UNIT – II

* **Regular Expressions** → Algebraic-like description to denote Regular Language.

* **Applications of Regular Expressions.**
  - UNIX.
  - 'grep' command.
  - Compiles constructions.

* **Operators of regular Expression.**
  → Operations on Languages.
  → Union of Languages – Set of strings in either L, or in M or in both.
  → Concatenation of languages.

Eg :- $L = \{0, 11\}$ $M = \{\epsilon, 00, 11\}$

$L \cdot M = \{0, 11, 000, 011, 1100, 1111\}$

→ The closure (star or Kleene closure) of a language

$L = \{0, 1\}$ L^* = {set of all strings of any length}

$L^0 = \{\epsilon\}$

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

$L^2 = \{00, 01, 10, 11\}$

$L = \{0, 11\}$ $L^2 = \{011, 110, 1111, 00\}$

$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \ldots$

* Building regular expression

Basis

Regular Exp.	Language
ε	$\{\varepsilon\}$ → The language contains empty string
ϕ	ϕ → The language is empty
a	$\{a\}$.

Induction ⇒

a) If E & F are regular expression then E+F is also a regular expression denoting the union of L(E) and L(F).

$$L(E+F) = L(E) \cup L(F)$$

b) If E & F are regular expressions then E.F or EF is also regular expression denoting the contatenation of L(E) and L(F).

$$L(E.F) = L(E).L(F)$$

c) If E is a regular expression, then E^* is also a regular expression denoting the closure of L(E)

$$L(E^*) = (L(E))^*$$

d) If E is a regular exp. then (E) is also a regular expression denoting the same language as of E.

$$L((E)) = L(E)$$

* Precendence of Operators → *, ., +.

Q. When a string can be called a regular expression?

Examples

1. Write a RE for any number of a's, $\Sigma = \{a\}$.

⇒ $R.E = a^*$, which will contain $\{\varepsilon, a, aa, aaa \ldots\}$

2. Write a RE for a or b. $\Sigma = \{a, b\}$

⇒ $R.E = a + b$

3. Write a RE for any number of a's and b's.

⇒ $R.E = (a+b)^*$, $\{\varepsilon, a, b, ab, ba, aaabbb, \ldots\}$

4. WARE for the sequence 01

$RE = 0.1$ (or) 01.

5. WARE for any 0's followed by any 1's.

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

6. Write RE for 01010101 ..... $R.E = (01)^*$

7. Write RE for 01010, 10101 ..... $R.E = (01)^* + (10)^*$

8. WARE for the sequence of a's & b's such that the last symbol is a.

$R.E = (a+b)^*a$.

9. WARE for the strings of a's and b's such that the substring ab is present

$R.E = (a+b)^* ab (a+b)^*$

10. WARE for the strings of a's & b's that ends with ab.

$R.E = (a+b)^* ab$.

11. WARE that either begins with ab or ends with abb

$R.E = (a+b)^* ab + ab(a+b)^*$

12. WARE for even number of a's

$R.E = (aa)^*$

13. WARE whose length is 2, $\Sigma = \{a, b\}$.

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

14. WARE for odd number of a's, $R.E = (aa)^*a$

15. Design a regular expression for the strings a's & b's such that string ends with abba

$(a+b)^*abba$

16. Length should be less than or equal to 2 (≤ 2)?

$R.E = \varepsilon + (a+b) + (a+b)(a+b) = \varepsilon + a + b + aa + ab + ba + bb$

$= (\varepsilon + a + b)(\varepsilon + a + b)$.

17. RE whose 4th symbol from the end is a and 2nd symbol is b

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

18. Any number a's followed by any number b's & followed by any number c's.

R.E = $a^* b^* c^*$

19. Atleast one 'a' followed by atleast one 'b' and atleast one 'c'.

$a\,a^*\,bb^*\,c\,c^*$

20. WARE that either ends with a or bb

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

21. RE that should not end with aa.

$(a+b)^*(ab+ba+bb)$

22. R.E whose length is multiple of 3 / divisible by 3/

$L = \{w / |w| \bmod 3 = 0,\ w \in \{a,b\}^*\}$.

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

23. Number of a's divisible 3 and followed by no of b's divisible by 4.

$(aaa)^*(bbbb)^*$

24. 10th symbol from the end is a.

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

25. WARE which has atleast 3 consecutive zero's

$(0+1)^* 000 (0+1)^*$

26. WARE that begin with ab and ends with ba

R.E = $ab(a+b)^* ba$

27. a's and b's whose length is either even or multiple of 3.

$R.E = \left(((a+b)(a+b))^* + ((a+b)(a+b)(a+b))^* \right)^*$

28. WARE for $L = \{ a^{2n} b^{2m} \mid n >= 0, m >= 0 \}$

$(aa)^*(bb)^*$.

29. $L = \{ a^n b^m \mid n >= 4, m <= 3 \}$

$aaaaa^* (\varepsilon + b + bb + bbb)$

30. RE for the string of a's & b's such a's are divisible by 3

R.E = $(b^* a\, b^* a\, b^* a)^*$

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

31. $L = \{ a^n b^m \mid m+n \text{ is even} \}$

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

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

32. WARE such that every block of four consecutive symbol contains atleast two a's

$((a+b)a(a+b)a + (a+b)(a+b)aa + aa(a+b)(a+b) + a(a+b)(a+b)a + (a+b)aa(a+b) + a(a+b)a(a+b))^*$ (Same thing)

33. $L = \{ a^n b^m \mid m >= 1,\ n >= 1,\ nm >= 3 \}$

(3×1) (1×3) (2×2)

