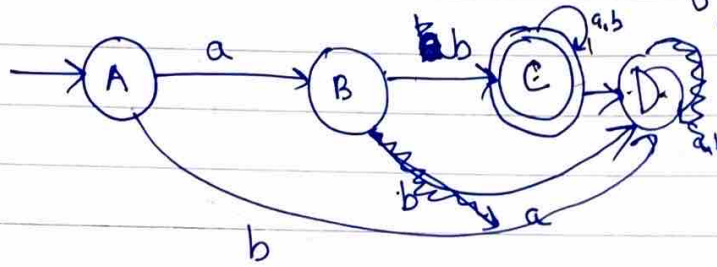
$n_b \bmod 3$

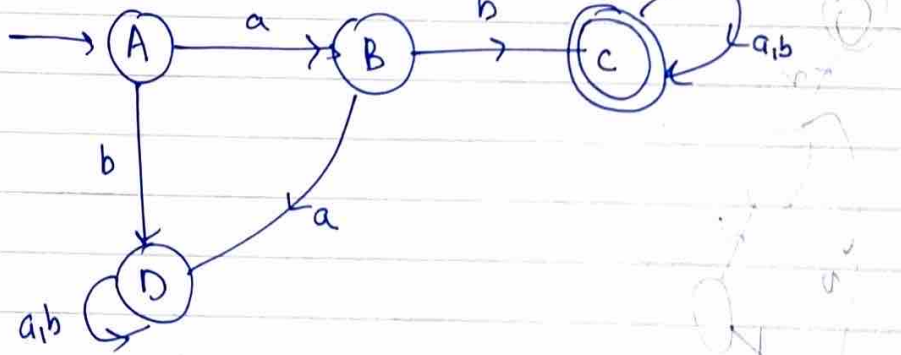
WONDER CEMENT

Date : 27/9

Q4 Design a DFA over the alphabet {a,b} that will accept all strings. w: (understood) w starts with ab

L = {ab, aba, abb ..... } → first construct dia for smallest string

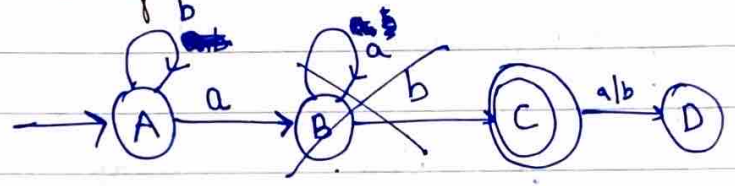




WONDER CEMENT

Date

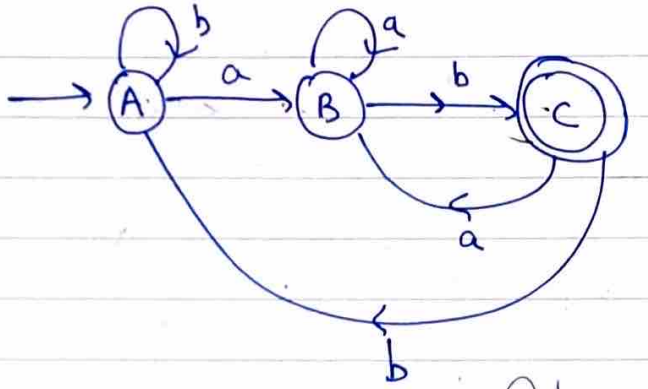
Q5 → Construct a DFA over the alphabet {a,b} that will accept all strings which ends with ab.

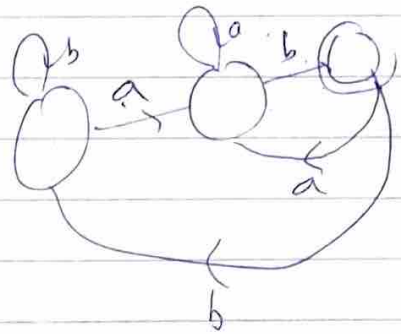


ab
aab
bab
aaab
bbab

L = {ab, aab, bab ..... }

baaaab a





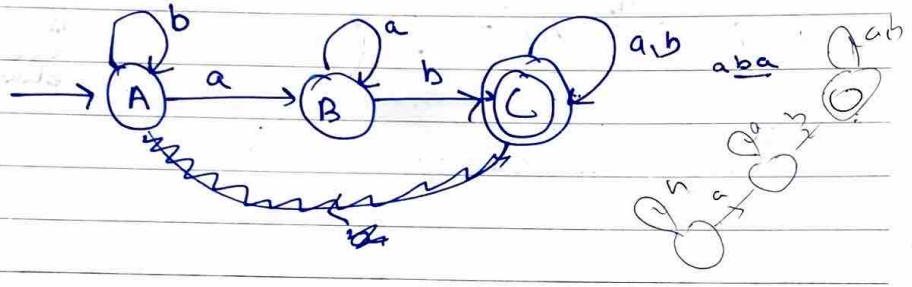
WONDER CEMENT

Date :

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

w: all strings contain ab

L: {ab, aab, bab, abg, abb ---- }

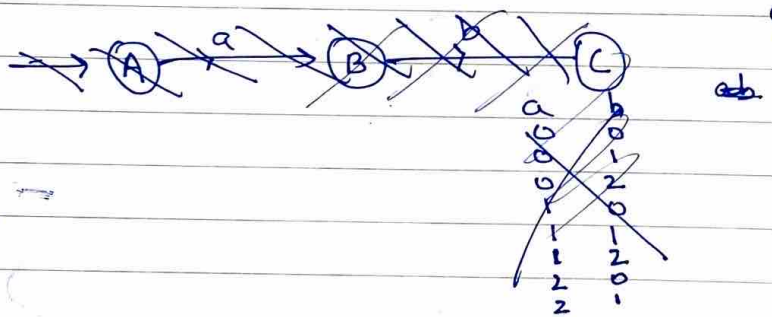


aba

Q. Construct a DFA over alphabet {a, b} that will accept all those strings w that satisfies $n_a(w) \bmod 3 > n_b(w) \bmod 3$

0 1 2 0 1 2

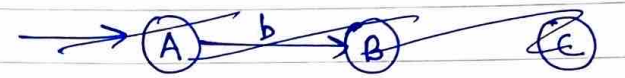
abba



~~ab~~

WONDER CEMENT

Date :

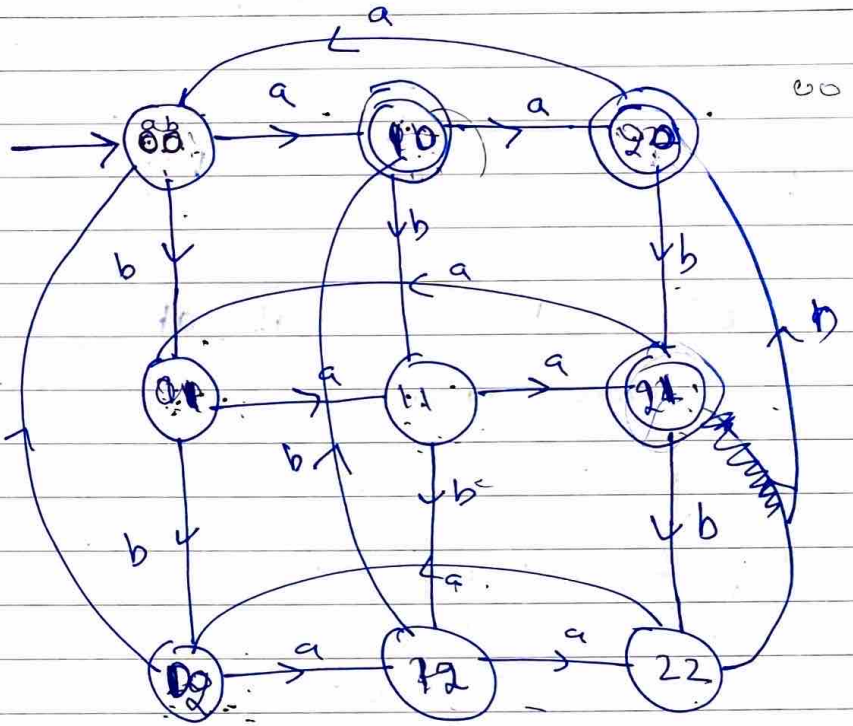


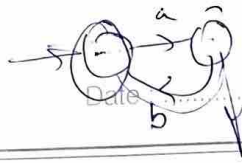
00
01
02
10
11
12
20
21
22

$n_a(w) \bmod 3$ $n_b(w) \bmod 3$

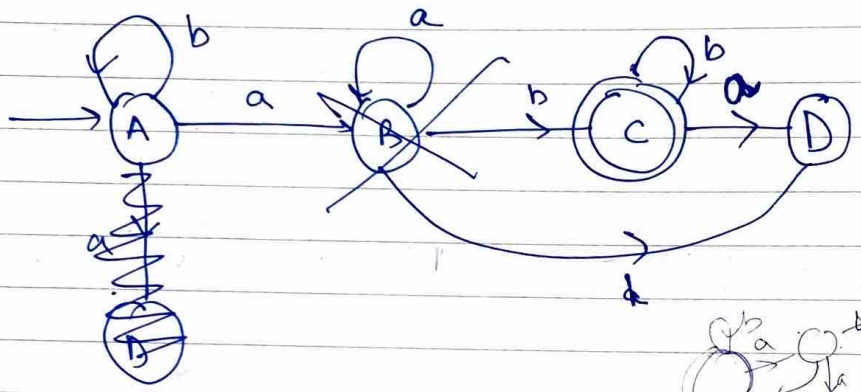
0, 1, 2 0, 1, 2

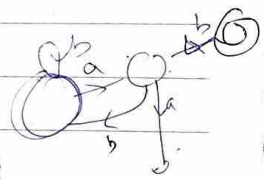
↳ 9 possibility





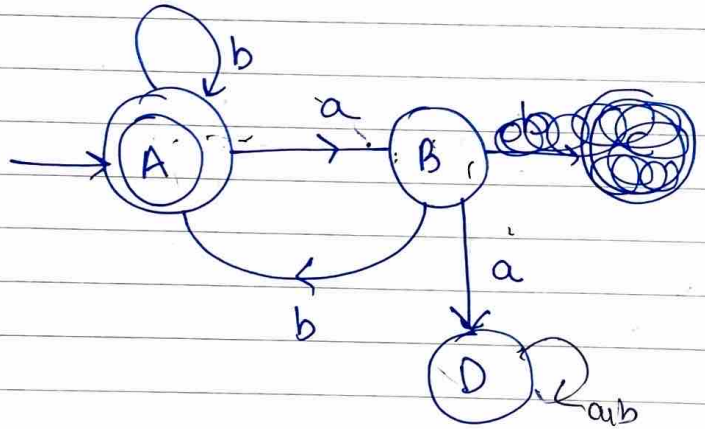
Q. Construct a DFA over the alphabet {a,b} that will accept all those strings in which every a should be immediately followed by b.

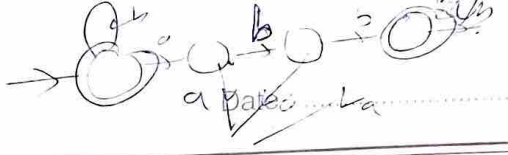




L: { ~~ab~~ ε, b, bbbb ..., ab, abb

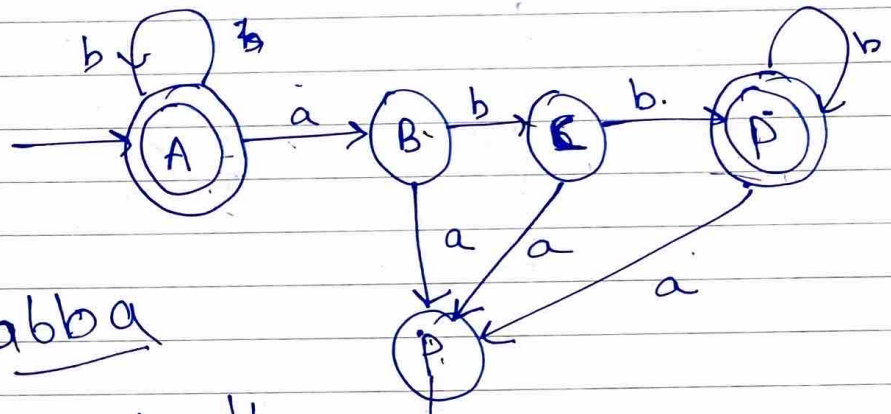
↳ here cond^n is on a.



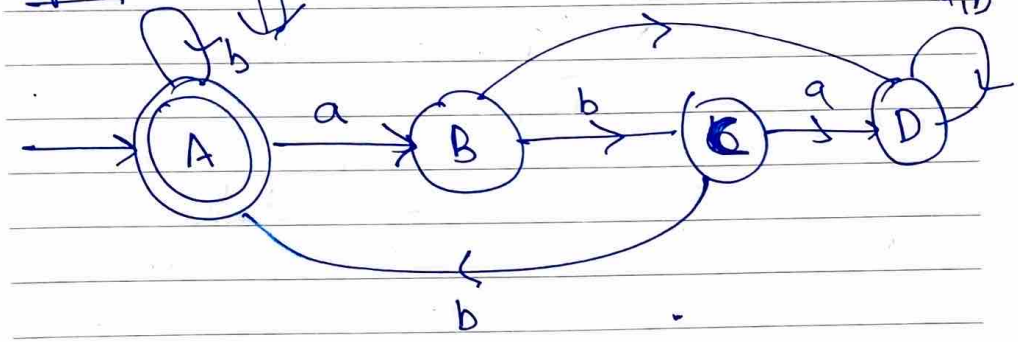


Q. Construct a DFA over alphabet {a,b} that will accept all those strings in which every 'a' is followed by immediately bb.

