What is Regular Expression

2. Regular Expression

* The language accepted by finite automata is called Regular language
* Generally we uses Regular Expressions inorder to use Regular language.
* Regular Expression means a pattern, an algebraic Expression which is useful inorder to describe (or) represent regular language.

Regular Expression definition :

Let Σ be the input alphabet then Regular Express (RE) can be defined as.

1. ϕ is a R.E which denotes empty set $\{\}$
2. ϵ is a R.E which denotes null string $\{\epsilon\}$ (↓ we have a string length is zero)
3. let 'a' be an input symbol in Σ (↑ alphabet), then 'a' is R.E.
4. if R_1 is a RE then R_1^* is also a R.E. → klen closure of R_1 = ex $\Sigma = \{a\}$ $\{\epsilon, a, aa, aa \ldots\}$ → any occurence
5. if R_1 is a RE then R_1+ is also a R.E

   ex: $\Sigma = \{a\}$ → Positive closure → we have Minimum 1 occurence

   $\{a, aa, \ldots\}$
6. if R_1 & R_2 are R.E then R_1+R_2 is also a R.E
7. if R_1 & R_2 are R.E then $R_1 . R_2$ is also a RE (dot → dot means concatination)

+ This about R.E. we can define R.E by using These 7 points.

Regular Expression Examples

1. design a R.E to accept all possible combination of a's & b's over $\Sigma = \{a, b\}$.

* so here that input alphabet contain two symbols a and b
* any combination a's & b's it means '0' combination, also there so ϵ

$L = \{\epsilon, a, b, aa, bb, aaa \dots\}$ (0 combination, 1 comb, 2 comb, 3 combination of a)

$R.E = (a+b)^*$ → Point 4 any occurrence of a so Klene closure

2. design a R.E to Ends with 00 over $\Sigma = \{0, 1\}$.

ends with 00. it means before we have any numbers

$L = \{00, 100, 000, 1000, 0000 \dots\}$

$R.E = (0+1)^* 00$ → ends with 00

any combination so Klene closure.

3. starts with 1 & ends with 0.

$C = \{10, 11000\,0 \dots\}$ → in b/w 1_0 we have any constraints.

$R.E = 1\,(0+1)^*\,0$ → combinations. → ends 0

↓ starts 1

4. Any no.of a's followed by any no.of b's followed by any no.of c's → 1a followed 1b followed by 1

$L = \{\epsilon, abc, aab, aac \dots\}$ (we have 0 of 1's, 0 of b's, 0 of c's → 2's as followed by single c)

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

5. Atleast 1 a followed by atleast one c.

$L = \{ ac, abc, aab, aac, bbc \ldots \}$ (ε not)

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

atleast means 1 occurance not possible.

↳ positive closure → it mean 0 occ not possible Minimum 1 occurance.

6. Begin (or) ends with either 00 or 11.

$R.E = (00+11)(0+1)^* + (0+1)^* (00+11)$

↳ starts with 00 (or) 11 ; ↳ combinations. ; ↳ ends with 00 (or) 11 . .

7. Third character from right end is 'a' over $\Sigma = \{a, b\}$.

$R.E = (a+b)^* a (a+b)(a+b)$ → 3 should 'a' only ; → 2 char right end ; → 1 character from right end

8. Atleast 2 b's over $\Sigma = \{a, b\}$

$L = \{ bb, abb, bba \ldots \}$

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

9. Exactly 2 b's over $\Sigma = \{a, b\}$

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

10. Even length string over $\Sigma = \{0\}$

$R.E = (00)^*$ $L = \{ \epsilon, 00, 0000 \ldots \}$

11. odd length string over $\Sigma = \{1\}$

$L = \{ 1, 111, 11111 \ldots \}$

$R.E = 1(11)^* \text{ (or) } (11)^* 1$

12) No. of a's are divisible by 3.

$$R.E = (b^* \underset{1}{a} b^* \underset{2}{a} b^* \underset{3}{a} b^*)^*$$

→ Maximum

13) Atmost 2 a's

it means 0a, 1a, (or) 2a.

$0a \rightarrow b^*$

$1a \rightarrow b^*a$

$2a \rightarrow b^*ab^*ab^*$

* in qustio 2a means

$$R.E = b^* + b^*a\,b^* + b^*a\,b^*a\,b^*$$

14) 3 consecutive a's

before we have any no. of a's & b's

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

→ after we have any no. of a's & b's.

15) doesn't contain the substring ab.

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

→ 2 occure : bb aa X
3 occu = bbb aaa X not on

Algebraic laws of Regular Expression
(or)
Identity Rules of Regular Expression

* we are using two operat + & . + means union
. means concatenation

we add ϕ + R we get output as R.

1. $\phi + R = R$ → R means R.E.