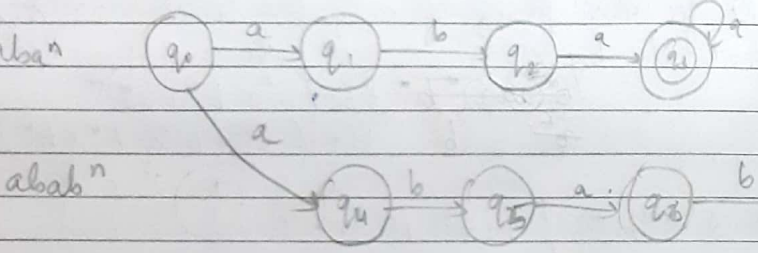
$(aa\,a)a^*bb^* + aa^*(bbb)b^* + (aa)a^*(bb)b^*$

34. R.E for the strings of a's & b's containing not more than 3 a's

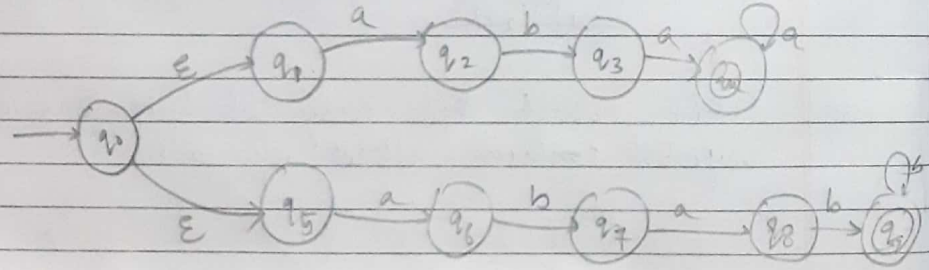
35. $L = \{ vuv \mid u, v \in \{a,b\}^* \text{ and } |v| = 2 \}$

Epsilon NFA.

Eg Construct NFA for aba^n or $abab^n$.



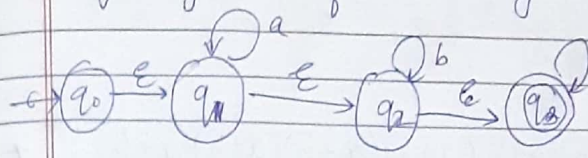
Create a new start state.



↓

Epsilon NFA

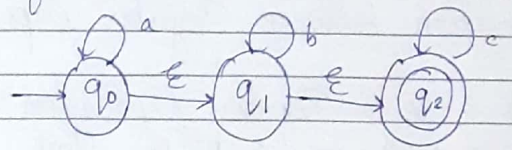
1. Construct an ε-NFA for any no of a's followed by any b's followed by any c's.



* Epsilon closure. (See ss).

* Conversion of ε-NFA to DFA

Example -



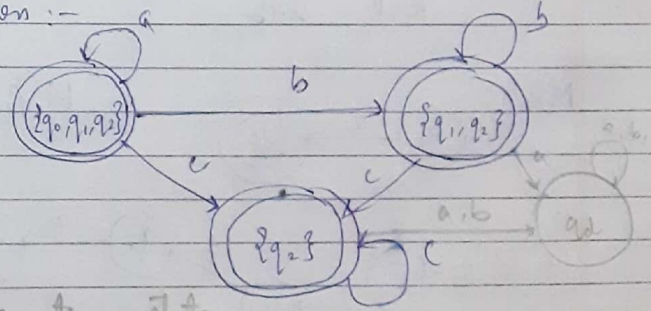
δ_{NFA}	ε	a	b	c
q_0	q_1	q_0	null	null
q_1	q_2	null	q_1	null
* q_2	null	null	null	q_2

δ_{DFA}	a	b	c
* $\{q_0,q_1,q_2\}$	$\{q_0,q_1,q_2\}$	$\{q_1,q_2\}$	$\{q_2\}$
* $\{q_1,q_2\}$	null	$\{q_1,q_2\}$	$\{q_2\}$
* $\{q_2\}$	null	null	$\{q_2\}$

$\varepsilon close(q_0) = \{q_0, q_1, q_2\}$
$\varepsilon close(q_1) = \{q_1, q_2\}$
$\varepsilon close(q_2) = \{q_2\}$

Start state of DFA is ε-CLOSE of start state of NFA

DFA construction :-



nulls are trap-state

* $L = \{ |w| \bmod 3 = 0 \text{ and } |w| \bmod 2 = 0 \text{ where } w \in (a+b)^* \}$

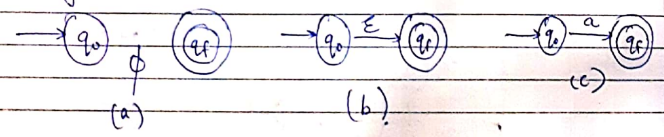
Remainders 2,0 1,0 2,1
2, 8, 4, 10, 16 5,

$(a+b)(a+b)\left((a+b)(a+b)(a+b)(a+b)(a+b)(a+b)\right)^* +$

→ Connection between Regular Expr. and Regular Languages.

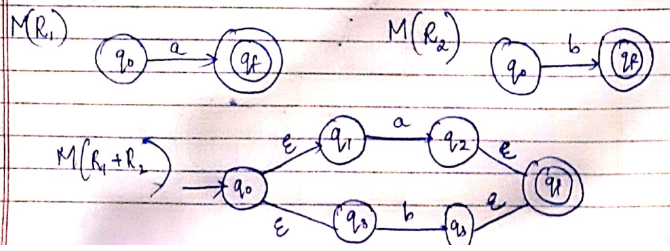
Theorem Let r be a Regular Expression Then there exists some NFA or ϵ-NFA that accepts $L(r)$. Consequently $L(r)$ is regular.

