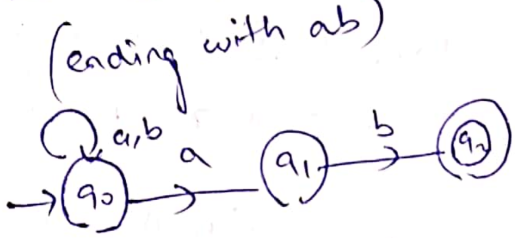
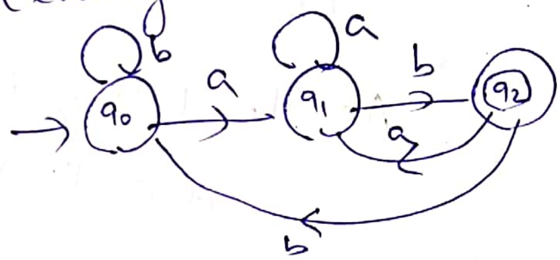
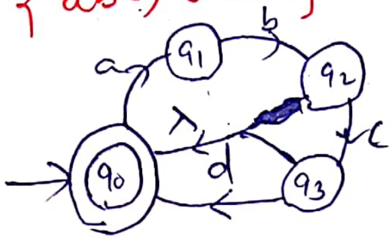
2019 Mid Sem. Solution

1) a) Differentiate between NFA and DFA, explain with an example.

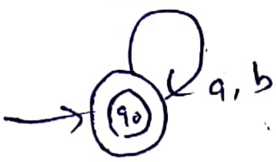
NFA	DFA
→ Transition function.	→ Transition function.
$\boxed{\delta : Q \times (\Sigma \cup \epsilon) \rightarrow 2^{Q}}$	$\boxed{\delta : Q \times \Sigma \rightarrow Q}$
→ There can be zero, one or more transitions from a state with or without giving any input	→ There can be exactly one transition from a state on an input symbol.
→ Ex: $\Sigma = \{a, b\}$ (ending with ab) 	→ Ex: $\Sigma = \{a, b\}$ (ending with ab) 

1) b) Design an NFA with 4 states for the set $\{abc, abcd\}^*$



1) c) Design a minimal DFA for $r = (b^*a^* + ab + \lambda)^*$

$$r = (b^*a^* + ab + \lambda)^* = (a+b)^*$$



Scanned by CamScanner

1) d) Which language $(\phi^*)^*$ and $a.\phi$?

$(\phi^*)^* = (\epsilon)^* = \epsilon$	(Empty string)
$a.\phi = \phi$	(Empty Language or Empty set)

1) e) Suppose $L_1 \cup L_2$ is regular & L_1 is finite. Can we conclude that L_2 is regular ? True / false. Justify your answer.

TRUE

L_2 is also regular. bcoz.

$L_2 = (L_1 \cup L_2) - (L_1 - L_2)$

As the $L_1 \cup L_2$ is regular, L_1 is also regular bcoz it is finite (being a finite set minus something) and regular languages are closed under set difference

