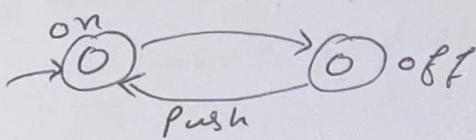
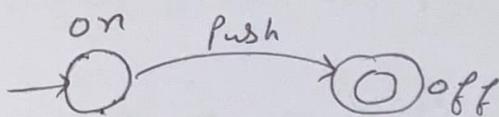
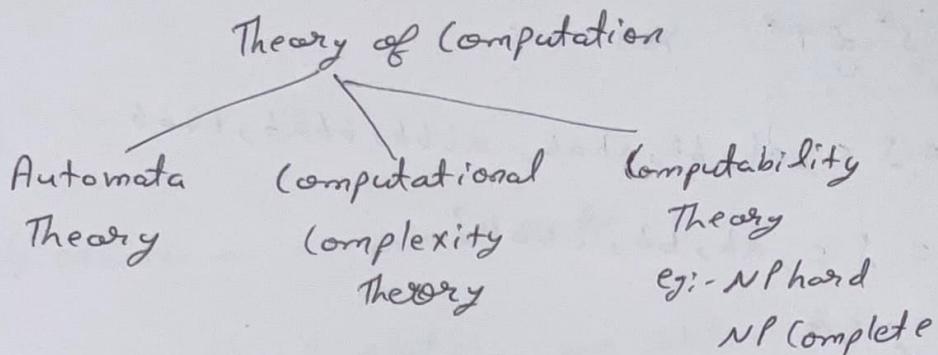


Automata Theory

Tokens:- is just like syntax



$$\Sigma^0 = \{\epsilon\} \xrightarrow{\text{Epsilon}} \text{null string} \quad \Sigma^1 = \{a, b\} \quad \Sigma^2 = \{aa, ab, ba, bb\}$$

$$\Sigma^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

$*$ → Kleene's closure

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$$

star → It is always be a infinite set

$$= \{\epsilon, ab, aa, ba, \dots\}$$

$$\Sigma^+ = \Sigma^* - \{\epsilon\} \quad (\Sigma^*)^+ = \Sigma^* \Rightarrow (\Sigma^*)^+ \subseteq \Sigma^* \Rightarrow \Sigma^+ \subseteq (\Sigma^*)^+$$

\downarrow
positive closure

$(\Sigma^+)^+ = (\Sigma^*)^+$ when $\Sigma \in \Sigma^*$ belong to the set so they are True
 $\therefore \Sigma \notin \Sigma^*$ not belong " " " " False

$$T = \{aa, ab\}$$

$$T^* = \{ \epsilon \} \cup \{aa, ab, aaaa, aaab, abaa, abab\}$$

$S = \{ab, bb\}$

$T = \{ab, bb, bbbb\}$

Is $S^* = T^*$ Yes

$S^* = \{\epsilon, ab, bb, abab, abbb, bbab, bbbb \dots\}$

$T^* = \{\epsilon, ab, bb, bbbb\}$

$S^* \subseteq T^*$

$T^* \subseteq S^*$

$S^* = T^*$

If $T = \{ab, bbb, bb\}$

Is $S^* = T^*$ No

because we can't make bbb from the S that's why
it is not equal

Natural no is countable infinite

(countable infinite ?)

Finite Automata (FA)

DFA

Deterministic

NFA

or

NFA

Non-deterministic

If we write FA so it is DFA

An FA is denoted by quadruple $(\mathcal{Q}, \Sigma, F, \delta)$ on $\Sigma = \{a, b\}$

$\mathcal{Q} = \text{finite set of states} = \{q_0, q_1, q_2, \dots, q_n\}$ $\overset{q_i}{\circ}$ (state)

$q_0 \in \mathcal{Q}$

\downarrow
would be

single element
initial state

q_0 is usually chosen as an initial state

$\rightarrow \circ, \ominus$
(initial state)

② $F \subseteq \mathcal{Q}$ = set of final states, end state, terminal state

If there is no final state then we can take $q_0, q_0 q_1, \dots$
as as final state

If initial state is \ominus then final state is \oplus

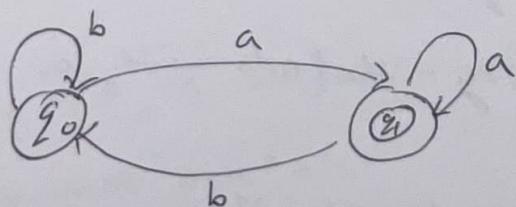
δ is a transition function

$\delta(q_i, a) = q_j$

$\downarrow \quad \rightarrow$
initial state after reading some element
 \rightarrow we go to q_j

if $q_i = q_j$ so it is loop edge
OR

$\delta : \mathcal{Q} \times \Sigma \rightarrow \mathcal{Q}$



$$\mathcal{Q} = \{q_0, q_1\}$$

$$\delta = q_0 \in \mathcal{Q}$$

$$F = \{q_1\} \subseteq \mathcal{Q}$$

δ	a	Σ
$\rightarrow q_0$	q_1	q_0
(q_1) final state	q_1	q_0

sign is not in inside
the table it is only
on left side

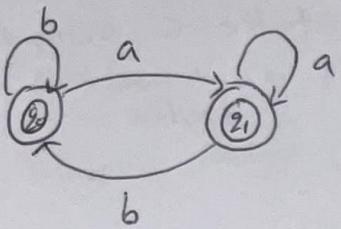
Transition table

- Here we find the strings that accept this automata
- is table me saari entry hona or 1-ki entry hona sabki
- it is sign of Deterministic
- if some columns are empty ~~and~~ or if one cell contain 2 entries (q_0, q_1) like so that is non-Deterministic

L = {a, aa, ba, aaa, aba, baa, bba, baababa---?}
 language accepted by FA

- Step1:- is ϵ take us to final state
 Step2:- is any state take us to final state
- final state is that state that accept string
- FA accept set of all strings on $\Sigma = \{a, b\}$ such that the string that ends with a

$L = \{s \in \{a, b\}^*: s \text{ must ends with } 'a'\}$



when we make every state as final state

$$L = \{ \epsilon, a, b, aa, ab, ba, bb, \dots \}^* = \Sigma^*$$

so it is

$$\Sigma^* = \{a, b\}$$

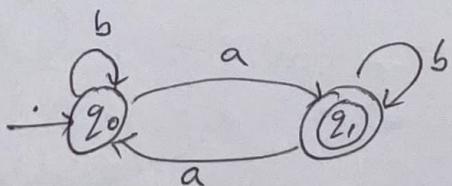
we also achieve this by $\rightarrow \textcircled{20}^{a,b}$

$\rightarrow \textcircled{20} \quad L = \{\epsilon\} \text{ (empty) NULL string}$

$\rightarrow \textcircled{20}^{a,b}$ it is also finite automata because $F \subseteq \Omega$
 $\phi \subseteq \Omega$
 $L = \phi$ null language

$\& L = \{a, ab, ba, aaa, abb, bab, bba, aaab, aaba, abaa, abbb\}$

This is finite automata of odd 'a's



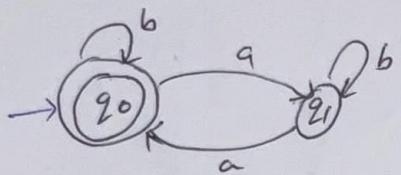
& in DFA is unique transition on signal letter

To make even 'a's

make all final state as nonfinal and all non-final states as final so it is negation of above.

If initial and final are same then we take ϵ without
 see anything

\downarrow
Epsilon



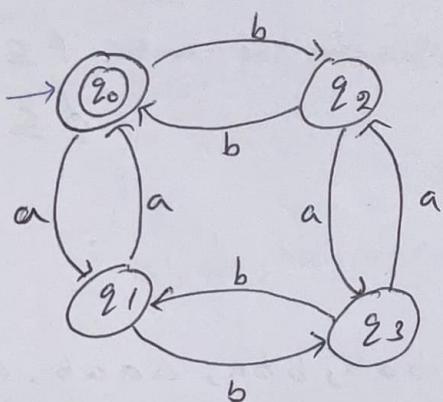
$$L = \{ \epsilon, b, aa, bb, aab, abba, baa, bbb, \dots \}$$

3

It is even a's automata

$$\text{Q } L = \{ \epsilon, aa, bb, aaaa, bbbb, abab, baba, abba, baab, bbaa, aabb, \dots \}$$

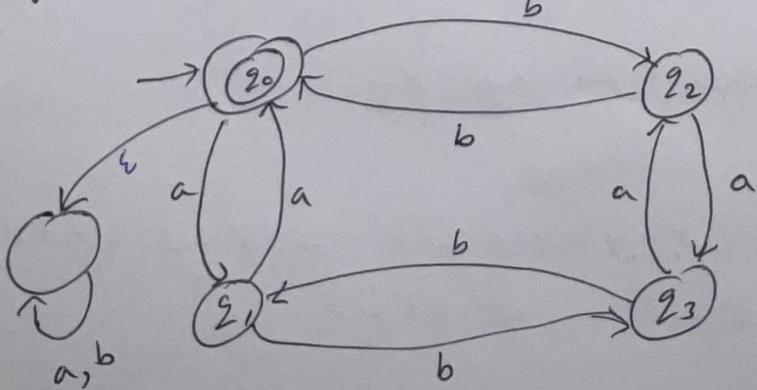
3



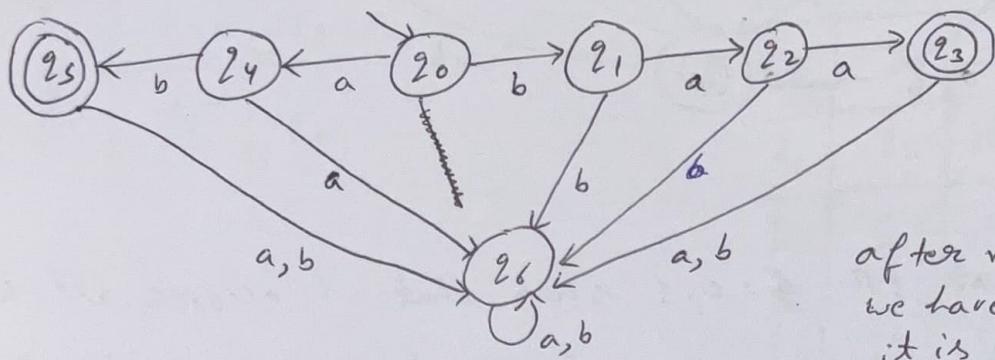
EVEN-EVEN FA

If we have ϵ so we can say our initial and final state are same

if we don't have ϵ so



Q Construct an FA which accepts only 2 strings which are bba, ab only (Specialized FA)

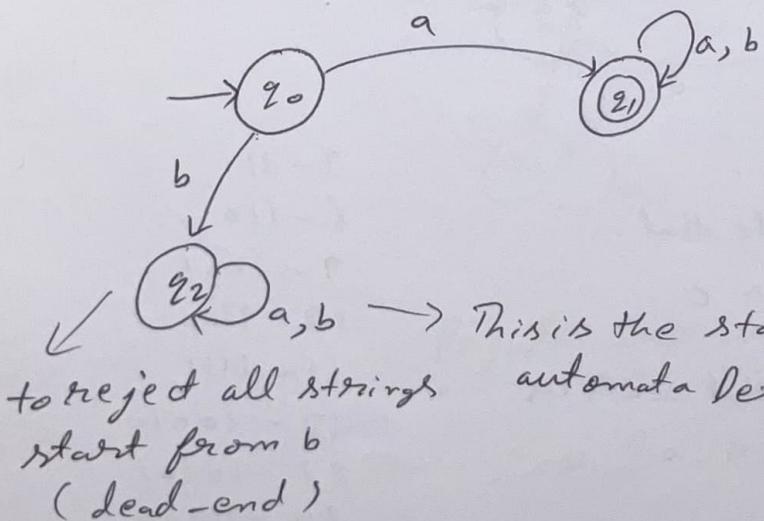


after making states(FA)
we have to make sure
it is DFA

if we can't make the dead-end state so
when we make table we have many blanks
so that its NFA so to make it DFA we have to
make dead-end state

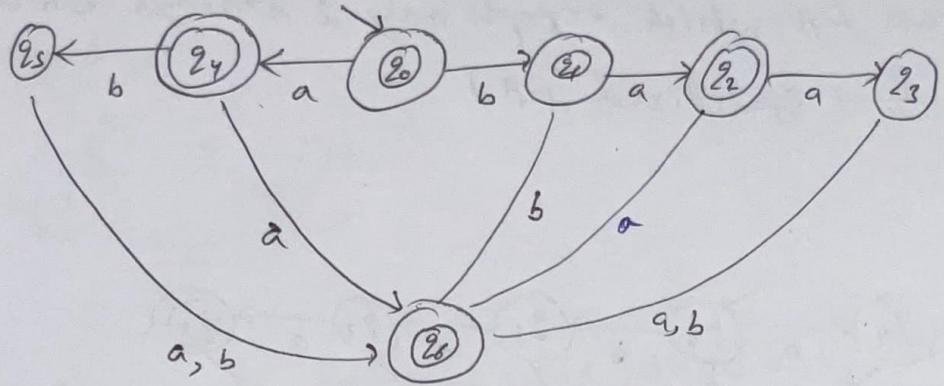
Q All the strings which starts with A

$$L = \{ a, aa, ab, aaa, aab, aba, abb, \dots \}$$



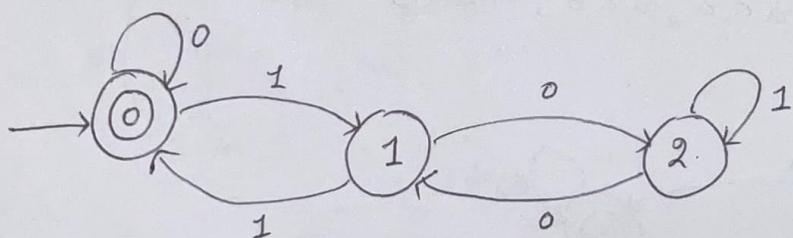
✓ q_2 a, b → This is the state which make this
to reject all strings automata Deterministic
start from b
(dead-end)

Q Construct an FA which don't have 2 strings which are baa, ab only (Specialized FA)



Q Construct an FA $\Sigma = \{0, 1\}$ such that it accepts all the bit strings which when divided by 3 should lead remainder 0.

Note!:- states are equal to remainder



Step 1 :- first make final state that is which give remainder 0

Step 2 :- Then we go to 11(3) that leads us to 0 (remainder)

Step 4 :- Then we go to 100(4) that leads us to 1 (remainder)

Step 5 :- Then we go to 101(5) that leads us to 2 (remainder)

So Now our FA is complete.

$3 - 11$
 $6 - 110$
 $9 - 1001$
 $12 - 1100$
 $15 - 1111$
 $18 - 10010$
 $21 - 10101$
 $24 - 11000$
 $27 - 11011$

Transition table

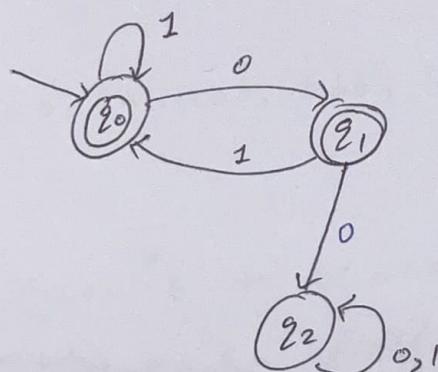
Σ	0	1
q_0	$q_0 \xrightarrow{0} q_1$	
q_1	$q_2 \xleftarrow{1} q_0$	
q_2	$q_1 \xrightarrow{0}$	$q_2 \xrightarrow{1}$

trick

for 4 states no is divide
by 4

Σ	0	1
q_0	$q_0 \xrightarrow{0} q_1$	
q_1	$q_2 \xleftarrow{1} q_3$	
q_2	$q_0 \xleftarrow{0}$	$q_1 \xrightarrow{1}$
q_3	$q_2 \xleftarrow{0}$	$q_3 \xrightarrow{1}$

Q Construct an FA on $\Sigma = \{0, 1\}^*$ such that
there is no two consecutive 0's



$$L = \{ \epsilon, 0, 1, 01, 10, 11, 011, 111, \dots \}$$

Regular Expression

$$\Sigma = \{a, b\}^*$$

Regular set/
language

ϵ, a, b, ϕ
 $\downarrow \quad \downarrow \quad \downarrow \quad \searrow$
 $\{\epsilon\} \cup \{a\} \cup \{b\} \cup \{\phi\}$
 \downarrow
 set containing empty string

$re_1 + re_2$ = Union of both Regular expression and duplicate removed

$re_1 \cdot re_2$ = Concatination and Remove duplicate

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

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

$$L_1 = \{ a, ab, aa, aaa, aba, abb, aab, \dots \}$$

$$L_2 = \{ b, ba, bb, bbb, baa, bba, \dots \}$$

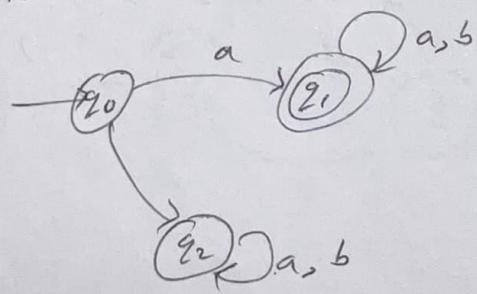
$$L_1 \cdot L_2 = \{ ab, aba, abb, abbb, abba, \dots \}$$

$$re^* =$$

$$(a, b)^* = \{ \epsilon, a, b, aa, ab, ba, bb, \dots \}$$

(+, ., *) \rightarrow are only allowed in Regular Expression

o Automata that start with a



o make a regular expression from pre^r automata

Q Construct a regular expression that do not except ab (which do not contain ab as substring)

$$L = \{ \epsilon, a, b, aa, ba, bb, aaa, baa, bba, bbb, \dots \}$$

$$\text{Ans } b^* \cdot a^*$$

$$b^* = \{ \epsilon, b, bb, bbb, \dots \}$$

$$a^* = \{ \epsilon, a, aa, aaa, \dots \}$$

$$\{ \epsilon, b, bb, bbb, \dots \} \cup \{ \epsilon, a, aa, aaa, \dots \}$$

Q Construct a regular expression that begins and ends with different letters

$$L = \{ ab, ba, aab, abb, baa, aaab, aabb, abbb, abab, \dots \}$$

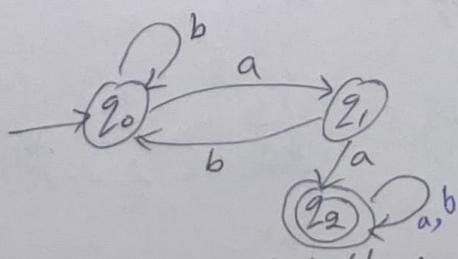
$$a(a+b)^*b + b(a+b)^*a$$

Note:- a b ya ba first end ho kuch bhi ho.

Q Construct a regular expression that it must has aa as a substring

$$L = \{ aa, aaa, aab, baa, aaaa, baab, baaa, aaab, \dots \}$$

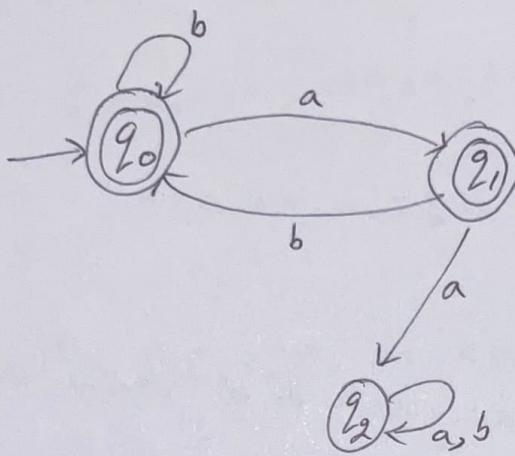
$$(a+b)^* aa (a+b)^*$$



Note! aa is mandatory left kuch bhi ho ya right kuch bhi ho

Q Construct a regular expression that do not except
aa (which do not contain aa as substring)

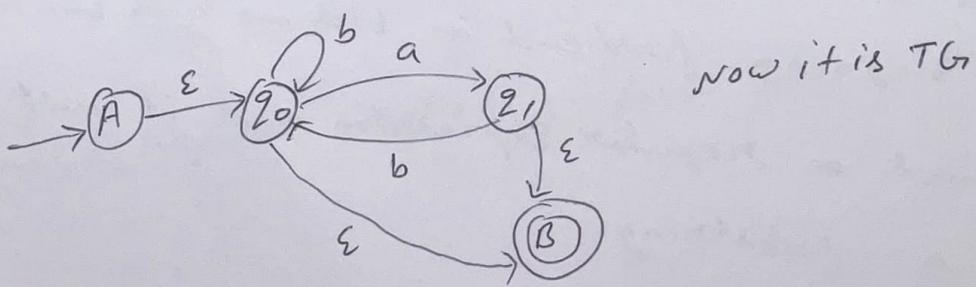
$$L = \{ \epsilon, a, b, ab, ba, bb, aba, bba, abb, bbb, bab, \dots \}$$



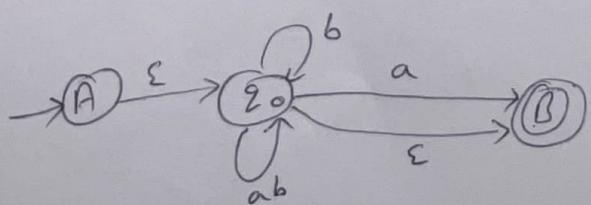
By using Bypass theorem

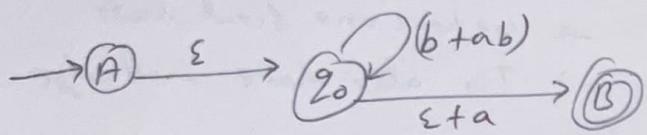
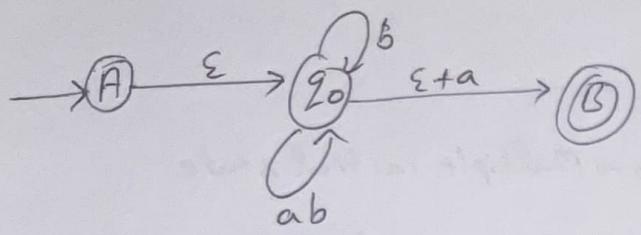
Step 1: eliminate q_2 edge because it goes to deadend state