2. $\phi . R = R . \phi = \phi$ → Multiplied with 0. we get zero only i.e ϕ.

3. $\epsilon \cdot R = R \cdot \epsilon = R$

4. $\epsilon^* = \epsilon$ & $\phi^* = \epsilon$

5. $R + R = R$

6. $R^* \cdot R^* = R^*$

7. $R \cdot R^* = R^* \cdot R = R^+$ → imp rule.

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

9. $\epsilon + RR^* = \epsilon + R^* R = R^*$

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

11. $(P+Q)^* = (P^* \cdot Q^*)^* = (P^* + Q^*)^*$

it contain ϵ also

$R^* = \{\epsilon$

$R^+ = \{$ not contains ϵ

if we add R^* to R $R^* = \{\epsilon$

$R^+ = \{\epsilon$

simplification of Regular Expression

Ex 1 :- $(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$ = RHS

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

1st term → so common this 2nd terms

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

↳ RHS

$\epsilon \cdot R = R$ ← $(1+00^*1)\left(\epsilon + \underbrace{(0+10^*1)^*(0+10^*1)}_{R}\right)$ → if we observe that

common common

$\epsilon +$ we assume R^*

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

1 as common ← $= 1(\epsilon + \underbrace{00^*}_{\epsilon + R \cdot R^*})(0+10^*1)^*$

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

Ex 2 : $\epsilon + 1^*(01)^*(01^*(01)^*)^* = (1+01)^*$

R R^*

$\epsilon + R \cdot R^* = R^*$ ← $= (1^*(01)^*)^*$

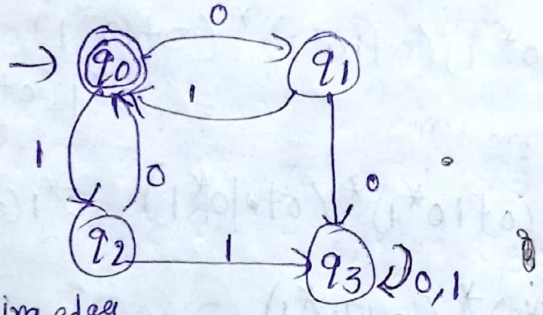
$(P^*Q^*)^* = (P+Q)^*$ ← $= (1+01)^*$

Conversion of finite Automata to regular Expression

- whereas input is finite Automata output is the Regular expression.

1. construct state equations for all the states based on incoming edges.
2. Add ϵ to the equation of initial state
3. simplify the equation with Arden's algorithm for R.E

Ex1:



1. incoming edges

$q_0 = q_1 \cdot 1 + q_2 \cdot 0 + \epsilon$ → for initial state we have to add ebsilon.

$q_1 = q_0 \cdot 0$

$q_2 = q_0 \cdot 1$

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

* so Here the final state is also q_0. we solve q_0 that is the solution of the Regular Expression.

* in Arden's theorem means

$R = Q + RP$ → in this form

$R = QP^*$ ↙ we can return this

* this is arden's algorithm.

substitute the remaining values into q_0

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

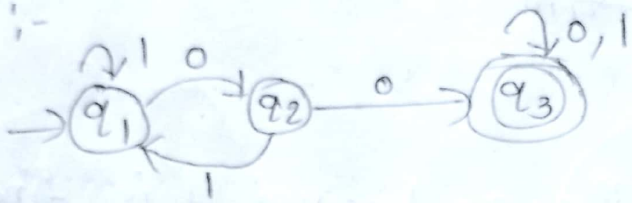
$$q_0 = q_0 \cdot 01 + q_0 \cdot 1 \cdot 0 + \epsilon$$

$$\underset{R}{q_0} = \underset{R}{q_0}\underset{P}{(01+10)} + \underset{Q}{\epsilon} \quad \rightarrow \text{this is in the form of } R = Q + RP$$

$$q_0 = \epsilon \cdot (01+10)^* \quad \rightarrow \boxed{\epsilon \cdot R^* = R^*} \qquad R = QP^*$$

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

EX2 :-



step1

$$q_1 = q_2 \cdot 1 + q_1 1 + \epsilon \rightarrow \text{step 2}$$

$$q_2 = q_1 \cdot 0$$

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

step3 Arden's algorithm

* the q_3 solutions for the Regular expression. but not matching

So replace all in $q_3 = q_2 + q_3$ also there.