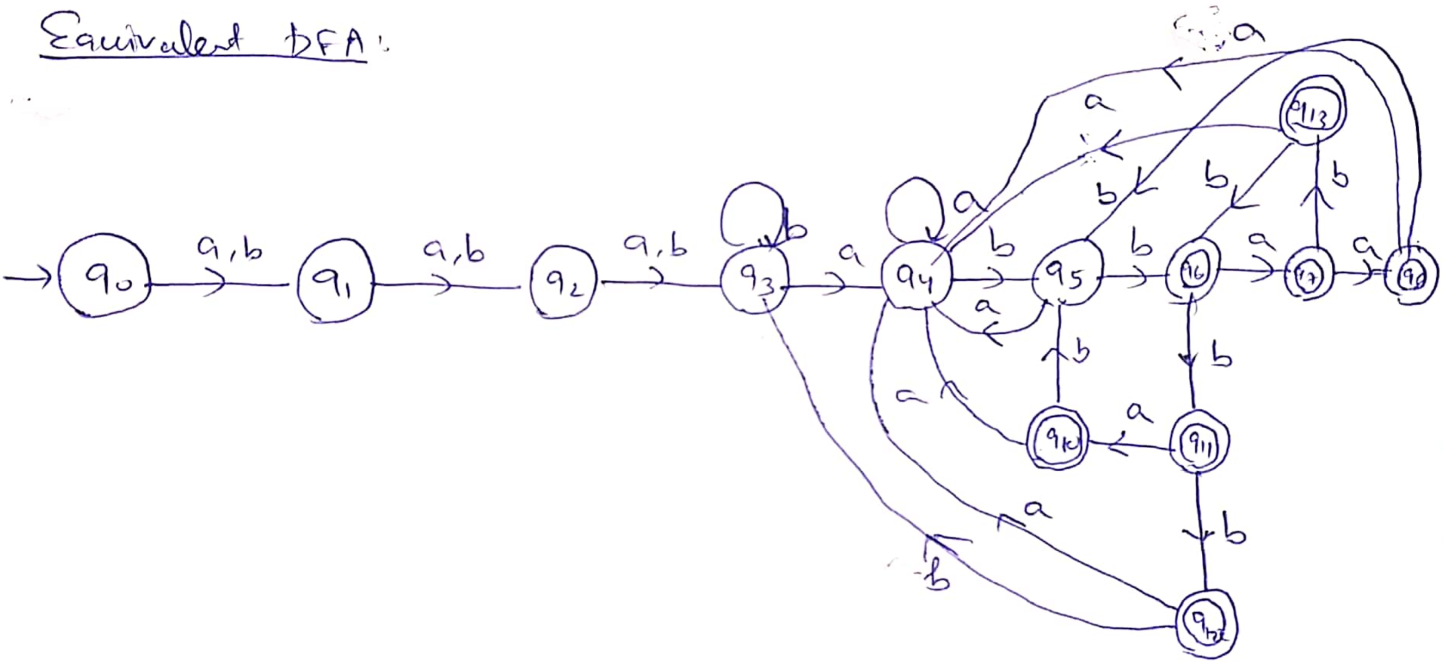
2) a) Design a DFA for the language over $\Sigma = \{a, b\}$

$L = \{ w_1 \, abb \, w_2 \,,\ |w_1| \geq 3,\ |w_2| \leq 2 \}$

RE: $(a+b)^3 \, b^* \, abb \, (\lambda + a + b)^2$

$= (a+b)(a+b)(a+b) \, b^* \, abb \, (\lambda + a + b)(\lambda + a + b)$

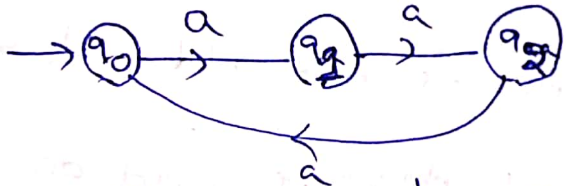
Equivalent DFA:



Scanned by CamScanner

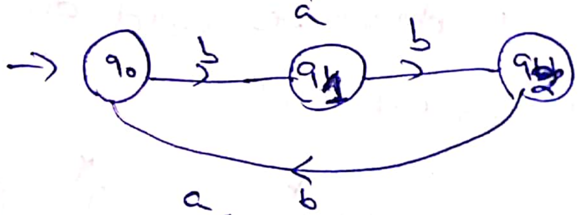
b) $L = \{ w : n_a(w) \bmod 3 < n_b(w) \bmod 3 \}$

FA1:



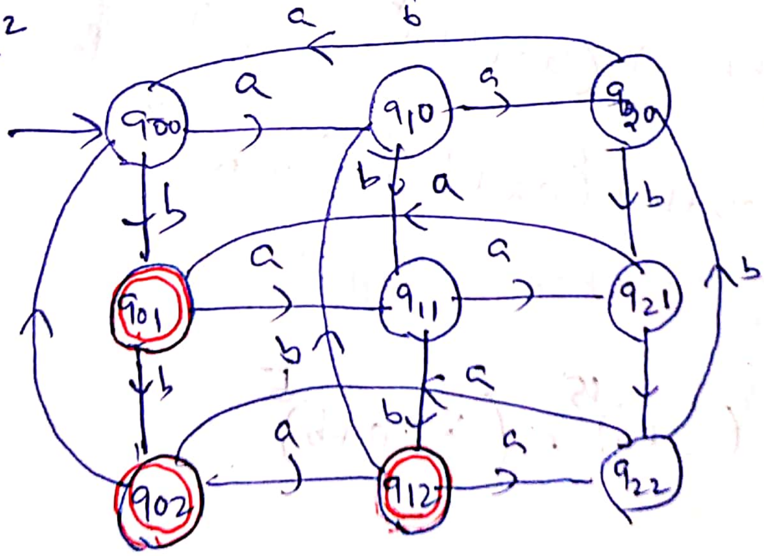
$n_a(w) \bmod 3$

FA2:



$n_b(w) \bmod 3$

Composed FA:
FA1 < FA2



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

Final state: $0 < 1$ ✓
$0 < 2$ ✓
$1 < 2$ ✓

$\therefore q_{01}, q_{02}, q_{12}$

Scanned by CamScanner

3) a) Write Regular expressions for the following languages over $\Sigma = \{a, b\}$

i) $L_1 = \{uwu \mid u, w \in \Sigma^*, |u| \leq 2\}$

$R = \epsilon \Sigma^* \epsilon + a \Sigma^* a + b \Sigma^* b +$
$ab \Sigma^* ab + ba \Sigma^* ba + bb \Sigma^* bb$
$aa \Sigma^* aa +$

ii) $L_2 = \{u \mid u$ has atleast 1 triplet letter $\}$

$R = (a+b)^* (aaa+bbb) (a+b)^*$

iii) $L_3 = \{u \mid u$ has even no. of a's and odd no. of b's $\}$

$R = \{ (aa)^* (bb)^* b \}$

iv) $L_4 = \{u \mid |u|$ is atleast 15 & atmost 20 $\}$

$R = (a+b)^{15} (\lambda + a + b)^5$

Scanned by CamScanner

3) b) Design a DFA for the set of all non-negative integers divisible by 4.

(String symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

or Same as question:

Design a DFA to accept decimal strings divisible by 4.

Solution:

Divisible by 4 means, divisor(k) = 4

radix (r) = 10 (As decimal)

digits (d) = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

remainder (i) = 0 to k-1 = {0, 1, 2, 3}

which implies q_0, q_1, q_2 & q_3 are the states of DFA.

Transitions can be computed using the following relation:

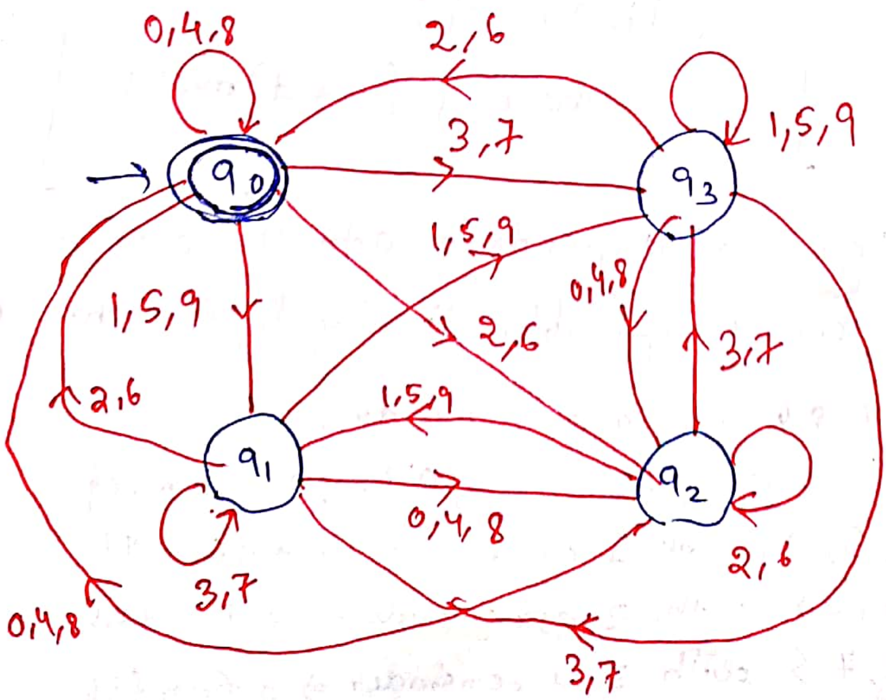
$$\boxed{\delta(q_i, a) = q_j \quad \text{where } j = (ri + d) \bmod k}$$