Step 2: ^{Check if} Initial state is unique or not ~~irreducible~~

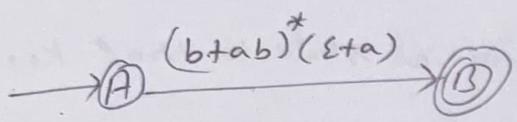


Now Bypass 2,





By part 20



we also write it like this

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

$TG \rightarrow$ Transition graph

FA \rightarrow Has one initial state

→ Every FA is TG, but
every TG is not FA

→ FA has no null edge

TG \rightarrow Multiple initial state

→ one initial state

→ we have one final state

→ They also have null edge

→ The should have non-

reenterable unique initial

state

initial state $k_1 t_{\text{ref}} k_0$

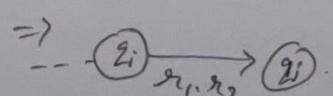
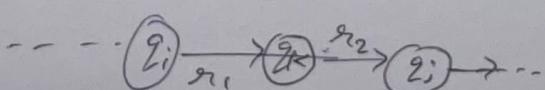
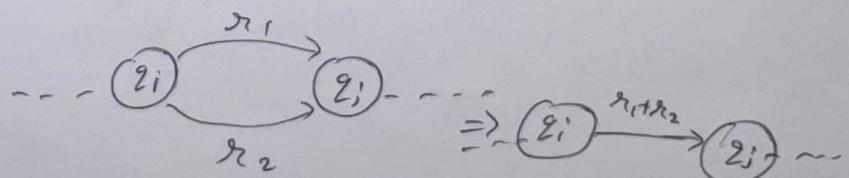
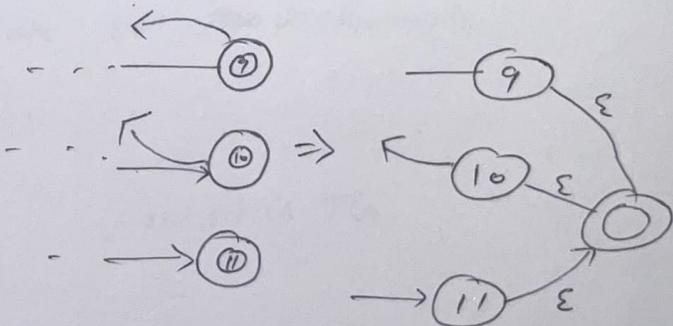
edge nahi aane chyea

→ Ke alava

→ we should have unleavable

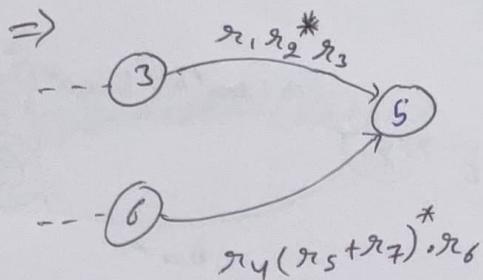
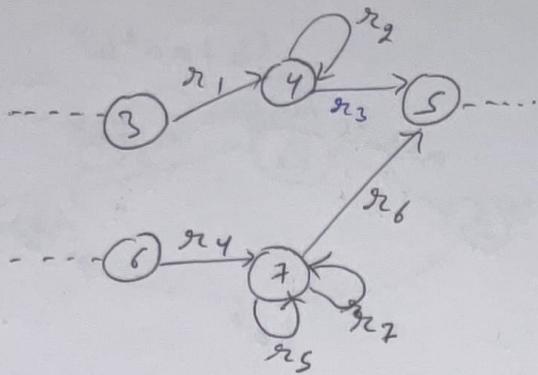
final state (final state se

Koi edge bhae nahi jaae chyea)

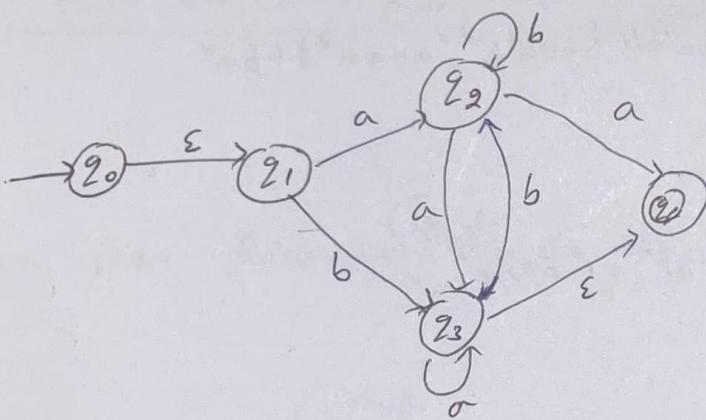


$$r_1, r_2 \neq r_3, r_1$$

$$r_1 + r_2 = r_2 + r_1$$



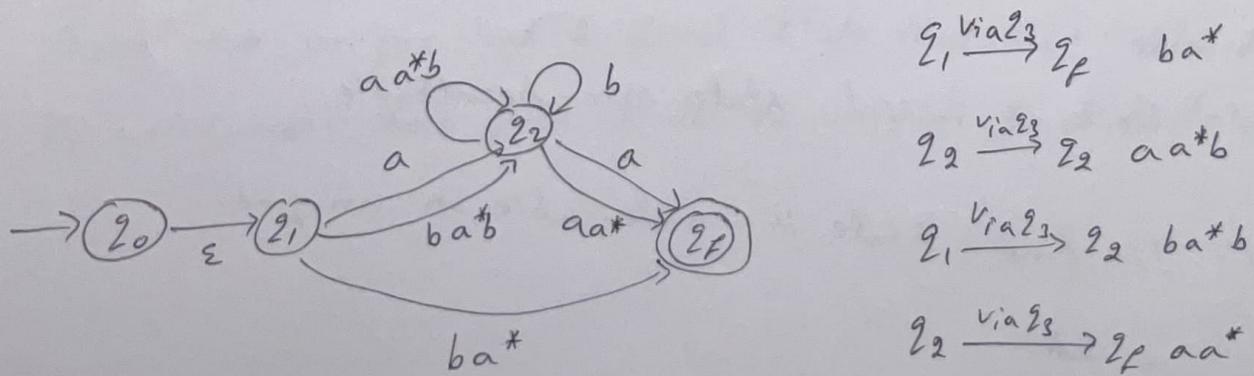
Q

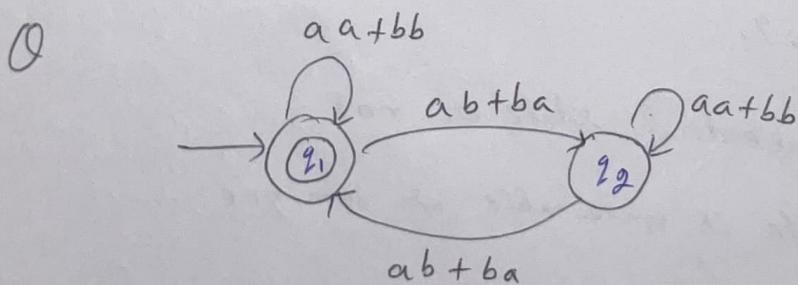
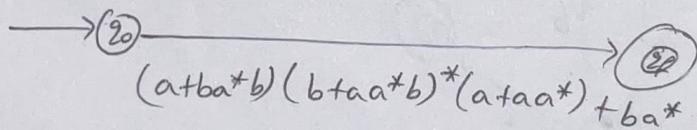
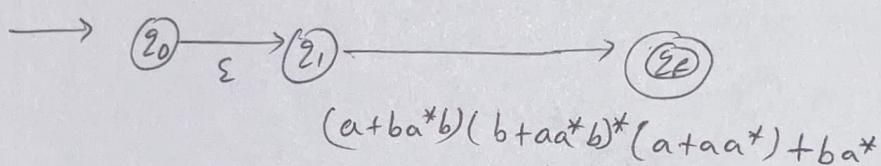
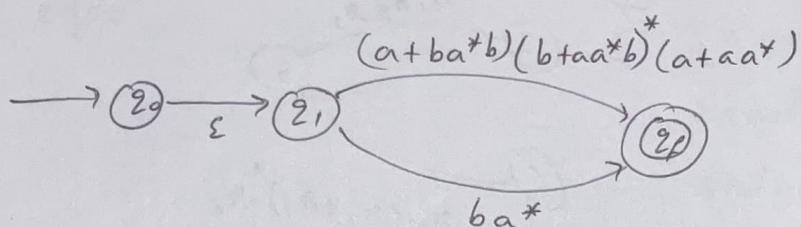
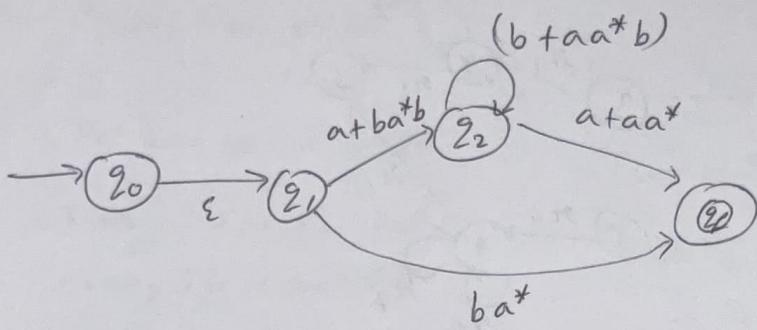


Eliminate 23, 22, 21

Step 1:- There is non-reachable state or not

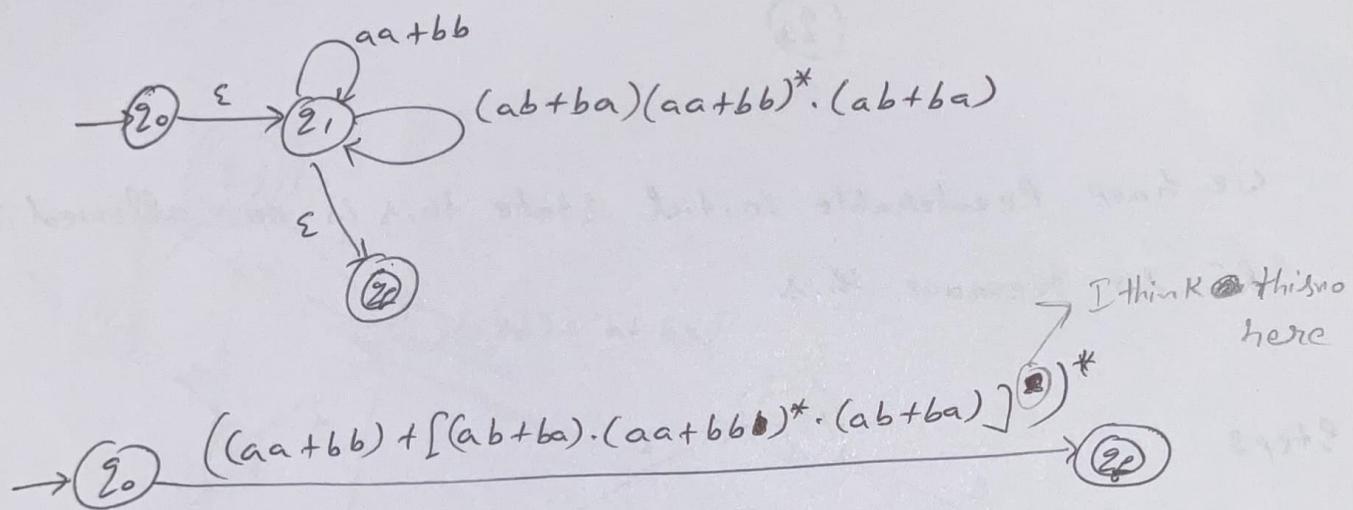
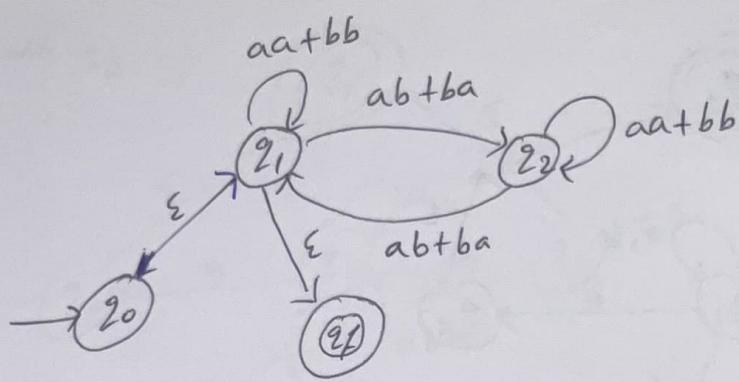
Step 2:- Is final state is unreachable or unique



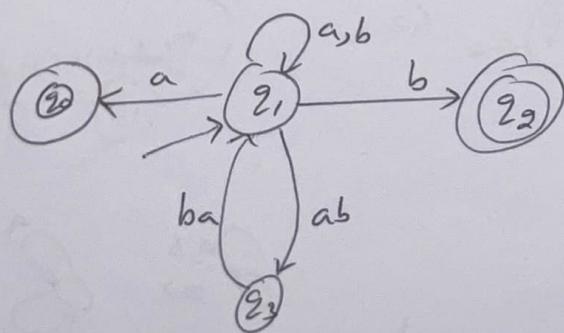


Step 1: There is initial state non-reachable

Step 2: Its final state is unreachable or unique

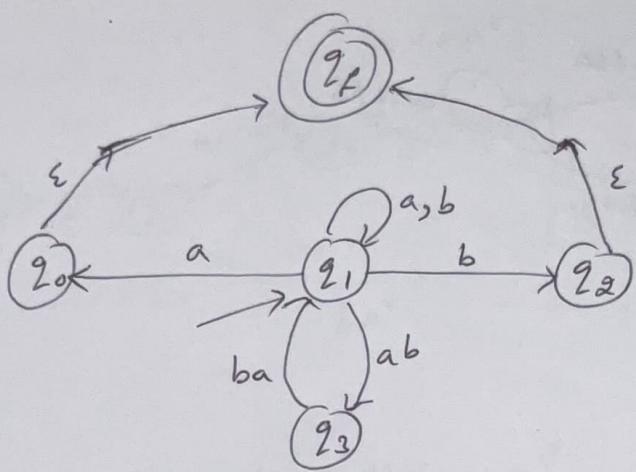


Q Convert the following TB to RE



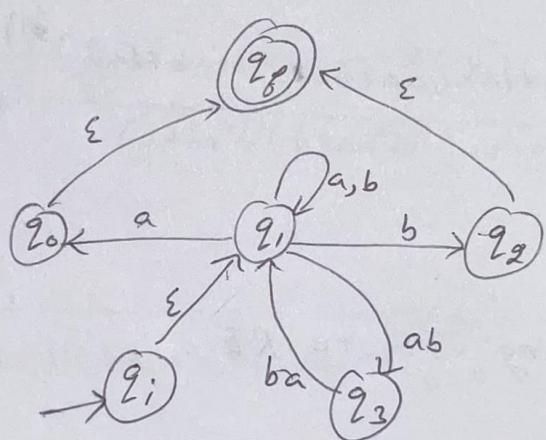
There are unique but 2 final state that not allowed in RE
So we convert these final states to unique final state

Step 1



We have Reenterable initial state this is not allowed in RE so remove this

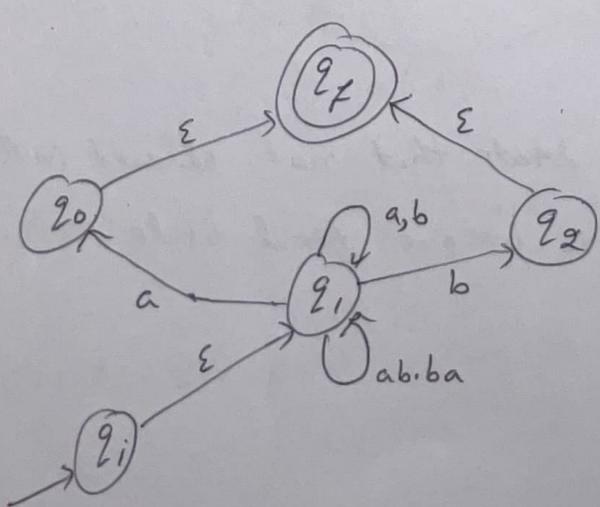
Step 2



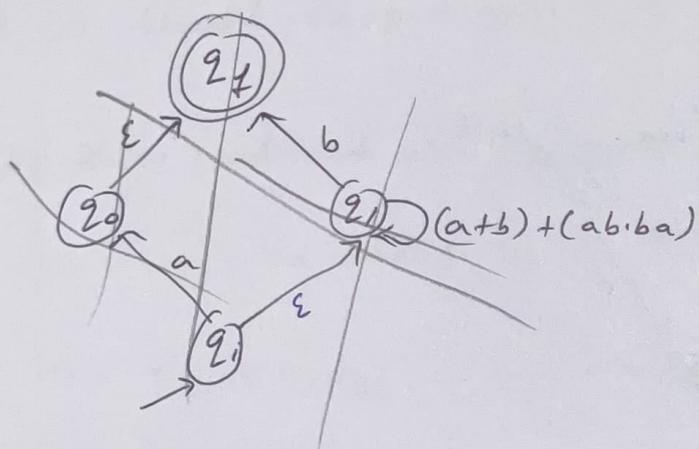
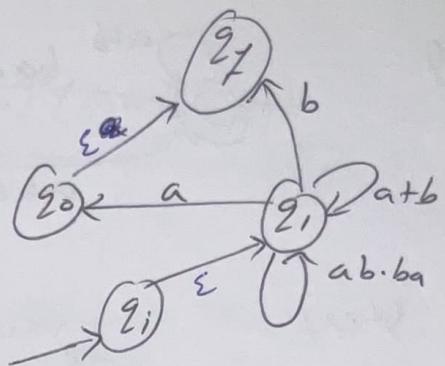
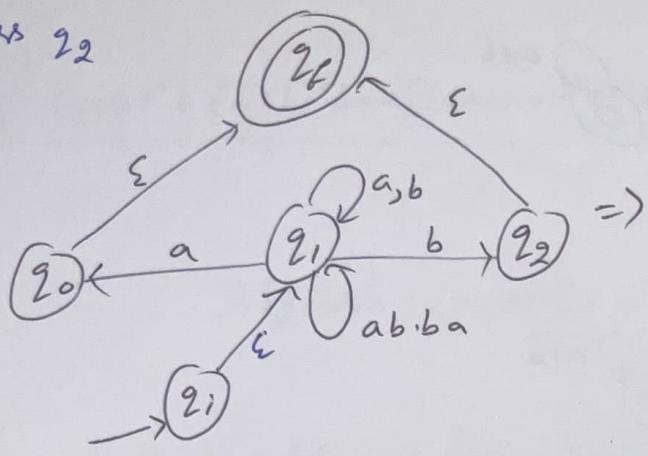
Step 3

By Pass q_3

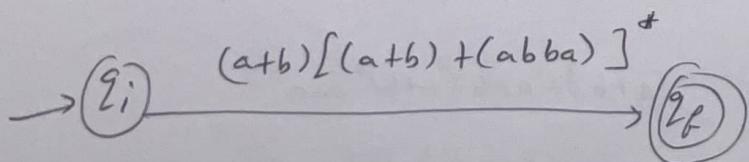
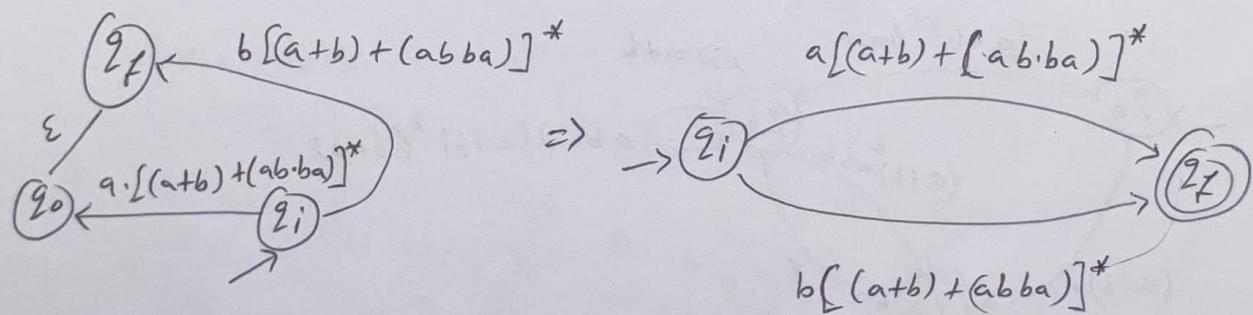
$$\beta^{a,b} \Rightarrow \begin{array}{c} a \\ \diagdown \\ q_3 \\ \diagup \\ b \end{array} \Rightarrow \beta^{a+b}$$

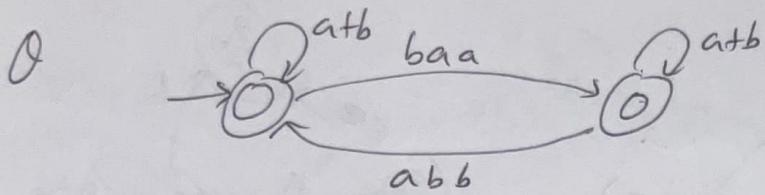


By parts 22

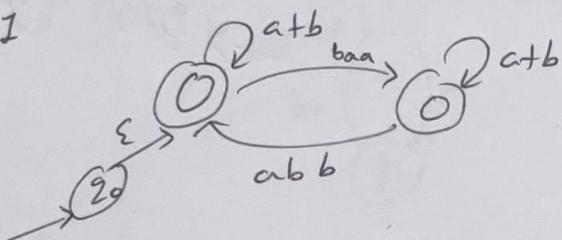


Step 4 :- Byparts 2,

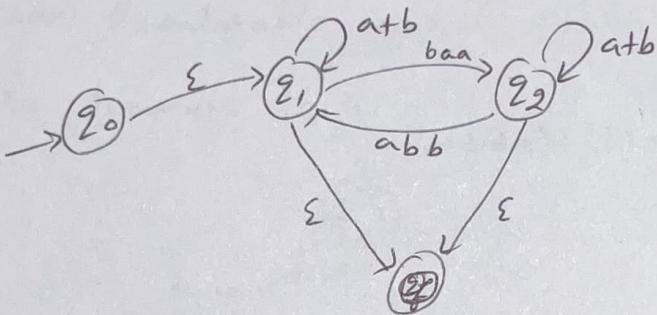




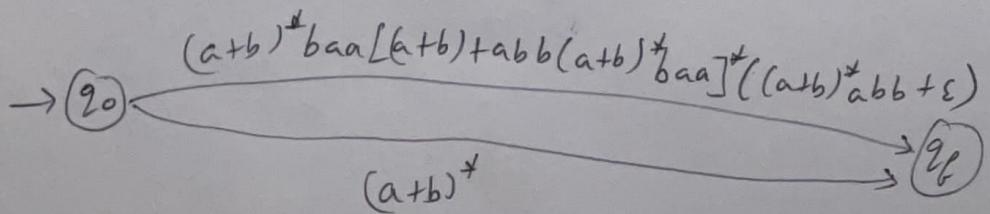
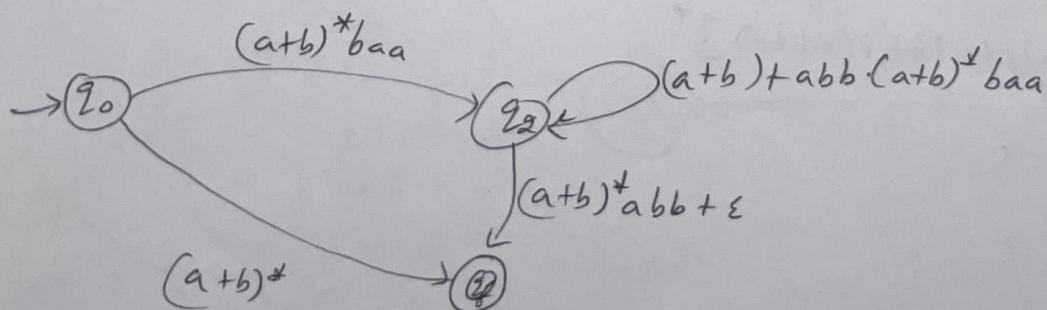
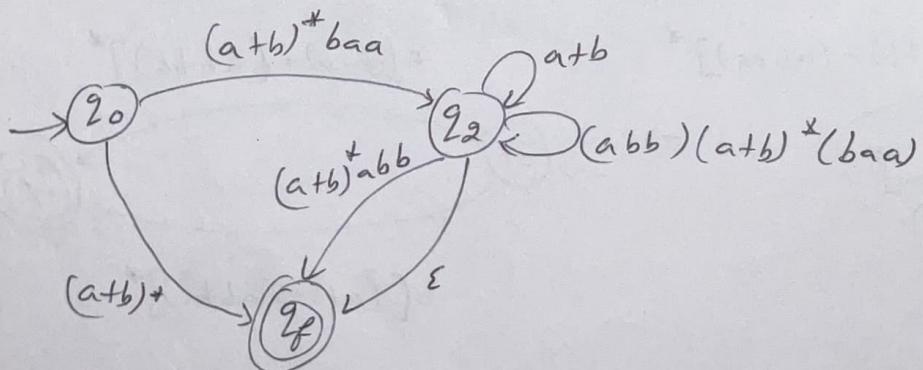
Step 1



