Regular Expressions:-

Epsilon ϵ-NFA

ϵ-NFA:- Empty symbols.

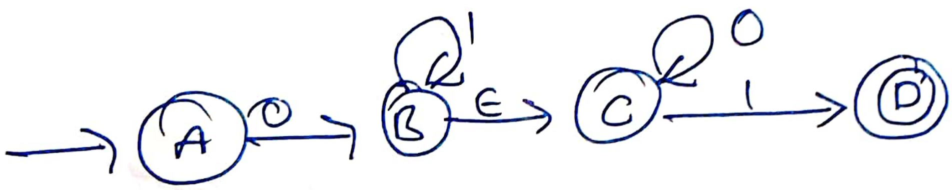
In NFA have $\{Q, \Sigma, q_0, \delta, F\}$

The transition function δ.

So, $\delta : Q \times \Sigma \rightarrow 2^{Q}$ in NFA

Now, you can add 'ϵ' into this

So, $\delta : Q \times \Sigma \cup \epsilon \rightarrow 2^{Q}$.



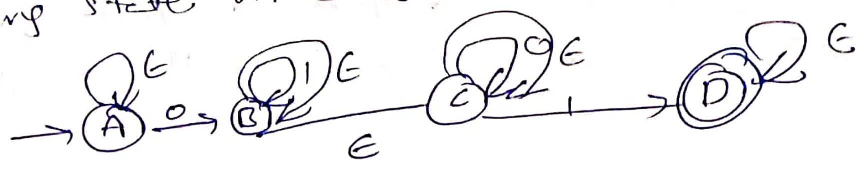
This is ϵ-NFA.

In ϵ-NFA, Every state on ϵ goes to itself.

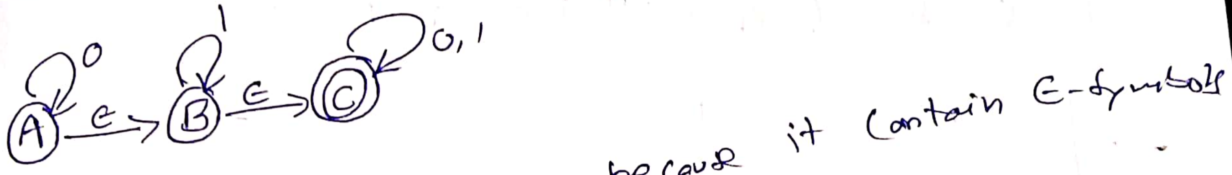
ϵ-closure (ϵ^*):- All the states that can be reached from a particular state only by seeing the ϵ-symbol.

So, Every state on ε goes to itself.

i.e



Conversion of ε-NFA to NFA



Know the NFA is Epsilon NFA because it contain ε-symbols.

So, procedure is to convert ε-NFA to NFA is that we have to check state ~~~~ that where does that the state go on ε* and they the set of states then you set here have to check where it goes to a particular input have to be ~~~~ again check on which state it go on to be ε*.

i.e

State	ε*	input	ε*

ε-closure (ε*):- All the states that can be reached from a particular state only by seeing the ε-symbol

(3)

Let us see, in our given state NFA.

Given

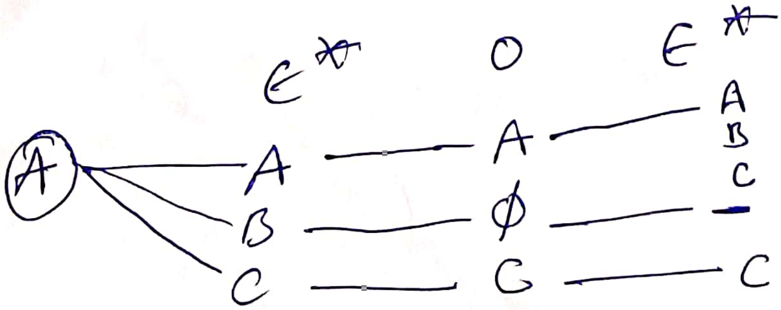
So, ε-closure of A will be A, B, C

	0	1
→ A		
B		
C		

Now, let us fill the above transition table

on A on input 0 where does it go?