Let us group of digits from 0 to 9 based on remainders, we get after dividing by 4 as shown below:

- {0, 4, 8} with 0 as remainder,
  So δ from {0, 4, 8} => δ from {0}
- {1, 5, 9} with 1 as remainder => δ from {1}
- {2, 6} with 2 as remainder => δ from {2}
- {3, 7} with 3 as remainder => δ from {3}

Scanned by CamScanner

remainder (i)	d	$(r\,i+d) \bmod k = j$ $(10i+d) \bmod 4 = j$	$\delta(q_i,d)=q_j$
q_0 ($i=0$)	0	$10\times0+0 \bmod 4 = 0$	$\delta(q_0,0)=q_0$
	1	$10\times0+1 \bmod 4 = 1$	$\delta(q_0,1)=q_1$
	2	$= 2$	q_2
	3	3	q_3
q_1 ($i=1$)	0	$10\times1+0 \bmod 4 = 2$	q_2
	1	$10\times1+1 \bmod 4 = 3$	q_3
	2	0	q_0
	3	1	q_1
q_2 ($i=2$)	0	$10\times2+0 \bmod 4 = 0$	q_0
	1	1	q_1
	2	2	q_2
	3	3	q_3
q_3 ($i=3$)	0	$10\times3+0 \bmod 4 = 2$	q_2
	1	3	q_3
	2	0	q_0
	3	1	q_1



Scanned by CamScanner

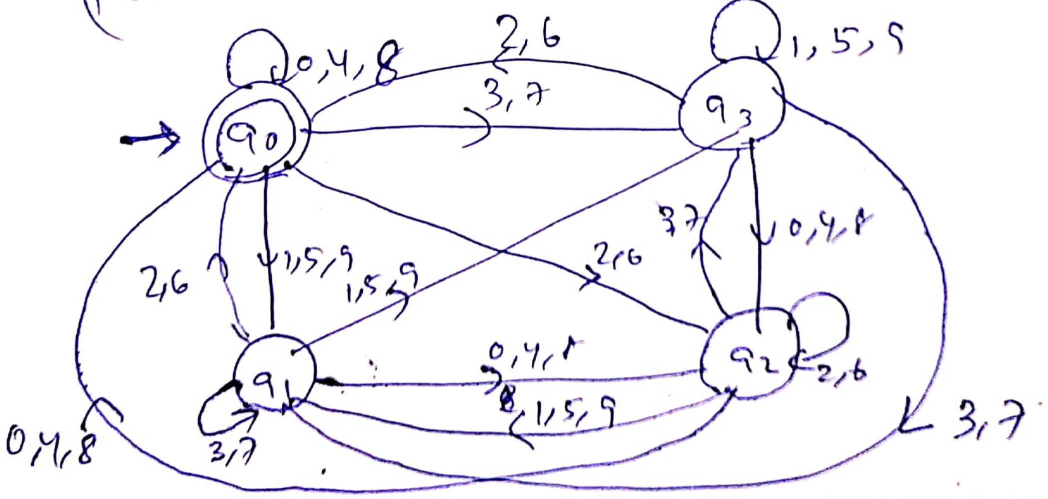
3)(b)

As divisible by 4, remainder = 0 so q_0 is final state

as mod 4 i.e. it requires 4 no of state.

