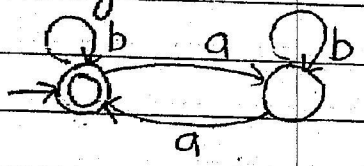
Page No.
Date : / /

Note

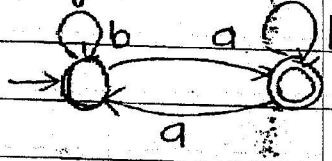
For Mode Machine, NFA & DFA will be same as no DC will be there & there will be trap.]

14. Even No of a



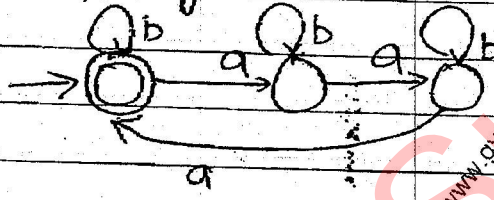
No of state = 2

15. odd no of 'a'



No of state = 2

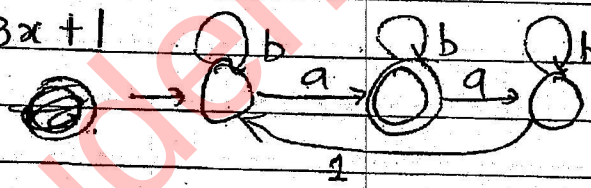
Multiple of 3



No of state = 3

{No of state in mod m = m+1}

for 3x+1

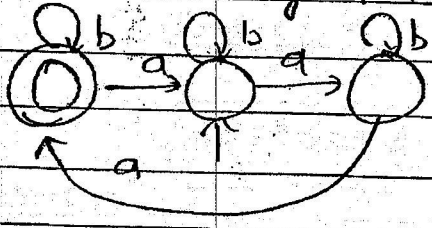


i.e. {$n_q \bmod 3 = 1$}

No of state = 3

{No of states in residue of Mod m = m}

Ex



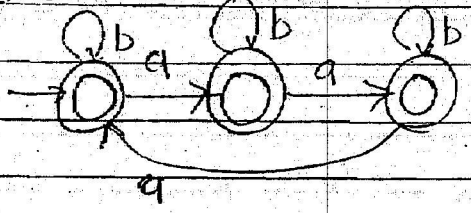
⇒ {$n_q(w) \bmod 3 = 2$}

EXCELLENT

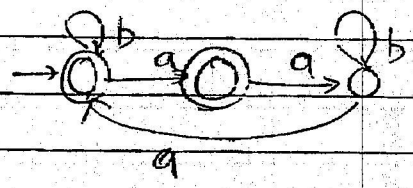
Page No.
Date : / /

2)

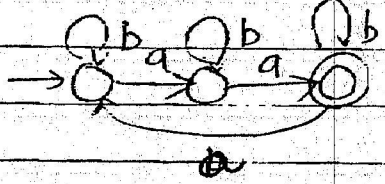
* $n_a \bmod 3 \leq 2$



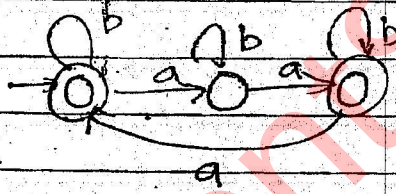
• $n_a \bmod 3 \leq 1$



• $n_a \bmod 3 > 1$



• $n_a \bmod 3 \neq 1$



• $\{n_a \bmod 3 \leq 1 = n_a \bmod 3 = 0\}$

• R.e for $n_a \bmod 4 \leq 1$

$$re = [(b^*ab^*ab^*ab^*ab^*)^* + b^*] + [(b^*ab^*ab^*ab^*ab^*) + b^*]ab^*$$
$$= [(b^*ab^*ab^*ab^*ab^*) + b^*)][\epsilon + ab^*]$$

4 → 2 residue

if $\epsilon + ab^* + ab^*ab^*$

↳ 3 residue

EXCELLENT

$n_a \bmod 4 \geq 2$

$re = [(b^*ab^*ab^*ab^*ab^*)^* + b^*)][ab^*ab^* + ab^*ab^*ab^*]$

ba^*b

- [No of final states is controlled by no of residue in the required o/p]

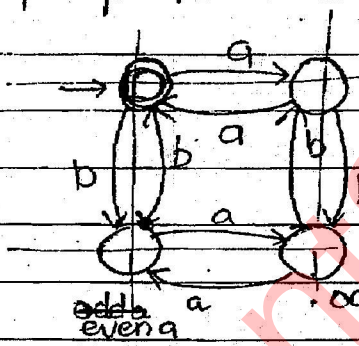
Mod Grid Machine-

16. Even no of a's & even no of b

Here if you have mod m & mod n on a & b

No of states, Grid size = m x n

$L = \{ w \mid n_a \bmod 2 == 0 \;\&\&\; n_b(w) \bmod 2 == 0 \}$



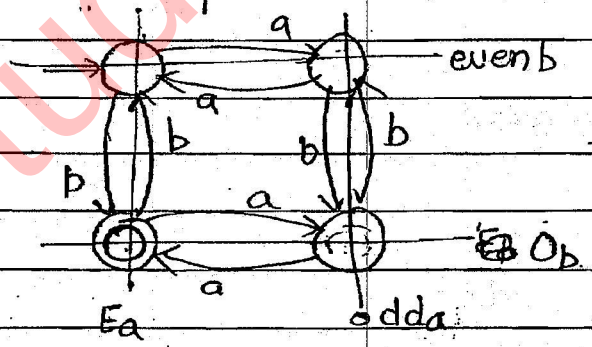
i.e

oddb

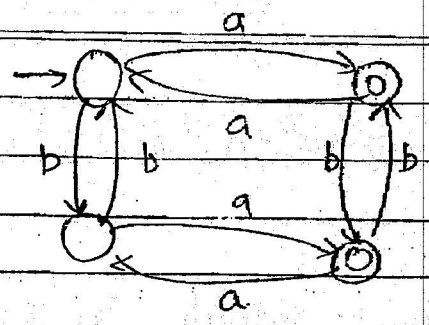
if No final state in a given DFA its language is empty

