M T W T F S S
Page No.:
Date:
YOUVA

Regular Expression :→

A Language is called as regular language if some finite automaton (DFA/NFA etc) recognizes it.

In basic arithmetic the basic objects are numbers and tools are operations for manipulating them, such as + & ×. In Theory of computation objects are languages and the tools include operations specifically designed for manipulating them. We define three operations on language called Regular Operations. and use them to study properties of regular languages.

Let A & B be languages, we define regular operations union, concatenation and star as follows

① Union: $A \cup B := \{x \mid x \in A \text{ or } x \in B\}$

② concatenation := $A \circ B = \{xy \mid x \in A \text{ & } y \in B\}$

③ star := $A^* = \{x_1, x_2 \ldots x_k \mid x_k \geq 0 \text{ and each } x_i \in A\}$

(See previous page)

We are already familiar with union operation. It simply takes all string of A & B and lumps-up them together into one language.

The concatenation operation is little trickier. It attaches string from A in front of string B in all possible ways. to get strings in new langage.

Star operation applies to single language. so star operation is unary operation. It attaches any number of strings in A together.

Eg: Let the alphabet Σ be the standard 26 letters $\{a, b, c, d \ldots z\}$. If $A = \{good, bad\}$ and $B = \{boy, girl\}$ then.

$A \cup B = \{good, bad, boy, girl\}$

$A \circ B = \{goodboy, goodgirl, badboy, badgirl\}$

$A^* = \{\varepsilon, good, bad, goodgood, goodbad, badbad, goodbadgood \ldots\}$

Regular operators :-

Basically there are three operators that are used to generate languages that are regular.

① Union: (U / | / +): If L1 and L2 are any two regular languages then

$L1 \cup L2 = \{ s \mid s \in L1 \text{ or } s \in L2 \}$

For example

$L1 = \{ 00, 11 \}$

$L2 = \{ \epsilon, 10 \}$

$L1 \cup L2 = \{ \epsilon, 00, 11, 10 \}$

② Concatenation (.): If L1 and L2 are any two regular languages then

$L1 . L2 = \{ l1 . l2 \mid l1 \in L1 \text{ and } l2 \in L2 \}$

for example: $L1 = \{00, 11\}$ and $L2 = \{\epsilon, 10\}$

$L1 . L2 = \{00, 11, 0010, 1110\}$

$L2 . L1 = \{1000, 1011, 00, 11\}$

so $L1 . L2 \neq L2 . L1$

③ Kleen closure: (*)

If L is any regular language then

$L^* = L_0 \cup L_1 \cup L_2 \cup L_3 \cup \ldots\ldots$

Precedence of operation

① * is highest precedence

② concatenation is second precedence.

③ finally union

The value of regular expression is language.

Eg: ~~(0 ∪ 1)0*~~
(0 ∪ 1) 0*

In above example value of language is consisting of all strings starting with a '0' or a '1' followed by any number of 0.

Defination of Regular Expressions:

Regular expression is a method to represent a ~~language~~ regular language.

Say that R is regular expression if R is
1. a for some a in the alphabet Σ
2. ε is regular expression denoting set {ε}
3. ϕ is ~~set~~ {}
4. (R1 ∪ R2), where R1 & R2 are regular expressions.
5. (R1 ∘ R2) where R1 and R2 are regular expressions
6. (R1*) where R1 is regular expression

In the items 1 and 2 are regular expressions a and ε represents the languages {a} and {ε} respectively. In the item 3, the regular expression ϕ represents the ~~languages obtained~~ empty language. 4, 5, 6: represents languages obtained by taking the union or concatenation of language R1 and R2 or the star (*) of R1 respectively.

Note: Don't confuse the regular expressions ε & ϕ. The expression ε represents the language containing single string - namely the empty/NULL string. Where as ϕ represents the language that does not contain any string.

Example: Assume that alphabet $\Sigma = \{0, 1\}$

1. $0^*10^* = \{w \mid w$ containing single $1\}$
2. $\Sigma^*1\Sigma^* = \{w \mid w$ has atleast one $1\}$
3. $\Sigma^*001\Sigma^* = \{w \mid w$ contains the string 001 as substring$\}$
4. $1^*(01)^* = \{w \mid$ every 0 in w followed by atleast one 1.$\}$
5. $(\Sigma\Sigma)^* = \{w \mid w$ is a string of even length$\}$
6. $(\Sigma\Sigma\Sigma)^* = \{w \mid$ the length of w is multiple of three.$\}$
7. $01 \cup 10 = \{01, 10\}$

Example; Write a RE for the set of string that consists of alternate 0's and 1's over $\{0, 1\}$.

⇒ first part: We have to generate the language : $\{01, 0101, 010101 \ldots\}$

second part: We have to generate the language : $\{10, 1010, 101010 \ldots\}$.

So let's start first part: Here we start with basic regular expression 0 and 1 that represent the language $\{0\}$ and $\{1\}$ respectively.

Now if we concenate these two RE, we get the RE 01 that represent the language $\{01\}$. Then to generate the language of zero or more occurances of 01, we take kleen closure i.e. the RE $(01)^*$ represents the language $\{01, 0101 \ldots\}$

Similarly, the RE for second part is $(10)^*$.

Now finally we take union of above two: first part and second part to get required RE i.e. the RE $(01)^* + (10)^*$ represents the given language.

Ex:

1. $R = \phi$, $L(R) = \{\}$
$R = \epsilon$, $L(R) = \{\epsilon\}$
$R = a$, $L(R) = \{a\}$
$R = a+b$ $L(R) = \{a, b\}$
$R = a.b$ $L(R) = \{ab\}$
$R = a+b+c$ $L(R) = \{a, b, c\}$
$R = (ab+a).b$ $L(R) = \{abb, ab\}$
$R = a+$ $L(R) = \{a, aa, aaa \dots\}$
$R = a^*$ $L(R) = \{\epsilon, a, aa, aaa \dots\}$
$R = (a+ba).(b+a)$ $L(R) = \{ab, aa, bab, baa\}$
$R = (a+\epsilon)(b+\phi)$ $L(R) = \{ab, b\}$ $[\because a+\phi = \phi \quad a+\epsilon = a, \epsilon]$
$R = (a+b)^2$ $L(R) =$
$\hookrightarrow (a+b).(a+b) = \{aa, ab, ba, bb\}$
$R = (a+b)^* . L(R)$
we will write as
$(a+b)^0 = \{\epsilon\}$
$(a+b)^1 = \{a, b\}$
$(a+b)^2 = \{aa, ab, ba, bb\}$
$\vdots$
$(a+b)^n = \{a^n, ab^n, ba^n, bb^n\}$

So $L(R) = \{\epsilon, a, b, aa, ab, ba, bb, aaa, ababab, bababa \dots\}$

$R = (a+b)^* . (a+b)$
Here for $(a+b)^*$ we will get minimum string as ϵ
so $\{\epsilon, a, aa, b, bb \dots\} . (a+b)$.
$\{a, b, a, aa, b, bb \dots\}$ we can call this as $(a+b)^+$
so $R = (a+b)^* . (a+b)$, $L(R) = (a+b)^+$

$R = a^* . a^*$. $L(R) = \{\epsilon, a, aa, aaa \dots\} . \{\epsilon, a, aa, aaa \dots\}$
$= \{\epsilon, a, aa, aaa \dots a^*\}$

$R = (ab)^*$: $L(R) = \{\epsilon, ab, abab, ababab \dots\}$

$\epsilon = \lambda$

Both are used to denote empty string.

Regular set:

Definition: A regular set is a set that can be generated starting from the empty set, empty string, and single element from the alphabet using concatenation, union and kleen closures in arbitary order.

A regular set is a set represented by a regular expression.

- The regular set defined by regular expression 01^* is a set of string starting with 0 & followed by 0 or more 1's.
- The regular ~~expression~~ set defined by $(10)^*$ is set of strings containing 0 or more copies of 10.
- A regular set defined by $(0 \cup 1)\,1\,(0 \cup 1)$ is the set of string $\{01, 011, 110, 111\}$

$\lambda = \epsilon$

Identities of Regular Expression

1) $\phi + R = R$.
2) $\phi R + R\phi = \phi$
3) $\epsilon R = R\epsilon = R$
4) $\epsilon^* = \epsilon$ and $\phi^* = \epsilon$
5) $R + R = R$
6) $R^* R^* = R$
7) $R R^* = R^* R$
8) $(R^*)^* = R^*$
9) $\epsilon + RR^* = \epsilon + R^*R = R^*$
10) $(PQ)^* P = P(QP)^*$
11) $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$
12) $(P+Q)R = PR + QR$ and $R(P+Q) = RP + RQ$
13. $\epsilon \cdot R^* = R^*$

Write Regular Expression for given language. (Before studying Regular sets)

① Language accepting strings of length exactly 2 $\Sigma(a,b)$

$L_1 = \{aa, ab, ba, bb\}$

$R = aa + ab + ba + bb$

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

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

M T W T F S S | Page No.: | Date: | YOUVA

② Language accepting string of length atleast 2.