	0	1	2	3	4	5	6	7	8	9
$\rightarrow * q_0$	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1
q_1	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3
q_2	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1
q_3	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3

[write q_0, q_1, q_2, q_3 in order, from input 0 to 9]



4) a)

Let L_1 & L_2 be two ~~co~~ languages over same alphabet Σ.

Given that L_1 & $L_1 \cdot L_2$ both are regular.

Prove or disprove that L_2 must be regular.

Sol) **FALSE**

Let L_2 be a not regular language

$$L_2 = \{a^n b^n \mid n \geq 0\}$$

$$L_1 = \phi \quad (\text{regular})$$

Then, we know,

$$\boxed{a \cdot \phi = \phi}$$

So, $L_1 \cdot L_2$

$$= \phi \cdot \{a^n b^n \mid n \geq 0\}$$

$$= \phi \quad (\text{regular}).$$

$\therefore$ So, we disprove that L_2 must not be regular.

i.e $\boxed{L_2 \text{ is not regular}}$ (Proved)

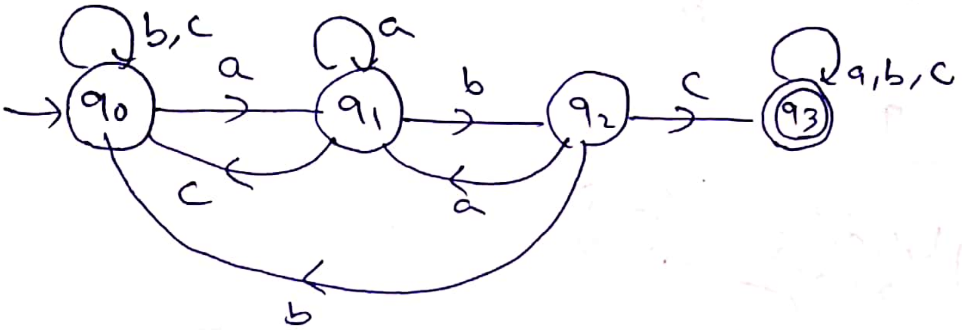
Scanned by CamScanner

4) b) Construct a DFA that accepts the language L;

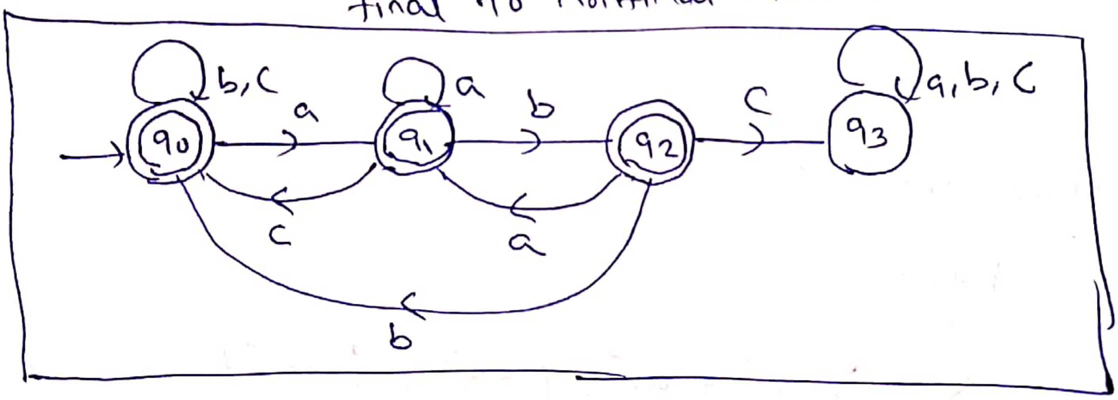
L = {w | w does not contain a substring abc}

over $\Sigma = \{a,b,c\}$. Convert this DFA to regular expression using state elimination method.