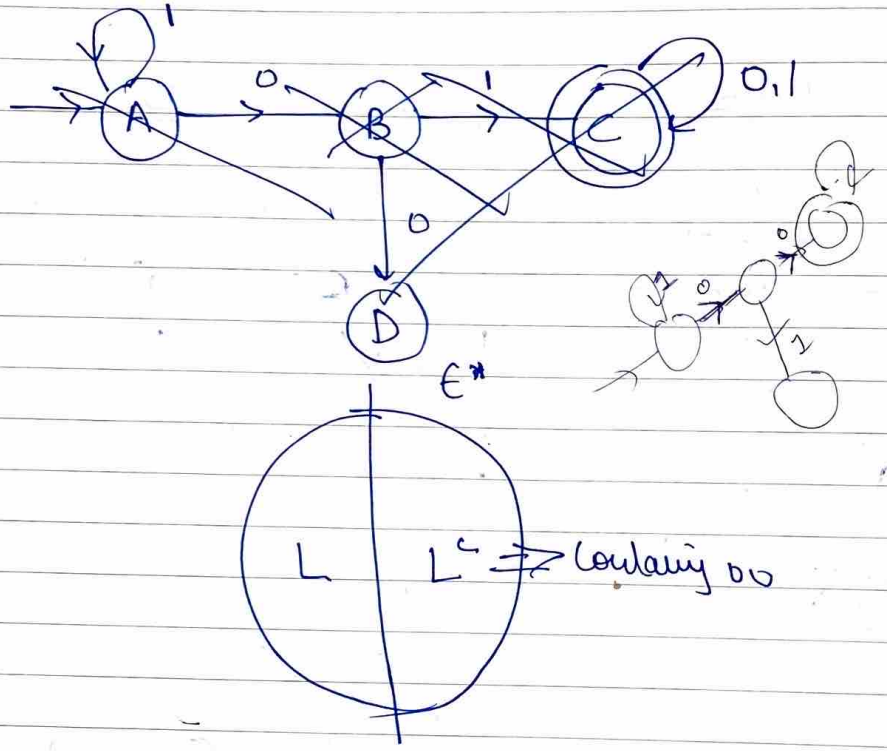
{ : { ε, bbb ..., abb abba



minimised form



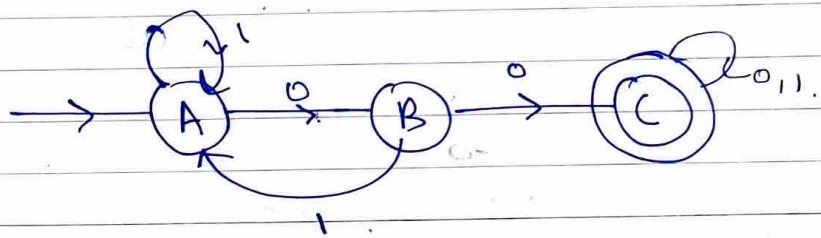
Q. DFA which will accept all strings not having 00 as substring



Initial state → final state

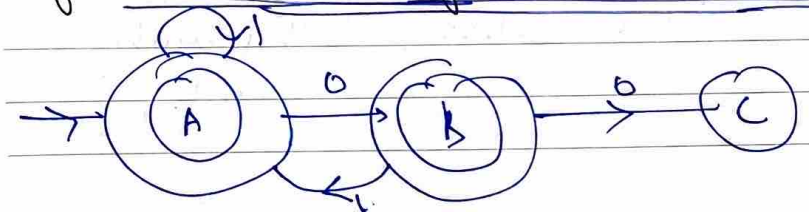
* Kleen's Theorem (only for DFA)

If we have a DFA for a language (L) & we want to find out the DFA for (L^c), we just need to complement DFA. ie Initial state becomes the final state & vice versa.



Agar B par loop lagaenge to 101 ka pattern ban jaega

* Kleens Theorem is applicable only for DFA not for NFA or ENFA



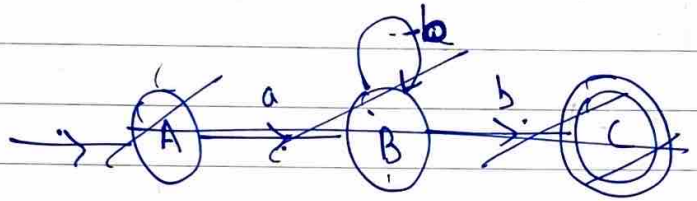
Q→ DFA is $\Sigma = \{a, b\}$ all strings in which first symbol is same as last symbol

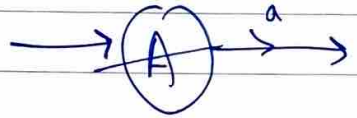
$L = \{$ ~~ab~~, aa, bb, ~~aba, bab~~

↓

Starting & ending with epsilon

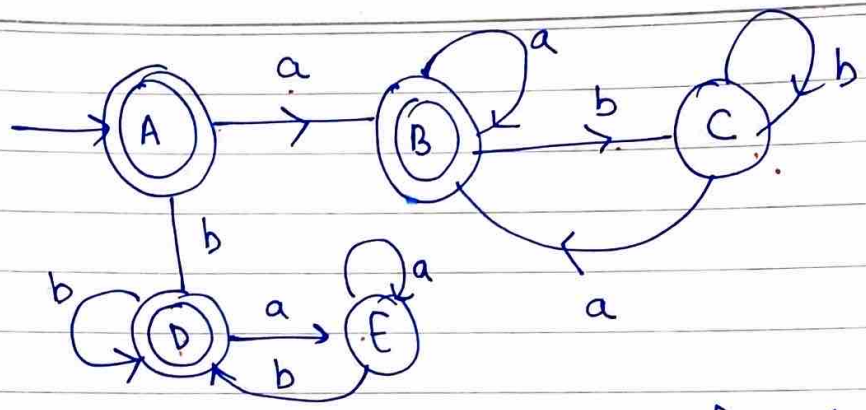
$\in ab \in$



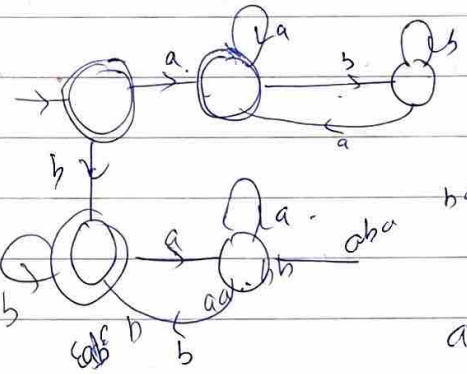


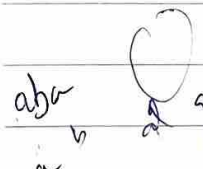
$L = \{ \varepsilon, a, b, aa, bb, aba, abba, abbba, bab, baab, baab \ldots$

C & t are temporal rejector aba



C se ~~aa~~ a aane par B & A dono par jasakte but but hum B par jaenge kyuki agar C se A par jaenge to uske baad A decide nai kar ~~patta~~ paega K a walla path lu ya b walla.





Q Construct a DFA over the alphabet ~~A,B~~ {a,b} that will accept all those strings in which first symbol is different from last symbol.

hum complement kar sakte hai but use phle lang.

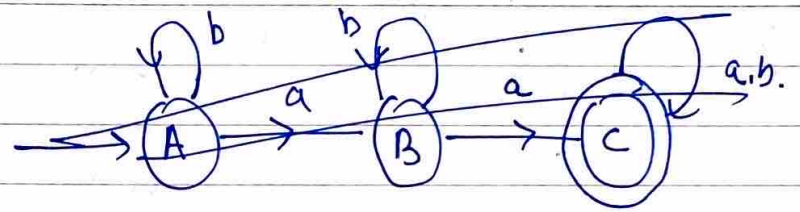
{aa, bb . . . }

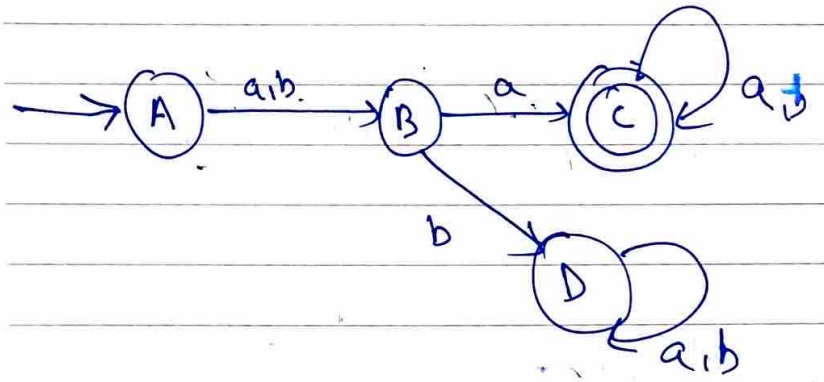
~~if~~

same dia but jo jo initial states hai & unko final bana do & vice versa.

Q → DFA {a,b} that will accept all those strings in which second symbol from LHS is a.

L: { aa, aab, bab, baa, ~~aab~~, aaa . . . .

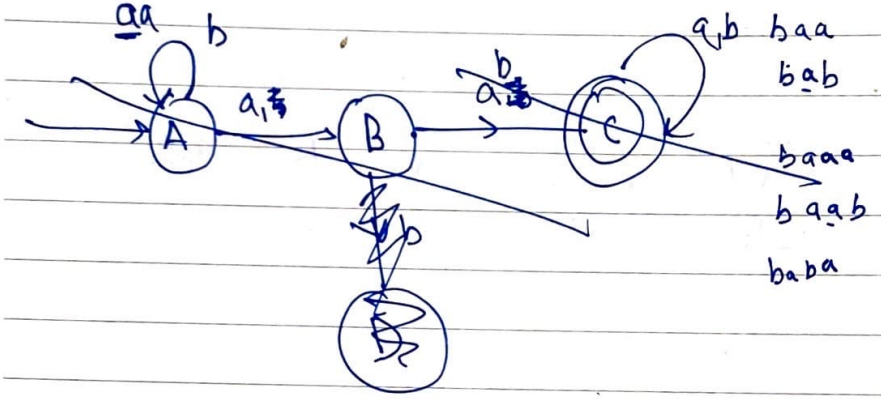




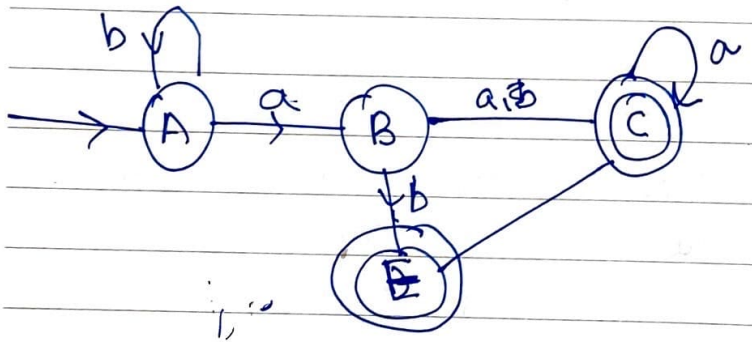
ques. me second symbol ki baat horahi hai hume first symbol de lena dena hai hai

N.f

Q. Construct a DFA over alphabet {a,b} that will accept all those strings, string in which second last or second symbol from RHS should be a.



L: {aa, ab, aab, bab, aaa}



→ This problem is difficult to design in DFA, so NFA came., will discuss later.

Machine ko sirf last k. do symbol yaad karne hai tbhi to decide karega 2 ke liye we have 4 possibilities

aa
ab
ba
bb.

→ will discuss in NFA

DFA

Q. Construct DFA {0,1} that will accept all strings in which 5th last digit is 0.

States = $2^5 = 32$

↓

Machine ko sirf last k 5 digit yaad rakhne hai

NFA

Non Deterministic finite automata.

Diff with DFA is:-

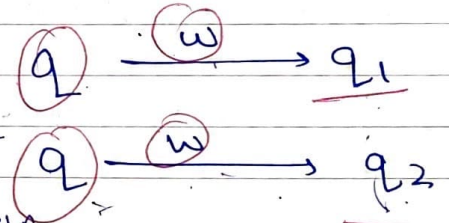
ek se start hokar ek par hi terminate karte hai.

But in NFA we can end up in many ~~dot~~ states.

we are not determined where to go

q par w input ane par q_1 par ja sakte hai

q par w input ane par q_2 par ja sakte hai



& so on

we make multiple states

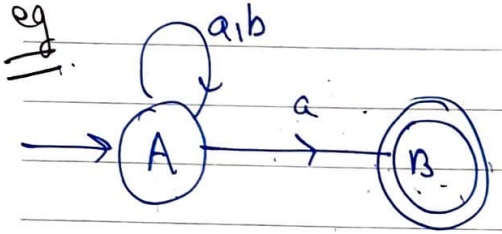
WONDER CEMENT

Date :

Ef in chess for one move opponent can ~~move~~ ~~nnn~~ take multiple moves, but at a time only one move is taken.

$\Sigma(a,b)$

eg



NFA for ending with a?

hum bas a se end. karne se matlab hai baki use pure a,b kuch bhi aye kya fark padta.

means at A if a comes we hai can move to B or stay at A.

Starting from one state for a particular input we can move to set of states.

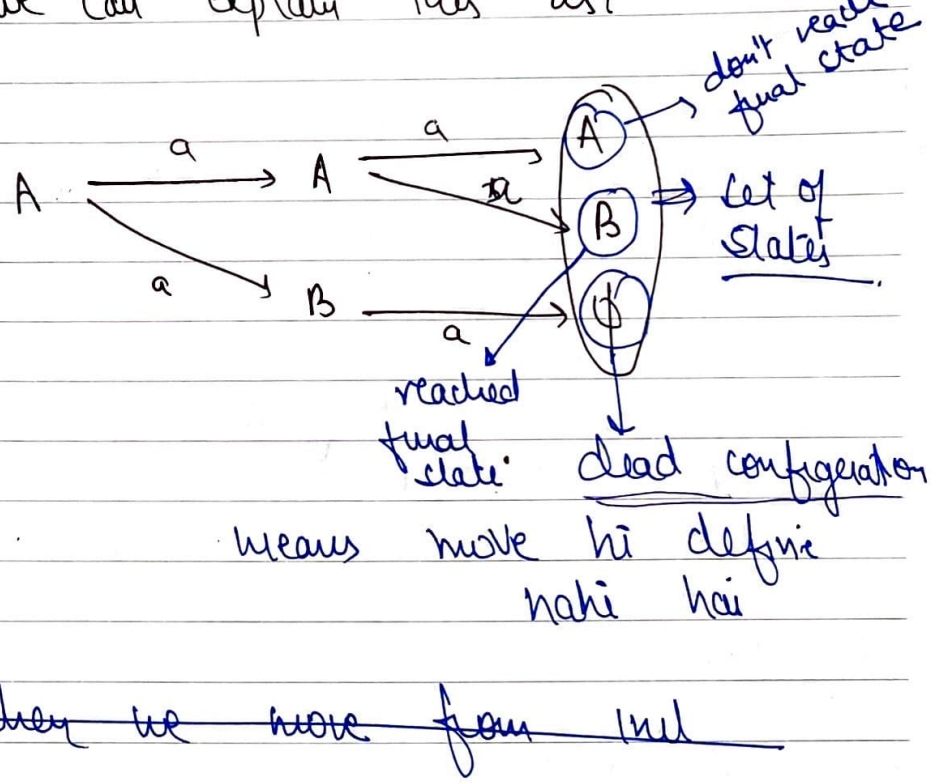
$\delta(B,a)$

↳ we are at B and a comes.

delta (transition fn)

NFA says we ~~can~~ are not certain to as where to go in next move.