So,



So,

	0	1
→A	{A, B, C}	
B		
C		

Now for 1: is ~~state~~ A on input '1'.

A	ϵ^*	1	ϵ^*
	A	ϕ	–
	B	B	B, C
	C	C	C

So, now

	0	1
→A	{A, B, C}	{B, C}
B		
C		

Now let us do the same procedure for state B and state C.

B

	ϵ^*	0	ϵ^*
B	—	ϕ —	—
C	—	C —	C ✓

So, B on input 0 is set C only. → C

Now B on input 1.

B

	ϵ^*	1	ϵ^*
B	—	B —	B, C
C	—	C —	C

So, that means B, C → B, C

So, now for C. will not solve for this You know that procedure

if we do C with 0 and 1 will get only C.

Finally that transition table is

	0	1
→ A	{A, B, C}	{B, C}
B	{C}	{B, C}
C	{C}	{C}

So, this is the NFA of given ε-NFA but we have to find final state

Here A is the initial state you known and C is the final state

in case of NFA, the final state will be any state that can reach the final state only by seeing 'ε'.

So, look at this above diagram given A only state with ε - goes to reach final state and B also only by seeing ε - it can reach the final state C.

So, A and B are also the final state

So, how many final states total ⇒ 3

is A and B and also C

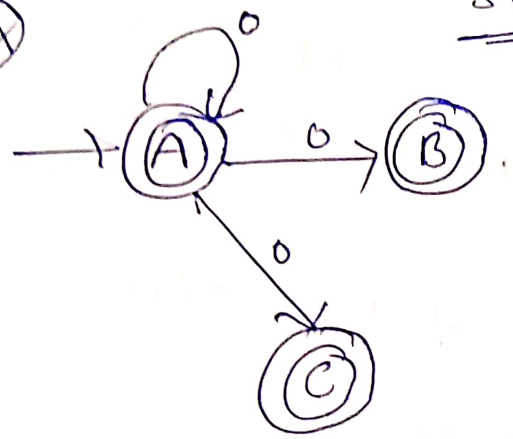
	0	1
→ A	{A, B, C}	{B, C}
B	{C}	{B, C}
C	{C}	{C}

Now, let we draw the transition diagram of the above NFA transition table made over here.

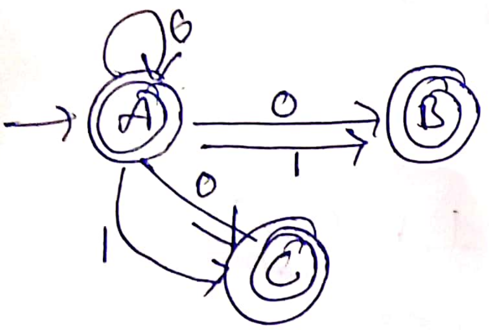
7

Step-1 :- A on input 0 for NFA

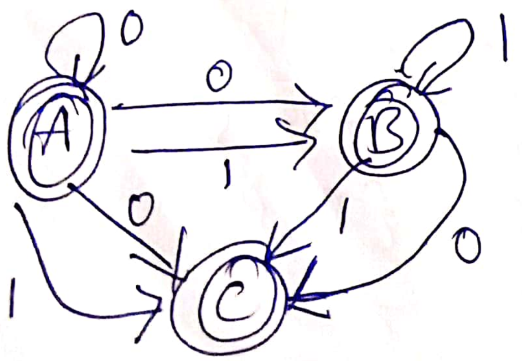
NFA is one to multiple



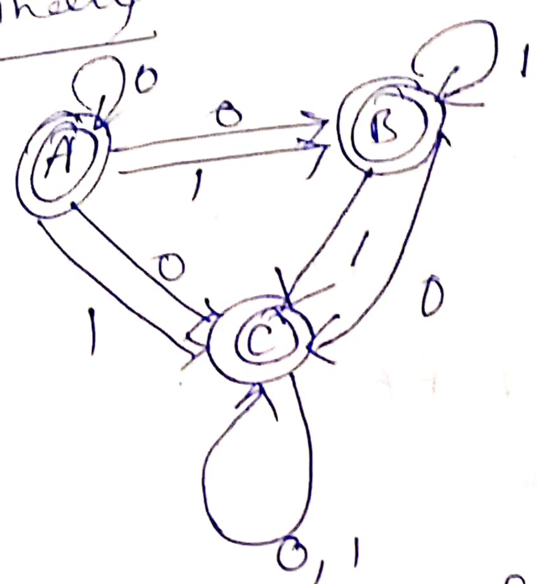
Step : 2 A on input 1 for NFA



Step 3: lets Come to ~~input~~ state B.