17 Even a & odd b

$L = \{ w \mid n_a \bmod 2 == 0 \;\&\&\; n_b \bmod 2 == 1 \}$

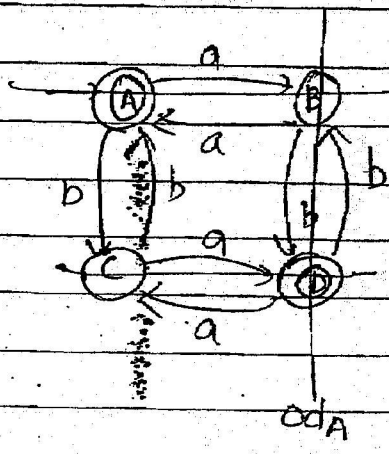


EXCELLENT



→ odd a acceptance
but it wont be minimal

Ex-



⇒ (even a & even b) or (odd a & odd B)

or odd a or even b but not both
= Odd a ⊕ even b
$= E_b + O_a - O_a E_b$
or, even a or odd b but not both
(A + D − C)

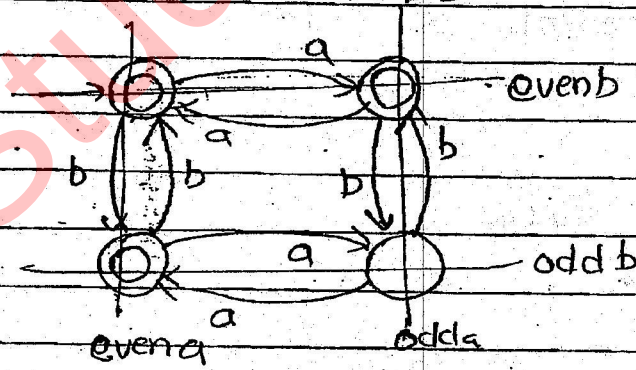
$$L = \{\omega \mid \underbrace{n_a(\omega)}_{O} + \underbrace{n_b(\omega)}_{O} = even\}$$
E E

if B & C are final state, then

$L = \{\omega \mid n_a(\omega) + n_b(\omega) = odd\}$

odd	Even
Even	odd

18- Even a's or Even b's

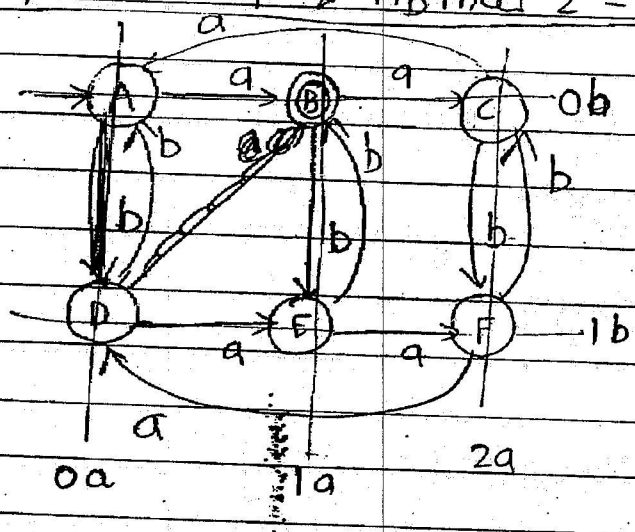


EXCELLENT

*To find the language
- find plane
- Do ∩ or ∪ acc to final state

Page No.
Date : / /

19. $n_a \bmod 3 = 1$ & $n_b \bmod 2 = 0$



- Make a grid
- Choose for final state (by intersection of plc

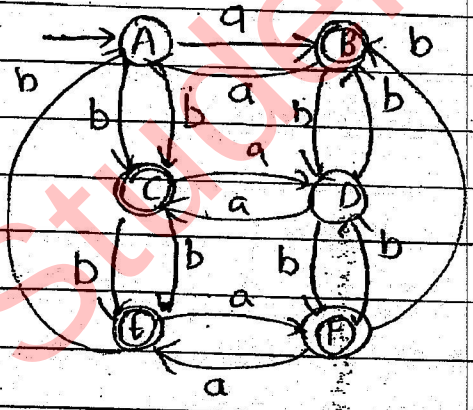
{No of states = m×n}

If B & DF are final state

$n_b \bmod 2 = 1$ or $n_a \bmod 3 = 1$

20 $\{w \mid n_a \bmod 2 \neq n_b \bmod 3\}$

- Do Grid view
- find the residue for each state



a b (residue combinat

Every A - 0 0

satisfies B = 1 0

C = 0, 1

D = 1, 0

E = 0, 2

F = 1, 2

for ≤ = A, C, D, E, F will be the final condⁿ

EXCELLENT

Page No.
Date : / /

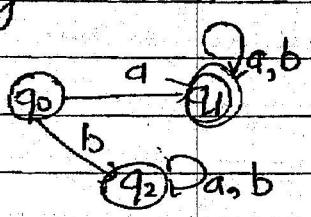
FA design

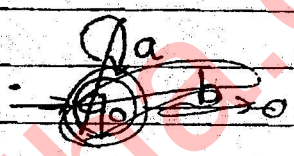
[• Minimal DFA is unique but minimal NFA not.]

• the no. of states is unique in DFA as well as NFA.

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

Ques1- starting with a,



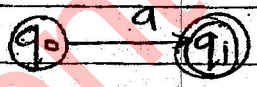


To design a Minimal Machine,

1.

Theorem - if |Wmin| has n states then no of state in (min. fa) ≥ n+1 (in case of infinite language)

as for start with a

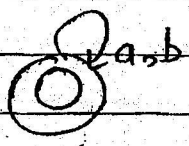


2 states are required atleast.

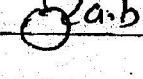
2. reuse the state if possible

3. filling up of blanks i.e for each state map for all I/P alphabet

Types of state in FA-

1. Permanent Accept - 

(It is used when string already satisfy the cond") one you enter, always accept.