We can explain this as!



means move hi define nahi hai

~~when we move from null~~

Starting from initial state, consuming the whole I/P, we will end up on a set of states & if atleast one of them is a final state

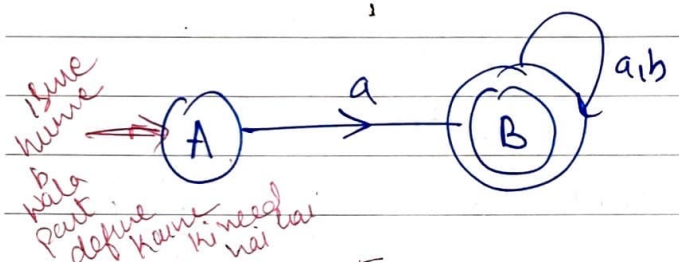
* In case of DFA if alphabet {a,b} then has state se 2 transitions

then only the string w said to be accepted else not accepted.

Designing of NFA's

Q→ ~~Starting with a.~~

Construct a NFA which will accept all strings start with a. Over $\Sigma = \{a,b\}$



Advantage :-

① easy to design.

Now in this if

ab comes

A —a→ B —b→ B.

bab

A —b→ ϕ dead configuration
'ie move is not defined

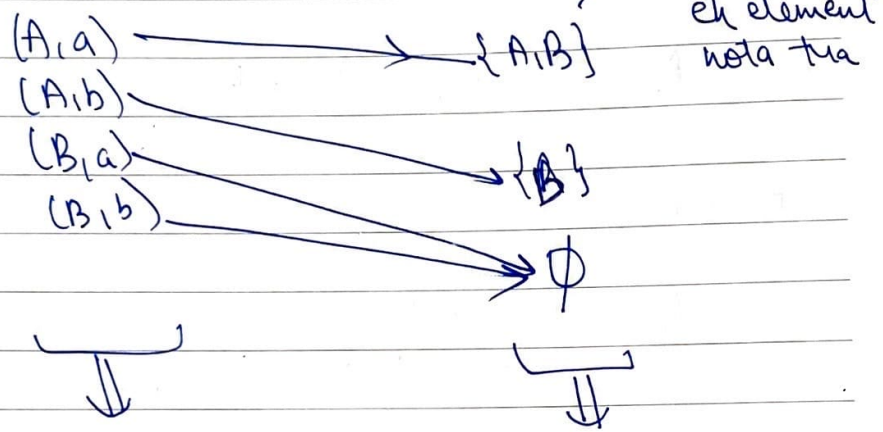
* NFA is a 5-tuple collection

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

- Q → {A,B} — set of states
- Σ → input alphabet {a,b}
- δ → define transitions
- q_0 → initial state {A}
- f → set of final states {B}

Q X Σ

{A,B}. X {a,b}



$Q \times \Sigma \rightarrow 2^Q$ NFA (all possible)

But in case of DFA

$Q \times \Sigma \rightarrow Q$ DFA

$|x| \cdot Q$

$Q \times \Sigma : Q^0$

DFA is a subpart of NFA

* Every DFA is a NFA but converse may not be true.

Q→ Is NFA as much powerful as DFA?

(NFA)

for this,

① DFA → NFA → always

Construct DFA for every NFA But we don't need to do this implicitly As we know every DFA is a NFA

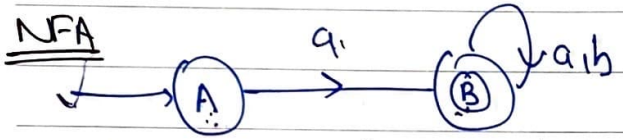
② NFA → DFA

This needs to be done

Conversion of NFA to DFA

Method used :- SUBSET CONSTRUCTION METHOD

NFA



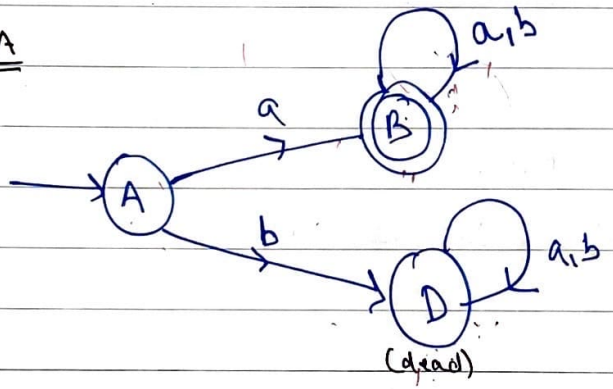
NFA for strings starting with a.

Transition Table

	a	b
A	B	ϕ
*B	B	B

final state

DFA

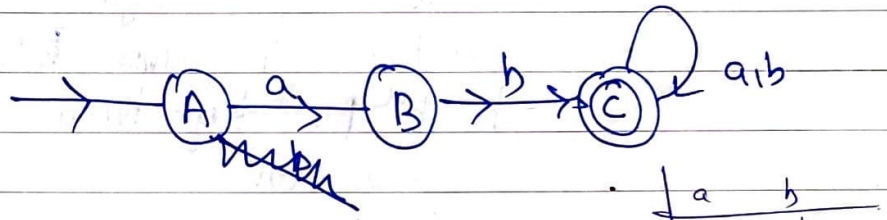


⇒ DFA for the corresponding NFA

DFA me final state wo hai jaha jaha B hoga.

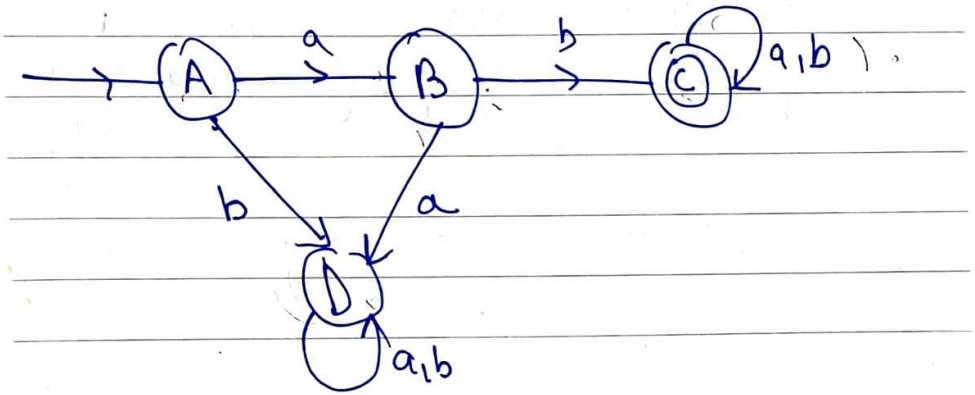
Make B as final state

Q→ Construct an NFA that will accept all strings starting with ab



	a	b
A	B	ϕ
B	ϕ	C
*C	C	C

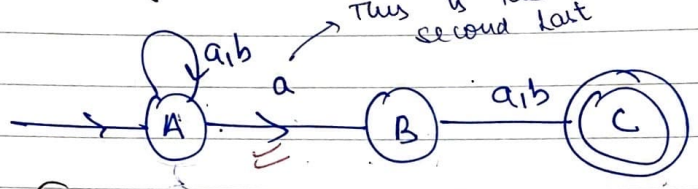
Now convert in DFA



WONDER CEMENT

Date :

Q. NFA for ~~toi~~ second last symbol as a

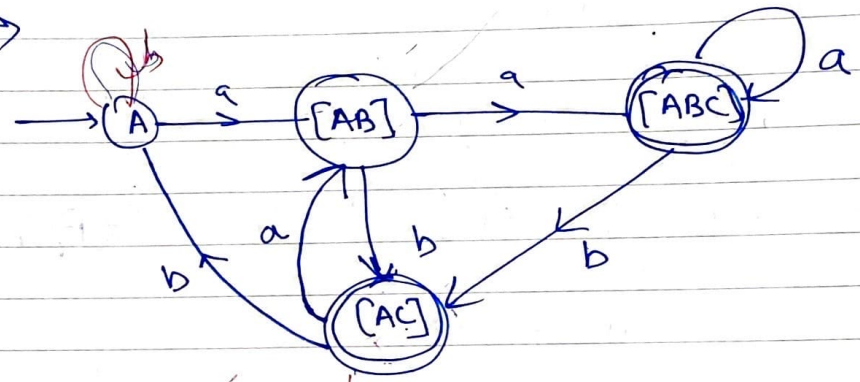


Now convert in DFA

[AB] → A ka a par kya milta hai union B ka a par ~~kuy~~ kaha jana hai

	a	b
A	{A,B}	{A}
B	{C}	{C}
C	—	—

AB ka a merge {A,B} {C}



[AB] [AC] [ABC],

Ye sirf merging ki hui hai

& also NFA me C was final state, ~~so~~ jaha jaha C hai waha final state ayegi

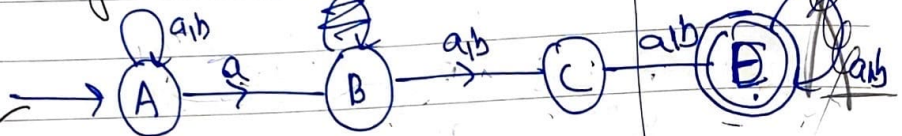
WONDER CEMENT

Date

WONDER CEMENT

Date

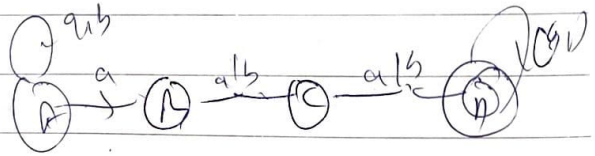
Q. Construct a DFA over alphabet {a,b} that will accept all those strings in which third last symbol is a.

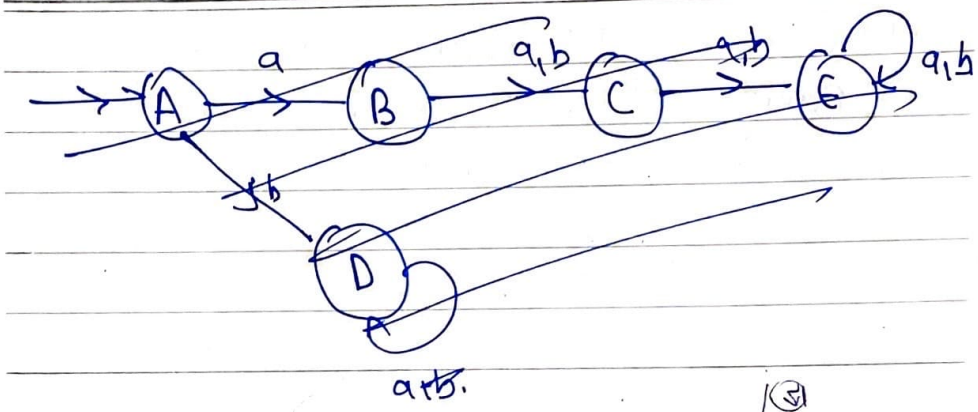


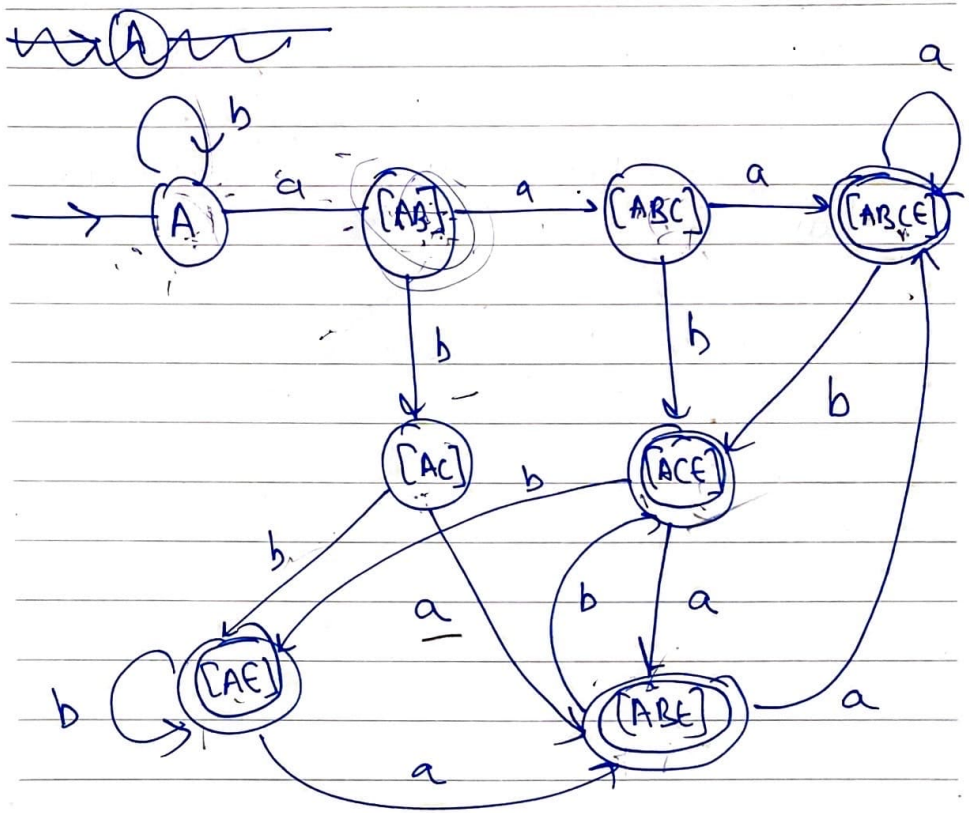
Now convert in DFA

machine ko last ki 3 symbols yaad rakhne padenge 2^3 = 8 states

	a	b
A.	{A,B}	A
B	C	C
C	E	C
E	E	C







Total 8 states ~~so~~ will be in a DFA, because machine ko last 3 digits learn karne padenge