Solution: First draw the DFA for w containing substring abc.



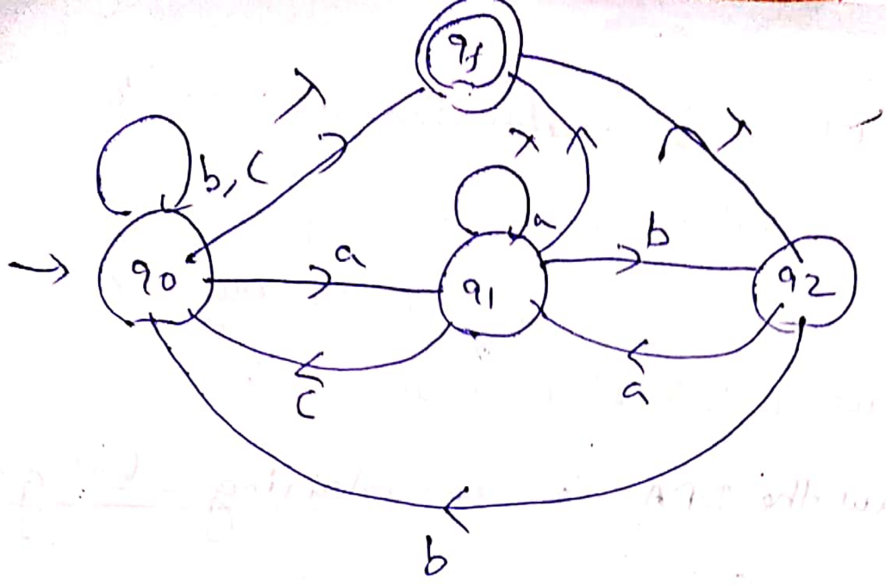
Then complement the language where 'w' does not contain abc' by converting Nonfinal to final & final to nonfinal state.



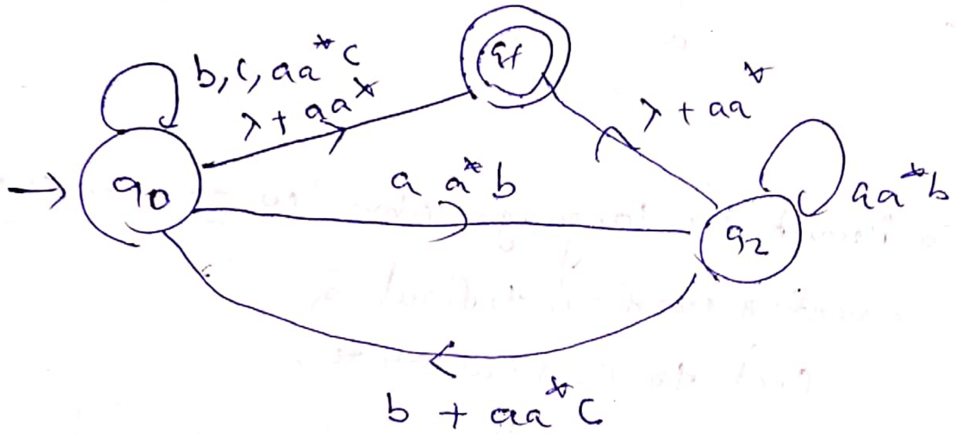
As, there are multiple final states, then introduce a new final states, & λ-production to that new final state

Here, q_3 is deadstate bcoz, there is no path to final state from q_3. So remove q_3.