$L_1 = \{aa, ab, ba, bb, aaa \ldots\}$

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

③ Language accepting string of length atmost 2.

$L_1 = \{\epsilon, a, b, aa, ab, ba, bb\}$

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

$= (\epsilon + a + b)(\epsilon + a + b)$

Equivalence of Regular Expression

①

Prove that $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$ is equal to $0^*1(0+10^*1)^*$

LHS $= (1+00^*1) + (1+00^*1)(0+10+1)^*(0+10^*1)$

$= (1+00^*1)[\epsilon + (0+10^*1)^*(0+10^*1)]$ $\epsilon + R^*R = R^*$

$= (1+00^*1)(\epsilon + (0+10^*1)^*)$ $\epsilon + R^*R = R^*$

$= (\epsilon \cdot 1 + 00^*1)$

LHS $= (1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$

$= (1+00^*1)[\epsilon + (0+10^*1)^*(0+10^*1)]$ $\epsilon + R^*R = R^*$

$= (1+00^*1)(0+10^*1)^*$

$= (\epsilon \cdot 1 + 00^*1)(0+10^*1)^*$ Rule: $\epsilon R = R$

$= (\epsilon + 00^*)1(0+10^*1)^*$

$= 0^*1(0+10^*1)^* = RHS$ Rule $\epsilon + RR^* = R^*$

② $((00^*01)^*0 + 0^*(10)^*)^* = (0+10)^*$

$(ab)^*a = a(ba)^*$

$= (00^*(01)^*0 + 0^*(10)^*)^*$

$= (00^*0(10)^* + 0^*(10)^*)^*$

$= (00^*0 + 0^*)(10)^*)^*$

Whatever is going to generated by 00^*0 is equal to 0^*

So, $= ((0^*(10)^*)^*$ $\Rightarrow (P+Q)^* = (P^*+Q^*)^*$

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

Q.1 ⇒ 10 M

Even:

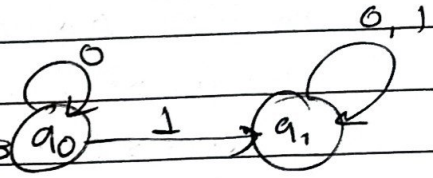
Prove that $10 + (1010)^* [\epsilon + (1010)^*] = 10 + (1010)^*$

Conversion of NFA with epsilon (ϵ) moves to DFA.

Let us understand ϵ NFA.

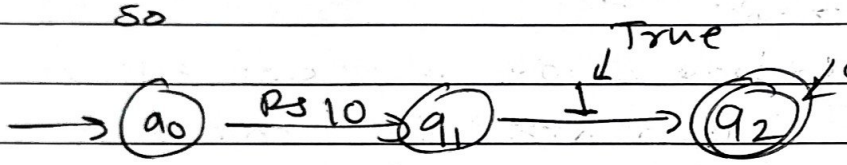
Ex. if we have Σ (0, 1) and given transition are

	0	1
q_0	q_0	q_1
q_1	q_1	q_1

Then FA will be 

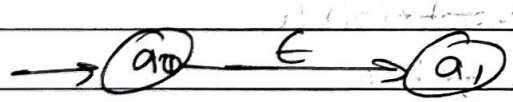
Now let us consider an example of automatic vending machine, where machine accepts a 10 Rs coin to dispense the product.

So



But suppose that user inserts the coins two coins of Rs 5, in that case where what will be next state. Here m machine is confused about the next state i.e. machine can keep the same state or it will change the state.

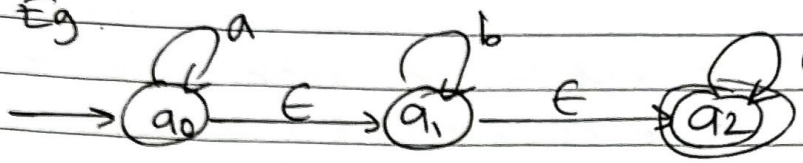
ϵ allows us to change the state without consuming input symbol.



Defn of ϵ NFA

Q - finite set of state.
Σ - input symbols
q_0 - initial state.
f - set of final state.
δ - Transition function
$\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$

Eg



Does above machine will accept 'a'. → Yes (a ϵ ϵ)

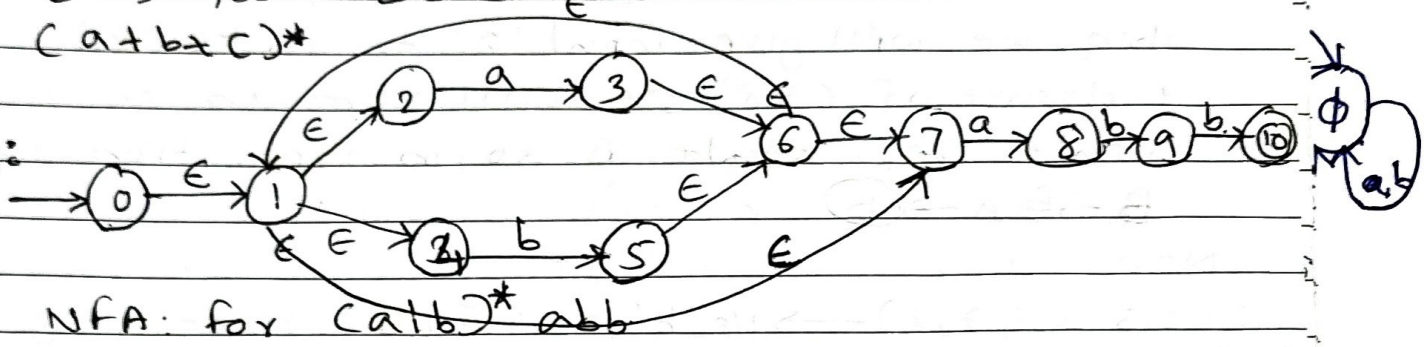
ϵ → yes ϵ ϵ

b → yes → ϵ b ϵ

c → yes - ϵ ϵ c

(a+b+c)*

Eg:



NFA for (a+b)* abb

We will start with initial state i.e. 0 and will find the ϵ* closure of initial state i.e. state 0.

ϵ* closure: Is a set of state (including current state) where we can go (transition) from current state without consuming input symbol. i.e. with only ϵ.

So,

Step 1.

ϵ closure (0) = (0, 1, 2, 4, 7).

This is initial state, we will call above ϵ-closure of (0) as DFA state 'A' and insert into transition table.

Now we will check the transitions on above ϵ closure (0) with i/p symbol a & b

⇒ i/p a to ϵ closure (0) we get → (3, 8), we will name this state as B.

so when we give i/p a to ϵ-closure (0) we reach to state B.

Now we will give i/p b to ϵ-closure (0)

(5) → we will name this a state C.

Transition Table:

DFA state	a	b
A	B	C
B	B	D
C	B	E
D	B	E
E	B	C

Step 2:

Now again we will find ϵ closure of (3, 8) and (5)

ϵ closure (3, 8) = (3, 8, 6, 7, 1, 2, 4) = (1, 2, 3, 4, 6, 7, 8)

ϵ closure (5) = (5, 6, 7, 1, 2, 4) = (1, 2, 4, 5, 6, 7)

Now we will check i/p a & b on both ε closure i.e. ε (3,8) & ε closure (3,8) and ε(6-(5).

so

(1,2,3,4,6,7,8) → i/p 'a' = (3,8)

Note that here on i/p a ε-closure (3,8) we got the result as (3,8) i.e. state is repeated means when we will give input 'a' to state (B) i.e. ε closure of (3,8) we will remain on same state i.e. state B so, in transition table

B —i/p a→ (B)

Now

(1,2,3,4,6,7,8) → i/p 'b' = (5,9). Here we got new state (i.e. not repeated so we will name this as the state D. make entry in transition table.

Step: 3.

Now find ε-closure of (5,9) i.e. state C

ε-closure (5,9) = (5,9,6,7,1,2,4)
= (1,2,4,5,6,7,9)

find transition for input symbol a & b on ε closure (5,9).