$2^3 = 8$

8 states

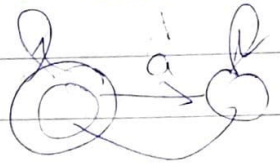
also as in NFA ε was the final state

So in DFA circle wherever we have ε

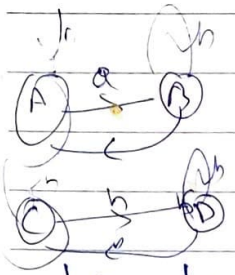
So 4 final states

DFA → NFA
NFA → DFA

* NFA is equal powerful as DFA * kyuki ye dono same chiz compute karte hai



DFA for problems containing AND



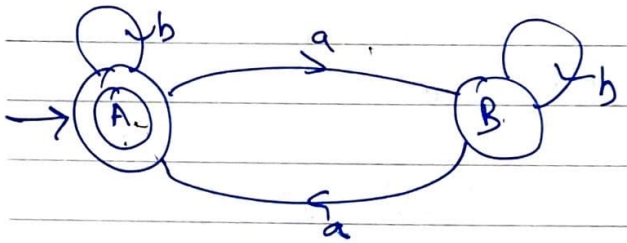
$n_a \bmod 2 = 0$

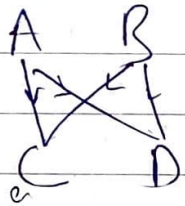
e e → AC

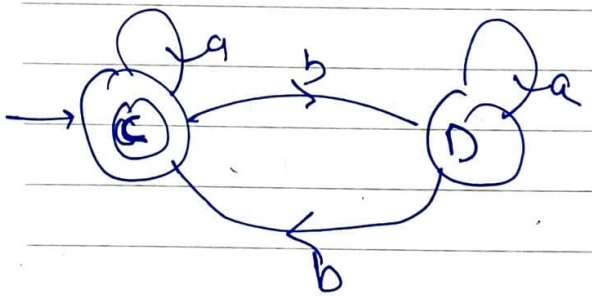
$n_b \bmod 2 = 0$

	a	b
A/C	BC	AD
AD		
BC		
BD		

We have read methods







AC
AD
BC
BD

$Q_1 = \{A, B\}$

$Q_2 = \{C, D\}$

$Q_1 \times Q_2$

$\{A, B\} \times \{C, D\}$

Cartesian Product :-

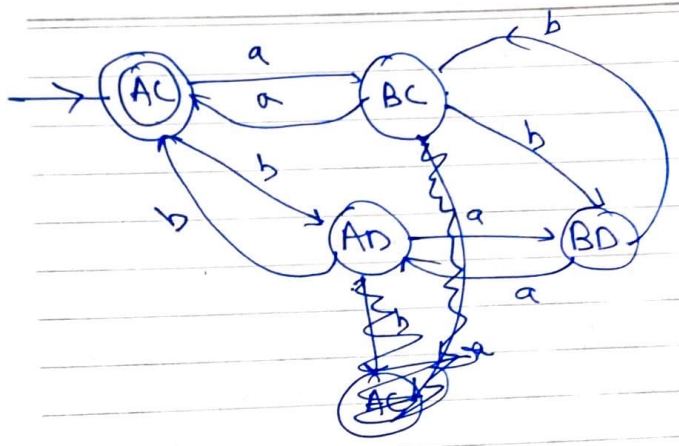
~~AE~~

	a	b
AC	BC	AD
AD	BD	AC
BC	AC	BD
BD	AD	BC

first data — second data

AC will be final state

AC (first data, second data) & AC will be initial state



Total no of states

Na mod 2 = 0

Nb mod 2 = 0

2 X 2

4

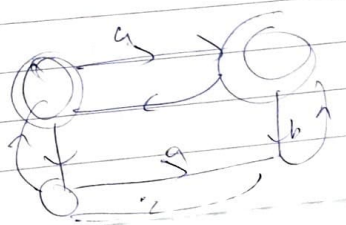
Q. Find no. of states

Na mod 4 = 0 → 4 states

Nb mod 5 = 0 → 5 states

valid type

so 4 X 5 = 20 states



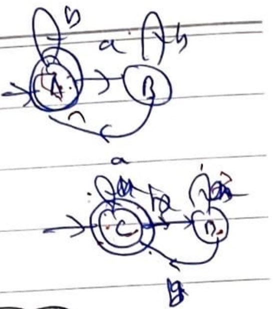
WONDER CEMENT

Date :

Q1 ~~Ata~~ $n_a \bmod 2 = 0$

OR

$n_b \bmod 2 = 0$



(AC)

(BC)

BD

(AD)

A & C were the 2 final states. To jaha jaha aya C hai le the ans

(A & C)

Jisme A & C me se koi bhi hoga will be final state

OR me sirf final state me change atta hai.

and → AC ← fn

OR A/C → f

WONDER CEMENT

Date :

and → ∩

Or → ∪

L^c

$L_1 - L_2$.

Q2 $\Sigma = \{a, b, c, d\}$

Count the states

$n_a(w) \bmod 2$ =] $2 \times 4 = 8$

0, 1

$n_b(w) \bmod 4$] $8 \times 5 = 40$

0, 1, 2, 3

$n_c(w) \bmod 5$

0, 1, 2, 3, 4

$n_d(w) \bmod 1$

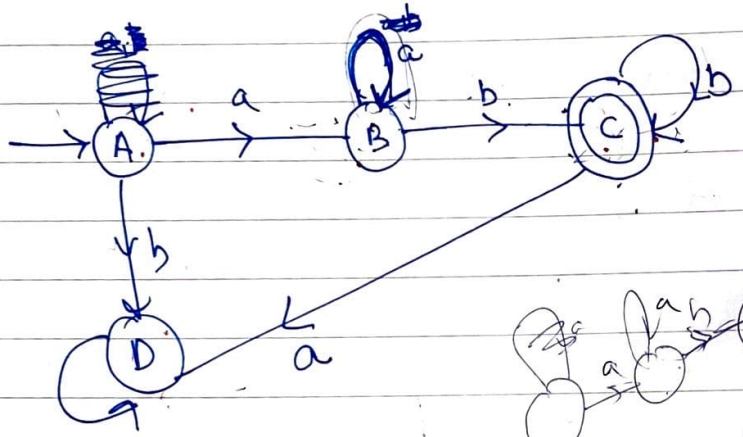
0

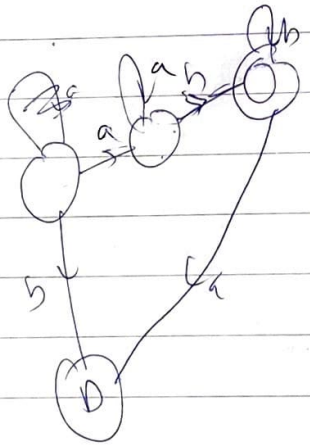
$40 \times 1 = 40$

Q. Construct a DFA that will accept strings $L = \{ a^n b^m \mid n \geq 1, m \geq 1 \}$ where no of a's is followed by any no of b's.

$L = \{ ab, aab, abb, abbb, aaab \}$

aba ✗

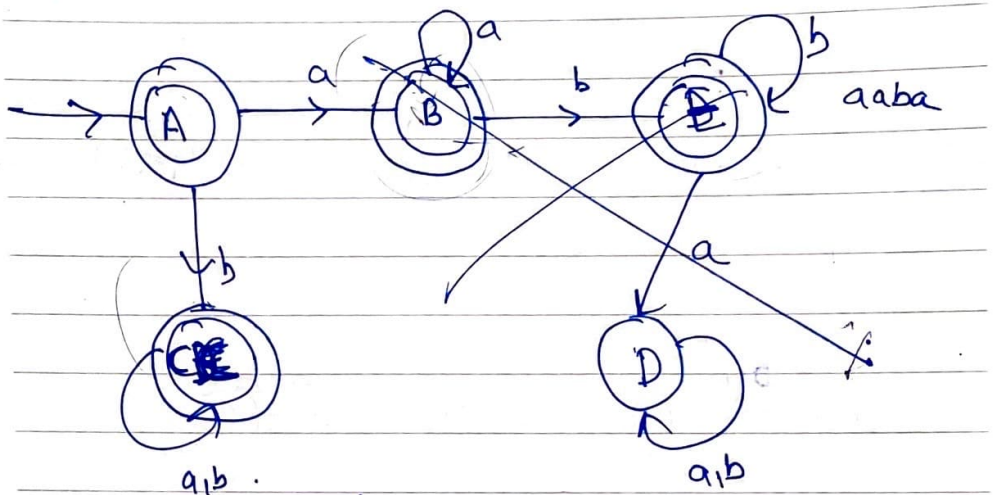


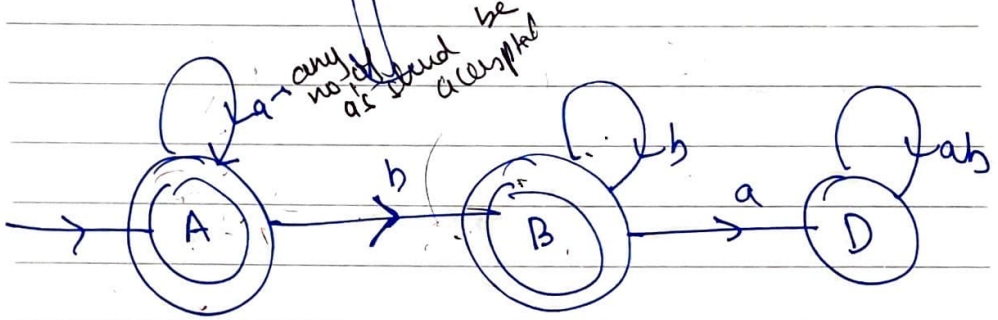


abb

Q. Construct DFA $L = \{ a^n b^m \mid n \geq 0, m \geq 0 \}$

$L = \{ \varepsilon, a, b, ab, aaab, abbb \cdots \}$



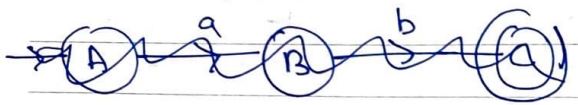


Date :

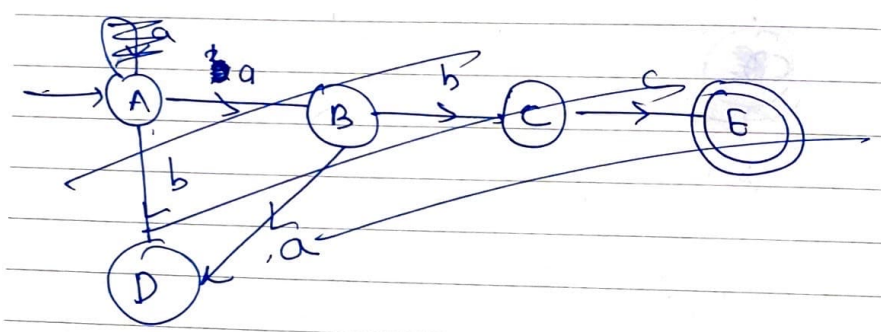
Q→ construct DFA for

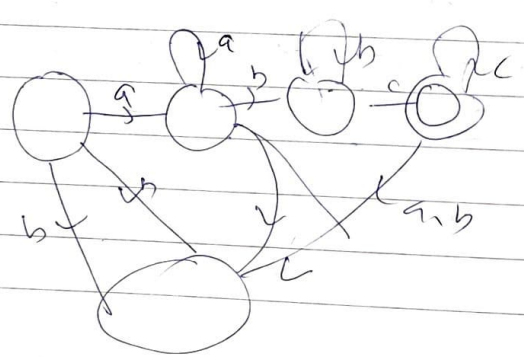
$L = \{ a^n b^m c^l \mid n, m, l \geq 1 \}$

L:- {abc, abbc, aabbcc - ——

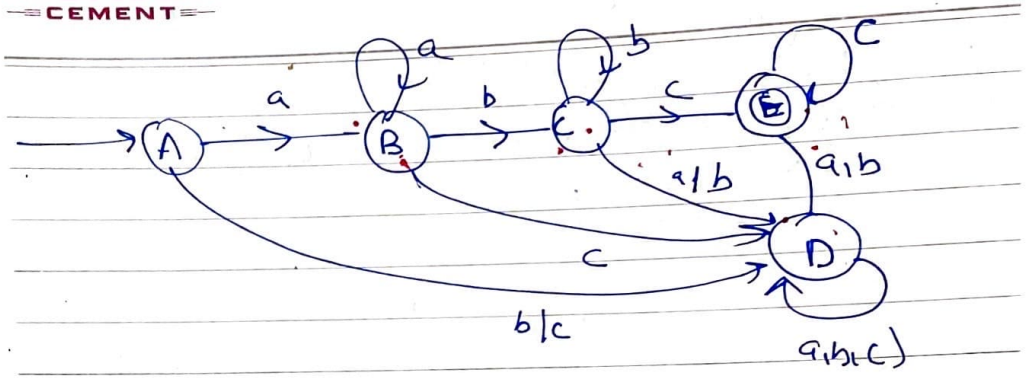


5 ⇒ 4+1 Dead State





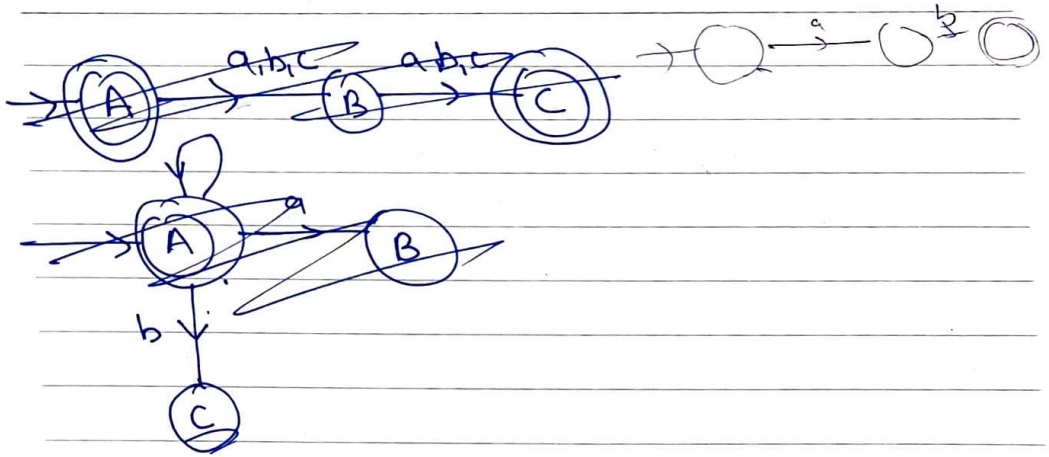
Date :