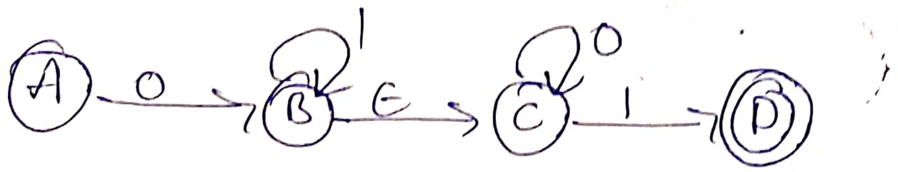
finally: ϵ state.



So, this is the final Equalent NFA.
by using given ϵ-NFA.

Example for conversion of ε-NFA to NFA.

①



① Convert the following ε-NFA to its Equivalent NFA.

Ans :-

NFA.

	0	1
→ A	B, C	φ
B	C	B, C, D
C	C	D
D	φ	φ

So, above is the transition table for given NFA.

So,

	ϵ^*	0	ϵ^*
A	A	B	A C
B	B C	ϕ C	ϕ C
C	C	C	C
D	D	ϕ	ϕ

	ϵ^*	1	ϵ^*
A	A	ϕ	ϕ
B	B C	B D	B C D
C	C	D	D
D	D	ϕ	ϕ

Now we got the transition table

~~First~~ to draw NFA transition diagram

Find, final state.

Any state that can reach the final state only by seeing the ϵ-symbol.

A is not seeing ϵ symbol to B.

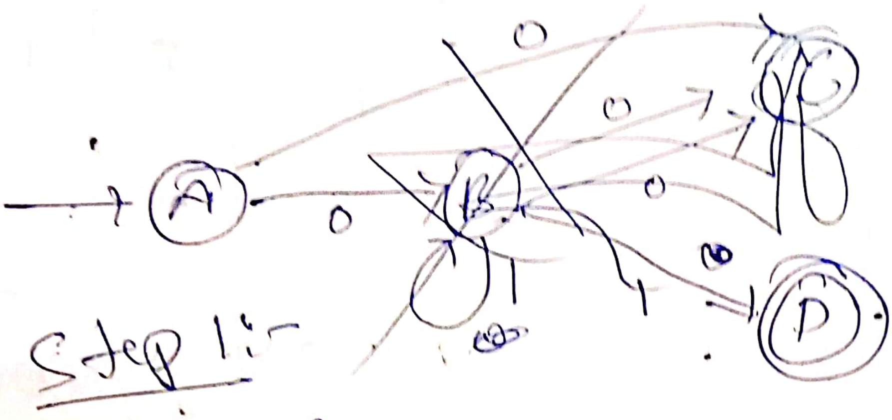
B is seeing ϵ symbol but not going to D

C is also not seeing ϵ to D.

So, there fore D state only final state.

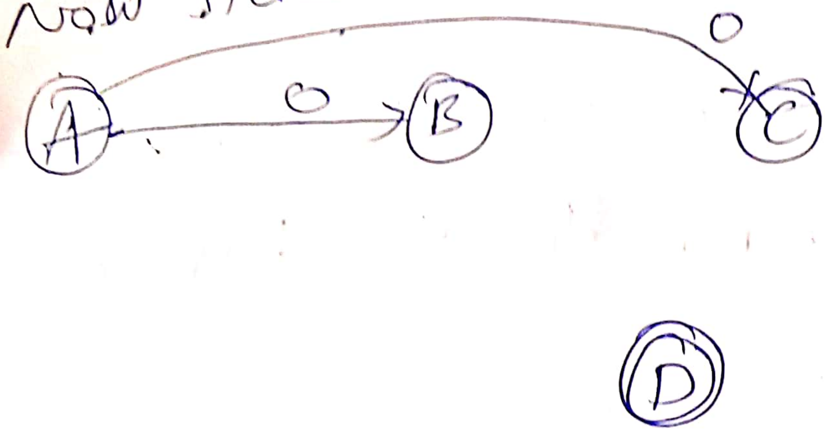
NFA transition diagram over the

State A to input 0 and 1.