Step 2



Bypass 2,



$$\xrightarrow{(20)} (a+b)^* + (a+b)^* baa [(a+b) + abb(a+b)^* baa]^* ((a+b)^* abb + \epsilon) \xrightarrow{(22)}$$

Algebra Expression using :- Ardan's Theorem

- (1) The graph should not have null node
- (2) It should have single initial state

If these conditions are true, then we can take vertices

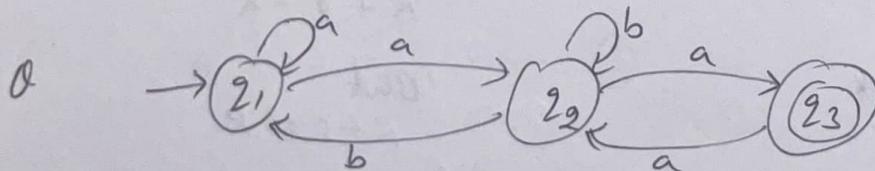
v_1, \dots, v_n where v_1 is the initial state then

$$v_1 = v_1 \alpha_{11} + v_2 \alpha_{21} + \dots + v_n \alpha_{n1} + \epsilon$$

$$v_2 = v_1 \alpha_{12} + v_2 \alpha_{22} + \dots + v_n \alpha_{n2}$$

⋮

$$v_n = v_1 \alpha_{1n} + v_2 \alpha_{2n} + \dots + v_n \alpha_{nn}$$



$$q_1 = q_1 a + q_2 b + \epsilon$$

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

$$q_3 = q_2 a$$

Property :-

$$(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^* \quad \text{--- (P1)}$$

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

$$\text{Ex: } R = \epsilon + 1^*(011)^* (1^*(011)^*)^* \quad \text{--- (1)}$$

Prove that the language represented by the regular expression above is the set of strings in which every zero is immediately followed by at least two one's

$$\text{Let } R_1 = 1^*(011)^*$$

$$R = \epsilon + R_1 R_1^* \text{ from above}$$

From property

$$R = R^*$$

$$= (1^*(011)^*)^*$$

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

$$R + \emptyset = R$$

But

$$R + \epsilon \neq R$$

from (P1)

$$= ((1^*)^* + ((011)^*)^*)^* = (1 + 011)^*$$

Remember Properties

$$1. (R^*)^* = R$$

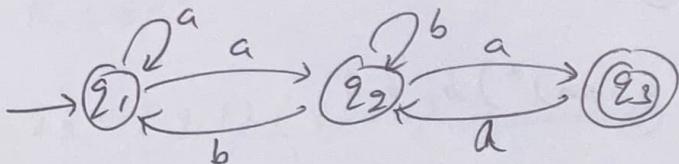
$$2. \varepsilon + RR^* = R^* = \varepsilon + R^*R$$

$$3. R + \phi = R \text{ But } R + \varepsilon \neq R$$

$$4. (P + Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$$

$$5. (P + Q)R = PR + QR$$

Q



$$q_1 = q_1 a + q_2 b + \varepsilon \quad \text{--- (1)}$$

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

$$q_3 = q_2 a \quad \text{--- (3)}$$

Substituting (3) in 2

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

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

Now from Arden's theorem:-

let P & Q are two RE over ε . If P does not contain ε then the following equation in R(RE)

$$R = \delta + RP$$

has a unique solution given by

$$R = \delta P^*$$

①

$$\overline{R} = \overline{\delta} + \overline{a} \overline{R} + \overline{aa} \overline{P}$$

$$\overline{R} = (\overline{\delta} + \overline{a}(\overline{b} + \overline{aa})^*)^* \quad \text{--- (7)}$$

Substituting (7) in ①

$$\overline{R} = \overline{\delta} + (\overline{\delta} + \overline{a}(\overline{b} + \overline{aa})^*) \overline{b} + \varepsilon$$

$$\overline{R} = \varepsilon + \overline{\delta} + \overline{a} \overline{R} + \overline{a}(\overline{b} + \overline{aa})^* \overline{b}$$

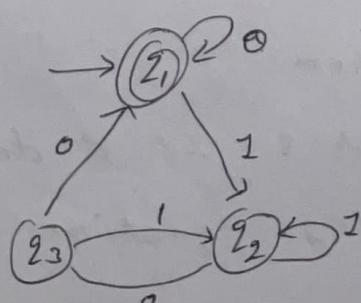
Using Arden's theorem

$$\overline{R} = \varepsilon \cdot (\overline{a} + \overline{a}(\overline{b} + \overline{aa})^* \overline{b})^* \quad \text{--- (5)}$$

$$\text{Putting } \overline{R} \text{ in } \overline{q} \quad \overline{q}_2 = (\overline{a} + \overline{a}(\overline{b} + \overline{aa})^* \overline{b})^* \overline{a} (\overline{b} + \overline{aa})^*$$

$$\text{Putting } \overline{q}_2 \text{ in } \overline{q}_2 = \overline{q}_3 = (\overline{a} + \overline{a}(\overline{b} + \overline{aa})^* \overline{b})^* \overline{a} (\overline{b} + \overline{aa})^* \overline{a} \quad \text{Ans}$$

②



$$g_1 = g_{10} + g_{20} + \varepsilon \quad \text{--- (1)}$$

$$g_2 = g_{11} + g_{21} + g_{31} \quad \text{--- (2)}$$

$$g_3 = g_{20} \quad \text{--- (3)}$$

from eqn 1 & 3

$$g_1 = g_{10} + (g_{20})_0 + \varepsilon \quad \text{--- (4)}$$

$$g_2 = \underbrace{g_{11}}_R + \underbrace{g_{21}}_0 \underbrace{(1+0.1)}_P$$

$$R = \delta P^*$$

$$g_2 = (g_{11})(1+0.1)^* \quad \text{--- (5)}$$

Put in eqn 6 in eqn 4

$$g_1 = g_{10} + (g_{11})(1+0.1)^* \cdot 0 \cdot 0 + \varepsilon \quad \text{--- (6)}$$

$$\frac{g_1}{R} = \frac{\varepsilon + g_{11}(0+1(1+0.1)^* 0 \cdot 0)}{P}$$

$$g_1 = \varepsilon \cdot (0+1(1+0.1)^* 0 \cdot 0)^*$$

Q Construct a regular expression which accept string that
2nd last character is a

$$L = \{ aa, ab, aaa, aab, bab, baa, \dots \}$$

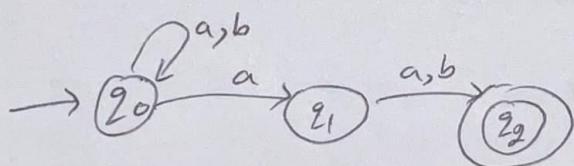
~~also~~ we have to restrict a ke baad sirf 1 single

letter aae $a(a+b)$

a se phale kuch bhi aae

$$\underline{(a+b)^* a (a+b)}$$

NFA



Construct NFA to FA

\emptyset	a	b
q_0	$[q_0, q_1]$	q_0
q_1	q_2	q_2
q_2	\emptyset	\emptyset

→ this \emptyset says us this FA is NFA

so convert NFA to FA

Transition Table for DFA

Step 1:- take unique initial state as it is (q_0)

Step 2:- Now fill table for initial state

Step 3:- Now take another state (q_1)

Step 4:- Now fill table row that go from both q_0, q_1

Step 5:- Now take another state q_3

Step 6:- Now fill table that go from q_0, q_3

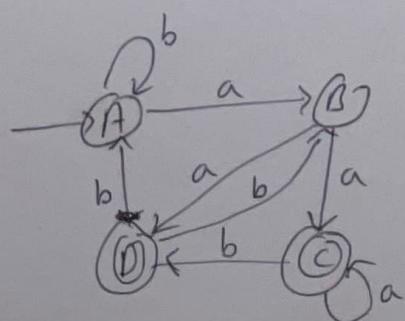
Step 6:- Now fill table that go from q_0, q_3
if q_3 not go anywhere so can't take ϕ in our
set

Step 7:- Now we take q_0, q_2 now fill table

Step 8:- So at last one make final state

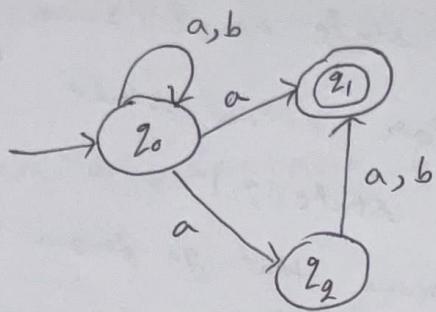
Step 8:- So at last one make final state
final state is that state which contain q_2

ϵ	a	b
$\rightarrow q_0 = A$	$[q_0, q_1] = B$	$q_0 = A$
$[q_0, q_1] = B$	$[q_0, q_1, q_2] = C$	$[q_0, q_2] = D$
$([q_0, q_1, q_2]) = C$	$[q_0, q_1, q_2] = C$	$[q_0, q_2] = D$
$([q_0, q_2]) = D$	$[q_0, q_1] = B$	$q_0 = A$



Q14

Convert NFA to FA



Transition Table for NFA

\emptyset	a	b
$\rightarrow q_0$	$[q_0, q_1, q_2]$	q_0
(q_1)	\emptyset	\emptyset
q_2	q_1	q_1

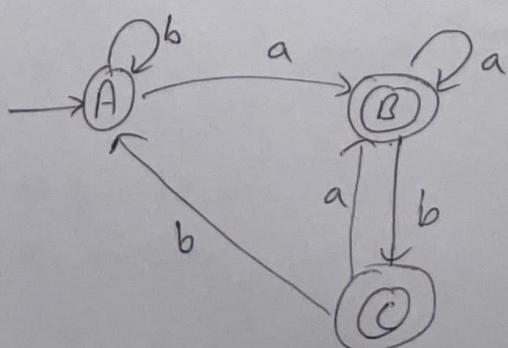
Transition table for DFA

\emptyset	a	b	b	b
$\rightarrow q_0 A$	$[q_0, q_1, q_2] = B$	$q_0 = A$		
$(q_0, q_1, q_2) = B$	$[q_0, q_1, q_2] = B$	$[q_0, q_1] = C$		
$[q_0, q_1] = C$	$[q_0, q_1, q_2] = B$	$q_0 = A$		

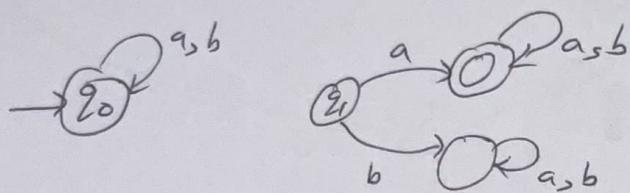
here we make both as final state because q_1 present on both

here for q_0 it go to q_0
 q_1 it go nowhere
 q_2 it go to q_1 ,
so we add q_0, q_1 here

→ here for q_0 it go to q_0 ,
 q_1 it go nowhere
so we add only q_0



when there is no way to reach to initial to final state
so our language is null but it's NFA

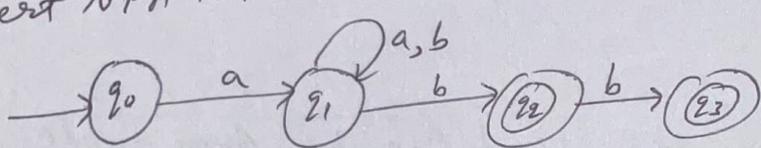


this is also a finite automata

because it say in linear time ki we reject this string

Q

Convert NFA to FA



+ = Union

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

Non Deterministic NDFAs or NFA is define by four tuple $(Q, \Sigma, \delta, S, F)$

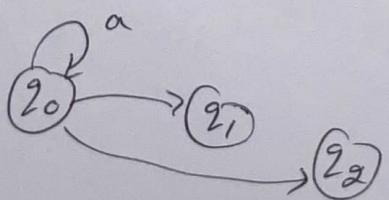
Q = finite set of nonempty set of states

Σ = finite nonempty set of symbols (letters)

* δ = is a transition function which maps $Q \times \Sigma \rightarrow 2^{|Q|}$
(set of all subset of Q (Powerset))

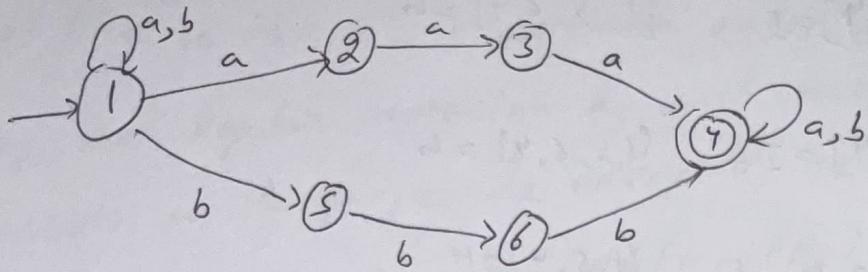
$S = s \in Q$

$F = \text{subset of } Q \quad S \subseteq Q$



$\{ \{ \}, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\} \}$

Q



Jitni NFA me states hogi utni hi ya usse zyda states DFA me hogi us se kam nahi hogi'

\emptyset	a	b
1	$[1, 2]$	$[1, 5]$
2	3	\emptyset
3	4	\emptyset
4	4	4
5	\emptyset	6
6	\emptyset	4

Transition table to DFA

\emptyset	a	b
$1 = A$	$[1, 2] = B$	$[1, 5] = C$
$[1, 2] = B$	$[1, 2, 3] = D$	$[1, 5] = C$
$[1, 5] = C$	$[1, 2] = B$	$[1, 5, 6] = E$
$[1, 2, 3] = D$	$[1, 2, 3, 4] = F$	$[1, 5] = C$
$[1, 5, 6] = E$	$[1, 2] = B$	$[1, 4, 5, 6] = G$
$[1, 2, 3, 4] = F$	$[1, 2, 3, 4] = F$	$[1, 5, 4] = H$

$$\begin{array}{l} \left[1, s, 6, 4 \right] = G \\ \left[1, s, 4 \right] = H \\ \left[1, 2, 4 \right] = I \end{array} \quad \begin{array}{l} \left[1, 2, 7 \right] = I \\ \left[1, 2, 4 \right] = I \\ \left[1, 2, 3, 4 \right] = F \end{array} \quad \begin{array}{l} \left[1, s, 6, 4 \right] = G \\ \left[1, s, 6, 4 \right] = G \\ \left[1, s, 4 \right] = H \end{array}$$

Algorithm which compute $FA_1 + FA_2$ 5marks
(11)

Let FA_1 be the finite automata excepting the language define by the regular expression r_1 , and there is an FA_2 that excepts a language l_2 excepted by the regular expression r_2 then there is an $FA(FA_1 + FA_2)$ that accepts the language $l_1 + l_2$ represent by the regular expression $r_1 + r_2$.

$r_1 + r_2$
Let FA_1 has states x_1, x_2, \dots, x_n and FA_2 has states y_1, y_2, \dots, y_m

Step1:- Build a new machine $FA_1 + FA_2$ with states z_1, z_2, \dots, z_k (k would be greater than or equal n, m)

where z something equal to x something or y something

Step2:- the combination state $x_{\text{start}} / y_{\text{start}}$ becomes the initial state of FA_1, FA_2

Step3:- If either x part or y part is the final state then the corresponding z state is the final state

How to construct Transition table

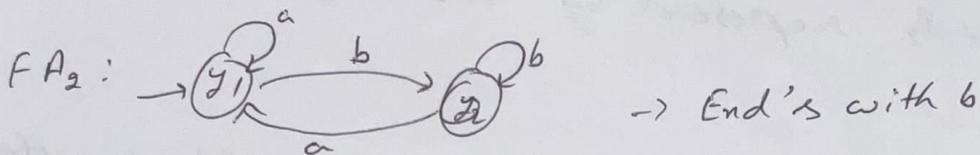
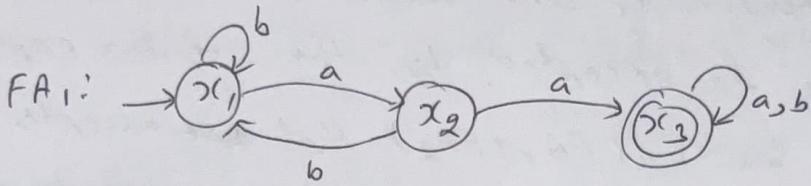
z_{new} after reading a letter (say p) = $\{x_{\text{new}} \text{ after reading letter } p \text{ in } FA_1, y_{\text{new}} \text{ after reading letter } p \text{ in } FA_2\}$

or $\{y_{\text{new}} \text{ after reading letter } p \text{ in } FA_2\}$

Note:- This algorithm work only in deterministic FA so firstly check FA is deterministic or not
Both FA's have diff state names must x_1, x_2, x_3
 y_1, y_2, y_3

Construct a FA that have substring aa

Q Either string have aa, end with b or both



Transition table for FA₁

Σ	a	b
$\rightarrow x_1$	x_2	x_1
x_2	x_3	x_1
x_3	x_3	x_3

Transition Table for FA₂

Σ	a	b
$\rightarrow y_1$	y_1	y_2
y_2	y_1	y_2

Max states are FA₁ states * FA₂ states

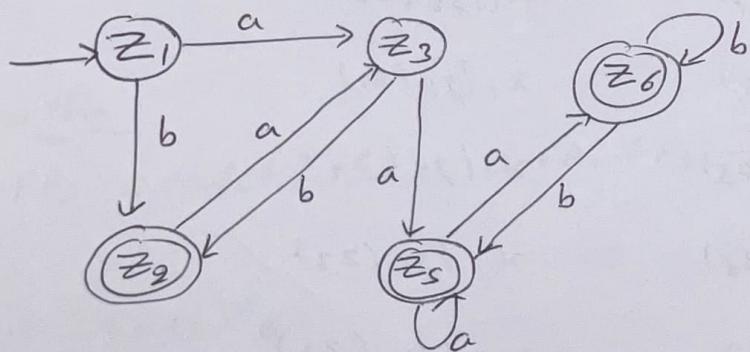
Transition Table for FA₁ + FA₂

Σ	a	b
$\rightarrow z_1 = x_1/y_1$	$x_2/y_1 = z_3$	$x_1/y_2 = z_2$
$(z_2) = x_1/y_2$	$x_2/y_1 = z_3$	$x_1/y_2 = z_2$
$z_3 = x_2/y_1$	$x_3/y_1 = z_5$	$x_1/y_2 = z_2$
$(z_4) = x_2/y_2$	$x_3/y_1 = z_5$	$x_1/y_2 = z_2$
$(z_5) = x_3/y_1$	$x_3/y_1 = z_5$	$x_3/y_2 = z_6$
$(z_6) = x_3/y_2$	$x_3/y_1 = z_5$	$x_3/y_2 = z_6$

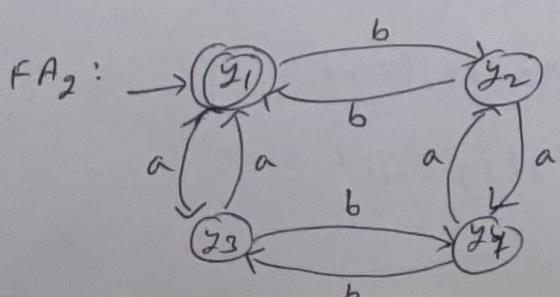
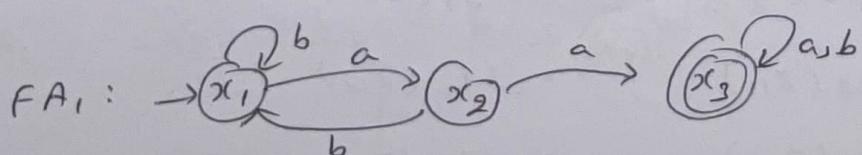
Initial state is where both FA initial state are there
final state is where FA₁ final or FA₂ final present

Note Trick:- firstly see first Transition Table and fill $FA_1 + FA_2$ Transition Table and give / after each fill to give space for second Transition Table to fill that also so after that see second transition table and fill $FA_1 + FA_2$

Note:- Remove that row where we can't go but don't remove initial state row
 here we can't reach to z_4 state so we remove z_4 row completely
 In $a \& b$ column we don't have z_4