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

$$\underset{R}{q_1} = \underset{P}{q_1(1+01)} + \underset{Q}{\epsilon}$$

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

$$= (1+01)^*$$

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

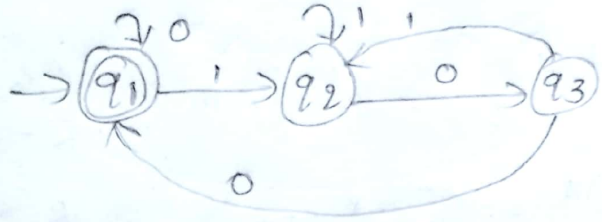
$$\underset{R}{q_3} = \underset{P}{q_1 \cdot 0} + \underset{Q}{q_3(0+1)}$$

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

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

Equivalence of DFA & Regular Expression

i.e conversion from DFA to R.E



input is DFA and output is R.E

$q_1 = q_1.0 + q_3.0 + \epsilon$

$q_2 = q_1.1 + q_2.1 + q_3.1$

$q_3 = q_2.0$

* we have to solve the q_3. because it is final solution to become the Regular Expression.

* The q_3 contain q_2 so first we find q_2. on substituting q_2 only we get the q_3.

$q_2 = q_1.1 + q_2.1 + q_3.1$

$q_2 = q_1.1 + q_2.1 + q_2.0.1$ ← substituting value

$q_2 = q_1.1 + q_2(1+0.1)$

R Q R P

$R = Q \cdot RP$

$R = QP^*$

$\boxed{q_2 = q_1.1\,(1+0.1)^*}$

* Now we can substitute q_2 value in q_3

$q_3 = q_1.0 + q_3.0 + \epsilon$ Then we get q_3 in terms of q_1. so let us directly substitute on $q_1 = q_1.0 + q_3.0 + \epsilon$

$q_1 = q_1.0 + q_2.0.0 + \epsilon$

$q_1 = q_1.0 + q_1.1\,(1+0.1)^*00 + \epsilon$

$q_1 = q_1\,(0 + 1\,(1+0.1)^*00) + \epsilon$

R R P Q

$q_1 = \epsilon + (0+1\,(1+0.1)^*00)^*$

$\epsilon \cdot R^* = R^*$

$q_1 = (0+1(1+0.1)^* 00)$

* so This is the Regular Expression for this DFA.

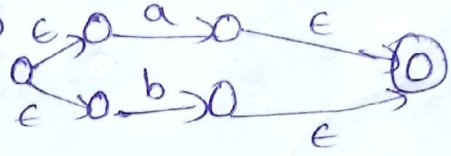
converting Regular Expression to finite Automata

* let the regular expression is ebsilon ϵ then the finite Automata is

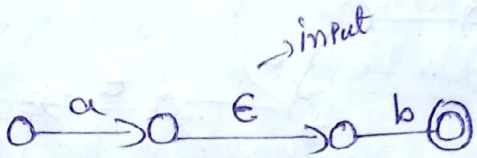
ϵ $\quad 0 \xrightarrow{\epsilon} (0)$

ex: a $\quad 0 \xrightarrow{a} (0)$

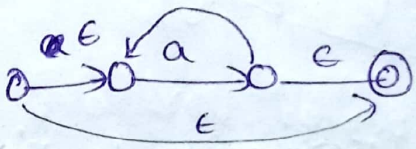
let the R.E is a+b



a.b



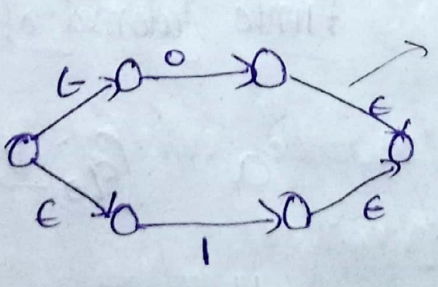
a^*



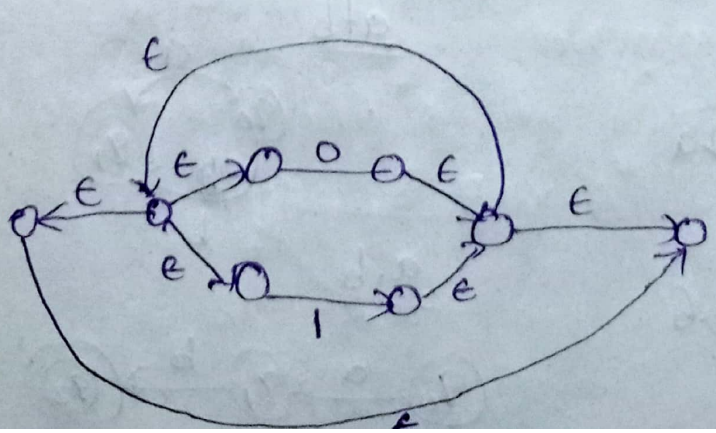
→ * mean zero occurance (a) any no. of occurances.