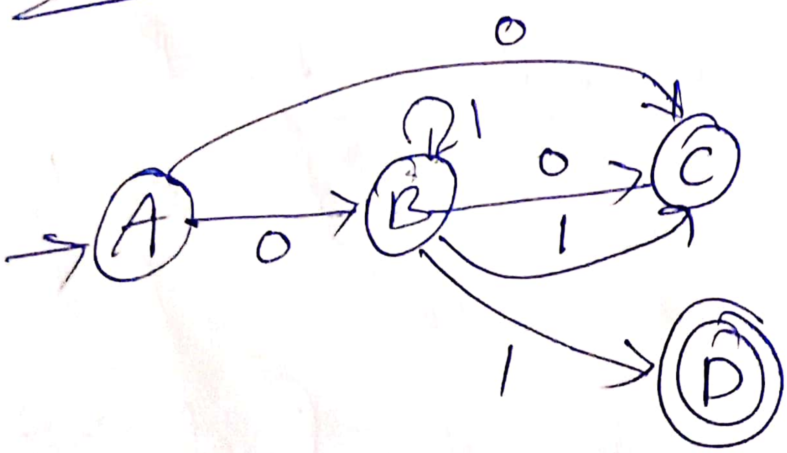


Step 1:-

Now state

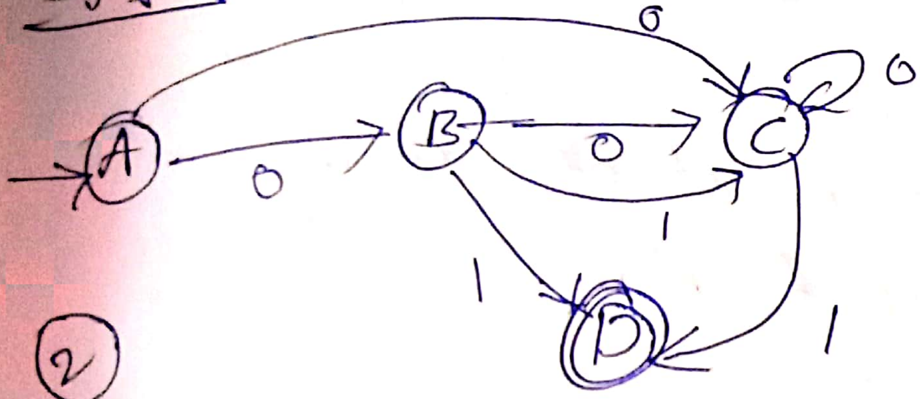


Step 2:- State B with input 0 and 1



Step 3:-

Step 3:- state C with input 0 and 1



thus is the final NFA over given ε-NFA.

2

UNIT-II Regular Expressions (1)

RE - Regular Expressions are used for representing certain set of strings in an algebraic fashion.

Rules for RE:-

1) Any terminal symbol i.e, symbols $\in, \Sigma$ including Λ (Empty) or ϕ (NULL) are regular Expressions.

Ex:- terminals a, b, c and Empty Λ, ϕ

2) The union of two regular Expressions is also a regular Expression.

Ex:- R_1, R_2 are two regular then $(R_1 + R_2)$ be regular.

(3) The Concatenation of two regular Expressions is also a regular Expression.

Ex:- R_1, R_2 are two regular Expression then $(R_1 \cdot R_2)$ are also RE.

(4) The iteration (or closure) of a regular Expression is also a regular Expression

Ex:- $R \to R^*$ $a^* = \Lambda, a, aa, aaa.$

(5) The regular Expression over Σ are precisely those obtained recursively by the application of the above rules once or several times.

Ex:- above all RE are using union or Concatination or iterations on Σ with all input symbol are following one or more times.