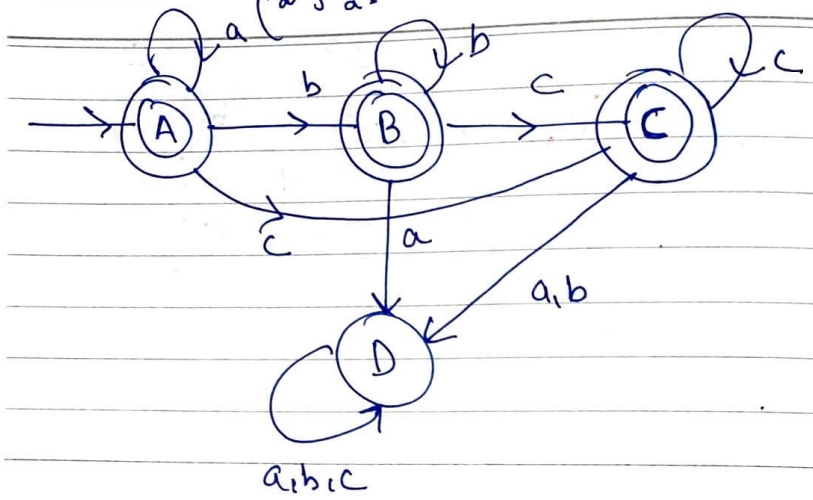


Q→ construct DFA

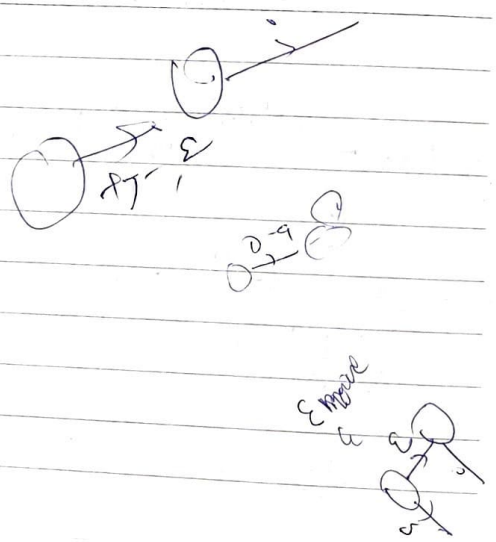
$L = \{ a^n b^m c^l \mid n, m, l \geq 0 \}$

L: {ε, ab, bc, ac, aabbcc, ——





⇒ 4 states

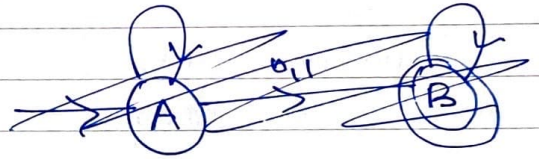


Epsilon NFA

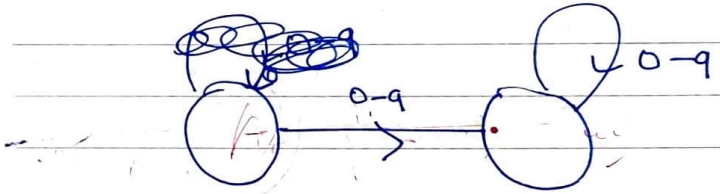
ε → move
or ε → transition

we can move from one state to another state without consuming any input

Q→ construct a ~~DFA~~ FA that accept all valid ~~integer~~ constants



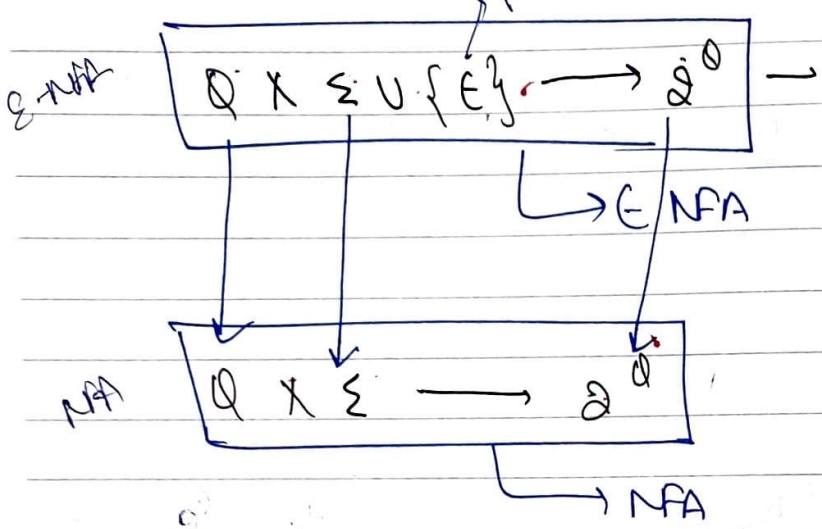
0-9 kyuki kisi se kisi ek state pe jaa sakte hain



+123
−123
→ in dono me dead

123

do we need to modify it



DFA → NFA → ε-NFA

DFA ⊆ NFA ⊆ ε-NFA

DFA ≡ NFA ≡ ε-NFA

Q X Σ → Q

* Every NFA is ε NFA. & every ε NFA can be converted into NFA.

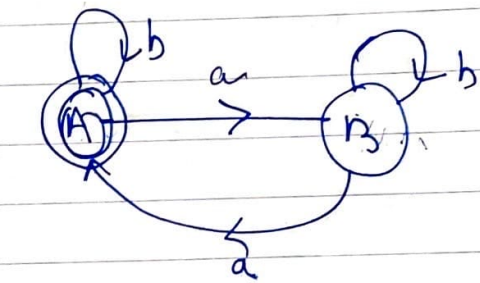
* All NFA, DFA, ε-NFA are equally powerfull.

Minimisation of DFA

[Minimisation is to used minimum no. of states]

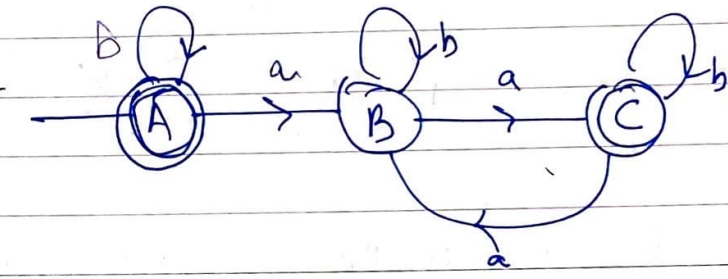
$n_a(w) \bmod 2 = 0$

I



less no of states req

II



more no of states required

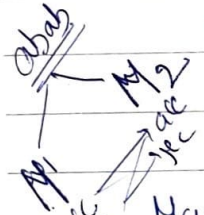
Both are doing the same task but I is more efficient & which requires less resources.

⇒ Minimised DFA is always unique in nature

⇒ We will reduce the no. of states.

EQUIVALENT STATES

Machine I ○ w_1 (string I) ⇒ accept
⇒ reject



Machine II ○ w_1 (string I) ⇒ also accepts
⇒ also rejects

If inside a machine, 2 states are there & let suppose same string w_1 is given on both, then both should perform same task either accept / reject

Me aur gf

Mi bhai nhi

$\delta(q_1, w) \in F \Rightarrow \delta(q_2, w) \in F$

state, string — state, string

$\delta^*(q_1, w) \in F \Rightarrow \delta(q_2, w) \in F$

Koi state q_1 par final par string w state milke

To state q_2 par string w se final milni chaye.

$\delta(q_1, w) \notin F \Rightarrow \delta(q_2, w) \notin F$

If a state q_1 rejects string w then q_2 must also reject

reading on final state could be any

matlab kisi bhi final state par lejae.

but for this we need to check for all strings which is infinite

Better Method

⇓

K - Equivalent states

Two states q_1 & q_2 are said to be k equivalent if

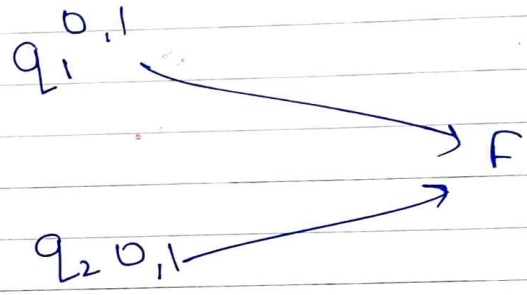
(1) $\delta(q_1, w) \in F \Rightarrow \delta(q_2, w) \in F$

(2) $\delta(q_1, w) \notin F \Rightarrow \delta(q_2, w) \notin F$

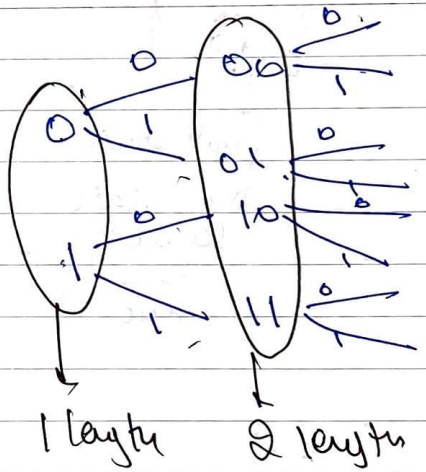
for all w (strings) whose length $\leq k$.

$|w| \leq k$

final states → show acceptance
non final states → " rejection



1 length equivalent



WONDER CEMENT

Date : .....

Minimise the fol. DFA ⇒ equival

Ques

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

⇒ here we have 5 states

0 - equivalent

$\{\{q_0, q_1, q_2, q_3\}\ \{q_4\}\}$

q_0, q_1, q_2, q_3 → non final yaha honge to rejection hoga

q_4 → final state this state cannot be removed

WONDER CEMENT

Date .....

⇒ 2 equivalent k lie equivalent ka I/P dekho & 2 eq. k lie 2 eq. ka I/P

1 equivalence → ek length ki strings k lie behaviour dekhna hai strings ka.

	0	1
q_0	q_1	q_2
q_1	q_1	q_3

→ both are present in set I as both are in same set which is written at equivalent set

same states or we can say done set I me se hai yo ki khud equivalent hai

$\{\{q_0, q_1\}\}$ $\{q_4\}$

	0	1
q_1	q_1	q_3
q_2	q_1	q_2

Same set

	0	1
q_2	q_1	q_2
q_3	q_1	q_4

Same set

But not in same set

do q_3 1 equivalent nahi hoga.

do 1 equivalent

$= \{\{q_0, q_1, q_2\}\ \{q_3\}\ \{q_4\}\}$

q_2 check karo ki ise kissi bhi ek k sath club karlo

2 equivalent

q_0 & q_1

	0	1
q_0	q_1	q_2
q_1	q_1	q_3

q_1 equivalent me same set me hai

1 equivalent k set me diff set me hai

Thus means q_0 & q_1 alag alag set me honge.

$\{\{q_0, q_2\}, \{q_1\}, \{q_3\}, \{q_4\}\}$

Now for q_2

q_2 & q_0

do q_0 & q_2 same set me hone

	0	1
q_0	q_1	q_2
q_2	q_1	q_4

same

same behaviour

WONDER CEMENT

Date : .....

3 equivalent.

q_0, q_2

	0	1
q_0	q_1	q_2
q_2	q_1	q_2
	↓	↓
	same set value behaviour	same set value behaviour

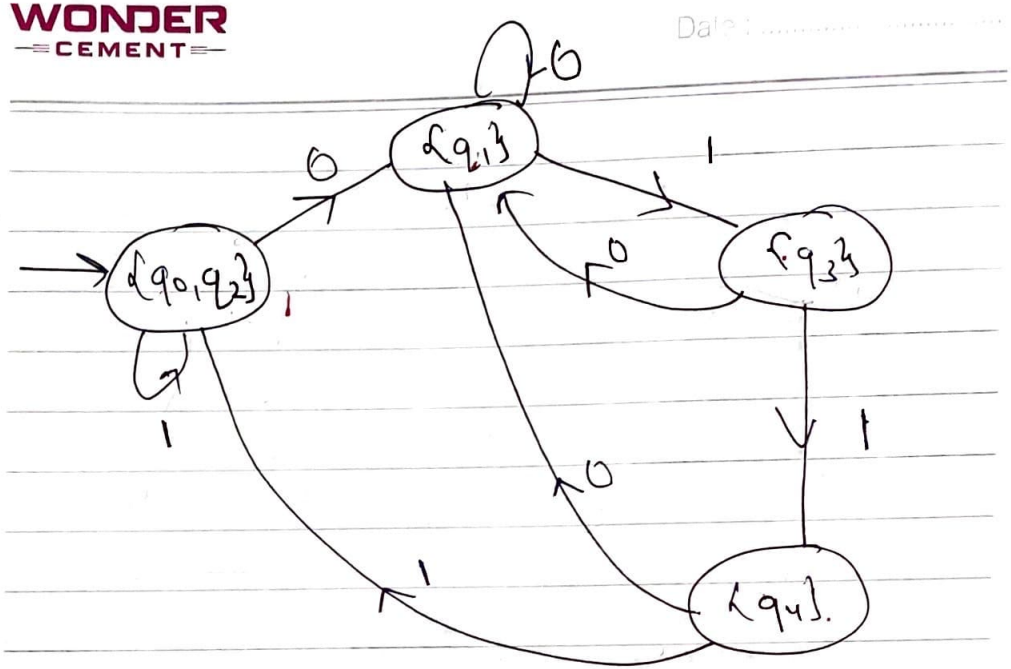
So 3 equivalent is same as 2 eq4. also same same O/P agae so

STOP

means k q_0, q_2 → m dono ka same purpose hai to we can eliminate any one of them, use ek state kam hogi.

WONDER CEMENT

Date : .....



⇒ 4 states

↳ after minimisation we are left with 4 states

Q→ Minimise the DFA given below

To minimise :-

(1) Remove unreachable states (jisse initial point se pahuch nai sakte)

	0	1	
→ q_0	q_7	q_5	initially we have 8 states
q_1	q_6	q_2	
find → q_2	q_0	q_2	
q_3	q_2	q_6	x
q_4	q_7	q_5	
q_5	q_2	q_6 ✓	
q_6	q_5	q_4	
q_7	q_6	q_2	

0 1 5 6 2 7 4 3

(3)

find reachable states from initial

= { $q_0, q_1, q_5, q_6, q_2, q_4, q_7$ }

remove q_3 (not reachable)

0 equivalent

~~q_0 q_1~~

{ { $q_0, q_1, q_4, q_5, q_6, q_7$ } { q_2 } }

1 equivalent

~~{ { q_0, q_4 } { q_1, q_7 } { q_5 } { q_6 } { q_2 } }~~

$q_1 \rightarrow q_5 \rightarrow q_7$

~~q_0 q_4~~

{ q_0, q_4, q_6 }

{ { q_0, q_4, q_6 } { q_1, q_7 } { q_5 } { } { q_2 } }

0 1 5 6 2 4 5 7

2 equivalent

$\{ \{q_2\} \ \{q_5\} \ \{q_0, q_4\} \ \{q_6\} \ \{q_1, q_7\} \}$

same same O/P ⇐ ishe we stop here.