Proof: As a part of Proof we first begin the construction of Automata for preliminary Regular expression namely ϕ, ϵ, and $a \in \Sigma$



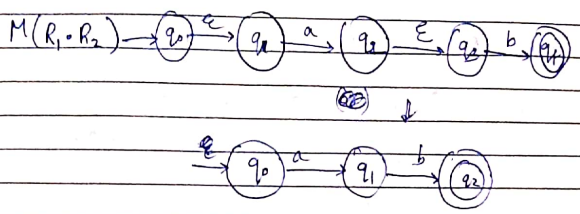
Case 1: Union $R_1 = a$ $R_2 = b$

$R_1 + R_2 = a+b$



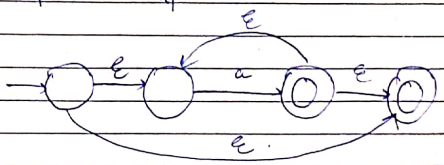
Case 2: Concatenation.

$M(R_1 \cdot R_2)$



Case 3: Closure $R_1 = a$ $R_1^* = a^*$

$M(R_1^*)$



Note: Thomson's Construction Scheme :-

Regular Exp. → ϵ-NFA → DFA → Optimized DFA → Regular Language

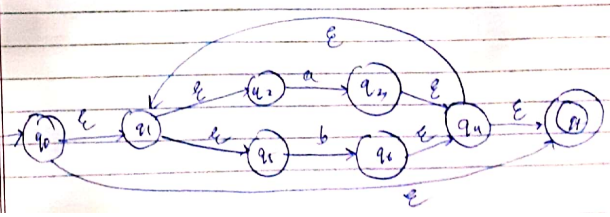
Examples:

1) $(a+b)^*$

Primitives involved :- $r_1 = a$ $r_2 = b$

$r_1 + r_2 = a+b$.

$(r_1 + r_2)^* = (a+b)^*$

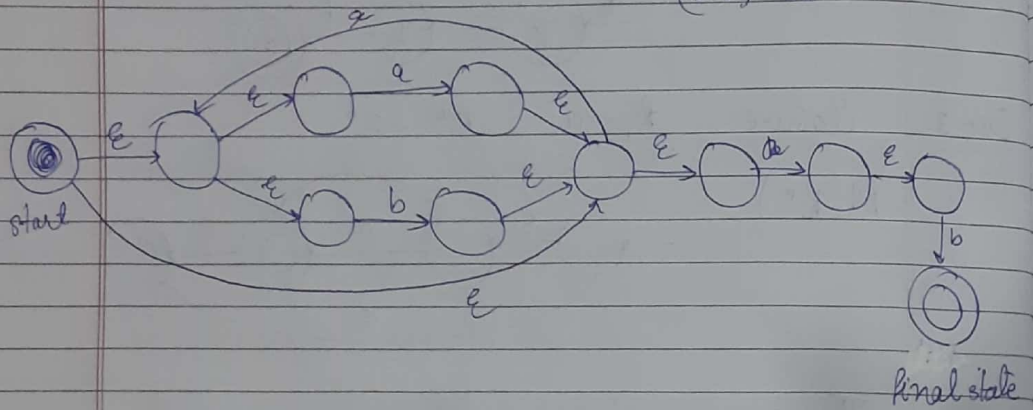


2) $(a+b)^* ab$

$r_1 = a \quad r_2 = b \quad r_3 = r_1 + r_2 = a+b$

$r_4 = r_3^* = (a+b)^* \quad r_5 = r_4 \cdot r_1 \cdot r_2$

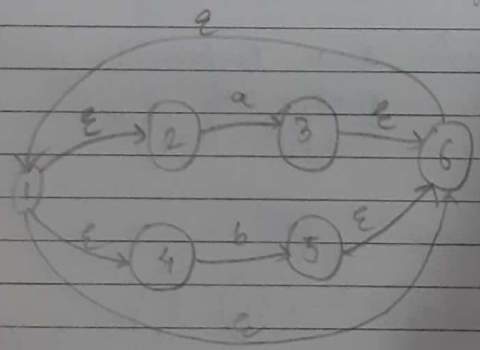
$= (a+b)^* ab$.



NOTE: Why do we require NFA?

→ There are limitations of DFA that it

* Capability of ε-NFA is Changes state (transition) without consuming any input.
* ε-NFA is more close to Regular expression



E closure(1) = {1, 2, 4, 6}

E closure(3) = {3, 6, 1, 2, 4}

E closure(5) = {6, 1, 2, 4}

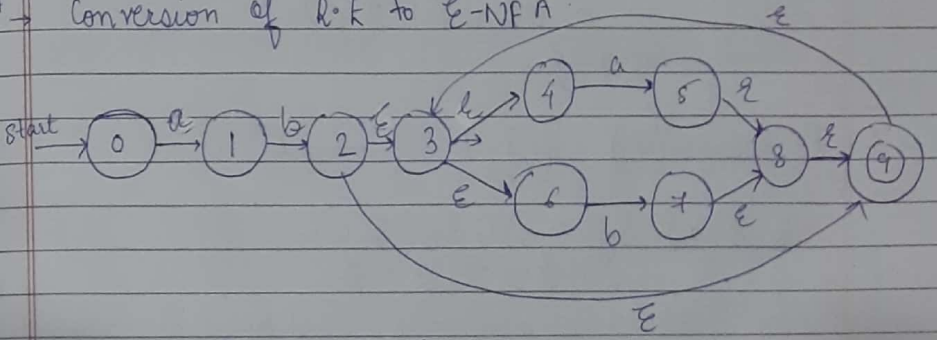
→ Purpose of ε-closure

- To eliminate all the Epsilon transition as they dont contribute anything in DFA

Eg: Conversion of ε-NFA to DFA.