So,

(1,2,4,5,6,7,9) —i/p→ a = (3,8) = B

(1,2,4,5,6,7,9) —i/p→ b = (5,10) = D New state, name this as E enter in transition table

Step 4

find ε closure of (5,10)

ε-closure (5,10) = (5,10,6,7,1,2,4,)
= (1,2,4,5,6,7,10)

· find transition on i/p symbol a & b on ε-closure (5,10)

Even:

M T W T F S S
Page No.:
Date:
YOUVA

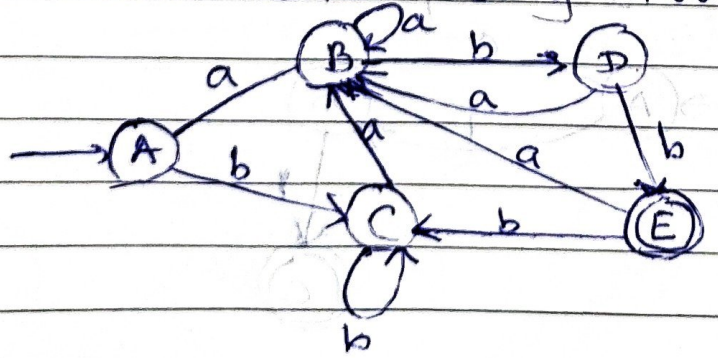
(1,2,4,5,6,7,10) i/p a = (3,8) = B
(1,2,4,5,6,7,10) i/p b = (5,

Step 3:
check i/p symbol 'a' & b for ϵ-closure (i.e. (5)

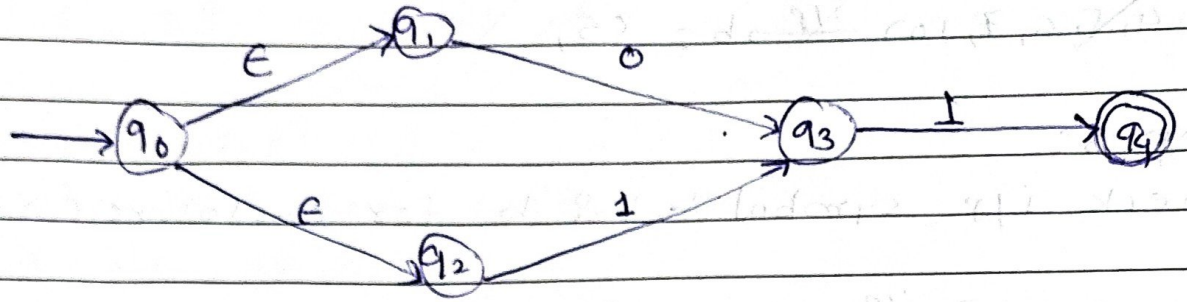
(1,2,4,5,6,7) i/p a = (3,8) = B
(1,2,4,5,6,7) i/p b = (5) = C

Step 4.
Check i/p symbol 'a' & b for ϵ closure D - i.e (5,9

ϵ-closure (5,9) = (5,9,6,7,1,2,4)
= (1,2,4,5,6,7,9)
find transition for i/p symbol a & b on
ϵ clouser (5,9) i.e. (1,2,4,5,6,7,9)
so
(1,2,4,5,6,7,9) i/p a = (3,8) = B
(1,2,4,5,6,7,9) i/p b = (5,10) = E (New state)
Step:5
check i/p symbol 'a' & b for ϵ-closure E i.e. (5,10)
ϵ-closure (5,10) = (5,10,8,7,1,2,4)
= (1,2,4,5,6,7,10)
find transition on for i/p symbol a & b on ϵ-clo (5,10)
(1,2,4,5,6,7,10) i/p a = (3,8) = B
(1,2,4,5,6,7,10) i/p b = (5) = C
Here we don't have any new state, so will stop here
Draw DFA using transition table



Example 2:



Here q_0 is initial state

Find ϵ closure of q_0

ϵ - closure $(q_0) = (q_0, q_1, q_2)$ – A – state A

Find the states where we can reach using input (0, 1) on ϵ closure (q_0) / (q_1, q_2) / A.

~~$\delta(A, 0) =$~~

$\delta(q_1, q_2(0)) = (q_3) = B$ – state B.

$\delta(q_1, q_2(1)) = (q_3) = B$

Now find ϵ closure (q_3) / B

ϵ closure $(q_3) = \phi(q_3)$

find the states where we can reach using input (0, 1) on ϵ - closure (q_3) or B.

$\delta(q_3, 0) = \phi$

$\delta(q_3, 1) = q_4$ – C – state C

Now find ϵ closure of q_4

ϵ closure ~~of~~ $(q_4) = \phi(q_4)$

Find the states where we can reach using i/p symbol (0, 1) on ϵ - closure (q_4) / C

$\delta(q_4, 0) = \phi$

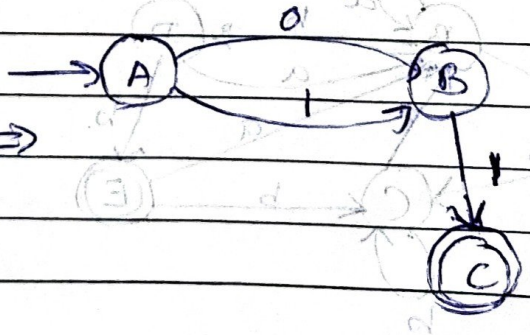
$\delta(q_4, 1) = \phi$

No new state so we stop Here

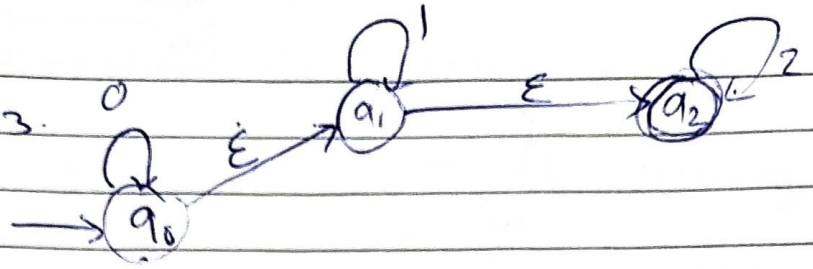
Transition Table

State	0	1
A	B	B
B	ϕ	C
C	ϕ	ϕ

$\Rightarrow$ DFA $\Rightarrow$



Ex-3.



Step1: find ε closure of q_0

ε-clo(q_0) = q_0, q_1, q_2 ⇒ A

Check i/p symbol 0, 1, 2 transition on A

$\delta(A, 0) = q_0$

$\delta(A, 1) = q_1 \Rightarrow B$

$\delta(A, 2) = q_2 \Rightarrow C$

State	i/p 0	1	2
A	A	B	C
B	ϕ	B	C
C	ϕ	ϕ	C

Transition table

find ε closure of state B i.e. q_1.

ε closure(q_1) = q_1, q_2

check i/p symbol 0, 1, 2 transition on B.

$\delta(B, 0) = \phi$

$\delta(B, 1) = q_1 \Rightarrow B$

$\delta(B, 2) = q_2 \Rightarrow C$

find ε closure of C i.e. q_2

ε closure(q_2) = (q_2)

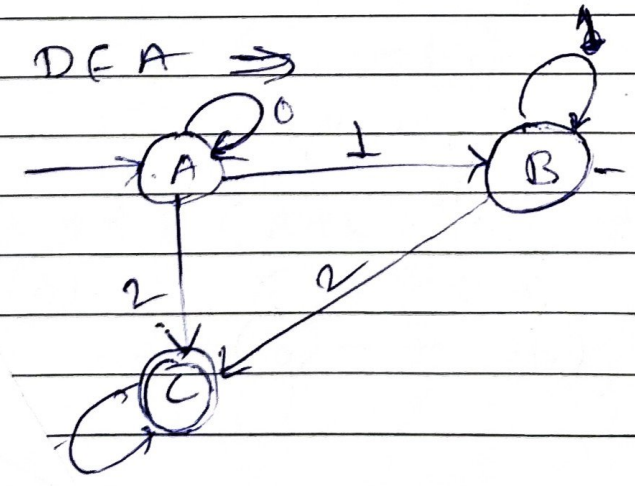
Check i/p symbol 0, 1, 2 transion on q_2

$\delta(q_2, 0) = \phi$

$\delta(q_2, 1) = \phi$

$\delta(q_2, 2) = q_2 = C$

DFA ⇒



M T W T F S S

Page No.: YOUVA

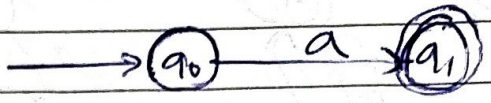
Date:

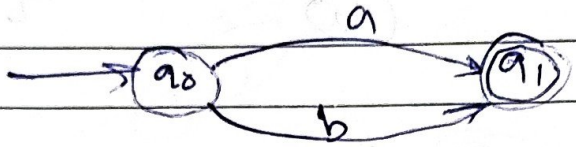
Regular Expression To DFA.