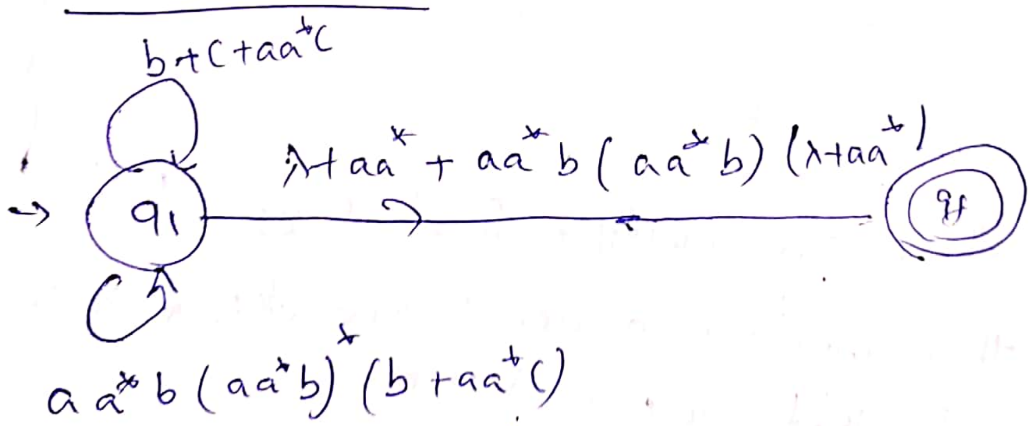
Scanned by CamScanner



Remove q_1



Now remove q_2



$\therefore$ regular exp^n is: $\left(b+c+aa^*c+aa^*b(aa^*b)^*(b+aa^*c)\right)^* \cdot \left(\lambda + aa^* + aa^*b(aa^*b)(\lambda+aa^*)\right)$

— Ans

Scanned by CamScanner

5) Convert given NFA to equivalent DFA.

δ	0	1
→ A	(B,D)	B
* B	C	(B,C)
C	D	A
* D	–	A

DFA TT

δ	0	1
→ A	(B,D)	B
* (B,D)	C	(A,B,C)
* B	C	(B,C)
C	D	A
* D	φ (Dead)	A
* (A,B,C)	(B,C,D)	(A,B,C)
* (B,C)	(C,D)	(A,B,C)
* (B,C,D)	(C,D)	(A,B,C)
* (C,D)	D	A
φ	φ	φ