* $ab(a+b)^*$

1st → Conversion of R.E to ε-NFA



2nd → conversion of ε-NFA to DFA

1. ε-closure(0) = {0} → A

$\delta_D(A, a) = \{1\}$ - Temporary state (Trap) called as trap state

$\delta_D(A, b) = \{\phi\}$.

2. ε-closure(1) = {1} → B

$\delta_D(B, a) = \delta_E(B, a) = \{\phi\}$

$\delta_D(B, b) = \delta_E(1, b) = \{2\}$ - new state

3. ε-closure(2) = {2, 3, 4, 6, 9} → C

$\delta_D(C, a) = \delta_E(2,a) \cup \delta_E(3,a) \cup \delta_E(4,a) \cup \delta_E(6,a) \cup \delta_E(9,a)$

$= \phi \cup \phi \cup \{5\} \cup \phi \cup \phi$

$= \{5\}$ → new state

$\delta_D(C, b) = \delta_E(2,b) \cup \delta_E(3,b) \cup \delta_E(4,b) \cup \delta_E(6,b) \cup \delta_E(9,b)$

$= \phi \cup \phi \cup \phi \cup \{7\} \cup \phi$

$= \{7\}$ → new state

4. ε-closure (5) = {5, 8, 9, 3, 4, 6} → D

$\delta_D(D, a) = \delta_E(5,a) \cup \delta_E(8,a) \cup \delta_E(9,a) \cup \delta_E(3,a) \cup \delta_E(4,a) \cup \delta_E(6,a)$
$= \phi \cup \phi \cup \phi \cup \phi \cup \{5\} \cup \phi$
$= \{5\}$

$\delta_D(D, b) = \delta_E(5,b) \cup \delta_E(8,b) \cdots\cdots$
$= \phi \cup \phi \cup \phi \cup \phi \cup \{7\}$
$= \{7\}$

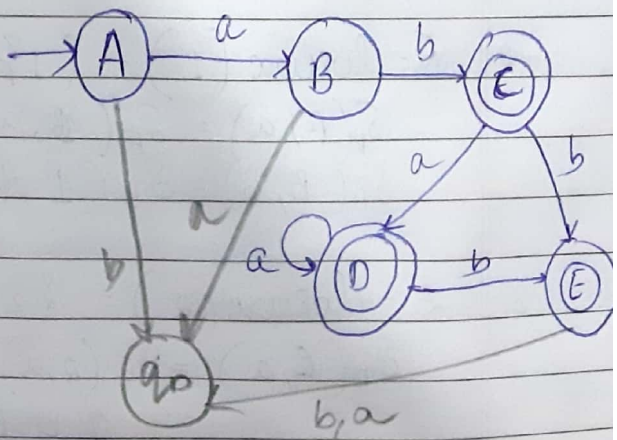
5. ε-closure (7) = {7, 8, 9, 3} ⟶ E

$\delta_D(E, a) = \delta_E(7,a) \cup \delta_E(8,a) \cdots$
$= \phi \cup \phi \cup \phi \cup \phi = \phi$

$\delta_D(E, b) = \delta_E(7,b) \cup \delta_E(8,b) \cup \delta_E(9,b) \cdots$
$= \phi \cup \phi \cup \phi \cup \phi$
$= \phi$

Table - DFA

δ	a	b
A	B	ϕ
B	ϕ	C
*C	D	E
*D	D	E
*E	ϕ	ϕ

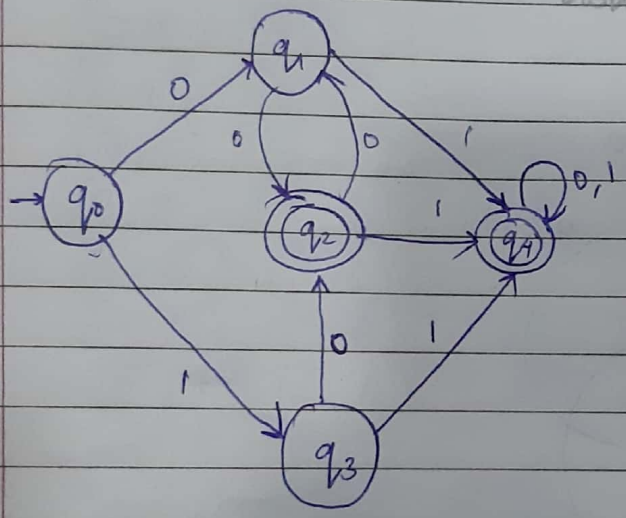


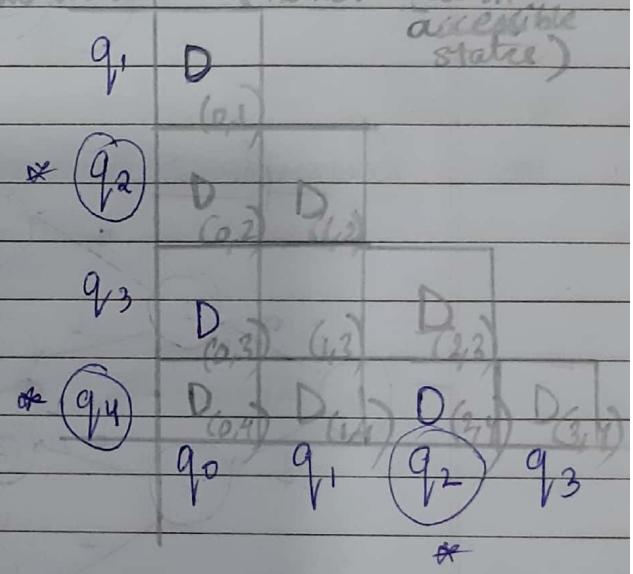
DFA Optimization

→ Optimality of DFA is to design whether the DFA with minimum state (less number of modules and the memory is used effectively).