$(0+1)^*.0.1$

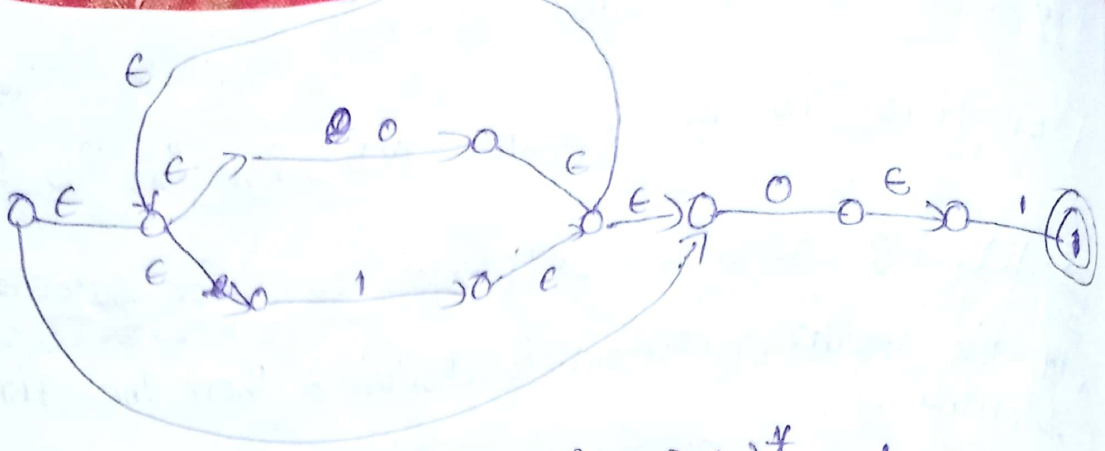
first we represent '0'. for representing zero we have 2 states



— (0+1) is over Nxt $(0+1)^*$

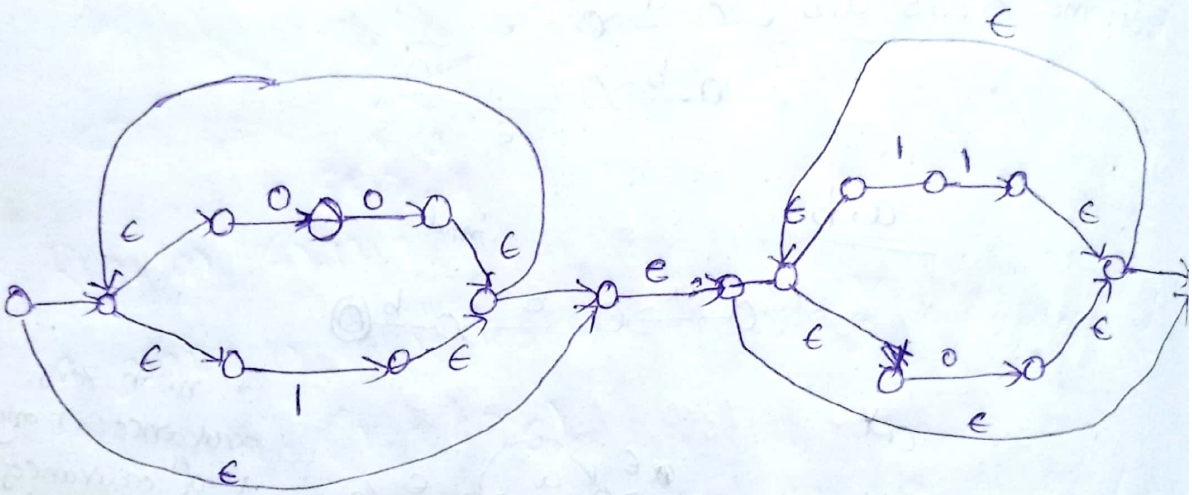


$(0+1)^*$ is over



So this is the diagram of $(0+1)^*.0.1$

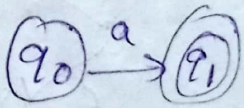
let the R.E is $(00+1)^*(11+0)^*$



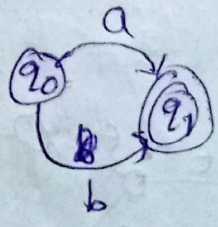
convert Regular Expression to Finite Automata by using Direct method

Ex1 : $a^*b(a+b)^*$

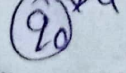
Finite Automat of a

a 

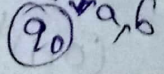
a+b



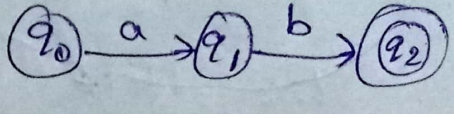
a^*



$(a+b)^*$

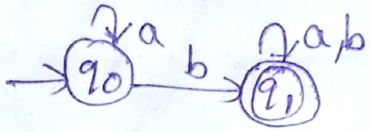


a,b

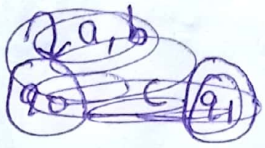


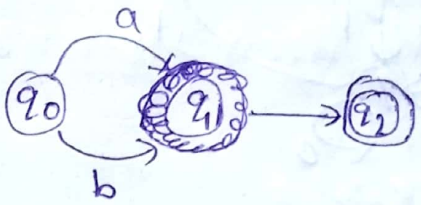
(a b)