Q Either string have even no of a or even no of b or both
 find $FA_1 + FA_2$



Transition table for FA₁

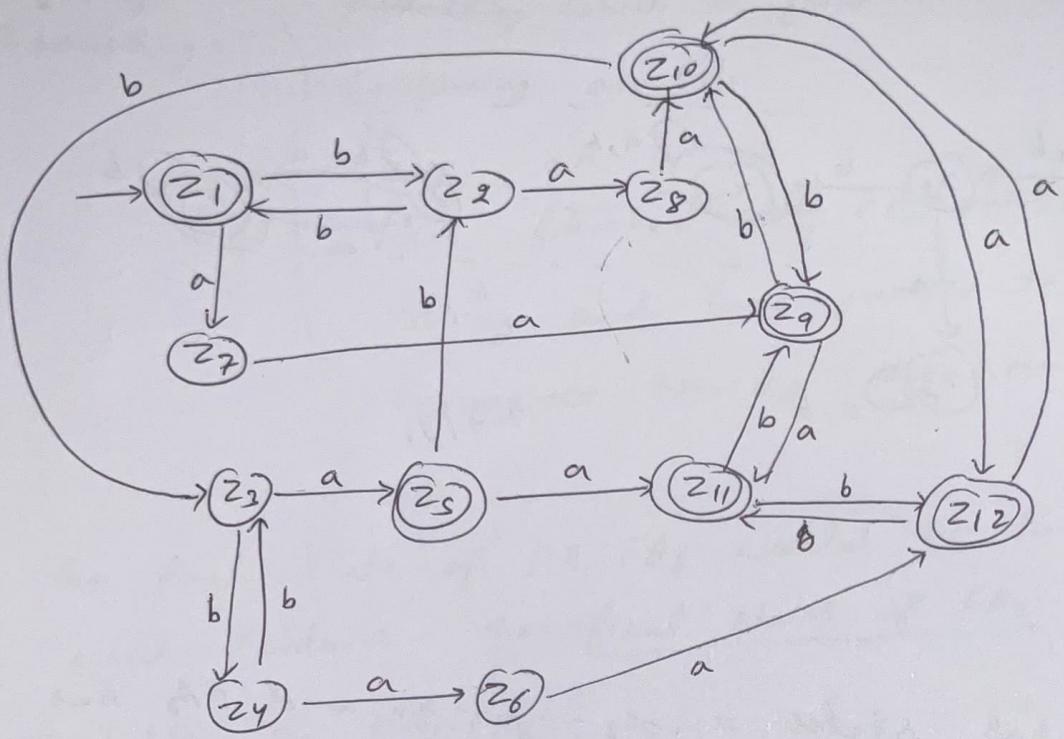
δ	a	b
$\rightarrow z_1$	x_2	x_1
z_2	x_3	x_1
$\circled{z_3}$	x_3	x_3

Transition table for FA₂

δ	a	b
$\rightarrow y_1$	y_3	y_2
y_2	y_4	y_1
y_3	y_1	y_4
y_4	y_2	y_3

Transition table for FA₁ + FA₂

δ	a	b
$\rightarrow \circled{z_1} = x_1/y_1$	$x_2/y_3 (z_7)$	$x_1/y_2 (z_2)$
$z_2 = x_1/y_2$	$x_2/y_4 (z_8)$	$x_1/y_1 (z_1)$
$z_3 = x_1/y_3$	$x_2/y_1 (\cancel{z_5})$	$x_1/y_4 (z_4)$
$z_4 = x_1/y_4$	$x_2/y_2 (z_6)$	$x_1/y_3 (z_3)$
$\circled{z_5} = x_2/y_1$	$x_3/y_3 (z_{11})$	$x_1/y_2 (z_2)$
$z_6 = x_2/y_2$	$x_3/y_4 (z_{12})$	$x_1/y_1 (z_1)$
$z_7 = x_2/y_3$	$x_3/y_1 (z_9)$	$x_1/y_4 (z_4)$
$z_8 = x_2/y_4$	$x_3/y_2 (z_{10})$	$x_1/y_3 (z_3)$
$\circled{z_9} = x_3/y_1$	$x_3/y_3 (z_{11})$	$x_3/y_2 (z_{10})$
$\circled{z_{10}} = x_3/y_2$	$x_3/y_4 (z_{12})$	$x_3/y_1 (z_9)$
$\circled{z_{11}} = x_3/y_3$	$x_3/y_1 (z_9)$	$x_3/y_4 (z_{12})$
$\circled{z_{12}} = x_3/y_4$	$x_3/y_2 (z_{10})$	$x_3/y_3 (z_{11})$



Algorithm

$FA_1 \cdot FA_2 \Rightarrow \text{concatenation}$

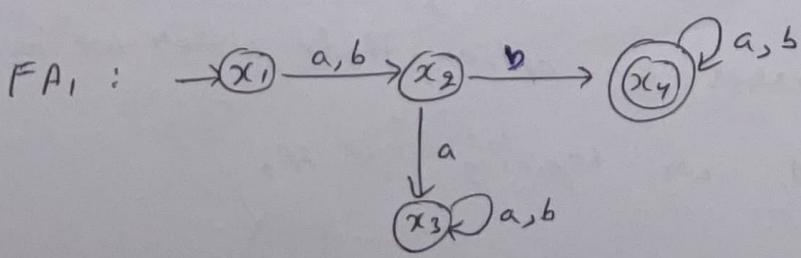
$FA_1 \cdot FA_2 \neq FA_2 \cdot FA_1$

$$r_1 : (a+b)^* a$$

$$r_2 : a(a+b)^*$$



$$r_1 \cdot r_2 : (a+b)^* a \cdot a (a+b)^*$$



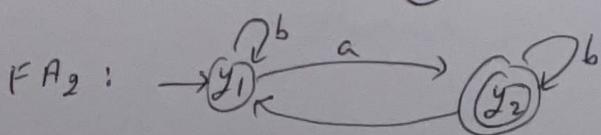
FA_1

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

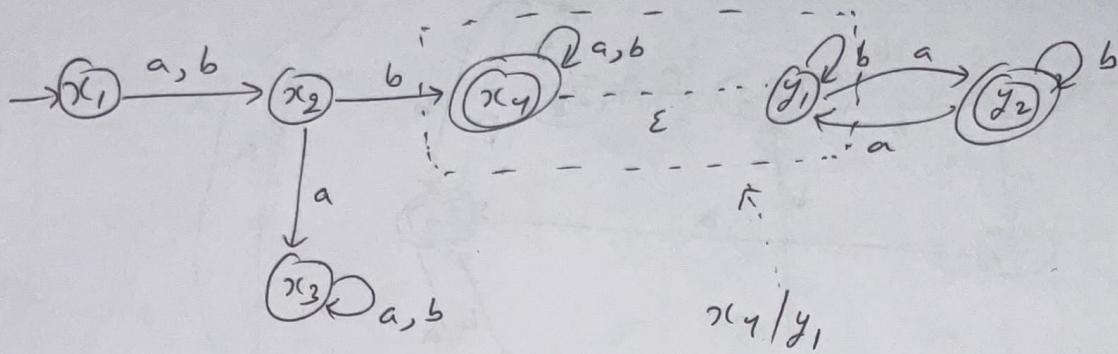
second letter must be b

FA_2

odd no of a's



we take initial state of FA₁ and final state of FA₂



Algorithm

Let FA_1 has states x_1, x_2, \dots, x_n and FA_2 has states y_1, y_2, \dots, y_m then $FA_1 \cdot FA_2$ will have states z_1, z_2, \dots, z_{n+m}

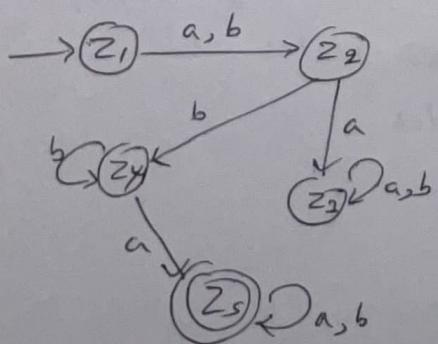
- 1) The initial state of FA₁, FA₂ will be the initial state of FA₁
 - 2) we make a γ state for every non-final state in FA₁ reached before ever hitting a final state in FA₁
 - 3) for every final state in FA₁ we establish a γ state that expresses an option that either we are continuing in FA₁ or begining with FA₂

$z_{\text{something}} = \left\{ \begin{array}{l} \text{"something which is final state and still continuing on FA}_1\text{,} \\ \text{OR} \\ \text{have finished with FA}_1\text{, part of the input string and have jumped to FA}_2\text{ to commence tracing the remainder of string} \end{array} \right.$

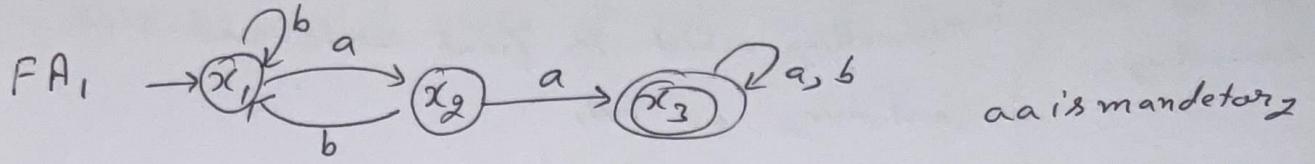
- 4) the final state of FA_1, FA_2 would be those states which contains the final states of FA_2 . ~~transition~~

Transition table of FA_1, FA_2

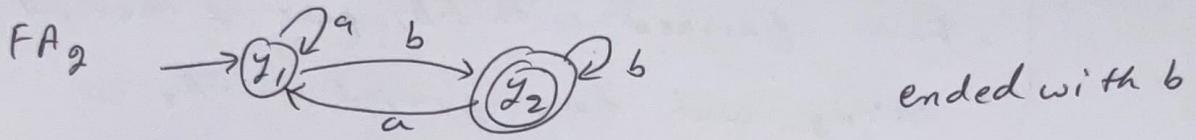
\emptyset	a	b
$z_1 = x_1$	$x_2 \notin z_2$	$x_2 (z_2)$
$z_2 = x_2$	$x_3 (z_3)$	$x_4/y_1 (z_4)$
$z_3 = x_3$	$x_3 (z_3)$	$x_3 (z_3)$
$z_4 = x_4/y_1$	$x_4/y_1/y_2 (z_5)$	$x_4/y_1/y_2 = x_4/y_1 (z_4)$
$(z_5) = x_4/y_1/y_2$	$x_4/y_1/y_2 (z_5)$	$x_4/y_1/y_2 (z_5)$



0



aa is mandatory



ended with b

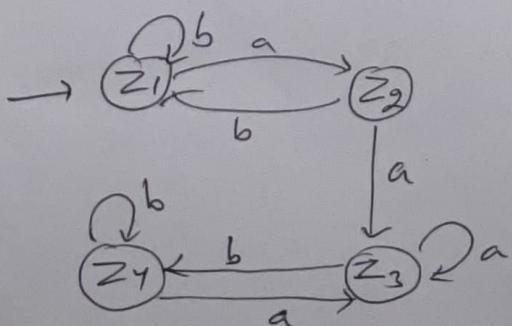
Transition table for FA₁, FA₂

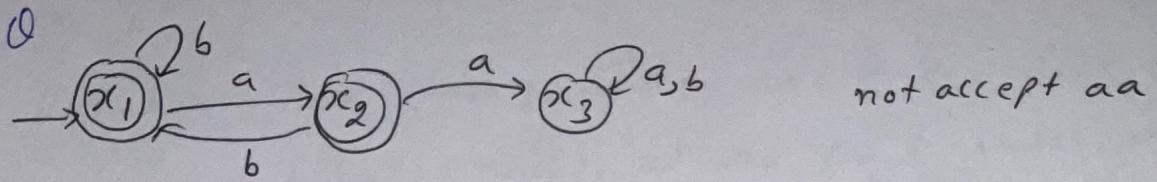
δ / Σ	a	b
$\rightarrow z_1 = x_1$	$x_2(z_2)$	$x_1(z_1)$
$z_2 = x_2$	$x_3/y_1(z_3)$	$x_1(z_1)$
$z_3 = x_3/y_1$	$x_3/y_1/y_1 = x_3/y_1(z_3)$	$x_3/y_1/y_2(z_4)$
$(z_4) = x_3/y_1/y_2$	$x_3/y_1(z_3)$	$x_3/y_1/y_2(z_4)$

when you go to the final state of FA₁, so it give two option just on FA₁ final state of jump to the initial state of FA₂

Note! - final state is that which contain y_2

Note! - New states are depend upon how many states on FA₂ if FA₂ have more states so FA₁, FA₂ have more states





not accept aa



odd no of a & b

we have x_1 is initial and final state also
so make $f\bar{A}_1$ and $F\bar{A}_2$ initial state as initial-state (x_1/y_1)

\varnothing	ϵ	a	b
$\rightarrow z_1 = x_1/y_1$		$x_2/y_1/z_2$	$x_1/y_1/y_2/z_3$
$\underline{z_2} = x_2/y_1$	y_2	$x_3/y_1/z_4$	$x_1/y_1/y_2/z_3$
$\underline{z_3} = x_1/y_2$	y_2	$x_2/y_1/z_2$	$x_1/y_1/y_2/z_3$
$\underline{z_4} = x_3/y_1/y_2$		$x_3/y_1/z_4$	$z_3/y_2/z_8$
$\underline{z_5} = \#$		y_2/z_7	y_2/z_7

Construction of Minimum state Finite Automata

K.P. Mishra, N. Chandra

Page 104

Minimization of finite Automata

here we construct an automata with min no of states equivalent to given automaton M.

As our interest lies only in strings accepted by M what really matters is whether a state is a final state or not. we define some relations in δ

Definition: Two states q_1 & q_2 are equivalent (denoted by $q_1 \equiv q_2$) if both $s(q_1, x)$ and $s(q_2, x)$ are final states. or both of them are nonfinal states for all $x \in \Sigma^*$

As it is difficult to construct $s(q_1, x)$ and $s(q_2, x)$ for all $x \in \Sigma^*$ (there are an infinite no of strings)

in Σ^*

Defination:- Two states q_1 and q_2 are k -equivalent ($k \geq 0$) if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final states or both nonfinal states for all strings x of length k or less. In particular any two final states are 0 -equivalent and any two nonfinal states are also 0 -equivalent.

Property 1:- equivalence and k -equivalence, are equivalence relations i.e they are reflexive, symmetric and transitive.

Property 2:- The partitions can be denoted by π and π_k resp
The elements of π_k are k -equivalence classes

Property 3:- If q_1 and q_2 are k -equivalent for all $k \geq 0$ then they are equivalent.

Property 4:- If q_1 and q_2 are $(k+1)$ -equivalent then they are k -equivalent

Property 5:- $\pi_n = \pi_{n+1}$ for some n . (π_n denotes the set of equivalence classes under n -equivalence)

Two states q_1 and q_2 are $(k+1)$ -equivalent if

(i) they are k -equivalent

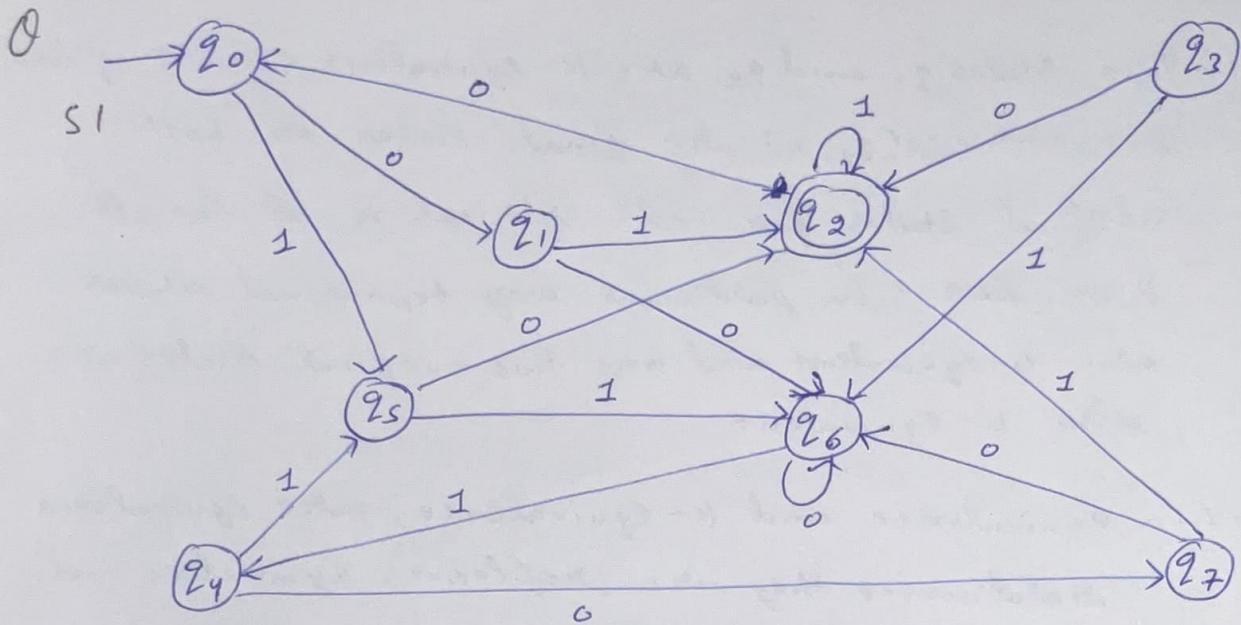
(ii) $\delta(q_1, a)$ and $\delta(q_2, a)$ are also k -equivalent for every $a \in \Sigma$

Step 1:- (Construction of π_0) By definition of 0 -equivalence $\pi_0 = \{\delta_1^\circ, \delta_2^\circ\}$
 $\delta_1^\circ \rightarrow$ set of final states and $\delta_2^\circ = \Sigma - \delta_1^\circ$

Step 2:- (Construction of π_{k+1} from π_k). Let δ_{ik} be any subset in π_k . If q_1 and q_2 are in δ_{ik} they are $(k+1)$ -equivalent provided $\delta(q_1, a)$ and $\delta(q_2, a)$ are k -equivalent. Find $\delta(q_1, a) \cap \delta(q_2, a)$ are ~~repeated~~ classes in π_{k+1}

Step 3:- Construct π_n for $n = 1, 2, \dots$ until $\pi_n = \pi_{n+1}$

Step 4:- Construction of min automaton. For the required minimum state automaton the states are the equivalence classes obtained in step 3 i.e the elements of π_n . The state table is obtained by replacing a state q by the corr equivalence class $\{q\}$.



State/ ϵ	a	b
$\rightarrow q_0$	q_1	q_5
q_1	q_6	q_2
$\textcircled{q_2}$	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

$$\Omega_1^\circ = F = \{q_2\}$$

$$\Omega_2^\circ = \Omega - \Omega_1^\circ$$

set of final states
 \nearrow \nwarrow
 $[\Omega_1^\circ, \Omega_2^\circ]$ set of non-final states

$$\pi_0 = \{q_2\}, \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}$$

in set Ω_1°
 q_2 can't be further partition.

Consider

$$\{q_0 \text{ and } q_1\}$$

$$q_0 \xrightarrow{a} q_1$$

$$q_1 \text{ and } q_6 \in \Omega_2^\circ$$

$$q_1 \xrightarrow{a} q_6$$

$$q_0 \xrightarrow{b} q_5$$

$$q_5 \in \Omega_2^\circ \text{ but } q_2 \notin \Omega_2^\circ$$

$$q_1 \xrightarrow{b} q_2$$

therefore $\{q_0 \text{ and } q_1\}$ is not 1-equivalent

$\{2_0, 2_4\} \rightarrow$ they are on δ_2^1

$\{2_0, 2_6\} \rightarrow$ they are on δ_2^1

$\{2_1, 2_7\} \rightarrow$ they are on δ_2^1 with a and δ_0^0 with b

$\{2_3, 2_5\}$ "

$\{2_4, 2_6\}$

$$\alpha_1 = \{2_0, 2_4, 2_6\}$$

$$\alpha_2 = \{2_1, 2_7\}$$

$$\alpha_3 = \{2_3, 2_5\}$$

$$\pi_1 = \{\{2_2\}, \{2_0, 2_4, 2_6\}, \{2_1, 2_7\}, \{2_3, 2_5\}\}$$

check again all belong on same or not take those that come on same

$$\{2_2\}$$

$$\{2_0, 2_4\}$$

$$\{2_1, 2_7\}$$

$$\{2_3, 2_5\}$$

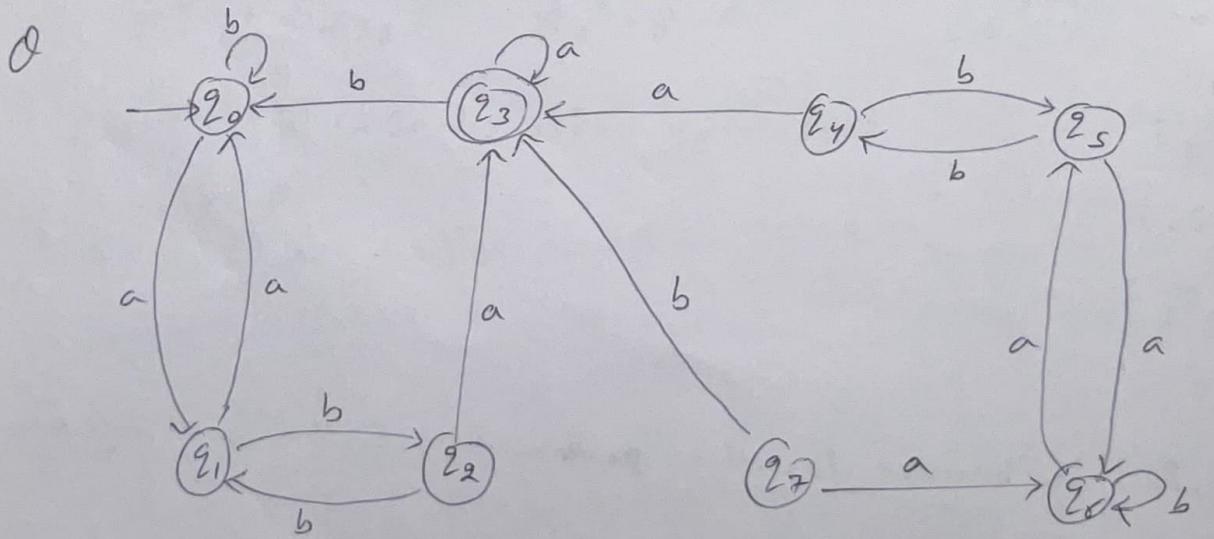
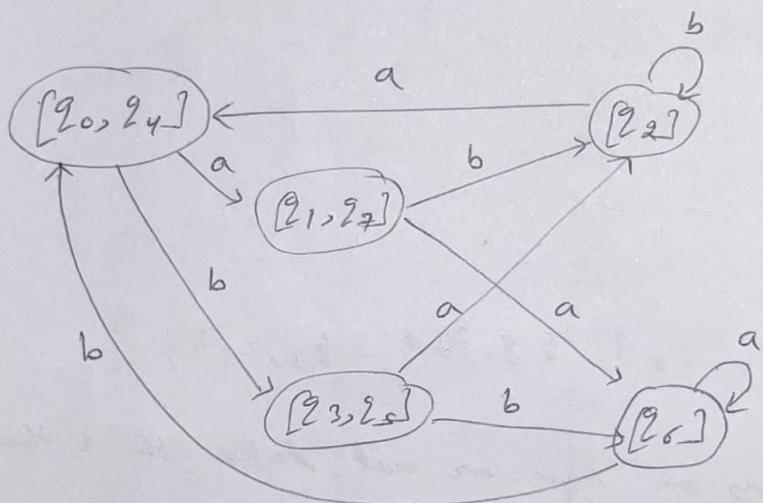
$$\{2_6\}$$