2. Permanent ~~Accept~~ Reject/Trap - 

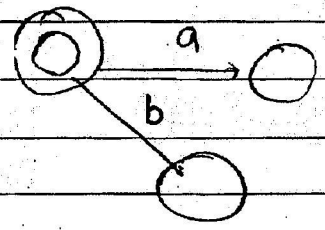
once enter just reject

(if string in that state it will never satisfy the cond")

EXCELLENT

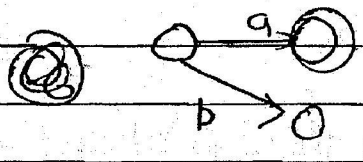
Page No.
Date : / /

3. Temporary Accept / ~~Reject~~ -



(it is used when condⁿ satisfy at moment but may fail in future)

4. Temporary Reject -

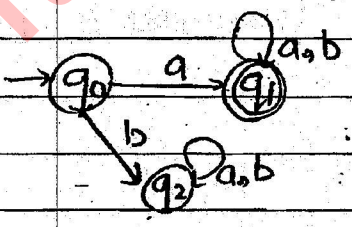


Can go anywhere (when condⁿ is not satisfied now, but may be satisfied in future).

- In Minimal DFA design, every state does an unique work.

Moves - staticscope
Regressed ← (hope)
forward →
trap

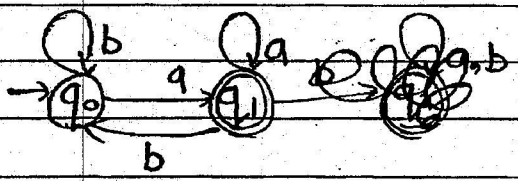
1) starting with a-



a(a+b)*

No of state = 3 (DFA)
No of state in min NFA = 2

2) End with 'a'



- RE = (a+b)*a

No of state in min DFA = 2
No of state in min NFA = 2

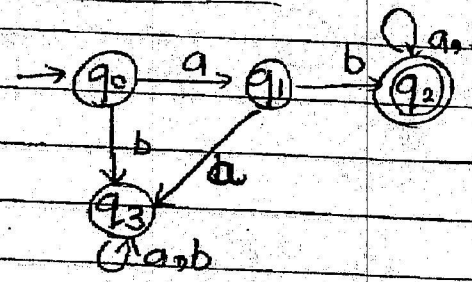
EXCELLENT

Use

- substring machine will always having PA state.
- starting with "_" will always have a trap & ending with "_" has regression

Page No.
Date

3. starting with "ab"

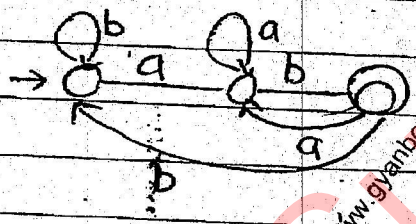


No of state (DFA min) = 4
No of state in (NFA min) = 3

RE ab(a+b)*

n size starting - require atleast 7 state + 1 for trap
8 for DFA minimal
7 for minimal DFA.

5. ending with ~~AB~~ ab



No of states = 3 (Minimal)

RE- (a+b)*ab

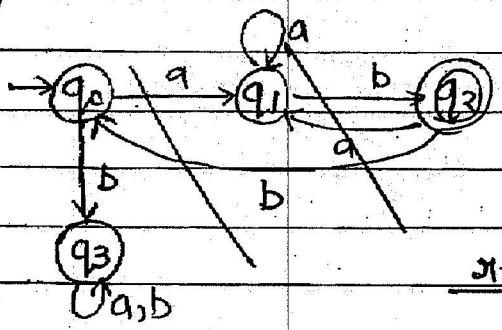
n size end string - require n+1 state (atleast)
as for n=2
it requires 3 states

~~Sad~~	TRAP	PA
starting with	✓	✓
Ending with	X	X
substring	X	✓

EXCELLENT

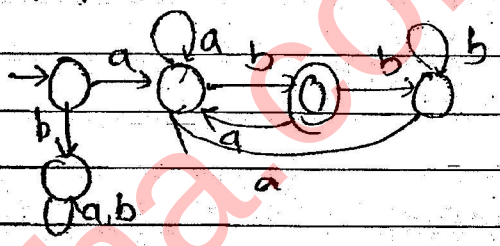
Page No.
Date : / /

6. Starting with a & end with b



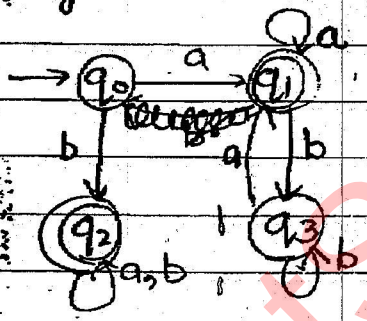
Min state (DFA) = 5
(NFA) = 4

re- a(a+b)*b



7. ~~Staring with aa end with ab~~

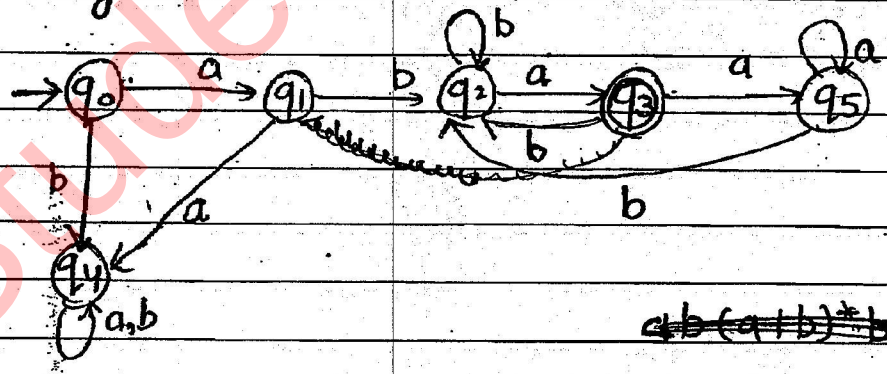
(7) Starting with a end with a



Min state (DFA) = 4
(NFA) = 3

~~a(a+b)*a~~
a(a+b)*a + a

(8) starting with 'ab' and end with 'ba'