* Example for DFA Minimization - Table Filling Algo.

Step1: Procedure mark - lower Triangular matrix to find Distinguishable pairs (Remove all inaccessible states)

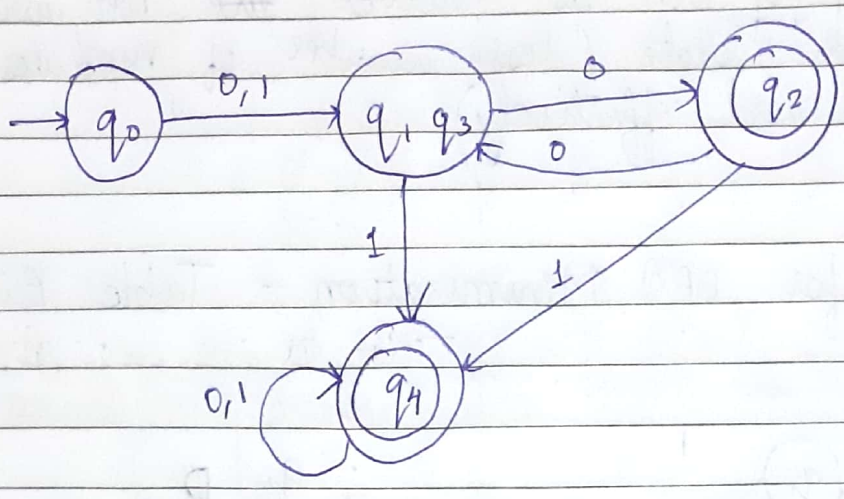




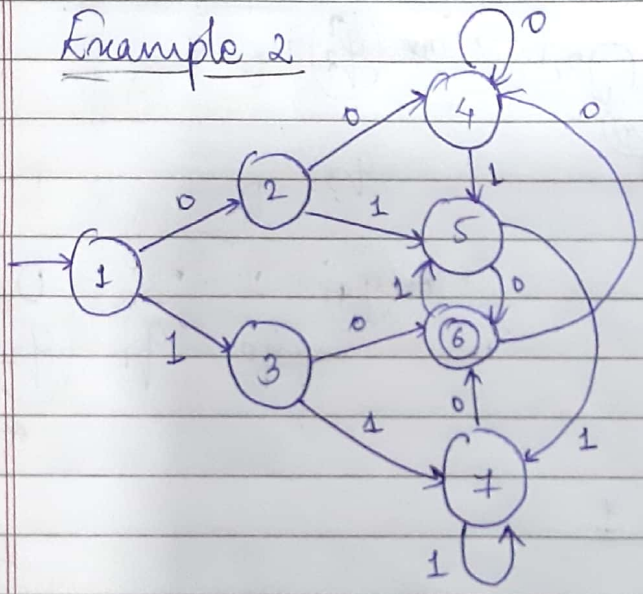
Unmarked pairs	0	1
(q_0, q_1)	(q_1, q_2)	(Since (q_1, q_2) is marked as distinguishable, we dont go for the input 1)
(q_0, q_3)	(q_1, q_2)	
(q_1, q_3)	(q_2, q_2) (q_4, q_4)	→ Not Marked as Distinguishable
(q_2, q_4)	(q_1, q_4)	

Concl: Pair (q_1, q_3) are Identical / Indistinguishable

Step 2: Minimization



Example 2



Step 1: Procedure mark(): Lower Triangular matrix

2						
3	D	D				
4			D			
5	D	D		D		
* 6	D	D	D	D	D	
7	D	D		D		D
	1	2	3	4	5	6 *

21 pairs

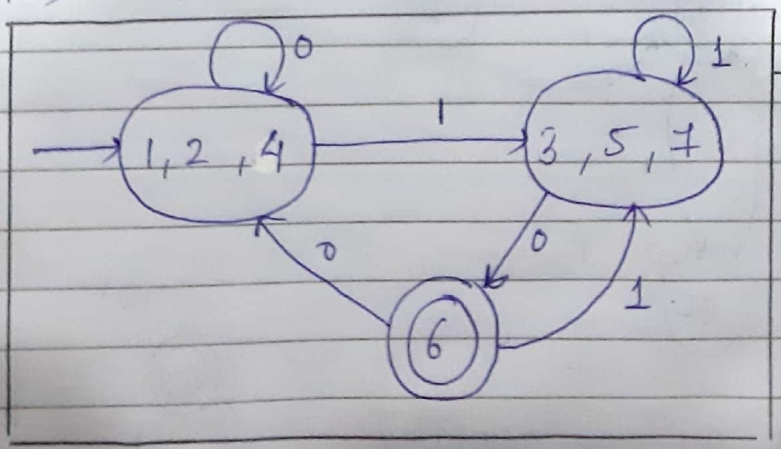
Unmarked Pairs	0	1	Remarks
(1, 2)	(2, 4)	(3, 5)	→ Not marked as Disting.
(1, 3)	(2, 6)		✓
(1, 4)	(2, 4)	(3, 5)	→ "
(1, 5)	(2, 6)		✓
(1, 7)	(2, 6)		✓
(2, 3)	(4, 6)		✓
(2, 4)	(4, 4)	(5, 5)	→ Cannot be marked as Dist.
(2, 5)	(4, 6)		✓
(2, 7)	(4, 6)		✓
(3, 4)	(6, 4)		→ (6,4) or (4,6) it's the same
(3, 5)	(6, 6)	(7, 7)	→ Cannot mark
(3, 7)	(6, 6)	(7, 7)	"
(4, 5)	(4, 6)		✓
(4, 7)	(4, 6)		✓
(5, 7)	(6, 6)	(7, 7)	→ Cannot mark.

Step2: Minimize the states

(1,2) (1,4) (2,4) → 1, 2, 4 are identical using equivalence prop.