$\left. \begin{array}{c} \\ \\ \\ \end{array} \right\} \rightarrow 2\text{-equivalent}$

$\{2_0, 2_6\} X$ because 2_0 with a ~~path~~ and 2_6 with a go to diff states
 $\{2_0, 2_4, 2_6\} \xrightarrow{2_1, 2_7} \rightarrow$ they go to diff

$$\pi_3 = \{\{2_2\}, \{2_0, 2_4\}, \{2_6\}, \{2_1, 2_7\}, \{2_3, 2_5\}\}$$

\emptyset	a	b	
$\rightarrow [2_0, 2_4]$	$[2_1, 2_7]$	$[2_3, 2_5]$	2_0^{\prime} with b go to 2_5
(2_2)	$[2_0, 2_4]$	$[2_2]$	so we take 2_5 and 2_5 comes with 2_3 in n_3
$[2_6]$	$[2_6]$	$[2_0, 2_4]$	
$[2_1, 2_7]$	$[2_6]$	$[2_2]$	
$[2_3, 2_5]$	$[2_2]$	$[2_6]$	



109 page

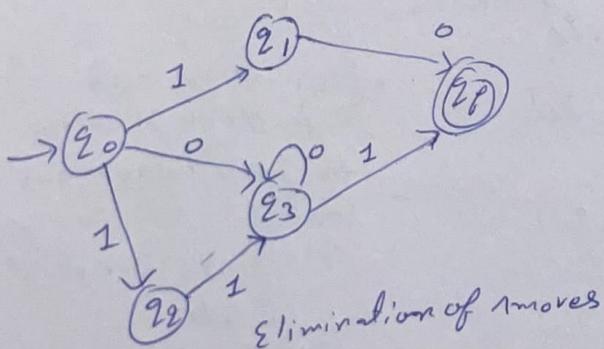
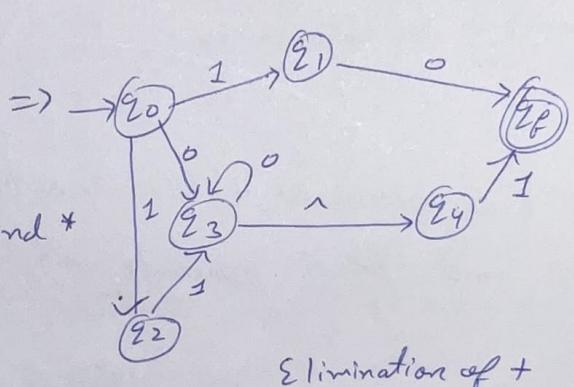
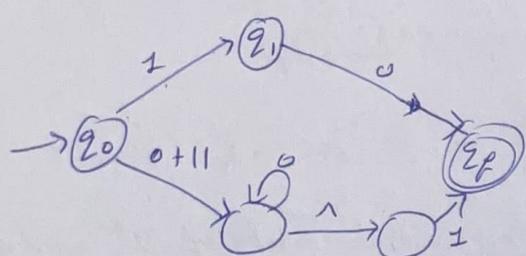
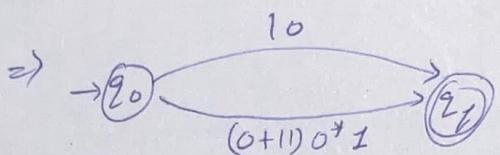
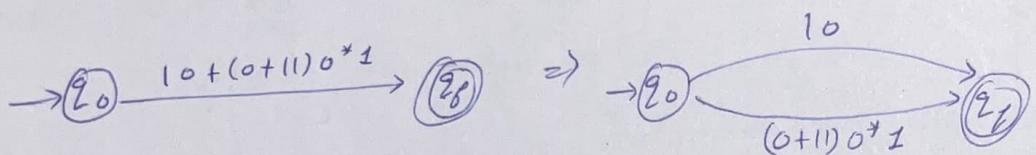
Transition System containing ϵ moves

& Construct a DFA with reduced states equivalent 169 page

to the r.e $10 + (0+11)0^*1$

K P Mishra N Chandra

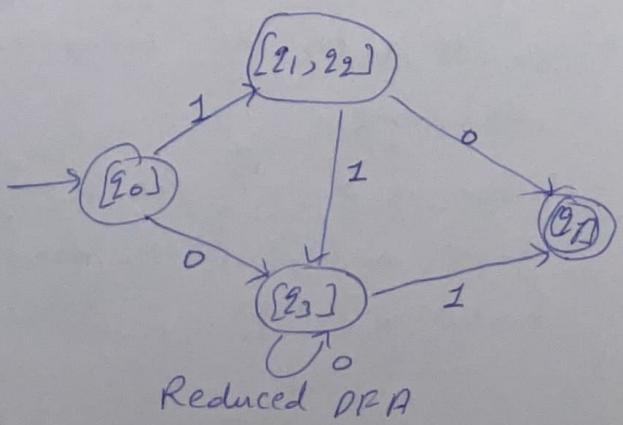
Construction of NDFA the NDFA is constructed by eliminating the operation + concatenation and * and the ϵ -moves in successive steps



States	0	1
q_0	q_3	q_1, q_2
q_1		q_1
q_2		q_3
q_3		q_3
q_f		q_f

0	0 ₀	0 ₁
$[q_0]$	$[q_3]$	$[q_1, q_2]$
$[q_3]$	$[q_3]$	$[q_f]$
$[q_1, q_2]$	$[q_f]$	$[q_3]$
$[q_f]$	\emptyset	\emptyset

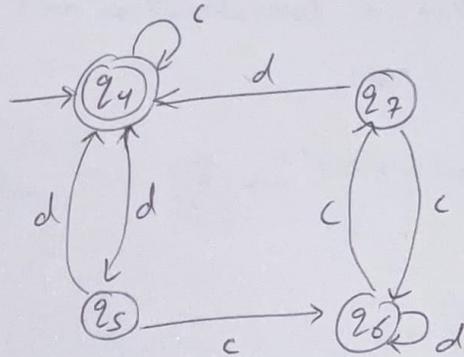
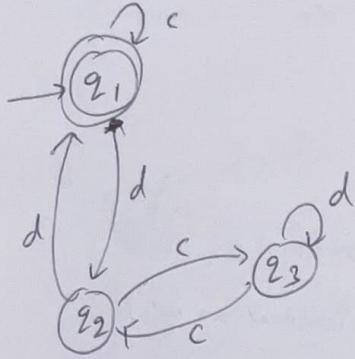
(DFA)



Equivalence of two FA

170 page

Determines whether m and m' are equivalent



Make comparison table firstly take initial state of 1st and 2nd FA and other generate on fly

q, q'	q_c, q'_c	q_d, q'_d
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_6)	(q_1, q_4)
(q_3, q_6)	(q_2, q_7)	(q_3, q_8)
(q_2, q_7)	(q_3, q_6)	(q_1, q_4)

after marking each state check it belongs to same category or not
eg - q_1, q_4 are final states
 q_2, q_5 are not final states

so all states are belong to same category so they are same

since we do not get a pair (q, q') where q is a final state and q' is a non final state (or vice versa) at every row so m and m' are equivalent

Transition system containing ϵ -moves

Part +

Suppose we are given with transition graph with ϵ -moves between a ~~non~~ state v_1 and state v_2

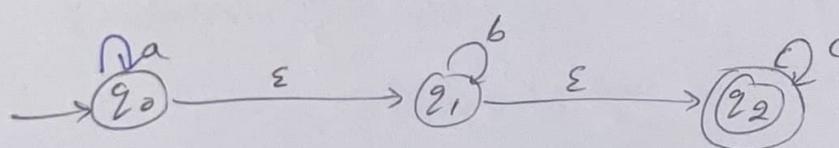
Step 1:- find all the edges starting from v_2

Step 2:- Duplicate all these edges starting from v_1 , without changing the edge labels

Step 3:- If v_1 is an initial state make v_2 also as an initial state

Step 4 If v_2 is a final state make v_1 also as a final state

Q Consider a finite automaton with ϵ -moves obtain an equivalent automaton ~~without~~ without ϵ -moves

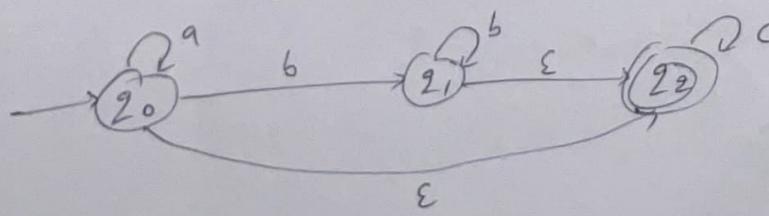


1. Removing ϵ -moves between q_1 and q_2

2. find all the edges starting from q_1

3. Duplicate all these edges starting from v_1 , without the edge labels

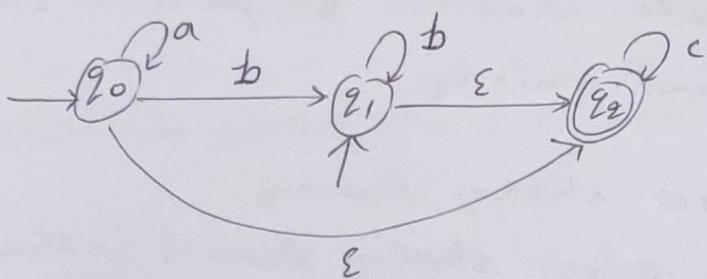
$q_1 \xrightarrow{\epsilon} q_2$ therefore we will duplicate this edge
having q_0 as a source state $q_0 \xrightarrow{\epsilon} q_2$



$q_1 \xrightarrow{b} q_1$
therefore, we will duplicate this edge having q_0 as a source state

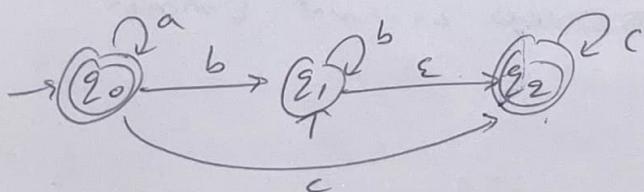
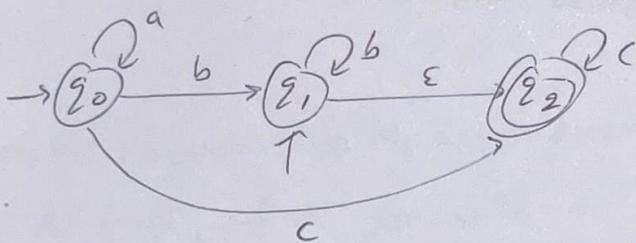
$$q_0 \xrightarrow{b} q_1$$

4. if q_1 is initial state make q_2 also as initial state
 (20) (21)



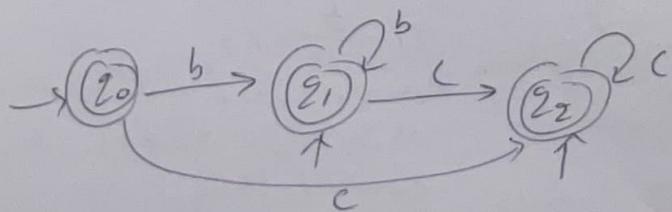
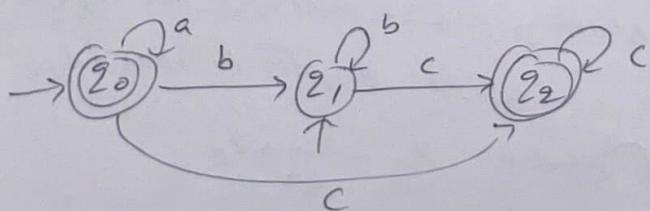
repeat all steps again $q_0 \xrightarrow{\epsilon} q_2$

$$q_2 \xleftarrow{c} q_2 \text{ so } q_0 \xrightarrow{c} q_2$$



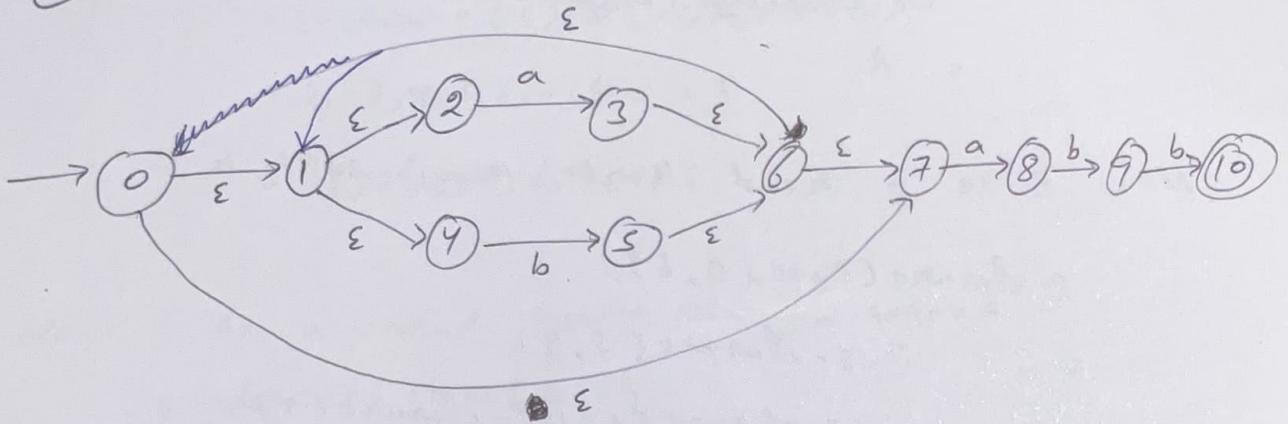
$$q_1 \xrightarrow{\epsilon} q_2$$

$$q_2 \xleftarrow{c} q_2 \text{ so } q_1 \xleftarrow{c} q_2$$



this algorithm fails because we have many initial states

NFA-ε to DFA



firstly find ϵ -closure of (0) \rightarrow Initial state

$$A = \cancel{\epsilon\text{-closure}}(0) = \{0, 1, 2, 4, 7\}$$

Now A se a read karke kha ja sakte h

ϵ -closure (Move{A, a})

$$= \epsilon\text{-closure}(3, 8)$$

$$= \epsilon\text{-closure}(3) \cup \epsilon\text{-closure}(8)$$

$$= \{3, 6, 7, 1, 2, 4, 3\} \cup \{8\}$$

$$B = \{1, 2, 3, 4, 6, 7, 8\}$$

Now A se b read karke kha ja sakte h

ϵ -closure (Move{A, b})

$$= \epsilon\text{-closure}(5)$$

$$C = \{1, 2, 4, 5, 6, 7\}$$

Now B se a read karke kha ja sakte h

ϵ -closure(Move{B, a})

$$= \epsilon\text{-closure}(3, 8)$$

we already compute it

$$= B$$

Now B se b read karke kha ja sakte h

ϵ -closure(Move{B, b})

$$= \epsilon\text{-closure}(5, 9)$$

$$= \epsilon\text{-closure}(5) \cup \epsilon\text{-closure}(9)$$

$$= \{1, 2, 4, 5, 6, 7, 3\} \cup \{9\}$$

$$D = \{1, 2, 4, 5, 6, 7, 9\}$$

Now C se a read karke kha ja sakte h

ϵ -closure(Move{C, a})

$$= \epsilon\text{-closure}(3, 8)$$

$$= B$$

Now C se b read karke kha ja sakte h

ϵ -closure(Move{C, b})

$$= \epsilon\text{-closure}(5)$$

$$= C$$

Now D se a read karke kha ja sakte h

ϵ -closure(Move{D, a})

$$= \epsilon\text{-closure}(3, 8)$$

$$= B$$

Now D se b read karke kha ja sakte h

ϵ -closure (Move {D, b})

$$= \epsilon\text{-closure}(S, 10)$$

$$= \epsilon\text{-closure}(S) \cup \epsilon\text{-closure}(10)$$

$$= \{1, 2, 4, 5, 6, 7\} \cup \{10\}$$

$$E = \{1, 2, 4, 5, 6, 7, 10\}$$

Now E se a read karke kha ja sakte h

ϵ -closure (Move {E, a})

$$= \epsilon\text{-closure}(3, 8)$$

$$= B$$

Now E se b read karke kha ja sakte h

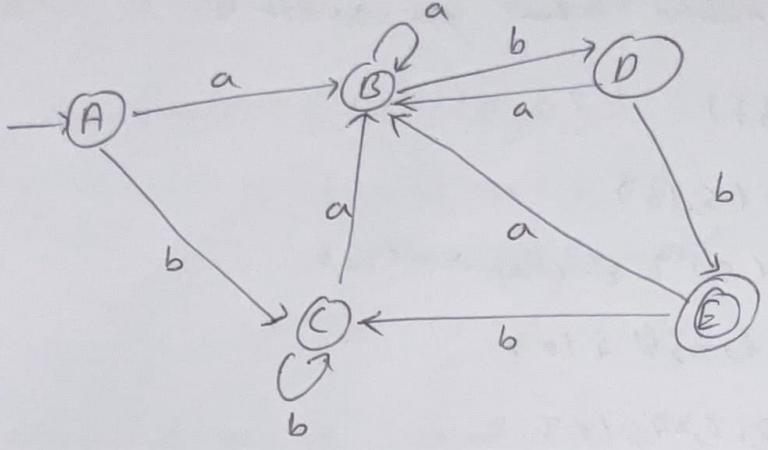
ϵ -closure (Move {E, b})

$$= \epsilon\text{-closure}(5)$$

$$= C$$

Note:- It can't guarantee
to min-state
FA

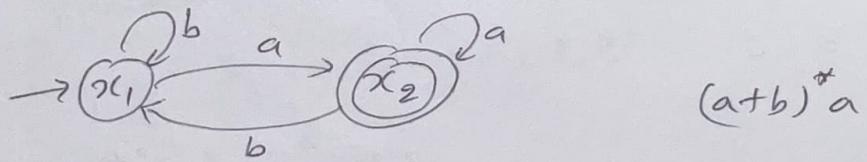
	a	b
$\rightarrow A$	B	C
B	B	D
C	B	C
D	B	E
E	B	C



When we run min-state finite automata algorithm on it then we have 4 states

$\{E\}, \{A, C\}, \{B\}, \{D\}$

Q Find FA^* for FA



1. find all subset's of the final set of states

$\{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}\}$

2. cancel all the subsets that contain the final states

but does not contain the initial state

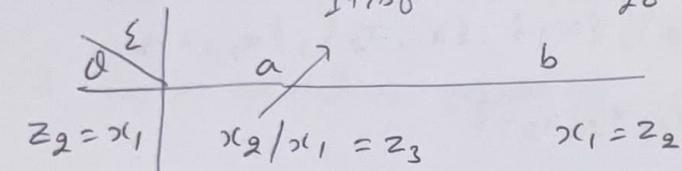
Therefore the possible subset remaining are as follows

$\{\emptyset, \{x_1\}, \{x_1, x_2\}\}$

$$\rightarrow z_1 = \phi \text{ [Initial as well as final]} \rightarrow$$

we see it later

It is final state so it have option it remain on x_2 or go to initial state



$$z_3 = \delta(x_1, x_2)$$

$x_1/x_2 = \cancel{z_2}$

$x_2/x_1 =$

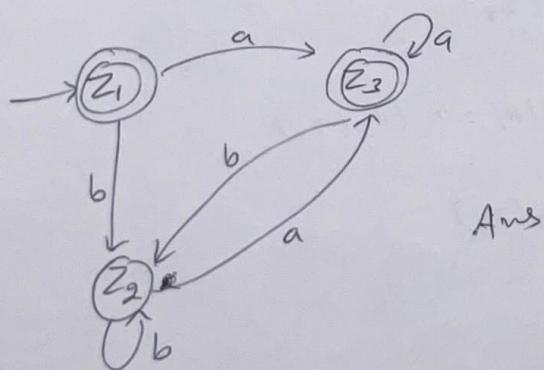
$= x_1/x_2 = z_3$

$$\rightarrow z_1 = \phi$$

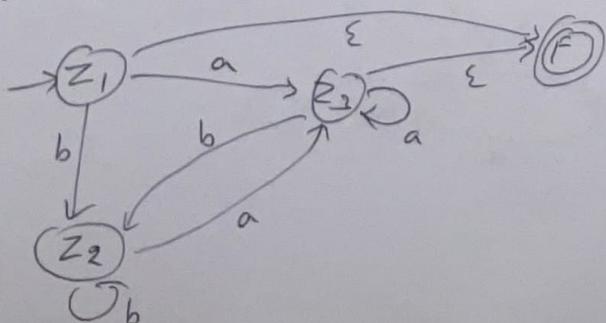
$x_2/x_1 = z_3$

$x_1 = z_2$

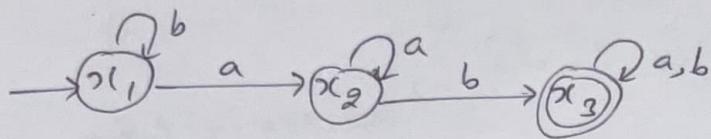
- * first and 3rd are same but we have to make 3rd $(z_1) = \phi$ because it is initial as well as final



If we check it is final so we make FA to regular expression



① find FA^* for FA



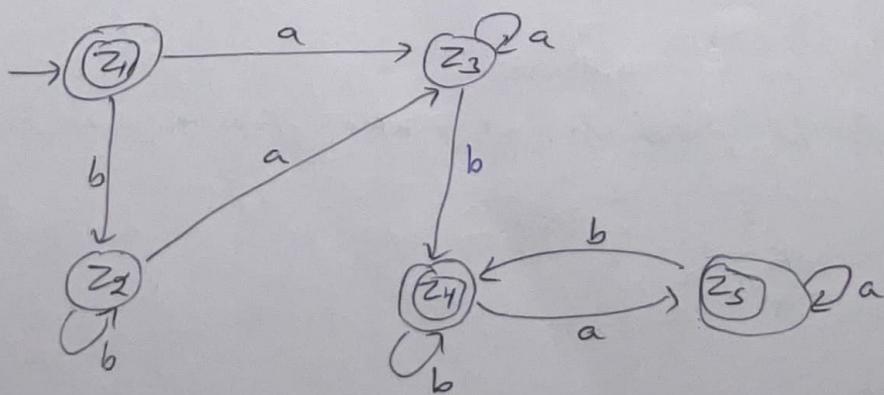
Possible subsets :- $\{\emptyset, \{x_1\}, \{x_2\}, \{x_3\}, \{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}$

Cancel all subset which contain final state but not contain the initial state.