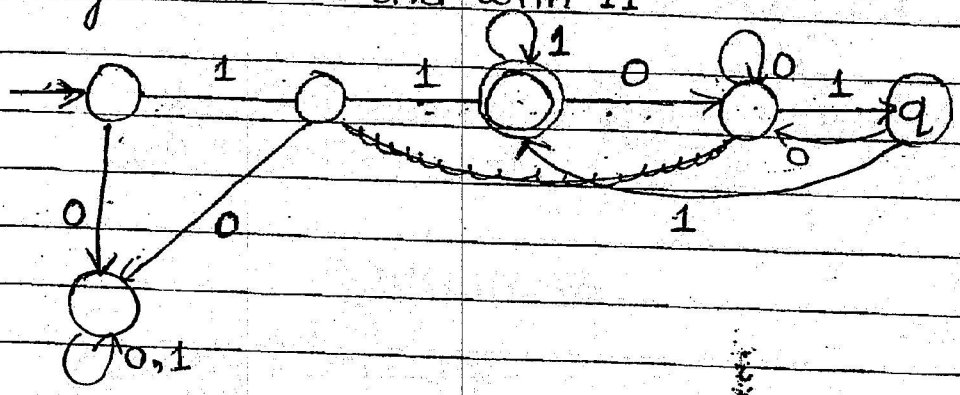
~~ab(a+b)*ba~~

EXCELLENT

1101101 1|001|10

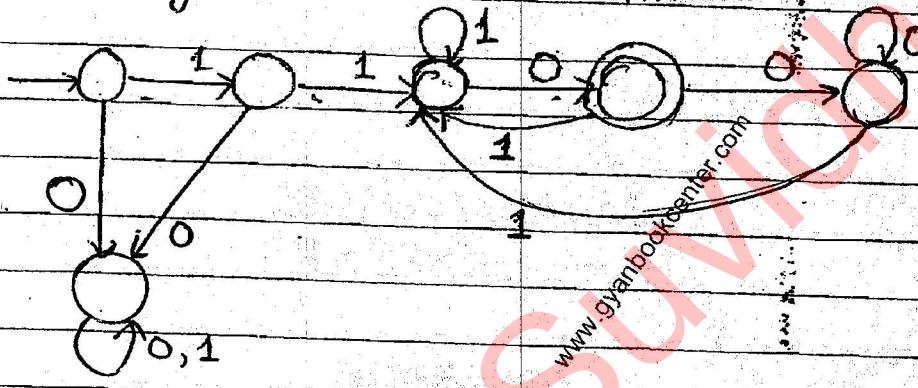
Page No.
Date : / /

9. starting with 11 & end with 11

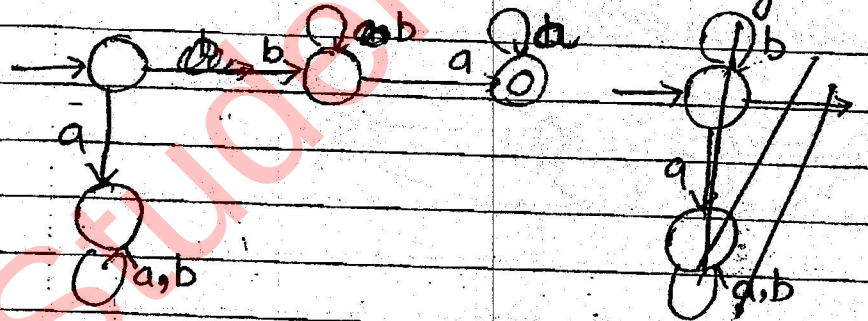


1111010

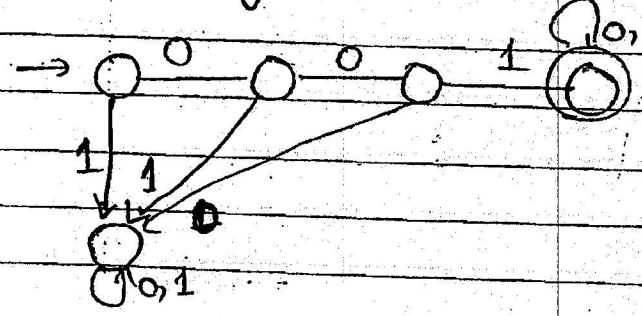
10. starting with 11 & end with 10



11. Not starting with a or not ending with b

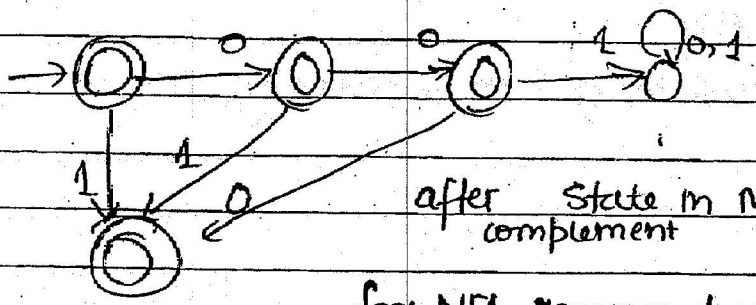


non starting with 001



Complement is

EXCELLENT



after complement State in Min DFA = state in original DFA

for NFA remove traps = state (min. NFA = 4

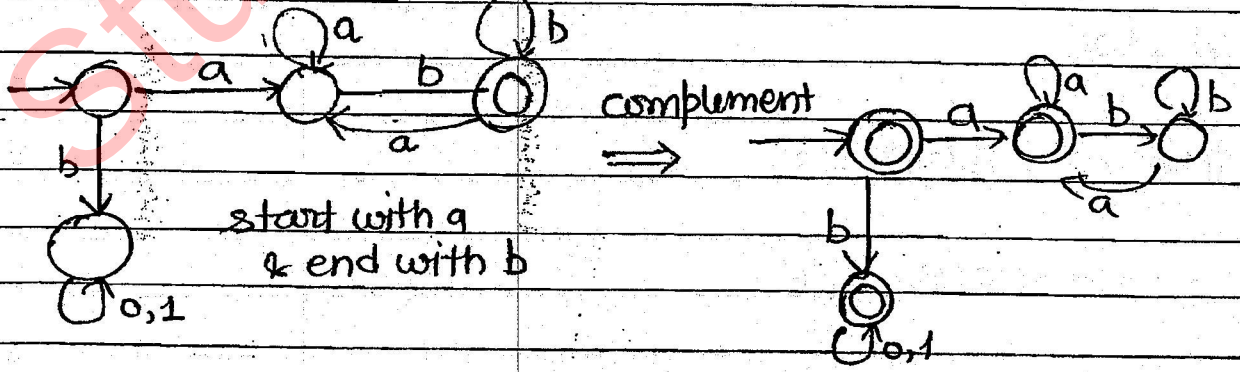
* if a DFA is given, its ~~equivalent~~ complement will have same state i.e no. of states.
* If a DFA has n state, state in its complement ie will be n.
* [If a NFA has n state, state in its complement will be not less than n as it will remove all the trap states.] (n-1)

12) Not start 'a' or not end with b

start A → A ∪ B ← B start B

$(\bar{A} \cup \bar{B}) = A \cap B$

∴ design start with a & ~~st~~ end with B & then complemen it.