Ex 1: $a^* b\ (a+b)^*$

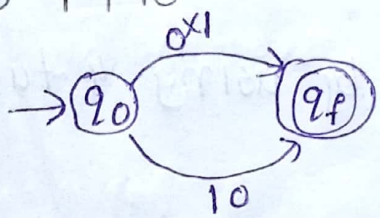


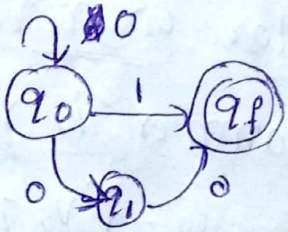
Ex 2: $(a+b)\ c$



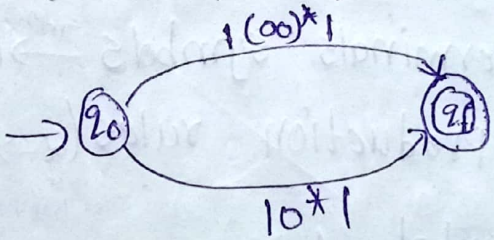


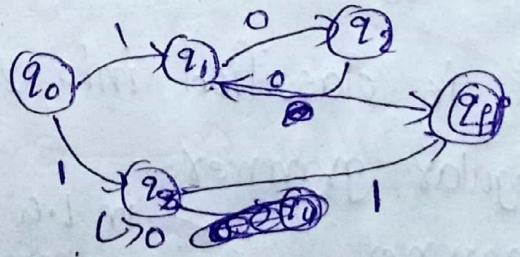
Ex 3: $0^* 1 + 10$



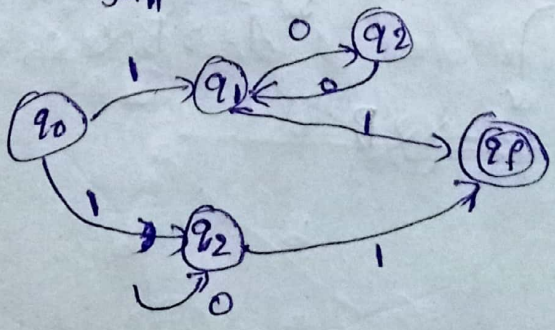


Ex 4: $1\ (00)^* 1 + 10^* 1$

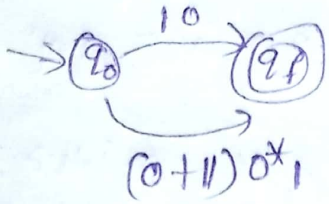


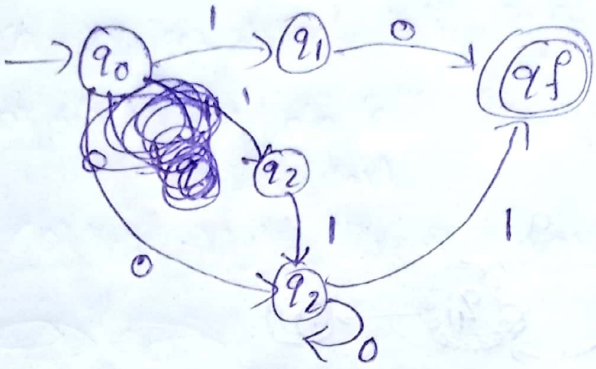


Neatly drawn again



Ex 5: $10 + (0+11)0^*1$





Regular Grammer

a grammer is represented by using 4 tuples

$G = (V, T, P, S)$

V → set of variables (or) Non terminals

T → set of terminals symbols → (lower case letters, bi

P → set of production rules $(\alpha \to \beta)$ → this form; α → left hand side, β → Right hand

S → start symbol $(A \to a)$

* Regular grammer is classified into 2 types.

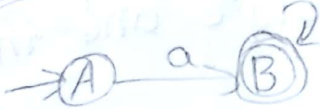
1. Left linear Regular grammer
2. Right linear grammer

Right linear grammer → $A \to \alpha B$

Left linear Regular grammer → in L.G in the form of $A \to B\alpha$

$A \to B\alpha$ (B: Non-Terminal, α: Terminal)

convert finite Automata to Right & left linear grammer



* conversion finite Automata to Right linear gramm is direct process

$A \rightarrow aB$ → This R linear gramer
→ in Right hand side Right most symbol is always a non-terminal.