$\{\emptyset, \{x_1\}, \{x_2\}, \{x_3\}, \{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\}\}$

transition table for FA^*

\emptyset/ϵ	a	b
$\rightarrow(z_1) = \emptyset$	$x_2(z_3)$	$x_1(z_2)$
$z_2 = x_1$	$x_2(z_3)$	$x_1(z_2)$
$z_3 = x_2$	$x_2(z_3)$	$x_3/x_1(z_4)$
$(z_4) = x_1/x_3$	$x_2/x_3/x_1(z_5)$	$x_1/x_3/x_1 = x_1/x_3(z_4)$
$(z_5) = x_1/x_2/x_3$	$x_1/x_2/x_3(z_5)$	$x_1/x_3(z_4)$



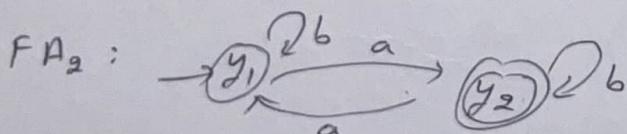
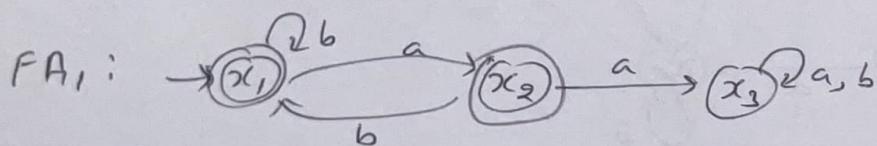
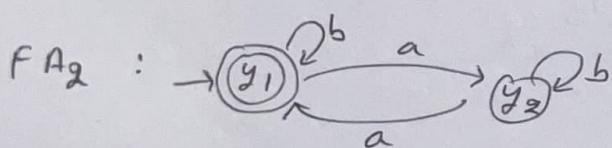
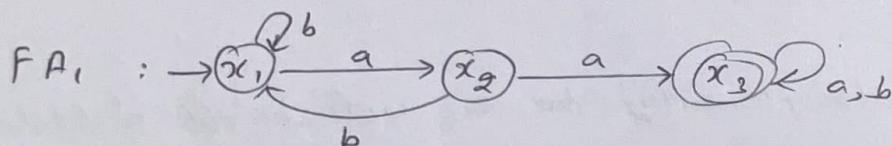
Property

- Intersection property :- If FA_1 is a finite automata and FA_2 is another finite automata then $FA_1 \cap FA_2$ is also an finite Automata
- Complimentation property

Q find $L_1 \cap L_2$ two languages defined as follows

L_1 = all the string with aa

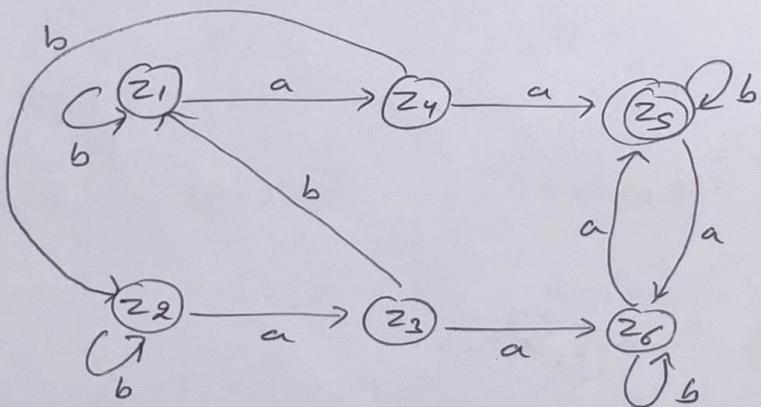
L_2 = all the string with even no of a's



first find Complementary of FA_1 & FA_2 and make transition table which contain x_1 & y_2 are final states

α / Σ	a	b
$\rightarrow z_1 = x_1/y_1$	$x_2/y_2 (z_4)$	$x_1/y_1 (z_1)$
$\circlearrowleft z_2 = x_1/y_2$	$x_2/y_1 (z_3)$	$x_1/y_2 (z_2)$
$\circlearrowleft z_3 = x_2/y_1$	$x_3/y_2 (z_6)$	$x_1/y_1 (z_1)$
$\circlearrowleft z_4 = x_2/y_2$	$x_3/y_1 (z_5)$	$x_1/y_2 (z_2)$
$z_5 = x_3/y_1$	$x_3/y_2 (z_6)$	$x_3/y_1 (z_5)$
$\circlearrowleft z_6 = x_3/y_2$	$x_3/y_1 (z_5)$	$x_1/y_2 (z_6)$

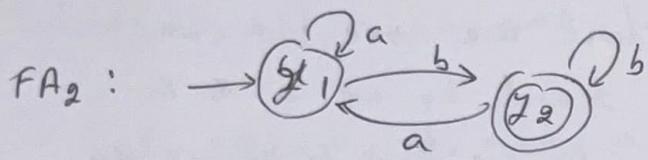
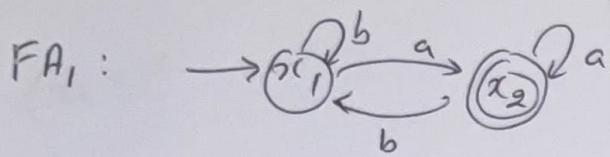
now again take complementry so z_5 will be final state



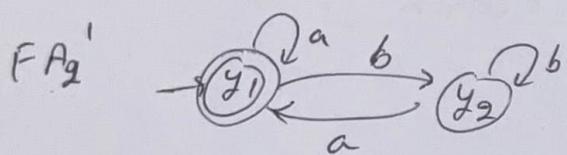
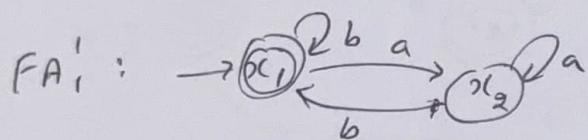
$$L_1 \quad (a+b)^* a$$

$$L_2 \quad (a+b)^* b$$

find $L_1 \cap L_2$?



$$FA_1 \cap FA_2 = (FA_1' \cup FA_2')'$$



Transition Table for FA_1'

	ϵ	a	b
ϵ	x_1	x_2	x_1
x_1	x_2	x_2	x_1
x_2	x_1	x_1	x_2

Transition Table for FA_2'

	ϵ	a	b
ϵ	y_1	y_1	y_2
y_1	y_2	y_1	y_2
y_2	y_1	y_1	y_2

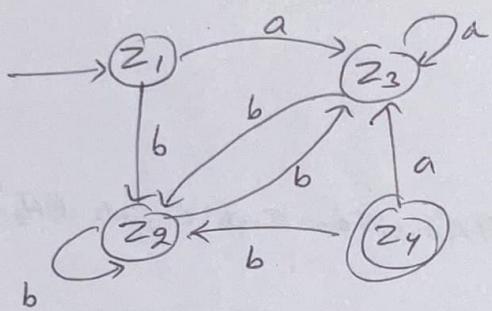
Transition Table for $FA_1' + FA_2'$

	ϵ	a	b
ϵ	$x_1/x_1 = z_1$	$x_1/x_2 = z_2$	
$x_1/x_1 = z_1$	$x_2/y_1 = z_3$	$x_1/y_2 = z_2$	
$x_2/y_1 = z_3$	$x_2/y_1 = z_3$	$x_1/y_2 = z_2$	
$x_1/y_2 = z_2$	$x_2/y_1 = z_3$	$x_4/y_2 = z_2$	
$x_4/y_2 = z_2$	$x_2/y_1 = z_3$	$x_4/y_2 = z_2$	

Transition Table for $(F - A_1' + F_2')'$

	a	b
$\rightarrow z_1 = x_1/y_1$	z_3	z_2
$z_2 = x_1/y_2$	z_3	z_2
$z_3 = x_2/y_1$	z_3	z_2
$(z_4 = x_2/y_2)$	z_3	z_2

here we see we can't reach z_4 we a & b so we have to remove z_4



since there is no way to reach z_4 so we proved

$$\emptyset \cap L_2 = \emptyset$$

chapter 9 denial I.A Cohen 1st & 17 part one in exam

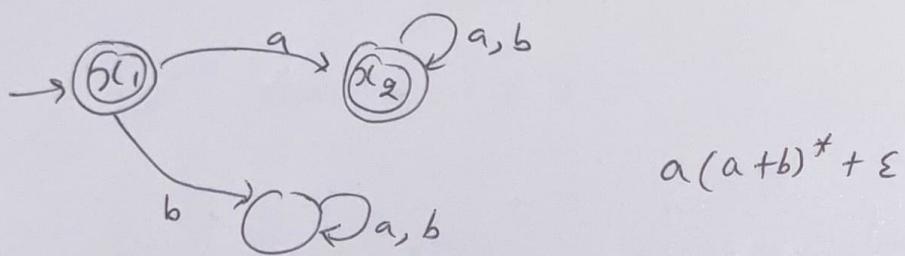
- A language can't be accepted by FA or RA that is non-regular language

$(ab^*)^*$

Chapter-9 14th Q

= { a, ab, abb, abbb, aab, ... } *

= { ε, aab, a, ab, abb, abbb, ... }



minimization to DFA *

$$\begin{matrix} z_2 + \\ z_4 + \\ z_5 + \end{matrix}$$

Because we are instead constructing the machine for

$$L_1 \cap L_2 = \text{all words in both } L_1 \text{ and } L_2$$

we put a + only in the state that represents acceptance by both machines at once:

$$z_2 + = x_2 \text{ or } y_2$$

Strings ending here are accepted if being run on FA_1 (by ending in x_2) and if being run on FA_2 (by ending in y_2). ■

Do not be fooled by this slight confusion:

$$z_2 = x_2 \text{ or } y_2 = \text{accepted by } FA_1 \text{ and } FA_2$$

The poor plus sign is perilously overworked.

$2 + 2$	(sometimes read "2 and 2 are 4")
$(a + b)^*$	(a or b repeated as often as we choose)
a^+	(a string of at least one a)
$L_1 + L_2$	(all words in L_1 or L_2)
$+ z_2, z_2^+$	(z_2 is a final state, the machine accepts input strings if they end here)
$1 + 1 = 2$	Arithmetic
$1 + 1 = 10$	Binary
$1 + 1 = 0$	Modulo 2
$1 + 1 = 1$	Boolean

If humans were not smarter than machines, they could never cope with the mess they make of their own notation.

PROBLEMS

For each of the following pairs of regular languages, find a regular expression and an FA that each define $L_1 \cap L_2$:

L_1	L_2
1. $(a + b)^*a$	$b(a + b)^*$
2. $(a + b)^*a$	$(a + b)^*aa(a + b)^*$
3. $(a + b)^*a$	$(a + b)^*b$
4. $(a + b)b(a + b)^*$	$b(a + b)^*$
5. $(a + b)b(a + b)^*$	$(a + b)^*aa(a + b)^*$
6. $(a + b)b(a + b)^*$	$(a + b)^*b$
7. $(b + ab)^*(a + \Lambda)$	$(a + b)^*aa(a + b)^*$
8. $(b + ab)^*(a + \Lambda)$	$(b + ab^*a)^*ab^*$
9. $(b + ab)^*(a + \Lambda)$	$(a + ba)^*a$
10. $(ab^*)^*$	$b(a + b)^*$

- | | |
|--|---|
| 11. $(ab^*)^*$ | $a(a + b)^*$ |
| 12. $(ab^*)^*$ | $(a + b)^*aa(a + b)^*$ |
| 13. All strings of even length
$= (aa + ab + ba + bb)^*$ | $b(a + b)^*$ |
| 14. Even-length strings | $(a + b)^*aa(a + b)^*$ |
| 15. Even-length strings | $(b + ab)^*(a + \Lambda)$ |
| 16. Odd-length strings | $a(a + b)^*$ |
| 17. Even-length strings | EVEN-EVEN |
| 18. (i) Even-length strings
(ii) Even-length strings | Strings with an even number of a 's
Strings with an odd number of a 's |
| 19. (i) Even-length strings
(ii) Even-length strings | Strings with an odd number of a 's and an odd number of b 's
Strings with an odd number of a 's and an even number of b 's |
| 20. We have seen that because the regular languages are closed under union and complement, they must be closed under intersection. Find a collection of languages that is closed under union and intersection but <i>not</i> under complement. | |

Pumping Lemma & 5 marks Doubt

Theorem 13

Let L be any regular language that has infinitely many words. Then there exist some three strings x, y and z (where y is not a null string) such that all the strings of the form ~~are words in L~~

$xy^n z$ for $n = 1, 2, 3, \dots$ are words in L

Note:- If we have loop in any state or loop in FA
do it have infinitely many words

Let us assume a language:-

$$L = \{ \epsilon, ab, aabb, aaabbb, \dots \}$$

$$L = \{ a^n b^n : n \geq 0 \}$$

Prove that $L = \{ a^n b^n : n \geq 0 \}$ is non regular

Suppose L is regular. If it is regular then L is accepted by an FA having ' K ' states (say)

By pumping lemma, there exist are some $w \in L$ and

$$|z| > K$$

which can be expressed as the concatenation of 3 strings

$$x, y, z$$

$$w = xyz \quad w \text{ is any string}$$

then

$$xy^i z \quad \text{for } i=1, 2, 3, \dots$$

must be in L

$$\text{let } w = a^n b^n \text{ for } n=4$$

aaaabbbb is in L

case I

$$n = aaa, y = a, z = bbb$$

for $i=2$ $aaa(a)^2 bbb$ must be in L

Pump $|y| \geq 0$ two times

But aaaabbbb $\notin L$

therefore our assumption was wrong so L is nonregular

$$|y| > 0$$

$$|xy| \leq n \quad |w| > k \rightarrow \text{states}$$

↑
length of string

$$\text{Let } x = a^p, y = a^q, z = b^r$$

$$\left. \begin{array}{l} |x| = p \\ |y| = q \\ |z| = r \end{array} \right\} \text{Length}$$

If $xy^i z$ must be in L

$$i=1 \quad a^p (a^q)^1 b^r \text{ must be in } L$$

$$\text{for } i=2 \quad a^p a^{2q} b^r \text{ must be in } L$$

$$= a^{p+2q} b^r \text{ must be in } L$$

$$p+2q > p+q = r$$

$$q \geq 0$$

$$\rightarrow \text{Hence } a^p(a^q)^i b^r \notin L$$

therefore our assumption was
wrong L is non-regular

Show that the language pallindrome is not regular

$$L = \{a^n b a^n : n \geq 1\} \text{ is not regular}$$

or

$$L = \{a^n b^n : n \geq 1\} \text{ is not regular}$$

Assume this is regular language

$x \ y^i \ z$ → Pump it i times

$y \neq \epsilon$

$|y| > 0 \quad |y| \geq 1 \quad |xy| \leq n$

Assume L is regular. If L is regular, then there is an FA with states ' n ' say which should accept L .

using pumping lemma

$w = a^n b^n$ can be expressed in term of three strings
must write this before start question
 x, y and z such that x, y, z is in L with $|y| > 0$
and $|xy| \leq n$
then $xy^i z$ must also in L

$$w = xyz$$

$$x = a^k, y = a^2, z = b^m$$

$$p+q = r \leq n \quad q \neq 0$$

then $a^k (a^2)^i b^m$ must also in L

$a^k a^{2i} b^m$ must also in L

$a^{k+2i} b^m$ must also in L

for $i=2$

$$p+2q \geq p+q > r > n$$

therefore $xy^i z$ is not in L

Hence our assumption is wrong the language is non-regular

$\{a^p \text{ where } p \text{ is prime}\}$ is non-regular

$$L = \{a^p \mid p \text{ is prime}\}$$

Suppose L is regular if it is regular then there is an FA with states in say which should accept L .

Let us suppose that FA has 25 states ~~$w = a^{29}$~~

is my choice can be expressed in terms of x, y, z such that x, y, z is in L with $|y| > 0$ and $|xy| \leq n$ then xyz must also in L

Pump y 30 times i.e $i = 30$

xyz^{30} must be in $L \Rightarrow xyzy^{29}$ must be in L
we do this because in our string we have all a 's
 $a^{29}y^{29}$ must be in L - ①

Let $y = a^m (m \neq 0) \quad |y| > 0$ - ②

Substitute 2 in 1

$a^{29}(a^m)^{29}$ must be in L

$a^{29(m+1)}$ must be in L

since $29(m+1)$ can never be a prime no since m is not equal to zero
Hence our assumption is wrong the language is non-regular

(Q) a^{n^2} is non-regular

$$L = \{ a^{n^2} : n \geq 1 \} \text{ is non regular}$$

suppose L is regular if it is regular than There is an FA with n states

$w = a^{n^2}$ can be expressed in term of 3 strings x, y, z such that x, y, z is in L with $|y| > 0$ and $|xy| \leq n$ then xy^iz must also in L

$$|x| + |y| + |z| = n^2 > n$$

pump y twice i.e $i=2$

xy^2z must be in L

$$|x| + |y| + |y| + |z| \cancel{\geq n^2} > n^2 \text{ since } |y| > 0 \quad \textcircled{1}$$

$$n^2 < |x| + |y| + |y| + |z| < n^2 + n \quad |xy| \leq n$$

$$|y| \leq n$$

$$n^2 < |x| + 2|y| + |z| < n^2 + n + n + 1$$

$$n^2 < |x| + 2|y| + |z| < (n+1)^2$$

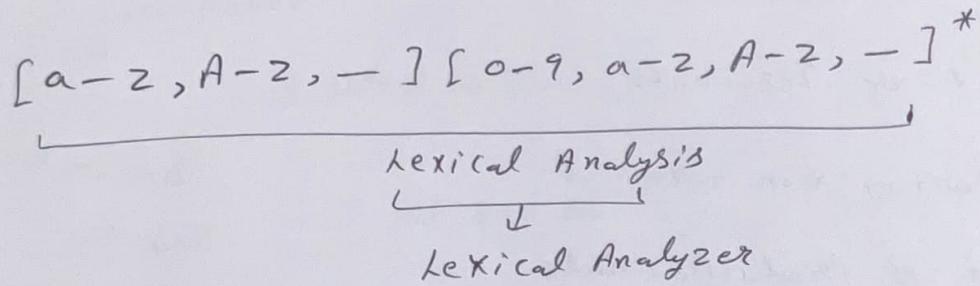
since $|x| + 2|y| + |z|$ lies b/w 2 consecutive square since this can't be perfect square so our assumption is wrong and it is non-regular

PDA

It Accept CFL (Context free language) represented by
 CFG (Context free grammar)

Lexical Analysis

int identifier = 20;



Non terminating

Production Rules

- | | |
|----------------------|----------------------------------|
| ① $NT = \{S, A, B\}$ | ③ $S \rightarrow AB$ |
| ② $T = \{a, b\}$ | $A \rightarrow aA \mid \epsilon$ |
| | $B \rightarrow bB \mid \epsilon$ |

Starting $NT = S$

Combination of all four things is called grammar

$$S \Rightarrow AB \Rightarrow S \Rightarrow aAB$$

\downarrow
generates to
or

Derivates to

$$S \Rightarrow aaAB$$

$$S \xrightarrow{*} aaad \dots \epsilon B$$

$$S \Rightarrow a^+ b^+$$

$$S \xrightarrow{*} a^+ b^*$$

$CFL \rightarrow RL$, some part of NRL , CFL , PDA

$$L = (NT, T, S, P)$$

$$= (V_N, T, S, P)$$

$$= (V_{NT}, V_T, S, P)$$

V_{NT} = Set of Non terminals

V_T = Set of Terminals

S = Starting non terminal

P = Set of Production Rule

Q Find the CFL for the given CFG

$$V_{NT} = \{S\}$$

$$V_T = \{a\}$$

Production Rules

$$P1: S \rightarrow aS$$

$$P2: S \rightarrow \epsilon$$

Production Rules

$$S \rightarrow aS/\epsilon$$

$$S \xrightarrow{\text{generates } \epsilon} \epsilon \text{ belongs to CFL}$$

$$S \rightarrow aS$$

$$\Rightarrow aas$$

put $S \rightarrow aS$

:

$$\Rightarrow a^n S \quad (S \xrightarrow{\epsilon} \epsilon)$$

$$\Rightarrow a^n \epsilon \in CFL$$

$$CFL = \{a^n : n \geq 0\} \Rightarrow a^* a^n$$

$$= \{\epsilon, a, aa, aaa, aaaa, \dots\}$$

Q find the CFL for the give CFG

$$V_{NT} = \{S\} \quad V_T = \{a, b\}$$

Production Rules

$$S \rightarrow aS/bS/a/b$$

$$S \rightarrow a$$

$$a \in CFL \quad \text{--- (1)}$$

$$S \rightarrow b$$

$$b \in CFL \quad \text{--- (2)}$$

$$S \rightarrow aS$$

$$\Rightarrow aas$$

$$\Rightarrow a^{n-1}S$$

$$\Rightarrow a^n$$

$$a^n \in CFL$$

$$S \rightarrow bS$$

$$\Rightarrow bbS$$

$$\Rightarrow b^{n-1}S$$

$$\Rightarrow b^n$$

$$b^n \in CFL$$

$$S \rightarrow aS$$

$$\Rightarrow abS$$

$$\Rightarrow ababS$$

$$\Rightarrow abababS$$

$$\Rightarrow (a+b)^n \in CFL$$

$CFL = \{ (a+b)^n : n > 0 \}$ when ϵ is not present we not use equal to to the n

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

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

to remove null we use $(a+b)$ with $(a+b)^*$ because $(a+b)^*$ also give us ϵ and ϵ is not acceptable in our production rule