⇓

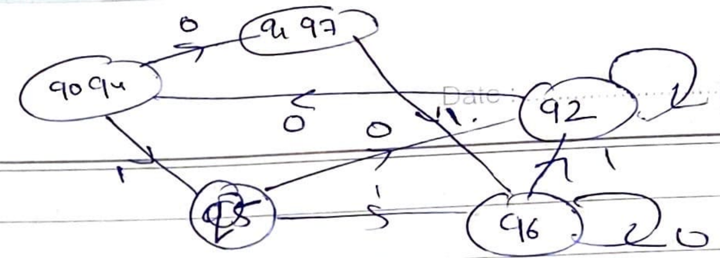
5 states

3 equivalent

$\{ \{q_2\} \ \{q_5\} \ \{q_6\} \ \{q_0, q_4\} \ \{q_1, q_7\} \}$

same as 2 eq.

so stop



Hint

1. first find reachable states from initial, jo unreachable ho to usse hata do.

2. Groupings karo

first q_0 ko sabke sath check karo, jo jo same hai unhe q_0 ke sath warna alag.

3. Jab 3 equ. nikalte hai to 2 eq. ka input lena hai

	0	1
q_0	q_1	q_5
q_1	q_6	q_2

ye dono same set me hone chahiye

$q_0, q_4 \rightarrow$ remove 1

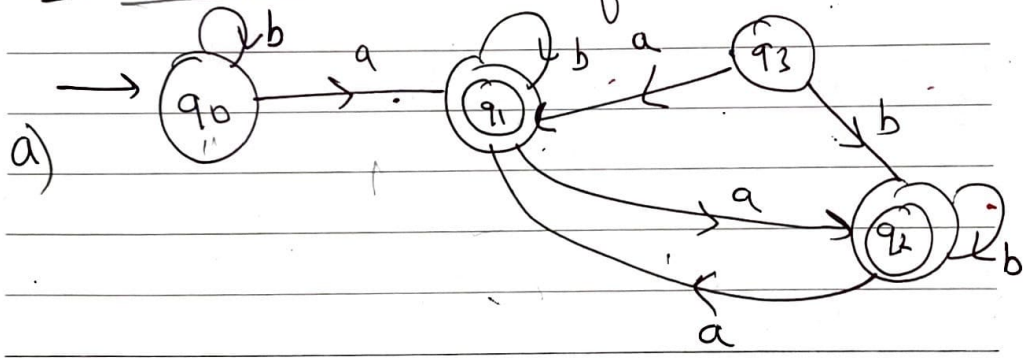
$q_1, q_7 \rightarrow$ remove any 1

$q_3 \rightarrow$ already eliminated

Total 5 dali → after minimisation

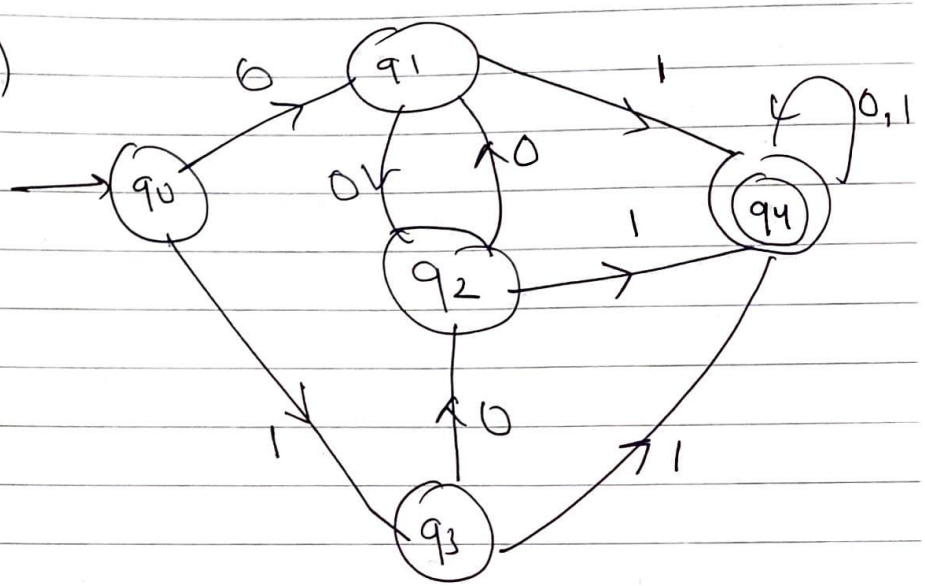
Q Minimise the foll. DFA

a)



M

b)



(a)

	a	b
→ q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_1	q_2
q_3	q_1	q_2

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

remove q_3.

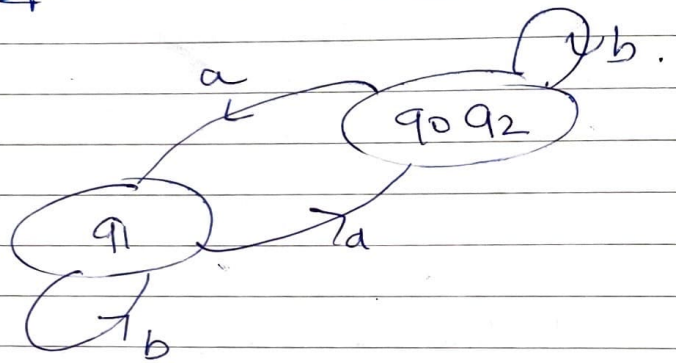
WONDER CEMENT

Date : .....

~~{q0, q1}~~

{q0} {q1, q2}

⇒ {q0} {q1, q2} =



WONDER CEMENT

Date : .....

(b)

	0	1
→ q0	q1	q3
q1	q2	q4
q2	q1	q4
q3	q2	q4
(q4)	q4	q4

q0 = {q0, q1, q3, q2, q4, q1}

Nothing is removed.

0-equivalent

{q0, q1, q2, q3} {q4}

1 equivalent {q0} {q4} {q3, q2, q1}

2 equivalent {{q4} {q0} {q1, q2, q3} ~~{q1} {q2, q3}~~}

→ 3 states

~~2 equivalent~~

{{q4} ~~{~~

Date :

SHORTCUT.

q_4 → final state
↓
sabse alag.

* esi state jo jaldi final state par phuch sakti hai ✓

q_1 q_2 q_3 read 1 char & reach q_4

whereas q_0 take 2 char.

$\{q_1, q_2, q_3\}$
⇓
equivalent

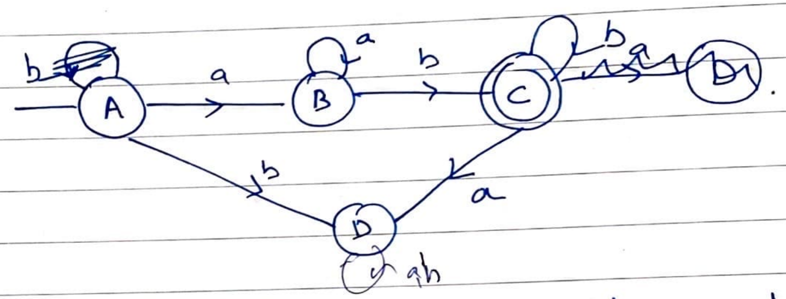
Please NOTE

esi state jo sabse phle final state par phuch rahi hai jese q_1, q_2, q_3 1 character read karke final q_4 par phuch rahe hai, so they are equivalent.

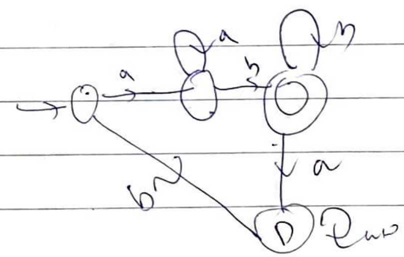
Date : 2/10

Q→ Design a DFA for the lang. $L = \{a^n b^n \mid n \geq 1\}$ → DFA NP

L: {ab, aabb, ~~abb~~ ... aaabbb



But here extra strings like aab, abb .. will also be constructed which can't be covered in lang given

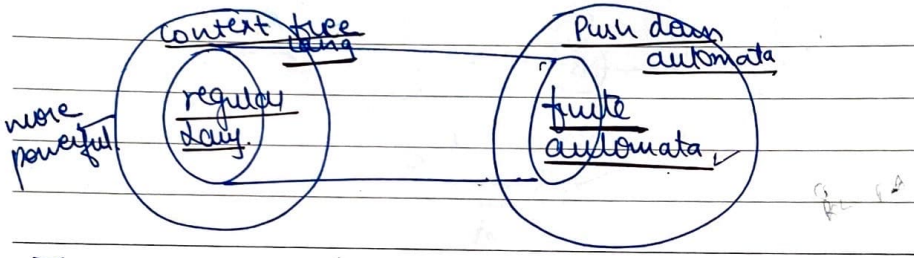


WONDER CEMENT

Date :

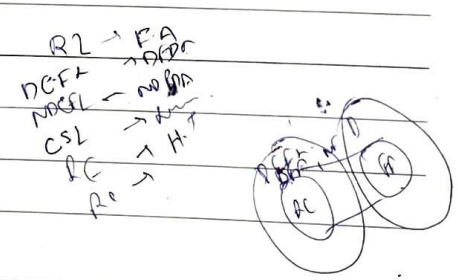
* Regular lang.

lang. for which either a DFA or NFA or ε-NFA can be constructed.



There exist a more powerful set of lang. which regular lang. can't support.

So we need a more powerful machine which support lang. which are more powerful & accept all lang.



WONDER CEMENT

Date :

Language

More Powerful

* regular lang ⟶ finite automata
* Context free lang ⟶ Push down automata
* Context sensitive lang ⟶ Linear bound automata
* Recursively enumerable set ⟶ Turing Machine

family of lang. ⟶ family of machines with min no of resources

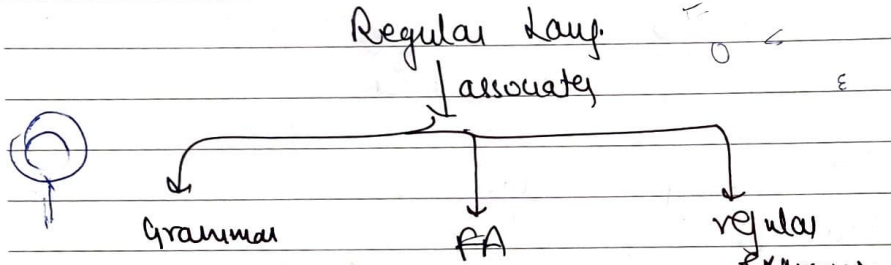
⇒ More powerful lang & machine can accept their lower powerful lang.

(7) $L_2 = \{a^n b^{2n} \mid n \geq 1\}$

No, computation involv (machine ko yaad rakhna padega kitne a aye hai)

(8) $L = \{a^n b^m \mid n \geq 1, m \geq 1\}$

Yes, it can be formed.



Regular Expression

RULE I] ϕ, ϵ, $a \in \Sigma$

ϕ → $\{\}$

ϵ → one string $\{\epsilon\}$

$a \in \Sigma$ → any symbol that belong to given alphabet

represents primitive regular expression / basic regular exp.

* Every regular expression represent some set

→ representing the sets.

ϵ → that represents only one string $\{\epsilon\}$

ϕ → empty set / null set.

RULE 2] If r_1 & r_2 represents primitive regular expression then

add ← $r_1 + r_2$

concatenate ← $r_1 \cdot r_2$

closure ← r_1^* (repetition of r_1 any no of times)

bracket ← (r_1)

⇒ all represents regular exp.

$L(r_1) \cup L(r_2)$

$L(r_1) \cdot L(r_2)$

to remove ambiguity brackets are used.

WONDER CEMENT

Date :

RULE 3 We can apply Rule 1 & Rule 2 in any combination & any no. of times.

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

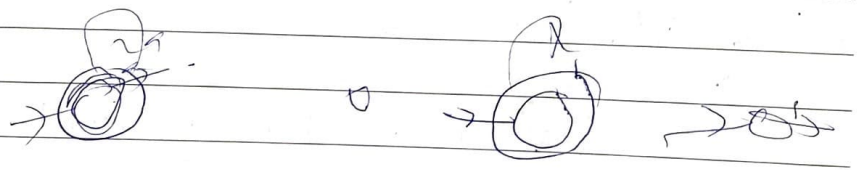
Rule 1 → $\phi, 0, 1, \epsilon$

Rule 2 → $\epsilon + 0$, $0 + 1$, 0.1 } regular exp.

Rule 3 → $\{(\epsilon + 0) \cdot (0 + 1)\}^*$

$((\epsilon + 0) \cdot (0 + 1))^*$

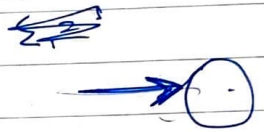
$\phi, 1^*$



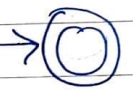
Date :

finite automata of regular expressions

ϕ →



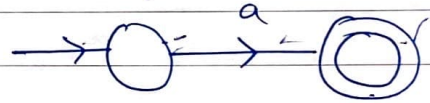
ϵ →

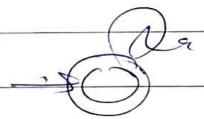


str ek string accept hoga change

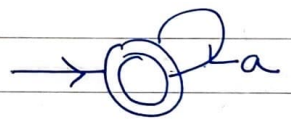
$\Sigma = \{a, b\}$

a →





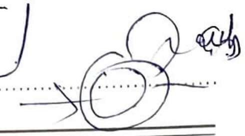
a^* → $\{\epsilon, a, aa, aaa \ldots\}$



WONDER CEMENT

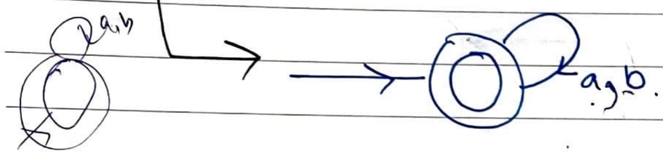
RE → NFA → DFA

Date :



sari strings

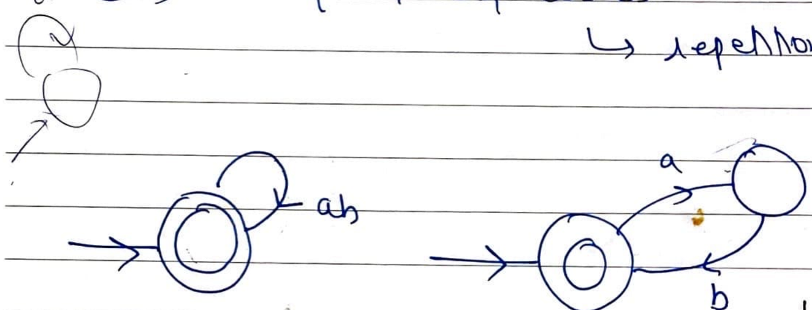
$\Sigma^{+} = (a+b)^{*} = \{ \epsilon, a, b, ab, \ldots$