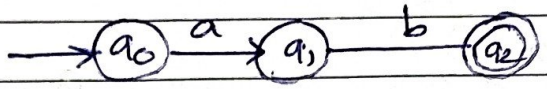
(1) ϕ 

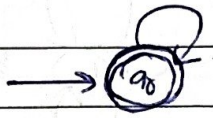
Nothing will accept i.e. ϕ

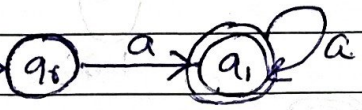
(2) ϵ 

(3) a 

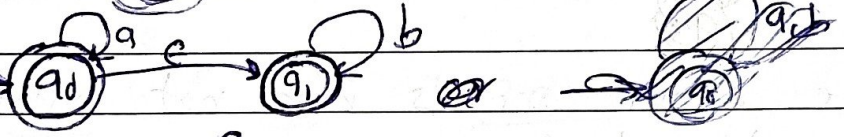
(4) a+b 

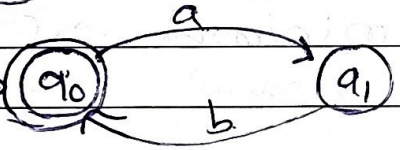
(5) a.b 

(6) a^* 

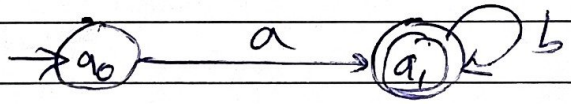
(7) a+ 

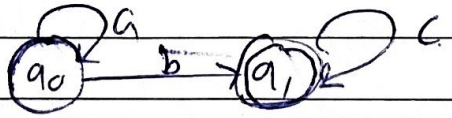
(8) $(a+b)^*$ 

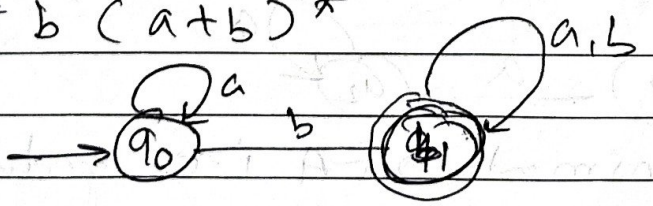
(9) $a^* b^*$ 

(10) $(ab)^*$ 

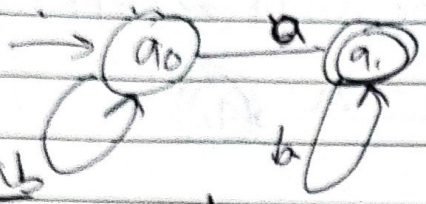
(11) $a^* b$ 

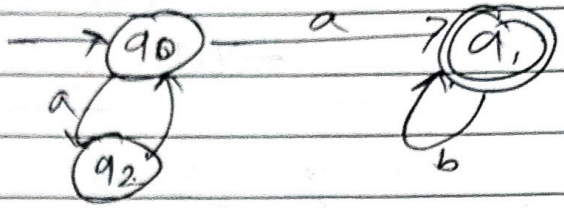
(12) ab^* 

(13) $a^* b c^*$ 

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

(15) $(ab)^*$ ab^*



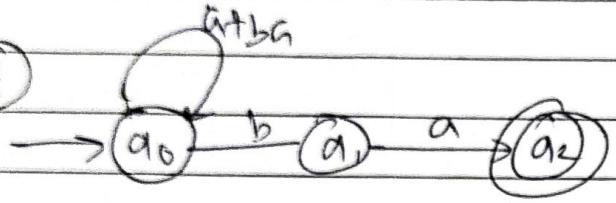


✗ ab

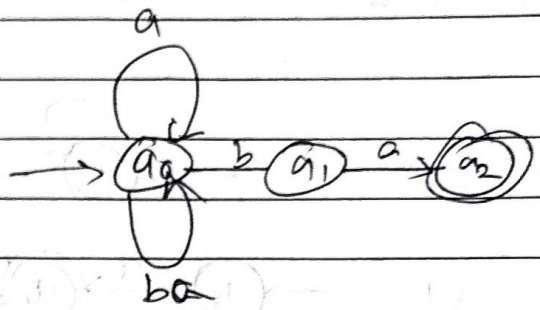
Not allowed. on single line there should single transition

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

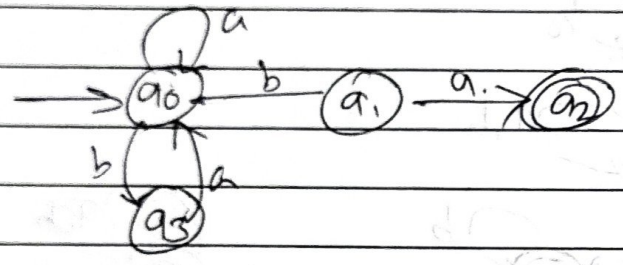
(i)



(ii)



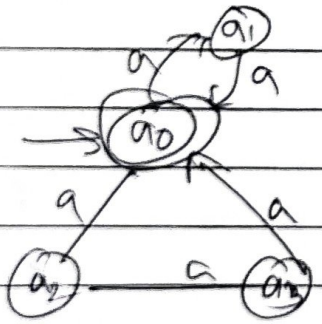
(iii)



The above DFA's are not minimal DFA Try to design minimal DFA's

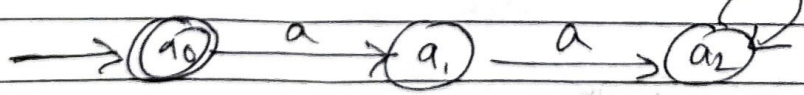
(17)

$(aaa + aaaa)^*$



Write language of above example.

$L = \{\epsilon, aa, aaa, aaaa, aaaaa, aaaaaa, \ldots\ldots\}$



This is minimal DFA i.e. reduced the states

M T W T F S S

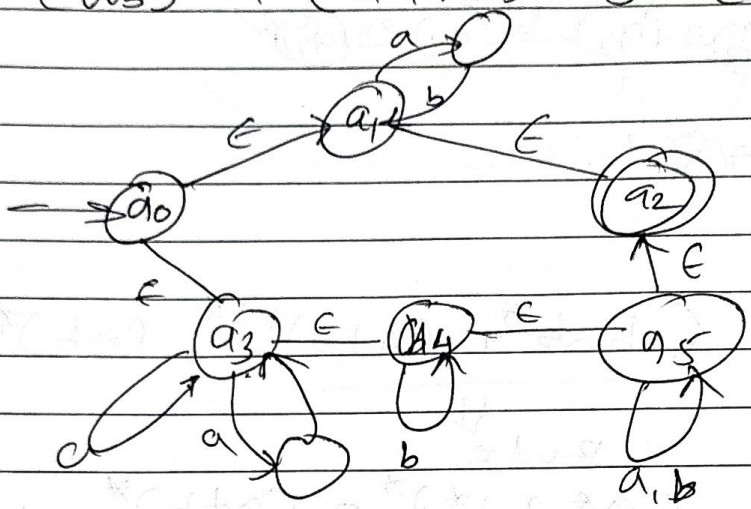
Page No.: YOUVA

Date:

(18) $(a + aaaa)^*$

$L = \{\epsilon, a, aa, aaa, aaaa, aaaaa, \ldots\}$

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



⇒ This is not minimal DFA. GATE question may asked that state the minimum states in DFA.

After minimizing you will get

because we have

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

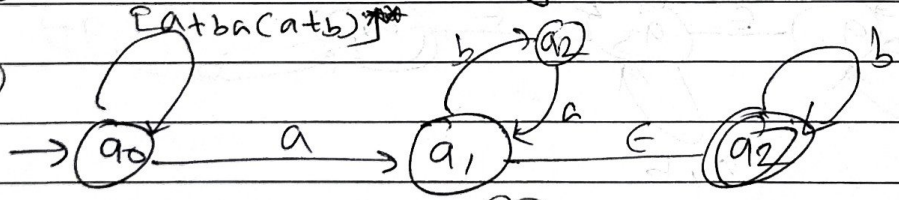
This is choice so, we can ignore i.e. ε

ε ignore

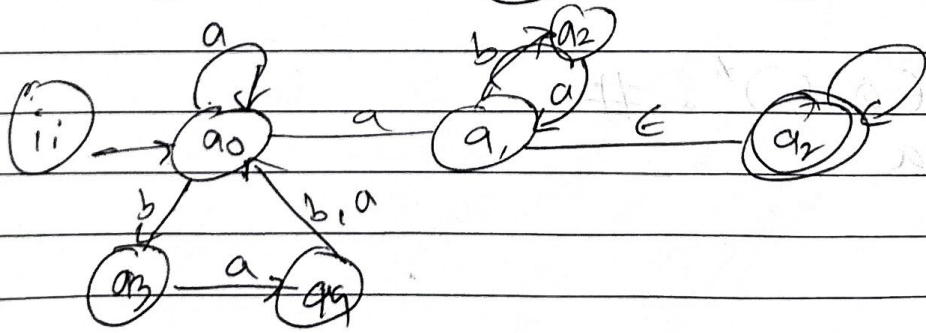
$(a+b)^*$ This is universal language so any string can be accepted.