Unmarked states

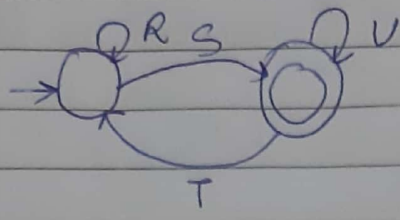
(3,5) (5,7) (3,7) → 3, 5, 7



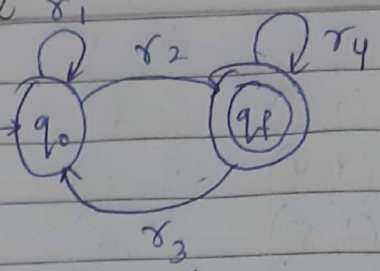
→ Final Optimal DFA.

Note:- If DFA D_1, D_2 and D_3 are DFAs for the same language then the minimized DFA Dm is same for all.

State Elimination Method



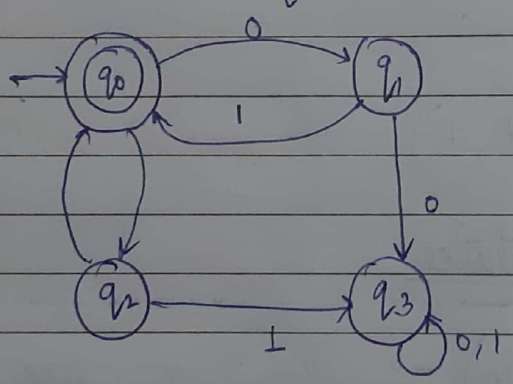
or



$R^* S(U+TR^*S)^*$ $\qquad r = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$

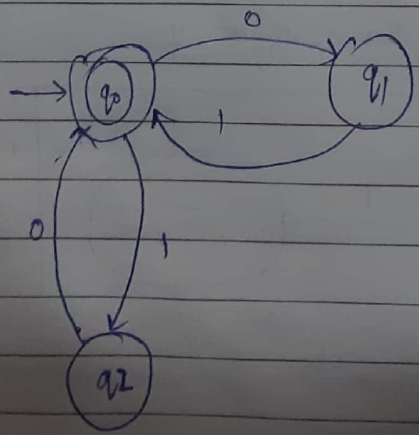
1. If r_3 is not there $\quad r = r_1^* r_2 r_4^*$
2. If q_0, q_1 are final state, $r = r_1^* + r_1^* r_2 r_4^*$

Ex 1] Obtain a RE for the FA.

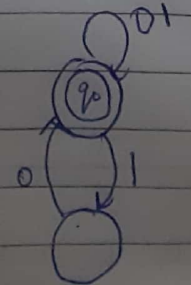


q_0 → is the start & final state
q_1, q_2 → Intermediate state
q_3 → Dead state

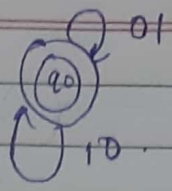
Step 1: Take out the dead state 'q_3'



Step 2: Process 01 is repeated

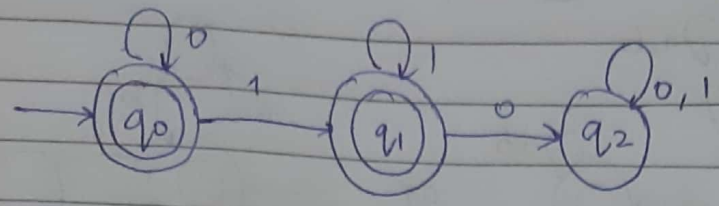


Step 3: Process 10 is repeated

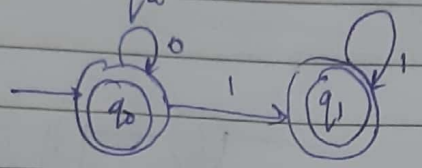


$RE = (01+10)^*$

2]



Step 1: Eliminate q_2

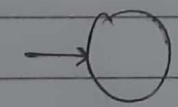


Since q_0 is the final state 0^*

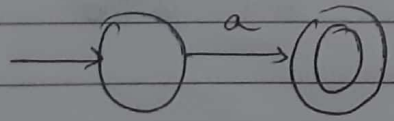
to reach q_1 0^*1 1^*

$$RE = 0^* + 0^*1\,1^*$$
$$= 0^*(\varepsilon + 11^*)$$
$$= 0^*(\varepsilon + 1^+)$$
$$= 0^*1^*$$