(NFA) ya a ya b ya kuch nahi

$(ab)^{*} = \{ \epsilon, ab, abab \ldots$

↳ repetition of ab



(NFA)

* Regular expression corresponds to NFA not DFA

(a+b)(a+b)

WONDER CEMENT

Date :

Q → Find R.E for Lang. L that will accept $\{ w \mid |w| = 2 \}$

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

aa

L = { ab, ba, aa, bb } → DFA, NFA, R.E

ab → regular set corresponds ab

b.a + a.a + b.b

(a+b)(a+b)

{ab} → ab +

ab + b.a + a.a + b.b

~~Common lete time~~ ~~Order~~ ka khyal rakho

~~a+b~~

~~aabbaab~~

a(

aa ab ba bb

a(a+b) b(a+b)

aa + ab + ba + bb

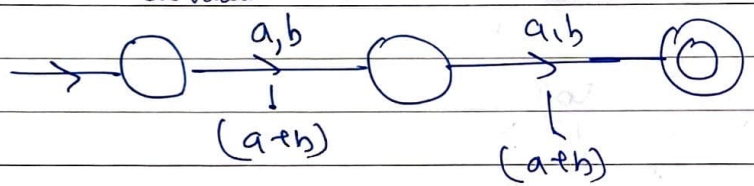
Common lete hue
order check karo

a·(a+b) + b(a+b)

$\boxed{(a+b)\ (a+b)}$

1st me se koi ek variable 2nd me se koi ek variable

OR



$\boxed{(a+b)\cdot(a+b)}$

(a+b) · (a+b)

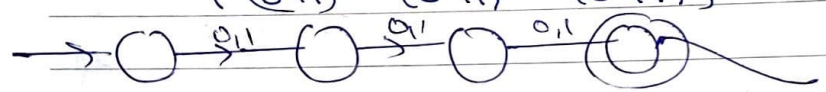
Qs $L_2 = \{ w \mid |w| = 3 \}$

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

{000, 001, 010, 011, 100, 101, 110, 111}

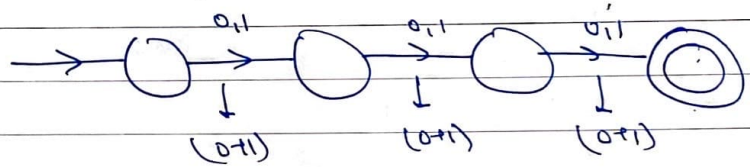
Take common & we get

$(0+1)\cdot(0+1)\cdot(0+1)$



Qs $L_2 = \{ w \mid |w| \geq 3 \}$

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



$(0+1)(0+1)(0+1)(0+1)^*$

$|w| = 3, 4, 5, 6 \ldots$

$(0+1)(0+1)(0+1) + (0+1)(0+1)(0+1)(0+1) \ldots$

$(0+1)^3 + (0+1)^4 \ldots$

$(0+1)^3 [\ \varepsilon, 0+1, (0+1)^2, (0+1)^3\ ]$

$(0+1)^*$

Date : ....

Qs $|w| \bmod 2 = 0$, $\Sigma\{a, b\}$

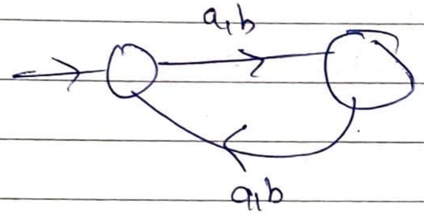
0, 2, 4, 6, 8, ...

$((a+b)(a+b))$

$((a+b)(a+b))^*$ → repeat

exactly 2 length wala

$*$ → optional, agar kuch bhi select nai karoge to ϵ



Qs $|w| \bmod 2 = 1$

1, 3, 5 ...

$(a+b)((a+b)(a+b))^*$

1 + even = odd

Qs $|w| \bmod 2 = 2$

jab hum mod 2 karenge sirf humesha 0, 1 ayega. uske alawa nai aa sakta

ϕ → regular expression

no string will be accepted

as mod 2 ≠ 2

Date : ....

$b^* a b^* a b^*$

Qs Write regular expression for lang. that accept all w's

$n_a(w) = 2$. (here no of a's should be 2)

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

{aa, aab, baa, ...}

$b^* a b^* a b^*$

b^* a b^* a b^*

a, a — ye fixed hai

Qs atleast 2 a : $n_a(w) \geq 2$

{aa,

$b^* a b^* a (a+b)^*$ or

$(a+b)^* a (a+b)^* a (a+b)^*$

ye 2 a to padna ayenge

⇒ kuch bhi matlab $(a+b)^*$

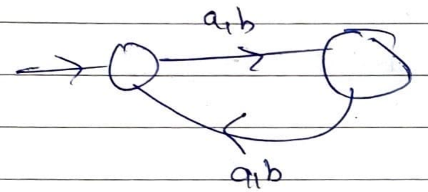
usse phle, bich & baad me kuch bhi aye.

Date : ....

Qs $|w| \bmod 2 = 0$, $\Sigma\{a,b\}$

$0, 2, 4, 6, 8, \ldots$

$((a+b)(a+b))$



$((a+b)(a+b))^*$ — repeat

exactly 2 length wali

$*$ → optional, agar kuch bhi select nai karoge to ϵ

Qs $|w| \bmod 2 = 1$

$1, 3, 5 \ldots$

$(a+b)((a+b)(a+b))^*$

1 + even = odd

Qs $|w| \bmod 2 = 2$

jab hum mod 2 karenge sirf humesh 0, 1 ayega. uske alawa hai ajakta

ϕ → regular expren

no string will be accepted as $\bmod 2 \neq 2$

Date

$b^* a b^* a b^*$

Qs Write regular expression for lang. that accept all w's $n_a(w) = 2$. (here no of a's should be 2)

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

$\{aa, aab, baa, \ldots$

$b^* \; a \; b^* \; a \; b^*$

ye fixed hai

Qs atleast 2 a: $n_a(w) \geq 2$

$\{aa,$

$b^* a b^* a (a+b)^*$ or

$(a+b)^* \, a \, (a+b)^* \, a \, (a+b)^*$

ye 2 a to padna ayenge

→ kuch bhi matlab $(a+b)^*$

usse phle, bich & baad me kuch bhi aye.

WONDER CEMENT

Date :

⇒ 2 regular expression me minimize
Nahi hota, there is a
concept of equivalent R.E

Q) Write R.E that will accept
all w's such that

$$L = \{ w \mid n_a(w) \bmod 3 = 0 \}$$

0, 3, 6, 9, 12 ---.

0 → null.
3 → smallest

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

~~$b^* a b^* a b^* a (a+b)^*$~~

$(b^* a b^* a b^* a b^*)^*$ $+ b^*$

or

$b^* (b^* a b^* a b^* a b^*)^*$

after a
ho hi na.

WONDER CEMENT

$(b^* a b^* a b^* a b^*)^2$

$b^* a b^* a b^* a b^* b^* a b^* a b^* a b^* ab^*$

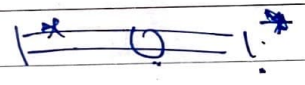
jo bracket me term hai
use hum ϵ nahi generate
kar sakte.

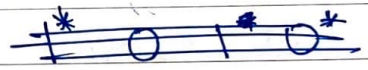
Q) $L = \{ w \mid w$ does not contain
00 as substring$\}$

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

{ ~~01, 10, 101,~~ 010,

010





Q- Write RE for the lang. that will accept all w's such that

atmost

$L = \{ w \mid Na(w) \leq 2 \}$

$\Sigma = \{a, b\}$ 0, 1, 2

{ε, b, bb, bbb - - - - - - ∼

ab, abb, abbb - - -

ba, bab, bba - - -

aab, aabb, aabbb - - -

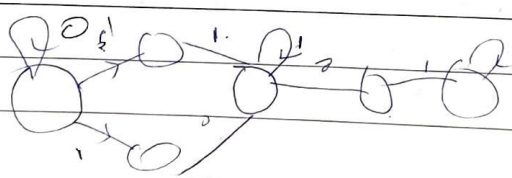
baa, aba - -

~~basic bb, ab.~~

$(b+a)^* a^* (a+b)^*$

a's 0 OR 1 OR 2

$b^* + b^* a b^* + b^* a b^* a b^*$

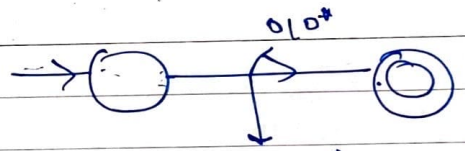


To show if R.E is powerful ?
draw RE for corresponding DFA
& DFA for corresponding RE

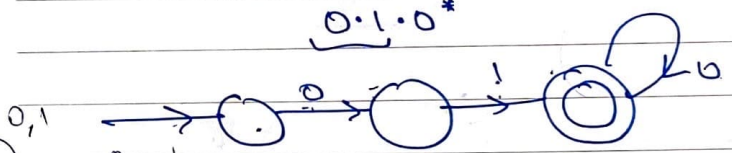
Regular expression TO FA

initially NFA then DFA

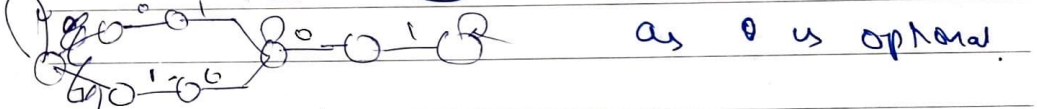
01.0*



0.1.0*



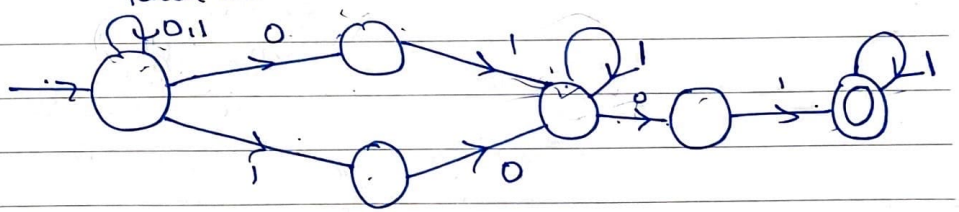
as 0 is optional.



Q→ $(0+1)^*$ (01 + 10) 1* 011*

OR

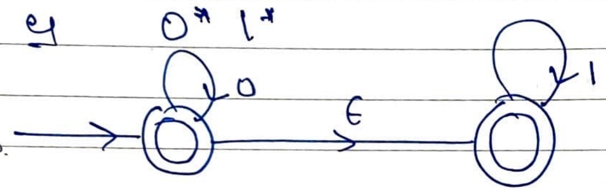
Kuch bhi



* If there is no mandatory path then we take null move

Kabhi bhi mandatory part ko loop me mat dalo.

eg $0^* 1^*$



Identities for regular expression

① $\phi + r = r = r + \phi$

(ϕ → no string; + → OR; r → lang with many strings)

② $\epsilon^* = \epsilon$ { epsilon ko kitni bhi bar likh lo length nahi badti }

③ $(r^*)^* = r^*$

④ $\epsilon + r \cdot r^* = r^*$

$\{r\} \cdot \{\epsilon, r, rr, rrrr \ldots\}$

$\epsilon \cup \{r\epsilon, rr, rrr, \ldots\}$

$= \{\epsilon, r, rr, rrr, rrrr \ldots\} = r^*$

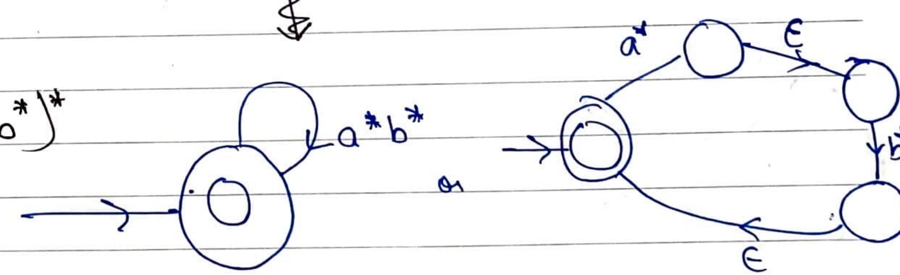
⑤ $\epsilon \cdot r = r = r \cdot \epsilon$

⑥ $(a+b)^* = (a^* b^*)^* = (a^* + b)^* = (a + b^*)^* = a^* (ba^*)^* = b^* (ab^*)^*$

NFA

$(a+b)^* = (a^* b^*)^*$

$(a^* b^*)^*$

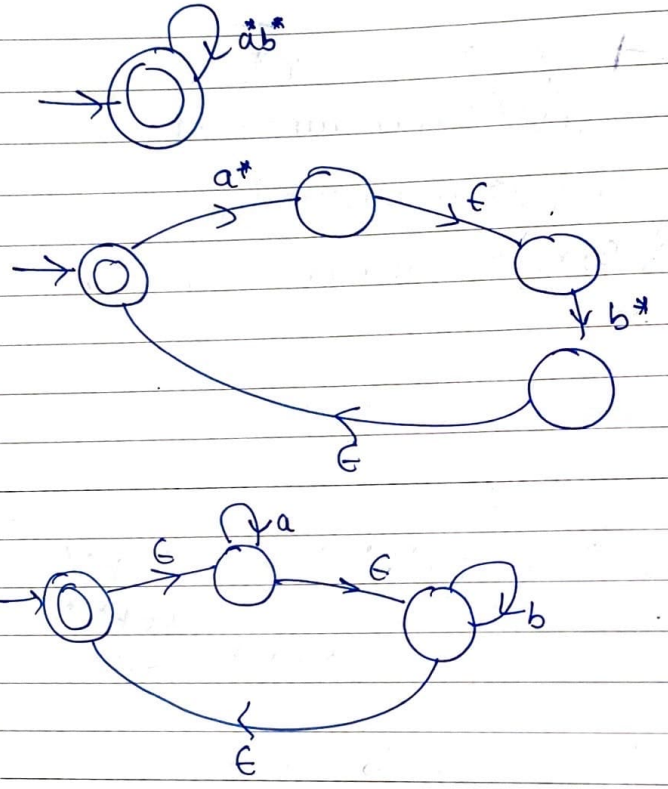


WONDER CEMENT

Date : .....