Q Find the CFL for the given CFG

$$V_{NT} = \{S, X\}$$

$$S \rightarrow XaaX$$

$$V_T = \{a, b\}$$

$$X = ax/bX/\epsilon$$

Here our starting terminal is S because S and generate X
but X can't generate S

$$S \rightarrow XaaX$$

here we have 2 non-Terminal

$$\Rightarrow aax$$

there are 2 types of derivation leftmost and rightmost

$$\Rightarrow aa$$

In leftmost we start from left and move

$$aa \in CFL \quad \textcircled{1}$$

to right

In rightmost we start from right and move
to left

At one time we change one non-terminal

find ans

Q Find CFL from given CFG

$$V_N = \{S, X, Y\} \quad V_T = \{a, b\} \quad S \rightarrow XY$$
$$X = ax/bx/a$$
$$Y = ya/yb/a$$

$$X \Rightarrow ax \quad X \Rightarrow bx \quad Y \Rightarrow ya \quad X \Rightarrow xb$$
$$\Rightarrow abx \quad \Rightarrow bax \quad \Rightarrow yba \quad \Rightarrow yab$$
$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$
$$(a+b)^*a \quad a(a+b)^* \quad a(a+b)^*b \quad a(a+b)^*$$

$$S \Rightarrow XY$$

$$(a+b)^*aa(a+b)^* \text{ Ans}$$

Note! - CFG that have less Non-Terminal are good for compiler

Q find CFL from given CFG

$$V_{NT} = \{S, B, U\} \quad S \rightarrow SS / BS / SB / \epsilon / USU$$

$$V_T = \{a, b\} \quad B \rightarrow aa/bb$$

$$U \rightarrow ab/ba$$

B = Balanced

U = unBalanced

$$S \Rightarrow BS \quad S \Rightarrow SB$$
$$\Rightarrow aas \quad \Rightarrow$$
$$\Rightarrow aa$$
$$aa \in \Sigma \text{ CFL}$$

$$[aa + bb + (ab+ba)(aa+bb)^*(ab+ba)]^*$$

Q find CFL from given CFG

$$S \rightarrow aSa / bSb / a / b / \epsilon$$

$$\begin{aligned} S &\Rightarrow aSa \\ &\Rightarrow aaaSaa \\ &\Rightarrow aabaaa \\ &\Rightarrow aaaaaa \end{aligned}$$

$$\begin{aligned} S &\Rightarrow aSa \\ &\Rightarrow abSba \\ &\Rightarrow aabbSbba \\ &\Rightarrow abbabbba \\ &\Rightarrow abbbbbba \end{aligned}$$

this is palindrom

$$a^n a^n \quad b^n b^n$$

$$a^n b a^n, a^n a a^n, b^n a b^n, b^n b b^n$$

Q Consider the CFG

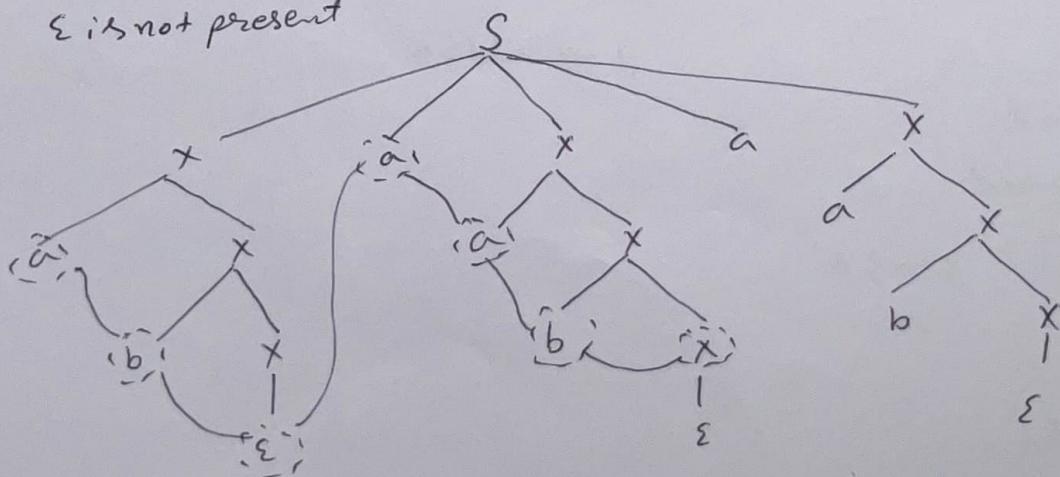
$$S \rightarrow XaXaX$$

$$X \rightarrow aX / bX / \epsilon$$

what is the language this CFG generates? find CFL

min string length $\rightarrow 2$

ϵ is not present



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

Q Consider the CFG

What is the language this CFG generates?
find CFL

$$S \rightarrow aX$$

$$X \rightarrow aX/bX/\epsilon$$

$$a(a+b)^* \text{ Ans}$$

Q Find a CFG for each of the language defined by the following regular expression

(i) ab^*

$$S \rightarrow aX$$

$$X \rightarrow bX/\epsilon$$

$\{a, ab, abb, abbb, \dots\} ?$

Q Find CFG's

$$\Sigma = \{a, b\};$$

All words that have exactly 2 or 3 b's

$$a^* b a^* b a^* + a^* b a^* b a^* b a^*$$

$$S \rightarrow X b X b X / X b X b X b X$$

$$X \rightarrow aX/\epsilon$$

give all
Terminal
Non-terminal
starting nonterminal

d) Find CFG

i) All words that have diff first and last letter

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

$$S \rightarrow axb / bxa$$

$$x \rightarrow a/x/b/x/\epsilon$$

(iii) $(baa + abb)^*$

~~S → aabb / baab~~

$$S \rightarrow baaS / abbs / \epsilon$$

iii) All words that do not have substring ab

$$b^*a^*$$

$$S \rightarrow XY$$

$$X \rightarrow bX / \epsilon$$

$$Y \rightarrow aY / \epsilon$$

Simplification of CFG

C++ is governed by CFG

(1) Eliminate Useless Symbol (Non-Terminal)

Eliminate useless productions

(2) Eliminate null or nullable productions

Non-Terminal \rightarrow Capital letter

~~Terminal~~ \rightarrow small letter

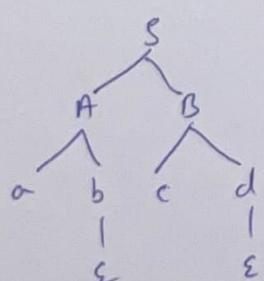
(3) Eliminate unit productions

left one nonterminal — Right one nonterminal
unit production

$S \xrightarrow{*} \alpha A \gamma \xrightarrow{*} w$
↓
useful symbol
y terminal string

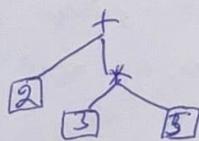
S that generate terminal string
in some steps so that is
useful symbol

Parse Tree



Internal nodes are
non-terminal

Syntax Tree



Internal nodes are ~~useful~~ symbols

① we say x is generating if $x \xrightarrow{*} \omega$ for some terminal string ω

And

② we say x is reachable if there is a derivation $S \xrightarrow{*} \alpha x \beta$ of some α and β

Q $S \rightarrow AB$
 $A \rightarrow aA/\epsilon$
 $B \rightarrow bB/\epsilon$
 $C = cC/\epsilon$

Here S can't reach C so C is useless we have to eliminate it

Q $S \rightarrow AB/a$
 $A \rightarrow b$

$S \rightarrow b$

Q $S \rightarrow aS/A/C$.
 $A \rightarrow a$
 $\therefore B \rightarrow aa$
 $\therefore C \rightarrow aCb$

$B \rightarrow$ not reachable from S
 $C \rightarrow$ not generating
so we have

$S \rightarrow aS/A$
 $A \rightarrow a$

Q $S \rightarrow aAa/aBC$
 $A \rightarrow aS/bD$
 $B \rightarrow ab/a/b$
 $C \rightarrow abb/DD$

$E & F$ are generating but they are not reachable

D is not generating because it

so we have

$S \rightarrow aAa/aBC$
 $A \rightarrow aS$
 $B \rightarrow ab/a/b$
 $C \rightarrow abb$

If $G = (V_{NT}, V_T, S, P)$ be a CFG and $L(G) \neq \emptyset$ then $G' = (V'_{NT}, V_T, S, P')$

by removing useless symbols/productions

will lead to a condition $L(G) = L(G')$

\downarrow
language generation by grammar G'

Remove Nullable production

$A \rightarrow \epsilon \rightarrow$ It goes directly to ϵ

$A \xrightarrow{*} \epsilon \rightarrow$ after some generate it goes to ϵ so it is nullable

$$G = (V_{NT}, V_T, S, P)$$

$$G' = (V'_{NT}, V_T, S, P')$$

$$L(G) \neq L(G')$$

$$L(G') = L(G) - \{\epsilon\}$$

$\epsilon \rightarrow$ directly null words
 $\text{nullabe} \rightarrow$ indirectly null words

$$\begin{aligned} S &\rightarrow Aa \rightarrow *|B|A|AB \\ A &\rightarrow Aa/\epsilon \rightarrow a|Aa \\ B &\rightarrow Bb/\epsilon \end{aligned}$$

$$S \rightarrow AB$$

$$S \rightarrow \epsilon B$$

$$S \rightarrow \epsilon \epsilon$$

$$\text{here } A \rightarrow \epsilon$$

$$B \rightarrow \epsilon$$

$$S \xrightarrow{*} \epsilon$$

AB	00	0
00	01	B
01	10	A
10	11	AB
11	A	AB
A	0	AA
0	1	AK
1	A	AK

$$\epsilon \in L(G)$$

so we remove ϵ if after removing ϵ from the string and string is empty so we have to add ϵ to the string because we have to make atleast one string $S \rightarrow \epsilon$

$$S \rightarrow AB$$

$$A \rightarrow aAA/\epsilon$$

$$B \rightarrow bBB/\epsilon$$

Since

$$\begin{aligned} A &\rightarrow \epsilon \\ B &\rightarrow \epsilon \end{aligned} \quad \} \text{ null productions}$$

$$S \rightarrow AB / A / B / \epsilon$$

$$A \rightarrow a / aA / aAA$$

$$B \rightarrow b / bB / bBB$$

$$L(G') = L(G) - \{\epsilon\}$$

Now we write
 aAA and bBB

Q Eliminate null productions from the following CFG

$$S \rightarrow ABaC$$

$$A \rightarrow BC$$

$$B \rightarrow b/\epsilon$$

$$C \rightarrow D/\epsilon$$

$$D \rightarrow d$$

if language L(S) accept ϵ so firstly
remove null and write all
combinations
And at the end add ϵ to
the language

Sol:-

$$\begin{array}{l} B \rightarrow \epsilon \\ C \rightarrow \epsilon \end{array} \quad \left. \begin{array}{l} B \rightarrow \epsilon \\ C \rightarrow \epsilon \end{array} \right\} \text{are } \epsilon\text{-productions}$$

$A^* \Rightarrow \epsilon$ is a nullable production

$$\begin{array}{l} S \rightarrow ABaC / ABa / AaC / Aa / BaC / Ba / aC / a \\ A \rightarrow BC / B / c \\ B \rightarrow b \\ C \rightarrow D \\ D \rightarrow d \end{array}$$

$$\begin{array}{ll} ABC & BC \\ 000 \Rightarrow a & 00 \rightarrow \epsilon \times \\ 001 \Rightarrow aC & 01 \rightarrow C \\ 010 \Rightarrow Ba & 10 \rightarrow B \\ 011 \Rightarrow BaC & 11 \rightarrow BC \end{array}$$

+ term: $00 \rightarrow Aa$
here that $01 \Rightarrow AaC$
we note all $110 \Rightarrow ABA$
here $111 \Rightarrow ABaC$

Eliminate unit production

Q $S \rightarrow Aa / B$

$$B \rightarrow A / bb$$

$$A \rightarrow a / bc / B$$

The CFG without unit productions can be written as

$$S \rightarrow Aa$$

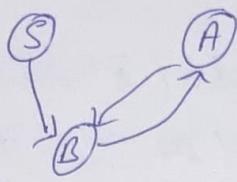
$$B \rightarrow bb$$

$$A \rightarrow a / bc$$

Plus the new production rules

$A \rightarrow bb \rightarrow$ Replace B with B production rule

$B \rightarrow a/bc \rightarrow$ Replace A with A production



$S \rightarrow a/bc/bb \rightarrow S \rightarrow B \rightarrow A \text{ so } S \rightarrow A \text{ Replace } B \text{ with } B \text{ production rules}$

$S \rightarrow Aa/a/bc/bb$

$B \rightarrow a/bc/bb \text{ // not reachable from } S \Rightarrow \text{useless symbol}$

$A \rightarrow a/bc/bb$

Q $I \rightarrow a/b/Ia/Ib/Io/I1$

$F \rightarrow I/(E),$

$() \rightarrow \text{Terminal}$

$T \rightarrow F/T^*F$

$E \rightarrow T/E+T$

sol^m Here starting non terminal is E

$$V_{NT} = \{ E, T, F, I \}$$

$$V_T = \{ a, b, o, 1, (,), ^*, + \}$$

$$S = E$$

The CFG without unit productions can be written as

$$E \rightarrow E+T$$

$$T \rightarrow T^*F$$

$$F \rightarrow (E).$$

$$I \rightarrow a/b/Ia/Ib/Io/I1$$

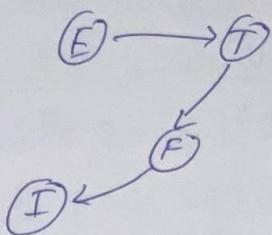
• Plus the new production rules

$$I \rightarrow a/b/Ia/Ib/Io/I_1$$

$$F \rightarrow a/b/Ia/Ib/Io/I_1/(E)$$

$$T \rightarrow a/b/Ia/Ib/Io/I_1/(E)/T^*F$$

$$E \rightarrow a/b/Ia/Ib/Io/I_1/(E)/T^*F/E^+T$$



F to E we can't draw
because in F E is (E) it
can't go directly

CNF

• Eliminate the NULL production

Step 1:-

Step 2:- Eliminate any unit production

Step 3:- Eliminate all the useless symbol in the grammar

Step 4:- Convert the resulting grammar CNF (Chomsky normal form)

Step 4:- Convert the resulting grammar CNF (Chomsky normal form)

$$S \rightarrow OAO/IBI/BB$$

$$A \rightarrow C$$

$$B \rightarrow S/A$$

$$C \rightarrow S/E$$

(AB)Bade \rightarrow non-terminal note "n"

(a) chose \rightarrow terminal note "t" Chomsky Normal Form ((NF))

Every CFL (without null) is generating by CF which all production are of the form $A \rightarrow BC$ or $A \rightarrow a$ where A, B, C are non-terminals and a is a terminal further the CF has no useless symbol such a grammar is said to be in Chomsky normal form

Step1:- Arrange that all bodies of length 2 or more consist only of variables

Step2: Break the bodies of 3 or more into a cascade of production, each ~~is~~ with a body containing 2 variables

Step3:- for every terminal a that appears in a body of length 2 or more create a variable, say (A) . This variable has only one production $A \rightarrow a$

Step2.1. for step2 if the production $A \rightarrow B_1, B_2, \dots, B_K$ $|K \geq 3$ then introduce new variable $(K-2)$ no of new variables

$$A \rightarrow B_1 C_1$$

$$C_1 \rightarrow B_2 C_2$$

$$C_2 \rightarrow B_3 C_3$$

:

$$C_{K-2} \rightarrow B_{K-1} B_K$$

$$C_K \rightarrow B_{K+1} C_{K+1}$$

we only want these type of production

$$\begin{aligned} P &\rightarrow AB \\ P &\rightarrow a \end{aligned}$$

Q Convert the following grammar into CNF

$$S \rightarrow aSa / bSb / a / b / aa / bb$$

Sol^m $\{ S \rightarrow a, S \rightarrow b \}$ already in chomsky normal form

① Arrange that all bodies of length 2 or more consist only of variables

$$\begin{cases} A \rightarrow A \\ B \rightarrow B \end{cases} \rightarrow \text{we change this}$$

$$S \rightarrow ASA / BSB / a / b / AA / BB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

Now

$$\begin{cases} S \rightarrow AA \\ S \rightarrow BB \end{cases} \rightarrow \text{in chomsky normal form}$$

(2) Break the bodies of length 3 or more into cascade of production each with a body containing 2 variable

$$S \rightarrow AX / BY / a/b / AA / BB$$

If length of body is n so time complexity

$$A \rightarrow a$$

$$B \rightarrow b$$

$$X \rightarrow SA$$

$$Y \rightarrow SB$$

Example CFG to CNF Simplification of CFG

~~Step 1~~

$$S \rightarrow ASA / aB$$

$$A \rightarrow B / S$$

$$B \rightarrow b / \epsilon$$

$$S' \rightarrow S$$

new production

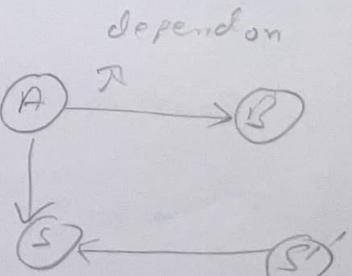
Step 1 Remove null production

$$S' \rightarrow S$$

$$S \rightarrow ASA / a / aB / AS / SA / S$$

$$A \rightarrow B / S$$

$$B \rightarrow b$$



Step 2 : Remove unit production :-

$$S' \rightarrow S$$

$$S \rightarrow S$$

$$A \rightarrow B$$

$$A \rightarrow S$$

are unit production

Rules without unit production

$$S \rightarrow ASA / a / aB / AS / SA$$

$$B \rightarrow b$$

New rules



$s' = ASA/a/aB/AS/SA$ $A \rightarrow ASA/a/aB/AS/SA/b$

final rules

 $s' \rightarrow ASA/a/aB/AS/SA$ $s \rightarrow ASA/a/aB/AS/SA$ $A \rightarrow ASA/a/aB/AS/SA/b$ $B \rightarrow b$

Step 3:- find the production that has more than 2 Non-Terminalin

RHS

 $s' \rightarrow ASA$ $s \rightarrow ASA$ $A \rightarrow ASA$ $s' \rightarrow AX/a/aB/AS/SA$ $s \rightarrow AX/a/aB/AS/SA$ $A \rightarrow AX/a/aB/AS/SA/b$ $B \rightarrow b$ $X \rightarrow SA$

Step 4:- we allowed only these types of production

 $P \rightarrow AB$

or

 $P \rightarrow a$

then we have to replace

 $s' \rightarrow aB$ $s \rightarrow aB$ $A \rightarrow aB$ $s' \rightarrow AX/a/YB/AS/SA$ $s \rightarrow AX/a/YB/AS/SA$ $A \rightarrow AX/a/YB/AS/b$ $B \rightarrow b$ $Y \rightarrow a$ $X \rightarrow SA$

Create Chomsky normal form ((CNF))

$E \rightarrow E + T \mid T * F \mid (E) \mid a \mid b \mid I_a \mid I_b \mid I_o \mid I,$

$T \rightarrow T * F \mid (E) \mid a \mid b \mid I_a \mid I_b \mid I_o \mid I,$

$F \rightarrow (E) \mid a \mid b \mid I_a \mid I_b \mid I_o \mid I,$

$I \rightarrow a \mid b \mid I_a \mid I_b \mid I_o \mid I,$

SOLⁿ

$P \rightarrow +$

$M \rightarrow *$

$L \rightarrow ($

$R \rightarrow)$

$A \rightarrow a$

$B \rightarrow b$

$Z \rightarrow o$

$N \rightarrow I$

$E \rightarrow EPT \mid TMF \mid LER \mid a \mid b \mid IA \mid IB \mid Iz \mid IN$

$T \rightarrow TMF \mid LER \mid a \mid b \mid IA \mid IB \mid Iz \mid IN$

$F \rightarrow LEF \mid a \mid b \mid IA \mid IB \mid Iz \mid IN$

$I \rightarrow a \mid b \mid IA \mid IB \mid Iz \mid IN$

We have to convert

$E \rightarrow EPT, E \rightarrow TMF, E \rightarrow LER, T \rightarrow TMF, T \rightarrow LER$

$F \rightarrow LER$

$E \rightarrow EP, TM, LE, | a | b | IA | IB | Iz | IN$

$T \rightarrow TM, LE, | a | b | IA | IB | Iz | IN$

$F \rightarrow LE, | a | b | IA | IB | Iz | IN$

$I \rightarrow a \mid b \mid IA \mid IB \mid Iz \mid IN$

$P_1 \rightarrow PT$

$M_1 \rightarrow MF$

$E_1 \rightarrow LER$

$P \rightarrow +$

$m \rightarrow *$

$L \rightarrow)$

$A \rightarrow a$

$B \rightarrow b$

$Z \rightarrow 0$

$N \rightarrow 1$

Examp 1: Consider the CFG

$S \rightarrow aaaaS / aaaa$ which generates

a^{4n} for $n = 1, 2, 3, \dots$

$= \{a^4, a^8, a^{12}, \dots\}$

Convert the above CFG to CNF

① $S \rightarrow aaaaS / aaaa$

② $S \rightarrow AAAAS / AAAAA$

$A \rightarrow a$

$S \rightarrow AA_1 / AAAA$

$A_1 \rightarrow AAAS$

$S \rightarrow AA_1 / AA_3$

$A_1 \rightarrow AA_2$

$A_2 \rightarrow AS$

$A_3 \rightarrow AAA$

$A_4 \rightarrow AA$

$S \rightarrow AA_1 / AAAA$

$A_1 \rightarrow AA_2$

$A_2 \rightarrow AS$

Examp 3: $S \rightarrow Na$

$N \rightarrow a/b$

$S \rightarrow NA$

$N \rightarrow a/b$

$A \rightarrow a$

} valid

$S \rightarrow NN$

$N \rightarrow a/b$

Not valid

Find whether the string exist in CFG

String :- baaba

start from 1

$$CFG = S \rightarrow AB/BC$$

$$A \rightarrow BA/a$$

$$B \rightarrow CC/b$$

$$C \rightarrow AB/a$$

CYK Algorithm

If here we got \emptyset so
string is not exist

S	X _{1S}
4	X ₁₄ X ₂₅
3	X ₁₃ X ₂₄ X ₃₅
2	X ₁₂ X ₂₃ X ₃₄ X ₄₅
1	X ₁₁ X ₂₂ X ₃₃ X ₄₄ X ₅₅

S	{S, A, S, C}
4	{B} {A, S}
3	\emptyset {B} {B}
2	{S, A} {B} {S, C} {S, A}
1	{B} {A, C} {A, C} {B} {A, C}

Substring

$$\begin{aligned}x_{12} &= \{B\} \cdot \{A, C\} \\&= \{BA\} \cup \{BC\} \Rightarrow \{S, A\}\end{aligned}$$

$$x_{23} = a_4$$

$$\{A\} \cdot \{C\}$$

$$\{S, A\} \cup \{A, C\} \cup \{C, A\} \cup \{C, C\}$$

$$= \{B\}$$

$$x_{34} = ab$$

$$\{A, C\} \cdot \{B\}$$

$$\{S, A\} \cdot \{B\} \cup \{B, C\}$$

$$\{S, C\}$$

$$x_{24} = aab$$

$$a/ab$$

$$aa/b$$

$$a = \{A, C\}$$

$$ab = \{S, C\}$$

$$aa = \{B\}$$

$$b = \{B\}$$

$$\{A, C\} \cup \{S, C\} = \{AS, AC, CS, CC\}$$
$$\downarrow$$
$$\{B\} \cup \{B\} = BB$$
$$B$$

$$x_{24} = B$$

$$x_{45} = ba$$

$$\{S, A\}$$

$$x_{13} = baa$$

$$b/aa \rightarrow \{B\} \cup \{B\} = \{B\}$$

$$ba/aa \rightarrow \{S, A\} - \{A, C\} = \{SA\} \cup \{SC\} \cup \{AA\} \cup \{AC\} \cup \{CA\}$$
$$B$$