(20) $[a + ba(a+b)]^* a(ba)^* b^*$

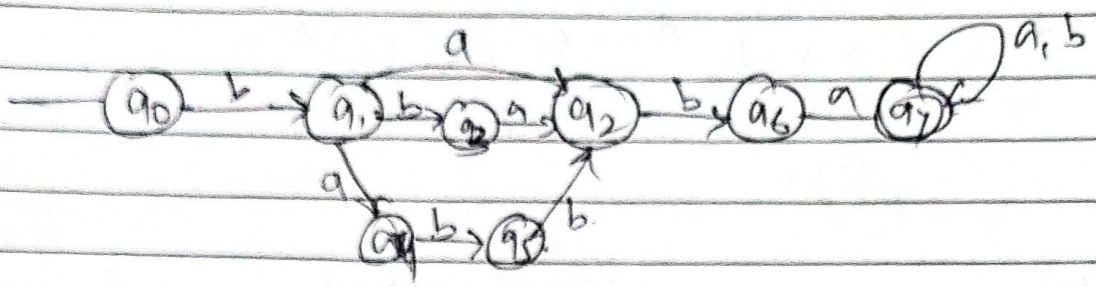
(i)



(ii)



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



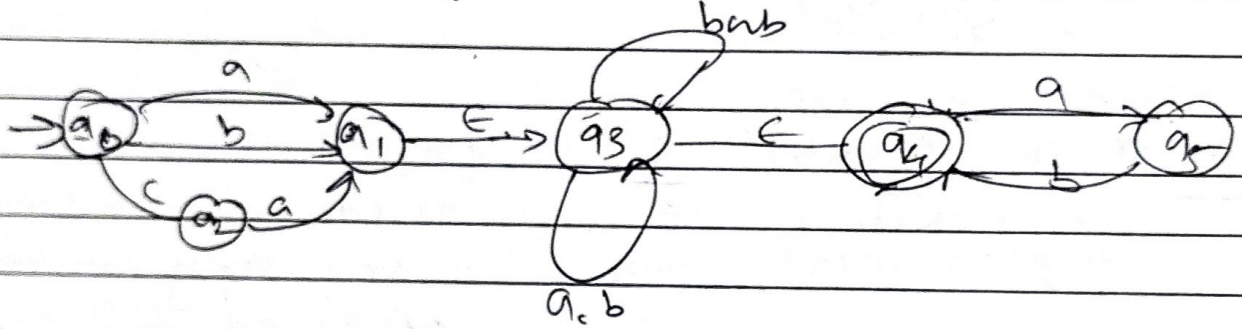
(22) $(a+b+(a)(bab^*+(a+b)^*)^*(ab)^*$

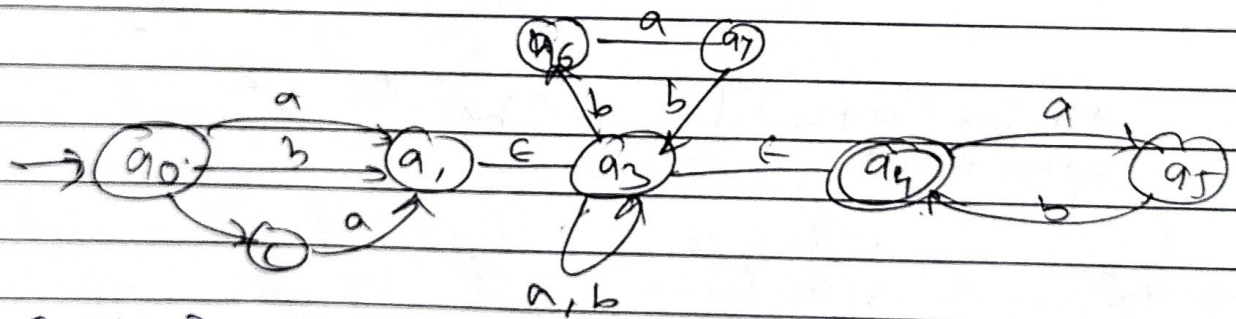
$\Downarrow$ Rule

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

so

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



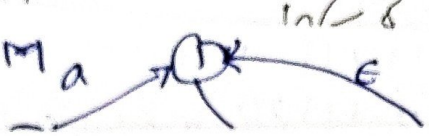


(23) $(a|b)abb$

(24) $(1a+b)^* + (a.c)^*.\#$

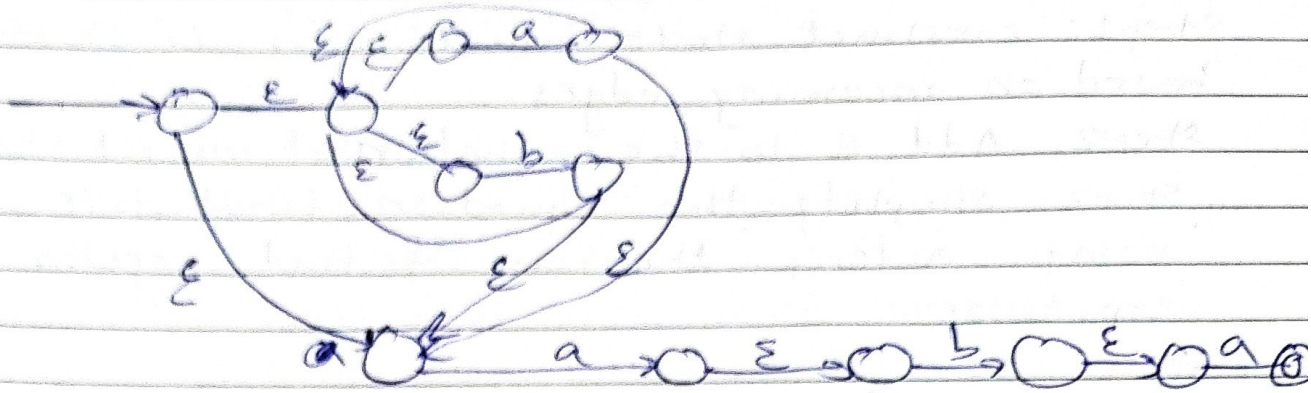
(25) $a(b+c)^*a$

Q.1 ⇒ 10 M



Even:

Convert RE (a∪b)* aba to NFA



Arden's Theorem: $R = Q + RP$ has unique solution

$R = QP^*$

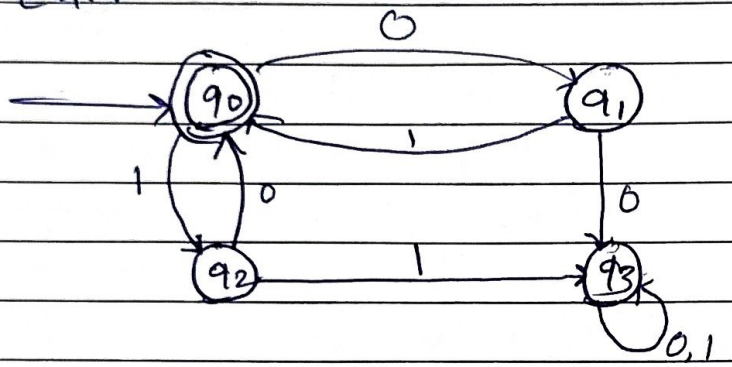
DFA → RE (Direct method)

Step1: construct state equations for all states based on incoming edges.

Step2: Add ϵ to the equations of initial state.

Step3: simplify the equation (final state) using arden's theorem & find regular expression.

Eg.1:



Step2: ~~$q_0 = \epsilon + q_2 0 + q_1 1$~~

$q_0 = \epsilon + q_1 1 + q_2 0$

$q_1 = q_0 0$

$q_2 = q_0 1$

$q_3 = q_1 0 + q_2 1 + q_3 0 + q_3 1$

Step3: simplify final state equation

$q_0 = \epsilon + q_1 1 + q_2 0.$

$q_0 = \epsilon + q_0 01 + q_0 10$

$q_0 = \epsilon + q_0 (01 + 10)$

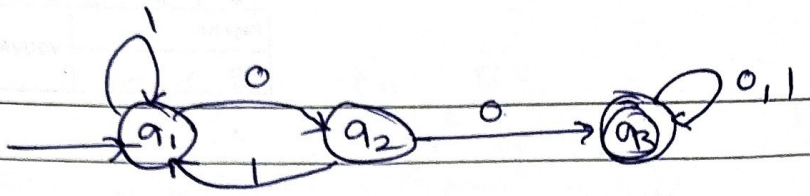
$R = \phi + R$ $\quad P$ Arden's Theorem.

$R = QP^*$

$q_0 = \epsilon (01 + 10)^*$

Identities of RE $[\epsilon + R^*] = R^*$

$q_0 = (01 + 10)^*$



$q_1 = \epsilon + q_1 \cdot 1 + q_2 \cdot 1$

$q_2 = q_1 \cdot 0$

$q_3 = q_2 \cdot 0 + q_3 \cdot 0 + q_3 \cdot 1$

Here we will solve q_3 because it's final state. to solve q_3, we will need q_2, so q_2 have to solve, to solve q_2 we will need q_1, so q_1 have to solve.