EXCELLENT

Page No.
Date : / /

If a ques is given then like
Machine is given & checking like

Not A OR Not B

↓

[just complement the machine & answer two & matches them.]

* To complement NFA of state n

1. Convert into equivalent DFA by adding trap
may be n or n+1 (when no need to add trap) By trap

2. Complement then no of state is same.
3. Convert again to NFA by removing trap
if trap added previously → no of state = n or n+1 as no removal of trap

If trap not added → no of state = n or n-1

(b)

13) {01, 100}

4) containing atleast 1 'a'

5) ——— exactly 1 'a'

6 atmost 1 'a'

5 containing substring '0'

EXCELLENT

Page No.
Date : / /

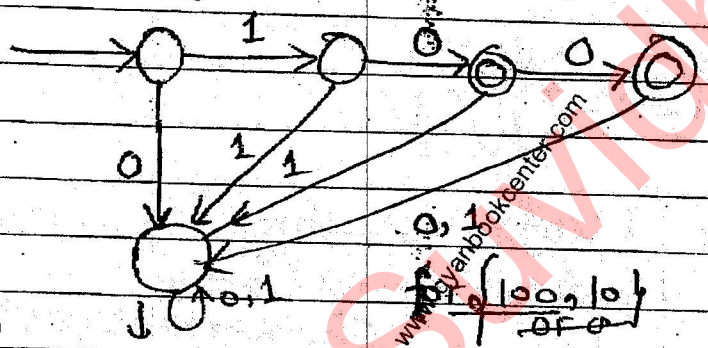
In case of finite language, if $|w_{max}| = n$ then

{no of states in min. nfa = n+1} — trap state may or may not required

{no of states in min dfa = n+2} — trap state again

↓ because n+1 (for max string) + 1

1. First accomodate biggest string.



if finite language & DFA is generated trap is required always at final state

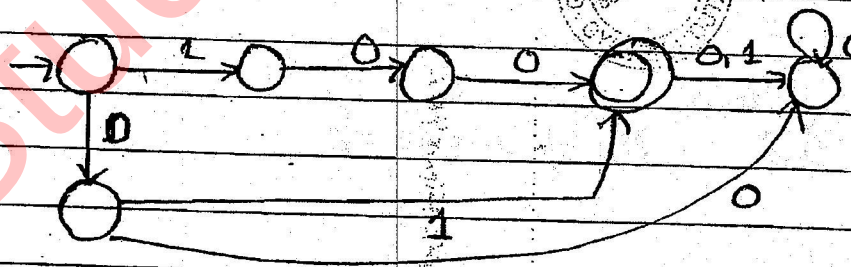
{10, 100} or {100, 10}

{10, 100} or {100, 10}

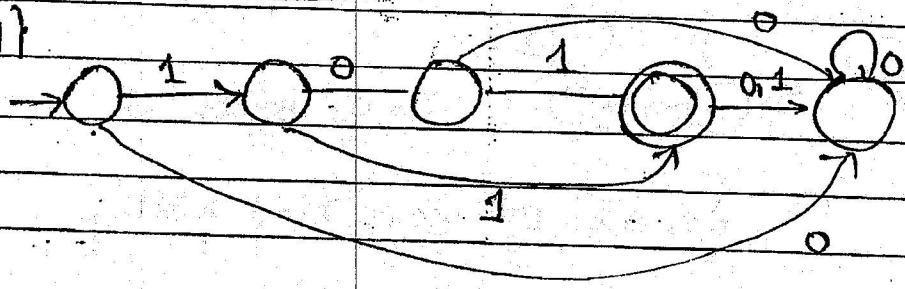
No of state = 4 (NFA)
= 5 (DFA)

2. try to accomodate other in it if possible.

{01, 100}



{101, 11}

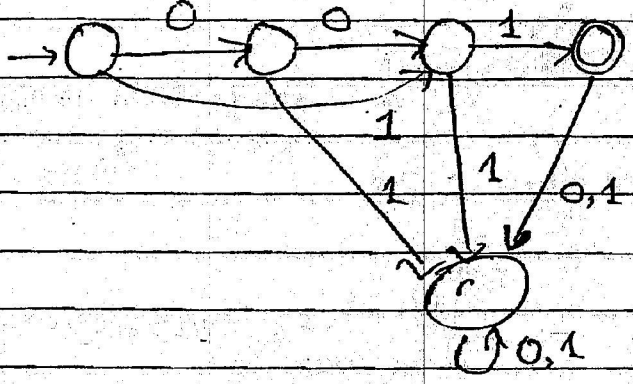


EXCELLENT

(DFA → min NFA)

Page No.
Date : / /

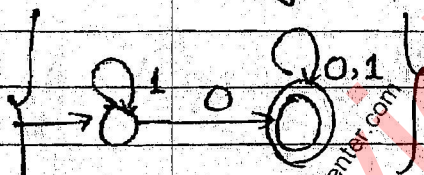
{001, 11}



{ try to use each start or ending }

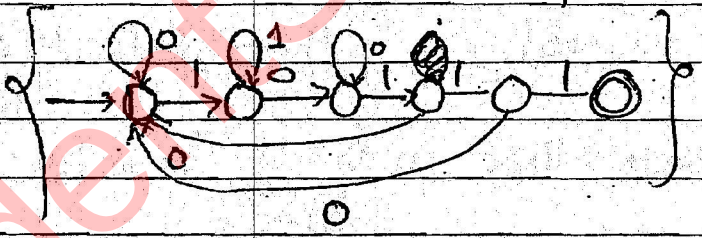
14 - containing the substring - Never had a trap

substring-0

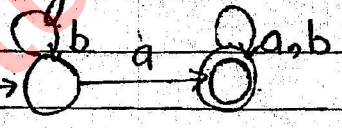


for [1 length substring = 2 states
n length → n+1 states
(for both NFA & DFA)]