~~8971436662~~

2

① Examples of Regular Expressions

Describe the following sets as Regular Expres

1) $\{0, 1, 2\}$

In the Regular Expression. The since set contain 0 or 1 or 2

'or' to symbolic notation in RE is '+'

RE ⇒ {0 + 1 + 2} ⇒ $\boxed{R = 0+1+2}$

② $\{\Lambda, ab\}$ Empty symbol and ab.

$\boxed{RE = \Lambda \ ab}$

A is any Empty

dont want to use + or any other symbol becoz

Empty symbol contain only one 'ab' symbol

③ $\{abb, a, b, bba\}$

it mean it can be any thing

⇒ abb or a or b or bba

so use '+' symbol So, ~~RE ⇒ {~~

RE = abb + a + b + bba

4)

5)

3

4) $\{\Lambda, 0, 00, 000, ---\}$

all the strings are forming by 0 along with empty string Λ.

so, it denotes ~~Clousure~~ Clousure of 0.

so $\boxed{RE = 0^*}$

5) $\{1, 11, 111, 1111, --\}$

→ all the strings are forming by 1 only ~~along~~ ~~with~~

→ does not contain empty string(Λ). so it is not clousure of 1.

so, excluding empty symbol and clousure of one with excluding 1 can be represented by RE as

$\boxed{R = 1^+}$ ~~R^+ excluding~~

① Identities of Regular Expressions

① $\phi + R = R$

Here ϕ Empty set mean union of ϕ with Regular Expression is also one Regular Expression.
→ R is Regular Expression.

② $\phi R + R\phi = \phi$

ϕ concatination R and R concatination of ϕ is equal to ϕ.

③ $\epsilon R = R\epsilon = R$

ϵ (Epsilon) contatination of R and Regular Expression with any ϵ (Epsilon symbol) will get Regular Expression.

④ $\epsilon^* = \epsilon$ and $\phi^* = \epsilon$

if you have ϵ Perform cloufure ϵ then you will get ϵ only.

$\phi^* = \epsilon$ (2)

Cloufure of ϕ is not ϕ it is Epsilon.

(5) $R + R = R$

Union of two Regular Expression also one Regular Expression.

(6) $R^* R^* = R^*$

If you have clouse Regulare Expression and Concatination with another same Regular Expression then you will get Regular cloufure expression.

$(R.R^*)$ a concatination.

(7) $RR^* = R^*R$

Regular with Concatination of Closure Regular Expression produces cloufure of R with concatination of Regular Expression.

(8) $(R^+)^* = R^*$

If you have closure expression and perform again closure expression then you will get closure of that regular expression.

(9) $\epsilon + RR^* = \epsilon * R^*R = R^*$

$RR^* \Rightarrow R^+$ means without ϵ.

So now, add ϵ with R^* then will get R^*.

(10) $(PQ)^*P = P(QP)^*$

(11) $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$

(12) $(P+Q)R = PR + QR$ and

$R(P+Q) = RP + RQ$

* ARDEN'S THEOREM * ①

1) If P and Q are two Regular Expressions over Σ, and if P does not contain ε, then the following Equation on R given by $R = Q + RP$ has a unique solution i.e, $R = QP^*$.

Ans :-

$$R = Q + RP \quad \text{——— ①}$$

let take $R = QP^*$ and substitute in Equation ①.

$$R = Q + QP^*P$$

$$= Q(\epsilon + P^*P)$$

By identity law $\boxed{\epsilon + R^*R = R^*}$

then use in this

$$R = Q(P^*)$$

$$R = QP^*$$

So, we have proved that $R = QP^*$.

Hence proved part ① i.e, $R = QP^*$

Now Part (2) is the unique solution. (2)

So.

$$R = Q + RP$$

replace R in $R = Q + RP$

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

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

replace R in again.

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

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

$\vdots$

Continue with n-no. of times

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

Now Replace $\boxed{R = QP^*}$

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

$$= Q\,[\epsilon + P + P^2 + \cdots P^n + P^* P^{n+1}]$$

observe this it is closure of P.

ie, P^*.

$$= Q\,[P^*] \Rightarrow \boxed{R = QP^*}$$

RP

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

(3)

Solution:-

Now take LHS.