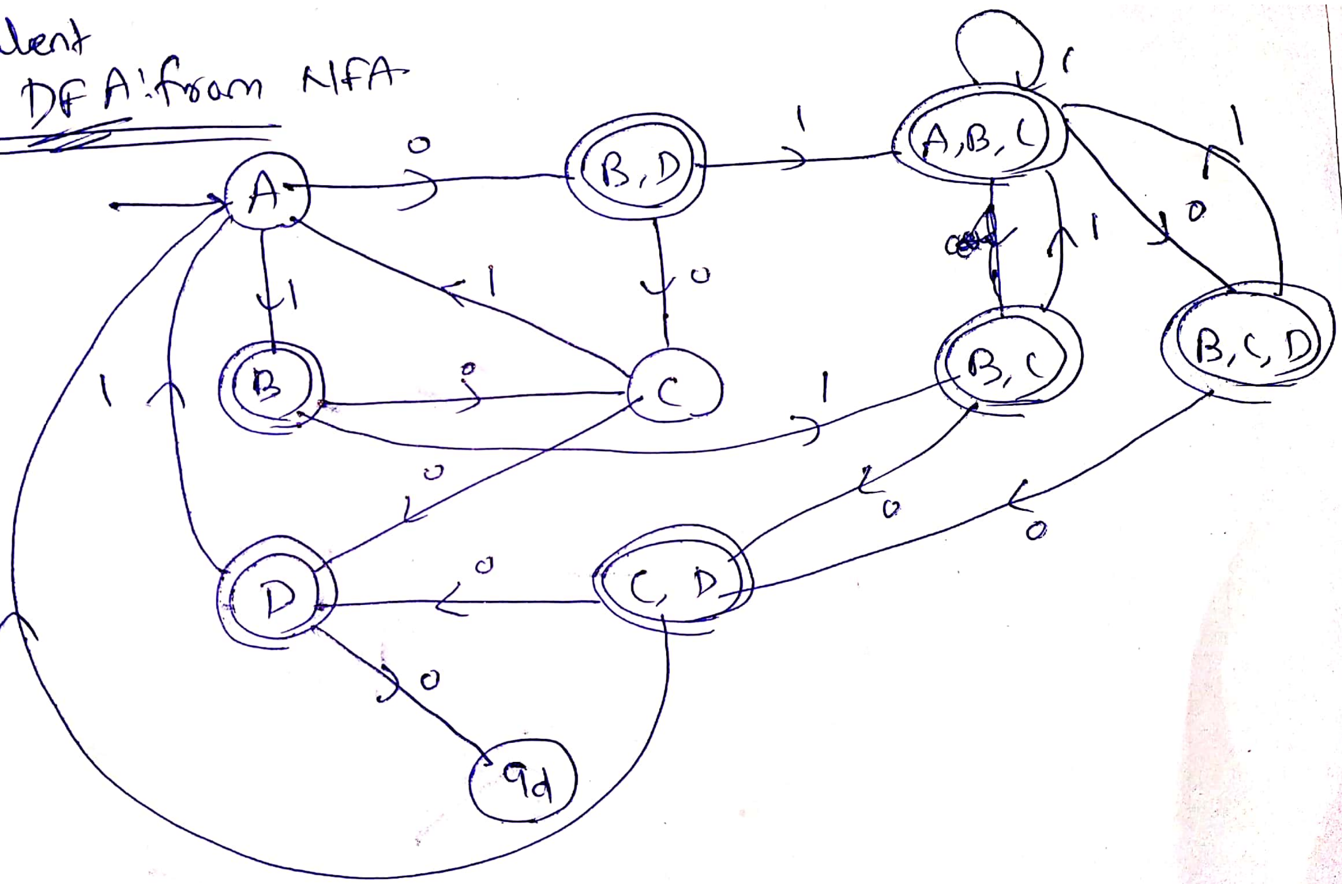
⇒

δ	0	1
→ q_0	q_1	q_2
* q_1	q_3	q_5
* q_2	q_3	q_6
q_3	q_4	q_0
* q_4	q_d	q_0
* q_5	q_7	q_5
* q_6	q_8	q_5
* q_7	q_8	q_5
* q_8	q_4	q_0
q_d	q_d	q_d

Scanned by CamScanner

Equivalent

DFA: from NFA



Minimization of DFA by State equivalence method:

0-equivalence: $\{\underbrace{q_0, q_3, q_d}_{\text{Nonfinal}}\}$ $\{\underbrace{q_1, q_2, q_4, q_5, q_6, q_7, q_8}_{\text{Final}}\}$

1-equivalence:
$\{q_0\}$ $\{q_1, q_2\}$
$\{q_3\}$ $\{q_4\}$ $\{q_8\}$
$\{q_d\}$ $\{q_5, q_6, q_7\}$

2-equivalence:
$\{q_0\}$ $\{q_1, q_2\}$
$\{q_3\}$ $\{q_4\}$ $\{q_6, q_7\}$
$\{q_d\}$ $\{q_5\}$ $\{q_8\}$

3-equivalence:
$\{q_0\}$ $\{q_1\}$ $\{q_5\}$
$\{q_3\}$ $\{q_2\}$ $\{q_6, q_7\}$
$\{q_d\}$ $\{q_4\}$ $\{q_8\}$

4-equivalence: same as 3-equivalence

Final Minimization

i.e. $\{A\}$, $\{B, D\}$ $\{A, B, C\}$
$\{B\}$, $\{C\}$ $\boxed{\{B, C\}, \{B, C, D\}}$
$\{q_d\}$ $\{D\}$ $\{C, D\}$