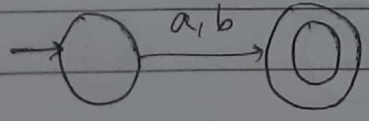
ϕ



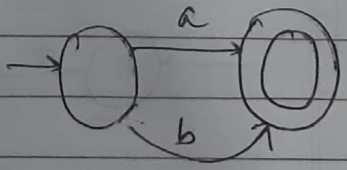
a



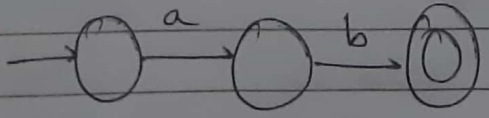
a+b



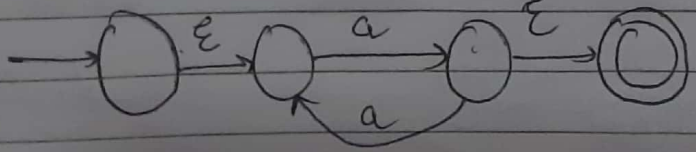
or



ab



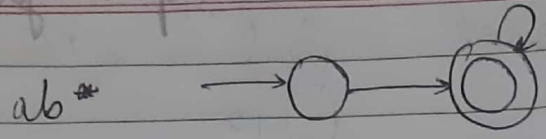
a^*

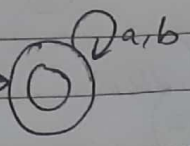


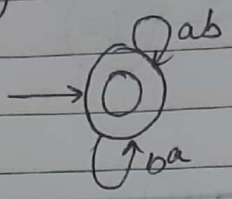
Parallel edges = Union

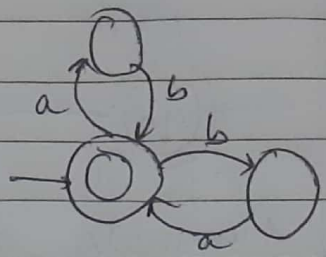
Sequential " = Concatenation

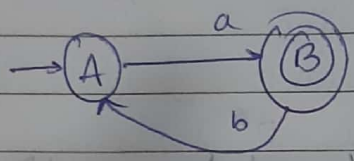
Self Loop = *

ab^* 

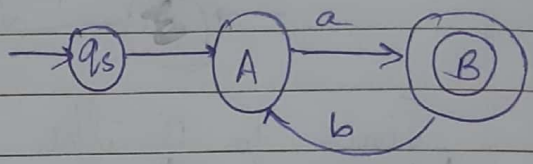
$(ab)^*$ 

$(ab + ba)^*$ 

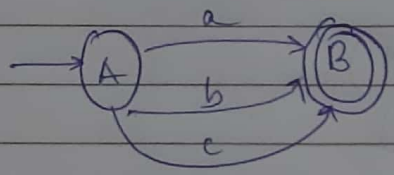
or 

1. 

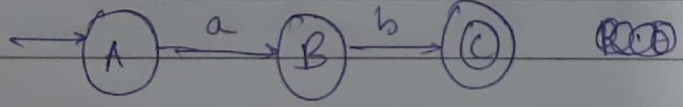
Start state should not have incoming edge.

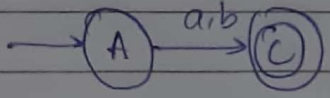


Create new state with epsilon move.

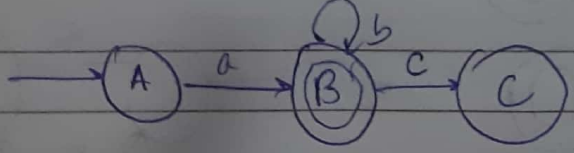
3] 

R.E = a + b + c

4] 



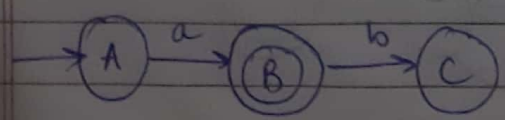
R.E = a + b.

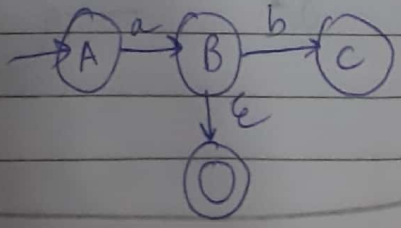
5] 

R.E = ab^*

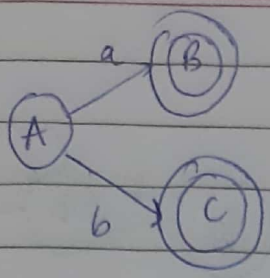
2. Final state should not have outgoing edge

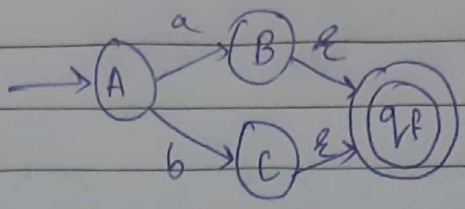
Create new final state with epsilon move





3.

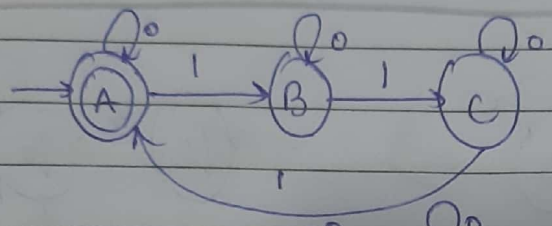


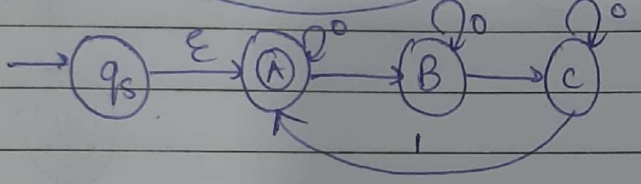


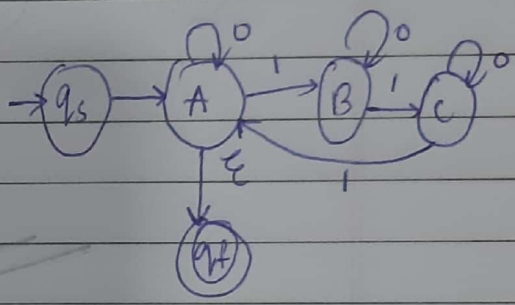
There should not be more number of final state

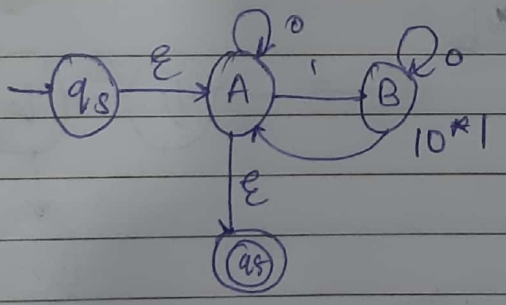
The existing final state is converted to non-final state a new final state is created with epsilon move