$B \rightarrow aB$
$B \rightarrow bB$
in left grammer the left most symbol is a Non-terminal

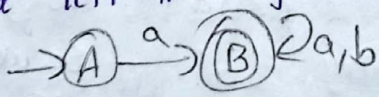
$B \rightarrow \epsilon$ → if it is final state we have to write ebsilon.

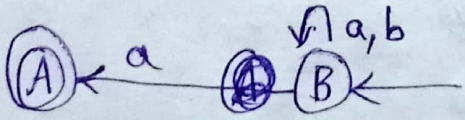
left linear grammer

we have to follow 3 steps

1. construct Reverse of F.A. → it mean make initial state as final state and final state as initial state & reverse the edges.
2. Then write Right linear Grammer.
3. The take Reverse of Right linear grammer.

* we follow these 3 steps to obtain left linear grammer. (or) construct left linear grammer.





→ 1. This is reverse of F.A

$B \rightarrow aAB$
$B \rightarrow bB$
$B \rightarrow aA$
$A \rightarrow \epsilon$

→ 2. Then write right linear grammer → simply to reverse right l.G

$B \rightarrow Ba$
$B \rightarrow Bb$
$B \rightarrow Aa$
$A \rightarrow \epsilon$

* you have 3 states (or) 4 states you may not change
* you just change the initial state and final state
* suppose in Examination, The Question given as convert Finite Automata to ~~Right~~ Regular Grammer
* so we construct both Right Linear & left linear gram

construction of left linear grammer

(or)

convert right linear grammer to left L.Grammer

construction of

$S \to bB$

$B \to bC$

$B \to aB$ } → these are Right linear gram

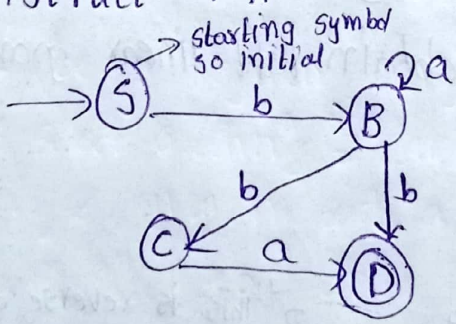
$C \to a$

$B \to b$

* we have to convert Right linear grammer to left L.G

Step 1 :-

construct F.A

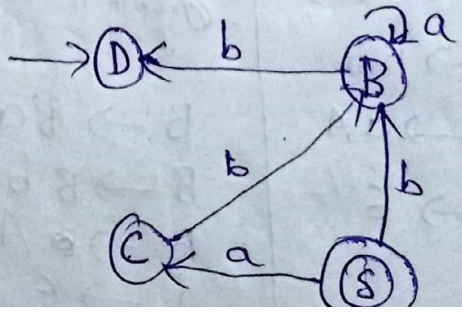


C → a → Non terminal st is given.

so we assume C is final state.

Step 2 :-

Exchange initial state & final state and Reverse the edges.



Step 3 :- write Right linear Grammer

$S \rightarrow bB$ → Right most Non-terminal.

$S \rightarrow ac$

$C \rightarrow bB$

$B \rightarrow aB$

$B \rightarrow bD$

Step 4 :-

write left linear Grammer

$S \rightarrow Bb$ → left most Non-terminal

$S \rightarrow ca$

$C \rightarrow Bb$

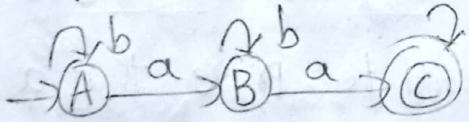
$B \rightarrow Ba$

$B \rightarrow Db$

Construction of Regular grammer for finite Automata

* The input is finite Automa output is Regular gramer

Ex1 :



$A \rightarrow aB$

$A \rightarrow bA$

$B \rightarrow bB$

$B \rightarrow ac$

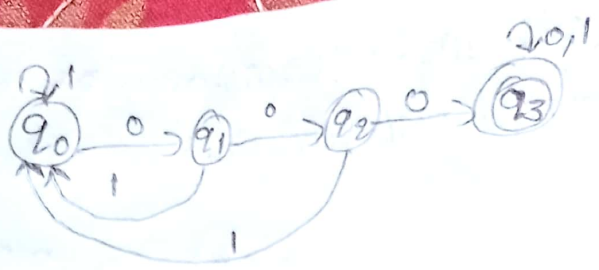
$B \rightarrow a$ → we reach the final state we have to right terminal symbol also.

$C \rightarrow bC$ → c is the final state, if we reaches the final state by applying an input symbol. Then without having the state also we can apply.