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

LHS $\Rightarrow (1+00^*1) + (1+00^*1)^*(0+10^*1)$

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

take identity law $\boxed{\epsilon + R^*R = R^*}$

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

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

$\boxed{\epsilon.R=R}$ → identity laws

$\boxed{\epsilon+R^*R=R^*}$ → identity law. so,

~~$= \epsilon + 00^*$~~ so. $\boxed{\epsilon + 0^*0 = 0^*}$

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

$\Rightarrow 0^*1(0+10^*1)^*$

$\Rightarrow$ RHS.

$\therefore$ LHS = RHS. Hence proved.

① # Designing Regular Expressions

Design Regular Expression for the following languages over $\{a, b\}$

1) language accepting strings of length Exactly 2.
2) language accepting strings of length atleast 2.
3) language accepting strings of length atmost 2.

Solution:-

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

$R_1 = aa, ab; ba, bb$.

use union operation(+)

$R_1 = aa + ab + ba + bb$

$= a[a+b] + b[a+b]$

$= (a+b)[a+b]$.

$\boxed{R_1 = (a+b)(a+b)}$

(2) $L_2 = \{aa, ab, ba, bb, \text{aaa} \ldots\}$ (2)

Here string accepting atleast 2 so we may use more than 2.

so. $L_2 = \{aa, ab, ba, bb, aaa, \ldots\}$

$$\boxed{R_2 = (a+b)(a+b)(a+b)^*}$$

(3) $L_3 = \{\epsilon, a, b, aa, ab, ba, bb\}$

Here atmost 2. so we have to add ϵ and one string, second string and all.

so.

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

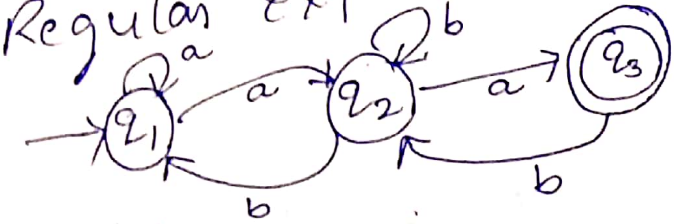
$$= (\epsilon + a + b)(\epsilon + a + b)$$

$$\boxed{R_3 = (\epsilon + a + b)(\epsilon + a + b)}$$

Scanned with CamScanner

(1) Finite Automata to Regular Expression.

(1) Find the Regular Expression for the following NFA.



Ans

given NFA is above.

→ first we have to prepare Equations for the given NFA for all the State.

→ put all the Equation into the final State then that will give us the final Regular Expression.

let see how many states.

first take final state & check all the incoming transition. q_3 = with all inputs to that state [incoming transactions]

so. $q_3 = q_2 a$. ——— (1)

Now, q_2 = with all input to its state

1) q_1 coming with input a
2) q_2 coming with input b
3) q_3 coming with input b

and union all the inputs

$$q_2 = q_1 a + q_2 b + q_3 b \quad \text{---} (2)$$

(2)

Now

q_1 = with all inputs coming to state

q_1 contain one initial state so, when you are taking initial take you have to add

ϵ also.

So, $q_1 = \epsilon + q_1 a + q_2 b$ --- (3)

Now, simplify the states. now take equation (1).

(1) $\rightarrow$ $q_3 = q_2 a$

substitute q_2 value from equation (2).

$$q_3 = (q_1 a + q_2 b + q_3 b)\, a$$
$$= q_1 aa + q_2 ba + q_3 ba \quad \text{---} (4)$$

(2) $\rightarrow$

$$q_2 = q_1 a + q_2 b + q_3 b$$

putting value q_3 from (1) so,

$$q_2 = q_1 a + q_2 b + (q_2 a) b$$

$\therefore q_2 = q_1 a + q_2 b + q_2 ab$

$\therefore q_2 = q_1 a + q_2 (b + ab)$

if you take above Equation it will be like

$R = Q + RP$.

$R = q_2$ | $P = b + ab$

$Q = q_1 a$

So, $R = Q + RP$ gives Arden's theorem

$R = QP^*$.

So.