$\rightarrow \phi$
 $\nearrow \text{divide}$

$$x_{35} = aba = a/ba = \{A, C\} \cup \{S, A\}$$
$$= AS, AA, CS, CA$$

$$ab/a \nearrow \text{divide}$$
$$= \{S, C\} \cup \{A, C\}$$

$$= SA, SC, CA, CC = \underbrace{\{B\}}$$

$$x_{14} = baab$$

$$ba/a/b = \{\phi\} \cup \{B\} = \{B\}$$

$$ba/ab = \{S, A\} \cup \{S, C\} = SS, SC, AS, AC = \{\phi\}$$

$$b/aa/b = \{B\} \cup \{B\} = \{B\}$$
$$= \{B\}$$

$$x_{25} = aaba$$

$$a/aaba = \{A, C\} \cup \{B\} = \{AB\}, \{C\} = S$$

$$aa/aaba = \{B\} \cup S, A^3 = BS, BA = A$$

$$aab/a = \{B\} \cdot \{A, C\} = BA, AC = A, S$$

$$\{A, S\}$$

$$x_{15} = baaba$$

$$b/baaba = \{B\} \cup \{A, S\}$$

$$\{BA\} \cup \{BS\}$$

$$\{A\}$$

$$ba/baba = \{S, A\} \cup \{B\}$$

$$\{SA\} \cup \{AB\}$$

$$\{S, C\}$$

$$baa/ba = \emptyset \cup \{S, A\}$$

$$\{S, A\}$$

$$baab/ba = \{B\} \cup \{A, C\}$$

$$\{BA\} \cup \{BC\}$$

$$\{A, S\}$$

$$= \{A, S, C\}$$

string is exist

PDA

PDA is used to define the context free grammar

PDA is an extension of a non deterministic finite automata with NULL edge

with the inclusion of push down stack

i.e PDA = NFA - ϵ + push down stack

NFA- ϵ FA - NFA they are on same level

NPDA have more power than NFA-FA - NFA- ϵ

Formal definition of PDA

gama \rightarrow those characters go to stack are part of gama

PDA is made up of 7 tuple ($Q, \Sigma, \Gamma, \delta, q_0, z_0, F$)

$Q \rightarrow$ finite set of states (like FA)

$\Sigma \rightarrow$ input symbol (like FA)

$\Gamma \rightarrow$ stack alphabet (Symbol pushed into stack)

$\delta \rightarrow$ Transition function

~~$S: Q \times (\Sigma \cup \epsilon) \times \Gamma \rightarrow Q \times \Gamma^*$~~

Deterministic PDA

$q_0 \rightarrow$ initial state

$z_0 \rightarrow$ stack start symbol it always on stack

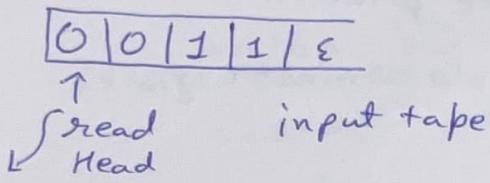
$F \rightarrow$ set of final states

$S \rightarrow$ for no deterministic PDA

$S: Q \times (\Sigma \cup \epsilon)^* \times \Gamma \rightarrow Q^* \times \Gamma^*$

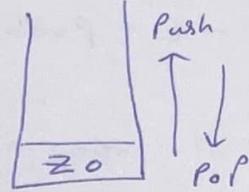
Q

$$\{0^n 1^n : n \geq 1\}$$



finite
control

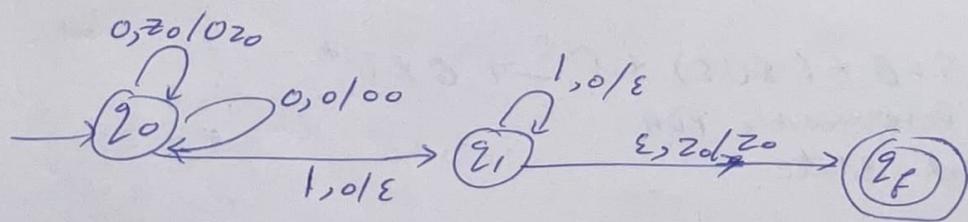
(it decide our stack grow →
or shrink)



Push 0's on to the stack pop symbols from the stack for each 1 being read

Notes:- Two version of PDA exist 1st that entering an accepting state(like FA do)

Another that accept by emptying its stack, regardless of the state it is in



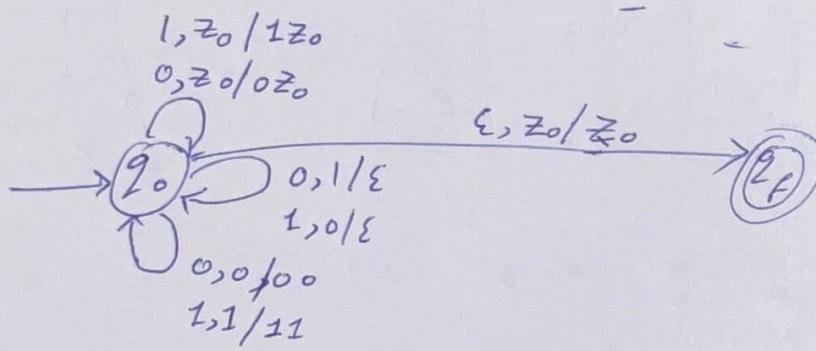
in initial state firstly we have to pop from the stack
than again push that on stack than push read head value
when we are not in initial state so pop from the stack
and we have Σ

$$\varnothing \quad L = \{ \omega \in \{0, 1\}^* \mid n_0(\omega) = n_1(\omega) \}$$

$$|\omega| > 1$$

$$\Rightarrow L = \{ 0_0 1, 1_0 0, 001, 010, 1_0 1_0, 1100, 0110, \dots \}$$

1	/	0	/	0	/	1	/	1	/	ε	/	...
---	---	---	---	---	---	---	---	---	---	---	---	-----



Linear way to represent

$$\delta(q_0, 0, z_0) = (q_0, 0z_0)$$

$$\delta(q_0, 1, z_0) = (q_0, 1z_0)$$

$$\delta(q_0, 0, 1) = (q_0, \epsilon)$$

$$\delta(q_0, 1, 0) = (q_0, \epsilon)$$

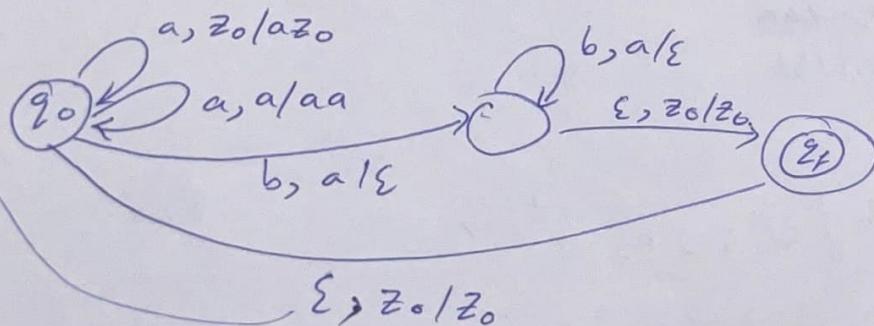
$$\delta(q_0, 0, 0) = (q_0, 00)$$

$$\delta(q_0, 1, 1) = (q_0, 11)$$

$$\delta(q_0, \epsilon, z_0) = (q_f, z_0)$$

If we want to accept NULL string

$$\varnothing \quad L = \{ a^n b^n \mid n \geq 0 \}$$



$$\delta(q_0, \epsilon, z_0) = (q_1, z_0)$$

Accept ϵ_0

$$\delta(q_0, \epsilon, z_0) = (q_0, \epsilon)$$

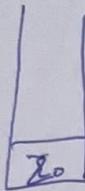
Accepted by empty stack (NULL string)

$\theta \{ w_c w^R \mid w \in \{a, b\}^+ \}$

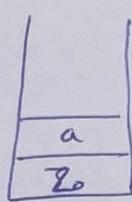
Guiding principle
split

a	b	b	c	b	b	a	ϵ
---	---	---	---	---	---	---	------------

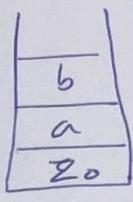
Step 1



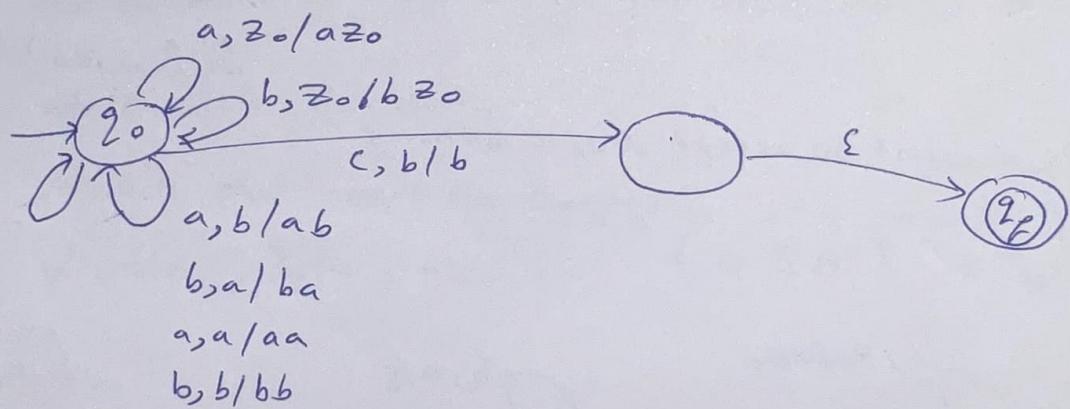
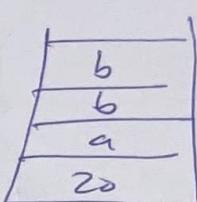
Step 2



Step 3



Step 4

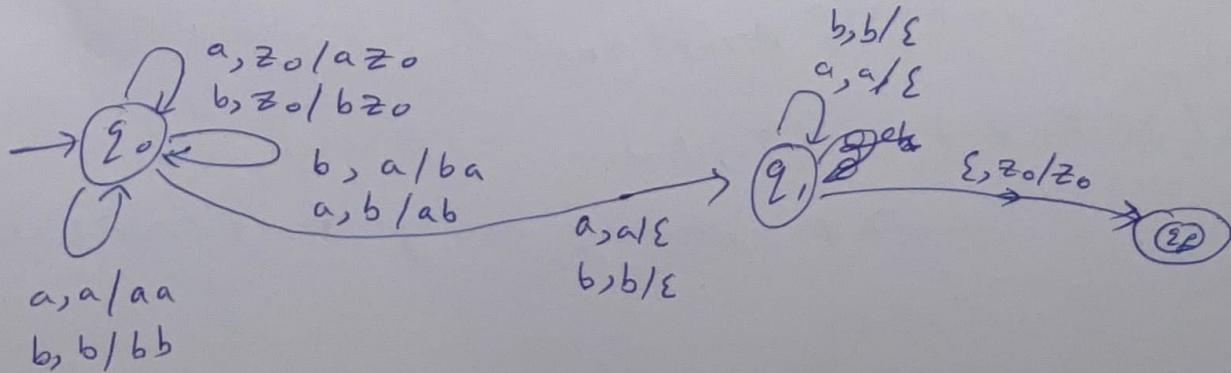


$\theta L = \{ ww^R \mid w \in \{a, b\}^+ \}$

a b a a b a

a b b b b a

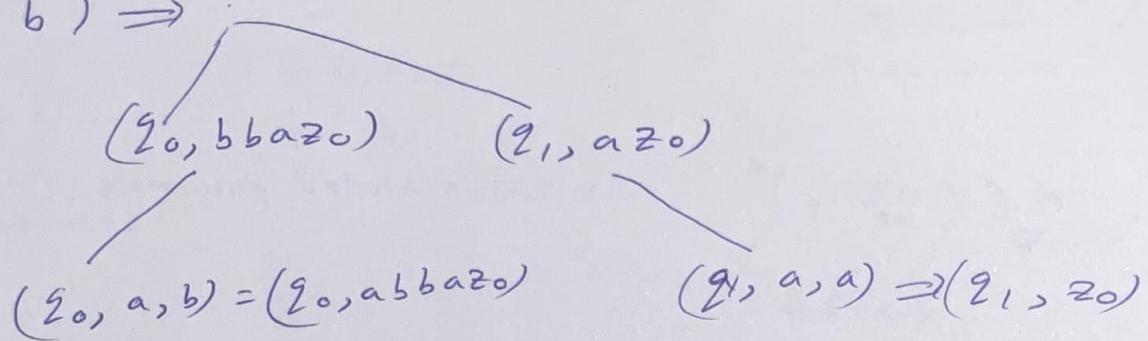
b a a a a b



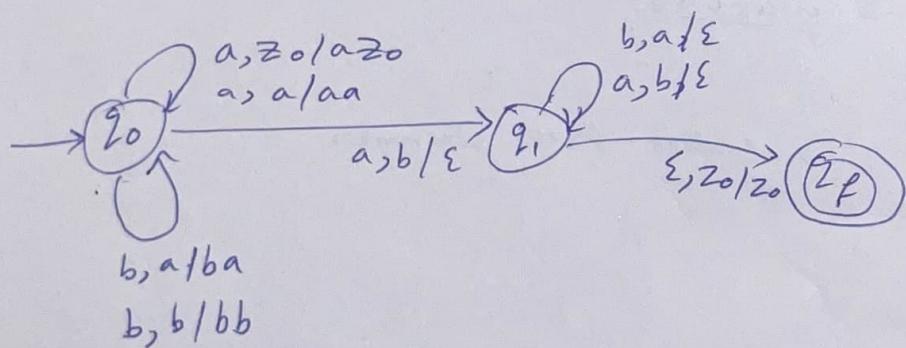
$(q_0, abba, z_0) \Rightarrow (q_0, az_0)$

$(q_0, bba, a) \Rightarrow (q_0, baza)$

$(q_0, ba, b) \Rightarrow$

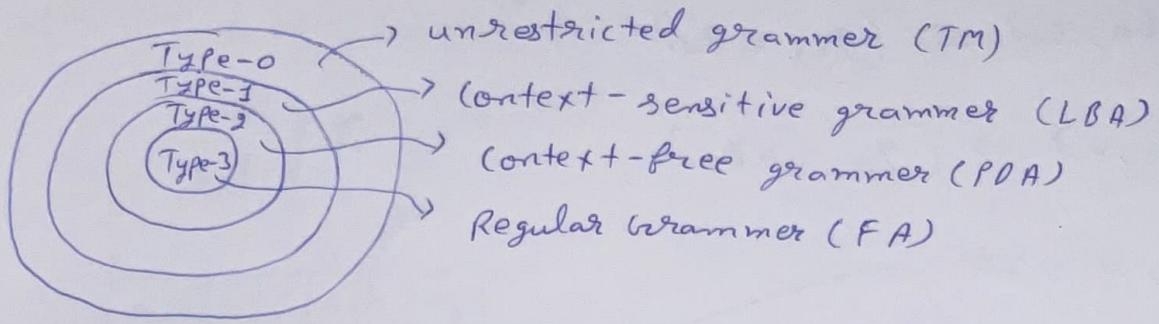


$\emptyset L = \{a^n b^m a^n b^m\}$, where ~~m, n are~~ and $n, m \geq 1$?
 m, n are independent



abbaab

$\emptyset L = 0^{n+2n}$, where n should be greater than 0 ?



1. $\alpha \in (\cup_{NT} V_U V_T)^*$ \rightarrow unrestricted grammar
 $\alpha \rightarrow \beta$

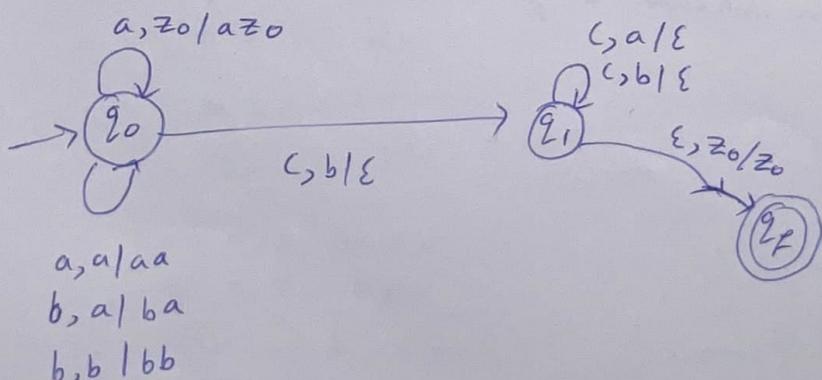
2. $|\alpha| \leq |\beta|$ \rightarrow context-sensitive grammar
 $\alpha \rightarrow \beta$

e.g.: $\underline{a} \underline{B} \underline{b} \rightarrow \underline{a} \underline{a} \underline{B} \underline{A} \underline{b}$

3. $|\alpha| \leq |\beta|$ \rightarrow context-free grammar
 $\alpha \rightarrow \beta$

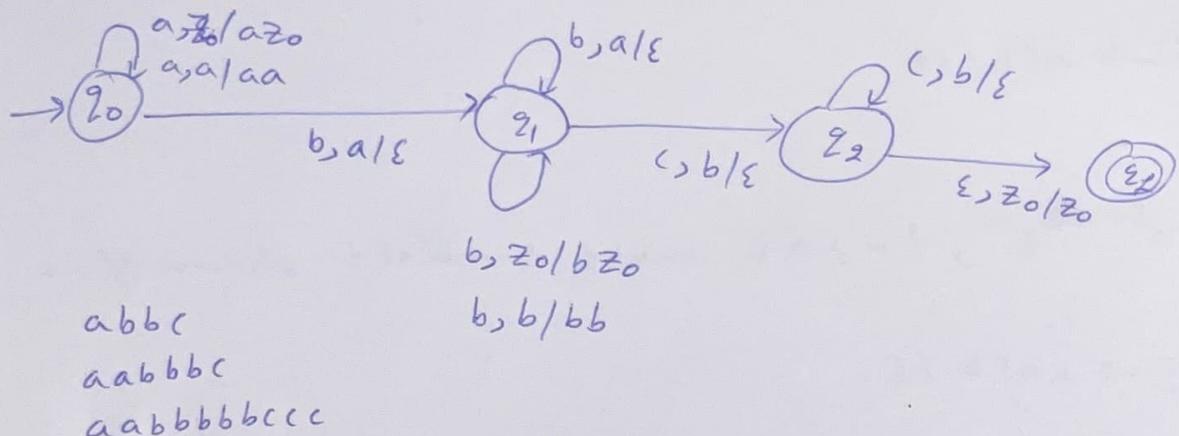
$A \rightarrow \beta\beta$

Q. $a^m b^m c^{(n+m)}$, $m, n \geq 1$ make PDA of it



$a b c c$
 $a a b b c c c c$
 $a a b c c c$
 ...

Q Make PDA of $a^n b^{(n+m)} c^m$, $n, m \geq 1$



Q $a^{n+m} b^n c^m$, $n, m \geq 1$ make PDA of it

Q $\{a^n b^n | n \geq 0\}$ make smallest grammar of it

$$S \rightarrow a S b / \epsilon$$

Q $\{a^n b^n | n \geq 1\}$ make smallest grammar of it

$$S \rightarrow a S b / ab$$

Q $\{a^m b^{n+2} \mid n \geq 0\}$, make smallest grammar of it non-regular language

$$S \rightarrow aSb/bb$$

Q $\{a^{2n} \cancel{b^n} \mid n \geq 0\}$, make smallest grammar of it

$$S \rightarrow aaaSb/\epsilon$$

Q $\{a^{2n+3} b^n \mid n \geq 0\}$, make smallest grammar of it

$$S \rightarrow aasb/aaa$$

Q $\{a^m b^n \mid m > n, n \geq 0\}$, make smallest non-regular language

$\{a, aa, aaa, aab, aaaa, \dots\}$

$$S \rightarrow AS_1$$

$$S_1 \rightarrow aS_2 b / \epsilon$$

$$A \rightarrow aA/a$$

Q $\{a^m b^n \mid m \geq n, n \geq 0\}$, make smallest non-regular language

$$S \rightarrow AS_1$$

$$S_1 \rightarrow aS_2 b / \epsilon$$

$$A \rightarrow aA/\epsilon$$

Q $\{w \mid n_a(w) = n_b(w)\}$, make smallest non-regular language

Pumping lemma for CFL

If it is used for proving the language is not context-free
 Assume that language is CFL then there exists a constant
 n such that the string z length ($|z| \geq n$) and
 z is broken into 5 parts $z = uvwxy$, subject to
 the conditions (1) $|vwx| \leq n$

(2) $|vx| \geq 1$ vx need to be pumped

(3) for $i \geq 0$

uv^iwx^iy must be in L

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

Assume L is context-free. There exist a constant n
 such that $z = a^n b^n c^n$ and $|z| = 3n > n$
 we break the z into five parts u, v, w, x and y

$$z = uvwxy$$

subject to the condition

(1) $|vwx| \leq n$

(2) $|vx| \geq 1$

(3) for $i \geq 0$, uv^iwx^iy must be in L

[w null bhi ho sakta h]

$$\frac{a a a b b b b}{u \bar{v} \bar{w} \bar{x}} \frac{c c c}{z}$$

$$\frac{a a a b b b b}{u \bar{v} \bar{w} \bar{x}} \frac{c c c}{z}$$

Case 1 :- $v = a^K$
 $x = b^K$

Case 2 :- $v = b^K$
 $x = c^K$
 $K = 2$

$$u = aa \cancel{bb}$$

$$\sqrt{K} = \cancel{bb} (a')^2$$

$$w^K = b$$

$$x^K = (b')^2$$

$$y = bccc$$

aaaabbbbccc

Here no of a's and b's
are greater than no of
c's

Since this is not belong to L so our assumption
is wrong and we reach to the contradiction
that the language is CFL Hence L is non CFL

Q L = { a^p : p is prime}

Assume L is context-free. There exist a constant n
such that $z = a^p$ and $|z| \geq n$
we break the z into five parts u, v, w, x and y

$$z = uvwxy$$

subject to the condition

$$\textcircled{1} |vwx| \leq n$$

$$\textcircled{2} |vwx| \geq 1$$

\textcircled{3} for $i \geq 0$, $uv^iwx^i y$ must be in L

$uv^iwx^i y$ must be in L

$|uwxy| = q \nearrow$ q is prime no (assume)
u, w and y composed of all a's)

$|vnx| = p \geq 1$ v and x composed of all a's)

pump i = q times $uv^q w x^q y$

$$= |uwxy| + q|vnx|$$

$$= q + qp$$

$$= q(1+p) \rightarrow \text{It is not a prime no}$$

so our Assumption is wrong and we got contradiction
so it is not a CFL