$C \rightarrow b$

* So This the Regular Grammer for the given F.A

Ex2:



$q_0 \rightarrow 0\,q_1$

$q_0 \rightarrow 1\,q_0$

$q_1 \rightarrow 0\,q_2$

$q_1 \rightarrow 1\,q_0$

$q_2 \rightarrow 0\,q_3$

$q_0 \rightarrow 1\,q_0$ $\times$ $q_2 \rightarrow 0$

$q_3 \rightarrow 0\,q_3$

$q_3 \rightarrow 1$

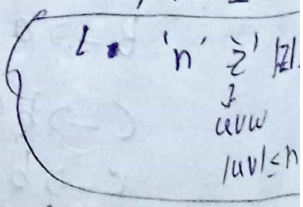
$q_3 \rightarrow 1\,q_3$

$q_3 \rightarrow 1$

Pumping Lemma for Regular languages

Pumping lemma is used to prove that a langu
is not Regular.

* let 'L' be a Regular language. then there is a consta
'n' s.t select a string 'z' such that $|z| \geq n$. Divide 'z' int
3 parts $z = uvw$ in such a way that $|uv| \leq n$, $|v| \geq 1$
and for all $i \geq 0$, uv^iw is in L.



Ex: prove that $L = \{a^n b^n \mid n \geq 1\}$ is not Regular.

$L = \{a^n b^n \mid n \geq 1\}$

* L be a Regular language

* we assume constant a n $a^1 = ab$ $b^2 = ab$

$L = \{ ab, aabb, aaabbb \ldots \}$ $n \geq 1$

we assume string as aaabb it is 6 length so we take aabb and nay 4.

let $n = 4$ → constant

$z = aabb$ → string, $|z| = 4 \geq 4$ ✓ → string length

a a b b
u v w

$|uv| \leq n$

$|aab| \leq n$

$3 \leq 4$ ✓

$|v| \geq 1$

$|ab| \geq 1$

$2 \geq 1$ ✓

$uv^i w$ $i=0$ belongs to

$i=0 \rightarrow a \cdot (ab)^0 \cdot b \Rightarrow ab \in L$

$i=1 \rightarrow a \cdot (ab)^1 \cdot b \Rightarrow ab.b \in L$

$i=2 \rightarrow a(ab)^2 . b$

$a.ab.ab.b \notin L$ → contradiction

↘ we got this but it is Not prsnt in L={}

* So, $\{a^n b^n / n \geq 1\}$ → is Not a Regular Language

2) Prove that $L = \{a^P / P$ is a prime$\}$ is not Regular.

* let L is a Regular language,
* let n is a integer constant.
* And we to select a string w from L such that $|w| \geq n$.

$L = \{aa, aaa, aaaaa \ldots\}$ → prime means 1, 3, 5..

let $n = 3$ → it is our choice

$z = w$ $z = aaa$ such that $|z| \geq n$

$|aaa| \geq 3$

$3 \geq 3$. ✓

* Divide z into 3 parts such as uvw, such that $|uv| \leq n$ and $|v| \geq 1$, for $i \geq 0$, uv^iw is in Language

$z = \underbrace{a}_{u}\underbrace{aa}_{v}\underbrace{a}_{w}$ $\quad |aa| \leq n$

$|aa| \leq 3$

$2 \leq 3$ ✓

$|v| \geq 1$

$|a| \geq 1$

$1 \geq 1$ ✓

uv^iw

let $i=0 \rightarrow a \cdot a^0 \cdot a = a.a \in L$ ✓ → aa present in language

$i=1 \rightarrow a \cdot a^1 \cdot a = aaa \in L$ ✓ prime means 1,3,5

$i=2 \rightarrow a \cdot a^2 \cdot a = aaaa \notin LX$ → 4 aas not present in

↓ contradiction. This is not

Show that $L = \{0^{n^2} / n \geq 0\}$ is not a Regular.

Sol) let 'L' be the regular language. n be a constant

let 'z' be a string from L such $|z| \geq n$

$L = \{\epsilon, 0, 0000, 00000000 \ldots\}$

$n=1 \quad 0^{n^2} = 0$

$n=2 \quad 0^{2^2} = 0^4 = 0000$

$n=3 \quad 0^{3^2} = 0^9 = 000000000$

let $n=2$

$z = 0000$

$|z| \geq n$

$0000 \geq 2$

$4 \geq 2$ ✓

* Divide the string into 3 parts $z \rightarrow (uvw)$ in sucha way that $|uv| \leq n$, $|v| \geq 1$

for $i \geq 0$, uv^iw is in L

0000 $|uv| \leq n$

u v w

$|oo| \leq 2$

$2 \leq 2$ ✓

$|v| \geq 1$

$|o| \geq 1$

$1 \geq 1$ ✓

$uv^iw \Rightarrow 0 \cdot 0 \cdot 00$

initially assume $i = 0$

zero pow zero means ebsilon

$i=0 \Rightarrow 0 \cdot \overset{0}{\theta} \cdot 00 \Rightarrow 0 \cdot \overset{x}{\epsilon} \cdot 00 \Rightarrow 00 \notin L$