So, $q_1 = \epsilon + q_1 \cdot 1 + q_2 \cdot 1$

$= \epsilon + q_1 \cdot 1 + q_1 \cdot 0 \cdot 1$

$= \epsilon + q_1 (1 + 01)$

$R = Q + R \cdot P$

$R = Q P^*$

$q_1 = \epsilon + (1 + 01)^*$

$q_1 = (1 + 01)^*$

~~$q_2 = (1 + 01)^* 0$~~ $q_2 = (1 + 01)^* 0$

$q_3 = q_2 0 + q_3 \cdot 0 + q_3 \cdot 1$

$= q_2 0 + q_3 (0 + 1)$

$= \underbrace{(1 + 01)^* 0 \cdot 0}_{Q} + \underbrace{q_3 (0 + 1)}_{R P}$

$R = Q P^*$

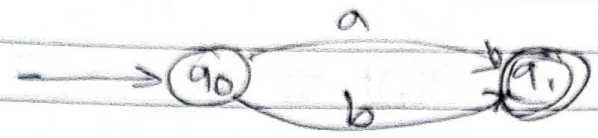
$q_3 = (1 + 01)^* 0 \cdot 0 (0 + 1)^*$

DFA TO RE

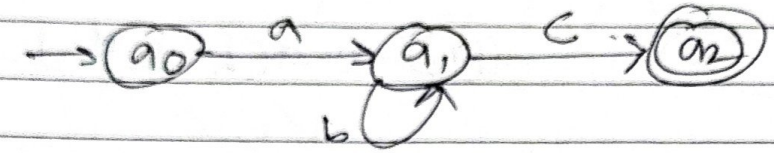
DFA ↓ RE ↓

(1) 

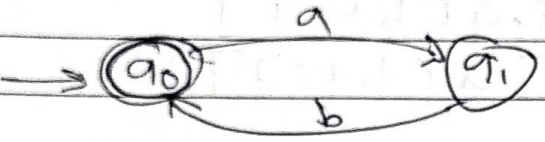
ab

(2) 

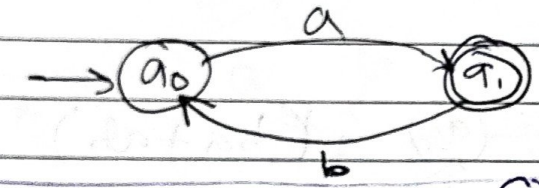
= ab+b

(3) 

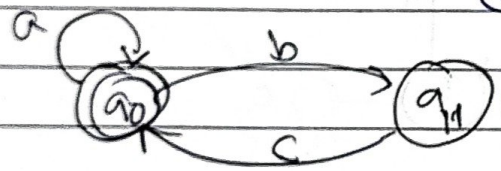
= ab*c

(4) 

= (ab)*

(5) 

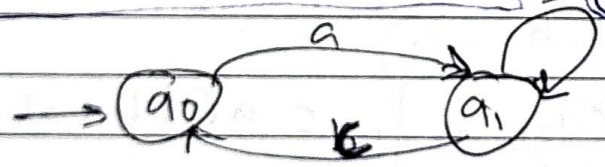
= a(ba)*

(6) 

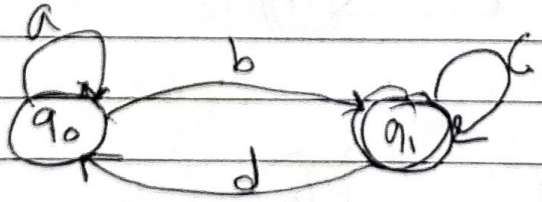
(i) a* + bc* This will not work because, This expression will not generate the string abc

(ii) a*(bc)*. This will not work because This expression will not generate bca string

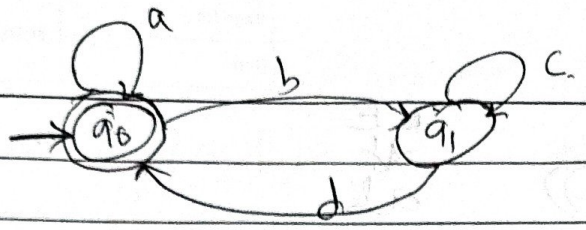
(iii) (a* + (bc)*)* = (a+bc)*

(7) 

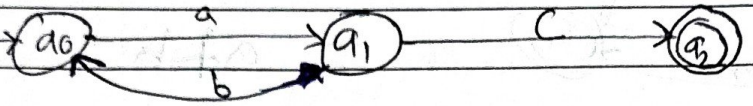
= a(b+(a)*

(8) 

= a*b(c+da*b)*

(9) 

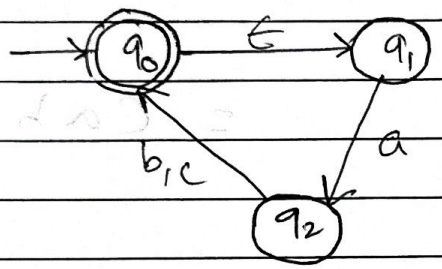
$= (a + bc^* d)^*$

(10) 

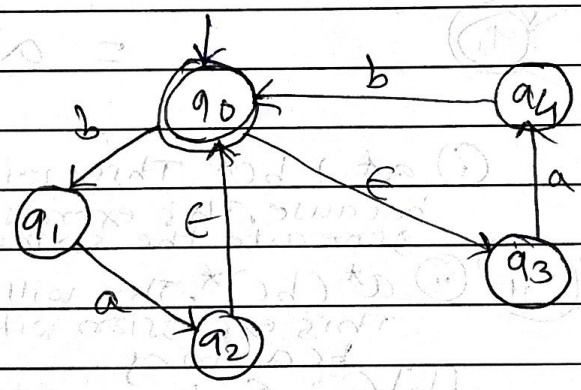
$= a(ba)^* c$

(11) 

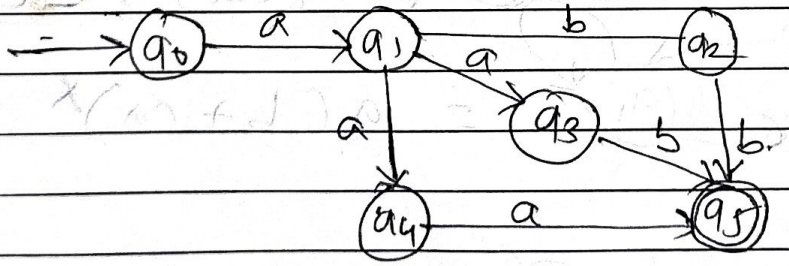
$= a(b + (a)^* d.$

(12) 

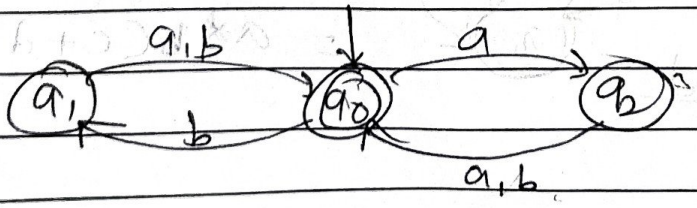
$= [\epsilon . a(b+c)]^*$
$= [a(b+c)]^*$

(13) 

$= [ba + ab)^*$

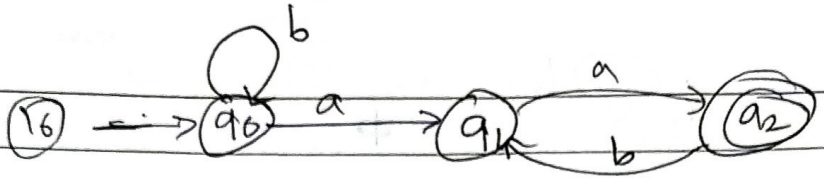
(14) 

$= a(bb + ab + aa)$

(15) 