$q_2 = (q_1 a)(b + ab)^*$ —— (5)

Now, let us take Equation (3)

(3) $\rightarrow q_1 = \epsilon + q_1 a + q_2 b$

putting value of q_2 from Equation (5)

$q_1 = \epsilon + q_1 a + ((q_1 a)(b + ab)^*) b$

$= \epsilon + q_1 [a + a(b + ab)^*] b$

again $R = Q + RP$ by Arden theorem

$R = QP^*$

So. $R = q_1$, $Q = \epsilon$

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

so. $R = QP^*$

$\Rightarrow q_1 = \epsilon((a + a(b+ab)^*b))^*$

Now, Identities law $\boxed{\epsilon . R = R.}$

$\therefore q_1 = (a + a(b+ab)^*b)^*$ —— (6)

Final step is to take final state (q_3) and substitute all Equations.

$q_3 = q_2 a$

q_2 we got equation (5)

$= q_1 a(b+ab)^* a$

Now substitude Equation (6)

$$\boxed{q_3 = (a + a(b+ab)^*b)^* a(b+ab)^*a.}$$

$q_3 \Rightarrow$ final Required ~~Expression is~~ Regular Expression for the given NFA.

Pumping lemma (for Regular Languages)

→ Pumping lemma is used to prove that a Language is NOT Regular.

→ It can not be used to prove that a Language is Regular.

If A is a Regular Language, then A has a Pumping Length 'p' such that any string 's' where $|s| \geq p$ may be divided into 3 parts $s = xyz$ such that the following Conditions must be true:

(1) $xy^iz \in A$ for every $i \geq 0$

(2) $|y| > 0$

(3) $|xy| \leq p$.

To prove that a language is not Regular using Pumping Lemma. follow the below steps:

So, we need to prove Contradiction.

1. Assume that A is Regular
2. It has to contain pumping length (say P)
3. All strings longer than P can be pumped $|s| \geq p$
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 3 pumping conditions at the same time
9. S can not be pumped = CONTRADICTION
   (Contradiction)

Pumping lemma Example

① Prove that the language $A = \{a^n b^n \mid n \geq 0\}$ is Not Regular language using pumping lemma

Ans

Assume that A is Regular

pumping length $= P$

$S = a^P b^P \Rightarrow S = aaaaaaabbbbbbb$

$$\underbrace{\quad}_{x}\ \underbrace{\quad}_{y}\ \underbrace{\quad}_{z}$$

$P = 7$

Case 1:- The y is in the 'a' part.

$$\underbrace{aa}_{x}\underbrace{aaaa}_{y}\underbrace{abbbbbbb}_{z}$$

Case 2:- The y is in the 'b' part

$$\underbrace{aaaaaaab}_{x}\underbrace{bbbbb}_{y}\underbrace{b}_{z}$$

Scanned with CamScanner

Case 3:- The y is in the 'a' and 'b' part

$$\underbrace{aaaagaa}_{x}\underbrace{bbb}_{y}\underbrace{bbbb}_{z}$$

Now take case 1:-

$xy^iz \Rightarrow xy^2z$

So, Case ①. aa | aaaaaaaa | abbbbbbb

2 8 7

Total $\Rightarrow$ a's $\Rightarrow$ 11

$\therefore 11 \neq 7.$ Hence it is not Equal

Case 2:-

$xy^iz \Rightarrow xy^2z.$

aaaaaaa bb | bbbbbbbb | b

7 2 8 1

$7 \neq 11$ Hence it is also not equal

Case 3! $xy^iz \Rightarrow xy^2z$ Hence it is also not Equal.