DFA

$(a^*b^*)^*$



DFA →
NFA → DFA
ENFA → NFA
RE → NFA

WONDER CEMENT

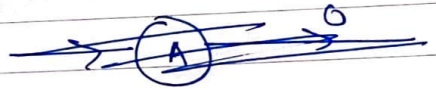
Date :

Given DFA ⟶ RE (find)

3 Methods :-

1. Arden's Method
2. * State elimination Method.
3. Warshall's algorithm.

Ardeis Method



If there is a recursive expression

$r = p + rq$ then its solution is given by $r = pq^*$

$r = p + rq$
$r = pq^*$

A.

$r = p^*rq$
rpq^*

$r = p + rq$
$r = pq^*$

Date : Volvo / Basket Red Nell to Palm Volvo

Grammar

Sent. → <Noun> <is/am/are> <V'> <ing> <to/by> <object>

⇒ Ram is going to Agra'

grammar is just a set of Rules.

→ Sita is ______

here Sita & Ram are not replaced ek dusre se, they are replaced by Noun

✓ Variable → That can be replaced
✓ Terminal → canot " "

Grammar 'G' is defined as 4 tuple

⇓ set of rules to validate a sentence

⇒ (V , T , P , S)

- V → set of symbols ie, variable symbols (can be replaced)
- T → set of symbols ie terminal symbols (cannot be replaced)
- P → set of rules called production rules
- S → special symbol ie start symbol.

Consider the grammar

S →	aSB
S →	aB
S →	b
B →	b

} ⇒ P :- Production rules.

Production rules
↳ means LHS can be replaced by RHS.

WONDER CEMENT

Date : .....

So LHS contains variables

non terminals | Variables → S, B
Terminal → a, b

Why need generator ?
to generate all strings from alphabet

eg S → a(SB)

↓

Then 2 can again be replaced by their corresponding LHS.

S → aSB
a(aSB)B
aabbB
aabbb

terminal variables ⟸ aabbb → now nothing can be replaced

WONDER CEMENT

Date

So we say

aabbb ∈ L(G)

This string belongs to language of grammer or generator

Ram is going Agra to
eg {a-z, A-Z, (space), ,, ., } → alphabet
do it is a string

eg (a-z, A-z)
then do it is a collection of string

& grammer is set of rules & in TOC we say Production Rules

Derivation

(can't be replaced)

The process of deriving a terminal string using production rules given in the grammar is called a **derivation**

⇓

string ko derive karna

Leftmost derivation ~~& Rightmost~~

During the process of deriving a string (terminal) from production rules we replace exactly one variable by its corresponding right hand side in each step. & If in every step we always replace the leftmost variable then the corresponding derivation is called as leftmost derivation

$S \rightarrow aSB$ (S can be replaced)

↳ can replace S & B but Leftmost is S so replace S

If S is replaced → left derivation

If B is replaced → right derivation

Rightmost derivation

If every step we replace the rightmost variable by its corresponding right hand side then the derivation is said to be rightmost derivation.

Application:- ~~left~~ leftmost → Top down parser

Rightmost → Bottom up parser (in reverse)

Sentential form | Sequental form

In the process of deivation all intermediate strings are said to be sentential / sequental form string

$S \rightarrow aSB$

Intermediate strings ∈
- $a(aSB)B$
- $aaSBB$
- $aabBB$
- $aabbB$

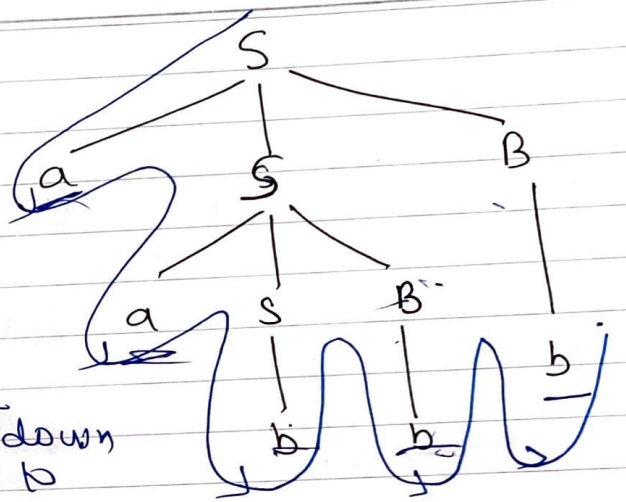
$\Rightarrow$ sentential / sequental

↓

$aabbb$ → final

Parse Tree | Derivation Tree

Another way to represent how a string is derived from a given set of production rules.



traverse top to down & left to right

representation of a string

↓

aabbb

This is also called as yield of the parse tree

Parse Tree
↓
representation of string
↓
Yield of Parse tree.

WONDER CEMENT

Date : .....

Let's construct some Grammar

Q→ Construct grammar for the language

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

$a(a+b) + b(a+b)$

$S \to aa$
$S \to ab$
$S \to ba$
$S \to bb$

This will generate all the strings

OR

$S \to aa \mid ab \mid ba \mid bb$

$|w| = 2$ (length 2)

R.E $(a+b)(a+b)$

WONDER CEMENT

Date : .....

Basic element $(a+b)$

$A \to a \mid b$

Now $(a+b)(a+b)$

$S \to A \cdot A$

$S \to aS' \mid bS'$
$S' \to a \mid b$

Corresponding Grammar

$S \to AA$
$A \to a \mid b$

Q→1 Construct grammar

~~$L = \{ w \mid |w| \geq 2 \}$~~

~~$L = \{ aa, ab \}$~~

$L = \{ a^n \mid n \geq 1 \}$

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

$Rt \to a.a^*$

$A \to a$
$B \to a^*$

$S \to AA^*$

$S \to AB^*$
$A \to a$
$B \to a^*$

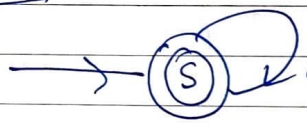
Q-2 $L = (a+b)^*$

$= \{\epsilon, a, b, \ldots\}$

$S \to aS$
$S \to bS$
$S \to \epsilon$

$S \to aS \mid bS \mid \epsilon$

NFA



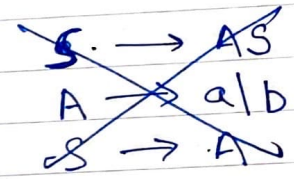
$S \to aS$
$S \to bS$
$S \to \epsilon$ (as initial & final are same)

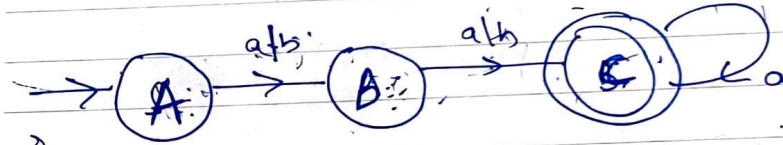
⇒ agar koi bhi grammar Regular likhni hai na samaj aaha ho to NFA banalo.

(a3)

Q-3 $L = \{w \mid |w| \geq 2\}$

$(a+b)(a+b)(a+b)^*$





$A \to aB \mid bB$
$B \to aC \mid bC$
$C \to aC \mid bC \mid \epsilon$

$S \to AAB$
$A \to a \mid b$
$B \to aB \mid bB \mid \epsilon$

Or

$S \to AS \mid AA$
$A \to a \mid b$

WONDER CEMENT

base cond : a.

Date :

Q) Construct a grammar $L = \{a^n \mid n \geq 1\}$

$\{a, aa, aaa \ldots\}$

$S \rightarrow aS \mid a$ ⇒ base cond^n.

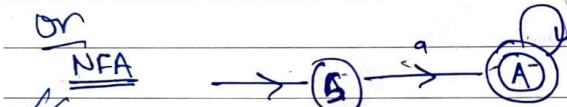
↓

terminate at terminal cond^n. (base)

```
A()
{
  a.
  A()
}
```

$S \rightarrow AS$

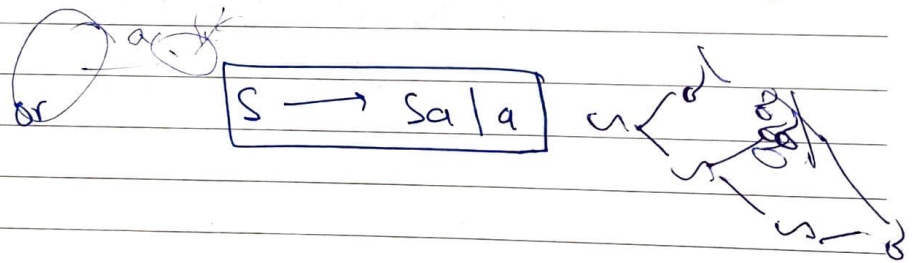
$A \rightarrow a.$



$S \rightarrow aA$

$A \rightarrow aA$ → (leading a's) continue for a's

$A \rightarrow \epsilon$ → stop as it's a final state.



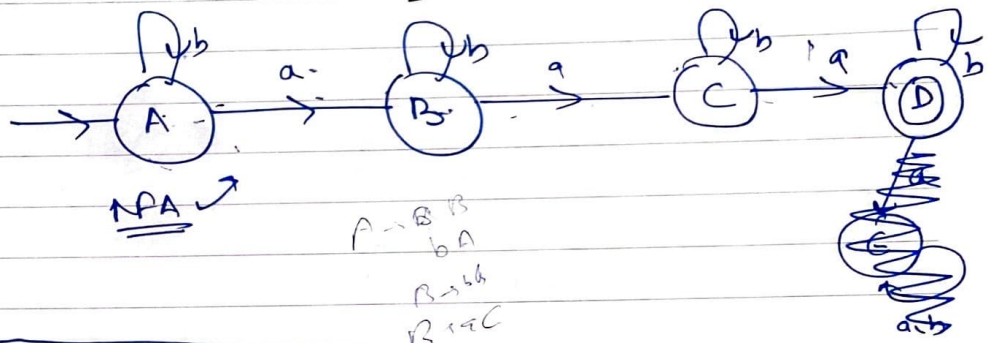
WONDER CEMENT

Date :

Qn) Construct grammar for the lang

$L = \{w \mid n_a(w) = 3\}$

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



$A \rightarrow aB \mid bA$

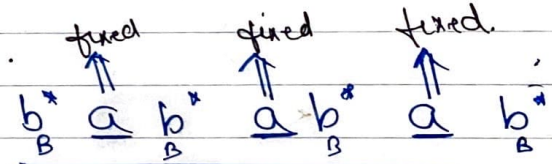
$B \rightarrow aC \mid bB$

$C \rightarrow aD \mid bC$

$D \rightarrow bD$

OR

RG



$S \rightarrow BaBaBaB$

$B \rightarrow bB \mid \epsilon$

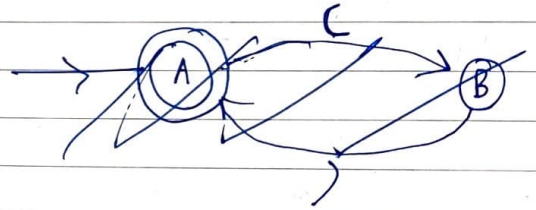
Q Generate grammar for balanced Parenthesis.

((())) ←

() () () ←

(()) ( ) (())

$\Sigma = \{ (,) \}$



( )

This is infinite so NFA & DFA don't exist

(( ))

$S \to (S) \mid \epsilon$

$S \to \epsilon$ (no bracket)

S( )

(S)

$S \to (S) \mid \epsilon$

Isse hume ((( ))) ese brackets nhi jaenge

$S \to (S) \mid \epsilon \mid SS$

$S \to (S)$

SS

HW

Q. Grammar for balanced if else?

hint :- single if ho sakta hai

Ques ① Construct grammar for the lang
$L = \{a^n b^m \mid n \geq 1, m \geq 1\}$

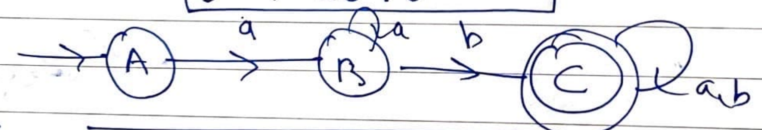
② $L = \{a^n b^n \mid n \geq 1\}$

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

$S \to AB$
$A \to aA \mid a$
$B \to bB \mid a$

ab, aab, aaab, abb

$S \to aS \mid [redacted] \mid \epsilon$
$S \to ab \mid b$



Or

$A \to aB$
$B \to bC \mid aB$
$C \to aC \mid bC \mid \epsilon$

② $L = \{a^n b^n \mid n \geq 1\}$

{ab, aabb, aaabbb.

~~$S \to Abb$~~
~~$A \to aA$~~

~~$S \to Ab$~~

$S \to aSb$
$S \to ab$

---

① $a^n b^m$ → inke bare me alag alag socheng.
A B

$S \to AB$
$A \to aA \mid a$ (generates a)
$B \to bB \mid b$ (generates b)
base case.

② ~~$a^n b^n$ jitne a utne b~~
~~$S \to ab$ first step~~

(2) $a^n b^n$ → like base me alag alag nahi socha

$S \rightarrow ab$

aabb

aaabbb

aaaabbbb

$S \rightarrow asb \mid ab$

Q. construct the grammar for the lang..

$L = \{ a^n b^{2n} \mid n \geq 1 \}$

$= a^n b^{2n}$

$S \rightarrow asbb \mid abb$

aabbbb

$S \rightarrow asbb \mid abb$

aasbsbb | abb

$S \rightarrow asbb \mid abb$

$S \rightarrow as'bb$

Q. $L_2 = \{ a^n b^m c^n \mid n, m \geq 1 \}$

$S \rightarrow aBc$

aabbcc

$S \rightarrow aSc \mid aAc$

$B \rightarrow bA \mid b$

$S \rightarrow ABC$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

$C \rightarrow cC \mid c$

Q. $L = \{ a^n b^m c^m d^n \mid n, m \geq 1 \}$

A B C D

$S \rightarrow ABCD$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

$C \rightarrow cC \mid c$

$D \rightarrow dD \mid d$

$S \rightarrow AB$

$A \rightarrow aAb \mid ab$

$B \rightarrow cBd \mid cd$

$S \rightarrow AB$

$A \rightarrow a$

$S \rightarrow AB$

$A \rightarrow aAb \mid ab$

$B \rightarrow cBd \mid cd$

WONDER CEMENT

Date : .....

Q. ~~Generate Grammar for tag~~

Grammar → Minimised DFA

$S \to Xa \mid Ya$

$X \to Z \mid a$

$Z \to Sa \mid G$

$Y \to Wa$

$W \to Sa$

$Y \to Wa$

$Y \Rightarrow Saa$

$X \to Z \mid a$

$X \to Sa \mid a \mid \epsilon$

$S \to Xa \mid Ya$

$Saa \quad aa \quad Saa$

WONDER CEMENT

Date : .....