Triple zero are not present in L

contradiction

* So we say that This language is not Regular language

* Suppose we got 4 zeros (0000). it is prsnt in language so we got $i=1$ ~~likewise~~. Nxt check $i=2$. until the language is not Regular

Show that a language of balanced parenthesis is not Regular?

balanced parenthesis means every left parant should have right parenthesis

* let 'L' be ~~regular~~ regular language, n be a constant
* let 'z' be a string from L such $|z| \geq n$.

$L = \{ (), (()), ()(), \ldots \}$

balanced para, balanced pair, balanced pair

let $n = 2$ $z = (())$ $|z| \geq n$

$|(())| \geq 2$

$|4| \geq 2$ ✓

* Divide the string 'z' into 3 parts such as uvw such that $|uv| \leq n$, $|v| \geq 1$, for $i \geq 0$ uv^iw in

(( ))
uv w

$|uv| \leq n$

$|cc| \leq n$

$|2| \leq 2$ ✓

$|v| \geq 1$

$|c| \geq 1$

$|1| \geq 1$ ✓

, uv^iw

$\Rightarrow i = 0 \rightarrow uv^0w = ((^{0}\,))$ (ϵ) $\Rightarrow (\,))$ → left paranthesis, Right par Right not present in L

$(\,)) \notin L$. ↘ contradiction

* Closure properties of Regular language

Regular language closed under

- union
- concatenation
- klene closure
- complement
- intersection
- Reversal.

union :- Let L_1 & L_2 are 2 Regular languages then $L_1 \cup L_2$ is also a Regular language

↘ union

Ex1: $L_1 = a^*$ $L_2 = b^*$

$L_1 \cup L_2 = a^* + b^*$ (or) $a^* \cup b^*$

any no. of a's any no of b's, This is also a Regular language. if L_1 & L_2 are Regular lang Then we can say that $L_1 \cup L_2$ are also Regular lang

concatenation

if L_1 & L_2 are 2 Regular languages Then we can say that $L_1 \cdot L_2$ is also a Regular language

Ex1:- L_1 = a^* $L_2 = b^*$

Regular lang $L_1 \cdot L_2 = a^* \cdot b^*$ is closed under concentanation also.

Kleene closure

if L is a Regular Language then L^* is also a Regular language.

Ex1: $L = a$

$L^* = a^*$

so a^* is also a Regular language. we can say that Regualar lang closed under Klenc closure

Complement

if L is a Regular language Then $\bar{L}$ is also a R.L

Ex1: L: set of all strings containing 1100 as substring

Then $\bar{L}$ will becomes

$\bar{L}$: set of " " not containing 1100 " "

* we can represent with these help of Finite Automata (F.A)

You know what is a Regular language. The language which is accepted by finite Automata.

* we can say that Regular language are closed under complement.

Intersection :-

if L_1 & L_2 are two Regular languages Then $L_1 \cap L_2$ (intersection) is also a Regular language.

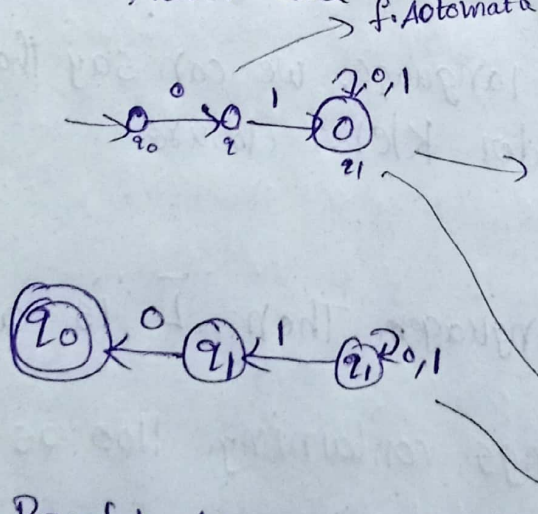
Ex: $L_1 = a^*$ $L_2 = b^*$;

$L_1 \cap L_2 = a^* \cap b^*$

Reversal :-

if L is a Regular language Then Reversal of L^R is also a Regular language.

* In Reversal what we have to do is. let us take a finite Automata



reversal of this F.A is change final state to initial state and initial state to final state & Reverse the direction.

This a Regular language
This is also a Regular langua

* so Regular language also closed under Reversal.