aaaaa | aabbaabbb | bbbbb

$\boxed{a^n b^n \text{ given condition}}$

$|xy| \leq P$

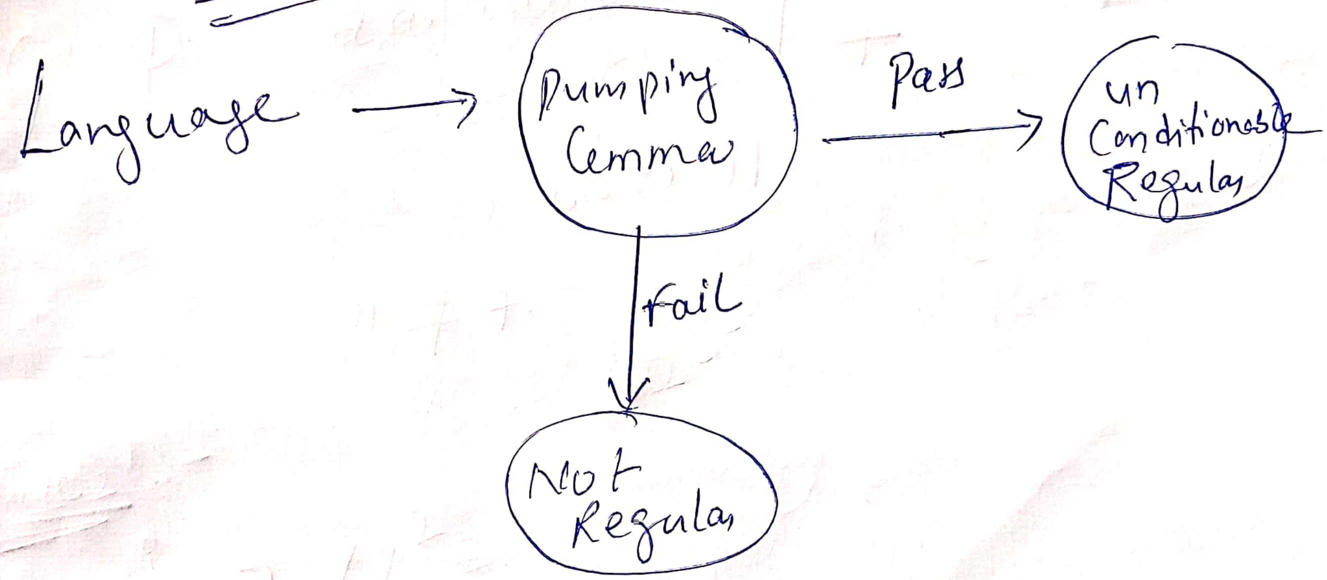
Hence the Three case are not proving

So. by taking three Condition getting Contradiction.

Hence the above language is not a Regular Language.

Hence proved.

whether the language is Regular or not



Pumping lemma:-

If L is an infinite language then there exists some positive integer 'n' (pumping lemma) such that ~~only~~ any string $w \in L$ has length greater than equal to 'n'. i.e, $|w| \geq n$ then w can be divided into three parts, i.e, $w = xyz$ satisfy following conditions

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

$$\begin{tikzcd} L \arrow[r] & \text{PLT} \arrow[r, "\text{Pass}"] \arrow[d, "\text{Fail}"] & \text{Undecidable} \\ & \text{Not regular} & \end{tikzcd}$$

Can be or Can not be

Reg / Not Regular

Example:

① Prove that the Language $L = \{a^n b^{2n} \mid n \geq 0\}$ is Not Regular using Pumping Lemma.

Ans

given Language $L = a^n b^{2n}$ $n \geq 0$.

(n-times a) (2n times b)

$\boxed{n=2}$

$\therefore L = aa\,bbbb$

$L = aabbbb \in L$.

Case 1:-

make three partitions

$L = \underbrace{aa}_{x}\underbrace{bb}_{y}\underbrace{bb}_{z} \in L$ Partition ①

So, now i values increase.

then $y^i = 2$. (pump the values)

$\therefore$ So, $L = \underbrace{aa}_{x},\underbrace{bbbb}_{y}\underbrace{bb}_{z} \notin L$ Not

The above string belongs to the Language?

No above string not following Language.

Scanned with CamScanner

$L = \underbrace{aa}_{x}\ \underbrace{bbbb}_{y}\ \underbrace{bb}_{z} \notin L.$

So, sting not belongs to language.

So. Test is failed.

Hence, the given language is not Regular

Case 2 :- [Partition NO 2]

given $L = \underbrace{a}_{x}\underbrace{ab}_{y}\underbrace{bbb}_{z}$ i value increase $y=2$

one more time

$L = \underbrace{aa}_{x}\ \underbrace{bbbb}_{y}\ \underbrace{bb}_{z}$ ε $\quad \underbrace{a}_{x}\ \underbrace{abab}_{y}\ \underbrace{bbb}_{z}$

the above string again not following the Language

So, $\underbrace{a}_{x}\ \underbrace{ab\ ab}_{y}\ \underbrace{bbb}_{z} \notin L$

So. Test is faild

Hence given language is not Regular.