{1011}

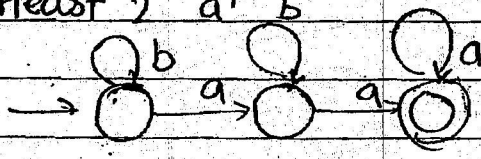


15) containing atleast 1 'a'



No of states = 2

atleast 2 'a' b



No of states = 3

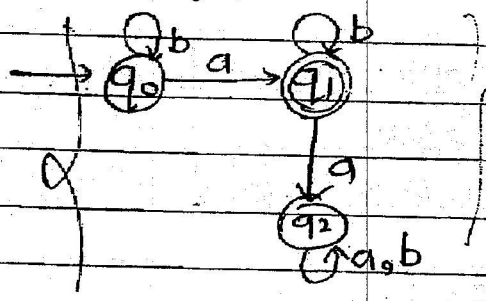
{n a's - will have n+1 state}

EXCELLENT

atleast - 1

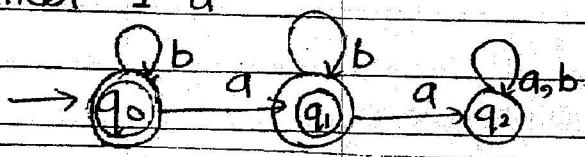
Page No.
Date : / /

16.) Exactly 1 'a' (trap exists here)



No of states = 3 (DFA)
= 2 (NFA)

17.) Atmost 1 'a'

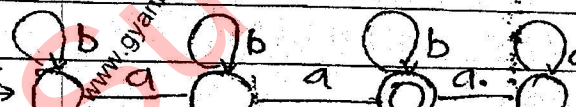


No of final state = 2
(trap also exists here)

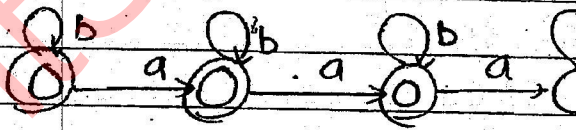
- if one final state, exactly come into existence.

for 2a

exactly 2a's



atmost 2a's



for n (exactly) states = n+2 (DFA)

n (atmost) = n+2 (DFA) but no of final stat = n+1

for NFA, n (exactly), states = n+1
n (atmost), state n+1

EXCELLENT

Page No.
Date : / /

for a language, there exist more than one regular grammar.

elegant expression- expression with no redundancy which is not required.

- It is possible to have more than one elegant expression

- Regular expression is defined as follows-

1. ϕ, ϵ are regular expression.
   - if $r = \phi$ $L(r) = \{\}$
   - $r = \epsilon$ $L(r) = \{\epsilon\}$
2. $a \in \Sigma$ is regular expression.
   - $r = a$ $L(r) = \{a\}$
3. r, s are r.e., then
   $\Rightarrow r + s$ is r.e.,
   $\downarrow$ + must operate b/w two r.e.
   - $\{ L(r+s) = L(r) \cup L(s) \}$
   - $r = a+b$
   - $\therefore L(r) = \{a, b\}$
   - $r = ab + ba$
     $L(r) = \{ab, ba\}$
   - if $r = a^* + b^*$
     $L(r) = \{$ string of any no of a , string of any no of b $\}$
4. if r is r.e., & s is r.e.
   then '$r.s$' is also r.e
   $\{ L(r.s) = L(r).L(s) \}$

   Ex- $r.s = a^*b^*$
   $L(r.s) = \{$ any no of a with any no of b but a followed by b but not b not followed by a $\}$

+ : binary operation.

7. $a^*.+$ – not a regular expression
8. $a^*.\phi$ – is a regular expression
   $a^*.\phi = \phi$
9. $a^*.\varepsilon = a^*$

10. If r is r.e., then r^* is also a r.e.,

11. $()^*$ is not a regular expression.
    $(\phi)^*$ is regular expression.
12. If R is a regular expression
    (R) is also a regular exp.
    () is used to violate the precedence

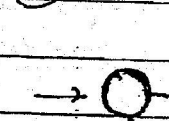
$$(\ (\ ) \geq * \geq . \geq +\ )$$

- $r = (ab)^*$ $\quad L(r) = \{\varepsilon, ab, (ab)^2, (ab)^3 \ldots\}$

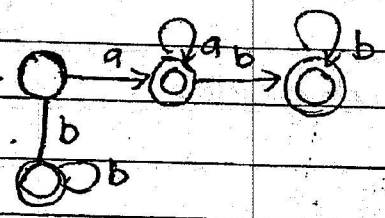
~~$r = (a^* +$~~

- ~~$r = a^*$~~
  a^*b^* = any string with out ba
  $= (a+b)^*\, ba\, (a+b)^*$

$r = (a+b)^*$ = [any combination of a & b]
~~a~~ a can follow b as well b can follow a as + is commutative.



a^*b^*



	a	b
q_0	q_1	q_3
q_1	q_1	q_2
q_2	q_2	o
q_3		

EXCELLENT

Page No.
Date : / /

• Power of Finite Automata is pattern matching

1. for starting with a - $a(a+b)^*$
2. end with a - $(a+b)^*a$
3. for a at 3rd pos^n - $(a+b)(a+b)a(a+b)^*$
4. for 3 bit string & a at 3rd pos^n - $(a+b)(a+b)a$
5. 3 bit is a & 2nd last is b
   - $(a+b)(a+b)a(a+b)^*b(a+b)$
6.
7. Starting with 11 & end with 11
   - $11(0+1)^*11+111+11$ (adding out the missed bits)
8. start - 10 end - 01 | 1
   - $10(0+1)^*01$ + 10
9. start with 00 & end with 11
   - $00(0+1)^*11$
10. not starting with 01
    - ~~$\{(1+0)^* - 01(1+0)^*\}$~~
    - $= 00(1+0)^* + 11(1+0)^* + 01(1+0)^* + 0 + 1 + \varepsilon$
      - $00(1+0)^* + 11(1+0)^* + 01(1+0)^*$ ↓ possible comb^n
      - $0 + 1 + \varepsilon$ [can have lesser length strings]
5. not starting with a or not ending with b
   - $b(a+b)^* + (a+b)^*a + \varepsilon$
     bcoz of 'OR'
     as b will be in not ~~ending~~ starting with a