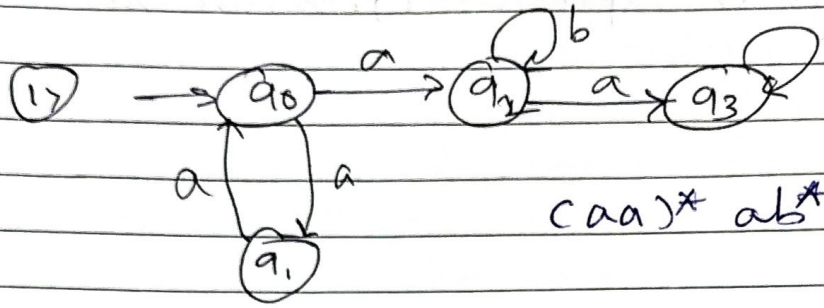
4 to

16



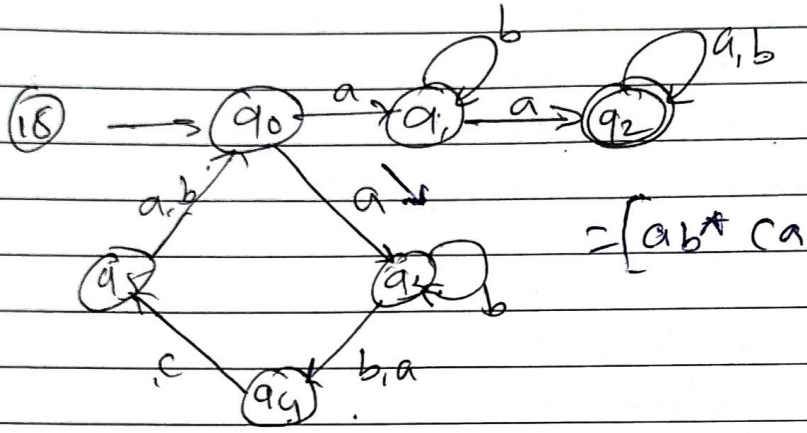
$= b^* aa(ba)^*$

17



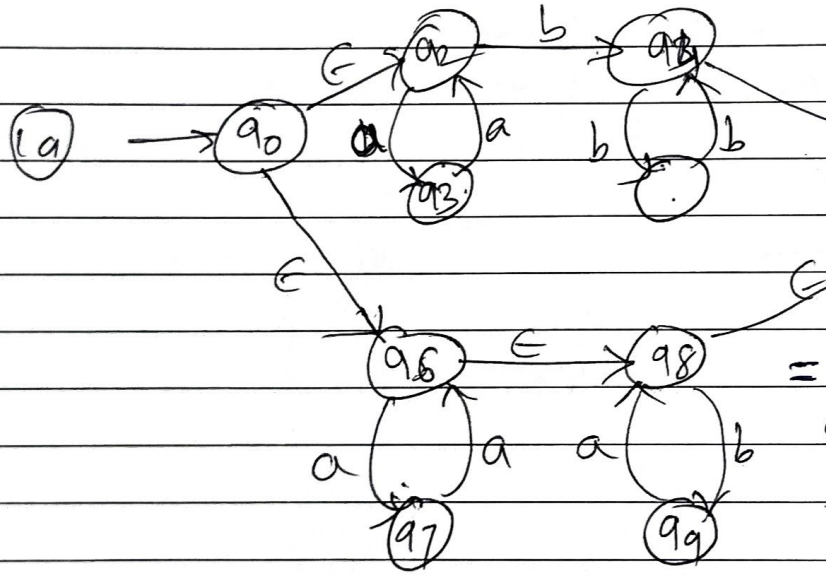
$(aa)^* ab^* a(a+b)^*$

18



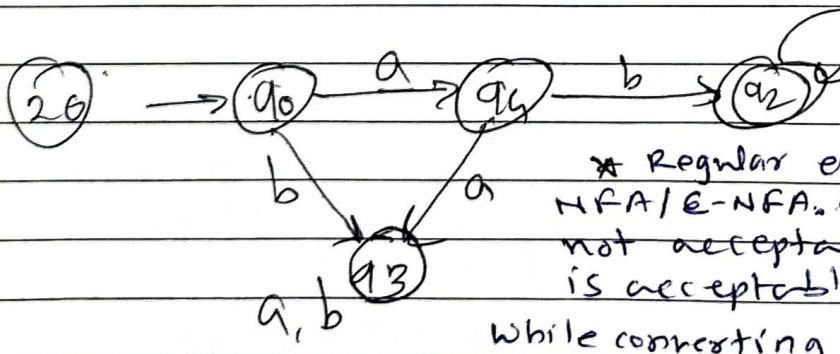
$= [ab^*(a+b)((a+b)]^* ab^* a(a+b)^*$

19



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

20



$= ab(a+b)$

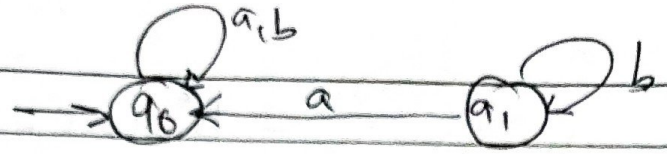
* Regular expressions are analogus to NFA/ε-NFA. RE doesnot tells what is not acceptable, it just tells what is acceptable. Here q3 is dead state. While converting DFA to RE, we don't consider dead state.

M T W T F S S

Page No.: YOUVA

Date:

(21)



ϕ

ϕ is valid regular expression (premitive expression)

Equivalence of DFA

We say two DFA are equivalent if following two condition are true

(1) for given two DFA's DFA1 and DFA2 In DFA1 there should be same Initial state and same final state. and In DFA2 also there should be same initial state and same final state

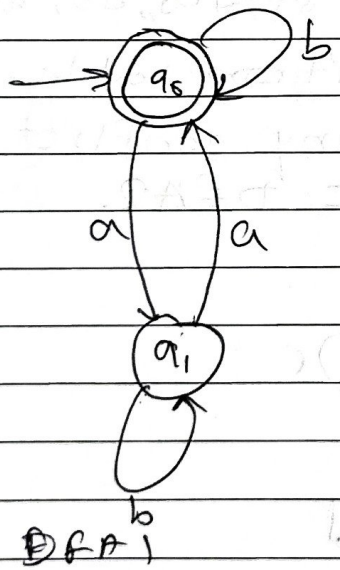
(2) Find out the transitions on initial and final state. Any of transition should not be conflict with each other. We say that transition conflict with each other when ~~any of transition~~ they are not compatible with each other

i.e.

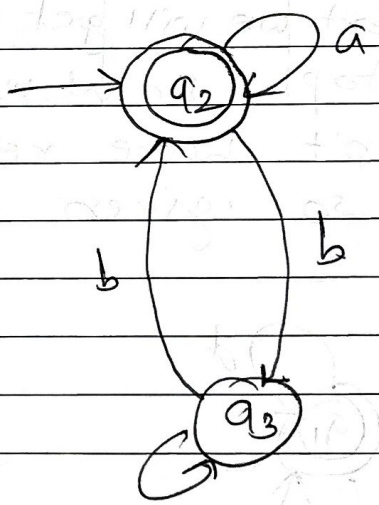
Initial state - Initial state } compatible
final state - final state }

Initial state - final state } conflict
final state - Initial state }

Eg



DFA1



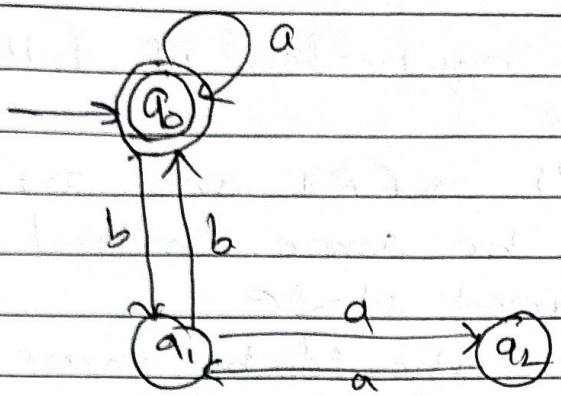
DFA2

for DFA1 Initial state = final state = q_0 }
DFA2 Initial state = final state = q_2 } True

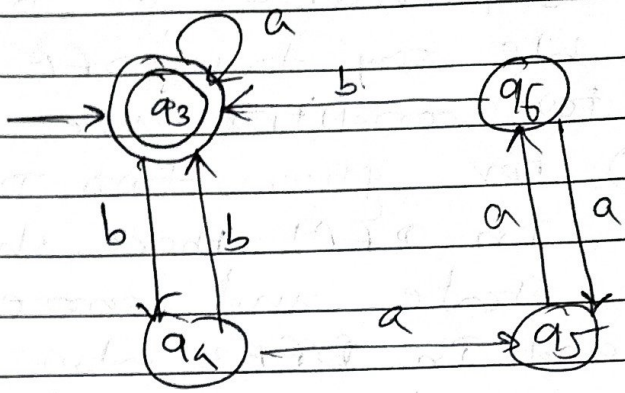
	a	b
$(q_0\ q_2)$	$q_1\ a_2$	

conflict with each

Here q_1 is non final or non initial state (Intermidiate state) and q_2 is final state, both state other so DFA1 ≠ DFA2



DFA 1



DFA 2

① In both DFA's Initial and final states are same i.e q_0 and q_3.

② find transitions

State	a	b
$q_0\ q_3$	q_0 q_3 FS FS	q_1 q_4 IS IS
$q_1\ q_4$	q_2 q_5 IS IS	q_0 q_3 FS FS
$q_2\ q_5$	q_1 q_6 IS IS	q_2 q_5 IS IS
$q_1\ q_6$	q_2 q_5 IS IS	q_0 q_3 FS FS

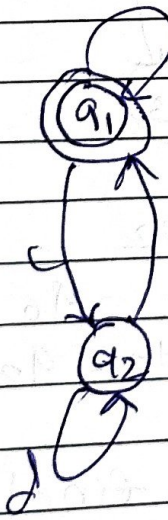
$\Rightarrow$ $q_0 \xrightarrow{i/p} a = q_0$ $\quad q_0 \xrightarrow{i/p} b = q_1$
$q_3 \xrightarrow{i/p} a = q_3$ $\quad q_3 \xrightarrow{i/p} b = q_4$