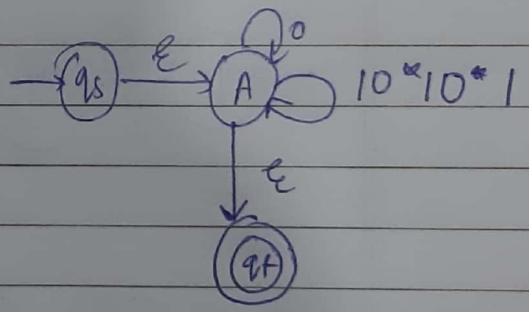
6]







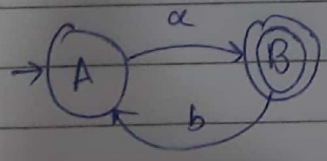


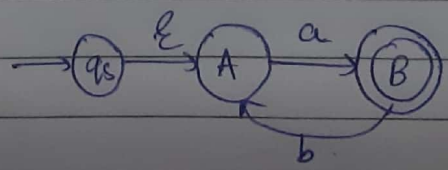


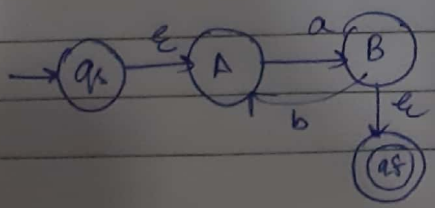
$R \cdot E = 0^* + (10^* 10^* 1)^*$

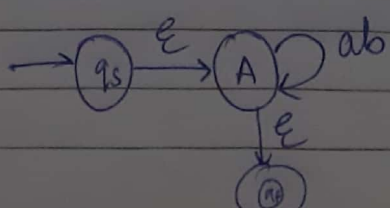
$= (0 + 10^* 10^* 1)^*$

7]



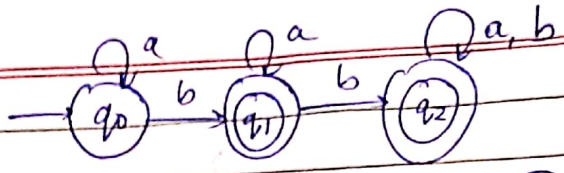


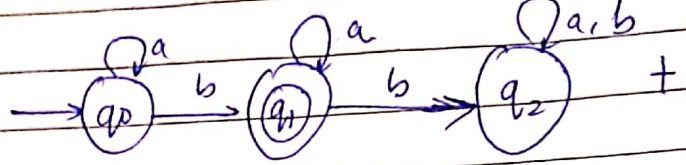




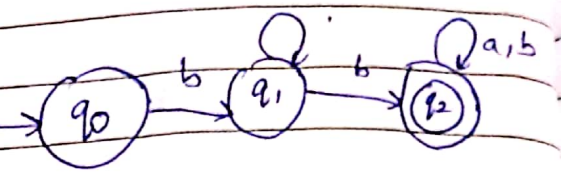
$R \cdot E = (ab)^*$

8]



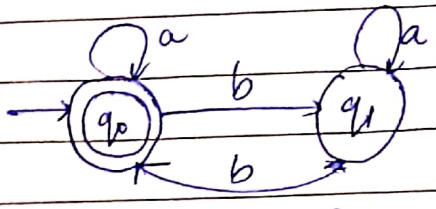


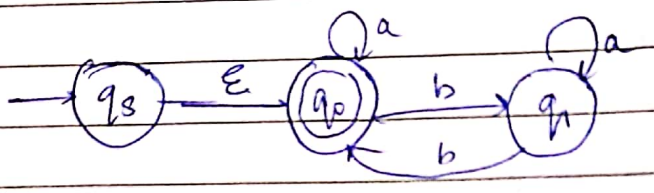
+



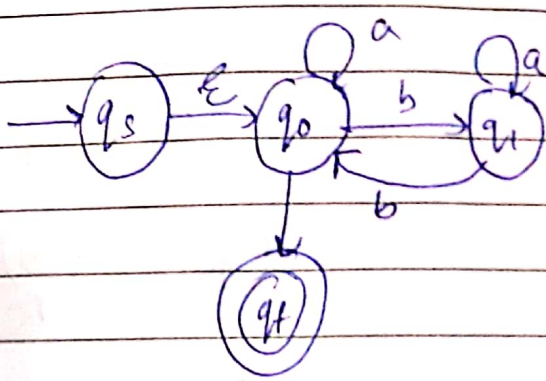
$a^* ba^*$ + $a^* ba^* b(a+b)^*$

9]

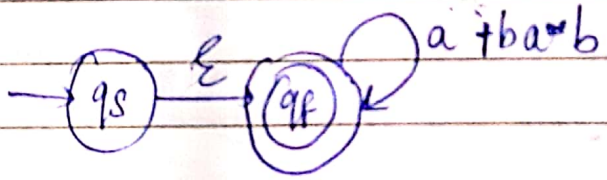




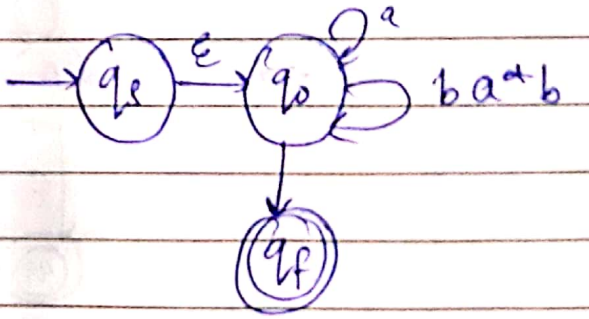
⇒



⇒



⇐



$R.E = (a + ba^* b)^*$