EXCELLENT

(a+b)*a
= a,

Page No.
Date : / /

Mode Machines are always regular

12. If L = {001, 11}
re = 001 + 11
L = {11, 001, 10, 110}
re = 11 + 001 + 10, 110
= 11(ε+0) + 001 + 10
L = {011, 111}
re = (0+1)11
L = {110110, 11001 0}
re = 11(01+00)10

13. containing the substring 'a'.
re = (a+b)*a(a+b)*

14. Contains atleast 1 a
(a+b)*a(a+b)*
exactly 1a b*ab*
atmost 1a = atmost exactly 1 a + exactly 0 a
= b* + b*ab*
= b*(ε+ab*)
atleast 2 a = (a+b)*a(a+b)*a(a+b)*
exactly 2a = b*ab*ab*
atmost 2a = b* + b*ab* + b*ab*ab*

atmost 3a = b* + b*ab* + b*ab*ab* + b*ab*ab*
= b*(ε+ab*(ε+ab*)) = b*(ε+ab*(ε+ab*))

atleast 1a
↳ (a+b)*a(a+b)* = b*a(a+b)* = (a+b)*ab*

EXCELLENT

for atleast 2 a's

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

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

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

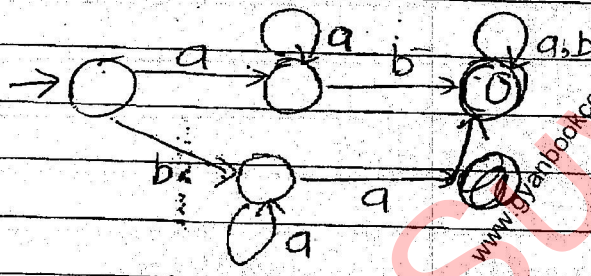
$\cong b^* a\,(a+b)^* a\, b^*$

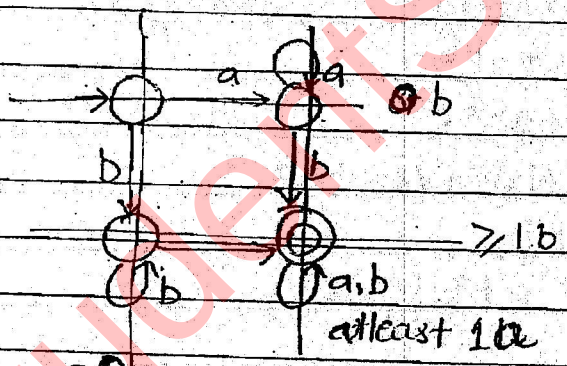
all these are equivalent to each other.

16-Ap-91

1. atleast 1 "a"

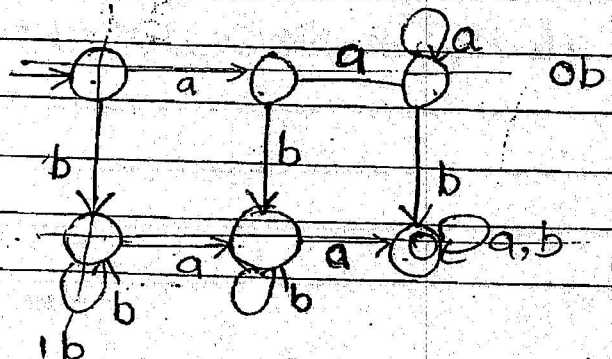
II. atleast 1 "a" & at least 1 "b"





final state is obtained by intersection of line metting the result.

atleast 2a & atleast 1b

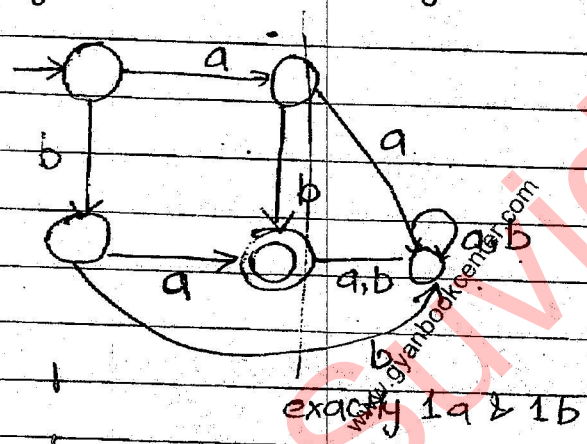


Page No.
Date : / /

for atleast m "a" & atleast n "b"s

Min. DFA	(m+1)(n+1)
Min. NFA	(m+1)(n+1) as there is no trap state.

12. Exactly 1 "a" & exactly 1 "b" (finite language)



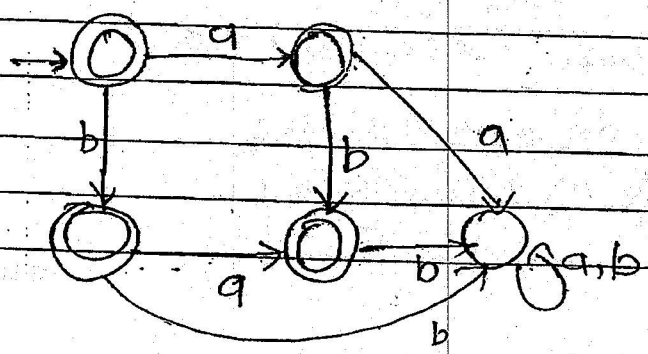
if there is a trap condition lead to exactly)

R.e = ab + ba

~~exactly 2'a' & exactly 1'b~~

Min DFA	(m+1)(n+1)+1 (as trap only introduced)
Min NFA	(m+1)(n+1)

13. atmost 1a & atmost 1 b
~~at~~ =0a or 1a (or 0 or 1 b)