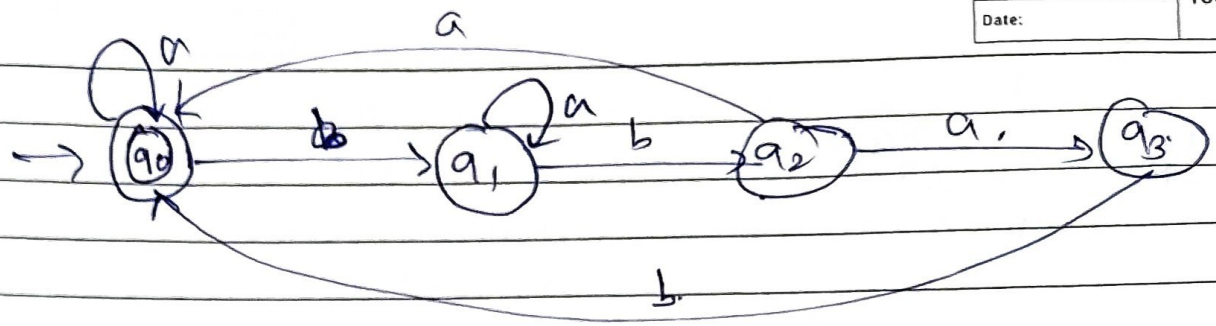
→ New state so, insert into state ~~(start with new state)~~

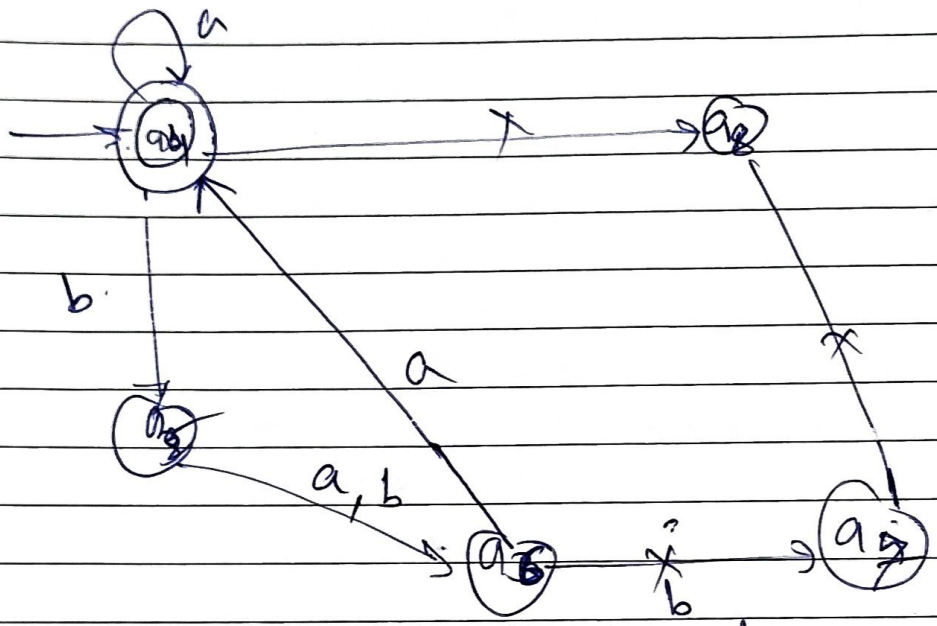
IS: Intermediate state
FS: final state.

Hence ~~fast~~ we will get repeated states, so, we will stop. In above transition table, We don't have received any conflict state. so given DFA1 = DFA2.

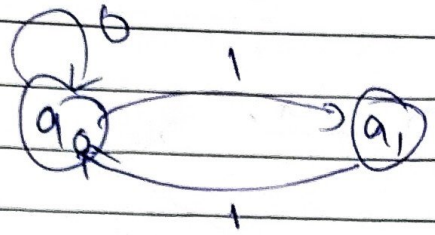
Ex: 2.







	a	b
q_0 q_4	q_6 q_4 ✓	q_1 q_5 ✓
q_1 q_5	q_1 q_6 ✓	q_2 q_6
q_1 q_6	q_1	


0
1
q_2
q_1
1

Arden's Theorem:

If P and Q are two regular expressions over Σ, and if P does not contain ϵ, then the following equation in R given by $R = Q + RP$ has unique solution i.e. $R = QP^*$

$R = Q + RP \longrightarrow (1)$

$= Q + QP^* \cdot P$

$= Q(\epsilon + P^* P)$ $\quad \because \epsilon + R^* R = R^*$

$= QP^*$

Hence proved.

Take above equation again

$R = Q + RP$

$R = Q + [Q + RP]P$

$R = Q + QP + RP^2$

$R = Q + QP + [Q + RP]P^2$

$R = Q + QP + QP^2 + RP^3$

$\vdots$

$R = Q + QP + QP^2 + \cdots + QP^n + RP^{n+1}$

Now for R if used $R = QP^*$ we will get

$R = Q + QP + QP^2 + \cdots + QP^n + QP^* P^{n+1}$

Take Q common

$= Q(\epsilon + P + P^2 + \cdots + P^n + P^* P^{n+1})$

$\epsilon + P + P^2 + \cdots + P^n + P^* P^{n+1}$ represents P^*

so,

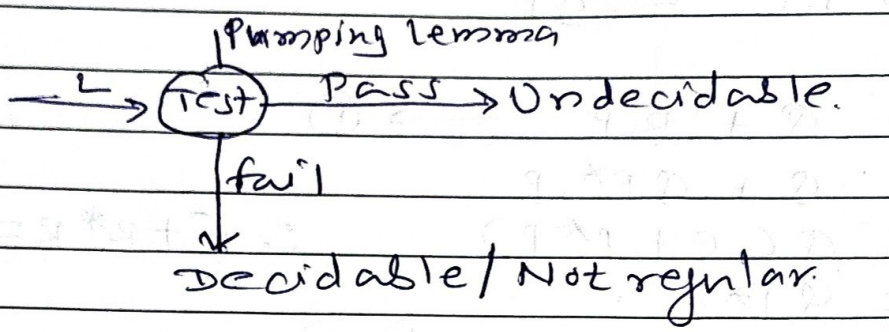
$R = QP^*$

M T W T F S S
Page No.: YOUVA
Date:

Pumping Lemma.

Pumping Lemma is used to prove that language is not regular.

~~For~~ for given language L



So we use pumping lemma to prove the language is not regular.

Theorem:

If L is regular language, then L has pumping length P such that any string 's' where $|s| \geq P$ may be divided into 3 parts i.e. $s = xyz$ satisfying following conditions

1. for each $i \geq 0$, $xy^iz \in L$
2. $|y| > 0$ and
3. $|xy| \leq p$

Recall the notation where $|s|$ represent the length of string s. y^i means that i copies of y are concatenated together and $y^0 = \epsilon$.

When s is divided into xyz, either x or z may be ϵ, but condition 2 says that $y \neq \epsilon$. condition 3 states that the pieces x & y together have length atmost p.

To prove that a language is not regular using pumping lemma, follow the below steps:
(we prove using contradiction).

① Assume that L is regular language
② It has a pumping length = P.
③ All strings longer than P can be pumped $|s| \geq P$.

Q.1 ⇒ 10 M

CIE-40 HA 20 SCE 20 - 10 certi + 10 Poster ESE-40

Even:

4 to

M T W T F S S	
Page No.:	YOUVA
Date:	

(4) Now find a string 's' in A such that $|s| \geq p$.
(5) Divide s into xyz.
(6) Show that $xy^iz \notin A$ for some i.
(7) Then consider all ways that s can be divided into xyz.
(8) Show that none of these can satisfy all the three pumping lemma conditions at the same time.
(9) s can not be pumped = = CONTRADICTION

Ex: Let B be the language $\{0^n 1^n \mid n \geq 0\}$. We use the pumping lemma to prove that B is not regular. The proof is by contradiction.

⇒ Assume that B is regular and has pumping length p. Choose any string $0^p 1^p$, (for example. if p=2 so $B = \{0^2 1^2\} = (0011)$ as string) where $|s| \geq p$., so s can be split into three parts i.e. $s = xyz$, where for any $i > 0$ the string xy^iz is in B. We consider three cases to show that this result is impossible.

(1) The string y consist only of 0's. In this case the string xy^2z has more 0's than 1's so is not member of B, violating condition of pumping lemma. This case is contradiction.

Ex: $s = (0011)$. $p = 2$, $i = 2$

$s = xy^iz = xy^2z$ for given string s

$s = \underset{x}{\emptyset}\ \underset{y}{0}\ \underset{z}{0\,11}$ so for xy^2z we will get

$s = \underset{x}{0}\ \underset{y^2}{0\,0}\ \underset{z}{11}$ This string does not belong to Language B, first condition violates i.e. for $i \geq 0$ $xy^iz \notin B$.

(a) The y consist of only 1's.
$s = (0011)$ $i = 2$ $s = xy^2z$ $s = xyyz$.
$s = \frac{0}{x}\frac{0}{y}\frac{1}{y}\frac{1}{z}$ This is contradiction
because this violates third condition
i.e. $|xy| \leq p$.

(b) The string y consist of both 0's and 1's
In this case string xyyz may have
same number of 0's and 1's but they
will out of order with some 1's before
0's.
Eg $s = \frac{0}{x}\frac{01}{y}\frac{1}{z}$ for xyyz

$s = \frac{x}{x}\frac{0101}{y}\frac{1}{z}$.

violate condition 1 and 3
Hence it is not member of B