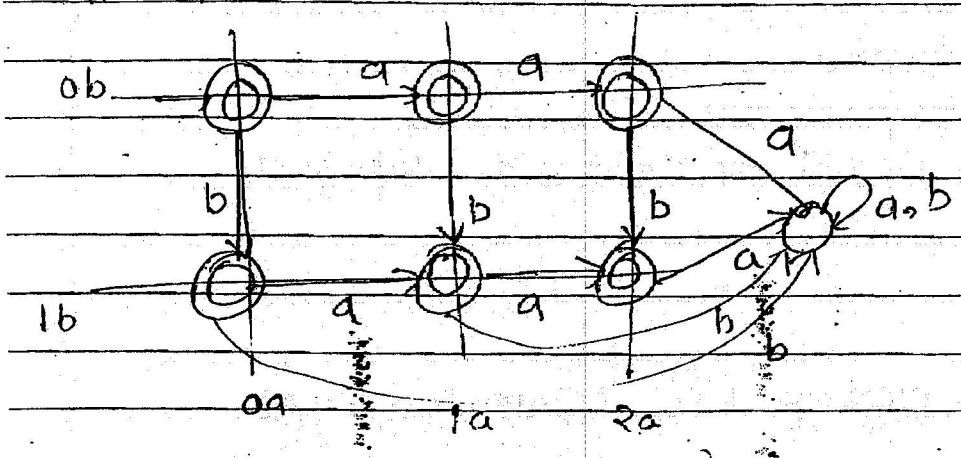
in any DFA, if all states are final state then language is Σ^* this property is only satisfied by DFA

EXCELLENT

Page No.
Date : / /

atmost

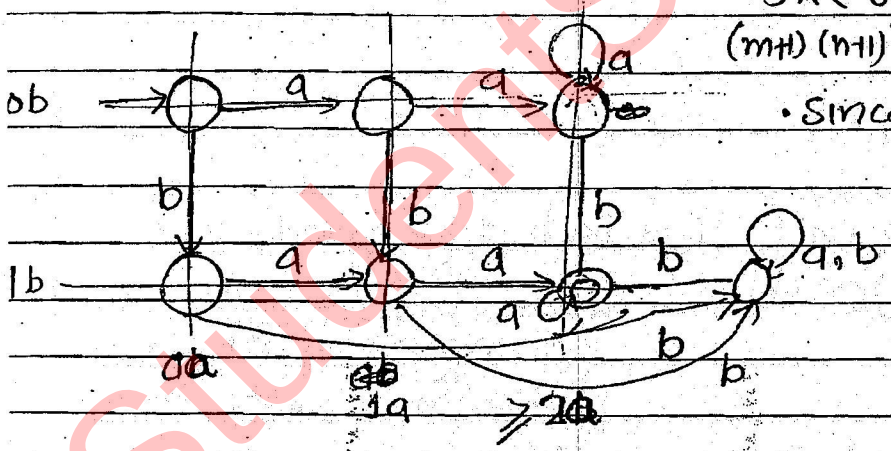
* atmost 2a's & ^1 b



min. DFA = (m+1)(n+1) + 1
min. NFA = (m+1)(n+1)

Atleast 2a & exactly 1b

[Grid is required of size 3×2 or 2×3 (m+1)(n+1)]



• Since exactly is there, trap is required)

Min. DFA- (m+1)(n+1) + 1
↓ for atleast ↓ for trap (as exactly is there)

for shortcut- { atleast m a → m+1 state
exactly n b → n+1 state
1 trap }

EXCELLENT

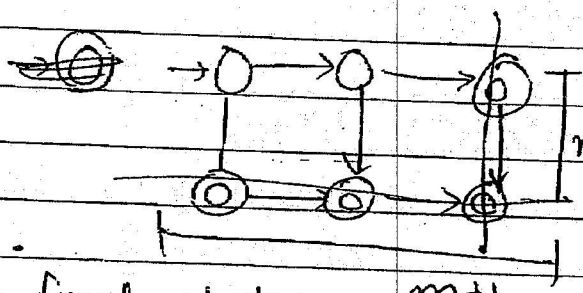
To know lang. of given machine-
map the line for final state & do intersection (for AND)

Page No.
Date : / /

If you want to do 'OR' you cannot go on with trap

(for OR) - map the line for your requirement & put the final state where Union of line.

atleast 2a OR atleast 1b



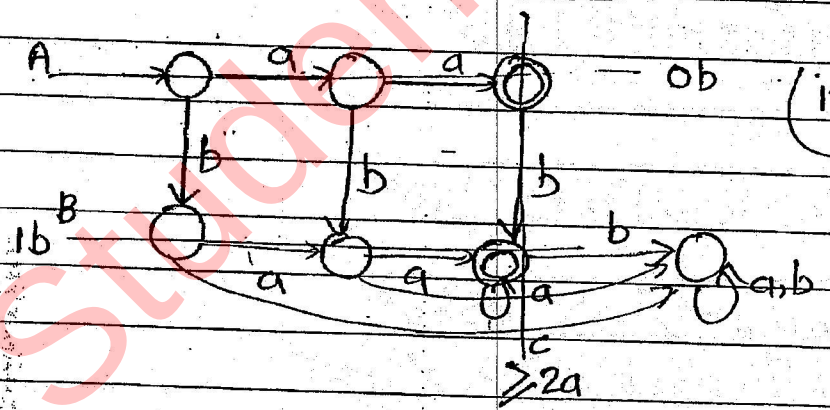
no trap & L shaped
n+1 final state
(OR)

No of final states -

m+1+n+1 - common states

= m+n+1

* atleast 2'a' & atmost 1'b'' OR Not possible



= final state =
(intersection of A, B, C)

AND machine have less ~~fini~~ state in comparison to OR

Product automata - AND operation can be done.

~~If then~~

Mode Machine - $\Sigma = \{a, b\}$

14. Even no "a" ~~$b^*(aa)^*$~~

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

$r.e = (b^*ab^*ab^*)^* + b^*$

15. odd no "a" $r.e = [(b^*ab^*ab^*)^* + b^*]ab^*$

attachment of 1 a ~~with~~ to even no of a

~~En~~ Even no "a" $\Rightarrow L = \{w \mid n_a(w) \bmod 2 == 0\}$

odd no "a" $\Rightarrow L = \{w \mid n_a(w) \bmod 2 = 1\}$

even no "a" $\Rightarrow L = \{w \mid n_a(w) \bmod 2 = 2x, x \in Z^+\}$

for no of a's to be multiple of 3-

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

$r.e. = (b^*ab^*ab^*ab^*)^* + b^*$

for "not multiple of 3"

$3x+1$

$r.e = [(b^*ab^*ab^*ab^*)^*a + b^*]ab^*$

for $3x+2$

$r.e. = [[b^*ab^*ab^*ab^*)^* + b^*]ab^*ab^*$

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

EXCELLENT